LaS³ Laboratory for Safe and Secure Systems

a discipline of software engineering

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

Industrial Electronic

Laboratory of Industrial Electronics

Migration of Safely Embedded Software to FPGA Based Architectural Concepts

M. Steindl, J. Mottok, H. Meier, F. Schiller, M. Früchtl

2009–06-03

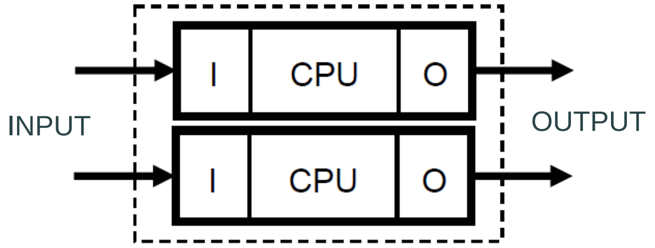
Content

Motivation

- **Main goal of safety-critical controllers**: recognition of erroneous data and wrong operations

- **General idea behind safety mechanisms**: ensure the right processing of data

- **Widely used approach:** use of two redundant hardware controllers,
  One channel is used to validate the results of the other channel
  - Increasing costs
  - Increasing space
  - Increasing energy consumption



- **New approach:** create diversity redundancy in software
  - Permanent and temporary errors can be detected
  - Independent of specific hardware
  - Independent of specific operating system

$\left.\right\}$ → portable

Motivation

- If a fault is detected an error reaction is initiated
  - Backward Recovery
  - Forward Recovery
  - Reset
  - Consecutive Calculation
  - Consecutive Transmission (Timing Redundancy)
  - Retry
  - Substitute Value
  - Degradation of Service (Limp Home)
  - Shutdown

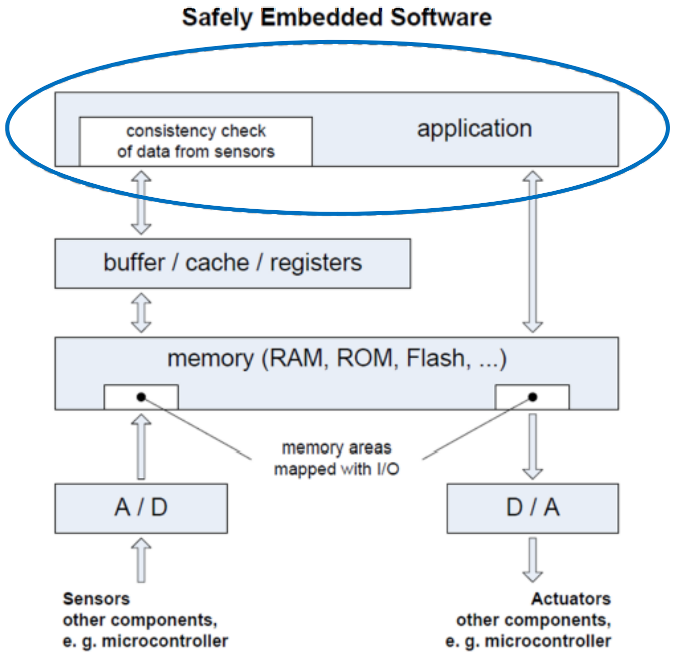
- **Problem:** arithmetic coding mechanisms are accompanied by larger data values and more complex operations in the transformation domain

