Heikki Saha

Some Methods to Improve Life Cycle Manageability of Low-Volume Mobile Processing Platforms

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB111, at Tampere University of Technology, on the 11th of June 2010, at 12 noon.
Advisor

Professor Jukka Vanhala
Department of Electronics
Tampere University of Technology
Finland

Pre-examiners

Doctor Pentti Vähä
Building Services and Indoor Environment
VTT Technical Research Centre of Finland
Finland

Doctor Martin Merkel
CANopen group
Ixxat Automation GmbH
Germany

Opponents

Doctor Pentti Vähä
Building Services and Indoor Environment
VTT Technical Research Centre of Finland
Finland

Doctor Pasi Tuominen
Wapice Ltd
Finland
Abstract

The life cycles of mobile control electronics are significantly longer than in the mainstream electronics. The low production quantities and high number of product variants together with long life cycles has led to scattered production volumes. Producing low quantities is expensive and inefficient and component obsolescence causes high maintenance costs during the life cycle of each control product. Increasing use of distributed sensors and actuators has improved the functional re-use, but has not solved the maintenance of the application processing platforms. Moreover increased use of networking has pointed out some constraints of the current field buses. AD-interfaces in the application processing platforms form another challenge due to the different application requirements.

Hubs and switches have been widely used in computer networks to solve the network constraints. The use of the same principles has been investigated with the standard high-speed physical layer of CAN. Due to the different bus access scheme, a thorough behavioral physical layer analysis was required. The problematic occurrences found in the theoretical analysis were proved with prototypes and real networks. The same tasks were performed for LIN to find out the effect of the different bus systems. The problem of the AD-interfaces has been solved by implementing and evaluating reconfigurable ADC implemented with the combination of FPGA and FPAA. To get a proper view of the possibilities, two main platforms and two conversion principles were evaluated.

Active CAN hub and switch can significantly move the constraints of current CAN networks. More flexible topologies can be implemented without baudrate restrictions and violating the standard physical layer. Advanced condition monitoring and fault isolation features are provided by both active hub and switch. Because of the differences between CAN and LIN, the physical layer constraints of LIN can be reduced only with a multiport interface embedded into the master node. According to the case implementations, reconfigurable hardware is required to support all the alternative structures.

Mobile processing platforms interface with the analog world and most of the analog outputs are pulse width modulated, which can be supported by
reconfigurable digital hardware, CPLDs and FPGAs. The flexibility of the mixed-signal platforms can also be spread to cover the analog inputs by utilizing the reconfigurable mixed-signal hardware, FPAAs. The concept of reconfigurable ADC has been proved in this thesis with two different AD conversion principles implemented into two existing hardware platforms. Despite the hardware differences, generic ADC-functions were successfully implemented.

Fully customizable standard hardware expects full reconfigurability from the platform. CPLDs and FPGAs have already supported the approach in the digital systems, including network hubs and switches. Using FPAAs together with FPGAs provides the same flexibility to the mixed-signal platforms needed in the mobile control systems. The hardware flexibility enables efficient higher volume production of the control electronics, because the same platforms can be adapted to any application of any system integrator without physical hardware modifications. Furthermore, the flexible hardware can be shared with multiple platform generations.
One person cannot make this kind of large work alone. Thus I am pleased to thank all those people who have helped me during the work, but it is impossible to name them all. Some people have given so much and such significant help that I cannot be without naming them. First, I want to thank my first supervisor Dr. Markku Kivikoski for his support during most of the time. My second supervisor, Dr. Jukka Vanhala, gave the important final support to get the work successfully completed. It has been a privilege to get persons like Dr. Martin Merkel and Dr. Pentti Vähä as reviewers.

Special thanks go to the personnel of TK Engineering Oy, who developed the CAN-switch hardware and software from my ideas and evaluations into a commercially successful product. I would also like to thank Lattice Semiconductor Corp. and especially Jim Krebs, together with Vesa Lauri and Hannu Hänninen from the local distributor companies. This dissertation would not have been completed without their great technical support and component samples. Mikko Talso from Oy MPL Graphics Ab has provided great support by improving some graphics into an understandable form. Dr. Erkki Ahola has given valuable advice regarding the structure for this dissertation, for which many thanks go to him.

K.F. and Maria Dunderberg Testament Foundation provided valuable financial support during finalizing of the dissertation.

Finally many thanks to my wife Nina and my sons Lauri and Leo for their patience during the work.

Tampere, June 2010

Heikki Saha
## Contents

Abstract ...................................................................................................................... 1  
Preface .................................................................................................................... 3  
Contents ................................................................................................................... 5  
List of abbreviations ............................................................................................... 9  
List of symbols ...................................................................................................... 13  
Definitions ............................................................................................................. 15  
1 Introduction ........................................................................................................ 17  
   1.1 Background ................................................................................................ 17  
      1.1.1 Application field ........................................................................... 17  
      1.1.2 Current trends in control system development ......................... 19  
      1.1.3 Major challenges in control system development .......... 21  
   1.2 Scope of the dissertation ........................................................................... 23  
   1.3 Objectives of the dissertation .............................................................. 24  
   1.4 Structure of the dissertation .............................................................. 26  
2 Enhancements to CAN networking ................................................................ 29  
   2.1 Introduction to CAN ............................................................................ 30  
      2.1.1 High-speed physical layer ....................................................... 30  
      2.1.2 Frame types ............................................................................. 32  
      2.1.3 Arbitration .............................................................................. 33  
      2.1.4 Error detection ....................................................................... 34  
      2.1.5 Fault confinement ................................................................... 35  
   2.2 Existing enhancements to CAN ................................................................ 37  
      2.2.1 Application layer protocols ..................................................... 37  
      2.2.2 Alternative physical layers ...................................................... 38  
      2.2.3 Redundancy solutions ............................................................. 40  
      2.2.4 Alternative topologies ............................................................. 41  
   2.3 Active CAN hub for high-speed physical layer ..................................... 43  
      2.3.1 Concept .................................................................................... 44  
      2.3.2 Implementation ....................................................................... 45  
      2.3.3 Evaluation ............................................................................... 51  
   2.4 CAN switch .............................................................................................. 54  
      2.4.1 Concept .................................................................................... 56  
      2.4.2 Implementation ....................................................................... 56  
      2.4.3 Evaluation ............................................................................... 59  
2.5 Summary ....................................................................................................... 70
3  Enhancements to LIN networking .......................................................... 73
  3.1  Introduction to LIN ........................................................................ 74
        3.1.1  Physical layer ............................................................... 74
        3.1.2  Frame types ................................................................. 75
        3.1.3  Communications concept .................................................. 77
        3.1.4  Error detection ............................................................... 78
        3.1.5  Fault confinement ........................................................... 79
  3.2  Existing LIN solutions .................................................................... 80
  3.3  Active hub for LIN ........................................................................ 81
  3.4  Reduced LIN hub ........................................................................... 82
        3.4.1  Concept ........................................................................... 82
        3.4.2  Implementation ............................................................... 83
        3.4.3  Evaluation ....................................................................... 84
  3.5  LIN multiport master interface ...................................................... 87
        3.5.1  Concept ........................................................................... 87
        3.5.2  Implementation ............................................................... 88
        3.5.3  Evaluation ....................................................................... 89
  3.6  Switch in LIN networks .................................................................. 90
  3.7  Summary ....................................................................................... 92

4  Reconfigurable AD interfaces .............................................................. 95
  4.1  Existing mixed-signal solutions ..................................................... 95
  4.2  Introduction to ADC architectures ............................................... 97
        4.2.1  Flash converters ............................................................... 97
        4.2.2  Reference DAC based converters ....................................... 98
        4.2.3  Modulating converters ...................................................... 100
  4.3  ADC performance boosting methods ............................................. 101
  4.4  Configurable general-purpose mixed-signal platforms .................... 102
        4.4.1  AN221E04 ................................................................. 102
        4.4.2  ispPAC20 ................................................................. 103
  4.5  Case A: ΔΣ-modulation ADC ......................................................... 104
        4.5.1  Digital part common to both mixed signal platforms .......... 104
        4.5.2  Mixed signal design in AN221E04 ................................... 106
        4.5.3  Simulation of AN221E04 based design ......................... 107
        4.5.4  Results of AN221E04 based design ................................ 109
        4.5.5  Mixed signal design in ispPAC20 .................................... 110
        4.5.6  Simulation of ispPAC20 based design ............................. 111
        4.5.7  Results of ispPAC20 based design .................................... 112
  4.6  Case B: Successive approximations ADC ...................................... 112
        4.6.1  Digital part for AN221E04 ............................................. 113
        4.6.2  Mixed signal design in AN221E04 ................................... 113
4.6.3 Simulation of AN221E04 based design .........................................114
4.6.4 Results of AN221E04 based design ........................................115
4.6.5 Digital part for ispPAC20 ......................................................115
4.6.6 Mixed signal design in ispPAC20 .............................................116
4.6.7 Results of ispPAC20 based design ............................................117
4.7 Summary .....................................................................................117

5 Volume and life cycle of control electronics ....................................121
5.1 Traditional life cycle and volume models .....................................122
5.2 Technical maintenance methods ................................................125
  5.2.1 Redesign ..............................................................................125
  5.2.2 Careful component selection ................................................128
  5.2.3 Component storing .............................................................129
  5.2.4 Modular design ....................................................................130
5.3 Administrative maintenance methods ..........................................134
  5.3.1 Reactive maintenance ..........................................................134
  5.3.2 Proactive maintenance ..........................................................134
  5.3.3 Generation skipping .............................................................135
  5.3.4 Standardization ....................................................................135
  5.3.5 System architecture .............................................................136
  5.3.6 Product platform strategy ......................................................136
5.4 Extended life cycle model ...........................................................137
5.5 Affect of platform reconfigurability ..............................................139
  5.5.1 Support for multiple applications ..........................................139
  5.5.2 Support for multiple generations ...........................................140
  5.5.3 Resulting enhanced life cycle mode .......................................141
  5.5.4 Resulting enhanced volume model .......................................141
5.6 Summary .....................................................................................142

6 Discussion .....................................................................................145
6.1 Scientific and technical outcome ................................................146
  6.1.1 CAN networking .................................................................146
  6.1.2 LIN networking .................................................................147
  6.1.3 Reconfigurable ADCs ...........................................................148
  6.1.4 Volume and life cycle ...........................................................149
6.2 Commercial outcome .................................................................151
6.3 Limitations ..................................................................................153
6.4 Future research ...........................................................................154
6.5 Summary .....................................................................................156

7 Conclusion .....................................................................................159

References .......................................................................................161
## List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABEL</td>
<td>Advanced Boolean Expression Language</td>
</tr>
<tr>
<td>ACD</td>
<td>Acknowledge delimiter field of a CAN frame</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledge field of a CAN frame</td>
</tr>
<tr>
<td>AD</td>
<td>Analog to Digital</td>
</tr>
<tr>
<td>ADC</td>
<td>AD Converter</td>
</tr>
<tr>
<td>AMS</td>
<td>Analog and Mixed-Signal</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>CAB</td>
<td>Configurable Analog Block</td>
</tr>
<tr>
<td>CAM</td>
<td>Content Addressable Memory</td>
</tr>
<tr>
<td>CAM</td>
<td>Configurable Analog Module</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CF</td>
<td>Compact Flash</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off-The-Shelf</td>
</tr>
<tr>
<td>CPLD</td>
<td>Complex Programmable Logic Device</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CRD</td>
<td>CRC delimiter field of a CAN frame</td>
</tr>
<tr>
<td>CSMA/CA</td>
<td>Carrier Sense Multiple Access/Collision Avoidance</td>
</tr>
<tr>
<td>CSMA/CD+AMP</td>
<td>Carrier Sense Multiple Access/Collision Detection and Arbitration on Message Priority</td>
</tr>
<tr>
<td>CSMA/DCR</td>
<td>Carrier Sense Multiple Access/Dynamic Collision Resolution</td>
</tr>
<tr>
<td>DA</td>
<td>Digital-to-Analog</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-Analog Converter</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DIL</td>
<td>Dual-In-Line</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>DLC</td>
<td>Data Length Code</td>
</tr>
<tr>
<td>DNL</td>
<td>Differential Nonlinearity</td>
</tr>
<tr>
<td>ECU</td>
<td>Electronic Control Unit</td>
</tr>
<tr>
<td>EIA</td>
<td>Electronic Industries Alliance</td>
</tr>
<tr>
<td>EISA</td>
<td>Extended Industry Standard Architecture bus system</td>
</tr>
<tr>
<td>EMC</td>
<td>Electromagnetic Compatibility</td>
</tr>
<tr>
<td>EMI</td>
<td>Electromagnetic Interference</td>
</tr>
<tr>
<td>EOF</td>
<td>End Of Frame</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In, First Out</td>
</tr>
<tr>
<td>FLASH</td>
<td>Non-volatile memory technology</td>
</tr>
<tr>
<td>FPAA</td>
<td>Field Programmable Analog Array</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State-Machine</td>
</tr>
<tr>
<td>FTU</td>
<td>Fault-Treatment Unit</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>HLP</td>
<td>Higher Level Protocol</td>
</tr>
<tr>
<td>HOL</td>
<td>Head Of Line</td>
</tr>
<tr>
<td>hub</td>
<td>Concentrating active networking element in star networks (See also definition of repeater)</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>Hz</td>
<td>Herz (1/s)</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IDE</td>
<td>A field in the CAN frame defining the number of the frame identifier bits</td>
</tr>
<tr>
<td>INL</td>
<td>Integral Nonlinearity</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>ISA</td>
<td>Industry Standard Architecture bus system</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standardization Organization</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>JEDEC</td>
<td>Joint Electron Device Engineering Council, a standardization body of the EIA</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group, a standardized configuration and test interface</td>
</tr>
<tr>
<td>kbps</td>
<td>Kilobits/s</td>
</tr>
<tr>
<td>kHz</td>
<td>Kiloherz</td>
</tr>
<tr>
<td>LDF</td>
<td>LIN Definition File</td>
</tr>
<tr>
<td>LIN</td>
<td>Local Interconnect Network</td>
</tr>
<tr>
<td>lsb</td>
<td>Least significant bit</td>
</tr>
<tr>
<td>LTB</td>
<td>Last-Time Buy</td>
</tr>
<tr>
<td>mA</td>
<td>Milliampere</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>Mbps</td>
<td>Megabits/s</td>
</tr>
<tr>
<td>MMC</td>
<td>Multimedia Card</td>
</tr>
<tr>
<td>NRE</td>
<td>Non-Recurring Engineering</td>
</tr>
<tr>
<td>NRZ</td>
<td>Non-Return to Zero</td>
</tr>
<tr>
<td>OSR</td>
<td>Oversampling Ratio</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>POR</td>
<td>Power-On Reset</td>
</tr>
<tr>
<td>PQ</td>
<td>Priority Queued</td>
</tr>
<tr>
<td>PROP</td>
<td>Propagation segment of a CAN bit time</td>
</tr>
<tr>
<td>PSEG1</td>
<td>Phase segment 1 of a CAN bit time</td>
</tr>
<tr>
<td>PSEG2</td>
<td>Phase segment 2 of a CAN bit time</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>RCM</td>
<td>Reliability-Centered Maintenance</td>
</tr>
<tr>
<td>RES</td>
<td>Reserved control field in CAN frame</td>
</tr>
<tr>
<td>ROI</td>
<td>Return On Investment</td>
</tr>
<tr>
<td>RTR</td>
<td>Remote request field of a CAN frame</td>
</tr>
<tr>
<td>RX</td>
<td>Receive</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>SA</td>
<td>Successive Approximations</td>
</tr>
<tr>
<td>SAR</td>
<td>Successive Approximations Register</td>
</tr>
<tr>
<td>SCI</td>
<td>Serial Communications Interface</td>
</tr>
<tr>
<td>SD</td>
<td>Secure Digital</td>
</tr>
<tr>
<td>SHA</td>
<td>Sample-and-Hold Amplifier</td>
</tr>
<tr>
<td>SJW</td>
<td>Resynchronization jump width</td>
</tr>
<tr>
<td>SoC</td>
<td>System-on-a-Chip</td>
</tr>
<tr>
<td>SOF</td>
<td>Start-of-frame field of a CAN frame</td>
</tr>
<tr>
<td>SPLD</td>
<td>Simple Programmable Logic Device</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>SYNC</td>
<td>Synchronization segment of a CAN bit time</td>
</tr>
<tr>
<td>TX</td>
<td>Transmit</td>
</tr>
<tr>
<td>TQ</td>
<td>Time Quantum</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>VHDL</td>
<td>Very high speed integrated circuits HDL</td>
</tr>
<tr>
<td>VLB</td>
<td>VME Local Bus</td>
</tr>
<tr>
<td>VOQ</td>
<td>Virtual Output Queue</td>
</tr>
<tr>
<td>ΠΔΣ</td>
<td>Parallel ΣΔ</td>
</tr>
</tbody>
</table>
### List of symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>Bit rate</td>
</tr>
<tr>
<td>$F_{\text{HUB}}$</td>
<td>Frequency of hub system clock</td>
</tr>
<tr>
<td>$L_{\text{BUS}}$</td>
<td>Length of bus line</td>
</tr>
<tr>
<td>$R_j$</td>
<td>$j$:th conversion result</td>
</tr>
<tr>
<td>$T_{\text{CHOKE}}$</td>
<td>Signal propagation delay through common-mode choke</td>
</tr>
<tr>
<td>$T_{\text{ECU}}$</td>
<td>Signal propagation delay inside ECU</td>
</tr>
<tr>
<td>$T_{\text{FRAME}}$</td>
<td>Duration of LIN frame</td>
</tr>
<tr>
<td>$T_{\text{HDR}}$</td>
<td>Duration of LIN header</td>
</tr>
<tr>
<td>$T_{\text{HUB}}$</td>
<td>Hub forwarding delay</td>
</tr>
<tr>
<td>$T_{\text{ID}}$</td>
<td>Duration of LIN protected identifier field</td>
</tr>
<tr>
<td>$T_{\text{LOOP}}$</td>
<td>Net propagation delay through transceiver from TX to CAN and from CAN to RX, typically 105ns</td>
</tr>
<tr>
<td>$T_{\text{MPMI}}$</td>
<td>Forwarding delay of LIN multiport master interface</td>
</tr>
<tr>
<td>$T_{\text{PROP}}$</td>
<td>Length of propagation segment of bit-time</td>
</tr>
<tr>
<td>$T_{\text{REFF}}$</td>
<td>Effective response time of a LIN-slave</td>
</tr>
<tr>
<td>$T_{\text{RESP}}$</td>
<td>Response time of a LIN-slave</td>
</tr>
<tr>
<td>$T_{\text{RX}}$</td>
<td>Signal propagation delay through receiver</td>
</tr>
<tr>
<td>$T_{\text{SBRK}}$</td>
<td>Duration of LIN SYNCH_BREAK field</td>
</tr>
<tr>
<td>$T_{\text{SW}}$</td>
<td>Forwarding delay of a switch</td>
</tr>
<tr>
<td>$T_{\text{SYNC}}$</td>
<td>Duration of LIN SYNCH field</td>
</tr>
<tr>
<td>$T_{\text{TX}}$</td>
<td>Signal propagation delay through transmitter</td>
</tr>
<tr>
<td>$x_{av}$</td>
<td>Averaged conversion result</td>
</tr>
<tr>
<td>$x_i$</td>
<td>$i$:th decimation output</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Signal propagation time in (CAN) transmission line, 5ns/m</td>
</tr>
</tbody>
</table>
## Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge</td>
<td>Bridge is a special implementation of the switch with only two network interfaces</td>
</tr>
<tr>
<td>Component</td>
<td>Electronics component can be discrete – i.e. resistor, capacitor, transistor and diode – or integrated circuit – i.e. operational amplifier, standard logic circuit, microcontroller, memory, ADC, FPGA and FPAA.</td>
</tr>
<tr>
<td>Logical platform</td>
<td>A part of a platform composed from logical components, which are HDL-descriptions of components’ behavior. The logical platform design defines the behavior of the physical platform.</td>
</tr>
<tr>
<td>Mobile</td>
<td>Mobile means in this dissertation machine – i.e. forestry, agriculture, earth moving, construction or mining machine, off-road or special vehicle, machine add-on-tool or vehicle superstructure – application field for processing platforms.</td>
</tr>
<tr>
<td>Module</td>
<td>Building block with a defined interface [5.1].</td>
</tr>
<tr>
<td>Nibble</td>
<td>A 4 bits wide data word</td>
</tr>
<tr>
<td>Physical platform</td>
<td>A part of a platform composed of physical components. Traditionally, physical and logical platforms have been closely coupled together. The use of reconfigurable components enables separate physical and logical platforms.</td>
</tr>
<tr>
<td>Platform</td>
<td>Complete set of modules [5.1].</td>
</tr>
<tr>
<td>Processing platform</td>
<td>Processing platform can be explained as a unit capable of performing application-specific computing. It can be implemented with any technology – a general purpose microcontroller, configurable logic or a fixed-function ASIC. A general purpose processing platform may also contain inputs, outputs and communication interfaces.</td>
</tr>
<tr>
<td>Repeater</td>
<td>Repeater is a special implementation of the hub with only two network interfaces</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter begins with an introduction of the application field of this dissertation. The introduction continues by a review of the main trends and challenges of control systems within the defined application field. The scope of this dissertation is defined as research questions after the introduction. Hypothesis is derived from and objectives set based on the research questions. Structure of the dissertation is presented as a last topic of this chapter.

1.1 Background

1.1.1 Application field

This dissertation covers control electronics used in various systems such as mining machines, forestry machines, agricultural equipment, automatic guided vehicles, military vehicles, lifts, construction and earth moving machines. The systems are typically operated in very harsh outdoor or underground environments and are exposed to wide operation temperature and humidity ranges together with different kinds of dirt, mechanical shocks and vibration. The systems are self-moving, mostly diesel operated but there are also battery or AC-supplied ones. Some of the systems may use various power sources depending on the phase of the work cycle. Some examples of typical target systems are presented in Figure 1.1.

Most of the target systems within the scope of this dissertation are part of some bigger processes. Mining and construction machines operate as part of mineral processing and transportation processes; forestry machines operate as part of wood processing processes; agricultural equipment operate as part of food production process and automatic guided vehicles are used as part of various logistic processes. Lifts can be either part of material transportation processes or moving aid for people. Military vehicles are more a part of bigger systems than processes, but the bigger surrounding system sets similar requirements for the vehicles.

When a system is used as a part of a process, all systems share either process data or data for supporting information i.e. for service activities and safety. Furthermore, process dependence has an impact on the availability
of the system. If a critical system is not available, the entire process is stalled. It can be said that every process has something unique. As a consequence of that fact, most systems being used as part of a process will be more or less customized to optimally serve the target process. From the system manufacturer point of view it increases the amount of product variants and respectively decreases the production volumes of individual systems. The processes have become essentially mature and accuracy, availability and efficiency are continuously improved. Due to the slowly developing target processes, the same target systems are used and maintained for a long time in the processes. The total life cycle for individual target systems may exceed 30 years. Development times may be up to 5 years and the minimum required spare part support time 10 years. Based on these values, the maximum life cycle for a single control product generation may be close to 50 years or even exceed it.

A large number of variants, low production volumes of the end user electronic control products, together with long service life [1.1] do not appear to be an attractive business case. This is problematic since control electronics in fact enables the business of the target systems discussed in the context of this dissertation. However, control electronics can become a bigger business if the number of different variants can be reduced and as a consequence the production volumes increased. The required long system life cycle results from the target applications and therefore cannot be changed.
Hydraulics is widely used in typical target systems together with 12V or 24V DC onboard electrics. Hydraulics is used for getting enough amplification power and electrics used for lighting and control system supply. Typical applications for control electronics are e.g. engine and transmission control, control of booms and other type of add-on tools, as well as diagnostics. In most cases control electronics provides an interface to the control application software (SW) between the analog world and digital processing platform, and communication interfaces for system integration. Controller area network (CAN) dominates onboard communication technology in hydraulics control [1.1], whereas the use of local interconnection network (LIN) is increasing in very cost sensitive I/O multiplexing applications [1.2]. Due to the increasing efficiency, accuracy and availability requirements, the use of control electronics in mobile systems is continuously increasing.

1.1.2 Current trends in control system development

Due to the maturity of the target applications, most designs are upgrades or in other ways based on existing designs. Therefore the role of maintenance has been emphasized for existing products and the role of maintainability for new designs. The main approaches in current mobile control system development are systems engineering, platform-based design and distributed system structures. A detailed review of technical maintenance methods and supporting administrative maintenance methods is given in the introductory section of Chapter 5.

Systems engineering concentrates on requirement and feature management but puts a minor effort into methods to maintain underlying hardware (HW). Some generic instructions are presented and assumptions about planned life cycle [1.3] are dangerous in the case of mobile control electronics. It is very difficult to predict life cycles of systems exactly enough to avoid corrective decisions. Selection of long-life components [1.3] is a good guideline. Moreover, utilizing commercial off-the-shelf (COTS) components or components equipped with standardized interfaces [1.3] can avoid a lot of long term maintenance effort. Storing spare parts and repairing broken parts [1.3] are good methods in theory, but in practice it is very challenging to determine the right number of components that need to be stored. Components may also be very difficult to process after a certain number of years in storage. The best guidelines given in systems engineering-related literature are re-using the same components in as many places within a single system as possible and more generally in as many
systems as possible [1.4]. In that way the maintenance effort of those systems is reduced, although the maintenance effort related to single components remains unchanged.

Re-use of components is proposed by systems engineering related literature. Modular design [1.5] [1.6] [1.7] is an approach where invariant parts are isolated into separate re-usable units having standardized interfaces. Modular design and the use of platform modules [1.8] [1.9] can significantly speed up the development of a single product or a product family. To improve the product development and maintenance further, a long term strategy [1.10] is needed. Current mainstream platforms are based on fixed architecture processors and peripherals. Those systems can be partially configured by application SW after manufacturing. Field-programmable gate-arrays (FPGA) are more flexible from the configuration point of view [1.11], but the problem of FPGAs in control applications is their totally digital nature. There are neither analog nor mixed-signal circuit elements built-in. The use of COTS platforms is preferred in low-volume systems, where lower project costs compensate higher unit cost [1.1].

After dividing system functions into re-usable modules – subsystems – with standardized interfaces, the re-usability of subsystems can be improved further by using bus interfaces instead of discrete wiring [1.12] [1.1] [1.2]. Bidirectional and configurable communication interfaces increase the freedom to locate the subsystems in systems. The best known benefit of distribution is the reduced amount of overall wiring [1.13]. Bidirectional communication interfaces everywhere enable system level diagnostics and increase system uptime by reducing troubleshooting time. A not so obvious advantage is that well isolated subsystems can be developed further and still kept backwards compatible by maintaining the backwards compatible interface. Despite possible limits in communication bandwidth, distributed control systems typically provide higher performance operation [1.13] [1.14] [1.1] than centralized systems. Unlike with discrete cabling or wiring, various active networking elements can be used with communication networks to solve physical constraints and increase reliability [1.15] [1.16]. These solutions are covered in detail in the introductions to Chapters 2 and 3.

