Self-Adaptive Network-on-Chip Interface

Rachid Dafali, Jean-Philippe Diguet, J.-C. Créput

Self-Adaptive Network-on-Chip Interface

Rachid Dafali, Jean-Philippe Diguet, *Member, IEEE,* and Jean-Charles Creput

***Abstract*—This paper presents an original approach of bandwidth-oriented self-adaptivity in the domain of Network-on-Chip, where reconfiguration is handled by network interfaces offering traffic with guarantee of service. Reconfiguration is first based on multiple FIFOs with variables bounds and implemented in a single dual-port memory with a dedicated controller. Secondly, it relies on multiple and compliant TDMA tables based on a new heuristic for path computation. Combination of both techniques provide significant bandwidth improvement with a negligible resource overhead. The proposed solution is demonstrated with cycle-accurate VHDL simulation and FPGA implementation for synthetic and image processing applications.**

***Index Terms*—Reconfigurable Network-on-Chip, dynamic reconfiguration, self-adaptivity, FIFO, TDMA, network-interfaces**

I. Introduction

NETWORK-on-Chip (NoC) have been introduced a decade ago [1], as an innovative approach to meet the expected bandwidth in modern System-on-Chip (SoC) that implement an increasing number of IPs (processors, distributed memories and peripherals). NoCs offer both performances and scalability properties expected in large SoC. Today, the NoC approach is completely adopted by industrial solutions.

In this paper, we explore the concept of Reconfigurable NoC (RNoC), which aims to introduce self-adaptive mechanisms in NoCs. Self-adaptivity is required for various reasons, we mainly consider the two following aspects. First, in distributed reconfigurable SoC, whatever the quality of service provided (best effort or guaranteed traffic), the conventional NoC are unable to handle constraints introduced by the load balancing mechanism. Secondly, in guaranteed traffic NoC, the design methodology is based on estimations of application traffic, but data transfers may be very variable especially in case of data-dependent applications (e.g. video codecs, image processing, networking), which derive from energy-conscious optimizations. Moreover the activation period of applications may change over time and consequently result in various configurations, which can be difficult to predict and model.

We can address these issues by considering the worst traffic pattern in the NoC design flow and decide NoC topology and parameters accordingly. Such a solution is safe and guarantees real-time constraints but results in a significant oversizing. Another possibility consists in accepting performance degradation in a proportion to be defined. This solution can be controlled if communications are prioritized but offers no guarantee. So, a relevant problem is the design of an adaptive and reconfigurable NoC, which guarantees traffic and introduces the minimum overhead in terms of size and power consumption.

The analysis of real issues shows that the network access time has, in practice, a major impact on the overall throughput. So we focus our work on Network Interface (NI), which also offer relevant and low-overhead adaptation opportunities. We propose two original and efficient mechanisms to introduce self-reconfiguration within NIs. The first one relies on dynamically reconfigurable memory buffers that allow for the runtime adaptation of FIFO depths with a single shared memory according to communication needs. The second one is complementary and controls the Time Division Multiple Access (TDMA) scheduler, which is dynamically reconfigurable. It can adapt the number of time slots allocated to different communication paths according to real bandwidth needs while preserving guaranteed traffic property. Another contribution is the architecture of an optimized Self-Adaptive NI (SANI) that supports these two mechanisms and ensures the necessary decoupling between the IP and the network.

The objective of the paper is to demonstrate the impact of the local adaptation, it is done with a reference NI where new mechanisms are introduced for fair comparison.

