PESARO 2020

The Tenth International Conference on Performance, Safety and Robustness in Complex Systems and Applications

ISBN: 978-1-61208-773-3

February 23 - 27, 2020

Lisbon, Portugal

PESARO 2020 Editors

Claus-Peter Rückemann, Westfälische Wilhelms-Universität Münster (WWU)/
DIMF / Leibniz Universität Hannover, Germany

PESARO 2020

Forward

The Tenth International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO 2019), held between February 23-27, 2020 in Lisbon, Portugal, continued a series of events dedicated to fundamentals, techniques and experiments to specify, design, and deploy systems and applications under given constraints on performance, safety and robustness.

There is a relation between organizational, design and operational complexity of organization and systems and the degree of robustness and safety under given performance metrics. More complex systems and applications might not be necessarily more profitable, but are less robust. There are trade-offs involved in designing and deploying distributed systems. Some designing technologies have a positive influence on safety and robustness, even operational performance is not optimized. Under constantly changing system infrastructure and user behaviors and needs, there is a challenge in designing complex systems and applications with a required level of performance, safety and robustness.

We welcomed academic, research and industry contributions. The conference had the following tracks:

- Methodologies, techniques and algorithms
- Applications and services

We take here the opportunity to warmly thank all the members of the PESARO 2020 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to PESARO 2020. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also thank the members of the PESARO 2020 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that PESARO 2020 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the areas related to performance, safety and robustness in complex systems. We also hope that Lisbon, Portugal provided a pleasant environment during the conference and everyone saved some time to enjoy the historic charm of the city.

PESARO 2020 Chairs

PESARO Steering Committee
Wolfgang Leister, Norsk Regnesentral (Norwegian Computing Center), Norway
Mohammad Rajabali Nejad, University of Twente, the Netherlands

PESARO Industry/Research Advisory Committee
Jean-Pierre Seifert, TU Berlin & FhG SIT Darmstadt, Germany
Roger Rivett, Jaguar Land Rover, UK

PESARO 2020

Committee

Roger Rivett, Jaguar Land Rover, UK
Joel Scheuner, Chalmers - University of Gothenburg, Sweden
Jean-Pierre Seifert, TU Berlin & FhG SIT Darmstadt, Germany
Omar Smadi, Iowa State University, USA
Kumiko Tadano, NEC Corporation, Japan
Yulei Wu, University of Exeter, UK
Piotr Zwierzykowski, Poznan University of Technology,Poland

Table of Contents

Accelerating Real-time Processing of Articles by Using an OpenCL-based FPGA for the OSS Syntactic Parser SyntaxNet

Yoshiki Kurokawa, Yuichiro Aoki, Yuki Kondo, Yaoko Nakagawa
Research and Development Group, Center for Technology Innovation - Digital Technology
Hitachi, Ltd.
1-280, Higashi-Koigakubo, Kokubunji, 185-8601, Tokyo, Japan
Email: yoshiki.kurokawa.ee@hitachi.com, yuichiro.aoki.jk@hitachi.com, yuki.kondo.fe@hitachi.com, yaoko.nakagawa.gn@hitachi.com

***Abstract*— To improve customer satisfaction is necessary to provide services that enable real-time responses to complaints for call-center operations. The real-time parsing of complaint documents is more important for the real-time responses. Using the Open Source Software (OSS) syntactic parser SyntaxNet as a vehicle, a high-speed method using FPGA and OpenCL to achieve throughput of 700 words/s (required for real-time processing) is proposed. According to the results of the SyntaxNet analysis, matrix size (which changes dynamically according to the progress of the analysis) was found to be a performance determining factor. The proposed method was evaluated using public data, and the evaluation results confirmed throughput of 661 words/s, which almost met the requirement. As a result, the prospect of realization of a real-time complaint document analysis service for call centers was obtained.**

Keywords-FPGA; OpenCL; Syntactic parser.

I. Introduction

For all companies, customer complaints point out problems with products and services of companies, and they provide important information for developing products and services with higher quality. In general, a complaint from a customer is first classified and extracted from a large amount of inquiry information. The content of the complaint is then analyzed, and countermeasures are investigated by customer-complaint analysis. Parsing is the most-important process for classification and analysis of complaints. The result of parsing is the input for a series of analysis processes such as grasping meaning, classifying sentences, and summarizing contents [1]. Referring to the "three-second rule", [2] that is the rule of the response time of a web site, the parsing complaint sentences part takes at least one second of the entire complaint-classification process (taking three seconds). Since the size of the complaint text is unknown, the text size is assumed as a general text size. The general text size of English-language news articles and magazine articles is an average of 500 words and 900 words, respectively [3]. If a general article is assumed as a news article or a magazine article, so general text size is assumed 700 words length, that is taken between 500 words of a news article and 900 words of a magazine article. The required processing throughput would be 700 words/s to process one document per second. Therefore, the goal of this study is to increase the processing throughput of the syntactic parser to 700 words/s.

