Appropriate Multi-core Architecture for Safety-critical Aerospace Applications
Certifiable Real-time Switching Network

Stephan C. Stilkerich [1], Christian Siemers[2] and Christian Ristig[2]
[1]*EADS Innovation Works, Willy Messerschmitt Str. 1, Munich, Germany*
[2]*Department of Computing, University of Technology Clausthal, Clausthal-Zellerfeld, Germany*

Keywords: Safety-critical, Multi-Core, Network on Chip, Beneš-Network, Real-Time Switching, Aerospace, Certifiable.

Abstract: The continues improvement of aircraft's as well as the steady optimization of the overall air traffic during the last decade increased the demand for processing power in the aircraft and on ground, simultaneously. Multi-Core platforms could offer the demanded processing power and form factor, but today's multi-core components are principally not usable for any safety critical industry and especially not usable for the avionic domain, because these components are optimized for average case performance and not for full fledged predictability. This is especially true for the inter-core communicaton network. We argue that a regular and low overhead Beneš-Network communication structure between the cores and between the cores and the shared resources, can smoothly pave the way for certification of multi-core architectures in the avionic domain. The presented details on the regular structure of the network, the scheduling variants discussed in the paper and the currently on-going research work to proof profound theoretical limits, bounds etc. substantiate our claim for an utilization of multi-cores in the avionic domain and with respect to the valid regulations for airborne equipment.

1 INTRODUCTION

Since more than a decade profound improvements of the aircraft itself and the overall air traffic management have continuously and substantially increased the demand for processing power in that field. Typical improvements include (1) the Fly-by-wire systems, (2) tighter aircraft separation during take-off & landing, (3) on-board maintenance and (4) provisions to reduce the environmental impact (noise, fuel usage). At the aircraft level, the fly-by-wire system is one of the most well known examples where computer systems have been introduce. The traditional hydraulic systems have been replaced by electrical components and a processing platform with software that performs and monitors these kinds of avionic functions. Furthermore this system ensures that the pre-defined flight envelopes are respected. Summarizing, today's modern aircrafts execute about 8-10 million lines of code on processing platforms that are classified as safety critical. At the air traffic management level, new functions to optimize and make the overall air traffic safer include tighter separations of the aircrafts during take-off and landing, collision warning systems and also the navigation at the airports for short and congestion free paths. Again, all these functions call on the one hand for massive processing power and on the other hand are classified as safety critical.

Due to the strict safety regulations enforced by independent authorities worldwide and the need to achieve product certification, so far only dedicated single core platforms have been utilized. In detail, the reason why only dedicated single core platforms have been deployed so far is founded by the fact that for system or sub-system certification *deterministic worst case behavior with no single point of failure* has to been proven. Currently all multi-core architectures and platforms have the following issues during certification (Stilkerich, 2013):

- Single point of failure with respect to the clock and power lines. Failures in a clock or power line affects the complete multi-core and hence all functions.
- Shared caches posses a complex access pattern and coherence strategy. A deterministic or predictable behavior is extremely difficult to proof.

DOI: 10.5220/0004801901800185
In *Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2014), pages 180-185
ISBN: 978-989-758-000-0

- Inter core communication network is mostly optimized for average case performance and non-deterministic. Consequently, not suitable for certification.
- Access to shared resources are non-deterministic and hence not suitable for certification.

In this position paper we argue for deterministic and real-time switching networks, Baseline and Beneš network, to systematically address the last two points mentioned before. The regular and low overhead structure, composed of simple switching elements, offer unmatched advantages and solutions for safety critical systems and in the end a way forward towards certification of multi-core architectures.

The rest of the paper is organized as follows: Section 2 provides some more background on the aerospace regulations and processes that have to be fulfilled to receive sub-system or system certification. The focus will be on the dedicated regulations for software and hardware. Section 3 presents the current status of our research work on the proposed network structure and preliminary results from first implementations and experiments.

2 SAFETY REGULATIONS

Each and every airborne component undergoes a strict and rigorous qualification process to proof that the component fulfills all its requirements and can finally receive credits for certification. These processes are described in different official documents respectively regulations (RTCA, 2013). Two regulations are of central interest for our discussion, namely the DO-254 called "Design Assurance Guidance for Airborne Electronic Hardware" addressing electronic hardware, including CPLDs, FPGAs and processors as well as the DO-178B called "Software Considerations in Airborne Systems and Equipment Certification" that addresses all kind of software that is executed, ranging from drivers, operating systems to application software.

