UDK621.3:(53+54+621+66), ISSN0352-9045 Informacije MIDEM 33(2003)4, Ljubljana

DISTRIBUTED EMBEDDED SAFETY CRITICAL REAL-TIME SYSTEMS, DESIGN AND VERIFICATION ASPECTS ON THE EXAMPLE OF THE TIME TRIGGERED ARCHITECTURE

Manfred Ley; Christian Madritsch

Fachhochschule Technikum Kärnten, Carinthia Tech Institute, Villach, Austria

INVITED PAPER
MIDEM 2003 CONFERENCE
01. 10. 03 - 03. 10. 03, Grad Ptuj

Abstract: The Time Triggered Architecture (TTA) and its related communication protocol, TTP/C is an emerging communication principle for distributed fault-tolerant real-time systems. Typical applications are safety-critical digital control systems such as drive-by-wire and fly-by-wire.

This paper highlights the hardware / software architecture and design of the first industrial single chip communication controller for the Time Triggered Protocol (TTP/C). An application specific RISC core with several specialized peripheral blocks, RAMs, flash memory and analog cells was implemented together with necessary protocol firmware to fulfill both cost and safety requirements. Whereas the controller chip itself can be seen as an embedded system, the composability characteristic of TTA enables a hierarchical system design style with nodes and communication clusters as higher level system components embedded into an application device like a car or airplane. A complete framework for hardware / software co-simulation and verification across all levels of hierarchy was buildt up to support the design work from chip to system level. Furthermore, system reliability and fault behavior of a safety critical system has to be shown to safety certification authorities. Extensive fault injection experiments have been performed at simulation and physical level to proof the concept, fault model and resulting implementation of an embedded TTA control system.

Distribuirani vgrajeni varnostni sistemi v realnem času – zasnova in verifikacija na primeru časovno prožene arhitekture

Izvleček: Časovno prožena arhitektura (TTA) in njen odgovarjajoči komunikacijski protokol, TTP/C, je porajajoči se komunikacijski princip za distribuirane sisteme odporne na napake v realnem času. Tipična uporaba so digitalni sistemi za nadzor kot so vožnja in letenje krmiljeno s povezavo.

V prispevku prikažemo programsko in strojno arhitekturo ter načrtovanje prvega industrijskega komunikacijskega nadzornega veza za TTP/C protokol na enem čipu. Uporabniško specifično RISC jedro z večimi specializiranimi perifernimi bloki, RAMi, FLASH pomnilniki in analognimi celicami je bilo uporabljeno skupaj s potrebnimi komponentami TTC/P protokola z namenom zadovoljiti varnostnim in stroškovnim zahtevam. Čeprav na celoten nadzorni čip lahko gledamo kot na vgrajen sistem, pa sestavne karakteristike protokola TTA omogočajo bolj hierarhični stil načrtovanja sistema z vozlišči in komunikacijskimi skupki kot sistemskimi komponentami na višji ravni vgrajenimi v neko uporabniško okolje, kot sta npr. avtomobil ali letalo. Zgradili smo celotno okolje potrebno za simulacijo in verifikacijo programske in strojne opreme na vseh hierarhičnih nivojih z namenom podpreti proces načrtovanja od čipa do sistema. Oblastem pristojnim za potrjevanje ustreznosti varnosti sistema je bilo dodatno potrebno prikazati zanesljivost in vedenje sistema v primeru napak. Opravili smo obsežne poskuse s povzročanjem napak na fizičnem in simulacijskem nivoju z namenom dokazati pravilnost koncepta, modela napak in delovanja dokončne implementacije vgrajenega TTA nadzornega sistema.

1. Introduction

The Time-Triggered Architecture (TTA) is designed for a wide range of fault-tolerant distributed real-time systems /1/. The application domain of the architecture is safety-critical by-wire systems in the automotive, aerospace and railway industries.

The key component of the Time-Triggered Architecture is a VLSI communication controller, which executes the Time-Triggered Protocol (TTP) /2//3/ and provides all communication and safety features of TTP to a host controller running the application of a network node. Based on a prototype implementation from the Technical University of Vienna /4//5/, Carinthia Tech Institute (CTI) designed an industrial (automotive specification) single chip version of such a communication controller, the TTA-C2. Together with the Technical University of Vienna and TTTech AG a complete HW-SW codesign environment was developed. Furthermore, related projects like FIT (see chapter 7.2) were carried out to proof concept as well as implementation. This paper should give an overview on this development with a focus on safety related issues and parts of the design.

After an introduction of the TTA system principles in Section 2 we describe the controller architecture and its building blocks in Section 3. Section 4 explains the design flow applied and gives technical details of the new chip. Sec-

tion 5 details the modeling strategy and tools for hardware/software co-development. The system development process and the proof of safety relevant system behavior are explained in Sections 6 and 7. Finally we summarize the work done within the TTA activities at CTI.