There are numerous different field buses in active use in the industry. Most of them have originally been used in factory automation [1.17], some others were derived from automotive applications [1.18] [1.19] [1.20] and some, like CAN [1.21] [1.22], are used in almost every industry area. The major trend in recent field bus technology [1.18] [1.17] is increasing the available
Another significant trend is a changeover from linear [1.22] topology to more flexible topologies like star and tree [1.18] [1.17], or even ring or mixed topologies [1.23]. Whilst there are new bus systems penetrating automotive and industrial applications, the use of CAN especially, but also LIN, will still strongly increase in mobile control applications covered by this dissertation.

1.1.3 Major challenges in control system development

While electronics design, maintenance and production have been studied a lot and efficient methods have been found, there is still a lot to be done. Most existing methods apply best for high-volume products, which typically have a shorter service life and support time after the end of active production. The biggest challenge with control electronics HW is that control system components form only a small part of a target system. Therefore it is often difficult to see the control electronics as "real business" from only the electronics point of view. Control electronics is not as big a business as consumer, telecommunication or automotive electronics, but it is an enabling technology for the significant business presented by the overall target systems. Many mobile system integrators are relatively small companies co-operating with relatively small electronics suppliers. That kind of scattered operation has hidden the high overall volumes behind these smaller ones.

The life cycles of typical control electronics products are much longer than in mainstream electronics. It would be preferable to use the same technologies and components, but additional effort has to be put into selecting the longest life components to minimize the maintenance effort caused by component obsolescence. It is challenging to find out the longest life technologies and components, but an even bigger challenge has been the extreme unit cost minimization familiar from high volume electronics. Typically, lowest-cost components are made for high volume products, where a short life cycle is not a problem. High maintenance costs caused by component obsolescence, together with low production volumes produce a long return on investment (ROI) times, which can be reduced only by selecting as generic and long-life components as possible, even if they are more expensive [1.1]. Product platforms reduce the design effort, but the current platforms are based on fixed architectures requiring some maintenance design cycles before product retirement. Work with current fixed architecture platforms is very challenging, because decisions have to be made to minimize the life cycle cost of control product families instead
of minimizing the unit manufacturing cost of a single product. Most current control products contain a significant level of application SW. Every modification to the hardware has inherent consequences for the SW – like processors, peripherals and memory – and requires a thorough testing of all applications written for the modified HW. It is often hard to imagine that re-testing the SW represents biggest cost factor of a HW redesign cycle, and not, for example, the PCB-rework or the production restart.

Because the typical target systems of control electronics are part of various types of processes, customizations and adaptations are needed. It is natural that totally different processes may need different kind of features from control products, but also one type of process may have different kinds of implementations with different requirements. Typical target system manufacturers are relatively small companies which have concentrated on certain types of machines and have implemented their own applications throughout the system independently of each other. There are some more generic products in the market, but the possibilities for adaptations are quite limited. Most of the adaptations are implemented as assembly options but some adaptations may also be made by application software.

The majority of target system manufacturers are manufacturing a relatively low number of machines. The amount of automation within one single machine is not high and after taking different component variants into account, the volume for each control product in each system remains low. As long as volumes are low, the unit costs of automation components remain high and tend to limit the use of automation technology in a system. Low volumes also result in high logistics and manufacturing costs of the products. The challenge is how to stop this vicious circle and enable more cost efficient adaptation, thus enabling an increased use of automation in the target systems.

Most bus technologies traditionally used in machine control systems support topologies limited to linear buses only [1.22]. Target applications are becoming larger and more flexible topologies need to be supported in the future [1.24]. The reliability of the control systems is becoming more important and alternative topologies are needed to reduce the impact of individual failures [1.15]. Some bus systems [1.23] support only a limited number of nodes on a single bus, which becomes a problem as systems become larger. Increased bandwidth will always be required in larger or higher performance control systems [1.18] [1.17]. The major problem of an increased spatial distribution is that COTS components typically support more traditional bus technologies having the mentioned limitations. If
alternative topologies and an increased number of nodes in a single bus can be supported, the use of existing and widely supported technologies can be increased to better serve current developments in the defined application areas.

1.2 Scope of the dissertation

The problems of current control electronics products are small and scattered volumes, the huge gap between life cycles of mainstream and control electronics and also between the digital nature of the most common in-system re-configurable semiconductor platforms and the analog world. The aim of this dissertation is to reduce this gap between mainstream and control electronics by providing methods for more cost-efficient utilization of the most advanced components, design and production technologies used in mass electronics products also in low volume control applications.

In terms of the time domain, the scope of this dissertation covers the entire control product life cycle from requirements acquisition to system retirement. Concentrating on the entire life cycle allows optimizing life cycle characteristics of control electronics, including life cycle cost. A similar approach has been taken in the application domain. Instead of concentrating on single or only a few applications or product lines, the entire application field has been included. This enables better possibilities to find common solutions among system integration companies, product families and applications. In the technology domain, the traditional approach has been to separate platforms according to the supported behavior – different HW has been used for networking, I/O interfacing and application processing. In this dissertation the approach is the opposite – a configurable platform concept supporting alternatively networking, flexible I/O interfacing and application processing has been the target.

Within this dissertation the scope in field bus networking is limited mainly to CAN, because it is the mainstream bus technology in the mobile application area thanks to many higher level protocols running on top of CAN. Application layer gateways are outside the scope and only hubs and switches operating at lower levels have been included in the scope of this dissertation. In addition to CAN, LIN is also covered because it is an emerging technology in low-cost networking and because it introduces some different requirements for the networking elements compared with CAN.
Because digital processing and peripheral integration into system-on-a-chip (SoC) implementations and memory interfacing are well-known technologies in application processing platforms, they are not included in scope of this dissertation. Instead, designing flexible analog-to-digital (AD) interfaces based on a combination of in-system re-configurable digital and mixed-signal HW is one of the main areas of this dissertation.

The research problems can be expressed by the following three questions:

A. How can the use of standard field bus networks be improved to better support higher bit rates, larger spatial system distribution and a higher number of nodes in a single network?

B. How can product owners be enabled to take control over the product life cycle instead of semiconductor component vendors?

C. How can higher production volumes of the physical platforms be enabled to provide better compatibility with the latest component and production technologies used in mass-production?

The following sub-problems can be derived from the previous questions:

1. How to adopt hub- and switch-technologies, successfully used in office and industry networks, into CAN and LIN networks to increase fault-tolerance, physical extension and maximum number of nodes per single network. (*derived from question A*)

2. How to improve the hardware component independency of mixed-signal processing platforms to save maintenance effort when one or more key components become obsolete. (*derived from question B*)

3. How to make the very same physical platform usable as both processing platform and networking element to increase production volumes of the physical platforms. (*derived from question C*)

4. How to make platforms more adaptive to different control application areas and generations to increase production volumes of individual platforms. (*derived from question C*)

1.3 Objectives of the dissertation

A new structure for mobile processing platforms will be presented in this dissertation based on the research of the sub-problems stated earlier. The traditional "single-chip all-in-one platform" approach will be discarded.
Instead, a platform will be divided into physical and logical platforms. The digital and mixed-signal in-system re-configurable components together form the physical platform, which can be adapted to different applications by application specific hardware descriptions – logical platforms. The expected solution can be expressed as the following hypothesis:

**Hypothesis:**

In-system configurable mixed-signal hardware platforms can significantly enhance the characteristics of mobile processing platforms over their required life cycle.

The aim of this dissertation is to review the most common problems of the mobile processing platforms and to develop a semiconductor vendor and product independent mixed-signal platform hardware concept. The hardware concept supports various systems and various generations of the systems. The basic idea is to offer as low dependency as possible between application behavior and HW platform. When a large number of processing platforms and networking devices can be based on the very same physical platform, component logistics and production will be significantly easier and cheaper. In order to test the plausibility of the concept, networking devices and re-configurable ADCs have to be evaluated by case implementations.

To improve the efficiency of distributed architecture in various control systems, existing repeater- or hub- and bridge- or switch-based solutions are reviewed. Based on the review results, improved hub and switch concepts are developed. The main criterion of the improved developments is support of standard physical layers of CAN and LIN networks. Especially in the new hub concept the focus is in the adaptation of the standard physical layer with the hub function and there will be only interfaces for fault-treatment units included, which can be directly adopted from the existing implementations [1.15] [1.16]. The performance target is naturally message loss-free operation at 100% bus load with all standard bit rates.

The analog inputs of processing platforms are those most diverse for different applications. Flexible filtering [1.25] is well known and excluded from the scope of this work. Requirements for conversion accuracy, sample rate, number of parallel sampling and multiplexed inputs are more or less different in all applications. The universality of current platforms suffers from fixed on-chip or off-chip analog-to-digital converters (ADC) providing fixed configurations. To be able to get really generic HW
platforms having analog inputs, in-system re-configurable ADCs are needed. 8-bit resolution is common in typical hydraulic controls in mobile applications and it has been selected as a starting point. Due to the restrictions of current component offerings, only two alternative ADC-architectures can be evaluated. The technical potential of an in-system re-configurable ADC concept is illustrated for the development into commercial products.

The objectives of this dissertation can be summarized as follows:

- To develop fieldbus hubs and switches to increase the physical extension, performance and fault-tolerance of modern distributed control systems based on field buses (answers to sub-problem 1)
- To develop in-system re-configurable mixed-signal HW capable to be an ADC platform and to achieve acceptable accuracy with that platform (answers to sub-problem 2)
- To develop a generic concept enabling the use of the same physical mixed-signal platform, which can be configured as a networking element or processing platform for various applications and generations with alternative logical platforms (answers to sub-problems 3 and 4)

1.4 Structure of the dissertation

The dissertation is organized according to Figure 1.2.

Chapter 2 begins with an introduction of CAN and different methods to build up CAN networks with various topologies. After the introduction, the purposes of hub and switch concepts in CAN networks are presented. After introducing those concepts, a CAN hub solution for ISO high-speed physical layer is presented. The last topic in Chapter 2 is a CAN switch concept and its performance analysis (answers to the sub-problems 1 and 3).

The major differences between CAN and LIN bus systems and limitations of active networking elements in LIN networks are presented in the introduction to Chapter 3. The differences to CAN are reviewed, followed by an explanation of, why an active hub is not a feasible solution in LIN networks. Design, constraints and evaluation of a reduced LIN hub and a multiport LIN master interface are presented after introducing the limitations of LIN networking. The chapter concludes with an explanation
of why switches cannot be used in LIN networks (*answers to the sub-problems 1 and 3*).

A totally new approach to building AD-interfaces into mobile processing platforms is presented in Chapter 4. First the ADC methods supported by configurable mixed-signal devices are introduced. The introduction is followed by implementations of the ADC-architectures supported by the two HW platforms available in the market. Finally, the DC characteristics of the example ADC implementations are presented and compared against each other (*answers to the sub-problem 2*).

An extended life cycle model with generation changes is presented in Chapter 5. Then the results of Chapters 2 and 3 are put together with the system life cycle and volume models to justify how it is possible to share the same physical platforms among communication and processing products. The conclusion of Chapter 4 is needed to justify the possibility to extend the presented concept to also cover mixed-signal applications (*answers to the sub-problem 4*).

The significance of the results from Chapters 2 to 5 is discussed and future work is suggested in Chapter 6.

The conclusions are then presented in Chapter 7.

![Figure 1.2: Structure of the dissertation](image-url)


2 Enhancements to CAN networking

Modern high-performance bus-systems support various topologies. POWERLINK [1.17] for example supports star and tree whereas FlexRay [1.18] supports point-to-point, linear, star, dual star and hybrid systems consisting of all the previously mentioned topologies. Depending on possible application constraints specific topologies may be required. Controller area network (CAN) is a mature and widely used bus system, but its topology is strictly restricted to a linear bus. As both compatibility with the commercial off-the-shelf (COTS) components and increased flexibility with respect to the network topology are needed also in CAN-systems, the existing hub-technology had to be expanded to also cover the high-speed physical layer to fulfill COTS-compatibility. In addition switch technology had to be introduced to support all topologies without further bit rate and physical extension restrictions.

This chapter starts with an introduction to CAN and its applications. Those features of CAN which are essential to the alternative topology solutions are emphasized in the introduction. After the introduction, the most important limitations of CAN and their impact on the CAN communication are discussed and existing solutions are reviewed. While mainly existing physical layer, redundancy and alternative topology solutions are reviewed, some application layer protocol oriented improvements are briefly reviewed to give to readers a good overview of what has already been solved and how some of the solutions can be implemented in hardware (HW). The application layer protocol solutions give some ideas as to why and how the described CAN hub or CAN switch may intrinsically improve the bus behavior.

After the introduction and a review, an active high-speed CAN-hub and a CAN-switch are presented. A detailed description of their theory of operation is followed by evaluation results and a comparison of hub-based and switched CAN-networks with the other existing approaches offering other than linear bus topology. Gateways are operating at the level of the application layer and are thus not within the scope of this dissertation.
2.1 Introduction to CAN

CAN is a field bus defining a communication protocol [1.21] and few alternative physical layers - high-speed [1.22], low-speed fault-tolerant [2.1] and single-wire [2.2]. Originally, CAN has been designed for automotive instrumentation by Robert Bosch GmbH in 1986, but since then it has been widely adopted in both automotive [2.3] [2.4] [2.5] [1.13] [2.6] [2.7] [2.8], industrial [2.3] [2.4] [2.9] [2.5] [2.10] [2.6] [2.11] and building automation applications [2.12]. Machinery and off-road applications incorporate features of both automotive and industrial applications. Typically, CAN is used in control systems [2.13] and I/O multiplexing inside defined application areas.

CAN has been developed to reduce the amount of wires and wiring assembly work in the increasingly large electronic systems in automotive applications [2.5] by offering bidirectional serial communication media between subsystems. Even though the amount of wires has been reduced in the first place, CAN has been designed to operate with low-cost twisted-pair wiring and is thus very inexpensive [2.10]. Most common CAN-physical layers [1.22] [2.1] have good electromagnetic interference (EMI) immunity and low emissions. The relatively simple structure of the protocol allows the use of inexpensive, yet efficient and reliable protocol controllers that can be operated at very low power ratings.

2.1.1 High-speed physical layer

To achieve the best possible EMI immunity against common-mode disturbances, the most common high-speed [1.22] and low-speed fault-tolerant [2.1] CAN physical layers are differential. The potential difference between any two CAN nodes can vary from 0V to +5V to enable operation at a variable current draw of the nodes [1.22]. To minimize signal reflections, only linear bus structures with a bus termination at both ends are permitted [1.22]. The influence of topology variations for the most common high-speed physical layer has been analyzed at a detailed level in [1.24] and [2.13]. The maximum bus length of the high-speed physical layer varies according to the bit rate.

A CAN bus can have two distinct states:

**Recessive:** The bus is in a recessive state when it is idle or when none of the nodes connected to the bus is driving a logical "0" to the
bus. As indicated by the name, a recessive state will be overridden by a dominant bus level.

**Dominant:** The bus is in a dominant state, when one or more nodes are driving a logical "0" to the bus. According to the name, a dominant state sent by one node will at anytime override a recessive state applied by another node.

The logical operation of CAN-bus is wired-AND: If one or more nodes are driving the bus dominant, the resulting bus state is dominant. Thus, the bus state can only be recessive if all nodes are transmitting recessive signal levels. This behavior is utilized by both arbitration and message acknowledgment. A non-return-to-zero (NRZ) bit-encoding is used. To allow receivers to synchronize themselves to the transmission, the so-called bit-stuffing mechanism has been implemented to limit the maximum time between the two consecutive recessive-to-dominant edges that are used for resynchronization. The bit stuffing mechanism adds one complementary bit after 5 consecutive bits with identical signal level. Bit stuffing covers all fields of the CAN frame from start-of-frame (SOF) to cyclic redundancy check (CRC).

![Figure 2.1: CAN bit timing parameters](image)

Each bit-time in a CAN-frame has been divided into several time quanta (TQ), which enables defining the bit timing properties of CAN-nodes according to Figure 2.1. A single bit-time consists of four time segments - synchronization (SYNC) segment, propagation (PROP) segment, phase segments 1 (PSEG1) and 2 (PSEG2). The synchronization segment is needed to compensate synchronization errors between the nodes in the same bus caused by the different clock frequencies and phases. In well-designed
CAN networks, the signal propagates from one end to the other end of the bus and back in the time defined by the PROP segment. Additional phase segments are required for resynchronization of the individual nodes attached to the network. The synchronization jump width (SJW) is the maximum time difference a CAN node can change its bit timing during a resynchronization. Bit rates from 10kbps to 1Mbps are supported.

2.1.2 Frame types

The CAN protocol defines four frame types: Data, remote request (RTR), error and overload frames. Data frames carry the application data. RTR frames are used by data consumers to request transmission of data frames by the producer to the bus. RTR frames never contain data but they define the amount of the requested data instead. Error-frames are used to signal bus errors to other nodes. Overload frames may be used to hold the bus busy during the duration of single overload frame and get some more time to process previously received data frames. The only difference between error and overload frames is their occurrence: Error frames are transmitted immediately after an error has been detected whereas overload frame may be transmitted only immediately after the end-of-frame (EOF) field. The structure of a CAN data frame with an 11-bit identifier field is presented in Figure 2.2 and with a 29-bit identifier field in Figure 2.3.

An error frame consists of either 6 dominant or 6 recessive bits, depending on the operational mode of the node. The operational mode is controlled by the fault confinement described later in this chapter. An active error frame is presented in Figure 2.4 and a passive error frame in Figure 2.5.
2.1.3 Arbitration

As CAN is a multi-master bus system and an arbitration mechanism is required that determines the sequence of frame transmission when two or more nodes start to transmit simultaneously. The literature gives some different expressions for the non-destructive arbitration mechanism of CAN, e.g. carrier sense multiple access/collision avoidance (CSMA/CA) [2.9] [2.7], carrier sense multiple access/collision detection + arbitration on message priority (CSMA/CD+AMP) [2.14] and carrier sense multiple access/dynamic collision resolution (CSMA/DCR) [2.3] [2.4].

Carrier sense applies because each CAN node always monitors the signal level on the bus and starts arbitration only after detecting an idle bus and as each node checks whether the transmitted signal level corresponds to the observed signal level. Arbitration is performed during the arbitration field of a CAN-frame, which includes identifier- and RTR-fields. In the case of a collision, two or more nodes transmit messages from their transmission buffers as long as the bits in the arbitration field are equal. When a node transmits a recessive signal level to the bus, but senses a dominant signal level on the bus, it detects a collision and immediately stops the transmission and starts a new arbitration after the end of the pending transmission. An example of the arbitration in the CAN bus is presented in Figure 2.6.

One should notice that arbitration affects the scheduling of the bus communication profile and that lower priority messages will be delayed until the higher priority messages are transferred [2.14] but higher-priority messages have to wait for the end of a pending lower-priority message.

Figure 2.5: The passive error frame consists of 6 recessive bits enabling error-active nodes to send the active error frame

Figure 2.6: Example of a non-destructive arbitration where node A loses and node B wins the arbitration in the 4th significant bit
Therefore a thorough schedulability analysis should always be performed during network design. Although arbitration is one of the key features of the CAN-protocol, it is not necessarily needed [1.13]. CAN communication can be designed to operate without collisions, i.e. in the following ways:

1. One node in a bus can act as a master having a fixed schedule and requesting all data frames by RTR-frames based on a fixed communication matrix. The communication is fully deterministic and no collisions will occur.

2. The same mechanism may also be implemented in the application layer (by an application layer protocol, like CANopen) and dedicated reference data frames may be used to request data frames.

3. Nodes can also use time-synchronization and transmit only in a fixed time slots when there are no collisions. In this approach one or more data frames are needed for time synchronization between the nodes.

### 2.1.4 Error detection

Because CAN has been designed to operate in harsh environments, thorough error detection mechanisms have been implemented [2.5], [1.21]. The mechanisms cover both data and remote frames. The most challenging occurrence for error detection is inconsistent message omission [2.9]. A frame omission may happen when the transmitting node fails after at least one other node has detected and signalled an error in the last but one bit of the EOF. If the transmitting node fails before retransmission, the node(s) that detected the fault will reject the frame. The errors covered by CAN error detection are categorized as follows:

**Bit-errors:** A bit error is detected if a node detects a dominant signal level on the bus while it has transmitted a recessive signal level outside the arbitration field. If a node detects a recessive signal level after transmission of a dominant level, a bit error is always detected.

**Form-errors:** A form error is detected if a state of a fixed field in the frame is incorrect. Fixed fields comprise identifier size identification field (IDE), reserved control field (RES), CRC delimiter field (CRD) and frame acknowledge delimiter field (ACD)

**Stuff-errors:** A stuff-error is detected when the bit-stuffing rule is broken.
**CRC-errors:** If the CRC computed over CAN data- or remote-frame does not match with the CRC encapsulated in the frame, a CRC error is detected.

**ACK-errors:** Every CAN message shall be acknowledged by at least one receiving node. If a message is not acknowledged, an ACK-error is detected.

After detection of an error in CAN-transmission, automatic retransmission will be performed as depicted in Figure 2.7. This mechanism increasing the message delivery reliability conflicts with timeliness [2.15] being important in control systems. CAN has been designed to provide extremely reliable message transfer by supporting excessive error-detection features and automatic retransmission. However, this extreme delivery reliability may cause late message deliveries. It has been recognized that in most cases it is less harmful to lose a message than get it late [2.7], [2.10].

![Figure 2.7: Form error, following error-frame and an automatic retransmission of the original frame](image)

An additional challenge for error detection is inconsistent message duplication. The receiving nodes perform error checking until the last but one bit and in the case of a bit-error in the last bit, they accept the frame as valid. The transmitting node performs an error-check until the last bit and in the case of a bit error in the last bit, the transmitting node will retransmit the erroneous frame and the receivers may receive an inconsistent duplicate [2.11].

### 2.1.5 Fault confinement

The fault-confinement mechanism is a top-level mechanism for error detection and its main target is to reduce the transmission capability of a
faulty node to prevent it blocking the operation of the entire system. It is based on two error counters – one for reception errors and another for transmission errors, and a state machine according to Figure 2.8 changing states according to certain counter value combinations. Fault confinement states are defined in [2.8] [1.21]:

**Error-active:** Normal operation mode with full-featured bus-access and active error-frame transmission capability

**Error-warn:** Equals error-active, but being a separate state it enables warning to the host processor. Not defined as a state in [1.21], but counter value ranges above error-warn level indicate serious problems.

**Error-passive:** Delayed arbitration start and passive error frame transmission try to minimize the impact of a faulty node on the rest of the system.

**Bus-off:** No activity until the bus is idle for a defined time interval or system reset.

![Fault-confinement state-machine of a CAN-node](image)

Figure 2.8: Fault-confinement state-machine of a CAN-node

In the error-passive state, a node starts arbitration after a suspend transmission field following intermission, after which error-active nodes can start arbitration. If a possibly faulty device is in error-passive state, this mechanism will guarantee that correctly operating nodes can override possibly faulty nodes. Figures 2.9 and 2.10 illustrate the additional suspend
transmission field in 11- and 29-bit CAN frames sent by error passive nodes.

<table>
<thead>
<tr>
<th>SOF</th>
<th>11-BIT IDENTIFIER</th>
<th>RTR</th>
<th>DLC</th>
<th>0..8 DATA BYTES</th>
<th>15-BIT CRC</th>
<th>ACK</th>
<th>ACK</th>
<th>EOF</th>
<th>INTER-MISSION</th>
<th>SUSPEND TRANSMISSION</th>
<th>BUS IDLE</th>
</tr>
</thead>
</table>

Figure 2.9: Structure of a data frame with 11-bit identifier when transmitted by error-passive node

<table>
<thead>
<tr>
<th>SOF</th>
<th>11-BIT IDENTIFIER</th>
<th>RTR</th>
<th>18-BIT ID</th>
<th>RTR</th>
<th>DLC</th>
<th>0..8 DATA BYTES</th>
<th>15-BIT CRC</th>
<th>ACK</th>
<th>ACK</th>
<th>EOF</th>
<th>INTER-MISSION</th>
<th>SUSPEND TRANSMISSION</th>
<th>BUS IDLE</th>
</tr>
</thead>
</table>

Figure 2.10: Structure of a data frame with 29-bit identifier when transmitted by error-passive node

2.2 Existing enhancements to CAN

2.2.1 Application layer protocols

Protocol solutions are not within the scope of this dissertation, but a review of the most common ones supports understanding of the behavior of CAN systems and the positive impact of the active networking elements. Protocol improvements are independent of standard physical layers and bus topologies. They also resolve the best-known deficiencies of the CAN communication. Due to some deficiencies, like the lack of the atomic broadcast and inconsistent message omission, adding active networking elements does not introduce additional fault types to be covered by the application layer protocols. Some of the application layer improvements are already supported by the mainstream application layer protocols.

One of the best-known deficiencies of CAN is the lack of an atomic broadcast. Different CAN nodes may receive the same transmission differently. Based on the literature, the most common scenarios are inconsistent message omission and duplication [2.9] [2.11]. Application-layer protocols such as CANopen provide basic services – such as message timeout monitoring, toggle bits and device state-machines, which can be used to detect and resolve these fault conditions. The other well-known problem is the high dependability of CAN, which causes timeliness problems in real-time signalling under harsh electrical environments. Every time a CAN-message has to be retransmitted because of an error, the original transmission consumes time until the occurrence of the error: error signalling consumes time and finally the automatic retransmission takes more time. The total delivery time may exceed the assumed deadline for the signal update.
There are many solutions around the issues of reliability and timeliness. Separate categorized protocols for timely and reliable transmissions are presented in [2.16]. The key issue is that only time critical information is transmitted using a dedicated real-time protocol and non-time-critical transmissions use standard CAN transmission policy. Also [2.10] suggests that it is better to prevent message transmission if it is not possible to meet the specified deadline. A transmission buffer with integrated timestamp management has been introduced in [2.6]. In [2.7] it is recommended to remove unsent messages being late from the message transmission buffer of CAN-controller. The defined approach tries to minimize the effect of delays caused by automatic retransmissions.

By default higher priority messages win the arbitration and pass through the CAN bus before lower priority messages. The literature [2.17] presents a protocol-based method which modifies purely priority-based transmission towards a round-robin approach. The idea is to lower the priority of a message immediately after the transmission so that the other unsent messages can reach the bus before the next transmission of the recently transmitted higher priority message. The target has been to prevent a single high-priority message negatively impacting on the transmissions of lower priority messages. In modern application layer protocols like CANopen, almost the same behavior can be reached by controlling the message transmissions with the configurable communications parameters of the protocols.

### 2.2.2 Alternative physical layers

While this dissertation concentrates on the most widely used ISO high-speed physical layer, some alternative physical layers are reviewed to give a better understanding of the field. A review also confirms the most typical physical layer challenges solved by the alternative physical layer approaches. Moreover, some fault treatment units have been introduced which can be directly re-used in the hub and switch concepts designed in the research work covered by this dissertation.

Although the high-speed physical layer utilizes differential signalling, there are still applications with high EMI disturbances such that traditional wiring does not offer the best possible immunity against that high disturbance. In some applications the high-speed physical cannot operate due to high EMI. There are a few concepts presented in the literature [2.18], [1.16], [2.19] utilizing a low-cost plastic optical fibre instead of traditional copper wiring or cabling. According to [2.18], passive optical multi-drop solutions suffer
from power drops. The active optical solutions are hub-based, where only a single node can be connected to each port. The latter solutions are reviewed in more detail in the alternative topologies section of this chapter.

