Simulation Model and Environment for Mixed-Criticality Networked Multi-Core Chips

Mohammed Abuteir[1], Zaher Owda[2], Hamidreza Ahmadian[2] and Roman Obermaisser[2]

[1] TTTech Computertechnik AG, Vienna, Austria,
[2]Chair for Embedded systems, University of Siegen, Germany
Email: [1]mohammed.abuteir@tttech.com, [2]zaher.owda@uni-siegen.de,
[2]hamidreza.ahmadian@uni-siegen.de,[2]roman.obermaisser@uni-siegen.de

Abstract—**The requirement for networked multi-core systems is expanding dramatically in many domains such as, aerospace, industry 4.0 and automotive areas. Today's systems that implicate multi-cores, complex architectures and a big variety of applications require well-structured and hierarchical simulation environments. In addition, an efficient and early validation of system requirements during design phase and not during implementation or deployment phases is essential. This has led to a higher demand for simulation tools and environments that can meet the increasing expectations for simulation accuracy, real-time and mixed-criticality support. However, currently there is a gap between the mixed-criticality integration at chip-level and off-chip level, which is a challenge for upcoming mixed-criticality systems with multi-core chips. In this work, a hierarchical simulation environment for mixed-criticality systems based on multi-core chips is introduced. The presented simulation model utilizes criticality-aware interfaces and gateways throughout the different simulated system levels, and successfully combines different simulation tools for the scope of the simulated case realization. In addition, an avionic use-case is presented to validate the presented simulation environment.**

I. Introduction

The use of multi-core processors in embedded systems enables new applications with high-performance requirements such as embedded vision systems for autonomous vehicles [1]. In addition, the computational power of a multi-core processor facilitates a higher physical integration, where several electronic functions can be implemented on a single chip. In large electronic systems, e.g. the distributed in-vehicle electronic system of a car, this higher integration allows providing given services with fewer ECUs compared to systems with single-core processors. Benefits include a reduction of cabling, lower hardware cost, less weight and easier installation.

The physical integration often leads to mixed-criticality systems, if the functions of the multi-core processor exhibit different safety assurance levels. In this case, mechanisms for temporal and spatial partitioning [2], [3] are required, which establish fault containment and the absence of unintended side-effects between functions.

At the same time, multi-core processors introduce significant challenges for safety-critical systems and mixed-criticality systems. An example of such a challenge is the sharing of resources (e.g., caches, buses and inputs/outputs), which can lead to temporal interferences precluding the assurance of the real-time requirements. Therefore the use of multi-core processors in safety-critical systems is a cause of concern to certification authorities. For example, avionic certification authorities point out that the features of multi-core processors could cause a loss of integrity, a loss of availability or non-deterministic behavior [4]. In order to overcome these challenges, the integration of functions with different criticality using time and space partitioning has been introduced at various integration levels in prior research. Operating systems and execution layers that provide these services based on task scheduling, memory protection and suitable hardware abstractions are available as products (e.g. PikeOS [5], Deos [6]).

However, a single multi-core chip is insufficient in many embedded applications. Therefore, hierarchical networks including off-chip and on-chip networks are required. Likewise, hierarchical networks are required to achieve a system reliability beyond the reliability of a single chip and to satisfy resource requirements exceeding the capacity of a single chip. As a consequence, hierarchical platforms emerge in which cores inside a multi-core chip interact by on-chip networks whereas multi-core chips are interconnected by off-chip networks.

At present, there is, however, a significant gap between the mixed-criticality integration at chip-level and off-chip level, which is a challenge for upcoming mixed-criticality systems with multi-core chips. This paper introduces a simulation framework of multi-core platforms for a hierarchical system perspective of mixed-criticality applications combining the chip and off-chip level. This combination is established through a gateway to enable vertical integration and seamless communication in hierarchical networks respecting mixed-criticality safety requirements. We support message-based NoCs and off-chip networks with different timing models, while also establishing real-time guarantees, fault isolation and protocol transformations.

The remainder of the paper is structured as follows. Section II provides an overview of related work in the area of simulation environments for cluster and chip communication. Section III introduces the simulation tools that are used to establish the simulation framework. The realization of the simulation framework using GEM5 and OPNET Riverbed is discussed in Section IV. The simulation framework is instantiated for an example in Section V. The paper finishes with a conclusion in Section VI.

II. Related Work

In the last decades, the use of embedded systems in many application domains (e.g. automotive electronics, avionic) has rapidly increased. With the increasing complexity of such heterogeneous systems, testing of embedded devices at the