2. Time Triggered Architecture

In contrast to a usual event triggered system, where messages are initiated by events independent from the communication system, the Time-Triggered Architecture uses a predefined global TDMA schedule for all communication activities. Each message on the bus is only initiated by the progression of time according to the preplanned message timetable. In a distributed system all TTA nodes are synchronizing themselves to a common global time using the fault-tolerant average algorithm and can therefore communicate without conflicts. TTA nodes connected to a bus using the same global time are called a TTA cluster. An autonomous communication controller decouples the host (application) subsystem from the communication subsystem in both the logical and temporal domain. No control lines or interrupts connect the application processor to the TTP/C controller, the only interface is a dual ported RAM called Communication Network Interface (CNI). On the bus side a bus guardian circuit monitors access to the physical layer. This effectively prevents any fault propagation from a faulty node to the whole system. A basic TTA node consists of the communication controller TTA-C2 and a host CPU system (e.g. Infineon C167CR) running the nodes application. A block diagram for three nodes of a TTP/C cluster is shown in Figure 1. Figure 2 shows the time division bus access of each node.

The communication controller delivers fault-tolerant services according to the TTP/C protocol specification to the host subsystem. Most important features are the synchronized global clock, message transmission, network consistency check and membership service, specification details can be found in /6/. The redundant physical communication layer of TTP/C can be realized as two copper twisted pairs or fiber optics connection.

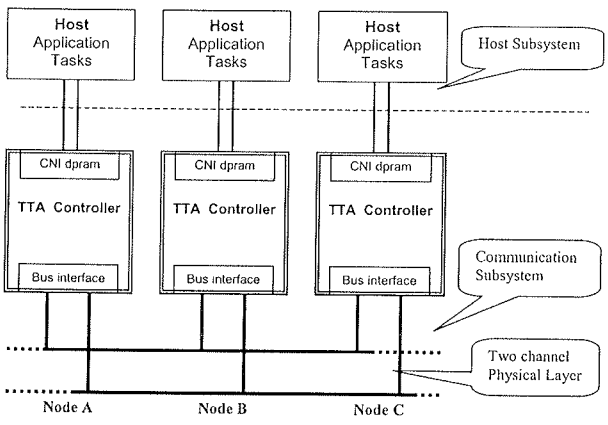

Figure 1: TTA Cluster Architecture

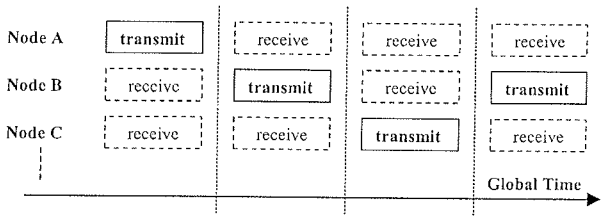

Figure 2: TTA Bus Access Schema

3. Communication Controller Hardware

3.1 Requirements

Discrete component implementations of the TTP/C protocol, using a micro controller together with FPGAs, lack in transmission bandwidth for industrial applications. Performance analysis shows that clock synchronization, data transmission, TTP bus timing, CRC calculation, and the bus guardian have to be implemented in dedicated hardware. Furthermore the integration in a single chip improves reliability and decreases system costs. Due to ongoing protocol improvement activities a programmable solution for the communication controller was preferred. These requirements lead to an implementation with a programmable control unit, supported by several highly specialized units needed for an efficient implementation of the protocol.

3.2 Implementation

Figure 3 shows the block diagram of the TTP/C controller TTA-C2. The Protocol Control Unit (PCU) is the central control circuit. It coordinates the register transfer based communication between the functional units and executes high-level protocol tasks at 40 MHz-clock rate. Functional units are connected through the internal 16-bit wide data bus, which provides a common interface for these units. The following sections give a short overview of the provided functionality /7/.

3.2.1 Protocol Control Unit

The PCU is implemented as an application specific 16-bit instruction processor with three pipeline stages for instruction fetch (IF), instruction decode (DEC) and execute (EX). Stage IF contains the program counter, reads instructions from the instruction memory and passes it to the decode stage. The instruction decode stage generates the control signals for the data bus (to move data between functional units) and the execute stage, and branches are resolved. The third stage contains the ALU, which is able to perform integer addition/subtraction, a wide range of logical and bit manipulation operations and shift / rotate operations.

Due to optimizing the instruction set for protocol execution, an instruction memory size of 16kB is sufficient. This memory is split up into two areas. A 8kB ROM holds the boot code, various safety critical subroutines and the load routines for RAM and Flash. The fast 8kB instruction RAM is

loaded at power-up either from the integrated flash or through the host interface. Special access resolution logic allows moving the instruction RAM content to the build-in CRC unit by normal move instructions executed from the same RAM, enabling a fast permanent firmware self test.

To support efficient protocol execution for the two communication channels, a doubled general-purpose register file supports fast task switching.

A hardware watchdog timer and some error signal traps monitor the controller operation putting the TTA-C2 controller into the fail silent state (means no bus activity) on unexpected events. Diagnosis can then be done via the host interface.

3.2.2 Flash Interface

The protocol firmware of the PCU is located in an integrated flash memory block (32kB) on chip. A dedicated "Flash Unit" controls erase/program/read operations initiated by the PCU and supports special functions to assure data integrity.

