HW/SW Co-Design Approach to Optimize Embedded Systems on Reliability

Andreas Strasser, Philipp Stelzer, Christian Steger

Institute of Technical Informatics
Graz University of Technology
Graz, Austria
Email: {strasser, stelzer, steger}@tugraz.at

Norbert Druml

Infineon Technologies Austria AG
Graz, Austria
Email: norbert.druml@infineon.com

***Abstract*—Autonomous driving is disruptively changing the automotive industry. The importance of safety, reliability, and fault-tolerance is steadily increasing through the complexity and autonomy of self-driving cars. In the past, developers relied on the driver as a fail-safe backup to transfer the control and the responsibility to him in case of unexpected faults. In fully autonomous vehicles this backup solution will be not available anymore. This requires novel safety concepts and methodologies such as an optimization of high reliability of the systems. For optimization it is necessary to quantify different algorithm solutions from a safety point of view because this enables the possibility of comparing different solutions. In this publication, we are analyzing the consequences of different hardware and software algorithm implementations on component reliability. For this purpose we have designed two novel algorithm safety validation methodologies that allow the quantification of algorithms from a safety point of view and applied them to two independent use cases to evaluate the effects on component reliability. Both methodologies are used for optimizing the reliability of safety-critical automotive embedded systems for autonomous driving during Hardware/Software Co-Design.**

Keywords–Safety critical systems; Aging of circuits and systems; Safety Validation HW/SW; Failure-in-Time Analysis; Algorithm Safety Evaluation

I. INTRODUCTION

50 years ago started the future about fully autonomous driving. In the 1960s, Continental tested their driver-less car in the Contidrom in Germany. It was used as a prototype for tire testing to ensure constant testing conditions [2]. Nowadays, 50 years later this vision still exists in our society and Tesla has shown that autonomous driving is possible with their "Autopilot" [3]. Tesla has triggered the hype about autonomous driving and has pushed the society into a new era. This new era is changing the individual's daily routines about mobility and enables smart mobility.

Smart mobility will create a fully connected urban environment and will bring benefits to cities, better quality of life, reduced costs and more efficient energy usage [4]. To achieve the goal of autonomous driving and smart mobility, novel Advanced Driver-Assistance Systems (ADAS) are necessary. The two best-known ADAS are the Electronic Stability Control and the Anti-Lock Braking System, especially for their positive effect on active safety. Moreover, in the last years, a new generation of ADAS such as the Adaptive Cruise Control (ACC) has been established in middle class cars to avoid collisions. The next big step is introducing a comprehensive system enabling the perception of urban environment, which is one of the main goals of the PRYSTINE project [5].

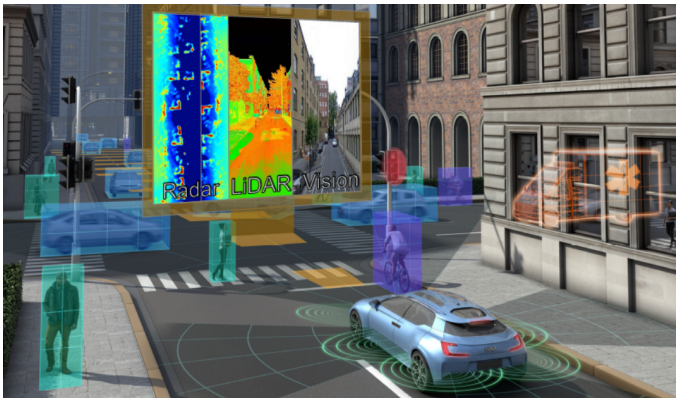

Figure 1. PRYSTINE's concept view of a fail-operational urban surround perception system [5].

PRYSTINE stands for Programmable Systems for Intelligence in Automobiles and is based on robust Radar and LiDAR sensor fusion to enable safe automated driving in urban and rural environments, as seen in Figure 1. These devices must be reliable, safe, and fail-operational to handle safety-critical situations independently [5].

In the past, developers of safety-critical automotive systems generally integrated the driver as the last safety chain link by handling over the control and the responsibility to the driver in unexpected situations or conditions. For fully autonomous vehicles, this fail-safe backup will not be available anymore because these vehicles needs to manage all critical unexpected situations on their own. This requires a rethinking of traditional safety concepts and methodologies. Novel safety-critical automotive embedded systems that will be equipped into autonomous vehicles needs to be high reliable, robust, and fail-operational [5]. One possibility that have been neglected in the past is about optimizing current systems from a safety point of view as increasing the component reliability. For this purpose, novel safety methodologies need to be developed that focus on optimizing embedded systems from a safety point of view.

This publication is an extended Version of the "FITness Assessment-Hardware Algorithm Safety Validation" [1] publication that was presented at the Ninth International Conference on Performance, Safety and Robustness in Complex Systems and Applications.

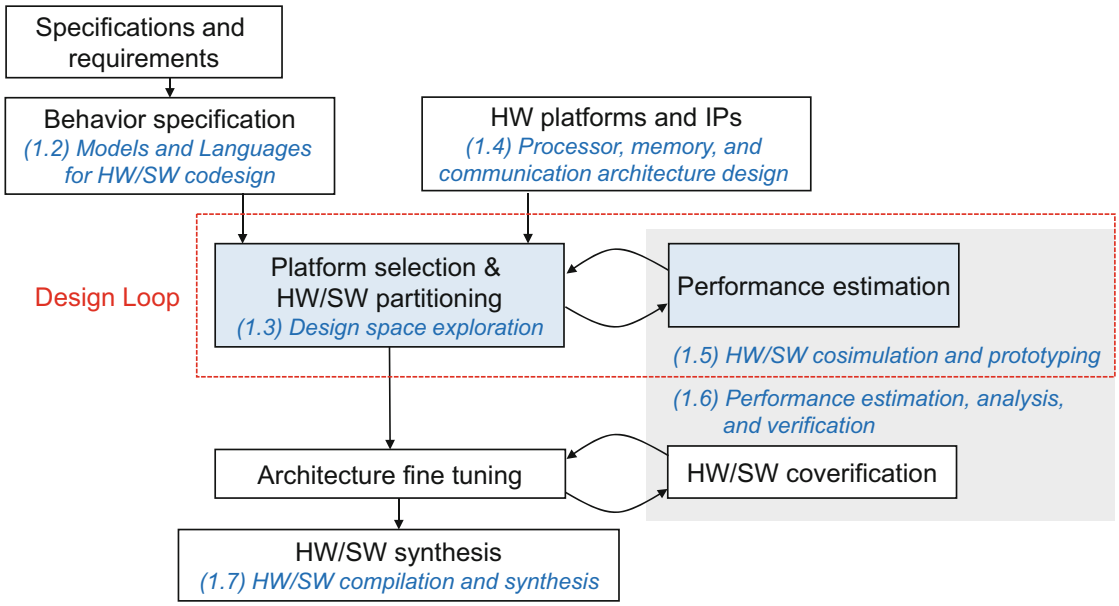

Figure 2. Overview of the HW/SW Co-Design design flow [6].

