Towards a generic platform for the distribution of avionics applications on manycores

Ghina Abdallah, Jérôme Ermont, Sandrine Mouysset, Jean-Luc Scharbarg

Towards a generic platform for the distribution of avionics applications on manycores

Ghina Abdallah, Jérôme Ermont, Sandrine Mouysset, Jean-Luc Scharbarg
IRIT - Université de Toulouse
2 rue Charles Camichel
31000 Toulouse, France
{firstname.lastname}@irit.fr

***Abstract*—The interconnection of many-cores by an avionics full duplex switched Ethernet network (AFDX) is envisioned for future avionics architecture. The principle is to distribute avionics functions on these many-cores. Many-cores are based on simple cores interconnected by a Network-on-Chip (NoC). The allocation of functions on the available cores as well as the transmission of flows on the NoC has to be performed in such a way that avionics timing constraints are never violated. Several theoretical solutions have been proposed for this distribution. However they have not been evaluated on real architectures. In this paper we introduce a framework for the prototyping of such implementations. This framework is based on the existing ProNoC tool which allows the configuration of an FPGA as a NoC. The goal is to be able to compare distribution solutions with different NoC features in terms of scheduling or routing.**

***Index Terms*—Many-cores, NoC, task distribution, avionics**

I. Application domain and challenge

Aircrafts include numerous electronic equipments. Some of them, like flight control and guidance systems, provide flight critical functions, while others may provide assistance services that are not critical to maintain airworthiness. Current avionics architecture is based on the integration of numerous functions with different criticality levels into single computing systems (mono-core processors) [1]. These computing systems are interconnected by an AFDX (Avionics Full Duplex Switched Ethernet) [2]. As depicted in the upper part in Figure 1, the End System (ES) provides an interface between a processing unit and the network.

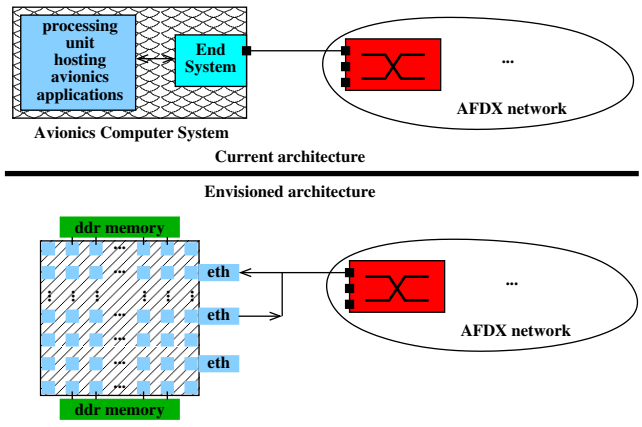

Fig. 1. An AFDX network.

Mono-core architectures are being replaced by multi- or many-core ones in many contexts. This move is also envisioned in aircrafts. However, multi-core architectures are based on complex hardware mechanisms whose temporal behavior is difficult to master. Conversely many-core architectures are based on simpler cores interconnected by a Network-on-Chip (NoC). These cores are more predictable [3]. Thus, many-cores are promising candidates for avionics architecture. Such an architecture integrating many-cores is illustrated in the lower part in Figure 1. A typical many-cores architecture provides Ethernet interfaces which are used for the connection with the AFDX network. Additionaly, memory controllers manage access to DDR. An an example, Tilera Tile64 has 3 Ethernet interfaces and 4 memory controllers [4].

The envisioned avionics architecture depicted in the lower part in Figure 1 is a mixed NoC/AFDX architecture. Avionics functions are distributed on the available many-cores. Communications between two functions allocated on the same many-cores (local functions) use the NoC, while the communications between two functions allocated on different many-cores (remote functions) use both the NoC and the AFDX. Main constraints on this communication are the following:

1) end-to-end transmission delay has to be upper-bounded by an application defined value,
2) frame jitter at the ingress of the AFDX network has to be smaller than a given value (typically 500 μs).

The first constraint concerns local and remote functions, while the second one only concerns remote functions. Transmission delays on the NoC have an impact on both constraints. These delays can vary for different reasons. First, a frame can be delayed by other frames crossing the same routers (router contentions). Second, in the case of a transmission between remote functions, the Ethernet controller can be busy, transmitting another frame (controller contention).

The mapping of functions on the many-cores has a major impact on this NoC delay variation. [5] proposes a mapping strategy that minimizes router contention. In this strategy, each core is allocated at most one function and each avionics flow is managed by its source function. [6] proposes a different strategy, based on a static scheduling of Ethernet transmissions: each transmission is assigned a periodic slot in a table. Thus there are no more controller contentions and router contentions

are reduced, thanks to the mapping of functions on cores.

II. Motivation

Both [5] and [6] consider Tilera Tile64 many-cores [4]. However delay computation are based on a model of the many-core and no implementation is provided.

Therefore the first motivation of this study is to map avionics functions on a hardware platform. One goal is to validate results in [5] and [6].