978-1-5386-4155-2/18/$31.00 ©2018 IEEE

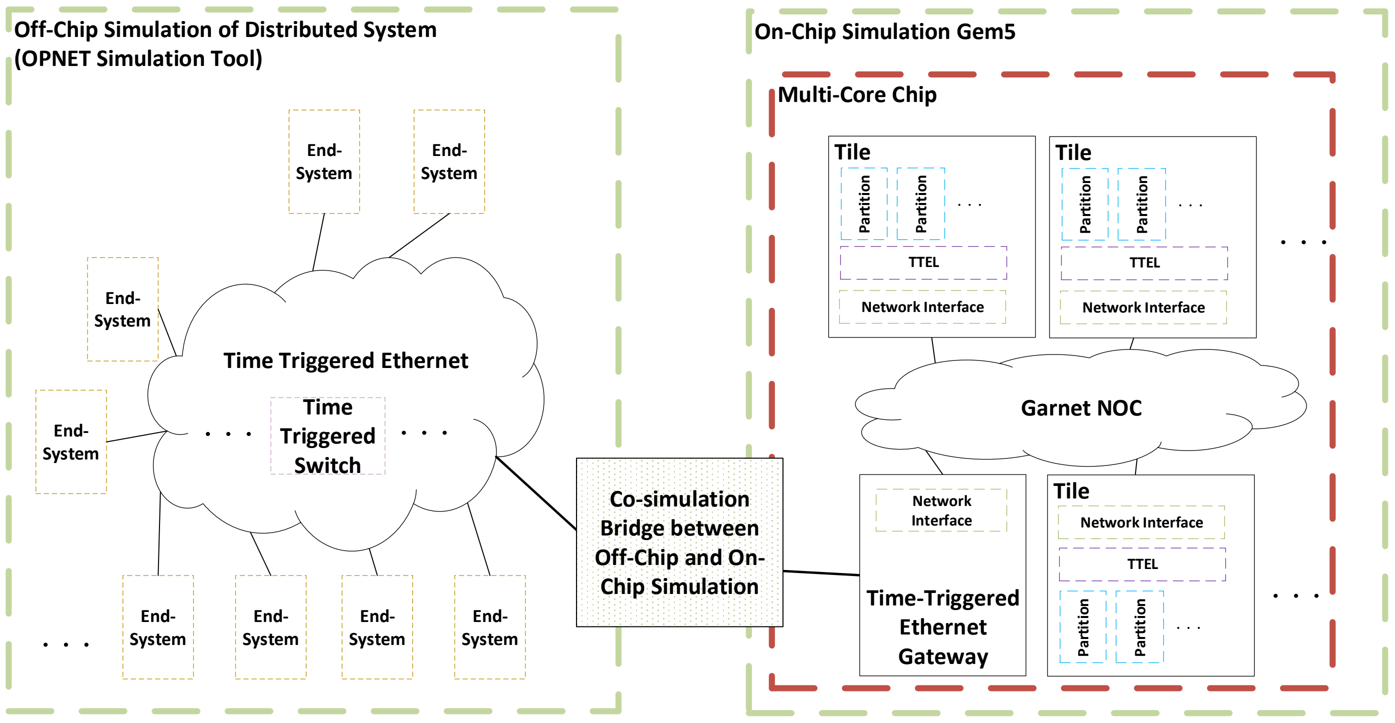

Fig. 1: Simulation Model

cluster and chip level has become a challenging and costly process.

In previous work [7] a Time-Triggered Ethernet (TTEthernet) simulation environment based on OPNET [8] was introduced. This environment introduces generic building blocks such as end-systems and switches that can be instantiated and configured based on application-specific communication schedules (e.g., period and phase of time-triggered messages, minimum interarrival time of rate-constrained messages) [7]. Also, the simulation of TTEthernet is supported using OMNeT++ [9], where a model based on standard Ethernet is introduced in the INET framework. A TTEthernet simulation model based on SystemC/TLM was realized for the design and integration of Cyber-physical systems (CPSs) [10], where communication is supported with different hardware components through a memory-mapped bus.

Beside the off-chip communication, the simulation of on-chip networks is necessary to understand the Multi-Processor System-on-a-Chip (MPSoC) system level implications of network techniques and evaluate MPSoC design proposals with a realistic interconnect model.

Many Network-on-Chip (NoC) simulators have been developed using SystemC such as NOXIM [11] and NNSE [12]. These NoC simulators simulate on-chip interconnection networks, that are only suitable to evaluate the performance of network architectures using synthetic packets. The NOXIM simulator can be instantiated and configured based on the user design by defining several parameters of a NoC (i.e. network size, buffer size, packet size distribution, routing algorithm, selection strategy, packet injection rate, traffic time distribution, traffic pattern, hot-spot traffic distribution). NNSE enables to analyze the performance impact of NoC configuration parameters (e.g. network topology, flow control, routing algorithm, application specific traffic patterns, etc.), and evaluate the network with the traffic patterns in terms of latency and throughput.

Several researchers have developed simulation platforms using QEMU and SystemC to simulate CPU cores as well as on-chip interconnection network such as [13] [14].

[15] developed a detailed cycle-accurate interconnection network model (GARNET). The GARNET model is a network model that covers the entire system at chip level and not only the network interconnection, and it is possible to interconnect multiple cores with different networks and components like caches and memory controllers.

III. Simulation Tools

This section gives an overview about the simulation environment that has been used to implement the simulation model of the mixed-criticality networked multi-core chips; OPNET Riverbed simulation and Gem5 simulation tool.

1) OPNET Riverbed Simulation Tool: OPNET Riverbed is a tool suite for discrete event simulations of communication networks. Simulation models in OPNET Riverbed are organized hierarchically consisting of three main levels: the simulation network, node models and process models. These three levels represent the structure of real-world distributed systems consisting of clusters with networked nodes (e.g., end systems, switches) and different protocol layers.

The top level refers to the complete distributed system and is used to create a network model using building blocks from the standard library and user-defined components. At this level, statistics about the network are collected, the simulation is executed and results are viewed. The node models are at the second level in the hierarchy and have a modular structure. The node is defined by connecting various modules with packet streams. The connections between modules allow packets and status information to be exchanged between modules. The modules in the nodes are implemented by using process models, the lowest level in the hierarchy. Process models are represented by finite state machines, and a process interface that defines the parameters for interfacing other process models and configuration attributes. Finite state machine models are described as embedded C or C++ code blocks. The hierarchical structure of the models, coupled with support for C and C++ code, allows for easy development of communication or networking models.

