Migration of SES to FPGA Based Architectural Concepts

M. Steindl[1], J. Mottok[1], F. Schiller[2], M. Fruechtl[2]

[1]Regensburg University of Applied Sciences
Department of Electronics and Information Technology
Seybothstr. 2, D-93049 Regensburg, Germany
{michael.steindl; juergen.mottok}@e-technik.fh-regensburg.de

[2]Technical University Munich
Institute of Information Technology in Mechanical Engineering
Boltzmannstr. 15, D-85748 Garching near Munich, Germany
{schiller; fruechtl}@itm.tum.de

Abstract

Creating diverse redundancy for fail safe or fail operational systems in software reduces costs for additional hardware and increases flexibility. A lack of performance occurs by implementing a diverse software channel for e.g. floating-point operations in a traditionally microcontroller architecture. This paper gives an approach for transferring the Safely Embedded Software (SES) into a coprocessor and to migrate SES to a flexible and powerful FPGA architecture.

1 Introduction

Embedded automotive systems use both fail safe and fail operational architectures that are mainly based on hardware redundancy. The Safely Embedded Software (SES) approach is an alternative to save the redundant hardware channel by creating software diversity [1, 4]. With SES, safety can be created independently of a specific hardware or operating system. Both permanent and temporary errors can be detected in order to initiate a suitable reaction (e.g. retry, recover, fail safe state or fail operational).

The paper is structured as follows: In Section 2, a general survey of controller technology for safety-critical systems is given. Section 3 gives an overview of SES. The performance of the approach is analyzed in Section 4. Section 5 describes the FPGA approach.

2 Systematics of safety-critical controller solutions

Advancements in controller technology are accompanied by increasing functionality and reliability, leading to an immense broadening of the application area. Consequently, the spectrum of tasks no longer includes only rare controlling tasks, but more and more responsible challenges like safety-critical scenarios are tackled. The basic requirement for a system to be used in a safety-critical application area is to fulfill certain safety properties. The main goal of safety-critical controllers is the recognition of erroneous data operations. So one of the measurements for the qualification of controllers for safety-critical systems is the so called residual error probability. This probability indicates to which degree errors can occur that are not detectable. Safety norms base their classification of systems on such measurements to determine if certain controller architectures are suitable for safety areas. As a single hardware controller is not meeting the requirements for the use in safety-critical systems, further precautions have to be taken. This Section gives a short overview of the systematics of controller technology for safety-critical application areas (Figure 1 shows a summary of the main approaches).

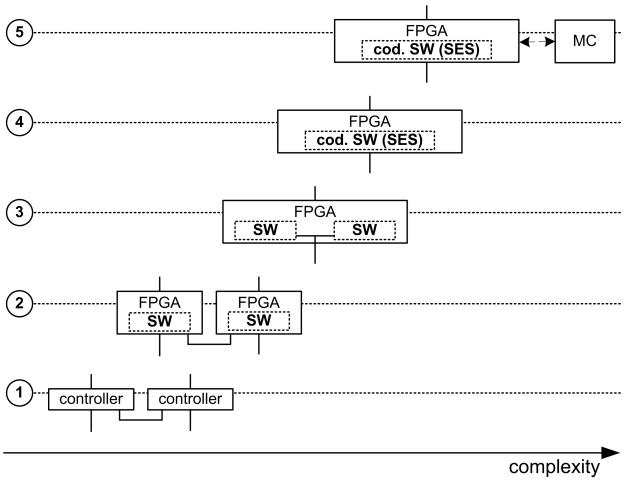


Figure 1: Systematics of controller technology for safety-critical systems.

The general idea behind safety mechanisms is to ensure the right processing of data. Therefore, a possibility to observe the validity of processed data has to be created. An obvious and widely used approach (for example in embedded automotive systems) to control processed data is the use of two redundant hardware controllers (as shown in Figure 1 Step 1). This solution consists of two channels, where the second, re-

dundant hardware channel is used to validate the results of the first channel, thus allowing the recognition of errors in data processing. In the case, that both channels provide different results, an error reaction is triggered. This error reaction often consists of an emergency stop of the system, as it is not possible to determine which of the two different results is the correct one. Due to this fact, the two channel solution is not applicable for all safety areas. The architecture can be extended to a three channel solution allowing the compensation of errors on the basis of a majority decision between the results of three channels. This type of architecture is mainly used in areas, where an emergency shutdown is not applicable (such as planes). Obviously, the two (respectively three) channel hardware approach has the disadvantage, that additional hardware controllers are used, meaning that depending on the particular environment, this could be an unacceptable cost and space factor.