The literature [2.18] presents one of the first experiments around an optical physical layer and especially the first optical hub for star-topology CAN networks. The optical physical layer is similar to the use of galvanic isolation between the CAN-controller and the transceiver, and it expects some kind of wired-AND function implementation in either a passive hub, the transmission medium itself or in an active hub with optical interfaces for the nodes. Galvanic isolation provides an excellent protection against unwanted ground loops, failures caused by the ground potential differences between the nodes or high electromagnetic radiation alongside the cabling path.

While the solutions presented in [2.18] were totally optical, [2.19] and [1.16] have introduced optical full-duplex links between all nodes and a hub, on which the wired-AND function has been implemented. The optical links in both approaches behave like optocouplers between the CAN-controller and the transceiver. In [1.16] the hub has been implemented as logic, but a short CAN-bus in the hub as presented by [2.19]. Hub implementations presented in both [1.16] and [2.19] also offer a standard high-speed physical layer port in addition to the optical ports.

A full-duplex connection between the hub and devices has also been presented in [1.15]. Instead of optical fibre, private standard ISO high-speed links have been used for both upstream and downstream links between the hub and each node. The wired-AND CAN emulation has been implemented inside the hub and the individual links are still utilizing the standard high-speed physical layer components and transmission media.

WideCAN [2.3] splits the physical layer into arbitration and propagation buses to enable a hierarchical arbitration in a hierarchical tree topology. The concept is not compatible with the standard CAN-networks because of the modified arbitration scheme. To compensate for the bandwidth loss of the slower arbitration, the CAN-communication has been partially modified. Because the critical timing issues cover only the arbitration and acknowledge fields, an increased data rate has been introduced for the payload data and CRC.
2.2.3 Redundancy solutions

According to long experience on bus systems, damage to bus cables, wires and connectors form the biggest single failure category. Reference [2.20] clearly confirms this understanding. Therefore redundancy solutions are reviewed. Even if linear bus topology is used, different kinds of redundancy solutions are needed in some cases to guarantee a high enough fault tolerance but still keep the compliance with the mainstream component offering [2.20].

Adding redundancy is one way to increase the fault-tolerance of CAN physical, medium access control (MAC) or link layer operation. The simplest method to implement parallel CAN-links is to just transmit everything into parallel links and receive the result of a logical AND of those links [2.21]. That will increase fault tolerance, but does not provide any additional diagnostics without any link-specific fault treatment modules. Reference [2.21] also emphasizes the importance of physically separated link cabling routes to minimize the effect of single point failure. The most obvious risk of simple redundancy is a corruption of the bit time when propagation time differs between the parallel links. To prevent the bit timing corruption that is possible with combined parallel links, an arbitration logic has been introduced in [2.20]. The arbitration circuit is capable of selecting one out of N redundant physical links as a signal source. Additionally, arbitration logic can provide the indication of a selected link [2.20] and various fault-treatment functions [2.21]. Together with the message reception status the active link information can be used to determine the condition of the link. The most obvious drawback is a need for additional logic when used with the normal microcontrollers [2.21]. Another significant drawback of simple redundancy logic is the risk of a corrupted synchronization caused by the link arbitration logic [2.21].

In addition to the physical layer redundancy, two [2.20] [2.23] or three [2.22] parallel CAN-controllers have also been proposed. Parallel CAN-controllers offer tolerance against the failures of the controllers [2.22] and less error-prone behavior in the case of physical layer errors. Parallel CAN-controllers greatly increase the system complexity and cost, with respect to both hardware and software. A special arbitration is needed to keep the error control of the parallel CAN-controllers synchronized [2.22]. It shall be also guaranteed, that only single transmissions are forwarded to the application software [2.20] [2.22]. The most complex approach with both two parallel physical links and two parallel CAN-controllers has been
presented in [2.23]. The solution tolerates both temporary and permanent physical layer and MAC-layer failures, even simultaneously. The main disadvantage besides the software redundancy manager is the very complex arbitration logic required between the parallel physical links and parallel CAN-controllers.

A replicated star topology with a modified physical layer has been introduced in [2.24]. In the ReCANcentrate architecture the hub with an extensive set of fault-treatment functions can be duplicated. A duplicated replication link has been introduced between the hubs to guarantee fault-tolerant interconnection between the parallel hubs. The results presented in [2.25] prove that any fault in any single link does not affect the behavior of the rest of the system. Neither fault in either hub affects the system behavior. The main problem of the proposed approach is that each node will require a special duplicated physical layer interface not supported by any COTS device in the market.

The reviewed redundancy solutions concentrated on adding redundancy against the failures in the physical transmission media and in the CAN-controllers but not in the application layer communication failures. In some communication systems, like FlexRay, so-called bus-guardians are used to prevent unintended transmissions reaching the communication media and breaking the communication schedule. There are two kinds of bus guardians; local and central. A node-local bus guardian is a simple functionality to cover rough transmission timing-related faults caused by the MAC-controller [2.26] [2.27]. In the highest performance solution, an independent bus-guardian equals a redundant MAC controller enabling or disabling the output of the primary controller. The more independent the bus guardian is from the rest of the node, the more it will increase the reliability. Potential dependencies with the rest of the node are power-supply, clock and initialization data stored in the same memory. The more separate the bus guardian is from the rest of the node, the more it will increase the complexity of the node. A bus-guardian can also be implemented as part of a hub and be central to the system [2.28].

2.2.4 Alternative topologies

Because of the tightly restricted linear bus topology of CAN, there has been a need for other topologies to better support the various topology requirements of the real systems. The constraints of the high-speed physical layer are simulated and tested in [1.24]. The results prove that at 500kbps and higher bit rates the allowed topology deviations are so small, that it is
not possible to reach alternative topologies without lowering the signal quality. Another reason for the use of the alternative topologies is a need to increase the fault-tolerance of the systems or subsystems by minimizing the impact of a failure in a single point in the system [1.15] [2.29] [2.24] [2.25].

A basic idea to increase a system's reliability by moving from bus to star has already been adopted in the Ethernet networks [2.3] during the last decade. With an active hub or a switch, any node can be connected to or disconnected from the operating bus without too much disturbing the rest of the system. A comparable behavior has been implemented and introduced by [1.16] and [2.29] in their hub-based concepts. Star and tree bus topologies have also been promoted by different companies offering repeater devices for the CAN high-speed physical layer. Based on review and evaluation of some of them, the behavior of those devices connecting two physical buses into single logical one has been unstable and therefore they are not covered in detail.

A star-topology approach, where only one node can be connected to each port has been presented in [1.16]. Hot-plugging is the only supported fault-treatment feature - an unconnected port can be disconnected and reintegrated again if a valid connection is found. While this solution is mostly optical, it supports one connection to the standard high-speed CAN bus. To reach a required performance level, a complex programmable logic device (CPLD) has been used as a platform instead of a microcontroller.

The CANcentrate concept [2.29] implements a star-topology, where only one node can be connected into each port of the hub. A full-featured fault-treatment module has been implemented for each port to enable the hub to automatically detect a faulty node and disconnect it from the network. Based on the fault-treatment information, the hub can also disconnect a faulty node without communication errors and also reintegrate a node after the faulty node has been recovered. A field-programmable gate array (FPGA) based implementation [1.15] has been presented, because the hub has to be so fast that SW-based implementations do not apply and the required special functional modules are not available as microcontroller peripherals.

The so-called WideCAN [2.3] enables a use of star and tree topologies, but it will increase the signal propagation time during the arbitration segment. WideCAN compensates for the extra delay during the arbitration field by over-clocking the data field and offering up to 80 data bytes per frame. The concept is attractive, but it does not use COTS components and is therefore
not practical in real applications. It is however a good example of how efficiently arbitration and in-frame acknowledge of CAN limit the possibilities to utilize alternative topologies.

2.3 Active CAN hub for high-speed physical layer

The bus topology of the high-speed CAN physical layer has been restricted to the linear structure and only minimal stub lines are allowed by the standard [1.22]. Based on both simulations and measurements of the real system, stub lines are increasingly reduced at bit rates of 300-400kbps and higher [1.24]. None of the existing alternative topology solutions [2.18] [2.19] [1.15] [1.16] [2.3] is 100% compatible with the existing COTS nodes in the market. While a star-topology presents a maximum amount of cabling [2.30] when compared with the linear topology, star topology networks may still be required for some reasons [2.31]:

- A star-topology may better serve the physical layout of some target systems than linear topology
- Bus cables with both power supply and bus wires are used with a star-shaped power supply scheme, which minimizes the power supply interferences between the nodes
- A star-topology is required to increase the system's reliability by reducing the impact of a single bus failure

Based on the above-mentioned requirements, an active CAN-hub supporting high-speed physical layer has been developed. Along with the CAN high-speed physical layer compatibility, all existing CAN-equipped transducers, actuators and electronic control units (ECU) can be utilized in the system implementations. Because all ports of the hub are fully compatible with the high-speed physical layer, either single or multiple nodes can be connected to each port. In most cases it will ease the design of systems when compared with the one node per branch limitation.

A totally new concept had to be developed instead of re-using the existing repeater designs because their operation has been found to be unstable. Only an interface for port enable and disable has been implemented, because fault-treatment modules have been studied and evaluated a lot and the presented fault-treatment [1.15], [1.16] and bus-guardian modules can be directly utilized in high-speed CAN hub. Different physical layer support does not prevent a full utilization of those modules, when needed.
2.3.1 Concept

The active CAN hub concept was initially developed during investigation of the reasons for the unstable operation of COTS repeater devices. After finding out the required operation principle, a stable multiport hub design was found. The active high-speed CAN hub has originally been presented in [2.32], where a preliminary behavioral model has been used to point out the timing and transmission direction determination challenges with the active hub concept. The detailed analysis and design presented in this chapter has been published in [2.31]. A hub is a simple device not capable of filtering CAN messages. It operates at the physical signalling layer. It handles collisions just like standard CAN-bus but in two phases, first locally for each branch and then between all branches. Arbitration and in-frame acknowledge are integral parts of the CAN-communication, which requires nodes to be synchronized within one bit-time. A hub provides practically unchanged node synchronization, but it carefully cross-checks and distributes dominant states between bus branches. The bi-directional dominant state distribution shall be performed carefully to prevent an infinite loop-back and total stuck-at-dominant fault of the whole bus with multiple physical branches.

The hub operation can be divided into three cases. The first case is the simplest one, where only one node in one branch is transmitting. There is nothing strange – the received dominant state from one branch is distributed into other branches and the dominant states sent by the hub itself are not re-distributed. The second case, first introduced in [2.32], can occur e.g. during arbitration, where nodes from different branches send identifier bits one-by-one and the consecutive bits may be in the opposite state. In the acknowledge field overlapping dominant states may also appear because of slight synchronization differences between nodes. In the bit transitions, overlapping dominant states may occur because of slightly different clock frequencies and phases between the nodes. A hub detects when a node in one branch releases the bus from dominant to recessive state and when a node in another branch still keeps the bus dominant after the hub has released the bus to recessive [2.32]. The recognition of a remaining dominant state after a removed distributed dominant state is time-critical. On the one hand, recognition shall be performed as soon as possible, but on the other hand the recognition can be performed just after the loop-delay of the whole bus branch, when the state of the bus branch has been settled. In the third case, nodes in two or more branches simultaneously drive the bus to dominant [2.32]. In that case, the hub shall not distribute simultaneously
received dominant states to the branches being driven to the dominant state. The hub still has to distribute the dominant state into branches being recessive.

All inputs shall always be active to enable continuous operation of the optional fault-treatment units. Those units are able to automatically disconnect a faulty bus branch and re-integrate it after the fault has disappeared [2.15] [2.16]. If a port needs to be disabled, only transmissions to the disabled port shall be prevented by forcing the port transmission signal to the recessive state.

2.3.2 Implementation

A block diagram of an active high-speed hub is presented in Figure 2.11 [2.32]. There are separate reception and transmission state-machines for every bus branch. Every reception state-machine outputs a dominant state distribution request signal to every transmission state-machine. For simplicity the branch disable signals are not shown.

![Block diagram of a three-port active CAN hub](image)

Figure 2.11: Block diagram of a three-port active CAN hub

The reception state-machine in Figure 2.12 [2.31] is always active, because it provides state information for the optional fault-treatment units presented e.g. in [1.15], [1.16]. Its main responsibility is to detect whether the hub itself or an external node is driving the bus to the dominant state. In case an external node is driving the bus (dominant state is received, but not transmitted), the reception state-machine distributes the state towards the
other ports and if a hub is driving the bus itself (dominant state transmitted before reception), the dominant state will be ignored to prevent infinite loop-back.

**Figure 2.12: CAN hub reception state-machine of a port C**

**IDLE**: Idle state is a default state after the power-on-reset (POR). It is active as long as recessive state is received from the corresponding bus or when a port is disabled.

**PASS**: Pass state is entered when the hub itself distributes a dominant state to the corresponding port. This state prevents infinite loop-back of the dominant state.

**WAIT1**: Wait1 state is needed for waiting up to the loop-delay after determining the remaining bus state after the hub has stopped the dominant state transmission. When this state is entered, a dedicated timer is activated and at a terminal count of the timer, the next state will be entered.

**ECHO**: In echo state the dominant state received from the appropriate bus interface will be distributed to the other interfaces.

The transmission state-machine in Figure 2.13 [2.31] is responsible for combining incoming distributed dominant states to a single TX-signal. It also keeps the TX-output constantly recessive, if the corresponding port is
disabled. Blocking of the distributed dominant states from other disabled ports is handled by the transmission state-machine.

![CAN hub transmission state-machine of a port C](image)

**SREC:** SREC state transmits recessive state and is active when other distributed states are passive and reset, or local distribution or any disable is active.

**SDOM:** SDOM state transmits dominant state and is active only when at least one of the other distributed states are active, but local state and local disable are passive.

![CAN bus simulation model state machine for bus C](image)

To be able to easily simulate the hub behavior as state-diagram, also CAN-buses had to be modeled as state-machines as presented in Figure 2.14.
The state-machine needs a timer to simulate the CAN bus loop delay, with which it forms a model of a single CAN-bus. The simulation has been performed in two phases: first in the StateCAD program as state diagrams and then as full-featured ABEL hardware description language (HDL) modules in the logic simulator. State diagrams have automatically been transformed to ABEL HDL by the StateCAD program.

CAN-bus simulation state-machine description:

**SREC:** SREC state drives the virtual bus into the recessive state. It is active after reset and when the virtual bus has propagated the recessive state based on a received signal and terminal count of the loop-delay timer.

**RTOD:** RTOD is a wait state emulating loop-delay of the bus in transition from the recessive to dominant state. A loop-delay timer controls the duration of this state.

**SDOM:** SDOM state drives the virtual bus into the dominant state. It is active after the virtual bus has propagated the dominant state based on a received signal and the terminal count of the loop-delay timer.

**DTOR:** DTOR is a wait state emulating the loop-delay of the bus in transition from dominant to recessive state. A loop-delay timer controls the duration of this state.

### Table 2.1: Typical bit timing parameters according to high-speed bit rates

<table>
<thead>
<tr>
<th>Bit rate</th>
<th>Bit time</th>
<th>Time quantum</th>
<th>Propagation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Mbps</td>
<td>1µs</td>
<td>125ns</td>
<td>4 TQ</td>
</tr>
<tr>
<td>800kbps</td>
<td>1.25µs</td>
<td>125ns</td>
<td>7 TQ</td>
</tr>
<tr>
<td>500kbps</td>
<td>2µs</td>
<td>125ns</td>
<td>11 TQ</td>
</tr>
<tr>
<td>250kbps</td>
<td>4µs</td>
<td>250ns</td>
<td>11 TQ</td>
</tr>
</tbody>
</table>

Table 2.1 presents the typical values of a bit-time and a time quantum and the worst-case values of the maximum signal propagation time for the standardized bit rates used with the high-speed physical layer [2.33]. For simplicity, an 8 TQ long bit-time and a 4 TQ long propagation time has been used for all bit rates. The listed values are needed in the following hub timing analysis [2.31].

The propagation delay internal to an ECU is defined as:
\[ T_{ECU} = T_{TX} + T_{RX} + T_{LOGIC} + T_{CHOKE} \] (2.1)

The ECU delay equals the transceiver loop delay when extra logic and common-mode choke are not used:

\[ T_{ECU} = T_{TX} + T_{RX} = T_{LOOP} \] (2.2)

The signal propagation delay in a bus line can be computed from the ECU delay, signal propagation speed \( \lambda \) and bus length \( L_{BUS} \):

\[ T_{PROP} = 2 \cdot (T_{ECU} + \lambda \cdot L_{BUS}) \] (2.3)

The total propagation delay for single branch is the sum of the signal propagation time in the bus and the hub processing delay:

\[ T_{PROP} = 2 \cdot T_{ECU} + \lambda \cdot L_{BUS} + 2 \cdot T_{HUB} \] (2.4)

According to Table 2.1, a propagation delay of 4 TQ can be spent on the signal propagation. 1 TQ is reserved for node synchronization and 1 TQ setup and 2 TQ hold time around the sample point.

\[ T_{PROP} = 4 \cdot TQ \] (2.5)

The bus propagation delay is taken twice, because the bus end-to-end path consists of two maximum length branches.

\[ 4 \cdot TQ = 2 \cdot 2 \cdot (T_{LOOP} + \lambda \cdot L_{BUS} + \frac{2 \cdot T_{HUB}}{2}) \] (2.6)

The resulting maximum bus branch length for a hub-based network is:

\[ L_{BUS} = \frac{1}{\lambda} \left( \frac{1}{8 \cdot BR} - T_{LOOP} - \frac{1}{F_{HUBCLK}} \right) \] (2.7)

According to Figure 2.15, at a clock frequency of 44.5MHz as used in the prototype, the resulting branch length is negative at 1Mbps bit rate. This indicates that a hub-based star topology cannot be supported at 1Mbps. Usable branch lengths are supported at 800kbps and lower bit rates [2.31]. One should notice that in star networks the branch length is comparable to a half of the linear network length. Thus the active hub reduces the achievable linear bus length at 800kbps from 40m down to 10m, at 500kbps from 100m down to 48m and at 250kbps from 250m down to 148m. The net bus length reduction caused by the hub can be compensated by adding more branches to the hub-based star network, as shown in Table 2.2.
Table 2.2: Maximum achievable net bus lengths at some bit rates

<table>
<thead>
<tr>
<th>Number of branches</th>
<th>Maximum achievable net bus length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>800kbps</td>
</tr>
<tr>
<td>2</td>
<td>10m</td>
</tr>
<tr>
<td>3</td>
<td>15m</td>
</tr>
<tr>
<td>4</td>
<td>20m</td>
</tr>
<tr>
<td>5</td>
<td>25m</td>
</tr>
</tbody>
</table>

Figure 2.16 presents the maximum achievable branch length according to the current design parameters and the typical bit rates from 250kbps to 1Mbps [2.31]. At the lowest hub clock frequencies below 50Mhz, the maximum achievable branch length is negative, which indicates that the hub cannot support 1Mbps operation at such low clock rates. The active hub can operate at 1Mbps bit rate by using higher hub clock rates. Equation 2.8 gives the maximum achievable branch length at 1Mbps bit rate and by an infinitely high hub clock rate.
Figure 2.16: Maximum achievable branch length at 1Mbps vs. hub clock frequency

\[
L_{BUS,\text{max}} = \lim_{F_{HUBCLK} \to \infty} \frac{1}{\lambda} \left( \frac{1}{8 \cdot BR} - T_{\text{LOOP}} \cdot \frac{1}{F_{HUBCLK}} \right) = 4 \text{ m}
\] (2.8)

The limit value in equation 2.8 shows that the transceiver loop-delay dominates the hub operation speed. At an infinitely high hub clock frequency, only 4m branch length can be reached at full 1Mbps bit rate, which is however too short for most real applications.

2.3.3 Evaluation

A prototype with 3 CAN-ports has been manufactured to be able to evaluate and verify the hub concept with real CAN-networks. At the core of the prototype presented in Figure 2.17 is a CPLD-chip, inside which the distribution logic between the hub ports is implemented. All ports have two connectors enabling efficient use of patch cables for connecting the hub into the target system. Three jumpers in the middle of the board are for enabling and disabling the ports. Those jumpers have been used for testing the operation of enabling and disabling the ports by the optional fault-treatment units.
There are only CAN-high signals presented in Figure 2.18 and Figure 2.19. The low level of the signals presents the recessive state and high level the dominant state. The data frame in the example has an 11-bit identifier and 8 data bytes. The duration of the example data frame is approximately 240ms. Figure 2.18 shows how the active hub has a minimal effect on the node synchronization properties of the CAN bus operating at 500kbps [2.31]. The net forwarding delay is approximately 150ns according to Figure 2.19. A typical net delay caused by the reception of one transceiver and transmission of another one is about 105ns. The internal state forwarding takes two clock cycles, which is 44ns at the 44.5MHz clock rate used in the prototype. At 500kbps bit rate the 150ns forwarding delay represents 15% of the whole propagation time segment. The measurements confirm the theory of timing constraints.

The hub behavior in the case of overlapping dominant states between two branches is presented in Figure 2.19 [2.31]. Originally a node in CAN1 drives the dominant state and the hub distributes the state into CAN2. After a node releases CAN1 to recessive, another node in CAN2 continues driving the bus dominant. After a certain delay, the hub recognizes the incoming dominant state and distributes it to CAN1. Finally, when a node in the CAN2 releases the bus to recessive, the hub releases the CAN1 to recessive.
Figures 2.18 and 2.19 present the stability test results, where a hub-based 3-branch star-topology network has been successfully tested with a relatively high load at 500kbps and 800kbps [2.31]. Error frames have not occurred within 5 to 8 million maximum length standard data frames. The passed test with 800kbps confirmed the maximum bus extension computations presented above. According to the theoretical analysis, the hub prototype
did not operate at 1Mbps because of the two consecutive arbitration phases and a longer propagation segment consumed by the hub logic.

The port disable functionality was tested by switching the state of the appropriate jumper. The port disable interfaces are reserved for the optional fault-treatment units. If a port was disabled, no distributions were visible in the bus connected to the disabled port. Also tested was, whether error frames were distributed correctly between the active ports [2.31].

2.4 CAN switch

A star-topology network implemented with repeaters leads to two repeaters in every signal path. While every repeater doubles the required propagation time, the topology freedom strongly reduces the maximum achievable bit rate. An active CAN-hub is a multiport device, reducing the bit rate degradation in case of a star-topology. Still, a few restrictions are characteristic of the all hub-based CAN-networks:

- Propagation time consumed by the hub itself reduces the maximum branch length. Either smaller net length or more branches shall be used.
• If a maximum supported number of ports are used and longer branches are needed, the bit rate shall be reduced.

• All branches of the network shall run at the same bit rate, which limits use of the different physical layers in different branches.

Bridges can be described as dual-port switches, but in star and mixed topology networks, many bridges are needed. Another drawback is a need for multiple configurable nodes instead of one configurable multiport device. Therefore an alternate solution had to be developed to enhance the limitations of existing solutions. The CAN-switch addresses the previously mentioned challenges well:

• Any branch can operate at any bit rate and support any physical layer.

• Signal propagations are separated to each bus and the switch forwarding delay affects only the signal arrival deadline.

• Every branch is a separate CAN-bus and the net bus length is multiplied by the number of branches.

All possible topologies can be implemented by the switch technology [2.34] and only timing constraints of the application limit the net transmission delay of the bus system [2.32]. A star topology network with three branches can be realized with a single switch, which results in the minimum achievable message delivery delay through the path. More switches are needed for tree topology networks. Switching is the only method to make the CAN systems composable, if the subsystems to be composed are using different ports. If inter-subsystem communications exists, it shall be taken into account during the schedulability analysis of both subsystems. Segmenting networks with switches can increase the net performance of the system [2.35] [2.34] by separating communications into independent collision and error domains as much as possible. Error isolation, maximum number of nodes in a single physical bus and net bus length will be improved when a switched network is used instead of a standard linear CAN [2.32].

The design of a switched network is a rather more complex task than the design of standard passive linear network, because a flow-based analysis [2.30] [2.34] has to be performed for the communication from one bus segment over a switch to another segment to guarantee both data throughput through the switch and transmission times in all bus segments in addition to the normal schedulability analysis.
2.4.1 Concept

The use of a switch in CAN networks has been presented first in [2.32]. The triggers of the development have been support for multiple bit rates in different parts of the network, division of the network into multiple independent error domains and support of star- and tree-topologies. Basically, a CAN-switch equals a managed Ethernet switch. The store-and-forward operating principle [2.30] is used to enable message filtering by the identifier of a CAN-message (CAN-ID), bit rate independence between the switching ports and efficient multicasting and queuing. CAN is a content-based network instead of an address based one like Ethernet, so content-based filtering is used instead of address-based filtering. [1.21] expects CAN-transmitters to be always priority-queue (PQ)-scheduled and the scheduling is performed by CAN-controllers used in the switch.

Predictable performance is important [2.34] to avoid message loss and to guarantee deterministic delays. It has been presented in Section 2.2.1 why it is important in control systems to deliver signal updates before deadline. Most of the implementation issues concentrate on the deterministic behavior of the switch sub-functions. Due to the determinism requirements, a static routing table shall be used unlike in the Ethernet switches.

2.4.2 Implementation

CAN-switch functionality is quite simple and straightforward – incoming CAN messages are either discarded or forwarded to one or more target ports, based on a routing table. Table 2.3 shows that a CAN-message with the maximum number of data bytes forms the worst case to the switch – there are a maximum number of bytes to be forwarded and minimum amount of time per byte available. Minimum time demands minimum number of stuff bits and shortest possible interframe space. The maximum data rate can be reached by using data frames with 8 bytes of data. The payload in Table 2.3 defines the number of bytes to be forwarded in each case, including the message identifier, number of data bytes in a message and the data bytes.

For a 3-port switch the theoretical worst case means continuous flow of one incoming byte in every 3.3\,\mu s from all 3 ports. But there shall also be empty space for transmissions in the target buses, which reduces the worst-case load down to one byte per 10\,\mu s from every port. In that case it is assumed that each received message is forwarded to all other ports of the switch. Such a forwarding principle follows the operation of the CAN physical
layer. In the worst-case scenario, all buses are assumed to operate at 1Mbps bit rate and 100% bus load. In practice the maximum achievable bus load in fully synchronous buses can be approximately 80% and in average asynchronous buses a load of approximately 33% can be reached in optimum conditions.

Table 2.3: Message lengths and payload contents with worst-case scenarios bolded

<table>
<thead>
<tr>
<th>Identifier size</th>
<th>Data</th>
<th>Duration</th>
<th>Payload</th>
<th>Max. data flow rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-bit</td>
<td>0 bytes</td>
<td>47µs</td>
<td>3 bytes</td>
<td>1 byte/15µs</td>
</tr>
<tr>
<td>8 bytes</td>
<td>112µs</td>
<td>11 bytes</td>
<td></td>
<td>1 byte/10µs</td>
</tr>
<tr>
<td>29-bit</td>
<td>0 bytes</td>
<td>68µs</td>
<td>5 bytes</td>
<td>1 byte/13µs</td>
</tr>
<tr>
<td>8 bytes</td>
<td>132µs</td>
<td>13 bytes</td>
<td></td>
<td>1 byte/10µs</td>
</tr>
</tbody>
</table>

Tight performance and reliability requirements raise three design issues. First, the filtering of incoming messages shall be fast enough to allow the switch to handle bus loads up to 100% without losing any messages. Second, queuing shall neither lose messages nor limit throughput of the switch at the maximum link speed. Third, the switch shall transmit messages from the queues in ascending priority order to avoid inner priority inversions.

