Abstract: The continuing 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. Typical improvements include (1) the Fly-by-wire systems, where hydraulic systems are partly replaced by electrical components and electronic platforms with software performing the avionic functions, (2) tighter aircraft separation during take-off & landing, (3) on-board maintenance to prevent unscheduled repair and (4) provisions to reduce the environmental impact (noise, fuel usage). Multi-Core platforms could offer the perfect balance of processing power and form factor to match with the limited resources of the avionic compartment. But today’s powerful components off-the-shelf multi-core platforms are principally not usable for the safety critical systems of the avionic domain, because these components are optimized for average case performance and not for predictability. We argue that a regular and low overhead structure, represented by a special structured real-time switching network 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.

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. 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 The traditional hydraulic systems have been replaced by electrical components and a processing platform with software that performs and monitors these kind of avionic functions. Furthermore this system ensures that the pre-defined flight envelopes are respected. Summarizing, today’s modern aircraft’s 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 aircraft’s 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, the need to achieve product certification, so far only dedicated single core platforms have been utilized. The reason why so far only dedicated single core platforms have been deployed 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 difficult access pattern and coherence strategy. A deterministic or predictable behavior is extremely difficult to proof.
- 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 a deterministic and real-time switching network, 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 for 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 process to 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 officially 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 are mostly based on cross-bar-switches as shown in Figure 1 (Aust, 2013). To configure such a switch, only one bit is required. To connect 2k (= N) sources to any of 2k destinations (or any number less than these limits), at least k layers each consisting of 2k-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): The first layer permutes 8 lines, the second divides into two sub-layers each with 4 lines, and the third layer consists of 4 subsections permuting each 2 lines.

![Cross-bar switch](image1)

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

![Baseline network](image2)

**Figure 2:** Baseline network [Aust13] [AR10a] [AR10a].

The baseline network shows significant advantages concerning resource consumption and routing. The network scales with N * log N, N being the number of sources and destinations (= 2k), and it is capable of connecting any source to any destination. Fur-
thermore there are permutations of sources and destinations that can be connected simultaneously as well, but this is not guaranteed for every permutation. 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 destination address for routing as shown in Figure 3 (including a routing conflict).

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

The above mentioned disadvantage, the lock of required routes by 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) 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), 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 some seconds was required on state-of-the-art-PCS. This concept actually appears to be usable for pre-compiled 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. This strictly limits the size of a message through the NoC to some bytes, but on the other side, this approach schedules the network traffic automatically. Figure 4 shows a part of the self-synchronizing NoC. The worst case is that the CBS is routed for a communication received by the other input, and the message has to wait for one communication cycle to receive the correct switch. If the receiving CBS is blocked (by a stored message also waiting to be routed and sent), it must wait for another cycle or even more, but if the final destination does not block, the communication time has a finite worst-case value (WCCT, Worst-Case Communication Time), and the communication is real-time capable.

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

Nevertheless, this approach still has some questions to be replied and drawbacks to 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, furthermore questions like introducing priorities and the 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 is inverted and the inner two
layers are melted into one. The most interesting characteristic of the NoC 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. This algorithm is complex and time-consuming, but if pre-configured routing patterns are used it is easy to integrate this into hardware. 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, all communications must be finished and the new configuration is loaded and switches the CBSs accordingly. Another approach is the separation routing which separates 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;

realized by 4 cross-bar switches with two inputs and outputs each.

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 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. This network works self-organizing, but the proof of real-time capability, specifically of the WCCT, is pending. Furthermore, this approach uses more hardware resources.

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 and the requirements and regulations valid in the aerospace domain without qualification and finally 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