2) Gem5 Simulation Tool: Gem5 is a cycle-accurate, modular platform for simulating system-level architectures and processors microarchitectures. It provides multiple instruction set architecture (ISA) support and several CPU models that can be combined with multiprocessors system simulations. Moreover, Gem5 is widely used for the evaluation of power and energy efficiency of the simulated systems, it is possible to run simulations in full-system and system-call execution modes. Besides that, gem5 provides simulation models for memory systems [16].

Gem5 is a discrete event simulation framework, it has an object-oriented structure that offers flexible compositions to describe complex simulation designs. Gem5 provides a high level of accuracy in performance estimation and high simulation speed [17]. Workloads of different domains can be employed using benchmarks.

Gem5 is built on M5 [18] which is used for modeling networked systems, and GEMS [19] multiprocessor simulator for memory systems. Moreover, modeling complex multiprocessor system-on-chips is possible by configuring the simulation parameters (e.g. number of cores, microarchitecture, memory system, cache levels, simulation cycles). The latter parameters are used in the Python-based Gem5 scripts to adjust the hierarchical simulation setup accordingly.

IV. Simulation Model

The simulation model of the simulation environment is depicted in Figure 1. The simulation model comprises a mixed criticality multi-core NoC, TTEthernet end-systems and TTEthernet switches. The mixed criticality multi-core NoC is implemented using the Gem5 simulation tool, while the TTEthernet system is implemented using the OPNET Riverbed simulation tool. Therefore, the integration of these simulation tools requires the presence of communication and coordination interfaces. Furthermore, the coordination of these simulation tools requires the synchronization of the simulation steps since the simulation step on one simulation tool may depend on the result of the other simulation tool.

In our system model, we distinguish three types of criticality according to traffic types:

- Time-triggered control for periodic messages: They are temporally defined by a period and phase with respect to a global time base and they have the highest priority.
- Rate-constraints for sporadic messages: Sporadic messages represent rate-constrained communication with minimum interarrival times between successive message instances.
- Best-effort for aperiodic messages: They have no timing constraints on successive message instances and no guarantees with respect to the delivery and the incurred delays and they have the lowest priority.

In the following, a description of the simulation environment for the mixed-criticality multi-core NoC [20] and TTEthernet system is provided. Moreover, the integration of the Gem5 and OPNET simulation tools is described.

A. Mixed Criticality Multi-Processor System-on-Chips

In a mixed-criticality MPSoC, tiles are interconnected by a mixed-criticality NoC. A tile can host a single or multi-core processor, an off-chip/on-chip gateway or an on-chip memory. Processor cores can be virtualized by mixed-criticality hypervisors (e.g., XtratuM or KVM) [21], which establish one or several partitions. Each partition executes application tasks that utilize the computation resource separately. According to the application and architecture requirements, the NoC has a corresponding topology for interconnecting the tiles and the on-chip routers (e.g., mesh, torus, folded torus, hypercube, octagon).

It is assumed that the NoC comprises three building blocks, on-chip Network Interfaces (NIs), physical links and on-chip routers. The NI serves as an interface to the NoC for the tiles by exchanging the messages between the tiles and the interconnect. Routers relay the messages from a sender NI or an adjacent router to the destination NI or the next router. Physical links establish the physical connection between the aforementioned two elements.

Below, the simulation model of each building block is described.

1) Network Interface: The NI injects the messages from the tile into the interconnect and delivers the received messages from the interconnect to the tile. A sender NI inserts the route information according to the configuration information into the message and builds the messages, based on the information given by the core. The NI at the destination side receives the fragments of the messages (known as flits) and assembles the messages to be delivered to the tile.

A mixed-criticality NI is composed of the following building blocks, the core interface, the Mixed-Criticality Unit (MCU) and the back-end.

The *core interface* provides the processor cores with *ports* that offer a memory-mapped access to the NoC [22]. From the application point of view, ports can be *input* or *output*; The core is able to write the message into an output port and thereafter the message is sent and queued at an input port, which is on the destination tile. Based on the semantic of the message, ports can be classified as *state* or *event*. In case of a state port, the data area is a buffer with update-in-place semantics, i.e., it is overwritten whenever a new data becomes available from the core or the network. In case of an event port, new messages are added to a queue of message buffers.

The MCU makes the NoC capable of mixed-criticality applications and offers fault containment by temporal partitioning of safety-critical and low-critical messages. This is basically done by buffering the messages that are originated from low-critical subsystems and timely injecting of the messages that are originated from safety-critical subsystems. In this way, a runtime scheduling of resource requests, such as allocating bandwidth or processing of queued messages is performed and the interconnect need not care about the priority of messages or even whether the message originated from a safety-critical subsystem or a non-safety-critical one.

The MCU provides also the timing grantees (e.g., bounded latency and jitter, guaranteed bandwidth) and the integrity of messages sent by other tiles. The contention between messages of different traffic types is resolved by considering periods and phases for periodic messages, the Minimum INterarrival Times (MINT) as well as the priorities for sporadic messages and in case of available bandwidth, the transmission of aperiodic messages. The MCU injects the Time-Triggered (TT) messages based on a given schedule. In case of Event-Triggered (ET) messages, the MCU injects the messages based on the priority and the rate-constraints of the messages.

The segregation between the TT and ET messages is established using timely blocking or shuffling (in some cases). *Timely blocking* [23] is established by time-triggered guarding windows, which block the transmission of any other ET messages. It is guaranteed that the bandwidth is free for the injection of the TT message and eliminates the impact of ET messages on TT messages by the cost of lower bandwidth usage. However, in case of shuffling, the MCU considers the TT message as the highest priority message and hence, TT messages may incur a delay, if an ET message has occupied the NoC prior to them. This results in a higher jitter for TT messages, but a more efficient usage of bandwidth by ET messages.