For this purpose, we will elaborate on the following two research questions:

- How can different hardware language description algorithm implementations be validated from a safety point of view?
- How can different software algorithm implementations be validated from a safety point of view?

The remainder of the paper is structured as follows. Related work will be provided in Section II. The method will be described in detail in Section IV and the results including a short discussion will be provided in Section VI. A summary of the findings will conclude this paper in Section VII.

II. Related Work

This section describes the related work in the field of component reliability considering HW/SW Co-design methodologies, software safety, hardware safety and component reliability.

A. Reliability Focused HW/SW Co-Design Methodologies

Schaumont [7] defines that the HW/SW Co-Design that is depicted in Figure 2 is used to design hardware and software components in a single design effort considering the partitioning and design of an application in terms of fixed and flexible components. In general, the most driving factor for the usage of the HW/SW Co-Design methodology is about making trade-offs, as depicted in Figure 3, between conflicting objectives such as performance, energy efficiency through fixed hardware implementations and flexibility through the usage of software implementations [7].

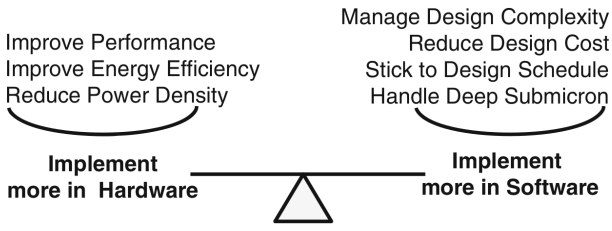

Figure 3. HW/SW Co-Design driving factors [7].

Beside the most common driving factors such as energy and performance there are also other factors that are more important in other domains such as the reliability for safety-critical embedded systems. Vargas et al. [8] introduced a novel HW/SW Co-Design approach that focus on the reliability of the overall system. Their approach decides on the basis of system reliability requirements which parts are partitioned into hardware or software including a verification of the overall reliability of the system. Vargas et al. focused in their publication on the correct function of the overall system and introduced primary hardware redundancy. Another work is the publication of Tosun et al. [9] that focus on soft errors such as bit flips. Both frameworks clearly shows that the overall reliability of safety-critical embedded systems are able to be improved by specific HW/SW Co-Design approaches. Nevertheless, both frameworks do not consider the component reliability of the hardware parts that are measured as the Failure in Time (FIT) Rate.

B. Software Design for Functional Safety

Nancy Leveson is one of the most known safety specialists and have published a book about software safety [10]. In 1995 Leveson described that in general software developers threat the computer as a stimulus-response system and that they seldom look beyond the computer. Consequently, software engineers usually constructed software without thinking about effects of the software on system safety [10]. 23 years later the perception of safety-critical software engineering has been improved and engineers are aware about the influences of software on the overall safety level [11]–[15].

Leveson [10] describes two common methodologies to

ensure run-time safety of safety-critical software systems: Dynamic and Static Analysis. Dynamic Analysis is a detection method for software errors or functional errors during run-time. Static Analysis by contrast focuses on formal errors such as race conditions or buffer overflows. Nowadays these two techniques have been advanced to frameworks that enhance the validation process.

Cruickshank et al. [11] have introduced a novel validation metrics framework for validating software safety requirements and have applied the method on a fictitious safety-critical surface-to-air missile system. Cruickshank et al. described that their framework supported the early identification of potential safety problems [11]. Baudin et al. [16] have described their novel tool for safety validation called "CAVEAT". CAVEAT is a statistical analysis tool to verify safety critical software and is used by Airbus to validate pieces of code as early as possible [16]. Michael et al. [15] also introduced a novel Hazard Analysis and Validation Metrics Framework. This framework is able to gauge the sufficiency of software safety requirements in the early software development process [15]. These frameworks illustrate the need of advanced methodologies to support safety-critical software development. However, these frameworks do not consider a validation of different algorithm implementations on the affects of component reliability.

Software algorithm validation is widely used to compare different implementations with respect to power consumption or run-time. Rashid et al. [17] have compared different sorting algorithms that are implemented in different programming languages on mobile devices. Their results clearly show that different implementations results in different power consumptions. Another example is the analysis of energy consumption of sorting algorithms on smartphones of Verma et al. [18]. Verma et al. have found out that the energy consumption depends on the data size as well as on the implemented sorting algorithm [18]. Bunse et al. have explored the energy consumption of data sorting algorithms in embedded environments and in their work different algorithms resulted in different power consumption. According to the automotive functional safety standard "ISO 26262" [19] power consumption is related to component reliability.

C. Hardware Design for Functional Safety

The validation of algorithms is an important method for achieving certain requirements such as area, power dissipation or run time. Therefore, there are numerous articles about enhancing efficiency of fault-tolerant mechanisms through algorithm substitution [20] [21] [22]. Rossi et al. analyze the power consumption of fault-tolerant buses by comparing different Hamming code implementations with their novel Dual Rail coding scheme [20]. Also, Nayak et al. emphasize the low power dissipation of their novel Hamming code components [21]. Another example is the work of Shao et al. about power dissipation comparison between the novel adaptive pre-processing approach for convolution codes of Viterbi decoders with conventional decoders [22]. Khezripour et al. provide another example for validating different fault-tolerant multi processor architectures by power dissipation [23]. Unfortunately, power dissipation is just one factor for reliability of safety-critical components and insufficient for safety validation.

The most important indicator for safety at hardware level is the component reliability, which is measured in failure in time (FIT) rates [19]. Component reliability is the main indicator for safe hardware components and describes the quantity of failures in a specific time interval, mostly one billion hours [19]. These values can be calculated by specific standards for electronic component reliability such as the IEC TR 62380 [24] or statistically collected by field tests. Oftentimes, these field test have already been conducted by the manufacturers and are compiled in specific data-sheets for component reliability [25]. For each component, the data-sheets usually contain the specific FIT Rate for a certain temperature. To determine the FIT Rate for other temperatures, the Arrhenius equation as seen in (1) can be used.

$$DF = e^{\frac{E_a}{k} \cdot (\frac{1}{T_{use}} - \frac{1}{T_{stress}}))} \quad (1)$$

where:

DF	De-rating Factor
E_a	Activation Energy in eV
k	Boltzmann Constant (8.167303 x 10^{-5} ev/K)
T_{use}	Use Junction Temperature in K
T_{stress}	Stress Junction Temperature in K

The Arrhenius Equation requires the Junction Temperature instead of Temperature values. The Junction Temperature represents the highest operation temperature of the semiconductor and considers the Ambient Temperature, Thermal Resistance of the package as well as the Power Dissipation as seen in (2).

$$T_j = T_{amb} + P_{dis} \cdot \theta_{ja} \quad (2)$$

where:

T_{amb}	Ambient Temperature
P_{dis}	Power Dissipation
θ_{ja}	Package Thermal Resistance Value

III. Problem Statement

The validation of different algorithms is crucial for designers to optimize their systems in terms of component reliability for highly robust and safe autonomous vehicles. Designers of safety-critical embedded systems should be able to pick the most safe algorithm with the advantage of lower FIT Rates. Especially for automotive Tier-1 companies lower FIT Rates imply higher component reliability, which is crucial for the economic success or failure of the whole system as profit margins are that small that every defect matters. Therefore, to support designers of safety-critical embedded systems, this publication's contributions to existing research are:

1) Developing novel methods for safety validation of hardware and software algorithms that is based on the approved ISO 26262 2nd Edition methods.
2) Applying the novel methods to quantify the differences between different algorithm implementations from a safety point of view.