Most of the appreciation of syntactic parser, immediate processing is required at an edge computer. Processing at the edge computer requires a hardware accelerator with low power consumption and excellent processing capability. Among hardware accelerators, a Field-Programmable Gate Array (FPGA) is known to have low power consumption and high power efficiency. As a logic circuit, an FPGA enables offload processing, so it is structurally power efficient. This study's purpose is to speed up OSS application SyntaxNet by FPGA, and to check offload feasibility.

The contributions of this study are that after probing SyntaxNet, we found that most of the SyntaxNet execution time is spent on matrix multiplication in Section II, to address that issue, a method for matrix multiplication with a high-speed external device FPGA, is proposed in Section III, and by evaluating the SyntaxNet execution performance, it is shown that SyntaxNet with FPGA can process a general sentence in about 1 second in Section IV, then the execution time was shortened and SyntaxNet was accelerated.

TABLE I. SYNTACTIC PARSER COMPLISON

Name	Cabocha	KNP	Stanford CoreNLP	SyntaxNet
Accuracy	89.29%	93.41%	93.91%	94.24%
Throughput	105,000 words/s	103 words/s	145 words/s (8 thread)	307 words/s ↓ (This study) 661 words/s
Functions	Dependency	Dependency Case Reference	Dependency Morpheme CFG Named Entry Extraction	Dependency Morpheme CFG
Language	Only Japanese	Only Japanese	English, 10 languages	Japanese, English 40 languages
Dependencies			Stanford Dependencies Universal Dependencies	Universal Dependencies
License	LGPL	BSD	GPL v3	Apache License v2.0
Algorithm	SVM	Rule basde	Transition-based + Neural Net	Transition-based + Neural Net
Software Language	C	C	Java	Python+C

II. State Of The Art

SyntaxNet [4] is an open-source syntax analyzer announced by Google in 2016. To clarify the position of SyntaxNet used in this study as a parser, it was compared with other parsers, namely, Cabocha [10] and KNP [11] [12] as parsers dedicated to Japanese and Stanford CoreNLP [13] and SyntaxNet as parsers widely used for other languages. The features and performances of those parsers are compared with SyntaxNet in TABLE I. Processing speed was calculated by measuring execution time ourselves. The other parameters are based on previously reported research. In terms of processing speed, Cabocha surpassed the other parsers with throughput of 105K words/s, KNP achieved 105 words/s, Stanford CoreNLP achieved 145 words/s, and SyntaxNet before speedup achieved 307 words/s. However, Stanford CoreNLP uses 8 threads, while the others use one thread. Accuracy of Cabocha is about 4% lower than the other parsers. KNP Stanford CoreNLP, and SyntaxNet all achieve accuracy of over 93% and show similar values for processing speed and accuracy. For Japanese, Cabocha and KNP are often used, but it looks like they are properly used according to accuracy and function. Stanford CoreNLP is popular for English and other languages, but SyntaxNet uses the same syntax rules as Stanford CoreNLP, and it surpasses the others in terms of number of supported languages and performance, so it may be used in the future. Therefore, we think that our study's speeding up SyntaxNet is relatively fast and the study is effective.

III. SyntaxNet

The SyntaxNet uses a transition-based algorithm [5] for syntactic parsing and a neural network for the decision process. SyntaxNet is overviewed in Figure 1. Parsey McParseface, a model running on SyntaxNet has demonstrated an analysis accuracy of 97.52% [4]. The interior of SyntaxNet is largely divided into a part that executes a transition-based algorithm (written in C ++) and a part that uses python and tensorflow (written in C ++) to execute judgments the next processing by using a neural network.

The transition-based algorithm is a kind of parsing algorithm that uses a state machine, stack, and buffer to parse sentences. First, a sentence is input to the sentence buffer, stack one word to stack at a time from the first word of the sentence, make a judgement on the top two words of stack, and one of the three actions is selected as a result of the judgment. This judgement and action are repeated after all the words disappear from the stack. After disappearing all words, an action sequence and dependency relationship of words is appeared. The process is completed with the result of the relationship of their words. In the Figure, the transition-based part performs other jobs except judgment.