2.4.2.1 Filter design

SW-based table lookup filtering is simple to implement, but too inefficient for large route tables [2.36]. In CAN-systems, the biggest drawback of a SW-based table lookup is that a hit into different entries gives different filtering delays. Data-dependent filtering delays cannot be accepted by deterministic applications. A HW-lookup filter with content-addressable memory (CAM) [2.36] is much faster and more deterministic, but CAMs are complex structures [2.37], [2.38] and the use of CAMs in filtering may increase the total cost and complexity [2.39] too much. Because the test system and the first applications use CANopen (with 11-bit ID-space by default), simple and fast table indexing by utilizing efficient relative memory addressing modes has been selected [2.44]. For the 29-bit ID space more complex methods will be evaluated in future work. The HW-filters of the CAN-controllers can be used to reduce the event rate for a SW-filter, but the reduction strongly depends on the identifier set to be filtered.

To decrease the memory footprint of the route table, it has been organized according to Table 2.4, where the main entry can be indexed from the table by a CAN message identifier [2.44]. From the returned main entry, a port-
specific nibble is taken out and the bits of the nibble act as target port flags defining into which ports the received message shall be forwarded. The described method has a constant execution time independent of the identifier, source or destination port. Only if a message is forwarded to multiple ports, will the delay increase. In that case, the total rate of incoming messages that can be forwarded will decrease, because there has to be free space in the target bus, too.

Table 2.4: Message filtering table structure

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>Forwarding rules for CAN ID 0x000</td>
</tr>
<tr>
<td>0x001</td>
<td>Forwarding rules for CAN ID 0x001</td>
</tr>
<tr>
<td>0x002</td>
<td>Forwarding rules for CAN ID 0x002</td>
</tr>
<tr>
<td></td>
<td>:</td>
</tr>
<tr>
<td>0x7FE</td>
<td>Forwarding rules for CAN ID 0x7FE</td>
</tr>
<tr>
<td>0x7FF</td>
<td>Forwarding rules for CAN ID 0x7FF</td>
</tr>
</tbody>
</table>

2.4.2.2 Queue design

Based on the literature [2.37] [2.40], output queuing is the most efficient and therefore it has been selected. Input-queued switches have head-of-line (HOL) blocking problems [2.37] [2.41] [2.34] which require virtual output queues (VOQ) and a scheduler [2.46] for efficient operation but this also increases complexity. Non-blocking functionality is essential especially in CAN-switches primarily being used in real-time control systems, where one blocked bus branch shall not block the other branches of a network. While an internal speedup [2.40] is used instead of physically parallel queues, RX and TX interfaces need small local buffers to avoid message loss caused by the internal service order [2.35].

A dedicated internal queue is assigned to each TX port [2.35]. With CAN, forwarding and queuing is intrinsically separated from the scheduling of messages to target media by CAN-controllers. It has been documented in [2.40] and [2.34] that separating the main functional blocks will minimize implementation and timing complexity and transmission medium dependence. Transmit queues with automatic prioritization offer a small output expansion [2.40] – providing buffer space for more than one message.
The internal structure of the improved CAN switch is illustrated in Figure 2.22 [2.44]. In a CAN-switch, SW-based output first-in-first-out (FIFO) queues can be used, because the automatically prioritizing 3-stage TX queue issues frames to the bus in ascending priority order. FIFOs are implemented as ring buffers to minimize the amount of maintenance effort. The input HW FIFOs provide buffering needed by time-interleaved forwarding between the ports.

### 2.4.3 Evaluation

The first application was selected as the test setup presented in Figure 2.23 [2.44]. CAN2 acts as a subsystem backbone located in a physically safe location. It connects the main components of a subsystem operating at 1Mbps bit rate to be able to provide enough bandwidth. CAN0 contains actuators and one transducer and it is also located in a physically safe location. The actuators limit the maximum bit rate of the bus by supporting only 500kbps. All other transducer nodes located in easily malfunctioning areas are connected to CAN1, whose bit rate is limited to 500kbps by its worst-case net length.

Embedded test generator nodes (TGn) were used to generate the stimulus instead of the real system. The main reason is that the worst-case burst-mode stimulus generated by the test generators reveals the potential problems in the queues of the switch. Table 2.5 defines the communication profile of message transfers [2.42]. Other transmissions use intervals of 1s and above and are omitted to keep the results understandable. Messages with identifiers 0x222..0x229 have 4 data bytes and messages with identifiers 0x1B2..0x1BA have 6 bytes payload data. All messages have 11-bit identifier field and 10ms refresh cycle.
Table 2.5: Test network communication profile

<table>
<thead>
<tr>
<th>CAN0 (500kbps)</th>
<th>CAN1 (500kbps)</th>
<th>CAN2 (1Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In</strong></td>
<td><strong>Out</strong></td>
<td><strong>In</strong></td>
</tr>
<tr>
<td>0x1B2</td>
<td>0x000</td>
<td>0x1B3</td>
</tr>
<tr>
<td>0x222</td>
<td>0x1B3</td>
<td>0x1B4</td>
</tr>
<tr>
<td>0x223</td>
<td>0x1B4</td>
<td>0x1B5</td>
</tr>
<tr>
<td>0x224</td>
<td>0x1B5</td>
<td>0x1B6</td>
</tr>
<tr>
<td>0x225</td>
<td>0x1B6</td>
<td>0x1B7</td>
</tr>
<tr>
<td>0x226</td>
<td>0x1B7</td>
<td>0x1B8</td>
</tr>
<tr>
<td>0x227</td>
<td>0x1B8</td>
<td>0x1B9</td>
</tr>
<tr>
<td>0x228</td>
<td>0x1B9</td>
<td>0x1BA</td>
</tr>
</tbody>
</table>

2.4.3.1 Preliminary performance tests

The preliminary tests were conceptual evaluations and were performed with COTS hardware as presented in Figure 2.24. It was known that performance problems due to the multiplexed host bus interfaces of 2 CAN-ports may be observed but still the test HW was the best existing platform in the market at the time of the first tests. The results were used as reference for the final implementation. There was a 6-stage HW input FIFO with a single output buffer in two of three ports and a 2-stage HW input FIFO with a 3-stage prioritizing output buffer in one port. So some incoming interrupts were lost while output prioritization was performed by the software. The internal buffering was message identifier oriented, where too fast retransmissions could override the trigger of the previous transmissions.
Figure 2.24: The switch prototype platform

Figure 2.25 shows approximately 135µs forwarding delay for single message in all forwarding directions [2.42]. Single-message performance represents the absolute minimum achievable forwarding delay because other messages neither load the switch nor increase the forwarding delay with higher message priority.

In all the system-level results of the prototype platform, there are samples in the X-axis and measured forwarding delay in the Y-axis. If delays of multiple messages are presented in the same figure, the curves are plotted in ascending order from bottom to top according to the message identifier. Curve labels are organized in the same order. Analyzers measure reception times only and the transmission time of the frames in the target bus has been subtracted from the measured values to be able to show the actual forwarding delays.
The faster to slower bus forwarding results in Figure 2.26 show how heavy transmission queuing software causes relatively long delays and how the delay varies according to the incoming message rate [2.42]. SW
prioritization seems to work well, because lower priority messages have longer delays than higher priority messages.

Figure 2.27: Forwarding delay from CAN0 to CAN2 with the switch prototype

Figure 2.28: Forwarding delays from CAN1 to CAN2 with the switch prototype
The forwarding delays from slower to faster buses in Figures 2.27 and 2.28 are shorter because of HW TX buffer management, but SW queue
management still has large variation due to the processor load variation [2.42]. Prioritization is not properly visible because of the limited timing resolution. Figures 2.29 to 2.31 confirm the assumption of heavy queue processing. The load variation is smaller especially when all the ports are running at same bit rate and only few messages have to be buffered inside the switch [2.42].

![Graph showing forwarding delay from CAN1 to CAN2 with the first switch prototype](image)

Figure 2.31: Forwarding delay from CAN1 to CAN2 with the first switch prototype

### 2.4.3.2 Final performance tests

An optimized switch prototype as presented in Figure 2.32 was manufactured for the final tests. Since the first evaluation, a microcontroller family optimized for switching purposes has become available in the market. There are up to 5 on-chip CAN-controllers and a dedicated I/O-processor optimized for communication applications operating in parallel with the host processor. It has solved the main problems recognized in the first evaluation [2.42]:

**Slow CAN-controller host bus interface:** All CAN-controllers are directly connected to the parallel memory bus to provide maximum achievable host bus bandwidth.

**Improved message buffers:** In the new microcontroller, all on-chip CAN-controllers have 5-stage RX FIFO and 3-stage automatically
prioritizing TX buffers which support well the buses operating at the maximum bit rate and high load.

**Host processing performance:** In addition to the reduced SW load, more simple routing software could be implemented and in the new microcontroller (MCU) there is a dedicated I/O-processor for communication applications running at high clock frequency, providing extreme queue processing performance and specially optimized to serve interrupt requests.

Figure 2.32: The optimized switch prototype

Figure 2.33: A CAN-message received (upper) and transmitted (lower) by the optimized switch
The optimized hardware presented earlier has been developed for the final tests. Because one of the main targets was the development of a new commercial product, CAN license costs for the FPGA-based implementation were too high in this phase of the work [2.43]. Therefore a fixed MCU-based approach with smaller non-recurring engineering (NRE) costs had to be selected. The improved best case forwarding delay for single message in all directions is presented in Figure 2.33. The delay has been reduced down to approximately 15µs [2.44].

Figure 2.34: Forwarding delays from CAN2 to CAN0 with the optimized switch

The triple bus test setup and communication profile is presented in Table 2.4. Use of the same test setup enables direct comparison of the results. The results of faster to slower bus forwarding are presented in Figure 2.34 [2.44]. The delay variation is negligibly small when compared with the results from the first prototype. This is mostly because of the hardware optimized to the communication purposes and well-designed forwarding software and queue structure. The magnitude of the worst case forwarding delay has decreased from 3.9ms down to 0.8ms. The effect of arbitration in the slower destination bus is clearly visible in the results. The time difference of messages is 80µs, which comes from the transmission time difference of the CAN message with 4 data bytes between the faster source and the slower destination buses.
Figure 2.35: Forwarding delay from CAN0 to CAN2 with the optimized switch

Figure 2.36: Forwarding delays from CAN1 to CAN2 with the optimized switch

The results from slower to faster bus forwarding are presented in Figures 2.35 and 2.36 [2.44]. The forwarding delay is almost constant because
previous message has been sent before receiving the next one. This is possible because there are no higher priority transmissions in the target bus and the target bus is operating at double bit rate compared with the source bus.

2.4.3.3 Error tolerance tests
While the main focus in the development has been in forwarding performance, error-tolerance and troubleshooting capability [2.42] have also been in focus with both platforms. The reaction of the optimized switch to the most common fault conditions has been verified according to the following list [2.44].

**Single errors and error frames:** No forwarding to the target port(s).

**Port error warn:** A warning is generated if serious communication problems exist in a port (see error-warn state in fault-confinement).

**Port error passive:** A warning is generated if serious communication problems exist in a port (see error-passive state in fault-confinement). A port in error passive state has decreased its transmission capability and may also generate other problems, like queue overruns.

**Port bus-off:** An error message is generated and a port goes to bus-off if any message cannot be sent to or received from a port.

**RX overrun:** RX overrun error message is generated when the switch cannot empty the receive buffer fast enough. As long as the HW is not damaged, there should be no RX overrun errors.

**TX overrun:** If no other nodes are connected to the target port, TX overrun error is generated by the switch to tell that messages cannot be sent to the target bus fast enough.

2.4.3.4 Long-term stability test
1Mbps bit rate and "copy all received messages to the other ports" forwarding method has been used for all ports. Traffic has been generated by embedded test generator nodes transmitting bursts of 8 messages cyclically. Bursts are used to verify the correct operation and performance of the software queues and buffer access routines. According to Figure 2.37, in total over 10 million frames were switched without lost frames and errors [2.44].
2.5 Summary

The use of CAN networks has spread all over industrial, automotive and building automation. In every application field, the use of topologies other than linear ones has been increased. Therefore solutions that extend the standard CAN physical layer with support for alternative topologies have been presented in this chapter. Based on the existing research, CAN does not support natively any other than linear topology. Thus, the only possibility is to utilize the active networking elements to provide more flexible topologies.

A typical hub regenerates signals and operates on the physical signalling layer and keeps all ports in the same collision domain [2.45], also in CAN networks. The CAN hub concept presented in this chapter does not modify the synchronization of nodes connected into the same logical bus. Many CAN applications utilize the synchronization between the nodes. The major drawback of CAN-hub is the need for a double propagation delay reducing either maximum bus length or network bit rate. Another restriction can be derived from the first one: propagation time reduction practically limit the maximum number of hubs to one and supported topologies to star. Just like Ethernet hubs [2.46] [2.47], a CAN-hub supports different physical layers in different interfaces, but expects them to operate at same bit rate. A CAN-hub does not support filtering either, so the bit rate is the only parameter to be configured.

In both Ethernet [2.46] [2.47] and CAN-networks, some management features can be implemented into the hub devices. Most management features are various diagnostics functions and in CAN-hubs interface fault treatment units. Fault treatment units have been investigated so well that only interfaces for those have been included in the CAN-hub design for the
high-speed physical layer presented in this chapter. Because a CAN-hub is operating at the physical signaling layer, it is almost fail silent. This means that detecting an error and disconnecting the faulty interface causes up to two error frames. Its effect on the bus operation is still small and the remaining part of the system can perform application dependent fault reaction and inform the operator or higher layer services about the problems. More advanced features can be implemented into CAN-hub by adding a local CAN-node as part of the hub. The most important features, already used in Ethernet hubs, are node polling and interface specific statistics [2.47].

Switches operate in a store-and-forward manner by receiving complete frames, optionally filtering them and sending to the target interfaces specified by a routing table [2.45]. For CAN networks, switching technology offers support for various topologies without bit rate limitations. Application layer latencies form the only timing constraints. The use of switches increases the possible physical extension of a CAN network by enabling each branch to be as long as a linear CAN-network running at the same bit rate. A CAN switch also allows the use of different physical layers in different interfaces, even running at different bit rates, similar to Ethernet switches [2.48]. In addition to the performance improvements, CAN switches increase the fault tolerance of a CAN network. When a CAN network is divided into parts by a switch, all physical layer and data link layer failures and electric disturbances are isolated within the originating segment. A CAN switch is intrinsically a fully fail silent component – the only affect on the other parts of the network is missing or delayed transmissions, but error frames and retransmissions are not propagated into other segments. That behavior significantly increases the timeliness of the remaining communication during errors local to a single branch [2.16]. The only drawback of a switch is the limited synchronization between the nodes due to the store-and-forward operating principle.

A CAN switch is a managed one to efficiently support different bit rates in different interfaces. The bit rates of the interfaces and the message routing table can be configured. Flexible filtering can be utilized for load balancing between the interfaces and for reducing the probability of passing through messages corrupted by masquerade error. Masquerade error occurs when more than one bit in a valid data frame are corrupted so that the frame looks like another valid data frame. Reducing bus load in every interface reduces the transmission error rate [2.5] and in that way improves the performance of the whole logical network. Basic CAN systems can not be composable,
but by using different switch interfaces for different communications, a switched CAN network can be. Because a CAN switch acknowledges an incoming frame and transmits it to the target interface(s), possible lost frames need to be monitored by the application layer protocol. Routing and interface specific diagnostics features are equal with Ethernet switches [2.49].

For physical HW platforms, switches provide support for several CAN interfaces and high-performance forwarding functionality. Performance tests have proved that, at least for four ports, processor-based forwarding can operate fast enough. Hubs are more challenging from the platform point of view because they operate 10 to 100 times faster than the bit rate. A hub cannot be implemented with software and processor core. Therefore a configurable logic platform, typically a CPLD or an FPGA, is required. A fully configurable platform also gives better flexibility to switch implementations. Small switches can be implemented with processor cores but larger ones need dedicated forwarding logic. Use of an FPGA also enables the flexible integration of various fault treatment units, media redundancy concepts and combinations of HUB and switch and maybe some application processing platform.
Chapter 2 concentrated on high-speed fieldbus networking and this chapter in turn concentrates on low-cost and low-speed fieldbus networking. There are a lot of low-cost, low-performance bus systems, of which many are proprietary [1.2]. Some of these, like ISO 9141 [1.20] and TTP-A are standardized but either offer too low performance, are too complicated or too cumbersome [1.2]. Local Interconnect Network (LIN) has been selected for this dissertation because of its high acceptance in the market and well-designed technical features when compared with competing technologies. The LIN specification defines more accurately than the competing standards the whole network management process, containing network design, configuration file formats, configuration data exchange, application programming interface (API) and hardware abstraction layer (HAL).

Low-cost networks are needed when various low-performance and cost sensitive subsystems are integrated into the systems. The purpose of using bidirectional communication down to intelligent sensors and actuators is to increase system flexibility and maximize component re-use. Bidirectional communication also enhances system-wide monitoring possibilities by providing full access to each sensor and actuator in the systems. A limited increase in component cost can be accepted to reach reduced system life cycle cost via better modularity, increased component re-use [1.2] and more sophisticated system condition monitoring. For system production, sophisticated low-cost bus systems provide more generic wiring or cabling. E.g. a general-purpose Smart connector allows for higher production quantities for the nodes and still enables use of conventional sensors and actuators [1.2].

The introduction continues with a review of existing LIN-based solutions. After the introduction the effects of the active hub on LIN communication is described. Because the full hub limits the bit rate much more in LIN networks than in CAN networks, a reduced hub concept is introduced. The reduced hub is simplified further to a multiport master interface to provide bigger physical extension and support of more slave nodes for a single network. Finally, there is a discussion on why a switch can not be used with LIN.
3.1 Introduction to LIN

LIN has been designed to provide a bidirectional, ultra low-cost communication network below e.g. CAN to increase system flexibility and reduce cabling or wiring costs and weight, system assembly time and costs. The unit cost of a single node has been minimized by using standard universal asynchronous receiver and transmitter (UART)/serial communication interface (SCI) as a basic communication controller instead of a dedicated one like in e.g. CAN. A single signal line for communication together with the low bit rates enables flexible topology and cost-efficient wiring without problems with EMI.

Originally LIN was very simple and had a very limited set of features [1.19], but it has quite rapidly been developed into more sophisticated but also more complex [1.23] bus system. While ISO 11898-1 specifies the data link layer and ISO 11898-2 the high-speed physical layer, the LIN specification package specifies the whole communication system including physical layer, data link layer protocol, transport protocol, and system management process. The transport protocol specification includes the most essential configuration and diagnostics services. The network management process is thoroughly defined, including the required design file formats. To maximize the process efficiency, both HAL and communication API are defined. Standardized interfaces enable efficient component re-use in various systems and interoperability between nodes provided by multiple vendors. LIN is the first standard bus system specifying power-saving functionality as an integral part of the communication system since the first version [1.19], [3.1], [1.23], [3.2]. It makes LIN very suitable for partially or totally battery-operated devices like combustion engine operated vehicles.

3.1.1 Physical layer

The physical layer has remained constant since LIN 1.0 and is based on the ISO 9141 [1.20]: only application refinements have been introduced afterwards concerning constraints between the maximum number of nodes in a network and net bus length [3.3]. The physical layer of LIN consists of one single-ended signal wire and ground wire. The LIN bus operates at a system battery voltage level which can vary between 6V and 18V. To minimize EMI emissions, the slew rate of the bus signal is strictly limited. The bus line is terminated at the master node with a $1\,\text{k}\Omega$ pull-up resistor connected to the battery voltage through a diode. In addition to the master
termination, each LIN node has a weak internal pull-up to the battery
temperature. LIN supports practically any topology – linear, ring or mixed [3.3].
The maximum allowed network length varies from a few meters up to 40m,
depending on the bus line capacitance and the number of nodes connected
to the bus. There can be one master node and up to 15 slave nodes in a
single physical LIN segment [3.3].

Just like CAN, the LIN bus can assume two states, recessive or dominant.
In the recessive state the bus is idle or driven to the logical "1" and the bus
voltage level is close to the system battery voltage. The dominant state
always presents logical "0". LIN bus behaves as wired-AND, just like CAN,
and if one or more nodes are driving the bus to the dominant state, the
resulting bus state is dominant [1.19]. The bit coding follows one of the
most typical used with UARTs – one start-bit, 8 NRZ-coded data bits
without parity bit and with one stop-bit [1.19]. The basic synchronization
for each 10-bit symbol is performed by the UART.

The typical serial line bit rates 2400, 9600 and 19200bps have been
standardized [1.19] and also some LIN systems are also running at
10417bps, which has earlier been used in ISO 9141 systems [1.20]. In CAN
systems, the position of the sample point can be adjusted quite much and
located close to the end of the bit time, but in LIN the sample point is
located between 43%-63% from the beginning of a bit-time [3.3]. The
SYNCH field of a LIN frame offers an additional mechanism for slave
nodes to measure the bit rate and adjust the internal communication clock
according to the transmission.

### 3.1.2 Frame types

Two frame types, data frame and wakeup frame, are defined by the LIN
protocol specification. The data frame is used for all data transmissions and
its structure is presented in Figure 3.1. A data frame consists of two main
parts: master header and slave response [1.23]. The master indicates with
the header which data frame shall be transmitted and the slave response
contains the requested data. Two data frames are reserved for manufacturer-
specific use and two other frames are reserved for e.g. LIN configuration
and diagnostics protocol. The wakeup frame is an integral part of the
power-management mechanism included in the LIN protocol.

The master request begins with a SYNCH_BREAK field, which consists of
a 13 bits long dominant break signal and one recessive bit. It always starts
the transmission of a new frame [1.23]. It is followed by the SYNCH field,
which enables slave nodes to automatically detect the bit rate and synchronize their internal clocks with the master node’s clock. It has always the value 0x55, which together with the dominant start bit provides 5 consecutive dominant to recessive edges for the communication clock synchronization. The last field of the header is a protected identifier field consisting of 6 frame identifier bits and two identifier parity bits. The parity bits have opposite polarity and they are computed over partially different identifier bits. In LIN versions 1.0 to 1.11 the two most significant identifier bits also defined the number of data bytes contained in the slave response [3.1], but in later versions the actual number of data bytes is defined only in the LIN definition file (LDF) defining the structure, timing and communication profile of the whole network [1.23].

The slave response consists of 1 - 8 bytes of process data and a single byte checksum. The checksum has two versions: a classical checksum defined by LIN 1.x [3.1] and an enhanced checksum defined by LIN 2.x [1.23]. The checksum algorithm is the same in both variants, but while the classical checksum is computed over the data bytes of the slave response only, the enhanced one also covers the protected identifier of the master header. The maximum duration of both header and response have been specified to 140% of the nominal duration. Furthermore the maximum slave response time has been specified and all nodes are capable of handling all messages. The inter-frame space consists of the unused frame transmission time and the unused time of the frame slot.

The wakeup frame consists of 8 dominant bits followed by at least 4 recessive bits and it is used by LIN slave nodes to wakeup the master node. Wakeup frames can collide if two or more slave nodes need to wakeup the system simultaneously. The wakeup sequence consists of up to two sets of three wakeup frames [3.3]. If a master does not start communication after the full sequence, the slaves give up and return to the sleep mode. The wakeup sequence will be stopped immediately when the master node
transmits the first SYNCH_BREAK. An example of successful wakeup procedure is presented in Figure 3.2.

![Figure 3.2: LIN bus wakeup process](image)

### 3.1.3 Communications concept

LIN is a typical time-triggered bus system. At run-time there are fixed, statically designed sets of schedule tables [1.23] as presented in Figure 3.3. The application in the master node selects the active schedule table according to the application state. In a schedule table there is a fixed time slot for each frame. Each frame transmission shall fit into the assigned frame slot and there shall be at least 40% of the nominal frame duration reserved spare time for each frame – Tolerance in Figure 3.3 – to compensate for minor variations in the communication. Inside a slot, the master initiates the communication by transmitting the master header. If a previous message transmission is pending for some reason, the SYNCH_BREAK field of the next slot will override and stop the delayed transmission. When a LIN bus is idle long enough or when the master sends a sleep-command, the nodes shall enter the sleep-mode, in which communication cannot be performed and system power consumption is negligibly small. Communication can be started again either by the master node just starting to transmit master requests or by the slave nodes proceeding with the wakeup sequence.

There are three different types of data frames. Unconditional frames were originally defined by LIN 1.0 and they are always transmitted when requested by the master [1.23]. Collisions are absolutely prohibited with the unconditional frames. Event-triggered frames are defined by LIN 2.0 and they have been designed to carry event data efficiently. An event-triggered frame will be transmitted only when it has been requested by the master and at least one signal assigned to it has been changed since the last transmission. Two or more nodes can simultaneously transmit event-triggered frames, and to enable the master to detect a collision each slave
enters the protected frame identifier of the corresponding unconditional frame to the first data byte. During reception, the master can detect the collision from the corrupted data in the first data byte and start determining the initiating node of the event with unconditional frames. If a master node has data to be transmitted, it can use sporadic frames to transmit the data. Sporadic frames can only be transmitted based on events and they cannot be included in a schedule table.

**Figure 3.3: Example of the LIN schedule table**

### 3.1.4 Error detection

LIN does not support atomic error handling. Each node handles errors locally, because global error messaging has not been defined as in CAN. The lowest level of error detection is performed in the data link layer by the UARTs. Because parity is not used, the data link layer error detection is limited to the checking correctness of the fixed start and stop bits. Master and slave nodes monitor the status of the communication differently. In
addition to the following cases, all transmitting nodes shall always monitor bit errors during transmission [1.23] [3.3].

Errors monitored by the LIN master node:

**Physical bus errors:** It is not possible to transmit valid bits to the bus

**Slave responding too late:** A slave does not start the response transmission within the specified time

**Slave not responding:** None of the slaves responded to the master request

**Transmission time exceeded:** Maximum allowed frame duration exceeded e.g. due to too long spaces between the data bytes in the slave response

Errors monitored by the receiving slave node:

**Identifier parity errors:** Parity of the protected identifier was not correct

**Inconsistent SYNCH field:** Synch-field differs from 0x55

**Checksum error:** The checksum contained with the frame does not match the checksum computed over the received frame

### 3.1.5 Fault confinement

Because LIN is an extremely simplified communication system, there is no fault confinement such as in CAN. Because all errors are handled only locally, it is recommended to include the communication status information in one of the data frames transmitted to the master node [1.23]. Only unconditional frames can be used for status information transfers. The status information is handled differently among different versions of LIN. In LIN versions up to 2.0, only a single Boolean type status signal is required in one frame of each cycle [1.23]. In J2602 systems, all unconditional frames shall contain a dedicated status byte [3.3]. The only exception for the local fault confinement occurs when a slave response transmission is delayed over the specified frame slot. A master node transmits the SYNC_BREAK of the next data frame just in time, which globally interrupts the pending delayed transmission and globally claims the delayed frame erroneous. To the local application the LIN status is provided with two Boolean signals [1.23].
3.2 Existing LIN solutions

There are not many publications covering LIN applications. LIN technology does not present any new technical advantages – the biggest advantages are economical and system integration related. LIN is very strong in drive-oriented applications, where a master controls multiple slaves. Reading inputs from multiple slaves is relatively slow when the slave nodes are polled cyclically [3.2]. LIN 2.0 has improved the performance by introducing some more functionality. Event-triggered frames provide a more efficient way to poll the possible events from several slave nodes [1.23]. The published solutions concentrate on systems having only a few nodes located in small areas mostly in the automotive application area [3.4] [1.2] [3.2]. Typical documented applications cover car [3.2] and bus [3.4] [1.2] body electronics. A LIN Smart Connector concept [1.2] minimizes the variety of different interface nodes and maximizes the re-use of few node types. Typical target positions in cars are e.g. interior and exterior lighting, window, seat and sunroof controls, control panels in doors, and steering wheel and other positions in the car cabin area. In the bus body electronics, the biggest advances can be found from passenger lighting, where scalability and flexibility provided by the intelligent lighting actuators are essential [3.4].