II. Related work

The use of RNoCs aims to optimize or to modify the management of communications by meeting unpredictable needs and events. These unpredictable requirements vary from a simple increase in data bandwidth, to the connection of a new IP. Thus, the first criteria, which characterizes a RNoC, is the set of services it can satisfy in real-time. The study of these services allowed us to build a first classification of RNoCs, we can distinguish three types. i) *Expanding* (E) RNoCs allow the connection of new IPs to the network by adapting the physical topology. ii) *Multimode* (M) RNoCs adapt the communications and their constraints according to the application (mode) performed. The multimode approach aims to avoid the network oversizing by sharing its resources. However, it requires an upstream computation of different RNoC configurations. Also, the transition from one configuration to another requires suspending transmissions and a full initialization of the network. iii) *Optimization* (O) RNoCs allow efficient use of network resources to increase performance. The reconfiguration mechanisms introduced by RNoC, combined with efficient configuration decisions, is used to continuously track network optimality. Another aspect is where the reconfiguration occurs, it can be at the architecture (e.g. router, topology, DVFS) or protocol level (e.g. packet vs circuit switching). Finally the important point is how the reconfiguration decision is made and implemented, it can be centralized (C) or distributed (D). With respect to this classification, we summarize the state of the art in table I.

J-Ph. Diguet and R. Dafali are with Lab-STICC UMR 6285, CNRS and Université de Bretagne Sud, 56100 Lorient, France (e-mail: rdafali56@gmail.com, jean-philippe.diguet@univ-ubs.fr).

J.C. Creput is with IRTES-SET, Université. de Technologie. de Belfort-Montbliard, 90000 Belfort, France (jean-charles.creput@utbm.fr).

Reference	Type	Computation	Decision	Execution	Architecure Protocol
Nollet [5]	O	online	C	Stop	BE TDMA slots
Pionteck [2]	E	online	C	Stop	(DPR) Routers
Stensgaard [3]	M	offline	C	Reset Stop	Topology, policy, packet/CC switching
Nicopoulos [4]	O	online	D	Runtime	BE Buffers, VC
Marescaux [7]	O	online	C	Reset	BE TDMA slots
Hansson [8]	M	offline	C	Reset Stop	paths reorganization
Huebner [9]	M	offline	C	Reset Stop	BE, multipaths Crossbar Switches
Bogdan [6]	O	offline online	C	Runtime	DVFS
This work (global/local)	M O	offline online	C D	Runtime Runtime	GT TDMA, NI, FIFOs

TABLE I
RNoC COMPARISON: TYPE, METHODOLOGY AND MAIN FEATURES.

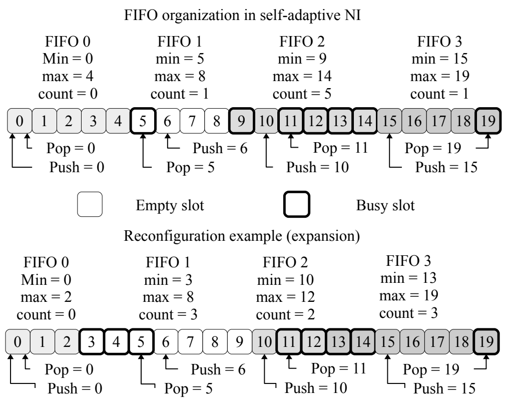
FIFO organization in self-adaptive NI
FIFO 0 Min = 0 max = 4 count = 0
FIFO 1 min = 5 max = 8 count = 1
FIFO 2 min = 9 max = 14 count = 5
FIFO 3 min = 15 max = 19 count = 1
Pop = 0 Push = 0
Push = 6 Pop = 5
Pop = 11 Push = 10
Pop = 19 Push = 15
Empty slot
Busy slot
Reconfiguration example (expansion)
FIFO 0 Min = 0 max = 2 count = 0
FIFO 1 min = 3 max = 8 count = 3
FIFO 2 min = 10 max = 12 count = 2
FIFO 3 min = 13 max = 19 count = 3