Obviously, both specific regulations for electronic hardware and software have strong dependencies and influence each other. But both regulations also share a common underlying principle, namely that both force the development (HW or SW) towards correctness by construction and not by testing to identify possible errors. Correctness implies for safety critical applications also deterministic worst case behavior of the hardware and the software that runs on a specific hardware platform. Hence a predictable and deterministic switching network between the cores is fundamental for the certification of any multi-core platform. Without that elementary feature of the core-to-core network no certification with respect to DO-254 or DO-178B of the hardware and the executed software on that platform is ever possible.

Hence we argue for the following structure and approach of the network that is certifiable.

3 NETWORK

As inter-processor communication shall be performed by a Network-on-Chip (NoC), the obviously necessary requirement is that this NoC must be real-time-capable. Therefore, switched networks-on-chip were and are the first choice for further research. Switched networks were studied in the past, a comprehensive summary is e.g. given in (Newman, 1988). Switched networks are mostly based on cross-bar-switches as shown in Figure 1 (Aust, 2013). To configure a switch like this, only one bit is required. To connect 2^k (= N) sources to any of 2^k destinations (or any number less than these limits), at least k layers, each consisting of 2^{k-1} cross-bar-switches, are used. This forms the so-called baseline network as shown in Figure 2. The baseline network uses the inverted shuffle permutations (-1): For $N = 8$, the first layer permutes $2^3 = 8 - 1$ lines, the second divides into two sub-layers each with $2^2 = 4$ lines, and the third layer consists of 4 subsections permuting each 2 lines.

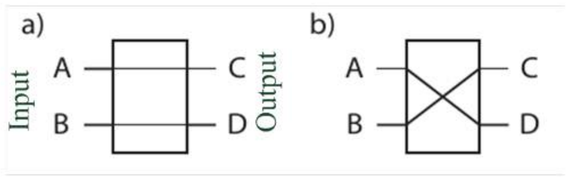

Figure 1: Cross-bar switch: a) Bar configuration, b) cross configuration.

The baseline network shows significant advantages concerning resource consumption and routing. The network resources scales with N * log N, N being the number of sources and destinations (= 2^k), and it is capable of connecting any source to any destination. Furthermore there are permutations of sources and destinations that can be connected simultaneously as well, but this is not guaranteed for every permutation: Actually, most permutation can not be routed simultaneously.

The second advantage is the capability of local routing. There is exactly one way between any source and any destination. This results in using the destina-

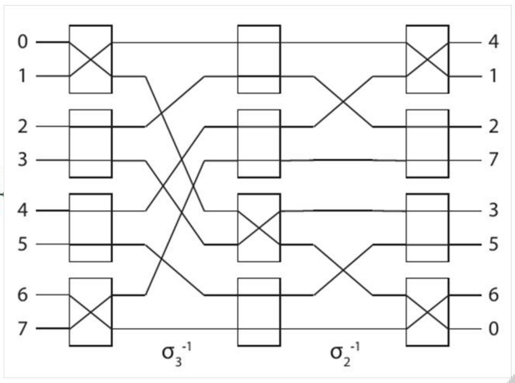

Figure 2: Baseline network for N=8, [Aust13] [AR10a] [AR10a].

tion address for routing as shown in Figure 3 (including a routing conflict).

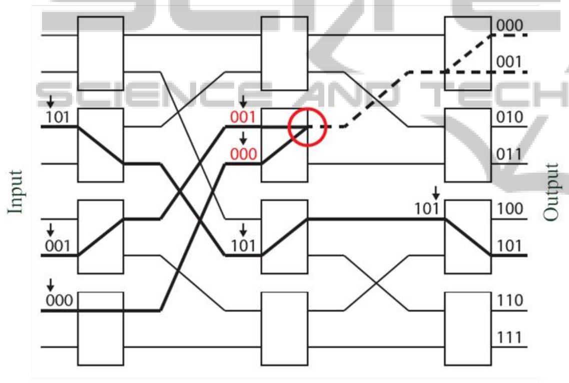

Figure 3: Local routing for baseline network [Aust13]: The CBS-configuration depends on the destination address bit resulting in routing conflicts.

As there exists only one way between any source/destination pair, the local routing is very easy and straightforward. At any layer of the switching network, the corresponding address bit decides whether to use the upper (address bit: 0) or lower (1) output, and no global routing is required. This enables the usage of self-synchronizing networks as discussed below.

The above mentioned disadvantage, the lock of required routes by already existing routes, must be solved for receiving real-time capability. To perform this, at least three scenarios are worth for further research:

- Scheduling,
- Self-synchronizing network using limited message length,
- Non-blocking networks.

