# On the Development of a Real-Time Ethernet Switch for Ultra-Highly Dependable Applications

Dumitru-Mircea Chimerel, Costel Patrascu, Sabina Anghel
Chip IP Design
TTTech Computertechnik AG
Bucharest, Romania
*firstname.lastname*@tttech.com

Yuval Katz, Wilfried Steiner
Chip IP Design
TTTech Computertechnik AG
Vienna, Austria
*firstname.lastname*@tttech.com

***Abstract*—**

**The development of systems for ultra-highly dependable applications requires restrictive development processes that adhere to industry standards to ensure that the end product is of superior quality. The knowledge of those standards and their application is, thus, essential for successful system development.**

**Time-Triggered Ethernet (TTEthernet) is a network platform that extends standard Ethernet with hard real-time capabilities and mechanisms for use in ultra-highly dependable applications and applications with mixed-criticality requirements. Over the last couple of years two teams in Romania and Austria have jointly implemented TTEthernet in the form of TTEthernet end systems and switches. Having first target applications in the space and aerospace area, a development process based on the avionics DO-254 standard has been used. This development process began with requirements capturing and ended with full verification for certification of the design.**

**In this paper we discuss the development process of the TTEthernet switch according to the DO-254 objectives and use a small running example, the ARINC 664-p7 age check, to illustrate the application of the standard.**

## I. Introduction

Dependable systems, like digital systems in general, tend to become more and more resource intensive with the number and complexity of tasks they perform. A standard way to meet these growing resource requirements is the adaption of technology developed in adjacent industries. In particular, the increasing demand on communication bandwidth makes Ethernet-based interconnects an attractive choice as communication network of dependable systems. Although these Ethernet-based networks share similarities with standard equipment (e.g., office equipment) there are, of course, also significant discriminators. For example, first, dependable Ethernet equipment has to be developed according to certification standards that are typically industry-specific and, secondly, additional protocol features have to be implemented to guarantee timely end-to-end transmission.

Ethernet variants that extend standard Ethernet with dependability features are for example Avionics Full-Duplex Switched Ethernet (ARINC 664-p7) [1] or TTEthernet [2], [3] (currently standardized by SAE as AS6802 [4]). ARINC 664-p7 is implemented in the Airbus A380 and Boeing 787 while TTEthernet has been selected for an upcoming space mission [5]. In this paper we discuss the development of an Ethernet switch that supports ARINC 664-p7, TTEthernet, standard Ethernet and a combinations of these protocols.

One of the main target areas of the Ethernet switch has been and still is ultra-highly dependable equipment in avionics. The relevant standards for avionics are DO-178b for software and DO-254 for hardware development. As the Ethernet switch is implemented in VHDL only, the development process had to comply with DO-254. In this paper we discuss DO-254 in general and give a "hands-on experience" by using a running example of the so-called "age check". The age check measures on a per Ethernet frame basis the time that the frame resided in the switch, i.e., the switch-imposed latency on the frame. Before the frame is forwarded by the switch, the switch checks whether the delay is below a statically configured threshold and discards the frame if the threshold is exceeded.

We continue in the next section with a detailed technological background on the TTEthernet technology as well as the age check and the development process. We then discuss individual phases of the development process in detail: the requirements capturing phase is discussed in Section III, the conceptual and detailed design in Section IV, and the verification in Section V. The conclusions are drawn in Section VI.

## II. Technological Background

### A. Short overview of TTEthernet

Ethernet is a Local Area Network (LAN) technology, standardized by IEEE and intensively used across different domains, from aerospace to home applications. The original design specifies that Ethernet does not exclude but also does not guarantee a real-time response, nor does it support access priority: the data delivery is on a "best effort" basis. Consequently, to use Ethernet for industrial, transport or aerospace applications, significant efforts have been invested to implement a certain control of the resource allocation to achieve the needed Quality of Service (QoS).