Fig. 1. Reconfigurable FIFOs principle

Regarding architecture reconfiguration, CoNoChi [2] is one of the most complete solution that enables topology expansion, meaning that it is designed to adapt the architecture by adding or removing routers. This solution is limited to topology adaptation and dedicated to network configuration on FPGA with dynamic partial reconfiguration (DPR) capabilities. In [3] authors combine architecture and protocol reconfigurations. A router can be reconfigured according to logical topologies, routing policies and protocols that can be either packet or circuit switching but at the cost of significant extra resources. Another distinguishable work is ViChaR [4], where buffer can be dynamically allocated to different virtual channels (VC), this solution fits best-effort type (BE) traffics and the cost of VC combined with memory management lead to costly routers implementations. Online reconfiguration strategy is rarely explored, except in [5], where a new OS service is proposed to adapt access time windows in a BE context. The decision is based on collected statistics that require a second monitoring network and to stop execution during reconfiguration. Finally we don't consider dynamic voltage and frequency scaling (DVFS), which is complementary to our work, however we might refer it as a dimension RNoC. A remarkable method is proposed in [6], this is an online DVFS approach applied on a multi region SOC that relies on a linear controller according to a fractal modeling of traffic.

NI are rarely considered in RNoC, however we believe that this is where major impacts on NoC performances can be obtained in practice with a low resource overhead. First, in multiprocessor architectures delays between the execution of read/write instructions on processors and the first header emission can be much larger than the number of cycles required for inter-router hops, it is so worth working on the NoC access latency. Secondly, NoC can offer impressive bandwidths but sequential IP/NI communications are real bottlenecks. Finally, hardware reconfiguration of routers is costly, whereas NIs can be simply configurable. In our project we consider TDMA as a solution to guarantee and control minimum bandwidths. The two reconfiguration techniques we propose are based on this choice and are handled by a local manager (LM). They also lie on our NoC design framework that includes topology selection and memory sizing, it also comes after offline profiling steps that allow to identify typical scenarios that will be used to specify TDMA tables. In the next sections, we first describe these techniques and then the LM implementation.

III. RECONFIGURABLE FIFO

FIFOs are required in NIs, to store data according to destinations and sources, FIFOs also represent a major part of NI area. In most cases a NI is connected to a processor or memory bus and to a router port, it means that in the worst case one read and one write operations can occur at the same time. It also means that a double port memory is enough, multiple in and output ports are not necessary. So we propose to implement multiple FIFOs within a standard shared RAM, so that costly multiplexers, to control accesses to independent FIFOs, are removed. This implementation also allows us to dynamically manage the size of each FIFO by means of push and pop pointers that can be moved within boundaries adapted to traffic requirements. The principle of FIFOs self adaptation is given in Fig.1. Push and pop pointers as well as the adaptation management is implemented in hardware. Based on rules checking such as the availability of free slots in adjacent FIFOs and the boundary constraints, FIFO can be expanded every cycle without performance penalty.

IV. RECONFIGURABLE TDMA

a) Architecture: TDMA is a well known solution to guarantee traffic by means of joint path computation and time slots allocation. A TDMA table is simply implemented as a circular buffer that contains FIFO IDs. At every cycle, each NI writes in the output FIFO corresponding to the destination the time slot is allocated to. The new approach we propose is to compute multiple compliant TDMA tables so that one NI can switch locally from a table to another without introducing conflicts in the NoC, whatever are the local choices of the other NIs. The configuration process doesn't introduce any performance penalty since in practice the choice of the next FIFO ID can be read from all available TDMA tables according to a switch controlled by the LM. TDMA slot allocations and paths are jointly computed offline with the method described hereafter.

b) Path computation: The requirement of traffic guarantee and dynamic reconfiguration is modeled as a combinatorial optimization problem. The problem is an extension and combination of a classical k-shortest paths problem, with a bin packing problem. It allows the sharing of non-conflicting time-slots between reconfigurable communication paths. The