3.1 Scheduling

Scheduling must be performed in a centralized manner. Any processor must request a communication line providing the source and destination address, the length of communication (e.g. in cycles or word to be transmitted) and the deadline. The sampled requests are then scheduled using a conflict graph to obtain a feasible schedule (or not). As shown in (Waldherr et al., 2013), this problem is NP-Complete, therefore finding a complete solution is intractable for a large number of sources, destinations and messages. To solve this, (Waldherr et al., 2013) shows that several heuristics exist including a stimulated annealing. In real-world applications (which are of course not located on one die but distributed in an embedded system), e.g. 50 nodes (sources and destinations) with more than 6000 messages and 700 transmit requirements are practical applications. To obtain a schedule for roughly 10000 nodes with 500000 edges within a conflict graph, a computational time of some 100 ms to few seconds was required on state-of-the-art-PCs. This concept actually appears to be usable for precompiled and scheduled applications but fails for dynamic approaches as the time for compute new schedules exceeds practical limits.

3.2 Self-synchronizing Network

Another useful approach might be a self-synchronizing network. In this case, the NoC uses buffer capabilities in front of each cross-bar-switch (CBS) to store a complete message. The buffer size strictly limits the size of a complete message through the NoC to some bytes, but on the other side, this approach routes (and schedules) the network traffic automatically.

The necessary condition for this type of network communication is that all routing information is available inside the message itself implying that the routing information is independent of other communications. Figure 4 shows a part of the self-synchronizing NoC. Any incoming message from a microprocessor or another network part must contain the destination address, e.g. in the first word. A small finite state machine related to the CBS will detect the next route by quickly analyzing the destination address. If the CBS is already occupied this means that the switch is currently active and a transmission from its own input buffer to the following input buffer is performed the switch might be either used for transmission, if the following conditions are given:

- The CBS is already in the correct state for the transmission

- The following input buffer is not occupied by any pending or not completed transmission

In the worst case, the message transmission has to wait until both conditions are given. Finally, if the receiving unit, e.g. another on-die-microprocessor core, does not block communication this can still happen if the communication buffer is still locked the communication time has a finite worst-case value (WCCT, Worst-Case Communication Time), and the communication is real-time capable, if this fits to the requirements of the application.

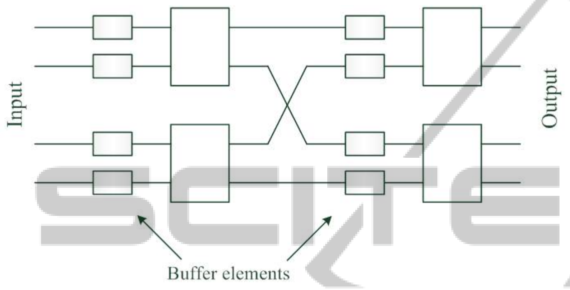

Figure 4: Self-synchronizing network using buffer elements in front of each switch.

This approach has some similarities to self-synchronizing networks-on-chip, e.g. the micropipeline approach. Nevertheless, some uncertainties and drawbacks must be solved. The required hardware resources and the synchronization protocol are still subject for further research, the computation of the WCCT might result in unrealistic high values, and furthermore questions like introducing priorities and their influence to real-time behavior and the use of periodic transmissions must be evaluated.

3.3 Non-blocking Networks

The third approach enhances the network itself and uses a non-blocking network type. Figure 5 shows the Beneš network for N = 8 (k = 3). This NoC is based on the baseline network using two networks, where the second one holds an inverted sequence of layers and the inner two layers are melted into one. The most interesting characteristic of the Beneš network is the fact that any connection request can be fulfilled as long as all destinations are distinct. To achieve this, more than one route between any two nodes must be available, and therefore routing is much more complicated and is no longer possible on local base.

The Beneš network requires a global routing algorithm. Currently the so-called looping-routing is used. This routing algorithm computes both outer layer first in parallel and continues this by computing the next inner layer. The algorithm is complex

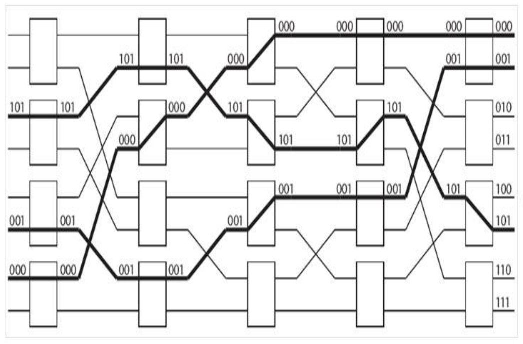

Figure 5: Beneš network for N = 8.

and time-consuming, if the actual required communications do not show any constraints.