The third building block is called as back-end and has no understanding about the concept of time and criticality. This is exactly the NIs in the existing NoCs and performs all the NoC-specific operations. This unit provides the building blocks of the NI with an interface to the routers by generating the packets and consequently flits and sending the flits to the routers at the sender tile [24]. In addition, this unit disassembles the packets and flits received from the NoC and sends them to the destination input ports for incoming messages at the destination tile.

2) Routers: On-chip routers relay the messages to the next NI or router, based on the information obtained from the header. Each router contains input and output units, configuration registers, a switch and control logics, which are continuously updated by the header to implement the flow control functions required to buffer and forward the messages to destinations. Depending on the implementation of the NoC, a distributed or source-based routing scheme can be used [20]. In a distributed routing, the path to the destination is computed or looked-up by a route computing unit at each hop. In case of a source-based routing, the route is embedded in the header and it will be decoded at each hop.

3) Implementation of the NoC Model in Gem5: The above described model was implemented using the Gem5 simulation environment. As shown in Figure 2, a Time-Triggered Extension Layer (TTEL) [22] was added to the on-chip NI of the fixed pipeline GARNET interconnection network [15]. The intention of using the TTEL at the on-chip NI is to realize the functionality of the MCU and to make the NoC capable of supporting mixed-criticality applications.

The simulated NoC, models a classic five-stage pipelined router, including input buffers, routing logic, allocators and the crossbar switch, with Virtual Channel (VC) flow control at flit-level. It also supports both mentioned communication paradigms, i.e., TT and ET.

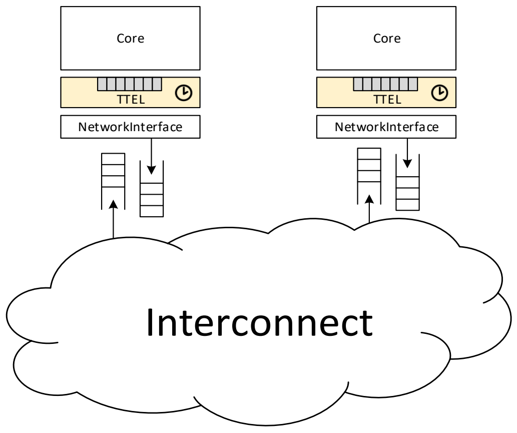

Fig. 2: Simulated NoC in Gem5

The time-triggered behavior in Gem5 was technically realized by extending the `Consumer` class and adding a *Scheduled Wake-up* to this class. This class is a virtual base class of all classes that can be the targets of wakeup events. This change enables the classes inherited from `Consumer` to be waked-up not only by the linked consumer, but also by the scheduled wake-up (using `schedule` method).

Each message is an instantiation of the extended `NetworkMessage` class. The extension was performed to add mixed-criticality relevant parameters (e.g., message type, temporal characteristics, deadline, virtual-link ID) to the class to be used by the MCU (e.g., to forward the message to the destination port). In addition, statistical information (e.g., timestamps, arrival times) are added to the `NetworkMessage` class to be used at the destination core for the simulation results.

The NI and the routers in GARNET handle the messages of different types and priorities in the same manner, as the type and the priority of the messages are abstracted from them. This is the task of the MCU to establish a collision-free communication of the TT messages and to guarantee no impact of the ET message on the TT messages due to contention at the resources (e.g., buffers at routers, physical links).

4) Simulation Model of the Off-chip/On-chip Gateway: Each multi-core NoC has an off-chip/on-chip gateway. The off-chip/on-chip gateways are used in order to establish the end-to-end communication over heterogeneous and mixed-criticality networks. The connection between off-chip and on-chip networks is established through gateways. An off-chip/on-chip gateway is responsible for the redirection of messages between the NoC and the off-chip communication network. Additionally, the off-chip/on-chip gateway provides a solution for mixed-criticality systems with different timing models, fault isolation and real-time guarantees.

The gateway was realized as a tile connected through the NoC to the others tiles. The NoC simulation model uses the fixed pipeline GARNET interconnection network [15] with a configurable network topology. Gem5 also allows to configure the number of cores (e.g., CPU, memory controller, L2 cache controller, etc.). The TTEL and gateway are integrated into the GARNET interconnection network.

The structure of the simulation building block of the gateway is illustrated in Figure 3. The constituting layers of

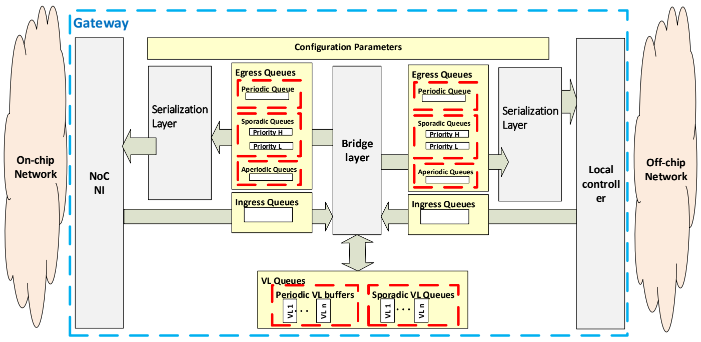

Fig. 3: Gateway simulation Building Block

the gateway are the bridge, serialization, local controller as well as ingress, egress and VL queues.

The cache coherence protocol "Network test" [1] was extended to allow to the gateway to interact with the GARNET interconnection network.

In what follows, a description of the simulation building block of the gateway is provided.