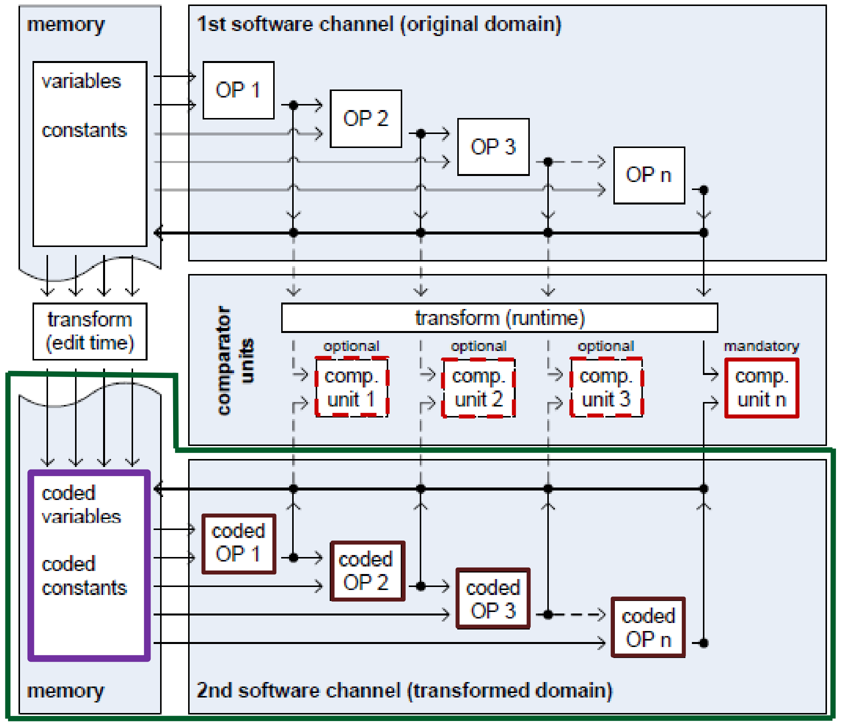
- Diverse Software channel is the main time-consuming factor in the system

➡ big influence on the performance of a system

Safely Embedded Software SES

- SES is **placed in the application layer**
- SES **adds a second channel of the transformed domain** to the software channel of the original domain.


Safely Embedded Software
consistency check of data from sensors
application
buffer / cache / registers
memory (RAM, ROM, Flash, ...)
memory areas mapped with I/O
A / D
D / A
Sensors other components, e. g. microcontroller
Actuators other components, e. g. microcontroller



- **second channel comprises:**
  - diverse data
  - diverse operations
  - comparator

Safely Embedded Software SES

Coding of Data

- Safely Embedded Software is based on the (AN+B)-code of the Coded Monoprocessor[1,2] transformation of original integer data x_f into diverse coded data x_c

- **Definition of Coded Data**: Coded data is data fulfilling the relation:

$$x_c = \boxed{A} * x_f + \boxed{B_x} + \boxed{D}$$

where

$$x_c, x_f \in \mathbb{Z},\ A \in \mathbb{N}^+,\ B_x, D \in \mathbb{N}_0,$$
$$\text{and} \quad B_x + D < A.$$

A: *prime number*, determines important safety characteristics like Hamming Distance and residual error probability of the code.

B: *static signature*, ensures the correct memory addresses of variables by using the memory address of the variable or any other variable specific number

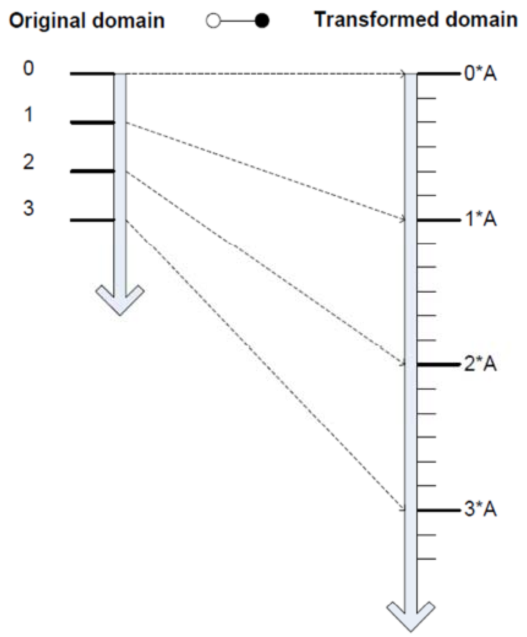
D: *dynamic signature*, ensures that the variable is used in the correct task cycle.

[1] Forin P.: Vital Coded Microprocessor Principles and Application for Various Transit Systems,1989

[2] Mottok J., Schiller F., Voelkl T., and Zeitler T.: A Concept for a Safe Realization of a State Machine in Embedded Automotive Applications,2007