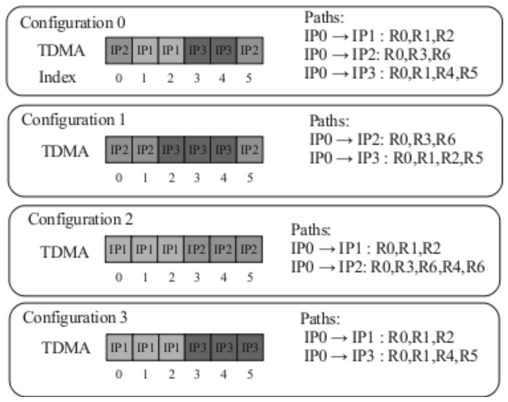

Fig. 2. Reconfigurable TDMA principle

problem is new, and it is strongly NP-hard since bin packing is. Accordingly, we found that only heuristic and metaheuristic methods were able to generate admissible solutions, for reasonable size instances, within reasonable computation time. To solve the problem, we adopt a time-expended graph representation to memorize temporal channel occupation and represent necessary and sufficient conditions for the reuse of time-slots. An important requirement is that adaptation of TDMA tables must occur in real-time with no re-initialization. Based on this structure, a local search procedure and a genetic algorithm were developed. They both operate by using a set of low-level solution construction operators. They can be an emission dates management, a greedy parallel paths construction and a single-path optimal Dijkstra procedure, or a neighborhood improvement operator. The local search is able to quickly find a solution of good quality. The population based genetic algorithm is able to diversify solutions when addressing constrained problems having many reconfigurable TDMAs. We found out that a cost function equal to the ”cumulated length of paths” was a good choice to introduce solution diversity while minimizing communication latencies. The algorithms were extensively tested on problem instances of different sizes. Computation time to build an admissible solution is about few milliseconds for a moderate size application with a dozen of IPs and about 30 communication paths. It increases to about 7 seconds for a larger size problem with 50 or so IPs and about 200 paths. A moderate size problem with reconfigurable TDMAs, where 3 IPs among 10 have 2 TDMAs each, can be solved within about 10 seconds. Our method can also be used to derive partially new solutions from existing ones.

V. Configuration decision

A Local Manager (LM) is in charge of fast and local configuration decisions. The LM monitoring sensors are mainly the FIFO status, given FIFO full or empty signals, it authorizes or not the extension of FIFOs over their neighbors and the switch to an alternative TDMA table. Different policies may be implemented as a set of simple rules. In the current version we consider that local decisions must be instinctive and simple, and the implementations are designed accordingly. Thus, FIFOs extensions are always allowed if slots are available regarding the position of Pop and Push pointers of neighbors. Regarding TDMA tables, reconfiguration is considered when full or empty FIFO signals are raised. For instance in Fig.2 priority is given to configuration 0. The LM decides a configuration from table 0 to table 1 if and only if IP1 FIFO is empty and if IP2 or IP3 FIFO is full. But the LM is back to configuration 0 when the IP1 FIFO isn’t empty anymore. Out of the scope of this paper we also we also consider a low rate global manager (GM) that can for instance decide the set of TDMA tables to be instantiated in the different SANI. Contrary to the LM the GM is a software task.

VI. Architecture

The protocol adapter (wrapper) and the SANI architectures are detailed in Fig.3, they have been designed to decouple IP and NoC routers, which may have different clocks and variable communication rates. The wrapper is connected to the IP bus and can use a DMA, if it is authorized, to write and read data in a shared memory without involving the local processor if any. In emission the IP specifies the local address in a shared memory, the amount of data and the destination ID. Then the DMA transfers the blocks of data from the memory to the wrapper cache, the size of blocks is equal to the cache size. In reception the controller sends a request to the IP (Interrupt), which can authorize a transfer to the wrapper cache if it is empty and if no concurrent transfers are in progress, finally the DMA completes the transfer from the wrapper cache to the shared memory. The SANI itself is composed of 7 components. The input port is connected to the wrapper and transfers data to the FIFOs controller according to a destination mapping table. The output port transfers received packets, stored in FIFOs, to the wrapper if a read order is received. The network input ports receives packets from the network and transmits data and the sender ID to the reception FIFO controller. The network output port manages packet injection according to the TDMA-based scheduling, this output port has been modified in order to manage switches between multiple TDMAs without loss of cycle. The controllers of emission and reception FIFOs manage read and write pointers of the different FIFO in a single dual port memory, they implement two MIN and MAX vectors that store FIFO bounds and counters that give the use ratio. Finally the LM implements rules to decide FIFO and TDMA configuration. The VHDL code of the SANI is automatically generated such that resources are minimized according to specifications choices. Our prototypes are designed on Xilinx FPGA, the FIFO are implemented in a distributed (LUT) dual port and dual clock RAM. BRAM could be used if bigger memories (e.g. 18Kb) are required but would mean a larger granularity and impose another resource constraint optimization problem.