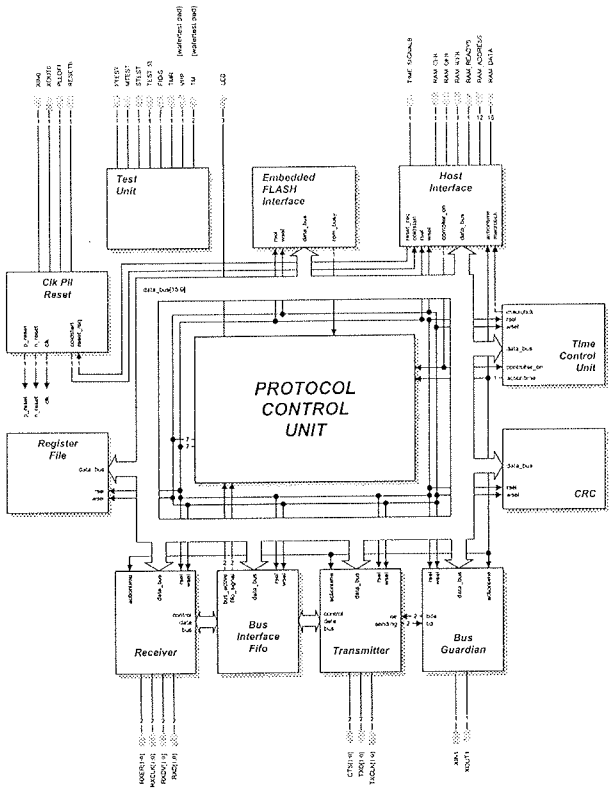

Figure 3: TTA-C2 Controller Architecture

3.2.3 Host Interface

The Host Interface unit implements the Communication Network Interface (CNI) between host subsystem and communication subsystem (Figure 1). It is implemented with a dual-ported RAM (4kB) which holds the messages exchanged among nodes. Also several status and control registers are accessible through this interface.

3.2.4 Time Control Unit

One key function of a time-triggered architecture is the generation of a global synchronized time base. The fault-tolerant average algorithm (FTA) is put in place for clock synchronization. The Time Control Unit enables an efficient implementation of the FTA. It consists of an adjustable counter, which allows fractional division of the system clock.

3.2.5 Transmitter, Receiver

Each controller contains a pair of independent receivers and transmitters with message fifo buffers to handle the physical layer of the two redundant serial communication channels between the TTP/C controllers. For each communication channel two different physical layer interfaces are provided, a low-speed interface for 5 Mbit/s to limit electromagnetic radiation and a high-speed interface for 25 Mbit/s usually using a fiber optics physical layer.

The receiver performs data synchronization and checks for noise, frame format and coding errors and watches the bus to check the temporal validity of transmitted frames.

3.2.6 CRC Unit

The CRC unit supports the calculation of cyclic redundancy checksums for two generator polynomials of 16 and 24 bits length. It allows the concurrent calculation of two checksums, one for each communication channel, in one clock cycle.

3.2.7 Bus Guardian

The bus guardian is an autonomous device that protects the channels from a timing failure of a controller. It contains a local crystal oscillator of its own in order to be able to tolerate a failure of the controller's clock. The bus guardian enforces the bus protection by applying plausibility and timing checks to the signals provided at the bus guardian interface.

3.2.8 Clock Pll, Reset

To meet industrial EMI requirements, the internal 40 MHz clock for the controller may be derived from an external 10 MHz crystal oscillator by a multiplying phase lock loop. Internal power-up reset generation and filter circuitry protects the flash memory contents and assures bus silence until proper startup of the controller.

3.2.9 Test Unit

Safety critical applications of the TTA-C2 controller demand test coverage of almost 100%. To shorten test time three test modes have been implemented: a flash test mode giving full access to the flash memory block; a functional test mode for memory test; and a scan chain mode for logic testing.

In order to test the RAM blocks we decided to make the instruction register accessible from outside. This makes it possible to pipe in arbitrary (memory) instructions and ob-

serve the results on the data bus, which is connected to output pins in functional test mode. Since we can trigger all functions from outside we can also guarantee the basic functionality of each unit.

For the flash-memory-test x/y address, data, control lines and analog charge pump signals are observable to allow fast memory test and characterization during production.

Automated test pattern generation (ATPG) in conjunction with automated scan chain insertion is used for testing standard cell logic parts.

4. HDL Design Flow and Implementation Results

VHDL has been used to model the controller and insert the memory IP blocks. The controller design was functionally verified and synthesized to gate level including the various memory blocks. A complete top-down synthesis strategy was applied, i.e. the design was synthesized as a whole including scan-path insertion and automatic test pattern generation (ATPG). Special attention had to be paid on synchronizing the dataflow through various clock domains and integration of memory-IPs from different vendors.

The floor planning / placement / routing process not only had to consider timing constraints but also placement constraints (separated receiver/transmitter areas, bus guardian with own supply lines, dedicated transmitter-off circuits etc.) to support safety certification. Sign-off simulation with back annotated parameters for worst/best case analysis was done with a mixed language simulator.