Instead of redundant hardware controllers, there is the possibility of replacing those with two FPGAs running the logic part (software) of a controller task (as shown in Figure 1 Step 2). In comparison to hardware controllers, FPGAs allow a simple reconfiguration, thus making this architecture more flexible and cheaper. Despite of the simple reconfiguration, this approach shares the disadvantages of the first one.

Examining the use of two redundant hardware FPGAs more precisely, there are basically two software channels running on separate hardware channels, so that the next logical step is the embedding of those two software channels in a single hardware FPGA turning the approach into a single hardware channel system (Figure 1 Step 3).

Instead of using two separate, identical software channels, coding mechanisms can be used to basically create the second diverse software channel within the original software channel (Figure 1 Step 4). One possible approach for such a coded channel is SES, achieving the required safety properties by using coding mechanisms. Starting from the original domain, diverse data is created by transferring the data and operations to a new domain. Based on the results of operations in the diverse domain, the original domain can be validated assuring the correct execution of the controller task. More information about the particular coding mechanisms of SES can be found in Section 3.

As arithmetic coding mechanisms are accompanied by larger data values and more complex operations in the transformation domain, those mechanisms can have a big influence on the performance of a system (The effect of SES on the performance of a system is covered in Section 4). For a system using the one hardware channel approach with a single software channel (SES), the main time-consuming factor is the SES part. This paper introduces a solution to this matter by sourcing out the SES part to dedicated hardware, a so called SES-coprocessor (as shown in Section 5). As a possible architecture for this solution, FPGAs are chosen. FPGAs have the advantage, that on the one hand, the FPGA can be used to realize a logic function. But on the other hand, the FPGA can also be used to implement a von-Neumann architecture, meaning that a complete microcontroller can be realized within a FPGA (a so called microcontroller soft cores). This paper introduces a complete migration of the SES architecture to a FPGA, where microcontroller and SES-coprocessor are realized within a single FPGA.

3 Safely Embedded Software (SES)

This Section gives an overview of the Safely Embedded Software (SES) approach.

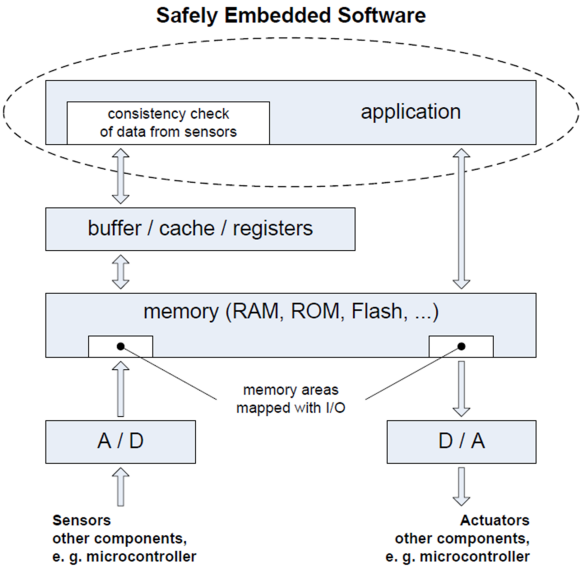


Figure 2: The Safely Embedded Software approach.

The method of Safety Code Weaving is a basic principle of SES in the application layer (Figure 3) [7]. Safety Code Weaving is the procedure of adding a second software channel to an existing software channel. In this way, SES adds a second channel of the transformed domain to the software channel of the original domain. In dedicated nodes of the control flow graph, comparator functionality is added. Though, the second channel comprises diverse data, diverse instructions, comparator and monitoring functionality. The comparator or voter, respectively, on the same ECU has to be safeguarded with voter diversity [2] or other additional diverse checks. Due to the safety code weaving, it is possible to check the validity of data in a given granularity. Based on this results, a dedicated fault reaction is initiated to achieve a fail safe state.