The second motivation of this study is to be able to tune many-core features, mainly NoC ones. Different many-cores implement different topologies, different buffer sizes in router ports, different scheduling algorithms in routers or different routing strategies. For instance, Kalray MPPA [7] has larger buffers than Tilera Tile64.

Therefore our goal is to map avionics functions on a generic hardware platform that can be configured, based on existing NoC features. As a first step, we consider a single many-core.

III. Problem statement

The problem is to distribute a set of n applications $A_0, \ldots, A_{n-1}$ on a many-core. Each application A_i is composed of n_i communicating tasks $t_{i,0}, \ldots, t_{i,n_i-1}$ Communication between tasks are modelled by a graph. The number of cores as well as the NoC topology, buffer size and scheduling algorithm in routers, routing are configurable. The distribution assumes a mapping strategy, e.g. SHiC [8], Map_{IO} [9], strategies proposed in [5] and [6] or an ad hoc one.

The resulting mapping is implemented on a hardware platform, typically an FPGA, e.g. a Nexis4 card and transmission delays are measured. These measured delays are then compared with theoretically computed values.

IV. Proposed approach and preliminary results

The proposed solution is based on a prototyping tool. In the next paragraphs we present its main features and a preliminary case study.

A. The prototyping tool ProNoC

ProNoC (Prototype NoC) has been defined in [10] and is a prototyping tool which allows the design of many-core system on chips (MCSoC). It proposes an interface to generate the hardware code of a complete MCSoC. As shown in Figure 2, this MCSoC is composed of processing tiles (PT) interconnected using a NoC.

A processing tile (Figure 2b) is composed of different IP (Intellectual Property) cores interconnected by a Wishbone bus, an internal shared bus defined by OpenCores [11]. These IP cores include memory (RAM), processor, GPIO, timer, UART jtag and NI (Network Interface). The NI core allows the transmission of the data from (to) the tile to (from) the NoC.

ProNoC allows to generate a NoC. A NoC router is shown in Figure 2c. It is composed of Input and Output ports and a crossbar switch. The route of the packet is computed in LRC. In order to support quality of service for different

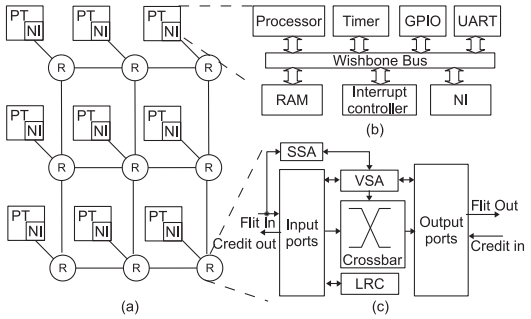

Fig. 2. Overview of an MCSoC obtained using ProNoC [10]

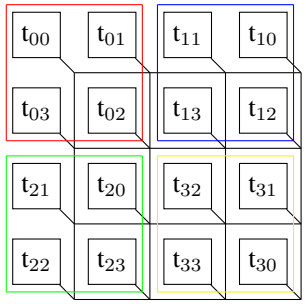

Fig. 3. Application mapping in the many-core (A0: upper left, A1: upper right, A2: bottom left, A3: bottom right)

messages, the NoC can use different virtual channels (VC). The management of these virtual channels is done by the SSA and VSA parts of the router.

ProNoC proposes a graphical interface in order to customize the MCSoC. It is possible to configure the definition of the PTs, *i.e.* what are the IP cores used for each PT. The parameterizitation of the NoC includes the number of virtual channels, the size of the buffers, the routing algorithm (XY, adaptative routing, ...), the switch arbitration (RRA, WRRA) and the topology (2D Mesh, Torus, ...).

The main goal of ProNoC is to provide the FPGA implementation of a fully functionnal MCSoC. Once all the MCSoC parts have been constructed, the tool generates the Verilog files that can be compiled for the FPGA. ProNoC tool provides also a NoC simulator to evaluate the performance of the NoC. Finally, ProNoC contains a NoC emulator. It provides a behavioural execution model of the MCSoc and the programming interface for processors cores.

B. A preliminary case study

We illustrate our solution based on ProNoC on a small case study. This case study is composed of 4 applications, named A_0 to A_3. Each application A_i is composed of 4 tasks $t_{i,0}$ to $t_{i,3}$. The communication graph for each application is given in Figure 4. The size of all the packets is 3 flits. One flit (flow digit) is 4 bytes.

As represented in Figure 3, the NoC is a 2D-mesh network. It uses XY routing algorithm. The flits are stored in input queues of the NoC routers. The size of these queues is 4 flits.

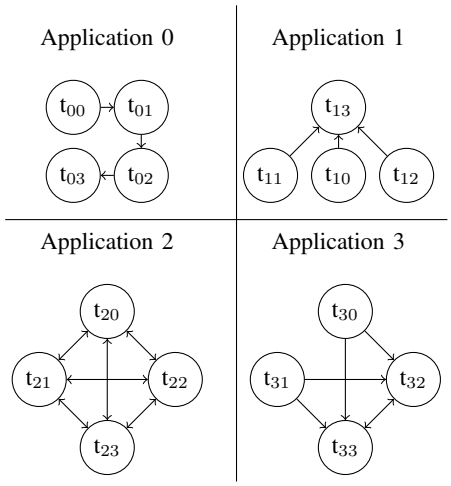

Fig. 4. Communication task graph for each application

The scheduling policy is round-robin. No virtual channel is used.