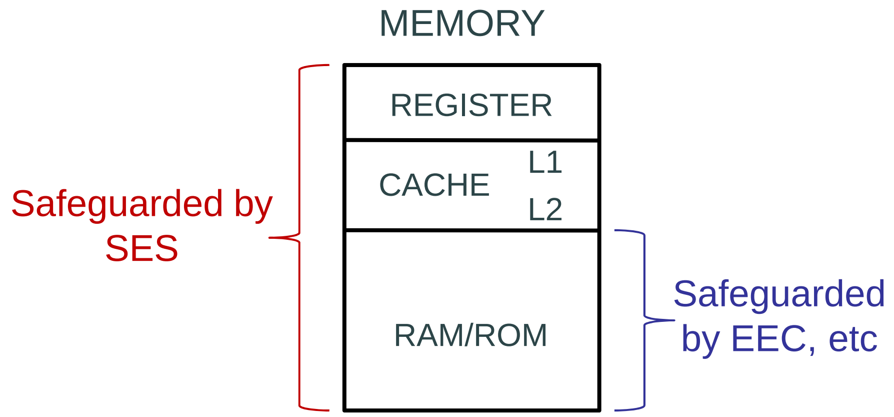
Safely Embedded Software SES

Coding of Data



Simple coding $x_c = A*x_f$ from original into transformation domain.

Safeguarded Areas



Verification

Comparison of both channels:

$z_c = A * z_f + B_z + D$?

Direct check of transformed channel:

$(z_c - B_z - D) \bmod A = 0$?

Coding of Arithmetic Operations

- **Coded Operator:** coded operator OP$_c$ is an operator in the transformed domain that corresponds to an operator OP in the original domain

$$
\begin{array}{rcl}
x_f & \circ\!\!-\!\!\!\bullet & x_c \\
y_f & \circ\!\!-\!\!\!\bullet & y_c \\
z_f & \circ\!\!-\!\!\!\bullet & z_c \\
z_f = x_f \, \mathrm{OP} \, y_f & \circ\!\!-\!\!\!\bullet & z_c = x_c \, \mathrm{OP}_c \, y_c
\end{array}
$$

$$
\begin{array}{rcl}
z_f = x_f + y_f & \circ\!\!-\!\!\!\bullet & z_c = x_c \oplus y_c \\
z_f = x_f - y_f & \circ\!\!-\!\!\!\bullet & z_c = x_c \ominus y_c \\
z_f = x_f * y_f & \circ\!\!-\!\!\!\bullet & z_c = x_c \otimes y_c \\
z_f = x_f / y_f & \circ\!\!-\!\!\!\bullet & z_c = x_c \oslash y_c
\end{array}
$$

Coding of Integer Addition

The addition is the simplest operation of the four basic arithmetic operations

$$z_f = x_f + y_f$$

$$\frac{z_c - B_z - D}{A} = \frac{x_c - B_x - D}{A} + \frac{y_c - B_y - D}{A}$$

$$z_c - B_z - D = x_c - B_x - D + y_c - B_y - D$$

$$z_c = x_c - B_x - D + y_c - B_y + B_z$$

$$z_c = x_c + y_c + \underbrace{(B_z - B_x - B_y)}_{const.} - D$$

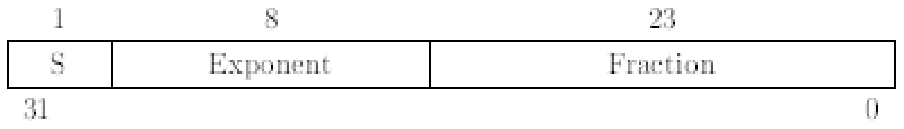
A comparison of the equations leads to the definition of the coded addition:

$$\boxed{\begin{aligned} z_c &= x_c \oplus y_c \\ &= x_c + y_c + (B_z - B_x - B_y) - D \end{aligned}}$$

Coding of Floating-Point Addition

- De facto standard for floating–point representation: **IEEE 754**
- **IEEE 754** specifies **formats** and **methods** for floating-point arithmetic

- Representation of a single 32-bit floating-point number:



- Floating-point components (sign, mantissa, exponent) are separately processed in the transformed domain

