

System Integration of a Degraded Vision Landing Aid Application on a Custom High-Criticality Research Platform

Michael Paulitsch
EADS Innovation Works

Dietmar Geiger, Bernd Koppenhoefer and Peter Ganal
EADS Cassidian

ABSTRACT

The progress of silicon integration has led to the ability to integrate complex systems on a single die. Integration of applications and different application software components on a distributed system-on-chip can be demanding unless one follows a structural system integration approach with architectural support from hardware. The ACROSS Multi-Processor System-on-Chip platform provides architectural means for integration, such as well-defined communication interfaces, deterministic communication schedules, fault-containment and error-confinement support. We present the non-functional requirements of a degraded vision landing system for a helicopter and show how the ACROSS Multi-Processor System-on-Chip research platform alleviates integration of software and system components.

CITATION: Paulitsch, M., Geiger, D., Koppenhoefer, B. and Ganal, P., "System Integration of a Degraded Vision Landing Aid Application on a Custom High-Criticality Research Platform," *SAE Int. J. Aerosp.* 5(1):2012, doi:10.4271/2012-01-2117.

INTRODUCTION

In recent years, the decreasing relative cost of electronics and the pace of electronics development has significantly increased the desire of adopting modern commercial off-the-shelf (COTS) computing architectures in avionics. Increasing demand for energy efficiency and performance leads to the will of adopting multi-core processors, and especially Multi-Processor System-on-Chip (MPSoC) in the avionics domain. The problem of leveraging COTS hardware in avionics, however, is many-fold. For guidelines on certification of COTS hardware originating from its high integration level confer e.g. to a certification memorandum from EASA [1]. Yet, MPSoCs potentially also introduce additional problems related to isolation of functionally disjoint functions; the topic this paper addresses.

In the past, increased integration led to centralized avionics systems called integrated modular avionics (IMA). Their advantages are lower purchasing cost, lower size and weight, reduced power needs and lower maintenance costs [2]. In contrast to pre-IMA systems, IMA integrates functions of different criticality on the same set of resources. In order to keep safety aspects simple and thus feasible, unrelated functions should be segregated from each other and ideally run in separated environments. These concepts of time and space partitioning are defined e.g. in ARINC 653 [3].

The partitioning concept comprises spatial and temporal separation. Spatial separation ensures that partitions are not able to access other partition's memory spaces, whereas temporal separation guarantees that partitions do not influence each other's timing behavior. In general words, partitioning ensures that applications have the same behavior on IMA systems as previously in separated platforms. Multi-core processors introduce some challenges regarding temporal isolation [2,4], originating from physically shared resources, such as Network on Chip (NoC), shared caches and memory controllers.

According to the European Aviation Safety Agency (EASA) [6], most existing MPSoCs are classified as "highly complex micro controllers". The EASA defines a micro controller as highly complex, if at least one of the statements below is true:

- "more than one Central Processing Unit (CPU) is embedded and they use the same bus (which is not strictly separated or which uses the same single port memory)"
- "several complex interfaces are dependent on each other and exchange data"
- "several internal busses are integrated and are used in a dynamic way (for example, a dynamic bus switch matrix)"

Most of the existing MPSoCs were not designed with a special focus on certification and fall into this category.

There are several solutions in hardware available or described in the works that target alleviating concerns during the certification process. E.g. Rosen et al. improve bus scheduling for NoCs [10], Paolieri et al [11] and Diemer and Ernst [12] provide NoC solutions with different traffic classes, one of which getting guaranteed response times. Of course there are also pure software solutions executed on COTS hardware components, e.g. [7,8,9]. A hardware-based solution described and used in this paper is based on a Time-Triggered NoC (TTNoC) approach [6] and described in more detail below.

The goal of this work is to address multi-core solutions in aerospace and to evaluate the TTNoC solution in the context of a sample application, a helicopter landing aid application. The main goal is to create a sample mapping to a prototype, integrate and leverage fault tolerance approaches as well as domain-specific standards. This work has been performed as part of a public research project called ACROSS (ARTEMIS Cross-Domain Architecture) [13].