One solution that integrates QoS in Ethernet is the Avionics Full-Duplex Switched Ethernet (ARINC 664-p7), an aircraft data network. In ARINC 664-p7 end-systems remain unsynchronized, but the Ethernet frame generation follows a bounded arrival distribution. Consequently, we can calculate the maximum traffic in a network at any point in time and provide sufficient buffer resources such that no frame is lost.

However, as the frame generation is still unsynchronized, the queuing delays in the switches may become unacceptably high.

TTEthernet is a communication platform that integrates best-effort (BE) traffic, rate-constrained (RC) traffic, which is compatible to ARINC 664-p7, and synchronized time-triggered (TT) traffic on a single physical network. The time-triggered paradigm ensures that the resources needed to transmit data are reserved in time and minimizes the latency and almost eliminates transmission jitter. For time-triggered communication, the connected devices' clocks are closely synchronized through a synchronization algorithm. From the transmitter point of view, this guarantees that it will be given the necessary bandwidth to transmit at a scheduled moment in time. From the receiver and switch point of view, this guarantees knowledge of the moment the message will arrive, so the necessary memory and processing capability can be allocated. Costs are kept at a minimum through intelligent resource allocation. Furthermore, TTEthernet also provides double and triple redundancy to achieve fault-tolerance for single and multiple failures.

### *B. Avionics Certifiable Development Process DO-254*

The safety-critical applications that TTEthernet has been designed for require careful guidance and a high level of control, not only related to the performance of the system, but also to the way it is planned, designed and verified. In this context, the Radio Technical Commission for Aeronautics (RTCA) published the DO-254 in order to provide guidance for design assurance of airborne electronic hardware, for the use of equipment designers and manufacturers. There are five classes of failure in aviation, defined by the effect a certain failure would have on the safety of the aircraft, flight crew workload and efficiency, passenger's comfort and well-being. In DO-254, the Level A through Level E classes of failure (catastrophic, hazardous, major, minor and no effect) are correlated to hardware design assurance levels. TTEthernet has been designed as Level A equipment.

Hardware design according to DO-254 is divided into five processes: requirements capture, conceptual design, detailed design, implementation, and production transition. In this paper we focus on the development of the Chip IP of the TTEthernet switch that covers the first three processes; it starts with requirements capturing and results in VHDL code. The verification and validation of the VHDL has been done on an FPGA platform. In an ongoing project, the last two processes (implementation and product transition) are conducted on a radiation-tolerant platform. In the following we further discuss the first three hardware development processes.

### *C. Example: "ARINC 664-p7 Age Check"*

ARINC 664-p7 (RC) traffic is unsynchronized, but by limiting the number of frames a node may source in the network per time interval, we can calculate the maximum network utilization and, thus, the maximum size of buffers and queues required in the network switches. Consequently, we can also calculate the maximum delay of a frame through a switch already during design time of the network. On the other hand, the ARINC 664-p7 age check is a runtime mechanism to track a frame's delay through a switch and to drop a frame if it exceeds a configured upper bound in time. So, obviously, there is a certain degree of redundancy: in a well-configured network we know the maximum delay and would set the delay threshold to a value higher than this maximum. No frame would then ever exceed the delay threshold. In some sense the ARINC 664-p7 age check is, thus, an example of a "second line of defense."

## III. Requirements Capturing

The design process starts with requirements capturing; these initial requirements are later refined and implemented. According to an internal requirement standard, each requirement is stated as follows:

| | |
|---|---|
| **XXX-007** | Y shall do something, in a particular setting. |
| *Guidance:* | Something is intended to meet the intention of the writer of the requirement. |

Requirements are identified by unique tags and the word *shall*. In the example above **XXX-007** is the unique tag of the requirement. "XXX" is a unique acronym identifying the author of the requirement. The tag of a requirement is placed in a box appearing within the text. Additionally to the formal requirement text there can be informal text as context to the requirements. In particular any guidance specific for a particular requirement is placed below the requirements text, starts with the keyword *Guidance* and is set in italics. The text of a requirement will be separated from the surrounding general guidance text by means of a dedicated paragraph. Any specific guidance is optional.