VII. Results

a) Object tracking, VHDL CABA simulation: The first experiment is based on the real traffic generated by an object tracking application. The application is organized in five tasks (T_i) running on dedicated hardware modules, the following percentages represent the ratio of the total execution time without parallelism. T1 (15%) computes in a row, for each pixel, the mean value from the four previous frames, the difference with the background frame and the binarization. T2 (2%) executes subsequently two classical image processing

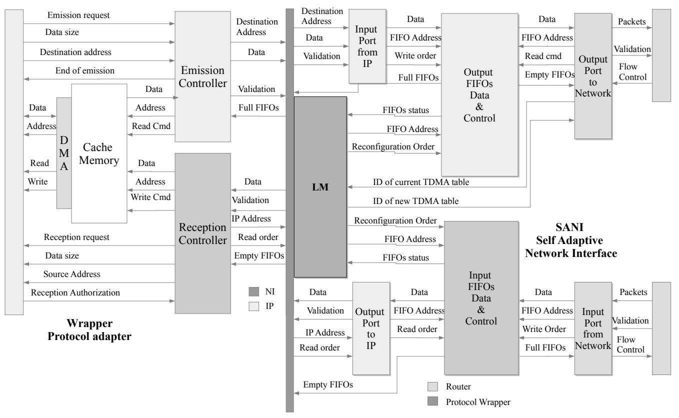

Fig. 3. SANI architecture

functions (erosion and dilation), while T3 (31%) achieves object extraction by means of a reconstruction algorithm. Then, T4 (24%) labels objects and T5 (28%) draws rectangles around labelled objects. A pipeline implementation is adopted to maximize the data rate that reaches 80 fps. Nine dedicated processors are considered, P1 and P2 execute T1 as well as acquisition and display tasks, P3, P4, P5 run multiple instances of T2 and T3 and P6, P7, P8 and P9 execute multiple instances of T4 and T5. For simulation purpose, processors are replaced by traffic generators, which reproduce data transfers according to tasks dependencies. An ad hoc NoC topology has been obtained with our methodology and unused ports are removed. It results in 9 routers and 36 links (32 bits). 2D mesh NoC and spidergon topologies would require {9,42} and {10,52} routers and links respectively. The simulation is cycle and bit accurate. With this implementation a 80 fps data rate is achieved. In a second step, the NI are replaced by SANI implementing both FIFO and TDMA self-reconfiguration mechanisms. The size of a SANI is 3.6% larger than the size of a NI, however the multi-TDMA path computation results in solution with less links and the RNOC cost is finally equivalent (-0.8%). The main important achievement is the data-rate improvement since the RNOC provides 97 fps (+21.25%) with a better use of link bandwidths.

b) FPGA implementation, Inter Microblaze communications: The second experiment relies on a complete implementation of 4 Xilinx microblazes that communicate through a NoC composed of 7 routers. The objective is first to validate software API and network interfaces and secondly to evaluate the impact of FIFO and TDMA self-reconfigurations. Random data are transferred, but the communications of each processor respect the communication dependency graph (CDG) of a video coder. Communication schemes are implemented as infinite loops so that all the available bandwidth is used. Processor 0 is the only one to communicate with all other processors. In this experiment, we observe the bandwidth use for four different cases: non reconfigurable NI, NI+reconfigurable FIFO, NI+reconfigurable TDMA, NI+reconfigurable FIFO and TDMA. Bandwidth as well as area results are given in Table II for communications from P_0 to other Processors (P_0, P_1, P_2). We observe that use of reconfigurable FIFOs depends on buffer extensions opportunities. These opportunities are frequent for