BACKGROUND

The ACROSS Multi-Processor System-on-Chip Architecture

The ACROSS Multi-Processor System-on-Chip MPSoC architecture has been developed with the following objectives and implemented by the following means [5,6]:

a). Increased level of abstraction: The ACROSS MPSoC introduces an increased level of design abstraction. Core of the architecture is the notion of a Micro Component (μ Component), a highly autonomous component, like a node in a distributed system. A μ Component has sufficient local resources (like memory) for executing its program code, and a precisely defined message-based interface as described below. A μ Component is considered to be a self-contained computational element with its own hardware (processor, memory, communication interface, and if needed interface to the physical environment) and software (application programs, operating system). A μ Component interacts with other μ Components by exchanging messages through the TTNoC.

b). Message-based Linking Interfaces (LIFs): A μ Component has an explicitly defined interface which is used to interact with other μ Components. This LIF is message-based and defines communication in the value and the time domain to ensure temporal composition. A μ Component cannot reconfigure traffic patterns directly.

c). All explicit interactions occur exclusively over the LIFs to ensure that understanding of the LIFs is sufficient to understand the entire system (i.e., the MPSoC) and no unwanted interaction occurs. This complexity-management strategy eases system integration, verification and certification.

d). Deterministic Interconnect: The network infrastructure interconnecting the LIFs of the different μ Components is temporally deterministic as a time-triggered NoC is implemented. The TTNoC transmits messages according to an a priori defined time-triggered schedule.

e). Fault containment and error confinement: Fault containment and error confinement are among the most important objectives of the ACROSS MPSoC to enable cost-effective certification. The architecture sees the LIFs as fault-containment and error confinement zones for externally induced faults (like single event upsets) on μ Components as well as design faults of μ Components. Design faults of applications cannot alter the communication patterns of LIFs due to the temporal and spatial partitioning mechanisms of the TTNoC. Common-mode faults (like power-supply faults) are not addressed at this architectural level.

The TTNoC provides the deterministic interconnect, fault containment and error confinement. This implements an encapsulation mechanism. An encapsulated communication channel is a unidirectional data channel that transports messages at pre-defined points in time. The encapsulation mechanisms prevent temporal and spatial interference between different μ Components. For instance, delaying or overwriting a message is not possible. In an ACROSS MPSoC, encapsulated communication channels are established between TISSes (Trusted Interface Subsystems) through the time-triggered operation of the TTNoC. TTNoC accesses are arbitrated between different μ Components through a Time-Division Multiple Access (TDMA) scheme. Every μ Component implements its own local memory. Because a μ Component can neither directly interfere with communication (e.g., through deliberately sending messages at certain instants) nor change a channel's configuration (these are exclusively managed by a dedicated HW instance called the Trusted Resource Manager (TRM)), an ACROSS MPSoC implements segregation in the temporal and spatial domain.

Figure 1 depicts the realization of an MPSoC architecture with application cores, I/O elements and other System on Chip (SoC) building blocks. The basic architectural elements of an ACROSS MPSoC are the μ Components, which are interconnected through a Trusted Subsystem (TSS). The TSS provides a deterministic on-chip communication. It consists of the Trusted Interface Subsystems (TISSes), the TTNoC and the Trusted Resource Manager (TRM). The TISSes are interaction points between the TTNoC and the μ Components. They act as TSS guardians by ensuring that a μ Component can only send at allocated time slots. The slot allocation follows a time-triggered schedule, defined by the TRM. The TRM is a dedicated system component that manages the whole TDMA-based communication inside the MPSoC. It exclusively configures the time-triggered schedules of the TISSes and the TTNoC.