Simulation Queue Elements: The gateway has three types of queues, as follows:

- Ingress queue: It consists of one FIFO queue for each network. The ingress queue that connects the NoC interface with the bridge layer is implemented using a specific language for specifying cache coherence protocols to establish the connection between the gateway and the GARNET interconnection network. Whereas the ingress queue that establishes the connection to the off-chip network is implemented using the queue C++ class.
- Egress queues: They consist of one periodic egress queue, two sporadic queues and one aperiodic egress queue. Each sporadic queue has its own priority level. The egress queues are implemented using the queue C++ class.
- VL queues: They belong to two groups: one for the periodic messages and the other one for the sporadic messages.
  - **Periodic VL buffers**: Each periodic VL has one periodic VL buffer, which provides buffer space for exactly one message. In case this buffer is full and another message arrives with the same VLID, the newer message replaces the old one.
  - **Sporadic VL queues**: Each sporadic VL has one queue. It is possible to store several messages of the respective VL in this queue.

  VL queues are implemented also using the queue C++ class.

Bridge Layer: The bridge layer classifies the incoming message based on the message types. The periodic and sporadic messages are stored in the corresponding VL queue. This is done by extending the virtual link id from the message and comparing it with virtual link id that is stored in the look-up table. While an aperiodic message is stored in an aperiodic queue in the egress queues.

[1]http://www.m5sim.org/Network_test

The bridge layer is determined based on the time parameters (i.e. period and phase) the point in time when the periodic message is relayed from the VL queue to periodic queue in the egress queues. This ensures the deterministic communication behavior of the periodic messages.

Furthermore, the bridge layer triggers the injection of the sporadic messages, which are stored in the corresponding sporadic VL queue, based on the predefined configuration parameters. These parameters define the communication flow of the sporadic messages, which is associated with two main parameters for each virtual link: minimum interarrival times and jitter.

Additionally, the bridge layer is responsible for storing aperiodic messages in the egress queue based in the direction of the destination address.

Serialization Layer: The serialization layer responsible for injection of the different messages from their egress queue into the NoC as well as the off-chip network based on the predefined schedule for the periodic messages and based on the priority of the sporadic messages. The serialization layer has two modes to inject the messages; Timely-block and shuffling. Using the Timely-block, the serialization layer guarantees that a network is free when a periodic message arrives at the periodic egress queue. This achieved by using so call guarding window to control the transmission flow to the NoC or to the off-chip network. The guarding window opens and closes based on the predefined schedule. The guarding window prevents the start of a transmission of a sporadic or aperiodic message if it would delay the subsequent periodic message. While the shuffling, there is no used of guarding window where no messages can be transmitted. The periodic messages are assigned higher priority than sporadic and aperiodic messages.

In the shuffling, the periodic message periodic message is delayed by at most one sporadic or aperiodic message that is already in transmission.

Local Controller: The local controller is responsible to provide a communication interface from/to their networks. Furthermore, the local controllers interact with the gateway functionalities such as queuing and mapping of the incoming/outgoing messages. The gateway functionalities are responsible for converting the exchanged message formats between the simulation tools, and mapping the destination addresses of the incoming messages to the target application. These functionalities require queuing structures that handle the transmitted messages.

B. *TTEthernet System*

For the off-chip network, we implemented the simulation environment for the TTEthernet. The TTEthernet is realized using the OPNET simulation tools [25]. The simulation building blocks of the TTEthernet are listed in the following.

1) TTEthernet Switch: As shown in 4, the TTEthernet switch consists of multiple physical links and a bridge. Each physical link contains a physical layer and a MAC layer. The physical layer is built according to IEEE 802.3 [26]. The MAC layer is based on IEEE 802.3 with the following extensions. The MAC layer checks the validity of the destination address for an incoming frame. This field is extracted from the destination address using the bit mask 0xffffffff0000. If the field has the predefined value of the critical traffic marker, this frame is

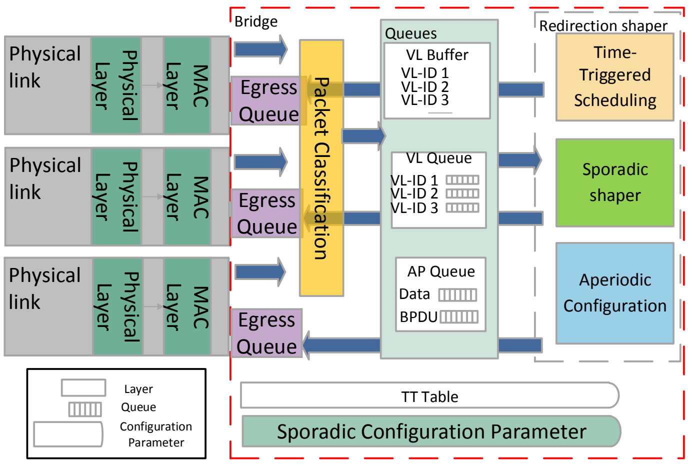

Fig. 4: Block Diagram of TTEthernet Switch

either periodic or sporadic. Otherwise, the frame is regarded as aperiodic.

The switch distinguishes between periodic and sporadic frames using the EtherType value. The IEEE Standardization Authority has assigned the values 0x88d7 [27] and 0x0800 [28] for EtherType fields of periodic and sporadic frames.