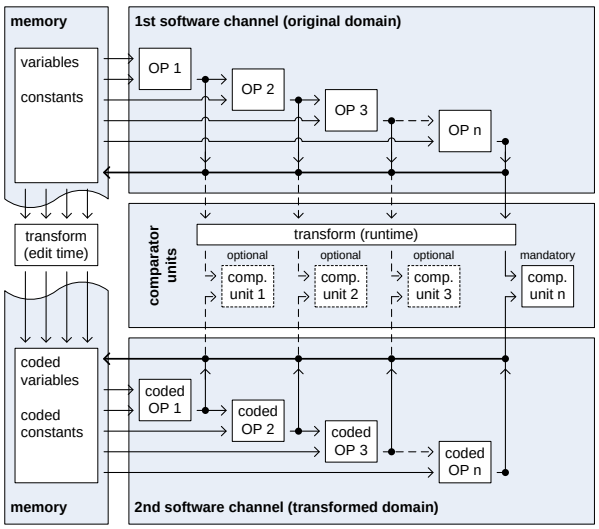


Figure 3: Safety Code Weaving.

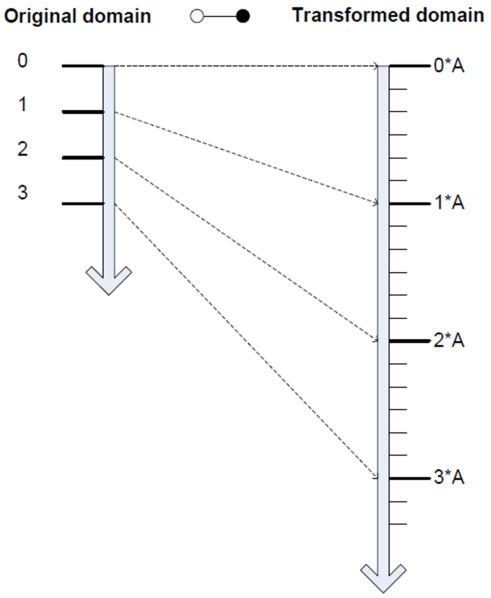


Figure 4: Simple coding $x_c = A{*}x_f$ from the original into the transformation domain.

3.1 Coding of Data

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

Definition 1 (Coded Data) *Coded data is data fulfilling the relation:*

$$x_c = A * x_f + B_x + D$$
$$where\ x_c, x_f \in \mathbb{Z},\ A \in \mathbb{N}^+,\ B_x, D \in \mathbb{N}_0,$$
$$B_x + D < A.$$

The prime number A [1, 10] determines important safety characteristics like Hamming Distance and residual error probability $P = 1/A$ of the code. Number A has to be prime because in case of a sequence of i faulty operations with constant offset f, the final offset will be $i * f$. This offset is a multiple of a prime number A if and only if i or f is divisible by A. If A is not a prime number then several factors of i and f may cause multiples of A. The same holds for the multiplication of two faulty operands. Additionally, so called deterministic criteria like the above mentioned Hamming Distance and the Arithmetic Distance verify the choice of a prime number.

Other functional characteristics like necessary bit field size etc. the handling of overflow are also caused by the value of A. The simple transformation $x_c = A * x_f$ is illustrated in Fig. 4.

The static signature B_x ensures the correct memory addresses of variables by using the memory address of the variable or any other variable specific number. The dynamic signature D ensures that the variable is used in the correct task cycle. The determination of the dynamic signature depends on the used scheduling scheme. It can be calculated by a clocked counter or it is offered directly by the task scheduler.

The instructions are coded in that way, that at the end of each cycle (i. e. before the output starts) either a comparator verifies the diverse channel results $z_c = A * z_f + B_z + D?$, or the coded channel is checked directly by the verification condition $(z_c - B_z - D) \bmod A = 0?$ (cf. Definition 1).

3.2 Coding of Operations

A complete set of arithmetic and logical operators in the transformed domain can be derived. The transformation in Definition 1 is used.