- Processing of Floating-point components is based on coded integer operations

SES Performance Analysis

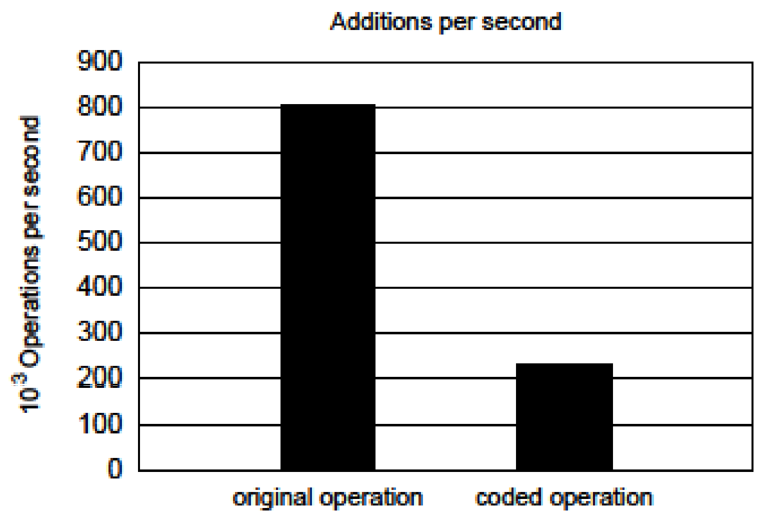
- **Performance Analysis**: Comparison between performance in the original and the transformed domain
  - Integer addition - coded integer addition
  - Floating-point addition - coded floating-point addition

- **Floating-point arithmetic:**
  - floating-point addition in original domain is completely emulated in software (no compiler library used)
  - Rounding mode: Round to zero

- **Test Setup**
  - Infineon XC167CI-32F40F (20 MHz) 16-Bit microcontroller
  - Keil µVision v3.53 C-Compiler (no optimizations)
  - no hardware support used (FPU)

Performance Analysis: Addition

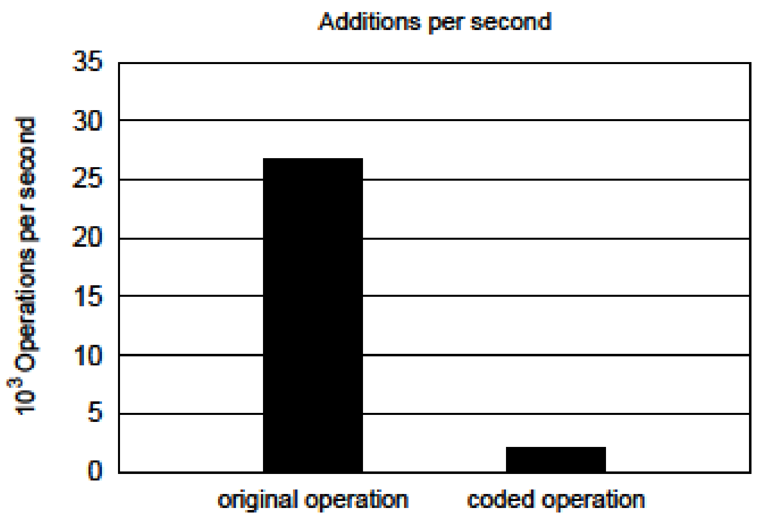
16-bit integer additions per second



original domain: 800x10³ op/s
transformed domain: 227x10³ op/s

➡ ~ 3.5 times slower

32-bit floating-Point additions per second



original domain: 26.7x10³ op/s
transformed domain: 2.1x10³ op/s

➡ ~ 13 times slower

Performance Analysis: Summary

- Significant difference in performance especially for floating-point arithmetic

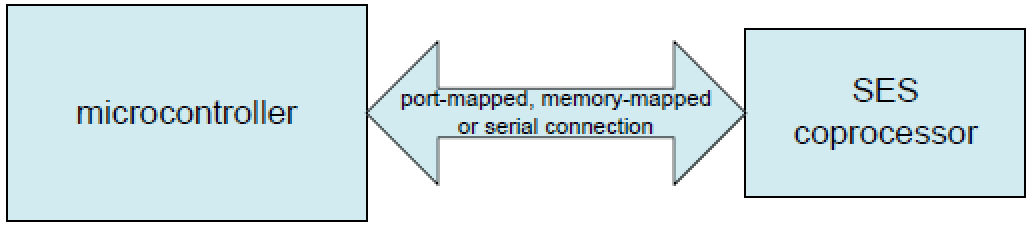
- Integer as well as floating-point addition are the **easiest operations** in the transformed domain
  - Expectation of **higher difference** for other arithmetic operations like **multiplication or division**, especially for coded floating-point operations

- If floating-point arithmetic is hardware supported (floating-point calculations in the original domain by a FPU), the SES approach is not practicable

➡ SES could lead to a growing demand
for performance if e.g. floating-point operations should
be safeguarded