To solve this drawback, pre-configured routings are used. Pre-configured routing means that the routing configurations are computed during compile time and stored in the target's non-volatile memory. If the actual configuration shall switch to another, communication must be finished and the new configuration is loaded and switches the CBSs accordingly.

The obvious disadvantage of this approach, the static routing during communication intervals and therefore inflexibility against dynamically changing situations, is uncritical as long as the application is static concerning active processes and threads. This is fact within the aerospace application domain, therefore we assume no restriction from this. If dynamic routing becomes a necessary condition, another approach is the separation routing approach, which divides and approximates routes until a solution is found. Future work will be performed to integrate this algorithm into hardware without using software capabilities.

4 FIRST RESULTS AND NEXT STEPS

As the hardware approach for non-blocking network was used, it could be shown that the pre-compiled routing could be integrated and the communication itself works dependable showing real-time behavior. This approach simply means that scheduling is still required in the sense that the requirements of the application to communicate between processors is now mapped to switching between configurations. Figure 6 shows the top-level schematic of a Beneš network with $N = 8$. On the left and on the right side you can identify the $N = 8$ input and output structures; these are realized by 4 cross-bar switches with two inputs and outputs each.

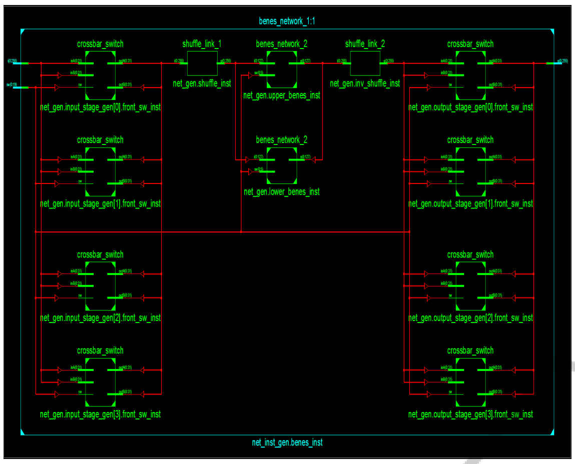
benes_network_1:1
crossbar_switch
shuffle_link_1
benes_network_2
shuffle_link_2
net_gen.shuffle_inst
net_gen.upper_benes_inst
net_gen.inv_shuffle_inst
net_gen.lower_benes_inst
net_gen.input_stage_gen[0].front_sw_inst
net_gen.input_stage_gen[1].front_sw_inst
net_gen.input_stage_gen[2].front_sw_inst
net_gen.input_stage_gen[3].front_sw_inst
net_gen.output_stage_gen[0].front_sw_inst
net_gen.output_stage_gen[1].front_sw_inst
net_gen.output_stage_gen[2].front_sw_inst
net_gen.output_stage_gen[3].front_sw_inst
net_inst_gen.benes_inst

Figure 6: Schematic of Beneš network with N = 8.

Figure 7 shows the Beneš network with $N = 8$ with expanded sub-hierarchies to visualize some of the typical intermediate stages (cf. figure 5) of a Beneš network.

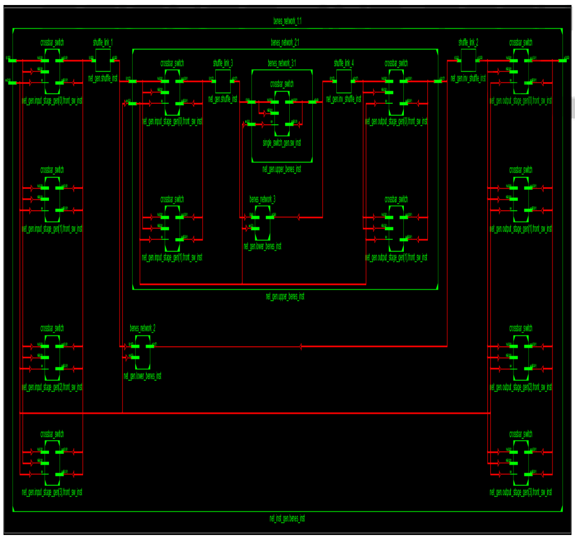

Figure 7: Partly expanded schematic for Beneš network with N = 8.