Application notes give a well-structured view of the philosophy of LIN systems. There are few common types of subsystems, of which numerous systems can be integrated. Typical subsystems are key panels with illuminating capability [3.5] [3.6], simple alphanumeric displays [3.7], and DC-motor drive units with integrated sensing and safety functions [3.8] [3.9] [3.10] [3.11]. The drive units are typically implemented as Smart Connectors [3.12] [3.10] [3.11], closing control loops locally and taking only setpoints from and providing status to the master node. The infrared receiver presented in [3.13] is not so common in real applications, but it emphasizes the modular approach. While most of the presented subsystems are based on a microcontroller, LIN slave nodes can also be implemented in FPGA [3.9] to offer maximum semiconductor level flexibility to the applications. Many types of automotive body control systems can be integrated from the subsystems presented earlier. Intelligent DC-motor drive units can be found e.g. in door locking systems [3.14], window controls [3.11], seat position control, door sliding, mirror positioning, light leveling, wiper, heating, ventilation and air conditioning applications [3.12]. The most extreme documented LIN-system is a 5 degrees of freedom arm.

### 3.3 Active hub for LIN

The topology of LIN has been defined as flexibly as possible since the first version – linear, star, tree and mixed topologies are natively supported [1.19]. Because of the flexible topology support, hubs are not needed to increase the flexibility. Instead, restrictions between the maximum number of nodes in the bus and the maximum allowed bus length have been stated in [3.3]. With the maximum number of nodes in a bus, physical extension is still usable for the most of the typical local machinery instrumentation applications and does not necessarily need enhancements. Together with the time-triggered communication, the bit rate is feasible for low-performance systems and especially for cost-sensitive auxiliary hydraulic drive subsystems. If higher performance is needed, LIN is not the correct technology. The most dominating drawback of LIN, especially in hydraulic drive applications, is a limited number of slave nodes in a single bus. If single input or output slave nodes like *Smart Connector* [1.2] are used, only up to 15 I/O-positions can be covered by a single LIN-network. However, there may typically be many more, even up to tens of I/O-positions to be controlled in a very limited area as in the example presented by Figure 3.4. The limited number of nodes in a bus is the major drawback, which can be improved by a LIN hub device.

![Figure 3.4: Typical control entity with more than 15 valve coils to be driven](image)

Hub behavior is equal in both CAN and LIN networks, independent of the different data link layers. Transceivers of both bus systems provide transmission and reception signals for the forwarding state-machines. As the behavior of the transmission media is similar, the basic functionality of
the CAN hub introduced in Chapter 2 also directly applies to LIN. An increased propagation time segment is an even bigger drawback in LIN networks than in CAN networks because the propagation delay relative to the whole bit time is shorter in LIN systems. The sample point location is defined to 43%-63% from the beginning of the bit time. The maximum allowed duty cycle variation is 20% and maximum network loop delay 10% [3.3]. The loop delay exists in both sides of the hub, which results in a very short worst-case margin before the sample point. A full-featured LIN hub supports all standard communications including overlapping dominant states during simultaneous wake-up frames and collisions in the first data byte of event-triggered frames. Because of the limited slew rate of the transceiver and earlier sample point, a LIN hub strongly reduces the usable bit rate and increases system complexity without providing a significant performance increase.

3.4 Reduced LIN hub

3.4.1 Concept

The idea of a reduced hub is to provide a faster operation during the bit-time and bit transitions in a single direction at a time, but still keep all transmissions visible in all branches connected to the reduced hub. To meet the idea, overlapping dominant state resolving and simultaneous dominant state reception management have been discarded to improve the worst-case forwarding delay. It is no longer needed to wait for the branches to settle and determine whether any node is still driving the bus dominant and propagate the determined state to the other branches after the maximum allowed propagation time. Instead, all received dominant states are directly distributed to all the other branches. The represented reduction enables support of the highest supported bit rate.

Overlapping and simultaneously received dominant states are so close to each other that resolving both scenarios is either supported or not supported. Simultaneous reception of a dominant state from multiple branches is a rare but critical condition which is marginal if compared with the more problematic reception of the overlapping dominant states. Most essential is that the reduced hub has a much shorter worst-case forwarding delay than the full hub. In addition to increased forwarding performance, the reduced hub consumes slightly less resources. This is however not critical, as the optional fault-treatment units take much more resources than the hub logic. Simple fault-treatment units are still needed in most applications. In the
simplest level they provide automatic branch disconnect in case of a stuck-at-dominant error and an automatic reintegration after the error has recovered. That simple functionality covers the most typical errors affecting the whole bus, and neither complex protocol controller nor analysis modules are needed.

3.4.2 Implementation

The structure of the reduced LIN hub is presented in Figure 3.5. Port A and Port C are normal LIN-ports and Port B acts as a LIN-transceiver for the local LIN-node. Because typical LIN-systems are extremely cost-sensitive, a separate hub component can not be introduced. It is more efficient to include a hub as an integral part of either a master or a slave node. Therefore the local interface port has been included to the prototype. The main difference between the full and reduced hub is that the transmission (TX) state-machine of the full hub has been replaced with TX logic. The logic consists of logical AND gate combining the bus signals to be transmitted and logical OR gates providing the port disables by freezing the signals to be forwarded to the recessive state. All disable functions are implemented in the TX logic instead of the reception (RX) state machine, because the RX state machines also feed the fault treatment units disconnecting the faulty and re-integrating the healthy interfaces. Each TX-logic is fed by each fault treatment unit (FTU). In the local Port B, a FTU is not used because there is no cabling or wiring between the local hub port and local LIN node.

Only bus division logic has been implemented in the reduced hub. Resolving overlapping dominant states is not supported. The forwarding becomes faster because it is not required anymore to wait for the propagation time. Therefore the reduced hub causes the whole bus to stuck-at-dominant error state if two or more nodes are simultaneously transmitting a dominant state. Error recovery can be performed only with the fault treatment units but the error recovery will corrupt the schedule in most cases. The RX state machine of the reduced hub, presented in Figure 3.6, consists of the following states.

- **PASS:** PASS state will enable forwarding of the dominant state driven by a node external to the hub
- **HIDE:** HIDE state is for preventing an infinite loop-back of the dominant state transmitted by the hub itself
**IDLE:** IDLE state is entered after the corresponding bus has returned to the recessive state, independently of the active state.

Figure 3.5: Block diagram of reduced LIN hub

![Block Diagram of Reduced LIN Hub](image)

Figure 3.6: Reception state-machine of reduced LIN hub

![State-Machine Diagram](image)

### 3.4.3 Evaluation

The reduced hub concept was evaluated with a prototype presented in the Figure 3.7. It consists of two LIN ports and one host port for a local LIN node. It operates at all bit rates with a single configuration, and proper operation at all standard bit rates from 2400bps to 19200bps was
successfully tested. A LIN analyzer program was used to simulate the LIN node connected to the host port. Embedded LIN nodes were used as slave nodes.

![Figure 3.7: Reduced LIN hub and multi-port master interface prototype](image)

There are two main configurations for the hub to be connected to the LIN system – either master or slave node integrated in the same platform with the hub as presented in Figures 3.8 and 3.9 respectively. It does not matter into which node the hub is integrated, but typical LIN systems are very cost-sensitive and the master nodes are higher end ones than the slaves and in that way a more natural location for the optional hub functionality. Another reason to integrate the hub into the master is to keep the existing slaves fully backwards compatible with existing systems.

![Figure 3.8: Reduced LIN hub test setup A](image)

Both test setups can produce both of the timing scenarios presented in Figure 3.10 and Figure 3.11, depending on the location of the responding slave in the system. The results show that the reduced LIN hub adds only a small delay $T_{hub}$ to the total network loop delay. It consists of a single clock cycle RX state machine delay and the propagation delay of the transmission logic. A relatively low 16 MHz hub clock rate was used in the
prototype and due to the low bit rates used in the LIN systems the delay caused by the reduced hub is negligible.

Figure 3.9: Reduced LIN hub test setup B

Figure 3.10: Reduced LIN hub scenario A

Figure 3.10 presents the timing scenario A, where the master node is connected to the branch \textit{BUS\_M} and the responding slave node A to the branch \textit{BUS\_A}. In that case the hub forwarding delay \( T_{HUB} \) occurs twice, before and after the in-frame response space \( T_{RESP} \) of the responding node A. The net effect on the effective response time \( T_{REF} \) is negligibly small. \( T_{REFF} \) may be slightly different for the second slave node B connected to the branch \textit{BUS\_B}, because of the variable response time. The hub delays are constant. In scenario B, presented in Figure 3.11, the responding slave is connected to the same branch with the master node. In this scenario the hub forwarding delay \( T_{HUB} \) causes only a short latency between the source
branch \textit{BUS\_M} and the target branches \textit{BUS\_A} and \textit{BUS\_B}. The effective response time \( T\text{\_REFF} \) equals the actual slave response time \( T\text{\_RESP} \).

![Diagram of Reduced LIN Hub Scenario B](image)

**Figure 3.11: Reduced LIN hub scenario B**

A long-term stability test was performed with LIN analyzer as a master and LIN evaluation boards as slaves according to test setup A. During 8 million frames at 19200bps, no errors occurred. According to the reduced hub limitations, only unconditional data frames were used in the test. Port disable was tested with direct disable signals from jumpers to keep the focus on the reduced hub behavior. A special setup with two parallel-connected transceivers was used to confirm that simultaneously transmitted dominant state to both the slave ports caused the \textit{stuck-at-dominant} error condition in the reduced hub.

### 3.5 LIN multiport master interface

#### 3.5.1 Concept

The full hub reduces the bit rate and the reduced hub prevents the use of wakeup frames and event-triggered data frames. However, fault-treatment units provide better reliability when faulty branches can be disabled during fault conditions and reintegrated after faults have recovered. Node membership monitoring can report the rest of the system about missing and potentially defective nodes as long as the remaining branches remain operational. The reduced hub is still too complex if the benefits are compared with the constraints it imposes on the communication. One should remember that a typical LIN subsystem operates on a master-slave
basis and communication between the slaves and branches can easily be avoided by performing application processing in the LIN master node. Because of the master-slave communication principle, a single AND-gate can be used for combining the branches [2.3] in the LIN master node. It combines the LIN bus states of different branches to the state of a single logical bus. Interface disable can be implemented with a single OR-gate, by which the transmitted bus state can be forced to recessive. There are neither restrictions for bit rate nor for protocol functions or collisions. Only communication between the parallel buses is performed indirectly via the master node.

3.5.2 Implementation

While the core hub functionality has been reduced down to a single AND-gate, the fault-treatment units and branch disconnect/reintegration logic will improve the overall reliability, even with small amount of extra logic. The block diagram of the implemented multiport master interface is presented in Figure 3.12. It consists of combinatorial logic only. Simple fault-treatment units, like stuck-at-dominant detection timers, fit well to an entry-level CPLD or a negligibly small part of an FPGA.

Figure 3.12: Structure of LIN multiport master interface

The minimum fault treatment unit is a stuck-at-dominant detector, as with CAN. Because all LIN data frames are requested by the master node, it can directly find out the transmitter of a corrupted data response frame and possibly disable the corresponding interface, if the functionality is needed. The best thing is that instead of the complex interface-specific protocol control units, only simple counters are needed. Automatic disconnection of
a faulty port can also be made in SW if a micro controller (MCU) is used. In the case of a permanent fault, the host MCU can disconnect a single port at a time and leave that port disconnected, which restores the bus operation. Due to a try-and-error operation the approach has a strongly negative impact on the bus operation. The main benefit of the approach is that additional logic is needed only for disabling the bus interfaces.

3.5.3 Evaluation

The multiport master interface was tested with the prototype presented in Figure 3.8. The only difference is in the internal operation – a hub can be a separate component or integrated into either master or slave node but the multiport master interface is always integrated into the master node. A LIN analyzer was used as a master and LIN slave evaluation boards as slave nodes. A typical sensor node transaction, where the master requests a data frame from the slave node is presented in the Figure 3.13. The master node transmits a header to the branch BUS_M, which is forwarded to the branches BUS_A and BUS_B. Each slave node receives the header after a hub forwarding delay T_{MPMI}. A slave replies with a data frame after the response time T_{RESP} and after another forwarding delay T_{MPMI} the master receives it. The only deviation from the standard LIN behavior is the slightly increased effective slave response time T_{REFF}. It remains negligibly short because the forwarding delay consists of only a single clock period of the hub system clock. The main limitation of the multiport master interface is that the slave response transmitted to the branch BUS_A is not visible to the slaves connected to the branch BUS_B.

Another typical transaction presented in Figure 3.14 is a typical actuator command transmitted by the master node to one or more actuator slave nodes. In this case the multi-port master interface adds only a short forwarding delay T_{MPMI} to both header T_{HDR} and response duration T_{FRAME}. In this direction, command values for several nodes located in the different branches can be mapped into the same data frame, because the master's transmissions are visible to all the branches. Longer-term stability tests were also run for the multi-port master interface. Over 8 million unconditional data frames were transmitted without lost messages or timing errors. Furthermore, it was verified that dominant states simultaneously received from branches BUS_A and BUS_B did not lead to a jamming of the forwarding logic, which confirms compatibility with wake-up and event-triggered frames.
Figure 3.13: LIN multiport master interface behavior A

Figure 3.14: LIN multi-port master interface behavior B

### 3.6 Switch in LIN networks

With CAN, a switch provides the best flexibility and minimum limitations with respect to topology, bit rate, maximum allowed bus length and number of nodes in a single physical network. In LIN-networks it is not possible to use a switch because of the frame structure – there are both fixed-time and fixed-form fields in each LIN data frame, as presented in Figure 3.1. Each frame begins with the SYNCH_BREAK, consisting of 14 bit-times in total.
A switch can forward a frame after detecting a recessive bit after the 13\textsuperscript{th} consecutive dominant bit – 14 bit-times after the begin of the master header. In the example in Figure 3.15 \textit{BUS\_M request} presents the original master header transmission, to which a slave device responds with a data frame. The data frame from the local branch is designated as \textit{BUS\_M response}. Requests forwarded to the other branches are designated as \textit{BUS\_A} and \textit{BUS\_B}. It is assumed that a node in the \textit{BUS\_A} responds to the example master header.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{lin_switch_timing.png}
\caption{LIN switch timing}
\end{figure}

The example in Figure 3.15 shows that a LIN switch would cause at least a 14 bit-time forwarding delay for the master header. In addition to that comes the slave response time. These times together form the effective response time $T_{\text{REFF}}$, which is at least 14 bit-times long. This is up to 140\% of the nominal frame duration reserved for the LIN frame transmission [1.23]. In the case of a minimum-length frame with one data byte and the checksum byte, the available tolerance is only 8 bit-times. Event-triggered frames will also prevent the use of a switch. If event-triggered frames from \textit{BUS\_M} and from \textit{BUS\_A} or \textit{BUS\_B} collide, the collision does not occur in the first data byte as specified. It is impossible for the switch to buffer the responses from slaves connected to the same bus with the master and to the other ports of the switch due to the LIN frame structure.
3.7 Summary

LIN is a low-cost and low-speed network. If higher performance is needed, another bus technology – e.g. CAN – can be selected. Performance is not a problem in typical LIN-systems, but in fully distributed applications the physical layer restricts the maximum number of nodes in a single physical bus and the achievable bus length with the maximum allowed number of nodes may easily become a problem. A hub concept similar to the one presented for CAN in Chapter 2 works also with LIN. Because of the differences in timing, the hub forwarding delay is too long compared with the bus propagation delay of LIN bit-time and therefore there is no sense to use a hub in real LIN networks.

The reduced hub concept provides operation at all bit rates. The internal performance has been enhanced by removing the capability to tolerate simultaneously received and overlapping dominant states between branches. As a result, these constraints prohibit the existence of collision in LIN systems equipped with the reduced hub. The reduced hub concept is not that useful because it cannot allow wakeup frames, which are an essential part of LIN communication. Furthermore, the reduced hub is not compatible with event-triggered frames supported by LIN 2.0 and later versions.

A multi-port master interface has been derived from the reduced hub. Typical LIN systems are master/slave systems, where sensor values are read from remote sensors to the master, processed and written from the master to the remote actuators. The multi-port master interface supports all features of the LIN protocol, all bit rates and standard bus length and up to 15 slave nodes in every branch. The only limitation is the visibility of the slave responses – each slave response is visible only for the master and locally in each branch, which is acceptable in typical, but not all master/slave, systems.

Network switches cannot be utilized in LIN systems because of the LIN protocol structure. Mixed use of time-based fields and form-based fields prevent the use of a switch. Moreover, a switch is an expensive component which is possibly not acceptable from an economic point of view in ultra low-cost LIN-networks.

Even if LIN is a low-speed bus system, SW-based solutions are too slow for physical layer expansions. Thus a reconfigurable HW will be needed for hub or multiport master interface implementations. LIN is also a low-cost bus system: the first multi-port master interfaces will be a CPLD-based
supplement to the current MCU-based LIN master nodes. The target application requirements determine which set of fault-treatment features is required and a platform is selected accordingly. In the longer term, a single-chip implementations with the MCU, with the multiport master interface included in the same FPGA is attractive.
4 Reconfigurable AD interfaces

A major problem of current mainstream analog-to-digital (AD) interface implementations is a fixed structure. At least in the case of the AD-converter (ADC) architecture, the number of parallel converters and the number of alternative inputs for the converters are fixed by semiconductor vendors. Moreover, the resolution and sample rates in most platforms have either one or two different operational modes. In some platforms the integrator may choose a resolution from two alternatives and automatically get the maximum supported sample rate. It is common to many control applications that a higher speed or higher resolution ADC external to the MCU has to be selected to meet the performance specifications set by the applications.

In the beginning of the introductory part of this chapter, existing mixed signal solutions are presented from the flexibility point of view. To expand the view, the most common ADC architectures and performance boosting methods are briefly introduced. At the end of the introduction, the internal structure and the capabilities of the two major configurable Field Programmable Analog Array (FPAA) platforms in the market, at the time of writing, are presented.

After the introductory part, both ΔΣ-modulation and successive approximation (SA) ADCs are implemented into both introduced major platforms as case implementations. 8 bit resolution and higher than 100Hz sample rate has been set as a target, because numerous real sensing applications in mobile control systems are currently operating in that resolution without any problems. After the implementation descriptions, differential (DNL) and integral non-linearity (INL) are measured to prove that the concept of the in-system configurable ADCs can reach the given performance target.

4.1 Existing mixed-signal solutions

Many semiconductor manufacturers today are trying to offer better low-cost 8-bit and mid-range 16-bit microcontrollers targeted for mixed-signal applications. There are several different approaches to providing a better set of peripherals to include AD and digital-to-analog (DA) interfaces. The
most traditional approach is a standard microcontroller with some additional analog and mixed-signal peripherals and signal interconnection capabilities [4.1] [4.2]. Generally, there are a number of analog multiplexers, amplifiers, comparators, voltage references, AD and DA converters. The interconnections between and behavior of these basic building blocks can be configured via configuration registers mapped into the memory space of the core microcontroller. In addition to the analog and mixed-signal peripherals, there may be various digital peripherals like counters, timers, SRAM, FLASH, external memory and communication interfaces. The most simple devices are actually closer to a feature rich ADC than a general-purpose microcontroller [4.3] [4.4]. There are on-chip controllers with either parameterizable fixed functionality [4.3] or a much reduced set of instructions and amount of application program memory [4.4].

More flexible implementations are the programmable system-on-a-chip (SoC) implementations called e.g. Field-Programmable System-on-a-Chip (FIPSOC™) [4.5] [4.6] [4.7] or Programmable System-on-a-Chip (PSoC™) [4.8]. These devices consist of both configurable mixed-signal and FPGA-style configurable digital resources in addition to the processor core. The PSoC™ is closer to traditional microcontrollers featuring a higher performance microcontroller and configurable resources primarily targeted for peripheral implementations [4.8]. In the FIPSOC™, the microcontroller core is mostly for system initialization and configuring of the configurable resources [4.5]. Configurable System-on-Chip (CSoC) [4.9] [4.10] [4.11] devices are closer to traditional microcontrollers having only configurable digital resources.

All microcontroller based implementations suffer from having fixed processor architecture and a set of dedicated software written for it, which needs to be maintained. Some of them use the widely supported 8051 [4.2] [4.5] [4.10] core, but this is only an 8-bit, relatively old, and low-performance architecture. Many other devices are based on a processor architecture supported by only one supplier [4.1] [4.4] [4.8]. Standardized cores, such as PowerPC® and ARM® are used in high-performance microcontrollers, also in [4.11]. With the system chips presented earlier, designers are very tightly coupled with the selected architectures, which may create challenges when the systems or product families are maintained and developed further and when larger scale derivatives are required by customers.

The basic idea of utilizing a dual-chip approach with separate digital and mixed-signal devices instead of a single-chip approach has been shortly
introduced for both larger processing platforms [4.12] and for intelligent transducer nodes [4.13]. The idea of a reconfigurable ADC has been presented in [4.14], but the presented implementation was targeted at boundary scan and built-in self-testing of mixed signal ASICs, not at general-purpose platforms. Reconfigurable logic has been used in data acquisition for control and increasing performance [4.15], but not as an integral part of AD conversion.

4.2 Introduction to ADC architectures

A detailed description of all different ADC architectures is not within the scope of this dissertation and therefore the reader is advised to consult the corresponding references for more details. Only the most common architecture categories without all possible nuances are reviewed. Only the counting, SA- and $\Delta \Sigma$-ADC architectures, which can be implemented into FPAAs [4.12], are presented in more detail. The fastest and simplest architectures are introduced first. They are the so-called Nyquist-rate converters, where the sample rate is at least twice the maximum frequency component of the input signal. The last introduced ADC-architecture is oversampling, where the typical sample rate varies from 16 to 256 times the maximum input frequency.

4.2.1 Flash converters

A flash converter is the most fundamental ADC. It consists of a reference ladder, one comparator for each voltage level and decoding logic for compressing the comparator outputs into a binary format conversion result [4.16] [4.17]. The number of comparators becomes a problem when the resolution is increased, and interpolation [4.18] [4.19] [4.20] [4.17] is a well-known method to reduce the number of comparators without losing the resolution. Another problem with flash ADCs is the increasing amount of decoding logic when the resolution increases. Folding is a method to reduce the amount of decoding logic [4.18] [4.20]. It is common to use both interpolation and folding in the same converter to minimize both the number of comparators and the amount of decoding logic [4.18] [4.20].

When the resolution of a flash ADC is increased, the required amount of resources increases strongly, too. To reach a higher resolution and still retain the resource requirements at a reasonable level, the conversion can be performed in multiple phases instead of a single phase. In pipelined converters [4.21] [4.22] [4.23] [4.24] [4.25], each stage consists of an ADC,
a DAC and an amplifier. Typically, low resolution ADCs, 1.5 to 4 bits [4.26] [4.22] [4.25] [4.27], are used in each stage. Interpolation [4.27] and folding [4.26] are the commonly used optimization methods also in the converter stages of the pipelined converters. After a sub-conversion, the conversion result is converted back to the analog format, subtracted from the original signal and the residual voltage is amplified and directed to the next stage. The final stage typically consists of a residue ADC only [4.25] [4.27]. Sample and hold amplifiers are used between the stages to enable pipelined operation. Pipelining is an efficient method to increase the sample rate by distributing the conversion delay among the converter stages. Finally, the conversion results of the converter stages are digitally combined into the final conversion result [4.21] [4.22] [4.23] [4.24] [4.25]. While combining the result, different kinds of error correction methods based on the use of extra bits [4.23] [4.24] [4.25] and calibration [4.22] are commonly used to improve the accuracy.

In pipelined ADCs the inter-stage DAC [4.28] and amplifiers decrease the accuracy [4.28] [4.22] [4.23]. In sub-ranging ADCs there are typically two stages – coarse and fine [4.19] [4.29] [4.17], and a subtractor between those. There is no amplifier between the stages decreasing the accuracy. A different conversion principle can be used in the fine stage instead [4.17]. Subranging ADCs perform the conversion typically in two phases and they can be categorized as a certain type of pipelined converter, where the previously mentioned error sources are avoided.

4.2.2 Reference DAC based converters

A counting or a slope converter [4.16], as presented in Figure 4.1, is based on a counter and a digital-to-analog converter (DAC), by which a stepwise increasing reference signal will be produced. The output value is stored in an output register when the reference voltage exceeds the input voltage during the conversion cycle. This architecture is very slow, but it can be expanded to a multichannel converter with extremely low resource overhead and in a very straightforward way – only a small part of the control logic CTRL and a comparator is needed for each additional channel. The reference generator counter CNTR and the DAC can be shared by the parallel converters. The output register has been excluded, because it may be combined with e.g. a digital filter or other processing elements after the ADC.

A counting converter needs one clock cycle for each step to be resolved, which makes the conversion time dependent on the input voltage - the lower
the input voltage, the sooner the reference voltage will reach the input voltage level and the faster the conversion will be. To accelerate the conversion, the reference voltage can be made to track the input voltage [4.16]. The control logic in the tracking converter compares whether the input signal is higher or lower than the reference voltage and either increases or decreases the reference voltage accordingly. Furthermore, it starts a new conversion cycle from the previous conversion result instead of zero. The tracking conversion will be faster the smaller the change is in the input voltage since the last comparison.

While the counting converter is slow and the conversion time of both counting and tracking converters depends on the input signal level or a change of it since the previous conversion, the conversion time of successive approximations (SA) ADCs is constant. It resolves one bit at a time, starting from the most significant bit and stopping at the least significant bit [4.28] [4.16]. Like counting and tracking ADCs, it is also based on varying reference voltage generation and comparing the varying reference voltage to the input signal. Unlike with the slope converter, each parallel SA converter requires its own successive approximation register logic SAR and can share only the conversion control logic CTRL, which makes the SA ADC less efficiently expandable to a multichannel converter than the slope ADC. A block diagram of the SA ADC is depicted in Figure 4.2

A modulated reference voltage [4.30] [4.31] can be used to achieve higher resolution DAC-based ADCs than with the on-chip DACs [4.32] [4.33]. Due to a net delay of the modulator and low-pass filter, conversion rates remain relatively low and decrease the sample rate further when the resolution is increased. A pulse-width-modulation (PWM) DAC is the best known approach, but also the use of a $\Delta\Sigma$-modulation DAC has been described [4.30].
In general, counting and SA converters need to have a sample and hold amplifier (SHA) to avoid quantization noise caused by the time jitter. It is also well known that a SHA is one of the biggest noise sources in an ADC [4.28]. If the conversion times of counting and SA ADCs can be decreased or the input signal filtered enough, the sample point jitter will be reduced and those converters can be used without the SHA [4.28].