Migration to SES Coprocessor

- **Approach**: Source the SES part out of the microcontroller



- SES Coprocessor contains a **coded arithmetic unit ($\mathbf{ALU_c}$)** for coded integer and/or coded floating point arithmetic

- **$\mathbf{ALU_c}$** is completely realized in hardware

- Platform: CPLD, FPGA, ASIC (for large number of units)

- Interconnection: CAN, memory-mapped, serial connection (safeguarded)

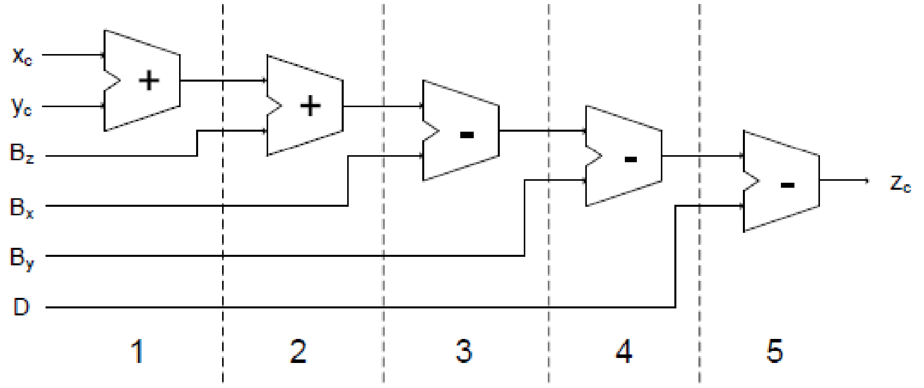
Migration to SES Coprocessor

- **Hardware implementation of coded addition**

  - Definition of coded addition:

$$z_c = x_c + y_c + B_z - B_x - B_y - D$$

  - Corresponding hardware model



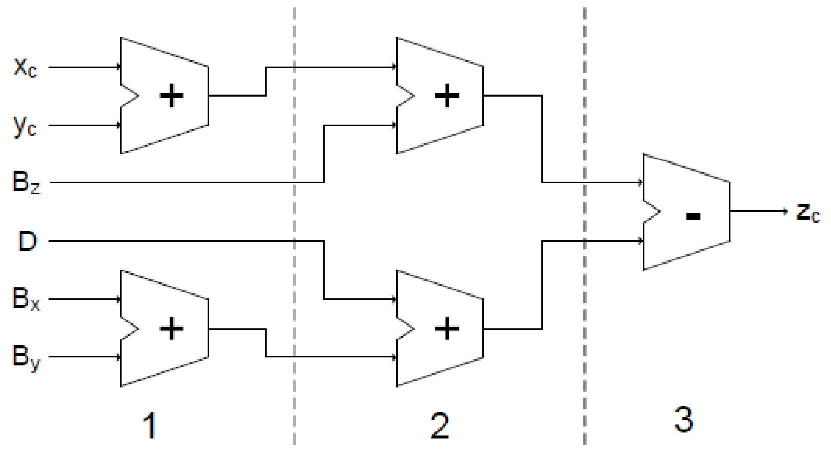
- 2 adder, 3 subtracter
- serial alignment
- concurrent design
- propagation delay of **one** adder/subtracter: typ. **6 ns**
- **total execution time: 30 ns**

Migration to SES Coprocessor

- **Design improvement**
  - Rearrangement of the equation for coded addition

$$z_c = x_c + y_c + B_z - B_x - B_y - D \Rightarrow z_c = (x_c + y_c + B_z) - (B_x + B_y + D)$$

  - Corresponding hardware model



- 4 adder, 1 subtracter
- serial and parallel alignment
- propagation delay of **one** adder/subtracter: **6 ns**
- **total execution time: 18 ns**

Migration to SES Coprocessor

- **Advantages**
  - Fulfills safety requirements with the SES approach
  - Increases performance
  - Possibly existing systems could be retained
  - Only low density FPGA is necessary