Figure 4 shows the block diagram of the bridge model. The task of the bridge is to handle and forward ingress frames to the egress ports depending on the traffic type. The bridge model contains five layers:

- Packet classification layer: it puts the ingress frames into one of the internal queues based on the traffic type and Virtual Link IDentifier (VLID). In the case of a periodic message, each periodic Virtual Link (VL) has one periodic VL buffer, which provides buffer space for exactly one message. In case this buffer is full and another message arrives with the same VLID, the newer message replaces the old one. A sporadic message has one queue for each sporadic VL. All aperiodic messages are stored in one queue since aperiodic messages have no timing constraints on successive message instances and no guarantees.
- TT scheduling layer: It is responsible for relaying periodic frames from the VL buffer to the periodic frame queue at the correct egress port according to the communication schedule. The communication schedule also determines the point in time when the time-triggered frame will be relayed, thereby ensuring deterministic communication behavior.
- Sporadic shaper layer: It realizes the traffic policing for the sporadic frame by implementing an algorithm known as token bucket [28]. This layer checks the time interval between consecutive frames on the same VL and moves rate-constrained frames from the virtual-link queue to one of the sporadic egress queues. We distinguish between two priority classes of sporadic frames using two corresponding sporadic egress queues.
- Aperiodic Configuration layer: It is responsible for relaying the aperiodic message, where the spanning tree protocol is used to establish a loop-free topology for communication of aperiodic messages [26]. The supported aperiodic messages include Bridge Protocol Data Units (BPDU) and aperiodic data messages. BPDU messages are exchanged between off-chip routers to determine the network topology, e.g., after a topology change has been observed.
- Egress queue layer: It forwards the frames from the egress queues to the MAC layer according to the priority. The highest priority is assigned to time-triggered frames, whereas aperiodic messages have the lowest priority.

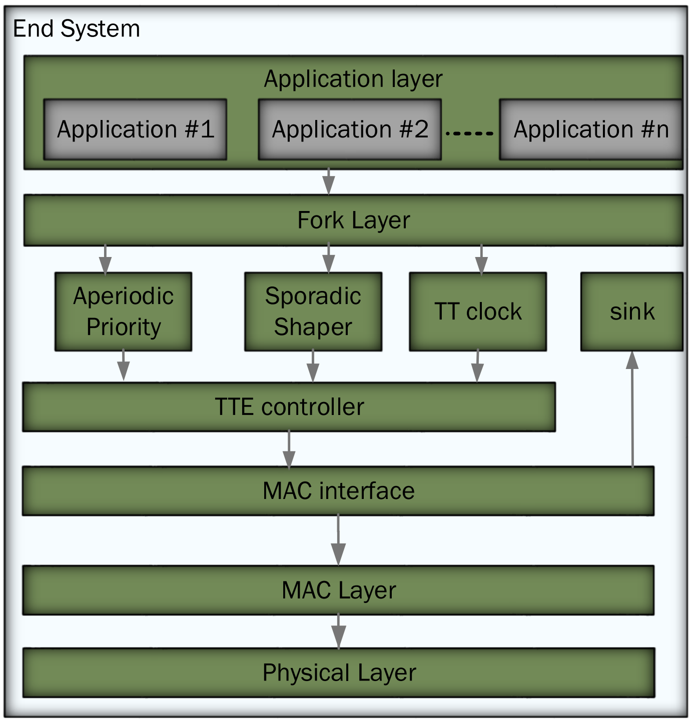

Fig. 5: Block Diagram of TTEthernet End System

2) TTEthernet End System: Figure 5 shows the block diagram of a TTEthernet end system. The end system consists of the following layers:

- application layer: it generates the payload traffic with a timing depending on the application specifications (i.e. periodic, sporadic and aperiodic). This payload frame includes meta-data with payload info such as the source ID, the sender ID, and the receiver ID. The meta-data is used to configure the lower layers.
- Fork layer: it is the interface layer between the technology-independent generic source and the technology-dependent lower layers. According to the configuration parameter in the fork layer, it passes the incoming frame to one of the following layers: TT clock, sporadic shaper or aperiodic Priority.
- TT clock: it is responsible for the deterministic behavior of the time-triggered frames. In the TT clock layer, the incoming frame is buffered in a corresponding VL buffer and then the frame is sent to the TTE controller layer at the time specified in the static communication schedule.

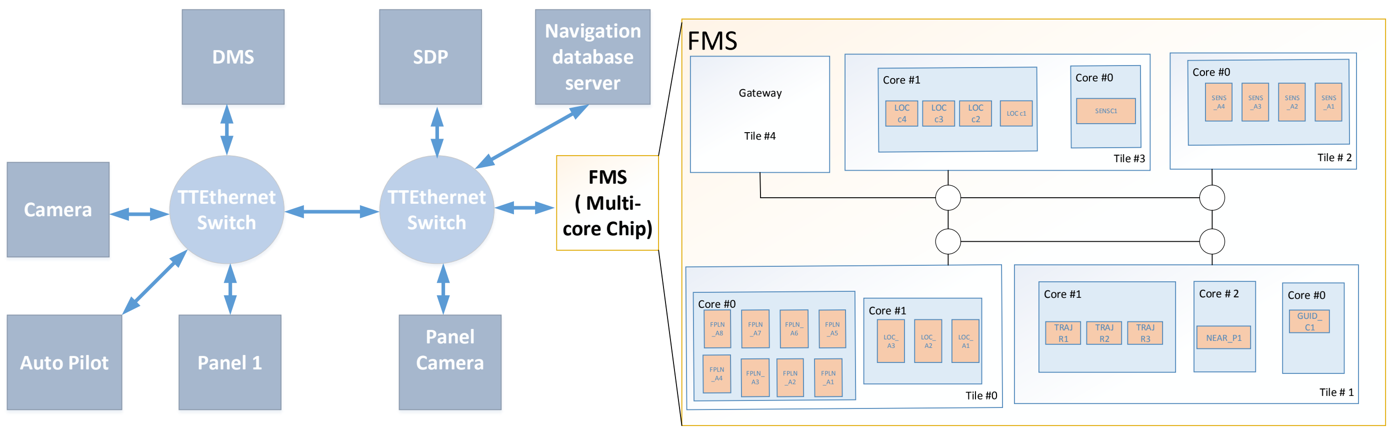

Fig. 6: Avionic Use-Case Topology

- Sporadic shaper: it is responsible for guaranteeing the minimum time interval between two consecutive instances of sporadic frames on the respective VL.