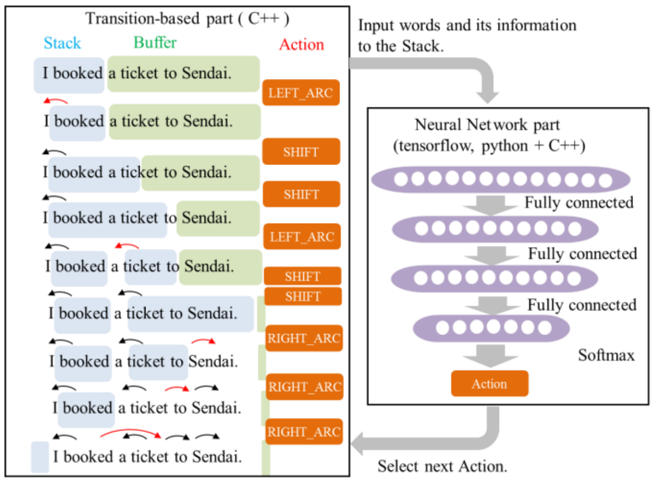

Figure 1. SyntaxNet algorithm

Information concerning the top-two words on the stack is sent to the neural network that performs only the judgment job, and the result of judgment is sent to the transition-based part. According to the result of execution-time analysis of SyntaxNet by Intel Vtune™ Amplifier, the tensorflow matrix-multiplication library Eigen [6] GEneral Block Panel (GEBP) uses 73% of the processing time, as shown by the pie chart in Figure 2. Since the GEBP is used for matrix multiplication, then matrix multiplication uses for 73% of the total processing time. In consideration of those results, the aim of the present study was to speed up matrix multiplication by the FPGA and improve the execution performance of SyntaxNet to 700 words/s.

IV. Proposal

OpenCL [7] is chosen for implementing logics on the FPGA and for activating FPGA from the host computer. OpenCL is a framework for implementing multithreading, and its specification is managed by Khronos Group Inc. Intel uses OpenCL as a framework for implementing FPGAs [8]. OpenCL was chosen for the reason explained below.

The development costs for offloading to the FPGA are shared between design cost for the logic circuit and system for starting up the FPGA, and these two costs must be minimized. When OpenCL is used, the logic-circuit design can be created in a shorter period of time than the Hardware Description Language (HDL) design by compiling a C program to be run on the FPGA. The system design has been implemented so it does not have any costs. From the above consideration, it is considered that the development costs can be reduced by using OpenCL on FPGA.

Then we consider how to call the FPGA from host computer. It would be efficient to call the FPGA from Eigen. But Eigen is programmed as allowing multi-threaded operation. If the FPGA is called form Eigen, multiple calling would happen to one FPGA. Therefore, it is good place to call where the point calling Eigen routine currently.

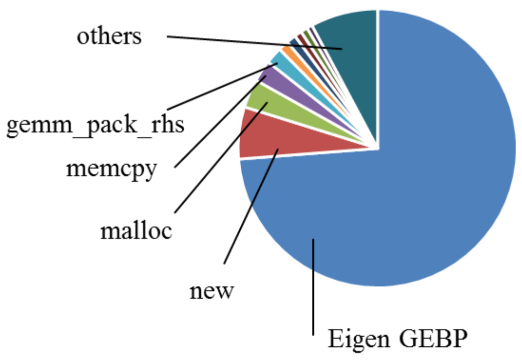

Figure 2. SyntaxNet Execution time analysis

```
for (int i=0; i < A_height; i++ ) do
    for (int j=0; j < B_width; j++ ) do
        float sum = 0.0 ;
        for (int k=0; k < A_width; k++ ) do
            sum += A[i][k] * B[k][j] ;
        end
        C[i][j] = sum ;
    end
end
```

Figure 3. Matrix multiplication algorithm

Another consideration, since data cannot be aligned in the program, Direct Memory Access (DMA) transfer to the FPGA is impossible with non-aligned data. A separate DMA buffer is added for DMA, and a memory-copy routine is added to copy data to the DMA buffer.