In the TTEthernet switch, there are a two types of requirements. High-level requirements can be found in the requirements document (RD) and they make up the initial requirements set. Low-level requirements can be found in the conceptual design document (CDD) and they are typically derived from non-functional aspects, e.g., timing, implementation and architectural aspects.

The Chip IP of the TTEthernet switch lists 595 high-level requirements in the RD and 863 low-level requirements in the CDD. These requirements cover different aspects, e.g., how frames are handled by the Ethernet MAC, the buffer management in the switch, or the synchronization module that is used to establish a network-wide synchronized time. This synchronization module, for example, is described by 74 high-level and 23 derived requirements.

TTEthernet supports three traffic types: best-effort standard Ethernet traffic (COTS), rate-constrained traffic (RC), which is compliant to ARINC 664-p7, and time-triggered traffic (TT). As an example for the requirement capture process, we will refer to a set of requirements which addresses the dependability of RC traffic.

The latency of an RC frame in the switch has a maximum value which guarantees the system performance and depend-

ability. Latency is the time interval between the reception of the first bit of the frame and the transmission of the first bit of the frame. In the TTEthernet switch, the maximum latency is a configurable parameter. The main age checking requirement is the requirement tagged as **VLU-4650**.

| | |
|---|---|
| **VLU-4650** | The frames policed at switch ingress as critical traffic in accordance with **DCH-2052** and dispatched as rate constrained traffic which reside in the switch more than a statically configurable amount of time (rc_latency parameter defined on a per output port basis described by MNI-3724), shall be dropped by the switch. |
| *Guidance:* | When a CT frame was dropped because of age violation, the diagnosis counter rcframe_age_err is incremented by 1, according to **DCI-2745**. |

As can be seen in the requirement body, since only frames classified as critical traffic are checked for age, this requirement relates to the requirement that handles classification of frames tagged as **DCH-2052**.

| | |
|---|---|
| **DCH-2052** | The switch shall classify a frame that passed line input acceptance testing as belonging to critical traffic if the upper 32 bits of the Ethernet destination address field of the frame, masked with a configurable constant value, match a configurable 32-bit value. |

When a frame is received by the switch on a certain port, it is classified as belonging to critical or non-critical traffic, according to the **DCH-2052** requirement. A critical traffic frame can be programmed for transmission as RC or TT. If the frame is transmitted as RC traffic, the age checking must be performed against the rc_latency parameter. This parameter is defined in the requirement tagged as **MNI-3724**.

| | |
|---|---|
| **MNI-3724** | The switch shall provide a configurable per output port limit for the maximum latency of the RC critical traffic class (rc_latency register from table Table 17.2), the default value being h'773593F. |

If the internal latency of the RC frame is lower than rc_latency, the frame will be transmitted by the switch and otherwise it will be dropped, as it is required in **VLU-4650**. The rc_latency which is defined in the General Parameters Table is a configurable switch parameter, as it is required in **MNI-3724**.

| Entry Name | Size Bits | Description Requirement |
|---|---|---|
| rc_latency[i] | 27 | Represents the maximum latency allowed for an RC frame defined per output port (i from 0 to 11) **MNI-3724** |

TABLE I
GENERAL PARAMETERS TABLE (FRAGMENT)

Requirements like the ones discussed above for the age check are easily found, because the switch has to be ARINC 664-p7 compliant. So, in a sense, these high-level requirements have been derived from the ARINC 664-p7 standard document. Other functionality, like the TTEthernet clock synchronization algorithms, have been developed by means of model-based design. For that we have used the SAL model-checker developed by SRI International.