IV. Component Reliability Focused HW/SW Co-Design Methodology

This section introduces two novel design processes that support designers of safety-critical embedded systems to find the most reliable solution during the HW/SW Co-Design

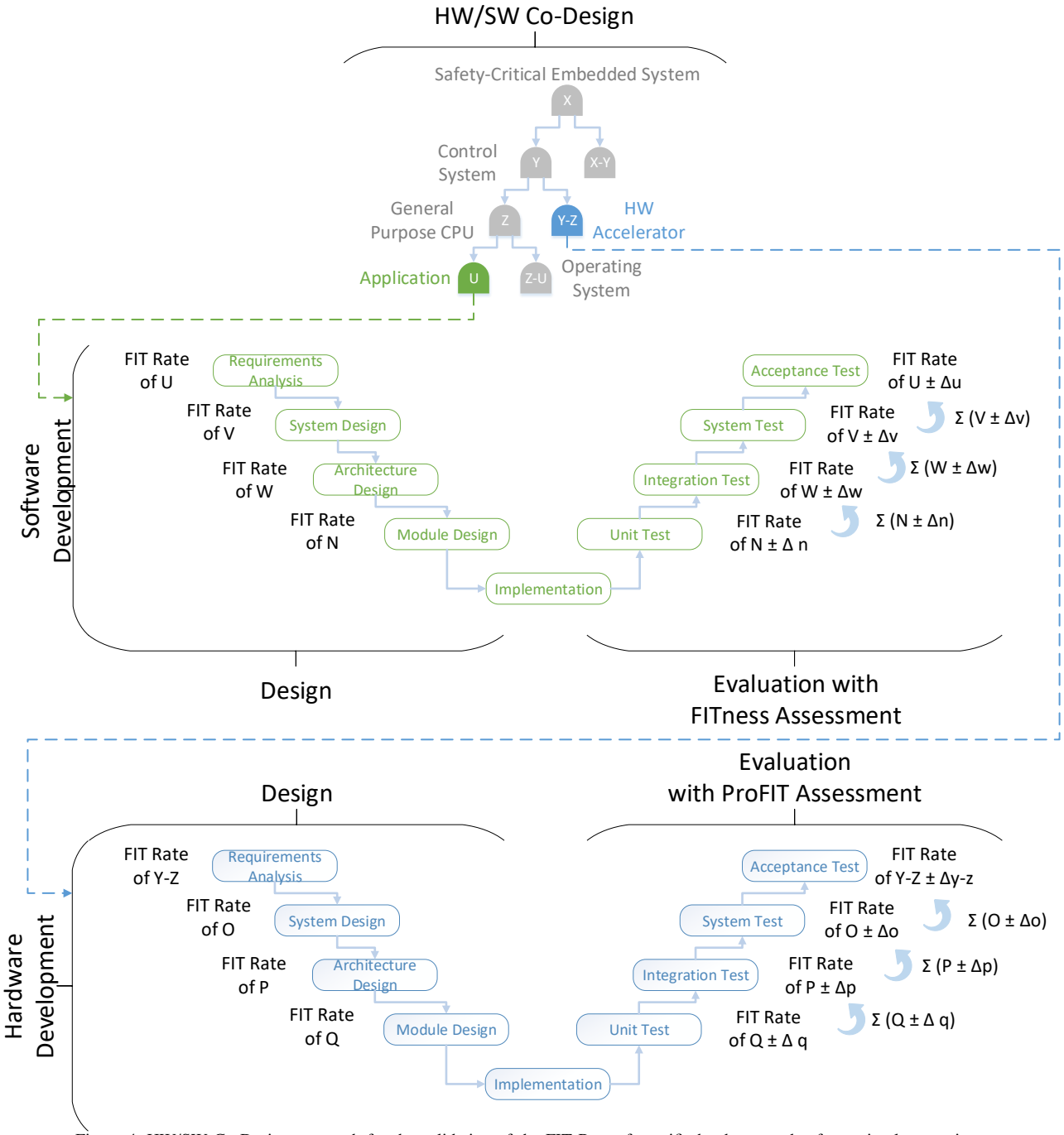

Figure 4. HW/SW Co-Design approach for the validation of the FIT Rate of specific hardware and software implementations.

process. The most reliable solution in this case is defined as the system with the lowest FIT Rate. To compare different hardware and software solutions it is necessary to measure the specific FIT Rate of each algorithm implementation. For this purpose, we need to introduce two novel measurement methodologies that enable the FIT Rate measurement. These two measurement methodologies that are presented in this publication are:

- **FITness Assessment - Hardware Reliability Evaluation** The "FITness Assessment" approach enables the FIT Rate determination of algorithms that are implemented in hardware description languages such as VHDL.
- **ProFIT Assessment - Software Reliability Evaluation** The "ProFIT Assessment" approach evaluates the FIT Rate of software implemented algorithms that are executed on micro-controller.

The FITness Assessment focuses on the estimation and validation of hardware related implementations and the ProFIT Assessment on software implementations. Both methods can easily be integrated in common HW/SW Co-Design design flows as depicted in Figure 4.

The novel HW/SW Co-Design approach that is enabled

through our two novel FIT Rate measurement approaches allows the evaluation of the FIT Rate of specific functionalities that are implemented in hardware or software. On the left side, a tree diagram of the overall safety-critical embedded system can be seen. The top leaf of the tree structure represents the whole embedded system and contains a FIT Rate of X. In the next hierarchical level the FIT Rate X is separated in the control system part and the additional hardware part that are represented with a FIT Rate of Y and X-Y. This strategy can be continued until we reach the smallest part of the overall system such as algorithms in software or hardware components. Based on this FIT Rate separation each designer and programmer is able to mind the overall FIT Rate of the system by complying with the given FIT Rate. Any deviance of a software algorithm can easily be recognized in the early phase of development and enables an intervention of the project team.

After the separation, each software programmer and hardware designer is able to determine if their solution matches the requirements of the designer considering the FIT Rate. Especially, the division of the overall FIT Rate into smaller sub-parts enables a reliability focused hardware-software development. A comparison between the designed reliability and the indeed reliability is possible through the summarization of the individual FIT Rates to the overall system. For this purpose, the individual FIT Rates of the software and hardware units are summed up to an overall system FIT Rate.

To enable this novel HW/SW Co-Design approach it is necessary to measure the FIT Rate of specific hardware and software implementations and this could be achieved by our novel hardware and software reliability evaluations called "FITness Assessment" and "ProFIT Assessment".