4.2.3 Modulating converters

A so-called dual-slope converter performs the conversion in two steps [4.16]. In the first step, the input signal is integrated during a fixed time, typically until the result counter reaches an overflow. After integrating the input signal, a negative reference signal is integrated until the integrator reaches zero. The later integration time produces directly the conversion result. There is also an iterative variation of a dual slope converter [4.34]. One of the greatest features of this kind of ADCs is an intrinsic compensation of offset and delay errors without additional circuits or operations [4.34].

A $\Delta \Sigma$-modulation converter differs from the Nyquist-rate converters described before. The input signal is added to an inverted modulator feedback and the result is integrated. The quantized integrator output forms a bit stream, which is decimated to the conversion result. $\Delta \Sigma$-converters require mostly digital resources – decimator $DEC$, delay register $Z^{-1}$ and an inverse gain $-G$ – which makes them very interesting in reconfigurable processing platforms. A large pool of digital resources, e.g. CPLD or FPGA, can be used as an integral part of an ADC. There are multiple alternative implementations for the decimator and the filter, depending on the requirements and the amount of resources available. The most interesting feature is that a $\Delta \Sigma$-modulation ADC does not necessarily
require a SHA for the input signal [4.35] [4.36]. That will avoid the power consumption and the errors caused by the SHA [4.28]. A block diagram of a ΔΣ-modulation converter is presented in Figure 4.3.

![Block diagram of the ΔΣ-modulation ADC](image)

Figure 4.3: Block diagram of the ΔΣ-modulation ADC

### 4.3 ADC performance boosting methods

This section reviews the most common documented methods to improve the performance of the ADCs – dithering, parallel and interleaved ADCs. The methods are external to the ADCs and will boost the power-efficiency, accuracy or sample rate of an ADC. Furthermore, the methods are compatible with the ADC architectures. The idea of dithering is to add a small, well-known error voltage into the input of the ADC and subtract the known error from the conversion result [4.37]. Dithering efficiently compensates the errors, which are somehow dependent on the input voltage levels. If an input signal contains random noise, it can be utilized as dither signal and the resulting variation can be removed by averaging a set of conversion results [4.37]. The accuracy increase depends on the number and magnitude of the synthetic dither steps and number of results, over which the average value is calculated.

Improving accuracy with parallel sampling ΔΣ-modulation ADCs has been introduced in the literature [4.38]. These so-called ΠΔΣ ADCs [4.38] are based on parallel ΔΣ-modulation ADCs preceded by analog modulators for the input signal and digital demodulators after quantization. The idea of the modulation is to distribute the inaccuracy caused by the component matching characteristics between parallel ADCs. The net effect of oversampling and parallelism has been introduced in [4.39] and [4.40]. Either the sampling rate can be increased by reducing the oversampling ratio (OSR) or accuracy can be increased by keeping the OSR constant and increasing parallelism [4.39] [4.40].
Numerous benefits can be obtained by interleaved sampling. A higher sample rate with a reduced silicon consumption is one approach [4.41]. If power consumption is an issue, both power and silicon consumption can be reduced and the sample rate maintained by interleaved sampling [4.42]. Furthermore, the use of the time interleaving can reduce the power consumption by simultaneously decreasing the internal operating frequency of the sub-converters [4.26] or achieving the same sample rate with lower internal operating frequency by avoiding oversampling [4.27], depending on the system requirements.

4.4 Configurable general-purpose mixed-signal platforms

There are two main architectures in the area of fully general-purpose mixed-signal platform chip families currently in the market: Programmable Analog Chip (PAC) and Field Programmable Analog Array (FPAA). Both architectures have members offering a set of general-purpose mixed-signal resources surrounded by a configurable interconnect structure without the need for external discrete components. Members of both families are in-system configurable and operate at single +5V or +3.3V supply voltage common in current fully digital designs. Both architectures are compatible enough with each other, which makes them very interesting platforms for future fully configurable mixed-signal processing platforms [4.12]. The highest capacity general-purpose members from both families, ispPAC20 and AN221E04, have been selected for the evaluation.

General-purpose, configurable mixed-signal devices have been used in high-end filtering [1.25] and filtering development [4.43], intelligent analog front end [4.44] and rapid prototyping [4.45] [4.46] [4.47] [4.48] [4.49] [4.43] applications, but not in the volume production of control system processing platforms. Due to their run-time reconfigurability, FPAAAs can be used in analog signal processing [4.50] and [4.43] and even in fuzzy control [4.43] applications.

4.4.1 AN221E04

FPAAAs are mixed-signal devices comparable to FPGAs with relatively fine-grain internal structure implemented with switched-capacitor technology [4.51]. The devices are totally differential and offer a net differential signal
level higher than a rail-to-rail common-mode voltage level. There are various building blocks available, from which the designers can build application specific analog or mixed-signal designs. Most of the blocks offer analog arithmetic and filtering functions. There is also one SAR ADC in each four configurable analog blocks (CAB), but the outputs of only two of them can be simultaneously routed out of the FPAA. There are relatively many analog and mixed-signal resources in a single FPAA, if compared with only four inputs and two outputs. One input has a 1-of-4 multiplexer, but it can be controlled only via a serial interface. The FPAA devices have SRAM-based on-chip configuration memory and they need serial output configuration memories or a processor-based host system with non-volatile memory for configuration storage and load SW.

The Anadigm Designer® software has a very easy to use user interface, where predefined building blocks, configurable analog modules (CAM) can be inserted to the drawing area. The software supports the simultaneous design of multiple chips and multichip systems can be simulated with an integrated behavioral simulator. Only digital clock output waveforms cannot be visualized. The design software supports direct configuration download to evaluation boards and export of the design into a file, which can be stored to a storage memory.

### 4.4.2 ispPAC20

The ispPAC™ product family has many members [1.25] [4.44] [4.32], but ispPAC20 [4.33] is the most interesting member as a platform for flexible mixed signal systems [4.12]. The continuous-time architecture is very similar to the simple programmable logic devices (SPLD) [4.47] – there are large analog macrocells surrounded by a configurable routing area. There are two amplifier/filter cells, two comparator cells and one digital to analog converter (DAC) cell in the ispPAC20. There do not seem to be very many resources, but an excellent analog I/O-interface and digital control interfaces will enable an efficient utilization of the existing resources. Three differential analog inputs and three differential analog outputs provide enough analog connectivity. Digital control interfaces to the input multiplexer of one amplifier cell, polarity control input for another amplifier cell, selectable parallel or serial interface mode for on-chip DAC and comparator outputs provide excellent possibilities for the flexible use of the ispPAC20. The digital interfaces in particular serve a CPLD or an FPGA companion chip, where the digital part of a mixed-signal system can be implemented.
The PAC-Designer™ software is very straightforward to use, because it has templates for each family member. In the template, interconnections can be drawn by mouse and the settings of each cell can be modified in dialog windows. Only a frequency domain simulation is supported, which does not enable behavioral analysis of the designs. The software is capable of connecting to a single device via a point-to-point JTAG interface and exporting the design in a standard JEDEC file format supported by the general-purpose configuration download tools.

### 4.5 Case A: ΔΣ-modulation ADC

The ΔΣ-modulation AD-conversion principle was selected for testing because its resolution can be configured independently of the platform structure and it can be similarly implemented in both the available platforms. The first order modulator was implemented to enable dual-channel implementations in both platforms. 8-bit resolution was selected as a starting point to be able to compare results with ADCs based on fixed resolution on-chip ADCs or DACs.

#### 4.5.1 Digital part common to both mixed signal platforms

![Diagram of digital part of the ADC test platform](image)

In the block diagram depicted in Figure 4.4, the PRESCALER block divides the 1.8432MHz system clock into 153600Hz ADC clock and 9600Hz serial communications clock. The COMM block recognizes whether a character has been received from the RxD signal and starts the conversion result transmission to the TxD signal. OUTREG is an output register for buffering the conversion result of the ADC under test between the two different clock domains. Its controls depend on the AD conversion method and the control signals come from the digital part of the ADC. The output register stores the
conversion result to provide a buffered conversion result to the serial interface. The ADC under the test operates continuously at 150Hz sample rate independently of the serial communication. The ADC INTERFACE block of the block diagram is the digital part of the ΔΣ-modulation ADC. It is specific to each ADC and is separately introduced in the appropriate sections [4.52]. For the ΔΣ-modulation ADC it consists of the combined accumulate-and-dump decimator [4.53] and an average filter presented in Figure 4.5. Equation 4.1 presents the decimator functionality, where R_j is the quantizer output and x_i ∈ {0, 1} is a decimation output computed over the time window of length 2^n cycles.

\[ x_i = \frac{1}{2^n} \sum_{j=0}^{2^n-1} R_j \]  \hspace{1cm} (4.1)

The average value \( x_{av} \) over m results \( x_i \) is presented in Equation 4.2.

\[ x_{av} = \frac{1}{m} \sum_{i=0}^{m-1} x_i \]  \hspace{1cm} (4.2)

Equations 4.1 and 4.2 can be combined to form Equation 4.3.

\[ x_{av} = \frac{1}{m \cdot 2^n} \sum_{j=0}^{m \cdot 2^n-1} R_j \]  \hspace{1cm} (4.3)

According to Equation 4.3, the accumulate-and-dump decimator and the average filter can be combined by lengthening the sample time respectively. The variable m can be set to \( 2^k \) to make the implementation simpler [4.54].

\[ x_{av} = \frac{1}{2^{n+k}} \sum_{j=0}^{2^{n+k}-1} R_j \]  \hspace{1cm} (4.4)

Equation 4.4 specifies the combined decimator and filter implemented with two counters. In the test implementation an 8-bit decimator (n=8) was combined with the average filter over 4 conversion results (k=2), which gives 10-bit resolution for the counters. To simplify the division of the average filter, the divisor has been selected to \( 2^k \) [4.54], which allows a replacement of the division by 4 with a logical shift of 2 bits. The logical shift by 2 bits can be realized by using only the 8 most significant bits as a result.

The REGISTER block contains the zero order hold of the modulator. The hold also converts the single-ended quantization result back to the differential signal required by the mixed-signal devices [4.52].
a combined result counter of the accumulate-and-dump decimator [4.53] [4.21] and digital average filter. The $D$-$COUNTER$ is a synchronizer, which keeps track of the conversion cycle and produces the accumulate-and-dump control to the result counter $A$-$COUNTER$ in the modulator block and latch enable to the output register [4.52]. Those two counters implement the combined accumulate-and-dump decimator [4.53] [4.21] and average filter.

**Figure 4.5: Implementation of the combined accumulate-and-dump decimator and the average filter**

### 4.5.2 Mixed signal design in AN221E04

The evaluation board presented in Figure 4.6 was utilized for the AN221E04 tests. The required configuration, depicted in Figure 4.7, of the analogue part fits into a single FPAA. An anti-aliasing filter is needed in the signal input to limit the maximum frequency of the input signal low enough relative to the system clock frequency [4.52]. Voltage limiters are needed for the input voltage, the modulator feedback and the offset compensation input to prevent the inputs to overload the modulator and cause stability problems [4.53]. A sample and hold amplifier (SHA) is not absolutely necessary with a $\Delta \Sigma$-modulation ADC and was left out to reach the maximum compatibility with the other platform under test.

The first order $\Delta \Sigma$-modulator circuit in Figure 4.7 equals the block diagram in Figure 4.3. There is a summing block summing the input, feedback and the offset compensation voltages, an integrator integrating the sum voltage and a comparator performing the quantization of the integrated signal [4.52]. The most important parameter is the comparator hysteresis – it must absolutely be as close to zero as possible to minimize the quantization noise. Another important parameter is the integration time of the integrator, which is inversely proportional to the modulation frequency. The additional
voltage reference is needed for compensation of the offset error caused by the input amplifier in the AN221E04 evaluation board.

![AN221E04 test platform](image)

**Figure 4.6: AN221E04 test platform**

![ΔΣ-modulation ADC design with offset and gain compensations for the AN221E04](image)

**Figure 4.7: ΔΣ-modulation ADC design with offset and gain compensations for the AN221E04**

### 4.5.3 Simulation of AN221E04 based design

A behavioral simulation model was developed in Anadigm Designer® software to allow a coarse verification of the analog part of the design. The model has two FPAA chips: DSM-AMS on the right of Figure 4.8 is the actual design and the mixed-signal part of the ΔΣ-modulator, and TESTBENCH on the left replaces the digital part of the modulator containing the zero order hold and the feedback inversion [4.52]. The
design has to be simulated in the Anadigm Designer® because there is no interface to a standard mixed signal simulation environment [4.55] [4.56] [4.57]. After a few iteration cycles, the model was operating correctly. Finding the optimal settings for the integrator and selecting the system clock frequencies was much easier in the model. According to the simulation results in Figure 4.9, the modulator seems to operate correctly. The net duty cycle of the decimator output is approximately 50% at 0V input voltage, which is in the middle of the full ±3V range. The signals in Figure 4.9 are, from top to bottom: input signal, integrator voltage, comparator output and feedback from the zero order hold [4.52].
### 4.5.4 Results of AN221E04 based design

To measure the DC-characteristics, a servo-loop test method [4.28] was used. The test voltage generated by a test PC was converted to the industry standard 0-20mA signal to avoid disturbances during the measurements and to get as realistic results as possible. All results include the characteristics of signal transformation circuits, signal conditioning and the ADC under test. The result from an ADC test board was transmitted via serial line back to the test PC. Each ADC was tested with 16 cycles – 8 cycles from the minimum to the maximum input value and another 8 cycles from the maximum to the minimum value. The most essential target was to avoid random phenomena affecting the results. Each set of 16 cycles was measured during 8 hours to reveal the possible temperature dependencies. During that time the temperature of the semiconductors rose from approximately 10°C up to 50°C.

![Figure 4.10: DC characteristics of ΔΣ-modulation ADC implemented in AN221E04](image)

Figure 4.10 presents the characteristics of the ΔΣ-modulation ADC implemented in the AN221E04. The output code is in the X-axis and DNL and INL relative to the least significant bit are in the Y-axis. There are clear periodical error peaks caused by the flat areas of the response. The error behavior can be caused by digital noise, spike currents or too small...
capacitors of the FPAA for the current modulation frequency [4.53]. The error could be avoided e.g. by using a higher order modulator [4.53] or compensated by dithering [4.37]. The biggest design challenge was to find out a high enough integration time, which lead to decreased CAM clock rate. A better approach could be the use of a dedicated ΔΣ-modulation ADC CAM due to its better control over the FPAA. Both DNL of -100% to +400 lsb and INL of -200% to +250% lsb are too large to be used in real systems without further development.

4.5.5 Mixed signal design in ispPAC20

A dedicated test printed circuit board (PCB) as presented in Figure 4.11 was developed for the ispPAC20 tests. The analog part of the design is depicted in Figure 4.12 [4.52]. PACblock1 forms a low pass filter from the input IN1. An anti-aliasing filtering is not needed because of the continuous-time technology of the PAC platform. Instead, the functionality was implemented to an equal level with the AN221E04 design. PACblock2 is configured as the 1st order modulator. The filtered signal input is connected through IA3 and the modulator feedback signal from a fixed 3V reference, which is toggled between +3V and -3V under the control of the PC-input. OA2 is configured to be an integrator. CP1 quantifies the integrator voltage to the differential zero voltage and its state is directed to the CP1OUT output. In the ispPAC™-family, an unconnected input is terminated at the differential 0V.
Figure 4.12: ΔΣ-modulation ADC design with offset and gain compensations for ispPAC20

4.5.6 Simulation of ispPAC20 based design

Figure 4.13: Frequency-domain simulator of PAC-Designer™ does not serve the ADC design
The PAC-Designer™ software supports only a frequency-domain simulation depicted in Figure 4.13, which cannot be used for a behavioral simulation of the ADC designs. PAC-Designer™ files must be manually converted into digital form to enable behavioral simulation [4.12], which allows a behavioral system-level simulation but unfortunately does not help in finding the optimum parameters for the design.

4.5.7 Results of ispPAC20 based design

The DC characteristics of the ΔΣ-modulation ADC implemented with the ispPAC20 are presented in Figure 4.14. One can see that the measured DNL of -25% to +25% lsb and INL of -40% to +160% lsb are almost optimal. A small gain-error can be seen in the INL and it can be decreased by compensating the gain error in the analog front end or by computing after the conversion.

![Figure 4.14: DC characteristics of ΔΣ-modulation ADC implemented in ispPAC20](image)

4.6 Case B: Successive approximations ADC

SA conversion has been included in the tests, because it is also supported by both platforms. The AN221E04 contains four on-chip SAR ADC blocks
and in the ispPAC20 the SA ADC can be constructed from the DAC and one comparator. The main problem of SA ADC is that the on-chip building blocks of the selected platforms limit the maximum achievable resolution to 8 bits [4.52].

### 4.6.1 Digital part for AN221E04

A digital part for the SA ADC design with AN221E04 is depicted in Figure 4.15. It utilizes the same serial interface and output register modules as the previous designs depicted in Figure 4.4. A separate SA ADC result reception block *SHREG* converts the serial conversion result into parallel byte which is sent to the host PC through the serial interface. An asserted SYNC-signal enables the read in of the result byte from DATA-bus and the falling edge of the SYNC-signal detected indicates the end of the conversion. The output latch enable (OLE) signal is generated by the *EDG* block, based on the falling edge of the SYNC signal [4.52].

![Digital interface for SA ADC implemented in AN221E04](image)

### 4.6.2 Mixed signal design in AN221E04

Using the on-chip SA ADC cannot be easier - only a dedicated SA ADC CAM-block is needed. An anti-aliasing filter is needed for the input signal and the output cell is configured as a digital output like the comparator output in the ΔΣ-modulation ADC. The SA ADC CAM requires two clocks – a conversion clock and a synchronization clock, which is the conversion clock divided by 16. If required by an application, more CAMs can be added to adjust the gain and the offset of the input signal or according to Figure 4.16 [4.52]. Also filters can be added if required by an application. There is also a SHA CAM-block available, but it can be synchronized only with the internal ADC synchronization clock. Because there is no SHA functionality available in the ispPAC20, it has not been used with the AN221E04.
4.6.3 Simulation of AN221E04 based design

Figure 4.17 presents the simulation results of the AN221E04 on-chip SA ADC. The topmost signal is a 0V input voltage, the middle signal is the conversion result synchronization signal (SYNC), and the lowest signal is the conversion result (DATA) – the most significant bit first. Both SYNC and DATA are synchronized to the conversion clock signal, which is assigned to the output and to which the conversion result bits are
synchronized. The clock output cannot be presented in the simulation. According to Figure 4.17, the conversion time is 4µs and the sample rate is 125 kHz [4.52].

4.6.4 Results of AN221E04 based design

The results of the SA ADC presented in Figure 4.18 match well with the specifications of the AN221E04 [4.51]. The DNL is -30% to +30% lsb and the INL is -100% to +100 lsb. In the first cycle the INL of 0 to +100% lsb has been reached due to the gain variation caused by the shunt resistor converting the input current to the input voltage and gain control resistors of the differential amplifier circuit before the AN221E04.

![Figure 4.18: DC-characteristics of SA ADC implemented in AN221E04](image)

4.6.5 Digital part for ispPAC20

According to Figure 4.19, the digital part of the SA ADC implemented with the ispPAC20 requires the successive approximation register SAR itself and a control logic CTRL. A serial interface, a simple UART, has been reused from the other ADC test designs. The SA ADC operates continuously at 19.2kHz sample rate [4.52].
4.6.6 Mixed signal design in ispPAC20

The mixed-signal part of the SA ADC constructed in the ispPAC20 is presented in Figure 4.20 [4.52]. PACblock2 is configured as an anti-aliasing filter and it can be used also for offset and gain compensation if needed. The comparator PACell CP1 is used for comparing the filtered input signal with a reference voltage generated by the DAC PACell. The default value of the DAC was configured to -3V, which equals the input code 0x00. The parallel input was activated for the DAC to allow an effective interface for the digital part of the ADC design. As mentioned in case A, the PAC-
Designer™ supports only a frequency domain simulation, which cannot be used for the functional simulation.

4.6.7 Results of ispPAC20 based design

The DNL of -20% to +10% lsb and the INL of -50% to 50% lsb of the SA ADC presented in Figure 4.21 are almost ideal. The results show that the accuracy of the scaling and filtering blocks and the routing paths is high enough to reach the accuracy specified for the on-chip DAC and the comparator. It also indicates that the noise level of the ispPAC™ architecture is low enough for the ADC-designs.

![Graph](image)

Figure 4.21: DC characteristics of SA ADC implemented in ispPAC20

4.7 Summary

Traditional microcontroller-based control products are based on fixed configuration ADCs with tightly limited resolution, accuracy, sample rate and number of parallel sampling channels. The restrictions typically lead to either increased overhead throughout the product family, compromises with the product features, or an increased number of product variants. An alternative approach has been presented in this chapter to provide better in-system configurability according to the requirements set by various control
applications. In-system configurability allows a more flexible and efficient use of the fixed amount of resources.

Current mixed-signal in-system configurable chips offer a relatively low amount of resources [4.47], but single-channel basic ΔΣ-modulation and SA ADCs were successfully implemented into both major mixed signal chips in the market. The tested ADC converter architectures were limited to ΔΣ-modulation and SA, because those were supported by both HW under evaluation. Because the target was only to evaluate whether it is possible to get in-system configurable ADCs working, very basic implementations were used. For ΔΣ-modulation ADC, the 1st order modulator, average filter and accumulate-and-dump decimator were selected. In SA ADC only a single 8-bit reference DAC was used. The main focus was on ΔΣ-modulation ADC because of its flexibility, but also SA ADC has been included to get comparison results and a better figure about the overall accuracy of the in-system configurable mixed-signal HW. All ADCs were tested without SHA because it was not supported by both HW. The example ADC designs showed that good digital interfaces [4.12] improve the usability of the mixed-signal platforms. Moreover, support for a behavioral simulation provided by a design tool significantly reduced the overall design time.

Based on the case examples presented in this chapter, the concept of ADC implementation into in-system configurable hardware has been proved to work in practice. Servo-loop measurements were performed for all tested implementations. To avoid random results, 8 cycles with ascending and another 8 cycles with descending input signal were measured. Each set of 8 cycles was measured over 8 hours to reveal possible temperature dependencies, if such exist. 8-bit accuracy set as target was reached in 3 out of 4 tested implementations, SA ADC implementations on both platforms and ΔΣ-modulation ADC implemented on the ispPAC20 platform. The selected approach of ΔΣ-modulation ADC did not work in an optimal way in the switched-capacitor AN221E04. Table 4.1 shows a summary of the results and comparison values of a few experimental and commercial monolithic ADCs. The results presented in this dissertation cover the whole AD-interface including current to voltage conversion, signal scaling, anti-aliasing filtering and the ADC itself to provide as realistic results as possible.

The absolute accuracies of all the tested converters, except one, are according to the target when DNL is examined. Somewhat worse INL
values can mostly be explained by a small residual gain error, which can be removed either by fine-tuning the signal scaling or after the conversion. Based on the results, repeatability is amazingly good. It can be concluded that there is great potential in in-system configurable ADCs. The results, especially from the continuous-time $\Delta\Sigma$-modulation ADC implemented with the ispPAC20, were promising because only very basic implementations have been tested. The results are encouraging to continuing the development of the in-system configurable $\Delta\Sigma$-modulation ADC in the future with e.g. higher order modulators, more sophisticated filters and decimators to reach a better accuracy and lower OSR. Higher resolution counting or SA ADCs can also be developed in the future with both mixed-signal platforms by using a modulating reference DAC [4.30] [4.31].

Table 4.1: Summary of results and comparison with some monolithic ADCs

<table>
<thead>
<tr>
<th>ADC</th>
<th>DNL (% lsb)</th>
<th>INL (% lsb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>$\Delta\Sigma$/AN221E04</td>
<td>-100</td>
<td>400</td>
</tr>
<tr>
<td>$\Delta\Sigma$/ispPAC20</td>
<td>-25</td>
<td>25</td>
</tr>
<tr>
<td>AD7863 [4.58]</td>
<td>-50</td>
<td>50</td>
</tr>
<tr>
<td>ADS1606 [4.59]</td>
<td>-15</td>
<td>15</td>
</tr>
<tr>
<td>SA/AN221E04</td>
<td>-30</td>
<td>30</td>
</tr>
<tr>
<td>SA/ispPAC20</td>
<td>-20</td>
<td>10</td>
</tr>
<tr>
<td>AD9283 [4.60]</td>
<td>-25</td>
<td>25</td>
</tr>
<tr>
<td>ADS7827 [4.61]</td>
<td>-10</td>
<td>10</td>
</tr>
<tr>
<td>Pipelined [4.26]</td>
<td>-20</td>
<td>43</td>
</tr>
</tbody>
</table>
5 Volume and life cycle of control electronics

Control system electronics have much longer life cycles and lower production volumes than consumer or telecommunication electronics, which are leading technology and component development. The use of common components has raised some problems because the typical product life cycle and production volume characteristics are opposite. This chapter presents how in-system reconfigurable mixed-signal hardware platforms can significantly improve the maintainability of low-volume long-life control electronics.

The chapter begins with an introduction to the traditional life cycle model and the scattered volume model applicable to the production of current control system electronics. The background for the models is presented by referring to published solutions concentrating on fixed hardware platforms and utilizing software methodologies to adapt to platform changes. Software maintenance is not within the scope, though some excerpts are included to clarify the application field. After a presentation of existing models, the most typical hardware maintenance methods are introduced. This introduction concentrates on maintenance, because the majority of the applications are mature. Therefore, the main focus of the design work lies on maintenance of existing platforms or otherwise developing new platforms largely based on existing ones. The methods are introduced with examples because most of them have been widely used but not documented and classified. The maintenance methods are divided into technical and administrative methods to emphasize the importance of the design for maintenance approach in all levels of organizations.

After the introductory part, the traditional life cycle model is enhanced to better correspond with the life cycle of long-life control electronics. The length of the maintenance phase in the resulting improved life cycle model has been extended and hardware generation changes have been included to match commonly experienced scenarios. At the end of the section, the resulting maintenance and production problems of long-life fixed platforms are described. The last section discusses the improved maintainability and producibility resulting from the utilization of the reconfigurable platforms.
At the end of the last section, the results are summarized as extended life cycle and volume models.

5.1 Traditional life cycle and volume models

The traditional life cycle model with a single set of phases is widely discussed in the literature [1.3] [1.4]. Figure 5.1 depicts a summary of this traditional life cycle model. Requirements acquisition, conceptual design, detailed design, production, use and support and retirement phases are correct by themselves but the model concentrates on system instances rather than system versions or systems. The approach is systematic from a single system point of view, but not from a component or production point of view. Maintenance design requirements are not emphasized enough in the literature to be realistic and only the maintenance of system features has been covered [1.3]. An evolutionary life cycle model has also been presented [1.4], but its viewpoint is the development of the system behavior, not maintenance of production and service capabilities of long life systems over a certain time window. The presented viewpoint is applicable for large and unique systems, for which enough spare parts can be stored to cover the entire lifetime of the system due to the relatively low quantities. On the other hand, spare parts for high-volume systems can be redesigned, because there is a large enough production volume strongly reducing the total maintenance cost assigned for a single unit. Especially examples provided by systems engineering [1.3] are system-instance oriented. The use of long-life components is suggested, but the challenge is not emphasized enough and life cycle problems of real hardware (HW) components are not covered. The existence of "long-enough life" components is erroneously assumed. Methods for design time change management based on system knowledge are presented in literature [5.1], but the applicability of the methods to long-term maintenance is excluded.