Definition 2 (Coded Operators) *A coded operator* $\mathrm{OP_c}$ *is an operator in the transformed domain that corresponds to an operator* OP *in the original domain. Its application to uncoded values provides coded values as results that are equal to those received by transforming the result from the original domain after the application* OP *for the original values. The formalism is defined, such that the following statement is correct for all* x_f, y_f *from the original domain and all* x_c, y_c *from the transformed domain, where* $x_c = \sigma(x_f)$ *and* $y_c = \sigma(y_f)$ *is valid:*

$$\begin{aligned} x_f &\multimap\!\circ\ x_c \\ y_f &\multimap\!\circ\ y_c \\ z_f &\multimap\!\circ\ z_c \\ z_f = x_f\,\mathrm{OP}\,y_f &\multimap\!\circ\ z_c = x_c\,\mathrm{OP_c}\,y_c \end{aligned} \qquad (1)$$

Accordingly, the unary operators are noted as:

$$z_f = \mathrm{OP}\,y_f \multimap\!\circ\ z_c = \mathrm{OP_c}\,y_c \qquad (2)$$

In the following, the derivation steps for the addition and subtraction operation are explained exemplarily.

3.2.1 Coding of Addition

The addition is the simplest operation of the four basic arithmetic operations. Defining a coded operator (see Definition 2), the coded operation $\oplus$ is formalized as follows:

$$z_f = x_f + y_f \qquad \circ\!\!-\!\!\bullet \qquad z_c = x_c \oplus y_c \tag{3}$$

Starting with the addition in the original domain and applying the formula for the inverse transformation, the following equation can be obtained for z_c using 1:

$$\begin{aligned} z_f &= x_f + y_f \\ \frac{z_c - B_z - D}{A} &= \frac{x_c - B_x - D}{A} + \frac{y_c - B_y - D}{A} \\ z_c - B_z - D &= x_c - B_x - D + y_c - B_y - D \\ z_c &= x_c - B_x - D + y_c - B_y + B_z \\ z_c &= x_c + y_c + \underbrace{(B_z - B_x - B_y)}_{const.} - D \end{aligned} \tag{4}$$

The equations (3) and (4) state two different representations of z_c. A comparison leads immediately to the definition of the coded addition $\oplus$:

Definition 3 (Coded Addition $\oplus$)

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

3.2.2 Coding of Subtraction

In analogy to the addition in Section 3.2.1, the coded operation $\ominus$ can be fomarlized as follows:

$$z_f = x_f - y_f \qquad \circ\!\!-\!\!\bullet \qquad z_c = x_c \ominus y_c \tag{5}$$

Applying the formula for the inverse transfornmation, the following equation can be obtained for z_c:

$$\begin{aligned} z_f &= x_f - y_f \\ \frac{z_c - B_z - D}{A} &= \frac{x_c - B_x - D}{A} - \frac{y_c - B_y - D}{A} \\ z_c - B_z - D &= x_c - B_x - D - y_c + B_y + D \\ z_c &= x_c - y_c + \underbrace{(B_z + B_y - B_x)}_{const.} + D \end{aligned} \tag{6}$$

According to this equation, the coded subtraction $\ominus$ can be defined as follows:

Definition 4 (Coded Subtraction $\ominus$)

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

On basis of this procedure, other arithmetic operations like multiplication and division can be formalized.

4 Performance Analysis

In this Section, the performance of the coded addition is discussed. In addition to the coded integer addition, the coded floating-point addition is introduced. This floating-point addition in the transformed domain is based on coded integer addition with separating mantissa, exponent and sign. For a better comparison between the performance of floating-point additions in the original and the transformed domain, the calculations in the original domain are completely emulated in software and no internal compiler library is used. The calculations in the transformed domain are completely handled in software following the SES approach.

All measurements are done on an Infineon XC167CI-32F40F BB-A(XC167) microcontroller. This 16-bit controller is broadly used in the automotive industry. For clock frequency, 20 MHz were selected and the Keil Vision3 v3.53 C-Compiler without optimizations is used.

As shown in Figure 5(a), the coded integer addition on the XC167 microcontroller is only about 3.5 times slower than the addition in the original domain. However, the corresponding floating-point addition (Figure 5(b)) is about 10 times slower. As mentioned above, the floating-point addition in the original domain is emulated in software, so that the difference to a calculation with a hardware floating-point unit (FPU) would be much higher. The mentioned integer addition as well as the floating-point addition are the easiest operations in the transformed domain. It is expected that for other arithmetic operations like multiplication or division the difference in performance between original and transformed domain is rising, especially for coded floating-point operations.