All design steps and bug fixes during the design process had to be documented and presented to the customer as well as certification authorities during several design reviews to get approval for usage in X-by-wire applications.

Tabel 1 summarizes technical data and Figure 4 shows the chip layout.

TTP/C protocol	Fully supported
Transmission speed	25 Mbit/sec MII
	5 Mbit/sec MFM
DPRAM to host	2kx16
Instruction ROM	4kx16
Instruction RAM	4kx16
GP Register	96x16
Flash memory	16kx16
Message fifo	2 x 19 words
CRC	16 / 24 bit
Clock frequency	max 40 Mhz
PLL clock	4 x XTAL clock
Analogue flash test	Build in interface
Technology	0.35µ CMOS
Die size	27 mm^2
Package	TQFP 80
Automotive Spec	OK

Table 1: Features and Technical Data

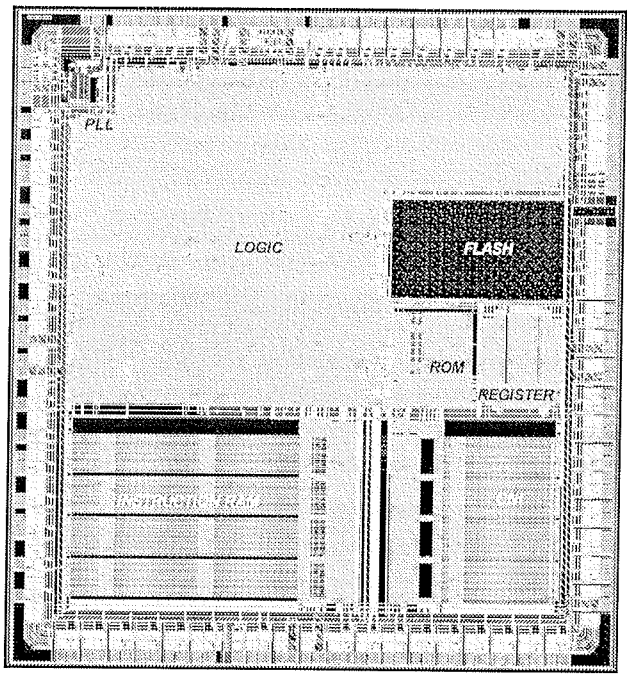

Figure 4: Layout View

5. Modeling Strategy and Verification

The modeling strategy focuses on hardware-software co-development and tool interoperability. It was our aim to do VLSI implementation and system validation in a single environment, i.e. to reuse the models for VLSI synthesis for system programming, functional tests for production test pattern generation, etc.

The RTL VHDL model of the communication controller is the interface for system simulation, verification and protocol programming, hiding the details of the hardware implementation, but the model also serves as reference for the synthesis process.

5.1 Chip Model

For the VLSI implementation the central VHDL model in the design environment is the register transfer level description. This description is restricted to a synthesizable subset of VHDL, all functional blocks are either synthesizable or can be mapped directly onto hardware IP blocks (e.g. RAM, Flash).

5.2 Cluster Model

For the system level development simplified VHDL behavioral models of memory blocks, host controller card and cluster environment are provided to speed up simulation.