As presented, requirements have been specified in natural text following some simple conventions. While in principle more advanced and formal modeling languages such as SysML are applicable in this process, our process has been selected to meet the experience and best-practice expertise of the responsible engineers. In general, the textual form of requirements capturing is accepted in our industries of interest and standard tools are available for requirements management.

## IV. CONCEPTUAL AND DETAILED DESIGN

The Conceptual Design process generates a high-level description of the hardware item and identifies major components, constraints between hardware and software and major features. Whatever requirements derived during this stage are to be fed back to the Requirements Capture process. The results are recorded in the CDD.

The captured requirements and the conceptual design are used to produce a detailed design during the detailed design process. Any more requirements derived during this stage are also to be fed back to the requirements capture process. The detailed design data includes the VHDL source code, which was reviewed to ensure that each functional block described in the CDD complies with the code, each requirement in the RD can be traced to the code and each executable statement in the code is traced from a requirement. The DO-254 recommends that an assessment should be performed to identify potential effects of unused functions on safety, but this was not the case with the TTEthernet switch, as no unused functions exist in the code.

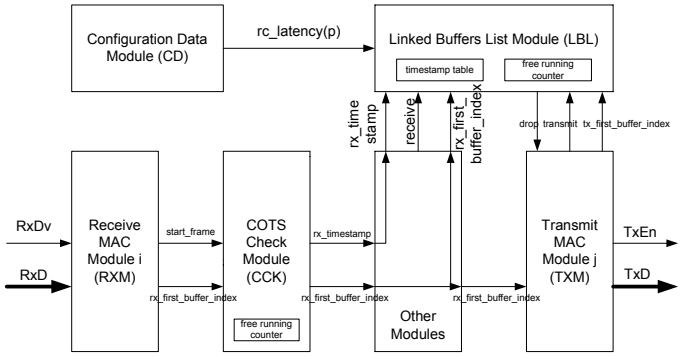


Fig. 1. Interconnection of Relevant Modules in the Detailed Design of the Ethernet Switch

The conceptual design for the requirements of the age check (**VLU-4650**) is structured in the following modules: Configuration Data (CD), Receive MAC (RXM), COTS Check (CCK), Linked Buffers List (LBL) and Transmit MAC (TXM). Figure 1 shows the interconnection of these modules. In the

following we discuss these modules and describe the detailed design in pseudo-code.

The Configuration Data (CD) module contains all configuration parameters loaded by the TTEthernet switch after reset. The COTS Check module classifies the incoming frame as critical traffic or non-critical traffic, as is required in **DCH-2052**. The Linked Buffers List module contains the lists of buffer indexes needed by the Transmit MACs for transmission. A buffer index represents the address of a 96-byte memory unit which is used to store the frame data.

Both the COTS Checker and the Linked Buffers List modules contain a free-running counter on the 125 MHz clock which is used as a free-running clock to measure the progress in real-time. This counter starts counting from 0 when the configuration of the TTEthernet switch is finished and covers approximately 1.073 seconds of elapsed time (Algorithm 1).

**Algorithm 1** Free-running Counter `gtime` Update

```
1: if config_done then
2:    if gtime == 2^27 − 2 then
3:       gtime ← 0
4:    else
5:       gtime ← gtime + 1
6:    end if
7: else
8:    gtime ← 0
9: end if
```

Whenever the first byte of an incoming frame is received by the Receive MAC a start_frame pulse is signalled to the COTS Check module. The module uses it to timestamp the incoming frame and compute its receive moment within the global time (Algorithm 2).

**Algorithm 2** New Frame Arrival

```
1: if start_frame then
2:    rx_timestamp ← gtime
3: else
4:    rx_timestamp ← rx_timestamp
5: end if
```

The timestamp of the incoming frame (rx_timestamp) is passed from the COTS Check module to other modules until it reaches the Linked Buffers List module. The Linked Buffers List module contains the timestamp table (4096 entries) where the receiving timestamps for each received frame are stored. The timestamp table is organized in such a way that the address of an entry represents the first buffer index of a received frame, while the data at the respective entry represents the receiving timestamp of the respective frame (Algorithm 3).