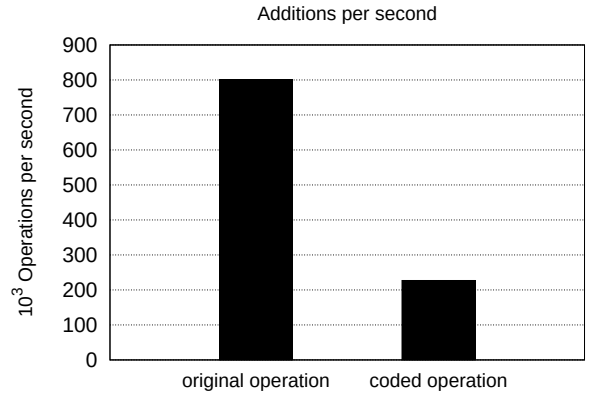
Due to this, SES could lead to a growing demand for performance if e.g floating-point operations should be safeguarded.

5 SES-Coprocessor

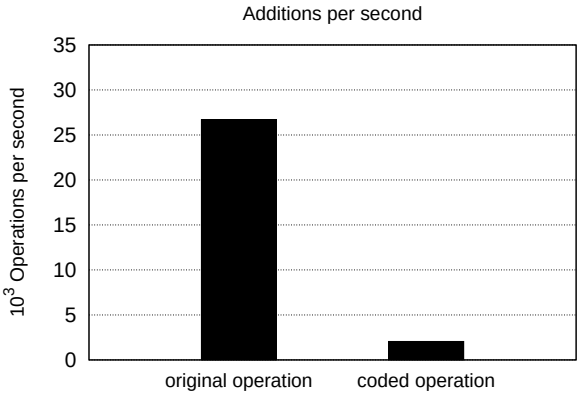
A possible option to raise the performance is to source out the SES part of the microcontroller by applying a SES-coprocessor.

5.1 External coprocessor approach

The SES-coprocessor connected to the microcontroller (Figure 6) contains a coded arithmetic logic unit (ALU) for integer and (if needed) floating-point arithmetic. The communication path between microcontroller and SES-coprocessor has to be safeguarded. This ALU could be completely realized in hardware, so a complex programmable logic device (CPLD) or a field programmable gate array (FPGA) would be a proper platform for the SES-coprocessor. If a large number of units is expected, an application specific integrated circuit (ASIC) is a possible alternative.



(a) integer additions per second



(b) floating-point additions per second

Figure 5: Performance of integer and floating-point additions in the original and coded domain on Infineon XC167CI-32F40F BB-A microcontroller

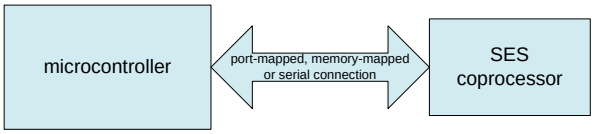


Figure 6: Microcontroller with a SES-Coprocessor. The interconnection between microcontroller and coprocessor could be realized by port- or memory mapping, also serial connections (CAN, I^2C, etc.) are possible.

5.2 Hardware implementation of coded addition

This Section describes the realization of the coded integer addition in hardware.
Based on the definition of the coded integer addition (Section 3.2.1)

$$z_c = x_c + y_c + \underbrace{(B_z - B_x - B_y)}_{const.} - D \qquad (7)$$

a simple hardware model could be created.

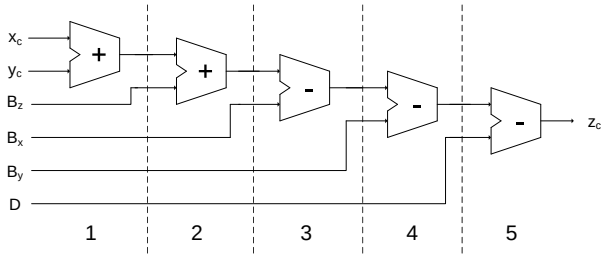


Figure 7: Structure of a coded ALU for coded integer-addition

The coded ALU for integer-addition shown in Figure 7 consists of 2 adders and 3 subtracters in a serial alignment.