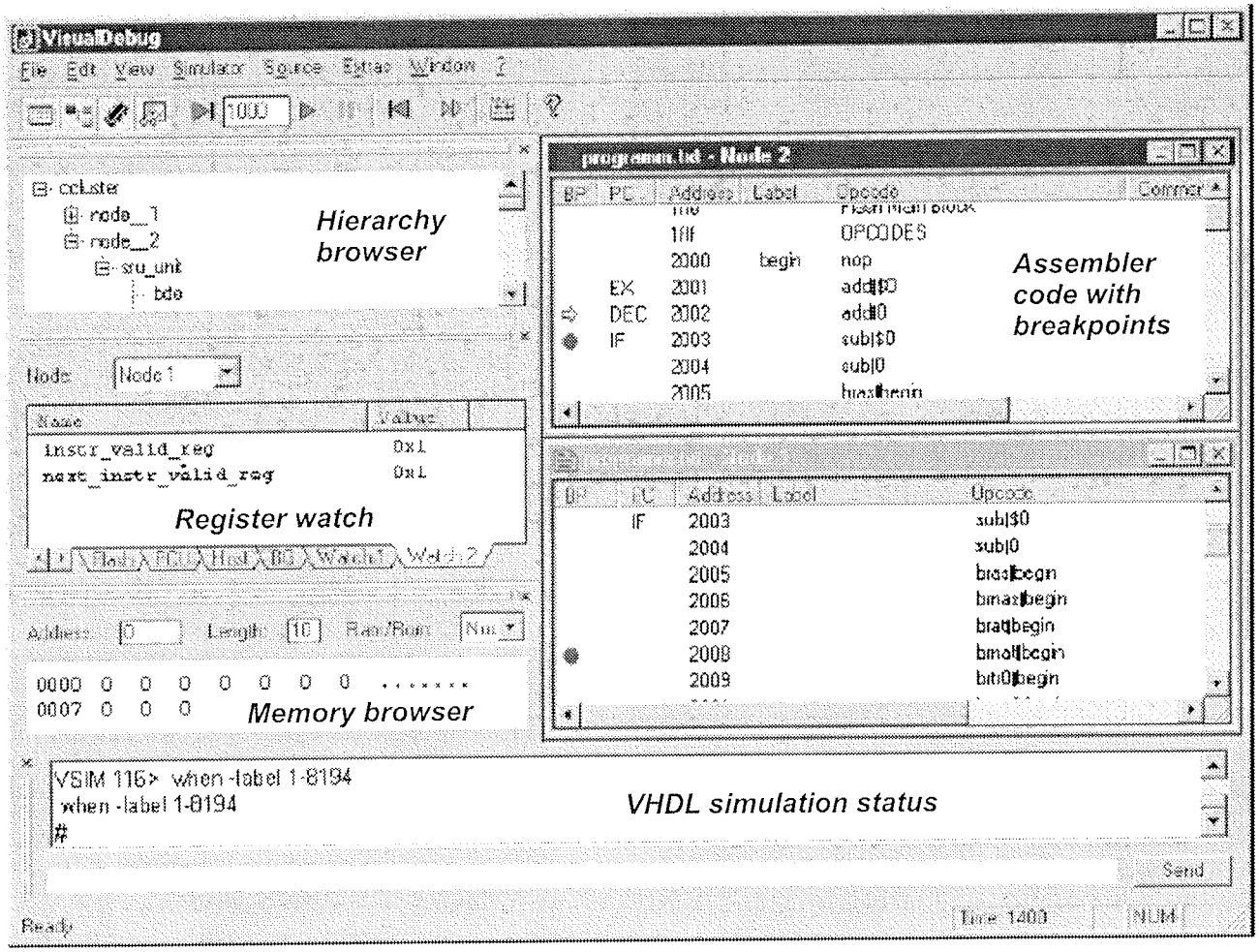

Figure 5: Visual Debug Graphical Debugging Environment

5.3 HW-SW Development

To hide the complexity of the implementation from the protocol programmer a graphical debugging environment (VDEBUG, see Figure 5) and an assembler (TASM) were implemented. VDEBUG allows the protocol programmer to program the Protocol Control Unit and to verify firmware code of an entire cluster of controllers using the same controller model as used for VLSI implementation.

Of course the concept of using only one VHDL model slows down the system simulation environment to some degree compared to using a fast executing C model for protocol programming. But this drawback is more than compensated considering the effort and risk of keeping two models consistent during the whole design cycle. Additionally verification confidence is improved by stressing the same VHDL model code both from a hardware designers view by functional simulation pattern and a system programmers view by executing protocol firmware.

During the software development process macros and procedures, as well as block header and line comments have been extensively used. The resulting firmware has 4000 RISC assembler instructions (from 4096 possible) in total, whereas the ROM functionality has around 1000 RISC assembler instructions. The assembler source code with expanded macros has about 16000 LOC.

6. TTA Cluster and Node Design

In a distributed system the overall system functionality (e.g. brake-by-wire) is divided into several subsystems (e.g. pedal-sensing subsystem, brake-force calculation subsystem, and wheel-speed sensing subsystem). To feature composability and fault-tolerance, the interface between the subsystems needs to be specified in both the time- and value-domain.

6.1 Two-Level Design Approach

At system level, a system integrator (e.g. an automotive company) defines the subsystem functions and specifies the communication interfaces in the time- and value-domain precisely. At the subsystem level, the component supplier has to fulfill exactly these interface specifications but retains complete responsibility on all hardware and software design decisions to implement the desired functionality.

This Two-Level Design Approach /8//9/ is supported by a tool-chain (e.g. TTPtools), which allows the development and seamless integration of different subsystems into one distributed system.

In Figure 6, the upper half reflects the role of the system integrator. The cluster-design is the process of partitioning a system into several independent subsystems and defining the interfaces among each other. The result of the cluster-design process is a cluster-design database. The cluster-design process can be done using the tool TTPplan.