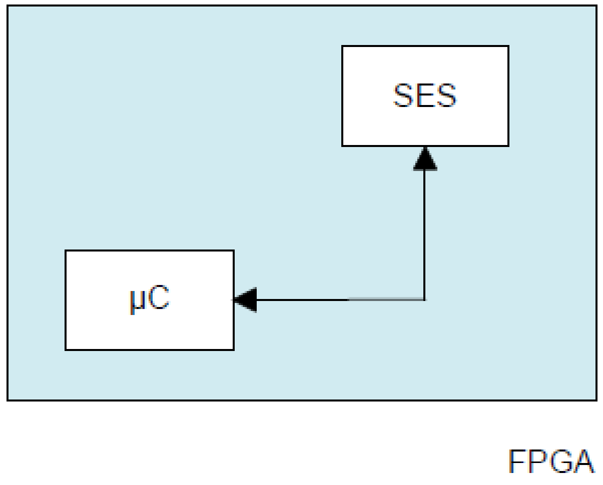
- **Disadvantages**
  - Additional Hardware needed
  - Additional costs
  - Additional space
  - Energy consumption

Migration to FPGA Architecture

- **Single core processor with SES**

  - State of the art FPGAs contain either hardwired microcontrollers (e.g. XilinxVirtex 4) or a microcontroller soft core could be implemented.

    Examples for soft cores: - 8-bit PicoBlaze (Xilinx)
    - 16-bit Nios II (Altera)
    - 32-bit Leon 3 (GNU GPL Licence)



FPGA including microcontroller soft core and SES-coprocessor

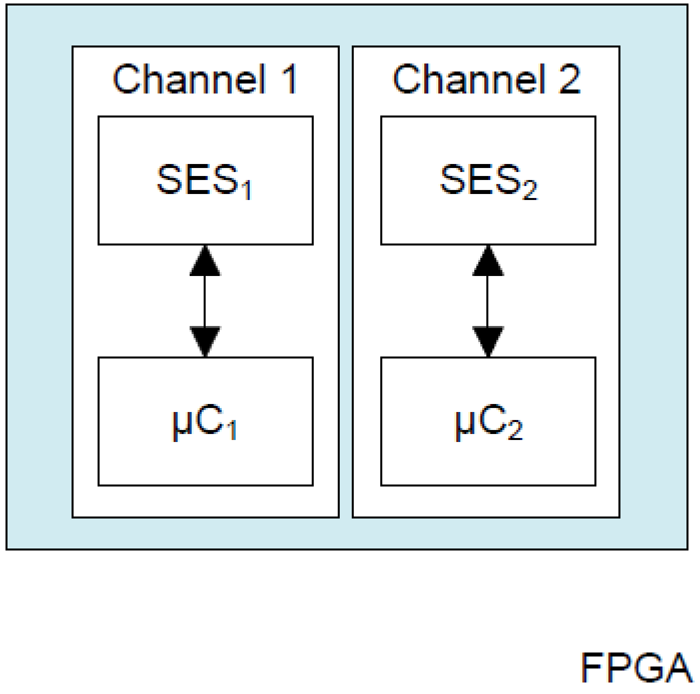
The SES-coprocessor is reconfigurable and contains the operations defined in the SES approach needed by the application

Migration to FPGA Architecture

- **Single core processor with SES**
  - **modular microcontroller concept** - several microcontroller could be used on the same hardware (FPGA), so the microcontroller could be changed quickly due to modified requirements without changing any physical components
  - **modular SES coprocessor concept** - the SES coprocessor could be easily adapted to the desired application, for instance only coded integer arithmetic could be installed or, if needed, coded floating-point arithmetic could be implemented too
  - concept verification with fault injection strategies (stimulated bit flips) in the soft core divider.
  - automatic regression tests are also possible as needed for a safety assessment of the SES FPGA

Migration to FPGA Architecture

- Dual core processor with SES
    - For higher SIL demands this approach could be easily expanded by adding a traditional redundant hardware channel



FPGA including dual microcontroller soft core and SES-coprocessor in a redundant hardware approach