Overflow handling is not explicitly mentioned in this paper. Due to the concurrent logic approach of this model, the execution time of a coded addition only depends on the propagation delay of the gates, as there is no dependency on the clock frequency (of course, the propagation delay affects the clock frequency on the whole design). If we assume that the propagation delay of an adder/subtracter would be about $6ns$, the calculation of a coded addition could be finished in about $30ns$.

In a next step, this design could be optimized. A main advantage of a hardware implementation in comparison to software is the possibility to parallelize operations. The equation 7 could be rearranged as follows:

$$z_c = (x_c + y_c + B_z) - (B_x + B_y + D) \qquad (8)$$

Based on the equation 8 we could modify the hardware model in the following way.

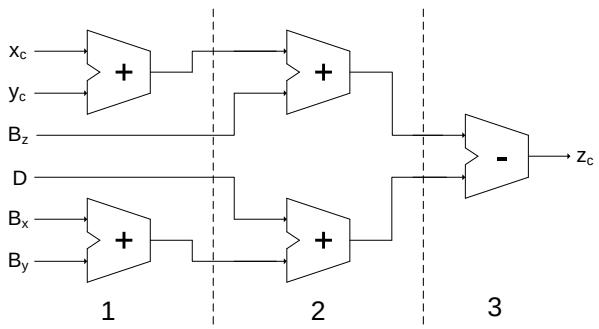


Figure 8: Structure of a optimized coded ALU for integer-addition

Figure 8 shows an optimized hardware model for

a coded integer addition. In contrast to Figure 7, this ALU consists of 4 adders and 1 subtracter, two additions are done parallel in stage 1 and 2. Due to this, the calculation time of the coded addition is reduced to $18ns$ in this way.

With this SES-coprocessor approach, the performance of the system could be increased, but on the other hand additional hardware is needed.

5.3 Migration to a FPGA architecture

FPGAs allow a flexible, low-cost solution for control functions, bridging an interface between components or simply as glue logic for a variety of customized systems. Multiple functions could be integrated into a single-chip solution to reduce board space and costs [3]. Many FPGA vendors provide powerful microcontroller soft cores so it would be obvious to integrate the microcontroller and the SES-coprocessor on a single FPGA.

5.3.1 Single core processor with SES

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

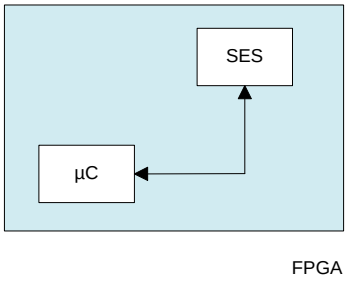


Figure 9: FPGA including microcontroller soft core and SES-coprocessor

Figure 9 shows a FPGA including a microcontroller soft core and a SES-coprocessor. The SES-coprocessor is reconfigurable and contains the operations defined in the SES approach needed by the application. There is an increasing number of microcontroller soft cores available, so there is a suitable device for many applications. Listing 1 illustrates a few examples (list is not exhaustive). Of course, a proprietary solution is also possible.

Type	Name	Vendor
8-Bit	PicoBlaze	Xilinx
16-Bit	Nios II	Altera
32-Bit	Leon	Gaisler Research
32-Bit	MicroBlaze	Xilinx

Table 1: currently available microcontroller soft cores

This approach offers a couple of advantages:

- 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 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 also be implemented.
- 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.

5.3.2 Dual core processor with SES

For higher SIL demands, this approach could be easily expanded by adding a traditional redundant hardware channel.

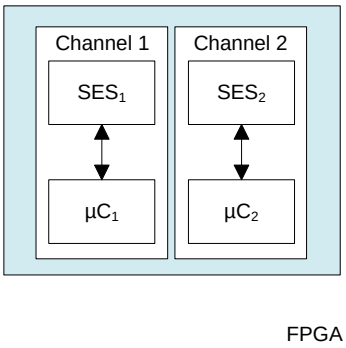


Figure 10: FPGA including dual microcontroller soft core and SES-coprocessor in a redundant hardware approach

In this approach, the existing know-how in redundant hardware design is still in use (Figure 10). The two microcontrollers have not necessarily be equal in this concept, two different soft cores can be implemented to reduce common cause failures. In this way, diverse hardware channels are achieved. This is a main advantage against a classical dual core processor, the two cores are mostly identical here. Although the SES-coprocessors could be different, alternative safety concepts e.g. ECC or Hamming-Code could be implemented. According to requirements the number and type of cores could be changed without changing any component. The voting technique for both channels is based on the normative regulations given in IEC61508[8] / ISO/CD 26262[9].