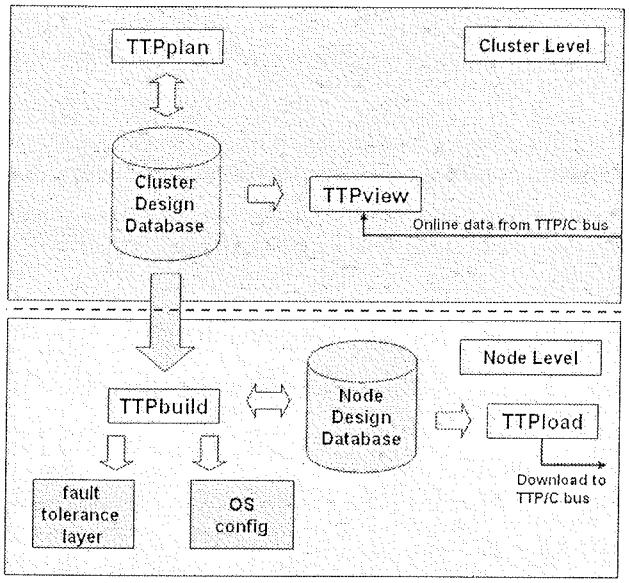

Figure 6: Cluster- and Node-Design

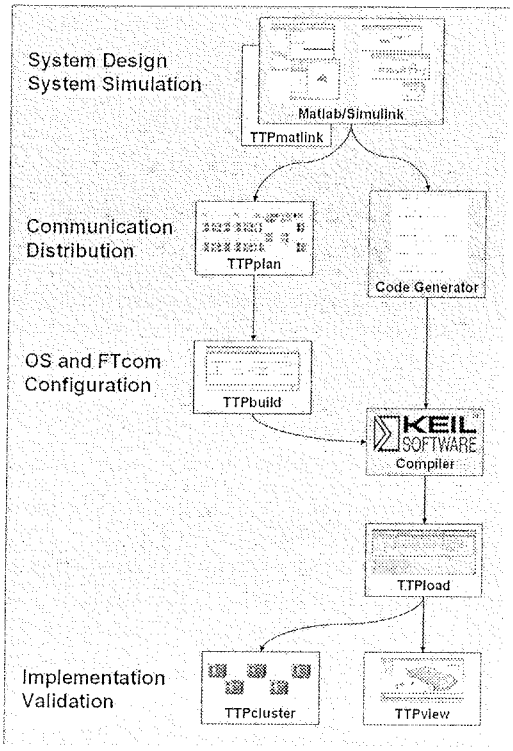

Figure 7: Model Based Design

The lower half of Figure 6 represents the role of the component supplier. At the node-design, individual subsystems are partitioned into software-tasks and the corresponding messages between each other are defined. The results of the node-design are the configuration information for the node's real-time operating system (e.g. TTPos), the automatic generated source-code for the fault tolerance layer (e.g. OSEKtime compliant FTcom layer) and a node-design database. This node-design process can be done using the tool TTPbuild.

6.2 Model Based Design

Figure 7 shows, how the TTPtools can be used in the case of a model based design approach. TTPmatlink is a Matlab/Simulink blockset, which enables a behavioral simulation of the distributed system in the time- and value-domain. It partitions the system into subsystems and it partitions these subsystems into tasks. Furthermore, it defines the messages between the subsystems and the tasks.

For the sake of completeness, the tool TTPload is used to download the communication schedule into the TTA-C2 controller's memory and to download the application files into the host controller's memory. The tool TTPview is used to monitor and debug the distributed system using a specific hardware (monitoring node), which is passively listening to all messages on the communication bus.

6.3 Example TTA Application

A Time-Triggered Car (TTcar) demonstrator (see Figure 8) has been implemented during the last two years at CTI. The TTcar is used for both educational purpose and to show drive-by-wire principles. It consists of 11 subsystems (one of them is triple redundant), 23 tasks and 11 nodes. The nodes are built up using the TTA-C2 communication controller and an Infineon C167CR (20 MHz) host microprocessor.

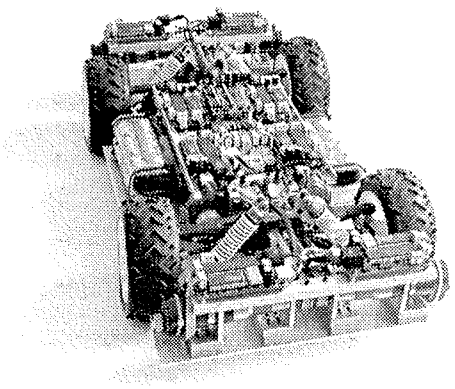

Figure 8: TTcar Application

Figure 9 shows *host1*, which is executing the *funk* and *drive_steer* subsystems. Each of these subsystems consists of two independent tasks. In Figure 10 the communication messages, which are exchanged between the t_funk_receive and the t_drive_steer tasks, are shown.

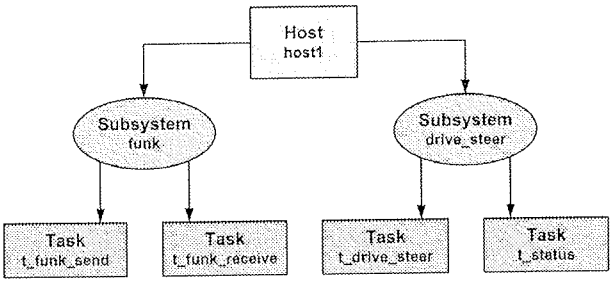

Figure 9: Host, Subsystem, Task Relationship

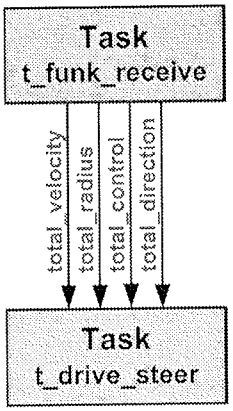

Figure 10: Task and Messages

7. Proof of concept and implementation

For the developers and users of TTA, the proof of concept and implementation is of utmost importance since its main application areas are safety critical and highly dependable systems like airplanes or cars. Requirements, like failure rates in the order of 10^{-9} failures per hour are beyond what can be proofed using methods like testing or simulation. According to this example, more than 100.000 years testing time would be needed. These circumstances lead to the increasing application of formal analysis and formal verification. Both rely on a formal model, which represents the systems properties. In the best case formal verification leads to general and complete statements about the properties of a system as such whereas testing and simulation leads to a probabilistic statement about its expected behavior (see Figure 11).