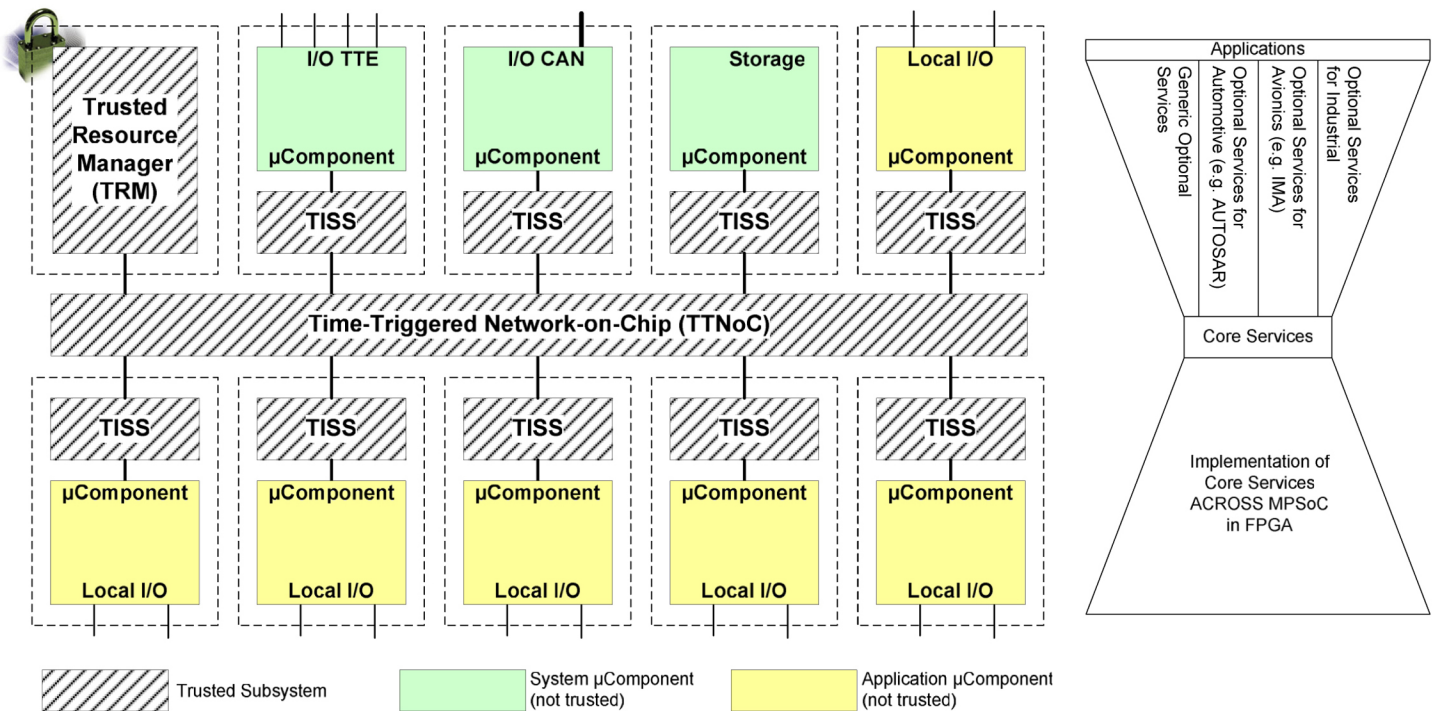


Figure 1. ACROSS SoC architecture with its Time-Triggered Network on Chip (TTNoC) (left); ACROSS provides a waist line architecture with core services building the foundation (right)

Prototype Implementation of the ACROSS MPSoC

The ACROSS MPSoC prototype is implemented using an Altera Stratix IV FPGA. The prototype consists of a TTNoC with 4 fragment switches, 8 TISSes and 8 μ Components as depicted in Figure 2. The μ Components are implemented using Altera Nios II soft-core RISC processors. Four of the μ Components are reserved to implement core system functions and the other four are reserved to execute user-defined applications. The core system functions are off-chip communication (I/O), file operations (Storage Component), debugging and tracing (Diagnostic Unit), and TTNoC management (TRM). Each μ Component for user-defined applications runs the real-time operating system PikeOS [14], a micro-kernel-based real-time operating system supporting mixed-criticality applications [6].

THE HELICOPTER LANDING AID SYSTEM

The example application, the helicopter landing aid system, is an implementation of a Degraded Vision Landing Aid (DEVILA) system designed for use in helicopter aircrafts. DEVILA is a system to support helicopter pilots, when landing in a desert environment with limited visibility. Visibility is limited since the air downstream from the rotor causes sandstorm conditions, which are also known as “brownout”.

Figure 3 shows the different hazards that the system intends to help prevent by providing sensor-based information to the pilot. To give the pilot an aid for orientation, a set of symbols is shown in the head up display (HUD), which shows e.g. a horizontal line or altitude information. This increase of safety is especially needed when starting or landing the aircraft. In those critical phases many accidents take place, which shall be avoided by providing the pilot a synthetic vision aid based on information collected from the inertial navigation system and radar altimeter system.

Figure 4 depicts the demonstrator integrating the ACROSS MPSoC. An Microsoft Flight Simulator provides realistic sensor data over an ARINC 429 communication interface. A display presents the resulting information computed in the MPSoC on a screen. Figure 4 also describes the symbols presented to the pilot (vertical speed, drift direction and velocity and acceleration).

The main goals of the implementation is to learn about the ACROSS MPSoC abilities. Especially with respect to

- Meeting hard real-time schedules (determinism)
- Integrating fault tolerance (different cores and partitions)
- Integrating different applications with different criticalities (Integrated Modular Avionics)
- Integration of domain-specific standards (e.g. ARINC)