5.3.3 Further options to increase the performance

For building performant systems, this FPGA approach is also well suited. Modern FPGAs can hold more than one single microcontroller soft core and a SES-coprocessor. If further performance for special applications is needed, additional coprocessors for specific requirements could be added.

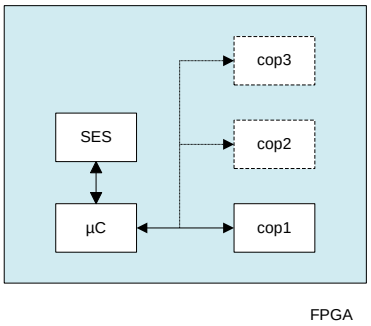


Figure 11: FPGA including microcontroller soft core, SES-coprocessor and several coprocessors for individual tasks

In Figure 11, an alternative for increasing the performance of an embedded system based on a FPGA is shown. The microcontroller SES-coprocessor approach discussed above is expanded by several coprocessors for individual tasks.

Some examples for such tasks:

- fast signal processing (DTC, FFT,...)
- fast arithmetic calculations (e.g. multiplications with high bit width)
- decoders
- multiplexers

Also extended digital hardware components could be reduced by including them into a FPGA. So this approach increases performance, fulfills safety requirements with the SES approach and reduces the hardware complexity.

5.4 Hardware - Software Codesign

Choosing a FPGA architecture including microcontroller and SES cores also affects the design methodology of the system. The possibility for hardware - software codesign allows the cooperative and concurrent development of hardware and software (co-specification, co-development, and co-verification) in order to achieve shared functionality and performance goals for a combined system[5], [6]. In this way, the decision if a task should be realized in software or in hardware could be changed rapidly and the parallelization of hard and software design reduces costs of development and the time to market.

6 Conclusion and Outlook

The FPGA approach presented in this paper offers considerable advantages. On the one hand the performance of SES guided systems could be improved by sourcing the SES part out into a dedicated hardware. For this only a low density FPGA is necessary and possibly existing system could be retained. The fast growing performance of FPGAs allows on the other hand a complete integration of microcontroller and SES-coprocessor into a single component. Also specialized coprocessors for individual tasks could be integrated. This leads to increased performance and a flexible approach for safety embedded systems. If the design methodology is adapted too, also economic advantages are possible.

References

[1] Forin, P.: Vital Coded Microprocessor Principles and Application for Various Transit Systems. IFAC Control, Computers, Communications, Paris, pp. 79-84, 1989.

[2] Ehrenberger W.: Software-Verifikation. Hanser, Munich, 2002.

[3] Actel Corporation: Reliability Considerations for Automotive FPGAs. white paper, September 2003

[4] Mottok, J., Schiller, F., Zeitler, T.: Safely Embedded Software for State Machines in Automotive Applications -to-do.

[5] Gupta, R., De Micheli, G.: Hardware-Software Cosynthesis for Digital Systems IEEE Design & Test of Computers, September 1993, pp. 29-41.

[6] Mahr, T., Gessler, R.: Hardware-Software-Codesign Vieweg, Wiesbaden 2007

[7] Mottok, J., Schiller, F., Voelkl, T., Zeitler, T.: Computer Safety, Reliability, and Security 26th International Conference, SAFECOMP 2007, Proceedings, LNCS 4680; Springer-Verlag GmbH; S. 283-288;

[8] International Electrotechnical Commission (IEC): Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems. 1998.

[9] ISO/CD 26262 International Organization for Standardization Road vehicles Functional safety actually committee draft

[10] Ozello, P.: The Coded Microprocessor Certification. International Conference on Computer Safety, Reliability and Security, SAFECOMP 1992, Springer, Munich, pp.185-190, 1992.