Most of the systems can be supported quite well, but only during their originally designed life cycle. A well-known and well-designed life cycle is also set as a pre-condition in the literature [1.3] [1.4]. The exact life cycle cannot be unambiguously determined for long-life control electronics. Problems tend to begin when the life cycle is extended or the production quantity is increased from the original. Extending the life cycle without increasing the overall production quantity increases the component obsolescence risk and also increases the payback time of the design and maintenance investments. It is possible by, e.g. replacing part of the volume with new versions. Keeping the defined life cycle but increasing the overall
production quantity introduces a higher component availability risk the later the increase is decided.

![Figure 5.1: Traditional life cycle model presented in the literature](image)

Traditional product strategies are based on fixed hardware platforms, where some minor physical adaptations can be controlled by software (SW), but all critical structures like the processor and the peripherals are fixed. Due to the short life cycle of critical components – like processors – different generations are based on different core components [5.4]. With traditional fixed hardware systems an update of the old SW to the new HW is the most common upgrade. Almost always at least some minor modifications have to be made and also tested, even if a standardized HW is used. The next common type of update is to get the new SW running also in the old HW. This works as long as there is not only enough processing performance and memory available, but also a possibly required set of peripherals in both HW generations. The most challenging case is when both HW and SW have to be updated simultaneously. Based on experience, this has to be done because e.g. the old operating system, compiler, SW component or library does not support the currently available HW or if there is not enough memory or processing performance in the current HW.

Typically, SW forms the core of the system and contains the actual system behavior. The statement "Ready SW does not wear out" is true, but only for a fixed time, during which the HW platform and the SW remain constant. Any change in the software is always a change in design [1.4] and needs not only to be implemented but also tested. A typical product platform contains an operating system with device drivers and various libraries, which will also change. The impact of fixed HW change is much more difficult to predict, because the HW may contain bugs affecting the SW. The bugs may be, e.g. accidentally unsupported operating modes of the peripherals, peripheral control register, or clock tree problems. The platform may also contain HW changes, which can affect the SW behavior such as bugs or
changed features in the instruction set. Many of the bugs are instruction-specific, but some of them depend on both the instruction and the data. It can be summarized that each application SW requires to be tested and fixed after any change.

An evolutionary life cycle has recently come into use in strongly evolving market areas. The key technology in the published case examples are in-system reconfigurable platforms, FPGAs, because they are based on digital technology [5.2] [5.3]. The first approach has been risk management – standards are strongly developing and it may not be possible to determine the dominating standard in the beginning [5.2]. A reconfigurable platform enables rapid entry to the market and it can be updated afterwards when standards are developing or changing and when new features need to be added. Another approach is market-oriented. Reconfigurable platforms enable rapid time-to-market with a reduced set of features. Later, the new features can iteratively be updated, also for products already sold to the customers [5.3]. Reconfigurable platforms enable the first product in the market to be smoothly developed into the best product in the market [5.2]. The flexibility of an FPGA enables utilization of user feedback from the existing products for the development of new versions. This flexibility significantly extends the life cycle of products emerging onto the market without significant costs for the manufacturer or customers [5.3]. Some of the reconfigurable platforms can be updated automatically over various communication media if such exist in the platform [5.2] [5.3].
documented evolutionary approaches are too short-term and too focused on new product development to be able to solve the long-term challenges within the scope of this dissertation. A further problem is that mixed-signal problems cannot be solved only with FPGAs.

Experience has shown that for unknown reasons many people have a strong need to intentionally differentiate the requirements between applications rather than look for potential synergies. This applies both between different product lines inside a single company and between companies. The tendency for strong differentiation of products has escalated volume scattering and is wasting the development, testing, maintenance, service, and education effort among products, product lines and system integrators. The net effect of strongly varying application requirements and fixed mixed-signal HW platforms amplified with an intentional differentiation leads to the scattered volume model shown in Figure 5.2.

5.2 Technical maintenance methods

Maintenance methods are divided into two logical categories. Technical methods cover the practical ways to maintain the control electronics products. The methods concentrate on everyday design and manufacturing work and do not require any strategic decisions. Administrative methods covered later will coordinate the use of technical maintenance methods. The introduced methods have been widely used, but they have not been so well documented. The methods are presented as case examples.

5.2.1 Redesign

5.2.1.1 Partial redesign

Partial redesign can be utilized in cases of hurry or when a small redesign can solve a problem or problems which have been met. Sometimes a new component is pin-to-pin compatible with the existing one, when only the bill of materials needs to be updated and a system needs to be thoroughly tested. More often, the printed circuit board (PCB) has to be partially redesigned because of the different package of the fully compatible component or because a different circuitry is needed as a replacement. To save costs, multiple changes are often tied together into the same redesign cycle. Sometimes the main PCB does not need to be changed and the change can be designed as a piggyback or a multichip module instead. The biggest risk of partial redesign is that due to the lower costs of a single cycle, partial redesign cycles may unintentionally delay a plausible bigger
design cycle, mess up the structure and decrease the overall reliability of the product.

An example of a partial redesign is presented in Figure 5.3. There were only a few years of active production time left for the product, which is why expensive SW-modifications had to be avoided. A hardware redesign has good predictability and limited costs per single cycle, but software platform updates are more risky and unpredictable. The more there are different programs written for a single HW platform, the higher the costs will be in the case of obsolescence of a processor, a microcontroller or a critical peripheral device. An ADC on the left side became obsolete and pin-to-pin compatible types did not exist. The 28-pin dual-in-line (DIL) package physical interface and +5V signal levels could be utilized and the old ADC was replaced with a module consisting of a replacement ADC. The serial host interface of the new ADC has been adapted to the original parallel interface with a small CPLD. The replacement concept is so generic that it can easily be redesigned in the case of either ADC or CPLD obsolescence.

Figure 5.3: Original system (left) and partially redesigned (right) part of the system

5.2.1.2 Total redesign

A total redesign is needed when at least one key-component becomes obsolete or difficult to purchase and there are no second sources available in the market. Other reasons for a total redesign may be the reduced availability of multiple components or known serious problems in the existing design. Sometimes the unit cost of the product can be significantly reduced by changing to totally new components and modern manufacturing process.

The original example PCB, on the left of Figure 5.4 had some randomly occurring grounding problems due to a poor layout. Power transistors were already specially stored because they were not manufactured anymore. After encountering obsolescence of some standard logic ICs, only a total redesign could be considered because there were at least five years of active production and 10 years of spare part support left. It was not feasible to replace the standard logic in DIL housings spread over the PCB with the piggyback solution described before because the obsolete logic was not
available in surface mount housings. Most components in the new version have three or more alternatives and the layout was redesigned with two more copper layers and an improved layout. The new version became more convenient and less expensive to manufacture and reached improved EMC characteristics. Replacement of all the discrete logic with a single CPLD provides the core functionality independent of the CPLD type and manufacturer. Thus, the later revision cycles will be rapid and inexpensive.

Figure 5.4: Original (left) and totally redesigned (right) board

5.2.1.3 Partial and total redesign

Quite often a partial redesign is needed to solve an unexpected problem or problems and to keep the production active, and in that way get enough time for a total redesign. Typically, unexpected problems are unnoticeable component obsolescence and slightly changed component characteristics due to, e.g. production process change. Another well-proven approach is to prepare a total redesign by designing and verifying the new design in smaller fragments. Such an approach decreases the overall risk by enabling more parallel redesign and verification of the changes to some extent. The verified sub-solutions can just be put together in the total redesign phase.

An example of a partial redesign is shown on the left of Figure 5.5. There were already some small fixes made by additional discrete components and an unexpected obsolescence of the surface mounted standard logic triggered a total redesign. The required time for the total redesign was arranged by designing a wired piggyback, shown in the middle of the board. The obsolete standard logic behavior was rapidly implemented into the CPLD in the piggyback to ensure uninterruptible deliveries of the end product.

A total redesign was needed because the manually assembled temporary fixes and the wired piggyback were too expensive to assemble and caused
some random problems in the production. A total redesign was the only option because there were also some specially stored components and other components potentially becoming obsolete soon. There were at least 5 years’ active production and 10 years’ spare part support left. Thus, the maintainability of the new version had to be maximized. The excellent flexibility of the FPGA was utilized to get rid of the functional dependencies on the special purpose semiconductors. The overall accuracy was significantly increased by replacing the old analog circuits with digital processing and manual fine tuning was not needed anymore. As in the previous example, most of the components could be selected so that there are three or more alternative types for each. Connectors were left as through-hole mounted to maximize the production synergies with other members of the same product family. All other components have been changed to surface mounted to maximize the production efficiency. The totally redesigned board is presented on the right of Figure 5.5.

![Figure 5.5: Partial redesign implemented as a wired piggyback (left) and totally redesigned board (right)](image)

### 5.2.2 Careful component selection

Selecting components for a design is crucial for the product future. The most important criterion, of course, is suitability for the application. Minimization of the number of components has been recommended in the literature [1.3], but in most cases it is not so simple. All components should be so common that there are fully compatible types available from at least two manufacturers. The more there are alternatives, the safer is the selection from both availability and cost point of view. There are masses of short-lived components available, mostly for the consumer and telecommunication markets, and use of them in long-life automation products should be avoided to reduce the obsolescence risk [1.3]. Some of
those components look perfect for many solutions, but still an alternative approach is recommended to ensure long-term maintainability. Separating component life cycles from the system life cycle [1.4] by using standard components instead of a single "perfect" one is a much better choice in the long term, regardless of the slightly higher cost, complexity or number of additional components required.

In the case of certain common components, like resistors, capacitors, diodes, transistors, operational amplifiers, memories and transceivers, the re-use of a few standard types and values is strongly recommended. The component purchasing effort will be reduced, because there are many alternative types for each component having equal characteristics. Fully compatible alternative component types also enable the adaptation of many products or product families to use the same components and benefit from increased total purchasing quantities. Moreover, non-productive actions like changing settings in the assembly line can be reduced or even avoided due to the reduced number of different components.

It is important to identify in which phase in their life cycle selected technologies and components are [1.10]. On the one hand, relying on well-proven technologies or components may look very safe from the technology point of view, as there is limited remaining lifetime. On the other hand, emerging technologies and components are in the beginning of their life cycle and thus have potentially a longer remaining life cycle, but there is no experience of their feasibility. It is always worth analyzing, which alternative provides the smallest overall risk.

5.2.3 Component storing

Normal component buffering in the production is an integral part of the production process, not maintenance, and therefore it does not fall into the scope of component storing. The first possible reason for specially storing one or more components is an exceptionally increased delivery time. Careful component selection should prevent most cases, but there are still some unavoidable risks remaining. Warnings of delivery time increases should be received in time, with an intensive component market follow-up. It is very difficult to determine how many components should be stored.

The second reason for special component storing is a last-time buy (LTB) or end of production announcement from the component supplier [5.4]. The most challenging action is to determine the end product quantity at which to store components. Without a good product life cycle plan it is impossible to
obtain a correct answer. There are many risks even in the best case. If too few components are purchased, it may be impossible to find more components on the market, or the change from old to new product cannot be performed without interruption. If another component or components become obsolete during the special storage time, they also make all the components in the storage obsolete. With a good product life cycle plan, special storage needs to cover only full production during the ramp-up time of the next generation product and possibly a defined number of spare parts.

Finally, if the availability change announcements have been missed for some reason and components are not anymore manufactured, it is very challenging to keep the production active. The availability of the component or components is fully random. Furthermore, the quality of the components cannot be guaranteed. Storing components over a long time may cause degraded soldering quality, or the components need additional chemical treatments before soldering. Regardless of the treatments, there is still a risk of decreased level of quality.

5.2.4 Modular design

Modular design is a well-proven technology to isolate the invariant parts of the design from the strongly variable parts. The literature concentrates on the modularity of mechanical structures [1.5] [1.6] [1.9], but also the modularity of electronics is shortly covered [1.7]. In control electronics, modular design improves production efficiency by enabling higher production quantities for the most expensive and most complex part of a product or a product platform. Existing design modules should be used in as many positions of a system as possible and in as many systems as possible to maximize the benefits of modular design [1.4]. Even more important results of modular designs are standardized interfaces between the modules. As long as those interfaces remain constant, all the modules can be maintained and developed independent of the others [1.5]. Those interfaces can be either legacy or implemented according to generic standards. Modularity can be utilized in many levels, but only the levels inside a single electronic control unit fall within the scope of this dissertation and are presented in this section.

5.2.4.1 Board level modularity

During the history of electronic systems, high complexity and well-defined functions are typically isolated into separate boards to provide the modularity required by applications and technology development. Board
level modularity is also well-known in the literature [1.7]. Numerous display adapters, data acquisition boards, communication and disk controllers of computers are the most obvious examples of board-level modularity. Standardized bus systems, like ISA [5.5], EISA [5.5], PC/104 [5.6] and VME [5.7], have typically been used for interfacing the boards with each other. The example in Figure 5.6 presents board-level modules of a distributed I/O-device family. The family consists of a single processor board and 3 alternative I/O-boards, from which 3 different I/O-units can be constructed. In the example there is an application specific serial interface between the boards, which unfortunately has been a tradition in many embedded systems.

![Figure 5.6: Processor board (top left) and three alternative I/O-boards](image)

5.2.4.2 Use of piggybacks and mezzanine boards

Many kinds of piggybacks and mezzanine boards can be used to increase modularity in a single board domain [1.7]. A module can have an interface and a form factor of an IC, like in Figure 5.7. Other common module
interface and form factor standards are, e.g. Multimedia Card (MMC) [5.8], Secure Digital (SD) [5.9] and Compact Flash (CF) [5.10]. There are also several industrial mezzanine board standards like MODULbus [5.11], M-Module [5.12] and VME Local Bus (VLB) [5.7] defining the add-on board dimensions, signalling interface, connector types and pinouts. Typical mezzanine boards offer various communication and discrete I/O interfaces, by which the control product can be adapted to the different applications. Another common category is memory cards, by which amount, type and speed grade of the memory can be configured. The board size can often be reduced with high density memory modules compared with the use of individual memory ICs.

Figure 5.7: Example of piggyback

5.2.4.3 Assembly options
Assembly options are widely used when two or more alternative functions have to be supported, or when the same function has to be parameterisable or scalable with the same board layout. In many cases product variants are implemented as assembly options and most of the production can be made with a single setup. The example in the upper part of Figure 5.8 presents a traditional assembly option arrangement with IC sockets and jumpers providing the maximum achievable flexibility. The alternative serial interface type can be selected without tools just by changing a transceiver and jumper positions. The same principle has also been used for adapting the mixture of different memory types and sizes.

Assembling different sets of components is also widely used when the function is fixed but one or more characteristics have to be scalable. One of the most used arrangements has been scaling the memory size with 24-, 28- and 32-pin JEDEC DIL packages, which can be assembled in the same pad layout. The lower part of Figure 5.8 presents an example where surface mounted assembly options are used to provide more alternative components for the same function. There are two sub-types of the key component IC10 with the same pinout. As both variants expect partially different connection, either R72 or R73 is assembled, depending on the selected type of IC10. The same concept has been widely used in the latest designs, where
multiple surface mounted PCB decals are used instead of IC-sockets to support, e.g. alternative component or package types.

![Image](image1.png)

**Figure 5.8:** Use of assembly options with sockets (left) and variable assemblies (right)

### 5.2.4.4 Circuit entities

Re-use of components is recommended because of high development costs [1.4]. Most examples in the literature cover SW-components, but circuit entities are similar to the SW - once they have been designed and verified, they can be efficiently laid out differently in different products [1.7]. The re-use of some components is, however, limited by their reduced life cycle [1.4]. There are many component categories providing well-standardized functionality, but with a different package type, pinout or a set of required additional discrete components. The challenge can be solved by designing and verifying the whole circuit entity consisting of a set of components. Then the designing of a new product can be improved by re-using the proven circuit entities. In the example in Figure 5.9, most of the circuit entities are shared with two or more products. Products C and D share the same processor entity including the MCU, crystal circuit and programming interface. They also share the CAN-interface entity consisting of transceiver and transient protection components. The same serial interface with the transceiver and transient suppression components is shared by products C and L. All three products utilize an equal power supply with transient suppression components and a switching regulator. While the circuit entities are equal, the layouts are slightly different in each product.

![Image](image2.png)

**Figure 5.9:** Products C (top left), D (right) and L (bottom left) sharing the same circuit entities

133
5.3 Administrative maintenance methods

Administrative maintenance methods are typically performed by company or product management. They are more about making strategic long-term plans and decisions than performing the practical design and maintenance work. Administrative methods support and improve technical methods, they do not replace them. Existing product documentation is the main prerequisite for successful maintenance administration. Re-engineering of an existing product is very expensive and time-consuming [1.4].

5.3.1 Reactive maintenance

Based on experience of electronics maintenance, the reactive maintenance is, unfortunately, the most common method being used. The major problem in reactive maintenance is the short-term or even missing focus. Something has to have happened before any action, or in typical case actions, takes place. Typically, actions take place immediately after the triggering appearance. Therefore, reactive maintenance activities are always late. For new and middle-aged products it is often problematic, because short-term focus continuously expects the solving of one problem at a time, which is time-consuming, expensive and inefficient. In the worst case, solving single problems may eventually obscure the product structure. When the product is near the end of its life cycle, it is better to use reactive maintenance than more systematic maintenance and continuous improvement [1.3].

5.3.2 Proactive maintenance

Proactive maintenance is a much better approach than reactive maintenance, because the product production is continuously followed up and the maintenance activities are performed to prevent the realization of both technical and obsolescence risks. Proactive maintenance has a longer-term focus and enables the collecting of small improvements into bigger maintenance cycles to reduce the overall effort and costs. Unlike with reactive maintenance, there is seldom hurry with proactive maintenance activities. It enables accurate plans and scheduling, which results in lower costs, long enough testing and a seamless version change in production. The literature presents a method called Reliability-Centered Maintenance (RCM) for designing a long-term proactive maintenance plan for products [1.3]. The main idea of RCM is that maintenance design is started as early in the product life cycle as possible, but it is never too late to move to proactive maintenance. The earlier the maintenance design is started, the
better will be the results achieved. The best results can be reached when the maintenance design is performed in parallel with the actual product design.

### 5.3.3 Generation skipping

The capability and the availability of current technology often meet the current needs well. Thus, there is no sense in taking all the possible small technological steps. One or multiple technology generations can be skipped instead [1.1]. Operating systems and office programs were used as an example [1.1], but generation skipping is also a well-known practice for e.g. real-time operating systems and compilers. In the example in Figure 5.10, there is enough overlap between generations one and three to enable the adoption of generation three before the retirement of generation one. Depending on the technology or the product, more than one generation may be skipped. A lot of time, effort and money can be saved during a generation skip if there is enough overlap. The savings can be invested in the development and testing of, e.g. new features for a next generation product or utilized to reduce the price of the end product [1.1].

![Figure 5.10: Example of generation skipping](image)

### 5.3.4 Standardization

Standards have already been referred to earlier, but the importance of standardization will be emphasized in this dedicated section. The use of standardized system components, interfaces, and technologies significantly eases not only the system design but also the system maintenance [1.3]. A general challenge is to find and sometimes guess the right standard or standards from the mass, because also the feasibility of a standard varies as a function of time and application field. Supporting standard technologies, components and tools will ease the design and maintenance by enabling the use of commercial off-the-shelf (COTS) system components [1.1]. Such an approach expects standardized interfaces and services from the system components. Moreover, relying on standards enables utilization of the COTS design and production tools supporting the standardized processes to
improve productivity and avoid process, component and tool development work [1.1].

5.3.5 System architecture

Standardization typically covers product and process technologies and interfaces. System architecture design goes much further by analyzing the entire product line or multiple product lines and by finding similarities among the different products or product lines. Based on analysis, the systems are divided into sets of common subsystems. The system approach minimizes problematic interdependencies between subsystems [1.1] [1.4]. The result in typical applications is a set of subsystems with standardized interfaces for information interchange between the subsystems. A system architecture can efficiently guide designers to utilize COTS component offering [1.4]. A properly designed architecture can significantly reduce the maintenance work by defining a modular structure. In a modular system any problem is located in a module and once it has been fixed inside the module the solution will be propagated to all the products containing that module [1.3]. A common architecture also helps to reduce the maintenance work by enabling re-use of the standardized design entities between the modules [1.5] [1.4]. Examples of re-use have been presented in Section 5.2.4, but a well-defined system architecture improves the efficiency of re-use by enabling the systematic recognition and management of re-usable entities globally.

5.3.6 Product platform strategy

As mentioned before, reacting means being continuously late. Proactive maintenance tries to predict the risks and solve them before realization. Standardization helps design, production and maintenance and a well-defined system architecture improves the efficiency of the preceding methods. These methods can be further improved by long-term strategic planning. A product platform combines the common elements of a product line or multiple product lines. It defines the underlying technology elements and helps in adopting the right standards, processes, components and tools [1.10]. A platform is only the lowest denominator of technology in a set of products, not an independent product. Instead, numerous products can be derived from the generic platform [1.10]. A platform needs not only to be designed: it has to be maintained, too. A platform is also strategic thinking – it is important to know where a platform is in its life cycle and when it is time to replace it with a new one, and how or is it better to extend the life of
the major platform instead of replacing it [1.10]. To be able to make the decision, the reasons for the decline of the platform life cycle shall be well understood.

Platform strategy focuses on key decisions made at the right time. It enables products to be deployed rapidly and consistently. The platform approach encourages to a longer-term view of product strategy. Platform strategy can leverage significant operational efficiencies. Platform principles help the management to anticipate the replacement of a major product platform [1.10]. To be capable of a successful platform strategy, the underlying elements of the platform shall be clearly understood. Especially the defining technology shall be distinguished from the other platform elements. A platforms unique differentiation provides a sustainable competitive advantage in the market.

5.4 Extended life cycle model

The life cycle models presented in the literature are not fully compatible with long-life control electronics [1.3] [1.4]. The most important disadvantage of the presented life cycle models is the lack of platform generation changes. It is assumed that the platform components will somehow remain available long enough, or the focus has been on individual systems, not on the production and maintenance of numerous system instances. According to the literature, the typical service life of a long-life system varies between 30 to 40 years [1.1] [5.4]. This results in at least a 35 to 45 year life cycle for the technology used in the systems, because all technologies need to be available during the design time of the systems. Extreme values for the life cycle of a single technology generation can be even longer. For example, some tower crane models were designed in middle of the 1960's and the last cranes are expected to retire beyond 2010. The resulting life cycle of slightly over 50 years may also be realized for control system technology.

Most of components have become obsolete within 50 years – only a few discrete transistors and operational amplifiers are still available in addition to the passive components. Microprocessors and controllers form the core of current control systems, and for them such a long life cycle is too far away from reality. To reach such a long life cycle, the platform needs to be redesigned with new components many times by just keeping the interfaces and behavior backwards compatible. These generation changes do not provide any new added value – they just have to be performed to keep the
production running. The extended life cycle model with generation changes is presented in Figure 5.11.

Figure 5.11: Extended life cycle model

Figure 5.12 depicts an excerpt from the life cycle of a typical control system component. Applications are running on the physical architecture based on a microcontroller equipped with a set of peripherals defined by the application requirements. Because the physical architecture is fixed, problems are typically met when the current microcontroller becomes obsolete. With the component storing method presented in Section 5.2.3, the production of an old platform can be continued, but for a limited time. The most significant limitation is that the product owner has to know the production quantity until the product is retired. After determining the end-of-life production quantity wrongly, it cannot then be corrected because there are no critical components, e.g. microcontrollers, available in the market. Sometimes semiconductor vendors may modify their end of life forecasts and stop the production of a critical component suddenly and much earlier than expected. Based on experience, it is not easy to find a new microcontroller having the same or compatible instruction set and the same set of compatible peripherals located at the same memory addresses. If any of these characteristics change, each active application written for the platform has to be updated according to the new platform, and tested. SW industry has developed a lot of methods to help the translation of applications, which do not fall within the scope of this dissertation. Even if the translation work could be minimized, each active application still has to be tested with the new platform due to changes in the platform. The testing effort should not be neglected, because there may be dozens of active applications written for a single processing platform.
Moving from an old to a new application is also critical. Typically, a new application needs more processing power and more memory capacity and quite often expects a new platform. Sometimes the application development has to be started with the old platform to get enough time for the application SW development. If the development of the new application is delayed and the production of the old platform has to be extended, serious problems can be met, as described earlier. If a new platform is not fully compatible with the old applications, as too often it is, two different HW platforms have to be produced in parallel, and the production of both platforms has to be supervised.

5.5 Affect of platform reconfigurability

5.5.1 Support for multiple applications

The use of an FPGA as a flexible core of various digital systems is a mature and well-proven approach [1.11] [5.3] [5.2]. The behavior of an FPGA-based system can even be changed dynamically [5.13]. In a reconfigurable HW the logical architecture is typically constructed of IP-blocks [1.11], like processor core and peripherals. To maximize the benefits of IP-blocks, they shall be implemented to be re-usable [5.14] [5.15]. The details of design for re-use are not within the scope of this dissertation: readers are advised instead to read, e.g. references [5.14] and [5.16] for more details.

Chapters 2 and 3 presented networking components supporting both standard physical layers and state-of-the-art fault treatment units to promote the possibilities provided by standard networking. To enable flexible
interfacing with analog real world occurrences, AD-interfaces based on reconfigurable digital and mixed-signal devices were introduced in Chapter 4. The resulting in-system reconfigurable mixed-signal platforms consisting of an FPGA and an FPAA provide separate physical and logical architectures. Such an approach enables multiple system integrators to share the same physical platform without compromises in the features and to utilize it in numerous control and communication applications.

5.5.2 Support for multiple generations

The idea of separated component and system life cycles is expressed perfectly in the literature [1.4], but without any practical approach. Updating new logical architectures into existing platform generations has also been presented [5.3], [5.2]. The same flexibility can be used also in the opposite direction – the old logical architectures can be implemented into the new platform generation [5.4]. The most significant benefit of this is illustrated in Figure 5.13. It is possible to keep only one physical platform in production and still keep the old logical platform for old applications constant. Keeping the old logical architecture constant will free a lot of maintenance resources for new developments because only the logical architecture in the new physical platform needs to be verified instead of dozens of SW applications. Another significant benefit is the reduced spare part logistics effort. There is a single physical platform which can be configured as either an old or a new logical platform. Furthermore, the production of only a single physical platform reduces the production management and component logistics costs.

![Diagram](image)

Figure 5.13: Reduced affect of the generation change enabled by a constant logical architecture
5.5.3 Resulting enhanced life cycle mode

When only a few general purpose and in-system reconfigurable physical components form the core of the physical platform, the logical platform can be composed of soft-IP blocks implemented with standard HDLs. During the time they behave like SW – once they have been implemented and tested, they do not wear out. In the case of the physical HW changes below, logical platforms required by applications have to be adapted and verified instead of all applications written for those logical platforms. With reconfigurable platforms and soft-IPs, the commitment to the underlying technology is weak, because the soft-IP defines only the component behavior independently of the underlying HW [5.15]. The selected capacity levels of the selected platform device families are the major limiting factors. The selected HDL is currently not an issue for digital designs. The mixed-signal extensions are not well supported yet. Most digital design tools support common HDLs and mixed use of them in several HDL modules of the same design. When the HW platform is functionally generic and there are no logical platform limitations, the enhanced life cycle model presented in Figure 5.14 applies to the in-system reconfigurable mixed-signal platforms.