A. FITness Assessment - Hardware Reliability Evaluation

To validate different algorithms that are implemented in hardware description languages such as VHDL or Verilog, it is necessary to quantify the essential values. Based on the functional safety standard ISO 26262 2nd Edition's approved methods, the FIT Rate is the most important factor for safety-critical hardware components. As stated in the Related Work Section II, the De-rating Factor influences the FIT Rate and is expressed in the Arrhenius equation (1). Combined with the Temperature Junction equation it is obvious that the power dissipation is the most significant quantity that can be influenced by designers of digital circuits (see (3)).

$$DF = e^{\frac{E_a}{k} \cdot (\frac{1}{T_{use}} - \frac{1}{T_{amb} + P_{dis} \cdot \theta_{ja}})} \quad (3)$$

Consequently, by decreasing Power Dissipation the designer increases component reliability. For Field Programmable Gate Array (FPGA), the power dissipation primarily depends on static and dynamic power consumption. Based on these physical principles, our novel method FITness Assessment for algorithm safety validation on FPGAs is segmented in the following parts, as seen in Figure 5:

1) **Algorithm Implementation**
To guarantee similar conditions for different algorithms, it is necessary to implement a generic framework that allows implementing algorithms without major changes.

2) **Power Consumption Measurement**
For each algorithm, a particular measurement is recorded. It is advisable to record the generic framework without any algorithm to be able to determine the algorithms' power consumption by subtraction.

3) **Determination of Base FIT Rate**
The Base FIT Rate may be calculated by using the IEC TR 62380 [24] standard or analyzed statistically by field tests. Oftentimes, these field test have already been conducted by the manufacturers and are compiled in specific data-sheets for component reliability.

4) **De-rating Factor Calculation**
The De-rating Factor can be calculated with the Arrhenius equation and the related Thermal Junction equation as seen in (1) and (2).

5) **Identification of Effective FIT Rate**
The Effective FIT Rate reflects the Base FIT Rate for a specific temperature and can be calculated with:

$$FIT_{ef} = FIT_{base} \cdot DF \quad (4)$$

where:

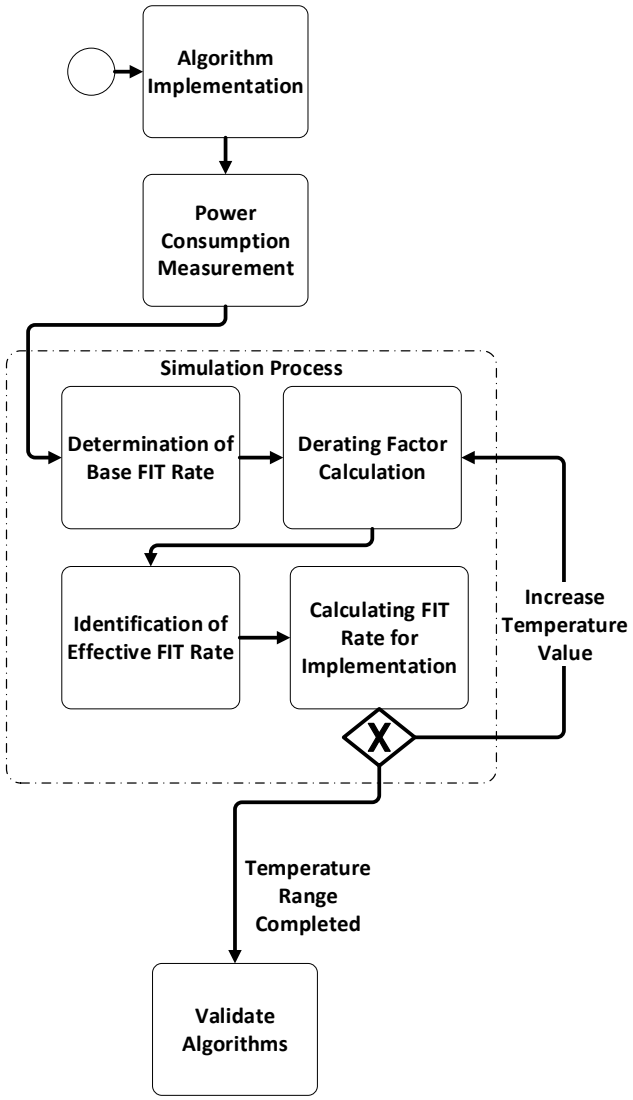

Figure 5. Workflow overview of our novel method FITness Assessment for algorithm validation from a safety point of view in Business Process Model and Notation.

FIT_{base} Base FIT Rate from FPGA Reliability Data-sheet
DF De-rating Factor as seen in (1)

6) **Calculating FIT Rate of the Implementation**
The Effective FIT Rate as seen in (4) represents the component reliability for the whole FPGA. However, an FPGA is made up of many different logic elements. Consequently, the Effective FIT Rate can be broken down into the amount used by each logical element as seen in (5).

$$FIT_{imp} = \frac{FIT_{ef}}{N_{le}} \quad (5)$$

where:
FIT_{ef} Effective FIT Rate as seen in (4)
N_{le} Total Number of Logic Elements of the specific FPGA taken out from Data-sheet

7) **Validate Algorithms**
The resulting FIT Rate of the implementation represents the FIT Rate of the specific algorithm and can be used for validation. It is advisable to measure each algorithm once at room temperature conditions and simulate the rest of the temperature range by starting with the De-rating Factor Calculation.

B. ProFIT Assessment - Software Reliability Evaluation

Validating software algorithms for safety-critical systems from a safety point of view can be obtained by using our novel "ProFIT Assessment". This method enables the impact measurement of different software algorithm implementation on component reliability. Our novel method is using approved methods from the functional safety standard ISO 26262 2nd Edition [19] of the automotive industry. As a starting base we have used equation (3). This equation represents the impacts on the component FIT Rate as a function of the power consumption. In Related Work we have introduced scientific results that clearly shows that different software algorithm implementations results in different power consumption. Therefore, the De-rating Factor can be used to determine the specific software algorithm FIT Rate. Our "ProFIT Assessment" is using these relations and can be separated into five parts:

1) **Implementation**
Different algorithms will be implemented in software. For better results and accuracy it is advisable to implement a general framework where the algorithms can be exchanged without any major changes. The framework will be compiled and programmed onto a specific micro-controller. In general any micro-controller can be used but it is advisable to look for public available component reliability data-sheets.

2) **Measurement**
In this step the software algorithms will be run on micro-controller and the power dissipation is recorded. This step will be repeated for each implementation. As an output result a measurement report is created, which contains the measurement setup, the used micro-controller, software algorithm implementation, power consumption and ambient testing temperature. These details are necessary for further analysis.