- Aperiodic priority: it forwards the frames to the TTE controller layer according to the priority. There are two priority levels of the aperiodic frames.

- TTE controller layer: The TTE controller layer contains three queues, namely one for each traffic type. It sends the frames to the lower layer according to their priority. Periodic frames have the highest priority, whereas aperiodic frames are assigned the lowest priority.

- MAC interface layer: it provides services for the outgoing frames from the upper-layer and the incoming frames from the MAC layer. The frames that come from the MAC layer are directly sent to the sink layer.

- Sink layer: the received message ends at the sink. The sink is responsible for logging and gathering statistical results.

- MAC layer: it implemented based on IEEE 803.2.

C. *Linking Simulation Tools*

The integration between the versus discrete event simulation tools (i.e. OPNET and GEM5) requires the presence of communication and coordination interfaces. Furthermore, the coordination of different simulation tools requires the synchronization of the simulation steps since the simulation step on one simulation tool may depend on the result of the other simulation tool.

The simulation tool executes its simulation based on its own local event calendar. The local event calendar contains the tool's local events as well as global events that are used to synchronize the co-simulation based on the co-simulation global calendar. The co-simulation global calendar represents the time model of the execution order of both simulation tools, it contains events that have causal relationship in the co-simulation. Moreover, managing and modifying the events of the local event calendar is mandatory in the selection of the simulation tools, As presented in [29].

Tile ID	Core ID	Description of the applications
Tile#0	Core#0	It presents different pilot actions corresponding to the management of the flight plan.
	Core#1	It has different tasks describing the pilot actions with respect to the localization tasks.
Tile#1	Core#0	It is a guidance task that is responsible for computing the parameters required to guide the aircraft.
	Core#1	Responsible for trajectory tasks
	Core#2	Nearest airports task builds during the flight a list of the nearest airports.
Tile#2	Core#0	It describes the different pilot actions that are translated into inputs for the sensor task.
Tile#3	Core#0	This task is responsible for gathering every sensor output to be passed to the localization task.
	Core#1	Different sensors used to manage and generate the most probable position of the aircraft.

TABLE I: Applications Description

V. Evaluation Use Case

This section demonstrates a real scenario of an avionic use-case using the proposed simulation framework. Figure 6 depicts the topology of the simulated avionic use-case. The off-chip topology is composed of eight nodes, namely a navigation database server, sensor data provider (SDP), display management system (DMS), flight management system (FMS), auto pilot, panel and two cameras, interconnected by two time-triggered switches in a star topology. The FMS is composed of five interconnected tiles, each of which containing a single or more cores. Tiles are connected by a 2*2 mesh topology as shown in Figure 6. Each tile contains one or more cores that run tasks as listed in Table I. As shown in Table II the data exchange of the applications is performed using periodic time-triggered messages, sporadic rate-constrained communication, and aperiodic messages.

Table II lists the simulation results for the evaluation of the avionic use-case. We observed significant discrepancies of the end-to-end jitter for different traffic types, i.e., the difference between the maximum and minimum end-to-end latency between the applications.

VI. Conclusion

A simulation environment for mixed-criticality multi-core systems is presented. This environment is based on a detailed simulation module that has the capability of simulating multi-core NOCs in combination with and off-chip system. The criticalities of the different cores and tiles can be simulated according on the simulated cases. The simulation environment presents off-chip gateway and switches in addition to on-chip interfaces, and communication paradigms required to guarantee the fault containment and preserve criticalities. The

Message Exchange						Results	
MSG. ID	Sender	Receiver/s	Type	time	meg size	Maximum Latency	Jitter
1	SENS_A1	SENS_c1	Aperiodic	(0-0.2)s	72	389 ns	87 ns
2	SENS_A2	SENS_c1	Aperiodic	(0-0.2)s	72	346 ns	60 ns
3	SENS_A3	SENS_c1	Aperiodic	(0-0.2)s	72	374 ns	76 ns
4	SENS_A4	SENS_c1	Aperiodic	(0-0.2)s	72	320 ns	55 ns
7	LOC_A1	LOC_c1	Aperiodic	(0-0.2)s	72	419 ns	81 ns
8	LOC_c2	Trajectory, nearest Airport	Periodic	1.6 s	72	249 µs	0
11	LOC_A2	LOC_c3	Aperiodic	(0-0.2)s	72	453 ns	85 ns
12	LOC_A3	LOC_c4	Aperiodic	(0-0.2)s	72	519 ns	93 ns
13	LOC_C4	DSM	Periodic	1 s	72	231 µs	0
14	NAVi.	F plan	Aperiodic	(0-2)s	72	348 µs	83 µs
15	mporary F p	DSM	Periodic	0.3s	72	215 ns	0
16	condary F pl	DSM	Periodic	0.3 s	72	240 ns	0
17	Active F plar	Trajectory	Periodic	0.2 s	72	179 ns	0
18	Active profil	DSM	Periodic	0.2 s	72	238 µs	0
19	mporary pro	DSM	Periodic	0.3s	72	221 µs	0
20	condary pro	DSM	Periodic	0.3 s	72	247 µs	0
21	NEAR _p1	DSM	Periodic	1 s	72	273 µs	0
22	GUID_c1	Auto pilot	Periodic	0.2s	72	236 µs	0
23	SPD	DMS	Periodic	200 ms	72	145 µs	0
24	Panel	DMS	Sporadic	(0-.2)s	72	283 µs	105 µs
25	SDP	SENS_A1	Periodic	0.1s	72	162 µs	0
26	DMS	Panel	Periodic	0.2 s	72	136 µs	0

TABLE II: Message Exchange and Simulation Results of the Avionic Use-Case

developed simulation environment combines multiple simulation tools that were coordinated using a co-simulation bridge responsible for avoiding data corruption and the timeliness in between the simulated cores. Finally, an avionic-based scenario is served for the evaluation of the simulation model and environment. The evaluation results provide detailed insides of the simulation at all levels of the system.