![Enhanced life cycle model enabled by reconfigurable HW](image)

Figure 5.14: Enhanced life cycle model enabled by reconfigurable HW

5.5.4 Resulting enhanced volume model

Production volume is the key issue in electronics production. The higher the volume, the lower the unit production costs are. The major savings come from component logistics – with the same effort any number of components can be purchased, and the higher the quantities that are purchased, the cheaper the components are. In production the assembly and soldering
processes are inexpensive, but setting up the production line for a product is time-consuming and expensive. In-system reconfigurable products can be adapted to the applications just by loading different configuration bit streams to them instead of complex assembly options, which avoids the need for different production line setups. Quality is improved, too, because the reduced number of assembly options furthermore reduces the production of products with wrong options.

Modern control systems consist of both communication devices and mixed-signal processing platforms. Chapters 2 and 3 covered a thorough analysis of the requirements of networking devices for both high-performance and low-cost networking technology. Communication devices can share the same physical platforms with the processing platforms, which, furthermore, leads to higher production volume. The utilization of reconfigurability has been expanded in Chapter 4 to cover also the AD-interfaces. It is essential, because the characteristics of analog inputs vary a lot among various control applications. According to Section 5.5.2, a common production volume can be expanded to cover also multiple generations of platforms. Figure 5.15 presents the enhanced volume model offered by in-system reconfigurable platforms.

Figure 5.15: Enhanced volume model enabled by reconfigurable HW

5.6 Summary

Production volumes of mobile control products have been scattered among different system integrators. More scattering of the volumes can be found
inside product lines and products of each system integrator. The long service life of the control products introduces an additional challenge by expecting simultaneous support of more than one control system generation at a time. The traditional life cycle model has been presented together with real world extensions and the resulting scattered volume model. Those models have been used as a base for the development of an improved approach to reducing the maintenance problems of long life control electronics.

All existing technical and administrative maintenance methods are needed, but reconfigurable general-purpose platforms provide strong improvements and introduce more freedom to select the best combination of existing maintenance methods for each phase of the system life cycle. Traditionally, spare part support has not been considered seriously enough, but reconfigurable platforms can reduce the effort required by spare part support. The use of reconfigurable platforms enables the division of platforms into physical and logical parts. A logical platform is a functional platform visible to the application SW, and the physical platform is only the semiconductor HW, to which the generic logical platform has been adapted. This approach enables the maintenance of any logical platform as long as is needed, instead of as long as the specific HW is available in the market. When the generic physical platform HW runs out of life, it can be changed into a new generic HW, to which the logical platform can be adapted. The adaptation of a single logical platform is less work than the adaptation and verification of dozens of applications written for the logical platform. It has been proven in Chapters 2 and 3 that networking devices can share the same physical platform components with application processing platforms. The outcome of Chapter 4 is that also mixed-signal platforms can be made in-system re-configurable with current mainstream components.

In the digital world standardized HDLs and design methodologies already support silicon-independence as long as product-specific special features are not used. Mixed-signal designs currently have to be manually translated from one architecture to another. Existing HDLs contain mixed-signal extensions but unfortunately the design tools support more device-specific design entry methods rather than standard HDLs. One should notice that the in-system reconfigurable HW offers only a possibility for efficient re-use and platform independence. Each design should be implemented carefully by keeping the design generic and re-usable.

When the whole way of working has been widely adopted, in-system reconfigurable mixed-signal platforms can be used in various control
products or product families among various system integrators. Even more significant is that separate spare parts production and storing is no longer needed, because a single configurable HW can be adapted to cover spare parts, too. The result is an enhanced volume model combining small sub-volumes together into a bigger volume, which will also be of more interest to the typical electronics manufacturer.
6 Discussion

Long life cycles, and low and scattered production volumes have been recognized to be the most significant problems with current mobile control electronics through the life cycle of the target systems. Distributing the control system can improve the re-usability of the system components, but current bus systems limit the maximum number of nodes in a single bus segment, topology and maximum length of a single bus segment. Inside the nodes, fixed hardware has led into scattering of the production volumes, emphasizing the already low production volumes of the control products. Analog inputs have been recognized to be the most diverse and challenging part to make flexible.

To solve the distribution problems, existing active hub solutions were reviewed. Based on this review, active hub and switch solutions were developed further for CAN and LIN networks, which are the main network technologies in the defined application area. Unlike in earlier published results, the focus in the current work is in standard physical layers to support current COTS component offering. Re-configurable digital processing platforms were examined and found to be a mature technology. Known re-configurable analog solutions have concentrated on signal scaling and filtering. This work concentrated on investigating how the flexibility could be spread to also cover ADCs. Two ADC architectures were tested in two alternative physical platforms to prove the required generality. The DC characteristics were measured to find out whether the concept can be considered a real alternative or not. Finally, the platform universality was expanded further to cover both processing platforms and communication devices and also different platform generations.

The main results of this work are divided into two sections. The scientific and technical outcome is presented first and followed by the commercial outcome. After summarizing the main outcome, the main limitations of the research work and the utilized technologies are explained and future work suggested. Finally the results are summarized and compared with the research sub-questions and the hypothesis defined in the introduction.
6.1 Scientific and technical outcome

6.1.1 CAN networking

Limited bandwidth and a strictly linear topology were identified as the most significant challenges in CAN networking, when systems have become more distributed. An active CAN-hub divides single logical bus segments into several physical buses. Unlike existing hub implementations, the hub presented in this work enables implementation of a star topology network without deviating from the standardized linear bus topology and high-speed termination scheme. Furthermore, the hub enables the connecting of more nodes in a single network than in the standard linear bus. The hub operates at the physical signaling layer – it synchronizes the bus states between the physical buses and keeps the synchronization between the nodes. Fault-treatment units presented in existing publications can be directly re-used in the active hub to increase the fault-tolerance of the logical bus. Regardless of the benefits of the hub, it increases propagation delay of the bus because of its operating principle. Due to the increased propagation delay, bit rates of up to 500kbps can be supported with feasible physical bus lengths. If galvanic isolation is needed, either the bit rate or the maximum physical bus length will be reduced more.

To reach the maximum 1Mbps bit rate and still maintain the physical extension, topology freedom and error isolation, a CAN-switch was successfully introduced. The restrictions of the hub can be avoided by using a store-and-forward type switch. On the one hand the operating principle intrinsically isolates all bus faults into originating physical bus and enables flexible and configurable frame filtering and forwarding. On the other hand, the synchronization between the nodes is lost due to the operating principle of the switch. The internal structure of the CAN switch has been adopted from ATM switches and it operates at 1Mbps bit rate and 100% bus load without lost frames. Switch ports can be galvanically isolated without the need to decrease the bit rate. Moreover, the switch multiplies the maximum bus length of the single physical network by the number of ports – at 1Mbps the CAN network without a switch can be 25m long and the switch allows 100m net length for the whole logical bus.

Hub and switch designs showed that the characteristics of high-speed CAN networks can be significantly improved without violating the physical layer standards. The maximum number of nodes in a single bus has not been a big problem in CAN-networks, but both switch and hub enable the connecting
of the maximum number of nodes in each physical network. In some applications protection components reduce the maximum node count for a single bus and such a reduction can be compensated by using a hub or a switch. The switch and the hub are both needed and selection between them shall be made separately for each system based on the requirements of the system under design. A CAN switch enabled the implementation of the tree topology network with an overall length over 150m, operating at 1Mbps and connecting together up to 821 CAN nodes. Illuminated outdoor lights at Vancouver airport – shown in Figure 6.1 – and a coal harbour are controlled through such networks. Based on the public discussions of CAN experts, these networks are the largest single CAN networks in the world [6.1].

Figure 6.1: Illuminated outdoor lights are controlled through the largest single CAN network in the world (Photo by courtesy of Automation Artisans Inc.)

6.1.2 LIN networking

LIN has been designed as a low-cost integration technology for subsystems with a single master and up to 15 slave nodes located relatively close to each other. LIN supports various topologies and its efficient master-slave protocol compensates the low bit rate well by enabling bus utilization of close to 100%. The active hub concept presented for CAN-networks can also be used in LIN networks with roughly the same benefits and constraints. Due to the low performance together with slow slopes, however, the active hub decreases the performance of LIN networks too much. The slopes cannot be accelerated because the LIN specification defines the slow slopes to minimize the electromagnetic emissions.
A reduced hub is not a generic solution, because increasing the performance requires removing the capability to resolve collisions occurring during wake-up sequences and event triggered frames. Neither switch can be used in LIN networks because of the tight timing constraints of the LIN frame transfers.

Due to the master-slave operation principle, typical frame transfers occur between the master and the slaves. It enables the use of a simple connection scheme between several parallel physical interfaces. Because the use of the concept has been limited to LIN master nodes, it has been named multiport master interface. It enables use of parallel physical LIN-buses, which are seen as a single logical network from the master node point of view. The multiport master interface supports all standardized bit rates and assigns the limitations of the standard LIN physical layer for each physical network – either maximum network length or maximum number of nodes can be achieved. The only limitation cause by the multiport master interface is that the communication between two slave nodes located in different physical networks cannot be supported.

### 6.1.3 Reconfigurable ADCs

An AD-interface is the most variable part of a mobile processing platform. At least the number of required analog inputs varies between all applications. In many applications a single ADC can be used for sampling many inputs, but in some applications more than one channel needs to be simultaneously sampled. Also resolution and sample rate vary strongly among the applications. In addition to the AD-conversion, different signal scaling and filtering is needed by various applications. Due to the strongly diverse properties in the analog inputs, an in-system re-configurable approach was proposed. The combination of re-configurable digital (FPGA) and mixed-signal (FPAA) HW can cover not only signal scaling and filtering but also AD-conversion and further digital processing. 8-bit resolution was selected for the tests firstly because of FPAA limitations, and secondly because such a resolution is widely used in entry-level hydraulic control systems.

The evaluation covered both SA- and ΔΣ-modulation ADCs implemented into two alternative FPAAAs with preceding signal scaling and filtering. Two ADC-architectures were tested to prove the flexibility of the concept – especially resolution and sample rate strongly relates with the selected ADC-architecture. Signal scaling and filtering were included to obtain the best achievable figure for overall accuracy of the tested FPAAAs. All
implementations achieved appropriate behavior and the required static accuracy was met by 3 of 4 implementations. The results showed that the concept of in-system re-configurable ADCs works, but identical implementations do not necessarily work optimally with every physical platform. Especially the accuracy of the \( \Delta \Sigma \)-modulation ADC implemented in the ispPAC20 was amazingly good, regardless of the very basic structures of the modulator and the decimator. Monolithic ADCs can offer much better but fixed accuracy, which may be optimal for some applications but too high or too low for the rest of the applications.

6.1.4 Volume and Life Cycle

The dominating challenge with the current processing platforms has been how to maintain the platforms long enough and with reasonable cost from the system integrator point of view. Together with low production volumes, the main problem is the extremely long life cycles of control electronics, compared with the much shorter component life cycles of mainstream electronics. In-system re-configurable HW has been proposed to solve the challenge by dividing the physical HW into physical and logical platforms. The physical platforms are provided by HW vendors and have life cycles typical of the mainstream electronics industry. HDL-format descriptions of e.g. processor core, memories, peripherals, amplifiers, filters and comparators form the logical architectures. Logical architectures are HW-independent and can be maintained as long as required.

Figure 6.2 presents an example where three different logical architectures have been implemented for a hub, a switch and a mixed-signal processing platform. Despite the very different requirements, all presented logical architectures can share the same physical architecture, which increases design, production and maintenance efficiency. One should note that ADCs utilize tightly coupled resources from both FPAA and FPGA. If mixed-signal resources are not needed by hub and switch logical architectures, the FPAA can be handled as an assembly option which does not need to be assembled. Furthermore, the same physical platform can be used to cover multiple products within the same system integrator or among several system integrators. It is also possible to use the same physical platform as a base for products in active production of the current system generation and spare part production for earlier system generation(s) – unlike in the traditional approach, where the focus has been in updating new features into older HW. However, such updates are intrinsically supported as long as the capacity of an older physical platform is not exceeded.
Re-configurable physical platforms provide easier design, maintenance and higher production volumes than the previous approaches – which is closer to mainstream electronics. Tools specific to the physical platforms provide translation from standard HDL-format design files into configuration files configuring the actual circuit elements of the physical platform – like registers, combinatorial logic, memory, amplifiers, comparators, DACs – in the desired way. Such a versatility of the physical platform enables an extremely flexible use of a single physical platform. A need for only single physical platform reduces overall design and maintenance costs, because only a single update effort instead of three, in the example, will be required in case of component obsolescence.

Figure 6.2: An example of a hub (top), a switch (middle) and a processing platform (bottom) sharing the same physical platform
6.2 Commercial outcome

A CAN-switch product is the main commercial result from the CAN-networking improvement research. There are already a few systems in production using the CAN-switch. Intrinsic error isolation between the physical buses has been the main reason to use the switch in all products. In one system the switch is used to enable connection of large number of nodes in a single logical bus. Load balancing between the physical buses has successfully been evaluated in one system. The existing systems with the CAN-switch are in mining, marine and outdoor applications, as presented in Figure 6.3. There are also some systems both inside and outside the defined application field under investigation, where the switch is a potential solution. During development of the CAN-switch, it was noticed outside the scope of this work that distributing some basic sensing and drive services into intelligent sensors and actuators efficiently increases the re-use of basic service implementations. Re-use intrinsically increases production volumes and supports well the enhanced volume model presented in Chapter 5.

Figure 6.3: Rock drilling rig (left) and marine diesel engine (right) are example systems equipped with the CAN-switch (Photos by courtesy of Sandvik Mining and Construction Oy and Wärtsilä Finland Oy)

To enable the use of LIN in cost sensitive drive applications, a multiport master interface for LIN master nodes was developed. It enables the connecting of more than 15 slave nodes to a single master node. The multiport master interface implementation relies on the master-slave operating principle of LIN systems. It slightly increases the cost of the LIN master node, but reduces the number of needed master nodes. The potential of the multiport master interface has not been utilized yet in the market.
Smaller scale tests with evaluation boards proved that it is possible to implement equal functionality with totally different FPAAAs. Accuracy and capacity are sufficient for the lowest-end but not enough for all commercial implementations. Short term second sourcing is needed to cover temporary availability problems. Furthermore, fully compatible alternative components shall exist in case of obsolescence of the primary component. An FPAA is such a new component type that nothing can be said about long-term issues. However, from the capacity point of view current FPAAAs seem to be comparable with the first reconfigurable logic SPLD devices.

Figure 5.4 shows an example of where a digital I/O-board suffering from obsolete discrete logic and EMC problems has been redesigned. With significant EMC-protection and thermal characteristic improvements the unit cost of the new version did not exceed the unit cost of the original one. It is a good example of the power of configurable logic. The most important thing is that the new version is fully compatible with the original one and software modifications with extensive testing were avoided.

Commercial use of re-configurable ADCs is not feasible yet, but an intermediate approach has been evaluated in one project. A PWM output board, presented in the bottom right of Figure 5.6, has been re-designed. The original analog coil current controls and analog inputs have been replaced with a monolithic ADC and FPGA. The board is one of the three boards in an I/O-unit. The unit cost of the whole unit was reduced by 24% and the new unit shown in the right of Figure 5.5 is fully compatible with the old one. In addition to the lower unit cost, the unchanged host processor interface enables use of the existing some dozens of applications without the need for testing each of them with the new HW. The amount of saved SW maintenance effort is significant. Obsolescence of the ADC prompts the selection of a new ADC and most probably redesign of the ADC interface in the FPGA design.

In both of the previously mentioned cases, the logic designs were implemented in standard format – VHDL. If the selected CPLD or FPGA becomes obsolete, the design in standard VHDL format can be directly used with design tools of a new CPLD or FPGA. It means that the next upgrade is mostly a partial PCB re-design with synthesis and floor planning for the new logical platform. Moreover, the use of in-system re-configurable components makes it easier to design a PCB layout by allowing optimization of pin-outs, because there are only a few fixed-function pins in the CPLDs, FPGAs and FPAAAs.
6.3 Limitations

This section describes the major limitations inside the focus of the work. Some of them had to be done and some others were found from the components, technologies and existing intellectual property rights during the work.

The limited amount of time and resources available for this work led to the exclusion of the integration and evaluation of already published fault treatment units [1.15] with the CAN-hub described in Chapter 3. Because it was clear from a very early phase of this work that a commercial CAN-switch product would be developed, target cost had to be considered carefully. An FPGA based solution would have been optimal from the technology point of view, but according to CAN license pricing [2.43], the initial license cost for a brand new product was too high. That is why a microcontroller-based approach was selected for the first CAN switch products. Moreover, the commitment to a fixed architecture microcontroller did not allow the evaluation of the more sophisticated message filtering and forwarding methods needed for a support of CAN frames with extended identifiers.

Because of the basic concepts of LIN bus technology, there is not much more which could be investigated. All networking solutions enabling an expansion of the bus were investigated and the absolute limits found. The major limitation found from the LIN protocol is that a network switch can not be used in LIN-networks because of the protocol structure.

Only the conceptual evaluation of a re-configurable ADC concept was performed due to the finite amount of time and resources for the work. Especially the structure of a ΔΣ-modulation ADC had to be limited to a first order modulator and accumulate-and-dump decimator, which are the most basic implementations. Moreover, the evaluation was limited to cover only the DC-characteristics of the ADCs. Resolution and architectures of the evaluated ADCs were limited by the capabilities and capacity of the available FPAAAs.

The fact is that implementing systems based on the FPGA and FPAA will require slightly higher NRE costs than traditional microcontroller based systems. That fact has hindered the possibilities of evaluating the overall concept in full scale production. The most significant obstacle recognized is the current way of working – it seems to be all too difficult to invest money during design to improve maintainability of the product, because the overall
maintenance costs can only be predicted, not exactly calculated, beforehand.

### 6.4 Future research

During the current work, the following main future actions were recognized. Because the idea of the reconfigurable ADC is new, most of the proposed future actions are related to ADC implementations in the in-system re-configurable hardware.

The performance of the presented CAN-switch relies on the adequate internal speedup of the switch. If the number of ports is increased, another approach than the processor – i.e. switching networks [2.41] [2.34] – should be considered. The table-based route table is another factor enabling fast operation. However, it limits the use of the switch to systems operating with 11-bit message identifiers and other, e.g. CAM-based filters [2.38] [2.39] should be investigated in the future to provide full compatibility and equal performance also for CAN-systems using 29-bit message identifiers. The troubleshooting capabilities of the switch could be improved by integrating fault-treatment units used in hubs [1.15] into switch ports. All the listed improvements require implementation of a switch into an FPGA, where various memory and bus structures and fault treatment units can be efficiently implemented.

The internal structures of existing FPAAAs do not optimally serve generic configurable ADC implementations, because most existing FPAA applications cover various filtering and signal conditioning solutions. Therefore, it should be investigated how the internal structures of FPAAAs could be improved to provide a wider selection of different types, resolutions and number of ADCs. ΔΣ-modulation ADCs are based on currently available on-chip structures. Higher order modulator and more complex modulator structures will require more than the currently supported resources. One approach to increase flexibility and resolution of the slope- and SA ADCs is the use of an external reference voltage DAC. It can be a modulating type [4.30] [4.31], which enables the implementation of a digital modulator in an FPGA, and analog filtering can be included into the FPAA. Both PWM and ΔΣ-modulation can be used. Another approach could be the use two DACs as fine and coarse ones [6.2] to provide a more flexible reference generation for DAC-based ADCs.
One interesting topic would be to implement design entry tools supporting some standard mixed-signal HDL, like VHDL-AMS and Verilog-AMS [5.15], instead of the current vendor- and device-specific design entry tools. Alternative approaches would be export tools from existing formats into a standard format. After getting all design files into the same standard HDL format, overall development and simulation could be performed without extra manual conversion effort from a device specific format into standard HDL format [4.12].

It is a well-known fact that the power-consumption of re-configurable chips is a little higher than fixed-function chips, like microcontrollers. In a defined application field power consumption is a minor problem, which should be considered after getting a sufficiently flexible HW, according to the presented concept, working with a standardized HDL design flow. Reduction of power consumption is mainly the responsibility of chip vendors, but possibilities to affect on the power efficiency by design principles should be investigated. Known methodologies already exist for digital FPGA and ASIC design [5.14].

One of the most interesting, but also most complicated, open question is, what are the "correct" capacity proportions between digital and mixed-signal resources? If such a fixed proportion exists, it could enable real single-chip solutions because reasonable amounts of memory have already been integrated into many FPGAs. If the single answer cannot be found, a further question is, whether there is any sense in looking for a single-chip solution. Or is a dual- or multi-chip solution with separate work- and non-volatile memories still more convenient, especially in the long-term focus? Single-chip solutions seem not to be important in the future because of the system-on-a-module approach enabled by advanced packaging technologies [6.3].

After getting the technology working well enough, the first real world cases can be implemented according to the proposed approach. One big question was encountered during the work – what is the "critical volume", below which the proposed approach is better than more traditional platforms? Long maintenance design experience has taught us that to be able to answer that question, tens or even hundreds of cases will be needed in order to obtain an answer.
6.5 Summary

The main results of this work can be summarized by comparing them with the four research sub-questions listed in the introduction:

Sub-question 1:

The CAN networking part answers sub-question 1 by presenting solutions for building alternative topologies and enabling more nodes in a single bus segment without message loss, violating the standard physical layer and reducing the bit rate. The LIN networking part answers sub-question 1 by offering a solution to the main problem of LIN networks – enabling more than 15 slave devices in single logical network without violating the physical layer characteristics and reducing maximum allowed bit rate.

Sub-question 2:

According to the refinements presented in the scope and objectives, 8-bit accuracy was set as the target. It was reached with 3 out of 4 ADCs implemented into in-system re-configurable mixed-signal hardware. The performed tests proved that it is possible to implement various ADCs with configurable mixed-signal hardware without significant semiconductor dependencies. The results answer very well sub-question 2.

Sub-question 3:

Based on the evaluations made in Chapter 4, the same in-system configurable mixed-signal platform can be utilized as a core of both networking elements and processing platforms. With a combination of separate FPGA and FPAA, assembly options can be utilized for digital-only products and FPAA can be left uninstalled to reduce the unit cost. Thus, the use of in-system configurable mixed-signal HW is the main answer sub-question 3.

Sub-question 4:

To be compatible with the majority of the applications, the physical architecture and the interfaces of the mixed-signal HW platform should be extremely flexible. A combination of FPGA and FPAA offers such flexibility for mobile processing platforms without physical HW changes. Moreover, the combination of FPGA and FPAA offers a generic physical platform simultaneously supporting
different logical platform generations and further increasing the production quantities of the HW. Such an adaptivity answers precisely sub-question 4.

It can be concluded that the work done provides answers to all four sub-questions set in the scope of the dissertation and in that way proves the hypothesis presented in the objectives. In-system re-configurable mixed-signal HW can bring production volumes of mobile processing platforms closer to the production volumes of mainstream electronics. A long life cycle can be achieved by separating the single fixed platform into separate physical and logical platforms with standardized interface between those.

In addition to meeting the scientific and technical targets, some direct commercial results were reached, too. A CAN-switch product broke through the market because it supports solutions increasing the re-use of the basic sensing, drive and monitoring services of mainstream mobile control systems. The proposed in-system re-configurable platforms were found to be cost-efficient also in the maintenance design of existing fixed architecture HW platforms.

While this work has concentrated on mobile machinery applications, most probably the results also apply to other application areas – like aerospace, marine and offshore electronics – having the same characteristics. The market development of the CAN-switch has already given positive signals for such possibilities.
7 Conclusion

This dissertation has examined how low volume control electronics can be produced and maintained more efficiently. Existing approaches have been adopted from mainstream electronics, where high production volumes are expected. This dissertation shows that a totally different approach is needed – the net life cycle cost of the target systems should be considered instead of unit cost of the control electronics products. When working with low-volume and long-life electronics products, more effort should be invested to optimize the maintainability of the products to obtain savings from the maintenance costs.

Distributed systems offer significantly better design re-use than the old centralized systems. The use of intelligent sensors and actuators provides well standardized basic functions which may be directly used in most existing systems. Current network technologies limit the topology, net bus length and maximum number of nodes connected to a single physical bus. The CAN hub and switch presented in this dissertation solve most of these challenges and provide additional diagnostics and error isolation services for CAN systems. In LIN systems the maximum number of nodes is the main limitation, which can be overcome with multiport master interface. The LIN protocol, however, prevents the implementation of a standard hub or a switch. All the presented developments are fully compatible with the nodes currently available in the market.

AD-interfaces are the most varying part among mobile control applications. The traditional approach based on fixed ADC(s) causes a significant design overhead, because most target applications require a customized analog I/O configuration. Mixed-signal in-system re-configurable platforms offer a general pool of resources not only for ADC(s) but also for signal conditioning. An acceptable accuracy for entry-level hydraulic controls has been reached in the tests, where very basic ADC structures were implemented and evaluated.

Earlier research provides proven concepts for the integration and the maintenance of digital in-system re-configurable physical platforms. This dissertation spreads the use of in-system re-configurability to cover also mixed-signal physical platforms. Fully configurable mixed-signal platforms
are remarkably flexible. They can be utilized in many control and communication applications and by various system integrators. Moreover, reconfigurable physical platforms can support different logical platform generations to minimize the amount of product variants in active production.

The main results of this dissertation can be summarized as follows:

- The development of active networking devices solves the main topology, physical extension and maximum number of nodes in single bus limitations of the modern distributed control systems.

- Instead of monolithic ADCs, in-system re-configurable ADCs can be used to enable a better adaptation to accord with the various applications without physical hardware modifications.

- Communication and processing devices for various applications, system integrators and application generations can share the same configurable mixed-signal physical platform.
References


[2.19] Serrano J.J., Campelo C.C., Capella C.V., Ors R., Bonastre A., Communications system based on CAN and extend with optical fiber for the implementation of a mobile robot immune to the sabotage provoked by electromagnetic noise, Proceedings of the 10:th iCC, CAN in Automation GmbH, pp. 04-09 - 04-15


[2.27] FlexRay Communications System Preliminary Node-Local Bus Guardian Specification, Revision 2.0.9, FlexRay Consortium, December 2005

[2.28] FlexRay Communications System Preliminary Central Bus Guardian Specification, Revision 2.0.9, FlexRay Consortium, December 2005


[2.33] CANopen Application Layer and Communication Profile, CiA DSP-301 rev. 4.1.1, CAN in Automation GmbH, 2006


[4.4] MAX1463 Low Power Two-Channel Sensor Signal Processor, Maxim Integrated Products


Continuous-Time $\Delta\Sigma$ ADC With Increased Immunity to Interferers, IEEE Journal of Solid-State Circuits, Vol. 39, No. 12, December 2004 pp. 2170 - 2178


[4.61] ADS7826/7/9 datasheet, SLAS388, Texas Instruments Inc., 2003