3) **Calculating FIT**
The idea behind this step is that each software algorithm needs a specific amount of time and the power consumption is measured at a specific sampling rate. For each sample we are calculating the specific Base FIT Rate and relates it to the sampling duration. Summing up all the individual FIT Rates of each timeslice results in the specific FIT Rate of the software algorithm implementation for a specific temperature. The impacts of the different implementations over the whole temperature range will be determined through the simulation process afterwards.

a) **Junction Temperature**
At first we are calculating the specific Junction Temperature for the ambient testing temperature as seen in (2).

b) **De-rating Factor**
Secondly the specific De-rating Factor is determined with the Arrhenius equation as seen in (1).

c) **Base FIT Rate**
The base FIT Rate can be determined by multiplying the base FIT Rate from compo-

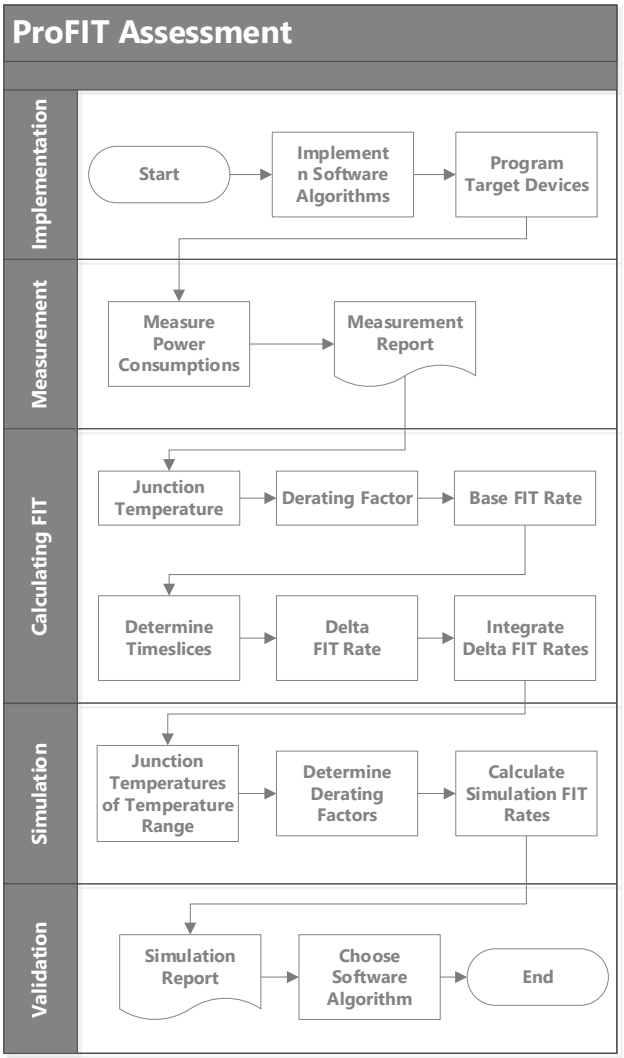

Figure 6. Work flow overview of our novel "ProFIT Assessment" method for software algorithm validation from a safety point of view.

nent reliability data-sheet with the De-rating Factor.

$$FIT_{Base} = DF \cdot FIT_{Ds} \quad (6)$$

where:

DF De-rating Factor as seen in (1)
FIT_{Ds} Base FIT Rate of Component Reliability Data-sheet

d) **Determine Time-slices**
In this step the Base FIT Rate will be adapted to the specific run-time.

$$FIT_{Timeslice} = FIT_{Base} \cdot \frac{T_{Sampling}}{T_{Runtime}} \quad (7)$$

where:

FIT_{Base} Base FIT Rate as seen in (6)
$T_{Sampling}$ Measurement Sampling Time
$T_{Run\text{-}time}$ Run-time of the Measurement

e) **Integrate Delta FIT Rates**
To determine the Software FIT Rate it is necessary to accumulate all individual Time-slices.

$$FIT_{Algorithm} = \sum_{1}^{n} FIT_{Ts} \quad (8)$$

$$n = \frac{T_{Runtime}}{T_{SamplingRate}} \quad (9)$$

where:

FIT_{Ts} Time-slice FIT Rate as seen in (7)
$T_{Sampling}$ Measurement Sampling Time
$T_{Run\text{-}time}$ Run-time of the Measurement

4) **Simulation**
The simulation step is necessary to determine the software algorithm FIT Ratio over the whole operational temperature range. The power consumption variation will be neglected because it affects all algorithm implementations equally.

a) **Junction Temperatures of Temperature Range**
This step is similar as during the Calculating FIT Rate step except the use of the whole operational temperature range.

b) **Determine De-rating Factors** This step is equal as seen in (1).

c) **Calculate Simulation FIT Rates**
This step is equal as seen in (6).

5) **Validation**
After the simulation there will be a Simulation Report with the specific FIT Rates for the whole operational temperature range. This can be used as a decision support to pick the most reliable software algorithm implementation.

V. Test Setup

This section describes the practical results of this publication by introducing the testing environment and the final results of the experiments. The validation of the HW/SW Co-design approach was divided in a software and hardware part and both parts have been validated independently.

A. FITness Assessment Evaluation Setup

In our research question, we analyze the differences between Single Error Correction - Double Error Detection (SEC-DED) and Double Error Correction (DEC). For this purpose, we chose the Hamming code for SEC-DED as this code is recommended in the new ISO 26262 2nd Edition and the BCH-code for DEC, especially because other ECC algorithms are often based on this concept and both algorithms fulfill the following requirements:

- 32 Bit data size
- Combinatorical Logic
- Including Fault Injection Module
- SEC-DED or DEC Functionality

The generic algorithm framework contains a test-bench with an automatic up-counter as well as a validator (see Figure 8). Both algorithms can be exchanged in the framework without any major changes. This enables a precise validation from a safety point of view.

In our test setup, we use the MAX1000 - IoT Maker Board by Trenz Electronic. This device is a small maker board for prototyping with sparse additional components. The main controller is the MAX10 10M08SAU169C8G, an FPGA device by Intel. For our research, the main advantages of using this board are:

- Small amount of additional hardware components
- Availability of Reliability Data-sheet

This board also contains an FTDI chip that draws about 50 mA on average, which we will subtract out for our analysis. The power consumption measurement is performed by the Mobile Device Power Monitor of Monsoon Solutions. The big advantage of this power monitor is the direct measurement of USB devices. The entire measurement setup is shown in Figures 7 and 9 and contains the following software and hardware parts:

- Quartus Prime 18.0 (Intel)
- Power Tool 5.0.0.23 (Monsoon Solutions)
- Mobile Device Power Monitor (Monsoon Solutions)
- MAX1000 - IoT Maker Board (Trenz Electronic)

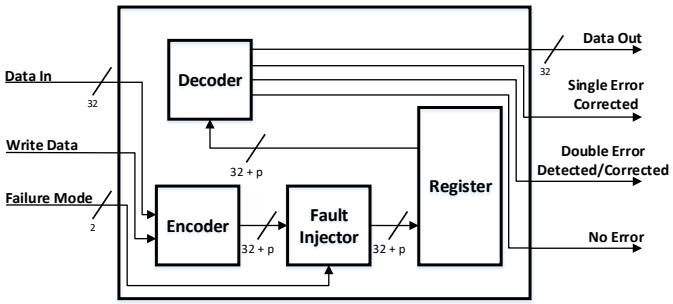

Figure 7. Pin configuration of both algorithms including an overview of functional blocks inside.