The Linked Buffers List contains an aging process that is executed about every 16 ms and it takes about 100 $\mu$s to check all the 4096 entries of the timestamp table. For each entry it reads the timestamp field and subtracts it from the current value of the global time counter. If the difference is greater than the maximum age time value CONST_1_SEC (which is 1 second), it writes $2^{27} - 1$ into the timestamp field of the respective entry, and otherwise it advances to the next entry. If the value read from the timestamp field already equals $2^{27}-1$, it also advances to the next entry (Algorithm 4).

**Algorithm 3** New Frame Received

```
1: if receive then
2:    timestamp_table(rx_first_buffer_index).timestamp
      ← rx_timestamp
3: end if
```

**Algorithm 4** Ageing Process

```
1: if age then
2:    if timestamp_table(age_index).timestamp
      == 2^27 − 1 then
3:       donothing
4:    else if
      |gtime − timestamp_table(age_index).timestamp| >
      CONST_1_SEC then
5:       timestamp_table(age_index).timestamp ← 2^27−1
6:    end if
7: end if
```

Whenever the Transmit MAC is ready to output an RC frame, it sends the first buffer index of the frame to the Linked Buffers List module which will perform the latency computation. The latency computation is done in order to drop RC frames that aged out inside the TTEthernet switch. For COTS and TT frames, the result of this latency computation is ignored. Age checking for RC frames is done on a per port basis, using rc_latency[p] values provided by Configuration Data module. If the computed latency exceeds this age value for the respective port, the drop signal is asserted and the Transmit MAC module will drop the RC frame (Algorithm 5).

**Algorithm 5** Age Check and Frame Drop

```
1: if transmit(p) then
2:    if timestamp_table(age_index).timestamp
      == 2^27 − 1 then
3:       drop ← true
4:    else if
      |gtime -
      timestamp_table(tx_first_buffer_index).timestamp|
      > CONST_1_SEC then
5:       drop ← true
6:    else
7:       drop ← false
8:    end if
9: end if
```

Requirements and detailed design have to be consistent and, furthermore, DO-254 requires that the mapping has to be complete. In other words, for each requirement in the RD

there must be code and for each piece of code in the detailed design there must be at least one requirement. However, quite typically, during the detailed design, code is generated for which no requirements have been defined and requirements have to be added. These additional requirements are called "derived requirements" and they can be classified in three categories: derived architectural, derived implementation, and derived timing. Derived architectural requirements are design decisions (e.g., where and how a FIFO or a counter needs to be used), derived implementation requirements are, as the name says, implementation decisions related to the particular ASIC/FPGA used in the project (e.g., maximum clock rate, memory sizes). Derived timing requirements are also design decisions, but relate to the synchronization processes that take place inside modules.

In our case, an example of a derived requirement is the derived architectural requirement **AST-1502**, which enunciates the way the Linked Buffers List (LBL) modifies timestamp memory entries, as described in the pseudo code above.

| | |
|---|---|
| **AST-1502** | Whenever the global time counter or its 21 least significant bits wrap, synchronous with the ismi.start tdma frame pulse, the LBL module shall start read-modify-write all the timestamp memory entries according to the following rules: if the read timestamp equals $0x7FFFFFF$, no update occurs; if the difference between the instant value of the global time counter and the read timestamp is larger than $0x773593F$ value, the timestamp is set to $0x7FFFFFF$. |

Derived requirements are considered as an update to the RD, but are written in the CDD, in our case in the chapter corresponding to the Linked Buffers List module. It has no traceability to a "parent" requirement in the RD.