To minimize FPGA development cost, matrix-multiplication open-source sample code written in OpenCL was chosen for FPGA logic program, and the code was modified for turning the code performance. The routine for matrix multiplication is overviewed in Figure 3. Matrix multiplication is described by a triple loop [9]. The outer double loop specifies the position where the result, and the innermost loop calculates the inner-product of each data. In the OpenCL sample code, parallel processing of the innermost loop and the other row are performed. The high-speed internal memory size on FPGA is limited by FPGA chip size, then all matrix-data are placed on low-speed external DRAM, and partial data are copied to internal memory before performing partial matrix multiplication. Performances of the matrix multiplication on FPGA has different values depend on row and column size. Figures 4 and 5 show the performance of maximum and minimum data sizes used by SyntaxNet. According to Figure 4, performance of the FPGA did not change for any submatrix shape at minimum data size. Increasing the degree of parallelism increases the number of invalid area of matrix multiplication, but it does not improve matrix multiplication performance. On the other hand, according to Figure 5, the performance of the FPGA is almost constant at maximum data size even if the size of the column changes, but the performance increases in proportion to the size of row. Since the column side size is parallelized as much as possible when OpenCL is executed, the circuit configuration can only be slightly changed, and these circuits have the same performance. Even so, since row size is a parameter expressing how many elements are calculated in parallel, it is thought that doubling the number of submatrix rows doubles computation performance. Submatrix size above 64×64 could not be configured due to lack of FPGA resources, so 64×64 was considered to be the maximum. If execution time for each matrix size is focused on, it is clear that minimum matrix size takes about 6 ms, and maximum size takes about 150 ms. For that reason, maximum matrix size of 64×64 was taken as the parameter of the matrix-multiplication kernel.

TABLE II MEASUREMENT CONDITIONS

Item		Condition
Host	CPU	Xeon E5-2630 2.4 GHz
	Memory	32 GBytes
	HDD	1TB
FPGA board		Nallatech 385A
	FPGA	Intel Arria 10 GX1150
	DRAM	DDR-3 8GB
	interface	PCIe Gen3 x8
Software	OS	CentOS version 7.4
	OpenCL	version 17.0
	Quartus	version 17.0
	GCC	version 4.8.5

V. EVALUATION

Using the study up to the previous section, On the basis of the results presented in Figures 4 and 5, a matrix multiplication implemented on OpenCL on FPGA, and the performance of SyntaxNet of the implementation was evaluated and verified. The measurement conditions are listed in TABLE II.

A Nallatech 385A board uses the FPGA to improve performance. First, total execution time of SyntaxNet using FPGA matrix multiplication was measured three times and the average was taken.

The word throughput (which is taken as the performance

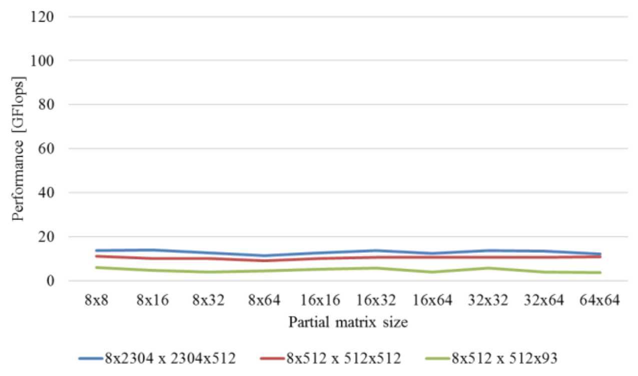

Figure 4. FPGA matrix multiplication performance (minimum)

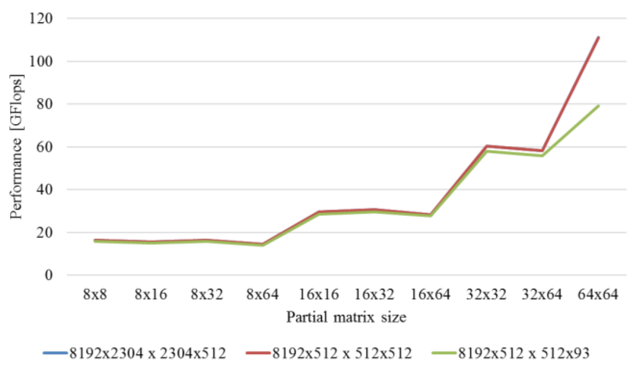

Figure 5. FPGA matrix multiplication performance (minimum)