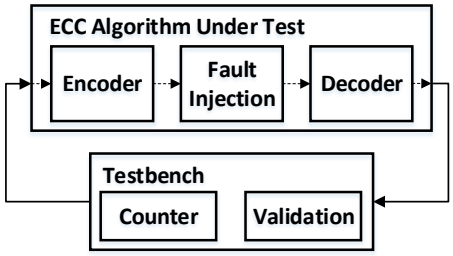

Figure 8. General framework for ECC algorithm validation including test-bench and ECC algorithm.

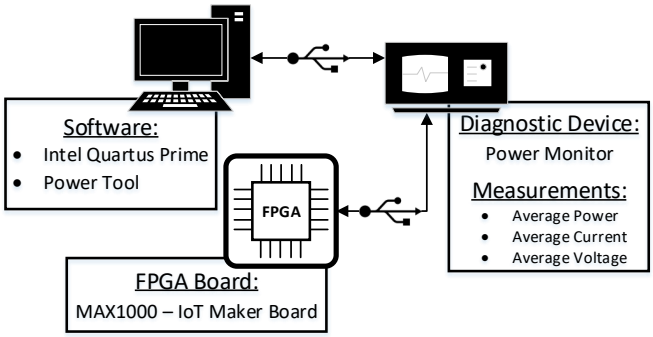

Figure 9. Overview of the entire measurement setup including software and hardware components.

B. ProFIT Assessment Evaluation Setup

For testing purpose we have chosen sorting algorithms as test candidates. The reasons for us are:

- Very often used
- Easy to understand
- Many different algorithms available
- Comparable results of power consumption available as seen in Section II-B

The sorting algorithms we chose are widely used and known and are known as:

- Binary Insertion Sort
- Heapsort
- Insertion Sort
- Mergesort
- Quicksort
- Shell Sort

All sorting algorithms were implemented in C programming language and programmed onto a micro-controller. For the micro-controller we have chosen the "MSP430 FR5969" from Texas Instruments by the following reasons:

- Measure Power Consumption with EnergyTrace++ Technology in "Code Composer Studio"
- Qualified for automotive usage
- Low-Power Device
- FIT Rates publicly available

As a operational temperature range for the simulation part we have chosen $-40°C$ up to $140°C$. This range is higher than the recommended operating conditions from the data-sheet but for our tests it is not relevant.
Test Setup Summary:

- Code Composer Studio 8.1
- MSP430 FR5969
- 6 different Sorting Algorithms
- 400 Numbers to Sort
- $-40°C$ up to $140°C$ Temperature Range for Simulation

VI. Results

A. FITness Assessment Evaluation

This section summarizes our results of the comparison of SEC-DED and DEC ECC algorithm. The validation was performed with our novel FITness Assessment method for algorithm validation from a safety point of view as described in Section IV.

The first algorithm we implemented was the Hamming code, which is a SEC-DED ECC algorithm. The implementation reserves 45 logic elements of the used FPGA and the whole board has an average power dissipation of 571.78 mW. With the second BCH-code DEC ECC algorithm, the board consumes an average of 599.05 mW and assigns 65 logic elements. The first result shows a difference between both algorithms in logic elements as well as in power dissipation resulting in a varying FIT Rate. The next step is the simulation process over the whole temperature range. We selected a temperature range between -40°C and 125°C and the values of Table I were used for the simulation process. In our simulation we neglected the alteration of power dissipation through temperature because it would affect both ECC implementations evenly.

Figure 10 points out that both algorithms vary in their FIT Rate and rise exponentially with increasing temperature. The FIT Rate may be neglected for temperatures up to 40 °C.

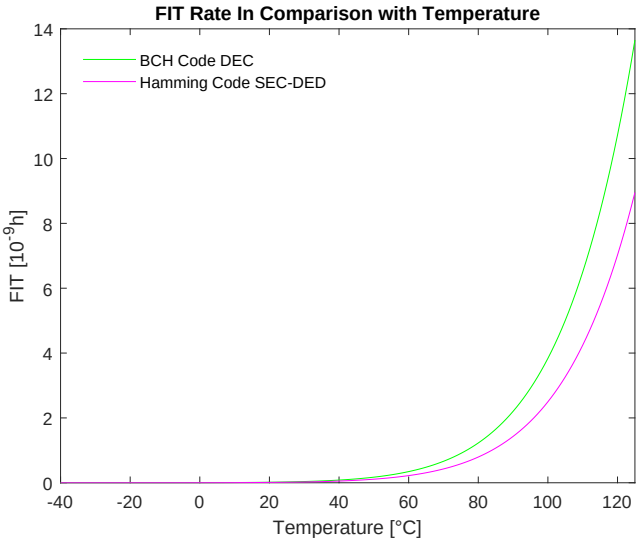

Figure 10. Simulation results of the resulted FIT Rates between -40°C and 125°C for both ECC implementations.

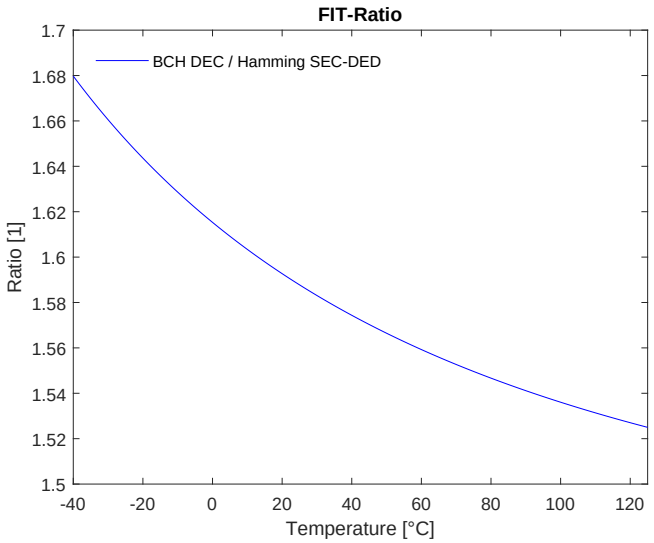

Figure 11. Overview of the FIT Rate overhead between SEC-DED and DEC ECC algorithm.

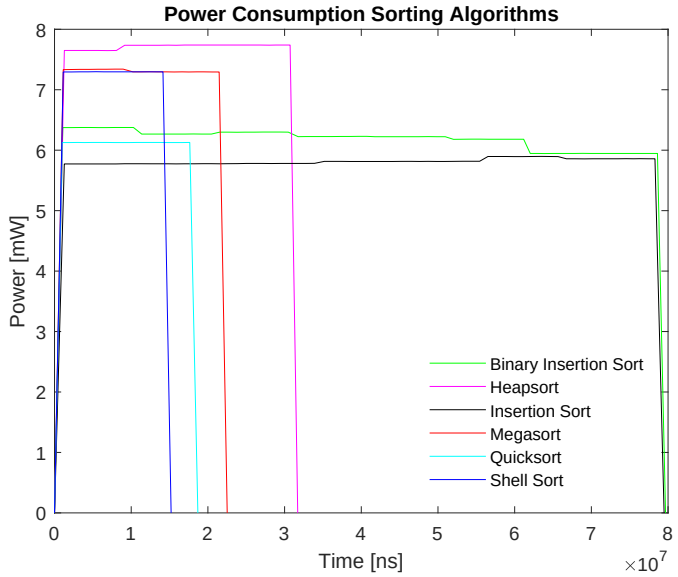

Figure 12. Power consumption results of the implemented sorting algorithms at 25°C ambient temperature.