VHDL has been selected for a multitude of reasons, but primarily for the sake of performance and reliability. Performance-wise, it allows us to operate the TTEthernet devices at a very low CPU clock rate, in the order of tens of MHz. From a reliability point of view, a hardware-specific VHDL solution has been preferred over, e.g., a software-based solution, as the VHDL solution is expected to be superior in terms of its resilience against transient failures in certain application environments and less likelihood of design errors.

## V. Validation and Verification

Validation is performed to ensure that requirements are correct and complete. Verification ensures that the detailed design meets the requirements. Furthermore, the DO-254 requires independence: a person who performs verification and validation cannot be the item's designer.

The requirement validation process has been done using checklists, as the DO-254 recommends. We created three checklists corresponding to every design module: the requirements checklist, the RTL code checklist, and the verification cases checklist. The first of these three relates to requirements validation and in this paper we will focus on this checklist. The requirements checklist lists all the requirements with their corresponding classification: either high-level or derived.

In the checklist, each requirement has a status that is either "approved", meaning that the requirement is satisfied by the detailed design, or "pending", if it is not satisfied. To determine the status of a requirement it has to fulfill several properties and in the checklist, questions are formulated that test each requirement for these properties. In particular, these questions test with respect to the characteristics of the requirements (e.g., ambiguity or fitness to the functional context), questions related to the design (traceability and consistency of the code to the requirement), and information related to verification (correspondence between the requirement and one or more verification cases, coverage, etc.).

The test engineer then uses this checklist in his/her assessment of the detailed design and will answer each question with "yes" or "no" and all questions have to be positively answered to set the status of a requirement to approved. If the answer to any of the questions is no, then the approval status of the corresponding requirement is set to pending. The responsible designer corrects the issue and then a new iteration of the validation process starts. Test and correction is repeated until all the requirements have status approved.

Requirements have been tested by applying stimuli at the inputs of the TTEthernet switch Chip IP. To generate such stimuli we have designed a "testbench" that is wrapped around the TTEthernet Chip IP. The testbench environment has been designed in the VHDL language and contains a model of the flash memory, a model of the host, and a GMII model.

The flash memory is used to store and load the configuration and the host reads and writes internal data structures of the TTEthernet switch Chip IP. GMII (Gigabit Media Independent Interface) is a standardized interface between the Ethernet MAC layer and the physical layer. Each port of the switch, i.e., the physical entity to which a network cable is plugged into, implements one such GMII interface. GMII is, therefore, used to simulate the reception and transmission of Ethernet frames by the TTEthernet switch.

The requirements have been verified by simulation using ModelSim, a simulator for VHDL providing a graphical interface. A typical test case consisted of the generation of a sequence of frames needed to trigger the tested requirement in all corner-cases, a sequence of commands executed by the host emulator, and sometimes a set of signals manually changed using the so-called "force command" of ModelSim.

The detailed design of the TTEthernet switch Chip IP consists of VHDL code that is structured into several modules (as defined in the conceptual design). In general, requirements could be tested by executing only individual modules. However, the DO-254 requires that all requirements are tested by executing the complete TTEthernet switch Chip IP. This is also called testing "at the top level". In addition to testing requirements at the top level, some requirements have also been tested at module level. Examples of such requirements are requirements that implement large counters (e.g., global timers covering one second or more of elapsed time) or rare events which are difficult to fully test only at the top level.

The verification procedure of each requirement has been

written in VHDL language or with scripts in Python language or TCL language, depending on the complexity of the requirement. TCL testing has been done for simple requirements, e.g., state machine transitions, internal counters, and also for rare events. While ModelSim interacts easily with TCL, the disadvantage of TCL testing is that the TCL scripts cause significant overhead in interacting with the ModelSim simulation. This generates long simulation times for complex requirements testing. On the other hand encoding tests directly in VHDL completely removes this interaction overhead with the scripting language and is much faster. Complex requirements have been tested therefore by encoding in VHDL.