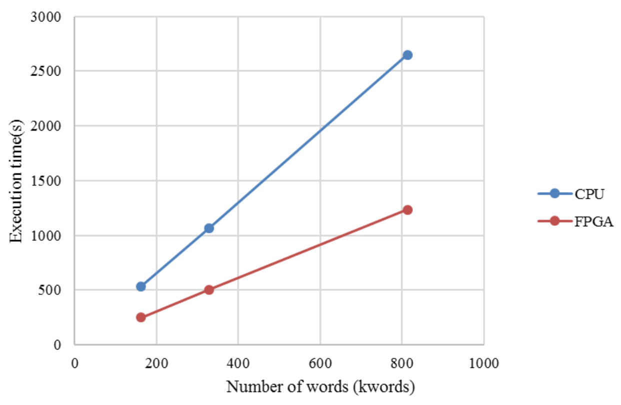

Figure 6. SyntaxNet performance (execution time)

measure of SyntaxNet) is shown in Figure 6. For comparison, the performance achieved with one Central Processing Unit (CPU) thread is also shown. In terms of word throughput, processing time is decreased. The CPU processed 307 words/s, FPGA offload processed 661 words/s, and the performance ratio of above two was 2.15 times. As a result, 661 words/s was achieved which was almost the target performance 700 words/s, and then 700 words were processed in 1.05 seconds. As a result, the FPGA achieved 661 words/s, compared to the target of 700 words/s, and it could process 700-word article in 1.05 seconds.

A breakdown of the execution time of SyntaxNet based on the above-described measurements and analysis, and the performance ratio of CPU and FPGA offloading is given in Figure 7. In Figures 7(a) to (d), processing performances of CPU 1 thread and FPGA offloading are compared, and the performance ratio is shown. Peak performance of matrix multiplication measured by FPGA alone and performance of matrix multiplication by Eigen processing routine of 1 CPU is shown in Figure 7(a). The performance ratio is 8.53 times. SyntaxNet uses various matrix sizes for actual matrix multiplication. So SyntaxNet effective performance would be lower performance than the peak performance.

The ratio of effective performance due to the matrix size decreases by 6.56 times compared to that of Figure 7(b). It is necessary to process various sizes of large and small size, and processing of small matrix size degrades performance in

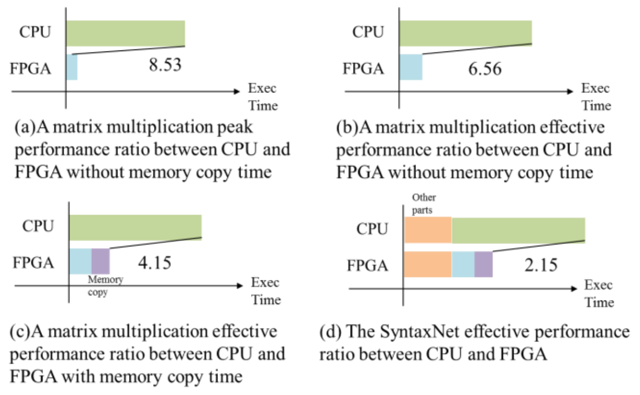
CPU
FPGA
8.53
Exec Time
(a)A matrix multiplication peak performance ratio between CPU and FPGA without memory copy time
CPU
FPGA
6.56
Exec Time
(b)A matrix multiplication effective performance ratio between CPU and FPGA without memory copy time
CPU
FPGA
4.15
Memory copy
Exec Time
(c)A matrix multiplication effective performance ratio between CPU and FPGA with memory copy time
Other parts
CPU
FPGA
2.15
Exec Time
(d) The SyntaxNet effective performance ratio between CPU and FPGA

Figure 7. SyntaxNet execution time breakdown

FPGA processing. It is necessary to process large and small matrix size, and processing of a small-size matrix degrades processing performance of the FPGA. As a result, effective performance is considered to decrease as a whole. Furthermore, memory processing is generated for FPGA processing. Therefore, when memory processing overhead is added, the performance ratio drops by 4.15 times compared to that shown in Figure 7(c). Then, in consideration of this result, the performance ratio becomes 2.15 times as shown in Figure 7(d) by adding other processing time of SntaxNet. Then offloading by FPGA cannot be achieved due to such overheads. It is difficult to measure these overhead previously. In consideration of the host-side software conditions and offload device characteristics, it will be necessary to make predictions.