Figure 8 illustrates the resource usage of a Beneš network with $N = 8$ on a Xilinx xc3s700an device. Precisely we have the following data: Number of Slices: 736 out of 5888 or 12% utilization respectively Number of 4 input LUTs: 1280 out of 11776 or 10% utilization.

Further work will be done to evaluate the other approaches, specifically the approach using a self-synchronizing network and the approach of global routing. For global routing, all approaches are time-consuming, and we will follow the approach to map at least one of them into hardware. For the self-synchronizing network approach on the other side, proof of real-time capability, specifically of the WCCT, is pending. Furthermore, both approaches uses more hardware resources must be quantified in future.

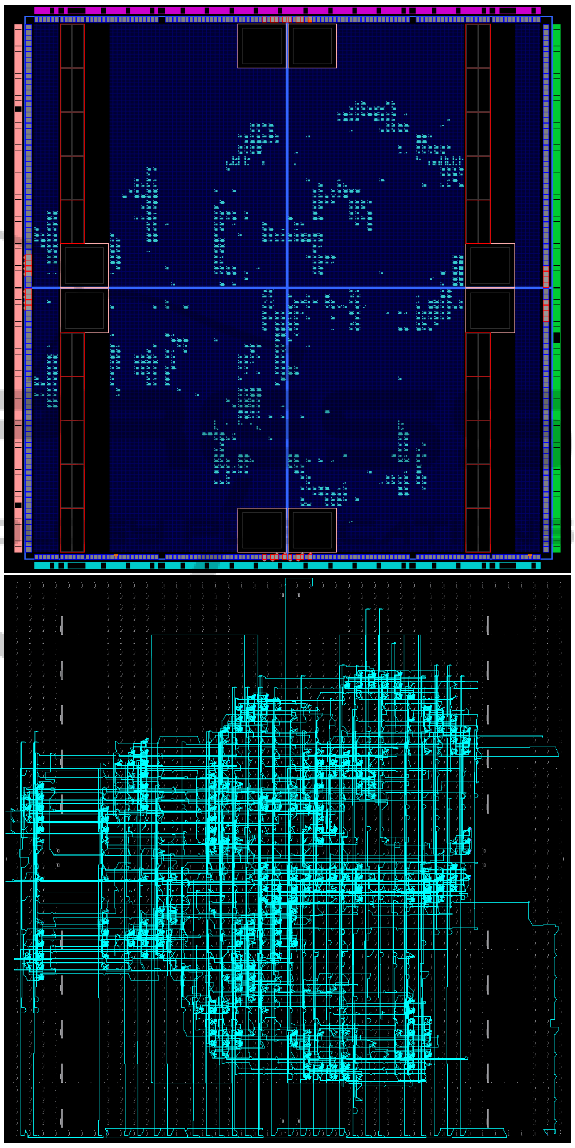

Figure 8: Place (top) and route (bottom) results for Beneš network with $N = 8$. Device: Xilinx xc3s700an.

5 CONCLUSIONS

In this position paper we have argued for a certifiable real-time switching network structure for multi-core architectures. The need for such a specific network was systematically derived from an analysis of currently available multi-core platforms and their corresponding drawbacks with respect to safety critical system. Due to the requirements and regulations valid in the aerospace domain, the use of multi-core platforms without the possibility of qualification and certification is not possible.

First results are promising and will be extended by a more concrete implementation with 4 cores at the beginning and our advocated network.

REFERENCES

Aust, S. (2013). *Ein Echtzeit-Parallelrechner zur Rezentralisierung von Steuergerten in Automobilen*. PhD thesis University of Clausthal, Clausthal-Zellerfeld Germany.

Aust, S. and Richter, H. (October 2012). Real-time processor interconnection network for fpga-based multiprocessor system-on-chip (mpsoc). In *4th International Conference on Advanced Engineering Computing and Applications in Sciences; ADVCOMP 2010*, Florenz, Italy.

Aust, S. and Richter, H. (September 2010). Space division of processing power for feed forward and feed back control in complex production and packaging machinery. In *World Automation Congress; WAC 2010*, Kobe, Japan.

Newman, P. (1988). Fast Packet Switching for Integrated Services. PhD thesis University of Cambridge, Cambridge, UK.

RTCA (2013). *RTCA documents*. http://www.rtca.org/store_list.aspl.

Stilkerich, S. C. (May 2013). Multicore systems in the light of aerospace safety regulations. In *8th MultiCore Developers Conference*, San Jose, USA.

Waldherr, S., Knust, S., and Aust, S. (2013). *Message Scheduling for Real-Time Interprocessor Communication*. to be published.