	Bandwidth (Mbits/s)			
	Static	Reconfigurable FIFOs	Reconfigurable TDMA	Reconfigurable. FIFOs & TDMA
Com P_0/P_1	287	$304\ (5,9\%)$	$354\ (23,5\%)$	$421,9\ (46,7\%)$
Com P_0/P_2	289	$282\ (-2,5\%)$	$367\ (27\%)$	$382,5\ (32,3\%)$
Com P_0/P_3	290	$322\ (10,9\%)$	$362\ (24,9\%)$	$358,9\ (23,8\%)$
	Total Area			
Slices	2190	2212 (1%)	2082 (-4,9%)	2017 (-7,9%)
LUTs	3942	4060 (3%)	3982 (1%)	4101 (4%)
Total Bandwidth	866	+4,8%	+25,1%	+34,2%
Bandwidth/LUT	0,219	+2,1%	+24,3%	+29,4%

TABLE II
BANDWIDTH AND RESOURCE COST FOR 4 VERSIONS OF NI/SANI$_0$

P_1 and P_3 FIFOs but not for the P_2 FIFO. On the other hand, TDMA reconfiguration can systematically take advantage of available bandwidth. It is also due to the strategy of the LM, which allocates at least the minimum FIFO size according to the number of TDMA slots. Finally the best solution is the combination of the two techniques, which allows to maximize the opportunities to improve bandwidth. In this case study, the total bandwidth is improved of 34%.

VIII. CONCLUSION & PERSPECTIVES

This paper presents a simple and efficient implementation of distributed self-adaptivity in a NoC, which is fully implemented in network interfaces. Our results show that the combination of reconfigurable FIFOs and compliant TDMAs can be efficiently used with a local manager to achieve a better use of available bandwidth and memory resource with a very limited resource overhead. FIFO adaptation takes advantage of extension opportunities that are application dependent, TDMA table selection amplifies the extension effects with the adaptation of allocated slots. Two immediate perspectives can be proposed to overcome current limitations. The order of FIFO can be optimized to improve FIFO extension opportunities and new TDMA tables can be additionally computed online by a global manager with a slower adaptation rate. The proposed solution is scalable, the cost of the SANI is mainly related to the number of FIFO for each processor interface, if necessary this cost can be controlled with dynamic FIFO allocation within the SANI that also offers this possibility.

REFERENCES

[1] P. Guerrier and A. Greiner, “A generic architecture for on-chip packet-switched interconnections,” in *DATE*, 2000.
[2] T. Pionteck, R. Koch, and C. Albrecht, “Applying partial reconfiguration to networks-on-chips,” in *FPL*, 2006.
[3] M. Stensgaard and J. Sparso, “ReNoC: A network-on-chip architecture with reconfigurable topology,” in *NoCS*, 2008.
[4] C. Nicopoulos et al., “Vichar: A dynamic virtual channel regulator for network-on-chip routers,” in *39th MICRO*, 2006.
[5] V. Nollet, T. Marescaux, and D. Verkest, “Operating-system controlled network on chip,” in *41st DAC*, 2004.
[6] P.Bogdan et al., “An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads,” in *NoCS*, 2012.
[7] T. Marescaux et al., “Dynamic time-slot allocation for qos enabled networks on chip,” in *ESTIMedia*, 2005.
[8] A. Hansson et al., “Undisrupted quality-of-service during reconfiguration of multiple applications in networks on chip,” in DATE, 2007.
[9] M. Huebner et al., “Scalable application-dependent network on chip adaptivity for dynamical reconfigurable real-time systems,” in *FPL*, 2004.