All in all, we have defined $189$ individual test cases to verify the TTEthernet switch Chip IP. After individual test verification of each one of the test cases, a comprehensive verification process took place, during which all the tests ran in a single, continuous flow. This is called "regression testing". A TCL script automatically centralized all the data obtained during regression, and traced all the requirements from the documents to the test results.

In addition to the official verification framework, personal verification was performed by the designer of the module, to detect and correct errors early (this is a general practice, but also recommended by the DO-254). Nevertheless, the official verification structure is one based on the requirements, and not on the individual modules.

Another important part of the verification process is code coverage analysis. The project standard demanded that any code written corresponded to a requirement. The code coverage analysis was necessary because multiple code blocks referred to the same requirement and this had to be justified. Code coverage is a very important dependability issue, as unused code leads to unused functionality and even though the probability for the switch to enter that state is close to zero, its behavior still has to be safe. In the Switch IP VHDL model, the code coverage analysis performed by ModelSim with stimuli at the top level resulted in $100\%$ statement and branch coverage. This means that the IP behavior is $100\%$ accounted for through the verification process.

**VLU-4650** was one of the requirements tested at the top level and also at module level. For the top level test, a sequence of RC frames has been generated on each port by the test. The rc_latency parameter was configured differently on each port with lower values and some frames were supposed to be forwarded by the TTEthernet switch while others were to be dropped.

For testing the requirement **VLU-4650** at the module level, the reception of a sequence of rate constrained frames was emulated at the interface of the Linked Buffers List module. In this case, the value of the rc_latency parameter was set to the maximum configurable (one second). The emulator of the Transmit MAC (TXM) triggers the transmission of the RC frame at a time greater than one second after reception. The content of the timestamp_table of the respective entry is checked and also the assertion of the drop signal.

## VI. Conclusion and Future Work

We have reviewed the design process of an Ethernet switch for ultra-highly dependable systems. The team of TTTech development Romania took great care during all design phases to apply the appropriate methods. Errors in applying them where sometimes found and corrected at timed reviews. In particular, maintaining the consistency between documents is currently a manual task. However, as requirements are tagged and traceable through the different documents, a more automated process could be implemented that assists the developer, e.g., in highlighting those parts in all documents that are affected by a change in a particular requirement. Thus, simple errors could be minimized or avoided entirely.

We are currently also pursuing future work in multiple directions. TTEthernet technology-wise we are working on extensions and compatibility modes to IEEE $802.1Q$, which is currently working on amendments to incorporate time-triggered communication. The requirements management process can certainly be improved towards a mathematically formal requirements management process. From this formalization we would expect an increase in both, coverage and consistency. Coverage in a sense that we reduce the probability of missing requirements in the initial specification and consistency with respect to avoiding requirements that contradict each other. For critical aspects like the clock synchronization module of the Chip IP we, indeed, applied model-based development and even formal verification. However, the model-checking methods that we applied would need significant improvements in scalability to be applicable for the complete switch design. However, even as formal methods come with a certain overhead, consistency and completeness of requirements is highly important in our systems of interest and we are carefully analyzing several options in this direction. Code generation as well as formal testcase generation are also areas which are relevant to our technology development and future products. We are researching several options in these areas as well.

## Acknowledgments

We would like to thank all the members of the development teams who made the TTEthernet switch a success. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement $n^{\circ}236701$ (*CoMMiCS*).

## References

[1] ARINC, *ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*, AERONAUTIC RADIO, INC., 2551 Riva Road, Annapolis, Maryland 21401-7465, Sep. 2009.

[2] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch, "TTEthernet: Time-Triggered Ethernet," in *Time-Triggered Communication*, R. Obermaisser, Ed. CRC Press, Aug 2011.

[3] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Seattle, Washington*, May. 2005.

[4] W. Steiner, *TTEthernet Specification*, TTA Group, 2008, Available at http://www.ttagroup.org.

[5] C. E. Howard, "Orion avionics employ COTS technologies," *Avionics Intelligence*, Jun. 2009.