TABLE I. RESULTS OF THE RESERVED LOGIC ELEMENTS AND AVERAGE TOTAL POWER DISSIPATION OF BOTH ECC IMPLEMENTATIONS.

	Hamming Code	BCH-Code
Used Logic Elements	45	65
Total Average Power Dissipation	571.78 mW	599.05 mW

The Hamming code with SEC-DED shows a better FIT Rate indicating more reliability of the hardware components which results in a higher safety level. The reason for this difference is the greater number of logic elements used for the DEC ECC algorithm and the resulting increase of power dissipation. The higher power dissipation results in a higher Thermal Junction temperature as seen in (2), which leads to a higher FIT Rate.

Both algorithms were implemented without any safety measures. This means that any damage to the Logic Element of the FPGA leads to failure of the whole ECC algorithm and the safe memory block. The ECC algorithm is the measure against SEU related altered flip flops inside the memory block, which decreases the specific FIT Rate of the memory block. The results of Figure 10 do not represent the FIT Rates of the memory block but the FIT Rate of the pure ECC implementation. It is important to understand that the ability of more bit error correction is not considered for the algorithm validation because it only positively influences the FIT Rate of the memory block.

Moreover, it is important to understand that the absolute values of the FIT Rate always correlate to a specific FPGA. Consequently, it is advantageous to look at the ratio between the algorithms because this gives a better overview of the overhead. The SEC-DED/DEC ECC FIT Ratio is depicted in Figure 11. The FIT Ratio overhead of the DEC ECC algorithm is slighly decreasing with increasing temperature, which is negligible in practice.

We recommend using the Hamming code algorithm for SEC-DED error correction for 32 bit memory size registers in automotive LiDAR systems. The SEC-DED algorithm used in our experiment resulted in a FIT Rate that was at least 52% lower than the DEC ECC algorithm.

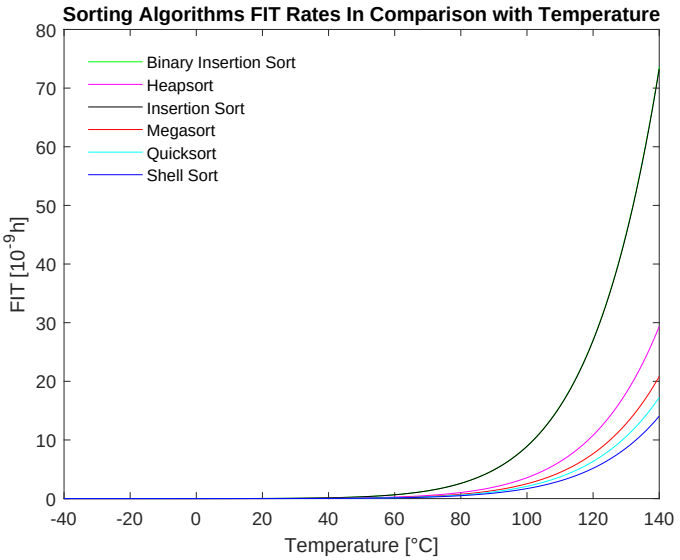

Figure 13. Simulation results of the sorting algorithms between −40°C and 140°C.

B. ProFIT Assessment Evaluation

In this section we are presenting our results of applying our novel "ProFIT Assessment" on sorting algorithms. This method enables the possibility to validate software algorithms from a safety point of view. It is important to understand that we are not comparing sorting algorithms instead we are applying our method on the sorting algorithms.

All algorithms are implemented in C and were tested on the "MSP430 FR5969" micro-controller board. This board has the possibility to measure the power consumption of each algorithm directly in the "Code Composer Studio". Table II gives an overview about our power measurement results of the implemented sorting algorithms. These algorithms were implemented in C and were executed on the "MSP430 FR5969" micro-controller board. The "Shell Sort" algorithm was in our test case the fastest at run-time and needed the least energy during run-time. Figure 12 shows the results of our power consumption measurements. In our setup "Shell Sort"

TABLE II. **Overview of the Power Consumption measurements of all C implemented sorting algorithms at $25°C$ ambient temperature.**

	Average Power in mA	Energy in uJ	Time in ms
Binary Insertion Sort	6.18	438.2	77.53
Heapsort	7.72	178.4	31.71
Insertion Sort	5.82	440.0	79.48
Mergesort	7.31	124.8	22.52
Quicksort	6.12	60.7	18.69
Shell Sort	7.30	58.5	15.20

TABLE III. **Results of the algorithm FIT Rates calculation of the implemented sorting algorithms on the MSP430 FR5969 micro-controller board.**

	FIT Rate in 10^{-9}
Binary Insertion Sort	1.87204922
Heapsort	0.747313371
Insertion Sort	1.865387949
Mergesort	0.529712728
Quicksort	0.438742916
Shell Sort	0.357627573

had the best run-time performance and "Binary Insertion Sort" had the worst run-time. This result clearly shows that different algorithm implementations result in different power consumptions. With these results the specific algorithm FIT Rates can be determined with the equations that have been introduced in IV-B.

The provided Table III represents the FIT Rate for a specific ambient temperature. In our case we have calculated the FIT Rate for the test ambient temperature of $25°C$. For other temperatures a simulation over the whole temperature range is necessary. For this purpose we have used the Arrhenius equation as seen in (1). In Figure 13 the FIT Rates of the implemented algorithms is displayed with the behavior over the whole temperature range. It can be seen that "Shell Sort" has the best FIT Rate over the whole temperature range and "Binary Insertion Sort" is the worst. For temperatures up to $50°C$ it does not matter what kind of algorithm is used but afterwards it has an affect on the component reliability and therefore on the overall safety level.

VII. Conclusion

In this publication, we introduced a novel HW/SW Co-Design approach that is optimizing the reliability of safety-critical automotive systems. To enable this approach, we have introduced two novel reliability evaluation methodologies that are able to analyze the impacts of different hardware and software algorithms on the component reliability also called Failure-In-Time Rate.

The hardware related part of the publication introduced the FITness Assessment, a novel component reliability hardware evaluation methodology and this was used to evaluate two different error correction code algorithms (SEC-DED and DEC ECC) from a safety perspective. The software related part introduced the ProFIT Assessment, a novel component reliability software evaluation methodology and this was used to analyze the impacts of six different sorting algorithms (Binary Insertion Sort, Heapsort, Insertion Sort, Mergesort, Quicksort and Shell Sort) to the overall component reliability of the micro-controller part of the overall embedded system.