The combined usage of formal verification and experimental proof finally leads to a trustworthy statement about the reliability and more general dependability of safety critical systems build using the TTA.

7.1 Formal Proof of TTP

The clock synchronization algorithm used in TTA is a modification of the *Welch-Lynch* algorithm. The Welch-Lynch algorithm is characterized by use of the *fault-tolerant midpoint* as its averaging function. It tolerates a single arbi-

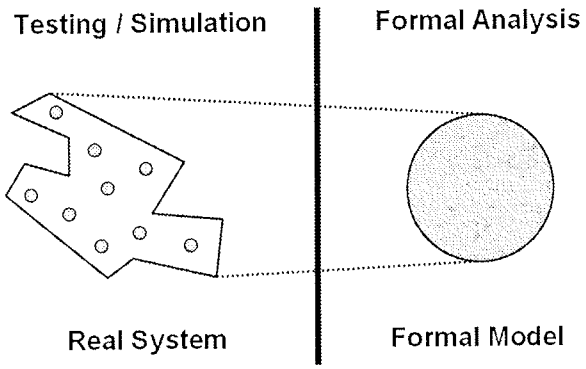

Figure 11: Testing versus Formal Analysis

trary fault whenever equal to or more than four clocks are present. Minor formally verified the Welch-Lynch algorithm and Pfeifer, Schwier and von Henke formally verified its TTA instantiation.

In keeping with the *never give up* philosophy that is appropriate for safety-critical applications, TTA remains operational with less than four good clocks however the applied fault model changes. Rushby extended the fault model used for the formal verification to support not only arbitrary faults /10/.

The clock synchronization algorithm tolerates a single arbitrary fault only. Diagnosing the faulty node and reconfiguring to exclude it tolerate additional faults. The group membership algorithm of TTA, which ensures that each TTA node has a record of which nodes are currently participating correctly in the TTP/C protocol, performs this diagnosis and reconfiguration. Pfeifer formally verified the group membership algorithm in respect to its validity, agreement and self-diagnosis /11/.

The *transmission window timing* guarantees that messages sent by no-fault nodes do not collide on the bus. A faulty node, however, could broadcast at any time - it could even broadcast constantly. This fault is countered by use of a separate *fault containment unit* called *guardian* that has independent knowledge of the time and the schedule: a message sent by one node will reach others only if the guardian agrees that it is indeed scheduled for that time. The transmission windows timing algorithm has been formally verified by Rushby /12/.

7.2 Experimental Proof of TTP

The objectives of the European Commission (EC) sponsored project *Fault Injection for TTA* were to proof the concepts and implementations of the TTA by means of fault-injection experiments thoroughly. Six academic (Chalmers University, CTI, TU Pilsen, TU Prag, TU Valenzia, and TU Wien) and three industrial partners (Motorola, TTTech, and Volvo) have been carrying out experiments on different levels of abstraction (specification, chip, protocol, and node level) for more than two years /13/:

- **Protocol Microcode Fault Injection** (see Figure 12): Systematic injection of illegal instructions and instruction transformation of the TTP/C controllers protocol firmware
- **Heavy-Ion Fault Injection** (see Figure 13): A Californium-256 source causes internal and random faults in the CMOS structure of the TTP/C controller
- **Hardware Implemented Fault Injection**: Faults are injected into I/O pins of the TTP/C controller
- **C-Sim Fault Injection**: Independent C-language reference-implementation of the TTP/C protocol to check the consistency of the TTA specification
- **VHDL-based Fault Injection**: Injects faults into the VHDL-model of the TTP/C controller
- **Software Implemented Fault Injection**: Black-box and white-box fault injections into the code- and data-area of the TTP/C controller

The FIT project showed that there are still improvements, mainly at implementation level, possible and necessary. It also displayed that the main concepts and algorithms of TTA are correct and resist against brutal-force fault-injection experiments. As one of the most important consequences the change from a bus- to a star-topology, using a central guardian, needs to be mentioned. It does not only improve the error detection coverage drastically but it also separates the TTP/C controller and its guardian into two spatially separated fault containment units.