Each core of the tiles executes at most one task.

The applications are allocated on the many-core using SHiC strategy [8]. Figure 3 shows the resulting mapping.

Using the ProNoC emulator, the tasks are executed by the processor core of the tiles. Our methodology is as follow. (1) The task send the packet using `ni_transfert` function. (2) This function asks the transmission to the network interface of the tile. (3) A timestamp function is started. This function gets the global clock value. (4) When the data are received by the destination tile, the NI sends an interruption to the processing core executing the reception function. (5) The handler of this interruption reads the packet and gets the clock value.

Finally, the difference between the sending time and the receiving time is the global transmission delay obtained using the ProNoC tool. The results are given in Table I. We compare the results with theoretical ones obtained using the recursive calculus method described in [12]. The results show that the delays obtained using the implementation are much larger than the ones computed using a theoretical tool. It is due the overhead in the source and destination tiles.

V. Envisioned solution

The theoretical computation of delays takes into account the transmission between source and destination tiles. However it ignores the delays within these tiles. Preliminary results on the small use case show that these tile delays cannot be neglected. Thus they have to be precisely characterized. Therefore a precise analysis of the delays induced by tile architecture has to be conducted.

In the case study, the ProNoC emulator is used. The case study has to be extended with an implementation on an FPGA which can then be connected to an AFDX network, in order to obtain the architecture in Figure 1. We also have to consider more complex (realistic) case studies and different NoC features.

TABLE I
Transmission delays of each flow (in μs)

Flows	Practical delays	Theoretical delays
t_{00} to t_{01}	27.28	0.12
t_{01} to t_{02}	24.22	0.12
t_{02} to t_{03}	24.7	0.12
t_{10} to t_{13}	54.06	0.12
t_{11} to t_{13}	25.04	0.12
t_{12} to t_{13}	38.02	0.12
t_{20} to t_{21}	16.9	1
t_{21} to t_{20}	8.49	1
t_{21} to t_{22}	24.78	1
t_{22} to t_{21}	24.36	1.2
t_{20} to t_{22}	23.02	1.2
t_{22} to t_{20}	38.22	1.2
t_{20} to t_{23}	15.76	1.2
t_{21} to t_{23}	17.5	1.2
t_{22} to t_{23}	24.68	1.2
t_{31} to t_{32}	24.24	0.6
t_{31} to t_{33}	52.02	0.6
t_{32} to t_{33}	23.46	0.4
t_{30} to t_{33}	37.68	0.4
t_{30} to t_{32}	38.66	0.24

References

[1] DO-RTCA, "178c," *Software considerations in airborne systems and equipment certification*, 2011.

[2] Aeronautical Radio Inc. ARINC 664, *Aircraft Data Network, Part 7: Avionic Full Duplex Switched Ethernet (AFDX) Network*, 2005.

[3] V. Nélis, P. M. Yomsi, L. M. Pinho, J. C. Fonseca, M. Bertogna, E. Quiñones, R. Vargas, and A. Marongiu, "The Challenge of Time-Predictability in Modern Many-Core Architectures," in *14th Intl. Workshop on Worst-Case Execution Time Analysis*, Madrid, Spain, 2014, pp. 63–72.

[4] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[5] L. Abdallah, J. Ermont, J. Scharbarg, and C. Fraboul, "Towards a mixed NoC/AFDX architecture for avionics applications," in *IEEE 13th International Workshop on Factory Communication Systems, WFCS*, 2017, pp. 1–10.

[6] J. Ermont, S. Mouysset, J. Scharbarg, and C. Fraboul, "Message scheduling to reduce AFDX jitter in a mixed NoC/AFDX architecture," in *Proceedings of the 26th International Conference on Real-Time Networks and Systems, RTNS*, 2018, pp. 234–242.

[7] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager, "Time-critical computing on a single-chip massively parallel processor," in *Proc. of the Conf. on Design, Automation & Test in Europe (DATE'14)*, 2014, pp. 97:1–97:6.

[8] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. of the 50th Annual Design Automation Conference*, 2013, p. 39.

[9] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul, "Reducing the contention experienced by real-time core-to-i/o flows over a tilera-like network on chip," in *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*. IEEE, 2016, pp. 86–96.

[10] A. Monemi, J. Wei Tang, M. Palesi, and M. N. Marsono, "Pronoc: A low latency network-on-chip based many-core system-on-chip prototyping platform," *Microprocessors and Microsystems*, vol. 54, pp. 60–74, October 2017.

[11] OpenCores, "WISHBONE System-on-Chip (SoC) interconnection architecture for portable ip cores." [Online]. Available: https://opencores.org/howto/wishbone

[12] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul, "Wormhole networks properties and their use for optimizing worst case delay analysis of many-cores," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, Siegen, Germany, June 2015, pp. 59–68.