Acknowledgments

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No. 700665 (CITADEL) and the Shift2Rail Joint Undertaking under grant agreement No 730830 (SAFE4RAIL). This Joint Undertaking receives support from the European Union‘s Horizon 2020 research and innovation programme and Austria, Spain, Germany, Czech Republic, Italy, France.

References

[1] B. Kisacanin and M. Gelautz, *Advances in Embedded Computer Vision*. Springer, 2014.

[2] J. Rushby, "Modular certification," Computer Science Laboratory SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA, Tech. Rep., Sep. 2001.

[3] ——, "Partitioning for avionics architectures: Requirements, mechanisms, and assurance," NASA Langley Research Center, NASA Contractor Report CR-1999-209347, Jun. 1999.

[4] Certification Authorities Software Team (CAST), "Position paper cast-32 multi-core processors," Tech. Rep., 2014.

[5] Sysgo, *PikeOS Safe and Secure Virtualization*, 2010.

[6] DDC-I, *DEOS – A Time & Space Partitioned DO-178 Level A Certifiable Family of RTOS Products*, 2011.

[7] M. Abuteir and R. Obermaisser, "Simulation environment for Time-Triggered Ethernet," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 642–648.

[8] OPNET Technologies, "OPNET Modeler Documentation," Tech. Rep.

[9] S. U. Palm, H. D. Kenfack, F. Korf, and T. C. Schmidt, "An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy," in *SIMUTools '11: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2151120$\backslash$npapers2://publication/uuid/9FBAEB27-8FCF-4A75-9B3A-26CDED1D5C54

[10] Z. Zhang and X. Koutsoukos, "Modeling Time-Triggered Ethernet in SystemC/TLM for Virtual Prototyping of Cyber-Physical Systems," 2013, pp. 318–330.

[11] D. P. F Fazzino., M Palesi. [Online]. Available: http://noxim.sourceforge.net

[12] Z. Lu, R. Thid, M. Millberg, and A. Jantsch, "Nnse: Nostrum network-on-chip simulation environment," in *In Proc. of SSoCC*, 2005.

[13] M.-C. Chiang, T.-C. Yeh, and G.-F. Tseng, "A qemu and systemc-based cycle-accurate iss for performance estimation on soc development," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 4, pp. 593–606, April 2011.

[14] K. Nakajima, T. Hieda, I. Taniguchi, H. Tomiyama, and H. Takada, "A fast network-on-chip simulator with qemu and systemc," in *Networking and Computing (ICNC), 2012 Third International Conference on*, Dec 2012, pp. 298–301.

[15] N. Agarwal, T. Krishna, L.-S. Peh, and N. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, April 2009, pp. 33–42.

[16] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[17] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system." in *ReCoSoC*, L. S. Indrusiak, G. Gogniat, and N. S. Voros, Eds. IEEE, 2012, pp. 1–7. [Online]. Available: http://dblp.uni-trier.de/db/conf/recosoc/ReCoSoC2012.html#ButkoGOS12

[18] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, pp. 52–60, 2006.

[19] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Computer Architecture News*, vol. 33, pp. 92–99, 2005.

[20] H. Ahmadian and R. Obermaisser, "A configurable simulation model for mixed-criticality multi-processor systems-on-chips," in *4th International Conference on Knowledge-Based Engineering and Innovation (KBEI-2017) Dec. 22th, 2017, Tehran, Iran*. IEEE, 2017.

[21] S. Trujillo, A. Crespo, and A. Alonso, "Multipartes: Multicore virtualization for mixed-criticality systems," in *2013 Euromicro Conference on Digital System Design (DSD)*, 2013, pp. 260–265.

[22] H. Ahmadian, R. Obermaisser, and M. Abuteir, "Time-triggered and rate-constrained on-chip communication in mixed-criticality systems," in *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, 2016, pp. 117–124.

[23] H. Ahmadian and R. Obermaisser, "Temporal partitioning in mixed-criticality nocs using timely blocking," in *Embedded Multicore/Many-core Systems-on-Chip (MCSoC-17), 2017 IEEE 11th International Symposium on*. IEEE, 2017.

[24] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, 2004.

[25] *OPNET Modeler 17.1 Documentation*, OPNET Technologies.

[26] "IEEE standard for local and metropolitan area networks: Media Access Control (MAC) bridges," *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pp. 1 –277, 9 2004.

[27] *AS-6802 – Time-Triggered Ethernet*, SAE Std., 11 2011.

[28] *AIRCRAFT DATA NETWORK PART 7 AVIONICS FULL DUPLEX SWITCHED ETHERNET (AFDX) NETWORK*, AIRLINES ELECTRONIC ENGINEERING COMMITTEE Std., June 27 2005.

[29] Z. Owda, M. Abuteir, and R. Obermaisser, "Co-simulation framework for networked multi-core chips with interleaving discrete event simulation tools," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2015, pp. 1–8.