Migration to FPGA Architecture

- Dual core processor with SES

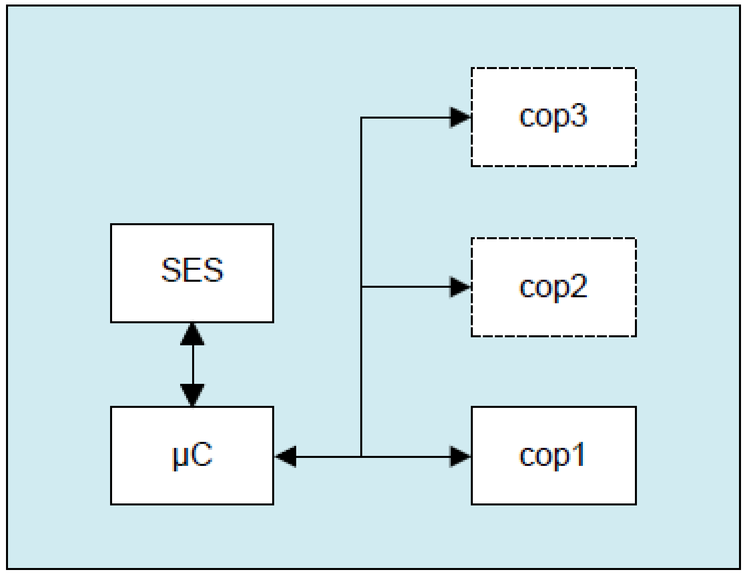
  - **existing know-how** in redundant hardware design could still be **used**
  - **microcontrollers have not necessarily be equal**, two different soft cores can be implemented to reduce common cause failures (diverse hardware channels)
  - **SES coprocessors could be different**, alternative safety concepts e.g. ECC or Hamming-Code could be implemented
  - **number and type of cores could be changed** without changing any physical component
  - voting technique for both channels is based on the normative regulations given in IEC61508[8] / ISO/CD 26262[

Migration to FPGA Architecture

- Further options to increase the performance

  - Modern FPGAs can hold more than one single microcontroller soft core and a SES coprocessor

  ➢ additional coprocessors for specific requirements could be added.



FPGA

FPGA including microcontroller soft core, SES-coprocessor and several coprocessors for individual tasks

Migration to FPGA Architecture

- **Further options to increase the performance**

  - Tasks for coprocessors can be:
    - Fast signal processing (FFT,...)
    - Fast arithmetic calculations (e.g. multiplications with high bit width)
    - Decoders
    - Multiplexers

  - Extended digital hardware components could be reduced by including them into a FPGA

  - Increases performance, fulfills safety requirements with the SES approach and reduces the hardware complexity

Hardware - Software Codesign

- Choosing a FPGA architecture including microcontroller and SES cores also affects the design methodology of the system

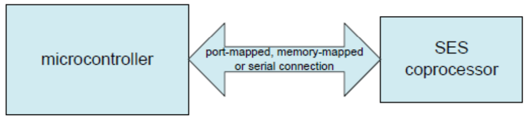
  - Allows the cooperative and concurrent development of hardware and software
  - Co-specification, co-development, and co-verification possible
  - The decision if a task should be realized in software or in hardware could be changed rapidly
  - Parallelization of hard and software design reduces costs of development and the time to market

- For further information about Hardware – Software Codesign see relevant literature, e.g. Mahr, T., Gessler, R.: Hardware-Software-Codesign Vieweg, Wiesbaden 2007

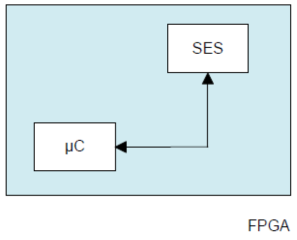
Laboratory for Safe and Secure Systems
a discipline of software engineering

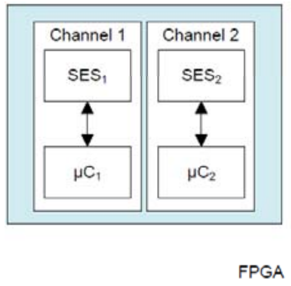
Conclusion and Outlook

- **The approach offers considerable advantages:**
  - **SES Coprocessor**
    - Possibly existing systems could be retained
    - Performance of SES guided systems could be improved
    - Only a low density FPGA is necessary



  - **Complete integration of microcontroller and SES-coprocessor**
    - Modular microcontroller concept
    - Modular SES coprocessor concept
    - Concept verification with fault injection strategies
    - number and type of cores could be changed
    - additional coprocessors for specific requirements could be added
    - Possibility for Hardware-Software-Codesign





Conclusion and Outlook

- **Next Steps:**
  - **Development of hardware library of coded operators**
    - Integer arithmetic
    - Floating-point arithmetic
  - **Verification of the existing software library**
    - Soft core on a FPGA using software library of coded operators
    - Verification by stimulated bit flips

Thank you!