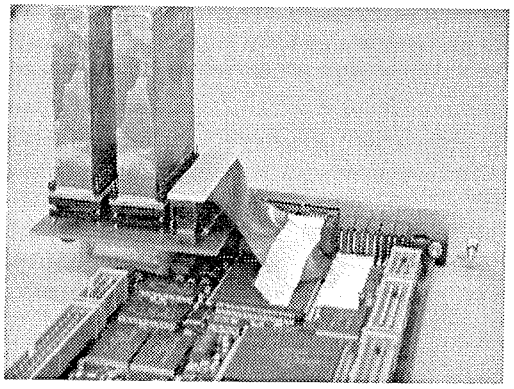

Figure 12: Protocol Microcode FI

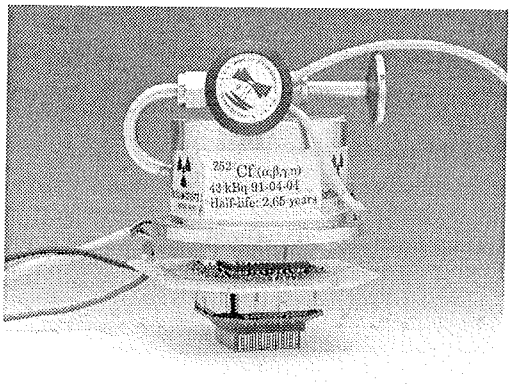

Figure 13: Heavy Ion FI

8. Conclusion

The Time Triggered Protocol (TTP/C) was implemented into the first industrial single-chip communication controller at Carinthia Tech Institute (CTI) in cooperation with austriamicrosystems, Unterpremstätten and TTTech, Vienna.

An application specific RISC core supported by highly specialized peripheral units was evaluated as well suited architecture to fulfill both cost and safety requirements. Design flow and implementation not only had to focus on functionality but also on system approval by safety authorities.

Furthermore, a complete simulation and programming environment useable for chip design as well as for system level design was developed. Related research projects were started in parallel to verify TTA principles and their implementation at all levels of a distributed hierarchical embedded system.

Consequently applying 'best practice' design techniques and combining industry experience with academic research spirit turned out to be a key factor of success in solving such complex design problems. The necessity to cover a widespread range of research subjects on one hand and the fact of limited budget and resources of academic institutions on the other hand inevitably leads to the setup of large multi partner projects. Whereas all partners benefit from such interdisciplinary collaborative work, the professional technical and financial management of such big projects is a new challenge to the academic community.

Resulting controller chips proved to be first time right and were nominated as 'Top Product of the Year' at 2002's SAE (Society of Automotive Engineers) world congress in Detroit. Beside other application fields, the TTA system received approval by US FAA (Federal Aviation Administration) and is going to be used for airplane engine control.

9. References

/1/ Scheidler Ch., Heiner G., Sasse R., Fuchs E., Kopetz H., Temple Ch. "Time-Triggered Architecture (TTA)". *Advances in Information Technologies: The Business Challenge,* IOS Press, pp. 758-765, 1998.

/2/ Kopetz H., Gruensteidl G. „TTP - A Protocol for Fault-Tolerant Real-Time Systems". *IEEE Computer,* 1994, Vol.: 24 (1), (pp. 14-23).

/3/ Kopetz H., Fuchs E., Hexel R., Krüger A., Krug M., Millinger D., Nossal R., Pallierer R., Poledna S., Schedl A., Sprachmann M., Steininger A., Temple Ch. "Specification of the basic TTP/C protocol". *Technical Report 18/96,* Institut für Technische Informatik, Technische Universität Wien, Vienna, Austria, Dec. 1996.

/4/ Sprachmann M., Grünbacher H. „TTA-C1: A Communication Controller Cell for Reuse". *Proceedings FDL'98 Forum on Design Languages,* Lausanne, Switzerland, 1998.

/5/ Sprachmann M. "Modeling a Controller for a Time-Triggered Protocol". *PhD thesis,* Vienna University of Technology, Jan. 1997.

/6/ http://www.ttpforum.org

/7/ Ley M., Grünbacher H. "TTA-C2, a SINGLE CHIP COMMUNICATION CONTROLLER for the TIME-TRIGGERED-PROTOCOL". *Proceedings ICCD 2002,* International Conference on Computer Design 2002, Freiburg, Germany

/8/ H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1997, ISBN 0-7923-9894-7

/9/ S. Poledna, H. Angelow, M. Gluck, I. Smaili, G. Stoger, C. Tanzer, TTP, "Two-Level Design Approach: Tool Support for Composable Fault-Tolerant, Real-Time Systems", *SAE 2000 World Congress*, 2000, Detroit, MI, USA

/10/ J. Rushby, "An Overview of Formal Verification For the TTA", *FTRT-FT02*, Oldenburg, Germany, September 2002

/11/ H. Pfeifer, "Formal verification of TTA group membership algorithm", *Formal Description Techniques and Protocol Specification*, Pisa, Italy, October 2000

/12/ J. Rushby, "Formal verification of transmission window timing for the TTA", *Technical Report*, Computer Science Laboratory, SRI International, CA, March 2001

/13/ http://www.cti.ac.at/fit

Manfred Ley; Christian Madritsch
Fachhochschule Technikum Kärnten,
Carinthia Tech Institute
Europastraße 4, A-9500 Villach, Austria
m.ley@cti.ac.at, c.madritsch@cti.ac.at

Prispelo (Arrived): 15.09.2003 *Sprejeto (Accepted):03.10.2003*