Both methods are based on approved methods of the novel automotive functional safety standard ISO 26262 2nd Edition. The result clearly shows that different hardware and software algorithms lead to different FIT Rates.

FITness Assessment allowed the measurement of each algorithm's specific FIT Rate, facilitating the selection of the most reliable ECC algorithm. Our case shows a DEC-ECC algorithm that has a higher FIT Rate than the SEC-DED ECC algorithm.

ProFIT Assessment focuses on evaluating component reliability of software algorithms on micro-controllers. In our results we have showed that safety validation of software algorithms is possible and that different algorithm implementations can result in different component reliability. These differences should not be neglected because they have an impact from a safety point of view.

The FIT Rate reflects component reliability, which is an important hardware indicator for safety. These differences should not be neglected from a safety as well as from a business point of view. The FIT Rate also statistically indicates the amount of defective components, which is an economically important indicator as lower FIT rates also result in less defect components.

Fault-tolerance, safety and reliability will become more and more important in the next years because of autonomous driving. The novel introduced FITness Assessment enables the validation of different hardware algorithms to be able to select the most reliable one, which helps improve the overall safety level of the automotive vehicle by increasing component reliability. "ProFIT Assessment", the second method we introduced in this publication enables the possibility to validate the FIT Rate of software algorithm implementations and enables the possibility to choose the most reliable one. Both methodologies can be used for HW/SW Co-design for optimizing safety-critical automotive embedded systems from a safety point of view.

VIII. Acknowledgments

The authors would like to thank all national funding authorities and the ECSEL Joint Undertaking, which funded the PRYSTINE project under the grant agreement number 783190.

PRYSTINE is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2018 and April 2021 (grant number 865310). More information: https://iktderzukunft.at/en/.

References

[1] A. Strasser, P. Stelzer, C. Steger, and N. Druml, "FITness Assessment-Hardware Algorithm Safety Validation," in The Ninth International Conference on Performance, Safety and Robustness in Complex Systems and Applications, PESARO 2019, pp. 12–17.

[2] "50 Jahre fahrerloses Fahren: Pressematerial," 2014, URL: https://publicarea.admiralcloud.com/p/a49d3d8ba92f3cdbfa864f [accessed: 2019-11-11].

[3] M. Dikmen and C. Burns, "Trust in autonomous vehicles: The case of Tesla Autopilot and Summon," in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct 2017, pp. 1093–1098.

[4] R. Faria, L. Brito, K. Baras, and J. Silva, "Smart mobility: A survey," in 2017 International Conference on Internet of Things for the Global Community (IoTGC), July 2017, pp. 1–8.

[5] N. Druml, G. Macher, M. Stolz, E. Armengaud, D. Watzenig, C. Steger, T. Herndl, A. Eckel, A. Ryabokon, A. Hoess, S. Kumar, G. Dimitrakopoulos, and H. Roedig, "PRYSTINE - PRogrammable sYSTems for INtelligence in AutomobilEs," in 2018 21st Euromicro Conference on Digital System Design (DSD), Aug 2018, pp. 618–626.

[6] S. Ha and J. Teich, Handbook of hardware/software codesign. Springer Publishing Company, Incorporated, 2017, ISBN: 978-94-017-7268-6.

[7] P. R. Schaumont, A practical introduction to hardware/software codesign. Springer Science & Business Media, 2012, ISBN: 978-1-4614-3736-9.

[8] F. Vargas, E. Bezerra, L. Wulff, and D. Barros, "Optimizing HW/SW codesign towards reliability for critical-application systems," in Proceedings Seventh Asian Test Symposium (ATS'98)(Cat. No. 98TB100259). IEEE, 1998, pp. 52–57.

[9] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, Y. Xie, and W.-L. Hung, "Reliability-centric hardware/software co-design," in Sixth international symposium on quality electronic design (isqed'05). IEEE, 2005, pp. 375–380.

[10] N. G. Leveson and J. Diaz-Herrera, Safeware: system safety and computers. Addison-Wesley Reading, 1995, vol. 680, ISBN: 978-0201119725.

[11] K. J. Cruickshank, J. B. Michael, and M. Shing, "A Validation Metrics Framework for safety-critical software-intensive Systems," in 2009 IEEE International Conference on System of Systems Engineering (SoSE), May 2009, pp. 1–8.

[12] W. Ahmad, U. Qamar, and S. Hassan, "Analyzing different validation and verification techniques for safety critical software systems," in 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Sept 2015, pp. 367–370.

[13] N. G. Leveson and P. R. Harvey, "Analyzing Software Safety," IEEE Transactions on Software Engineering, vol. SE-9, no. 5, Sept 1983, pp. 569–579.

[14] E. M. E. Koursi and G. Mariano, "Assessment and certification of safety critical software," in Proceedings of the 5th Biannual World Automation Congress, vol. 14, June 2002, pp. 51–57.

[15] J. B. Michael, M. Shing, K. J. Cruickshank, and P. J. Redmond, "Hazard Analysis and Validation Metrics Framework for System of Systems Software Safety," IEEE Systems Journal, vol. 4, no. 2, June 2010, pp. 186–197.

[16] P. Baudin, A. Pacalet, J. Raguideau, D. Schoen, and N. Williams, "Caveat: a tool for software validation," in Proceedings International Conference on Dependable Systems and Networks, June 2002, p. 537.

[17] M. Rashid, L. Ardito, and M. Torchiano, "Energy Consumption Analysis of Algorithms Implementations," in 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Oct 2015, pp. 1–4.

[18] M. Verma and K. Chowdhary, "Analysis of Energy Consumption of Sorting Algorithms on Smartphones," in Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT), 2018, pp. 26–27.

[19] I. n. E. ISO, "Draft 26262 2nd Edition: Road vehicles-Functional safety," International Standard ISO/FDIS, vol. 26262, 2018.

[20] D. Rossi, A. K. Nieuwland, S. V. E. S. van Dijk, R. P. Kleihorst, and C. Metra, "Power Consumption of Fault Tolerant Busses," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 5, May 2008, pp. 542–553.

[21] V. S. P. Nayak, C. Madhulika, and U. Pravali, "Design of low power hamming code encoding, decoding and correcting circuits using reversible logic," in 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), May 2017, pp. 778–781.

[22] W. Shao and L. Brackenbury, "Pre-processing of convolutional codes for reducing decoding power consumption," in 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, March 2008, pp. 2957–2960.

[23] H. Khezripour and S. Pourmozaffari, "Fault Tolerance and Power Consumption Analysis on Chip-Multi Processors Architectures," in 2012 Seventh International Conference on Availability, Reliability and Security, Aug 2012, pp. 301–306.

[24] T. IEC, "Iec 62380," Reliability data handbook–universal model for reliability prediction of electronics components, PCBs and equipment (emerged from UTEC 80-810 or RDF 2000), 2004.

[25] "Intel Reliability Report," 2018, URL: https://www.intel.com/content/www/us/en/programmable/support/quality-and-reliability/reports-tools/reliability-report/rel-report.html [accessed: 2019-11-11].