VI. Discussion

Performance improvements are considered in the future. One of the methods for speeding up FPGA offloading is simultaneously executing DMA transfer and matrix multiplication by FPGA. However, as for SyntaxNet, the second-layer neural-network matrix multiplication is based on the result of matrix multiplication of the first-layer neural network. Therefore, the second-layer DMA transfer cannot be started until the first layer result is obtained. The second and the third layers are the same. Furthermore, a transition-based calculation is performed after the matrix multiplication of the third layer, and the transition-based calculation result is used to next neural network calculation. Therefore, in a loop that handles sentences, all matrix multiplications depending on the result need to be executed serially. Therefore, two SyntaxNet calculations of sentences need to be performed in parallel to hide the transfer time with the matrix multiplication time in FPGA. These calculation dependencies are eliminated by inserting irrelevant processing. As a result, the DMA transfer time can be hidden.

The other method for improving the performance of SyntaxNet is memory-copy reduction. In the current implementation of FPGA offload, the host cannot be used for DMA transfer from the array data area of the structure prepared by tensorflow because of data alignment. It is necessary to copy data to an area that aligned to 64 bytes. To eliminate this copy for speedup SyntaxNet, the memory area for the array data of the structure prepared by tensorflow must be aligned to 64 bytes. It is due to the specification of the PCIe bus of DMA transferring to the FPGA.

However, the FPGA matrix operation code divides matrix to submatrix. Then the code can calculate only the matrix size which is divisible by submatrix, but the code requires that a matrix of a certain size can be processed. When using the 64 × 64 submatrix, and matrix column size is not divisible by 64, for example, the column is 8, the remaining 56 parts should be 0 stuffed. It is necessary to make the matrix multiplication an accurate answer by the 0 stuffing process to manage. Since the 0 stuffing process perform memory copy, it can perform to change memory alignment to 64 bytes. In this study, the percentage of 0 stuffing that did not require 0-bit stuffing was estimated to be less than 5%. In the case of 95%, it needs memory copy because of 0 stuffing, then only 5% of memory copy would be eliminated by tensorflow modification. Then, it was found that this method had little prospect of performance improvement. It is thus concluded that the memory copy elimination method cannot improve the performance of SyntaxNet.

Another method to improve performance of SyntaxNet is implementing kernel code as a systolic array, which is a structure of logic circuit that repeats operations such as multiplication and addition while moving data. A systolic array is known as an efficient multiply-add operation method when there are many combinations of operations. The systolic array is improved calculation efficiency.

Especially in recent years, Google applied systolic array to matrix-calculation circuits for deep learning such as TPU [14]. Circuit logic diagrams of a matrix multiplier using a self-designed systolic array are shown in Figures 8 and 9.

Each arithmetic unit in Figure 8 is a simple one consisting of a multiplier and an adder. The arithmetic unit multiplies a value (A) sending from the left, and another value for multiplication (B) is held by the register. After that, the arithmetic unit adds the sent value (S). Each value coming from the left is sent to the right (An), and the value after addition is sent to the bottom (Sn). This arithmetic unit is arranged in two dimensions as shown in Figure 9. Its operation consists of four steps as follows.

(1) Set all the values of matrix B.

(2) Send the values in the order from the left to the right. The transmission one step down is started one cycle later, and the pattern is transmitted diagonally in space.

(3) Send A for the operation and wait for the result to appear in the lower buffer.

(4) Repeat (1) to (3) with the next data.

Data movement and calculation are performed at the same time by such operation, and multiply-add operation is efficient. We attempted to create this systolic array using OpenCL. The proposed systolic array circuit was created on the basis of OpenCL. In particular, a systolic-array code was written with OpenCL as the hardware shown in Figure 9, and the code looked like working. Then, the circuit ran on FPGA, but its performance was three digits lower than we expect.

It would happen because the arithmetic unit has a lot of latency because of its floating-point multiplication, then OencCL compiler create a lot of processing latency in the array. The latency cannot be changed by changing the OpenCL code only. The OpenCL is designed to make hardware from an algorithm written in C. However, it is difficult to describe hardware itself like the systolic array.

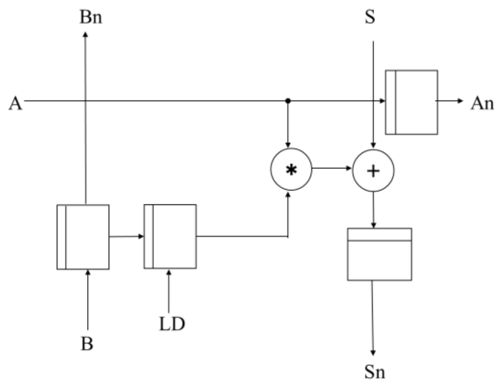

Figure 8. Systolic array arithmetic unit

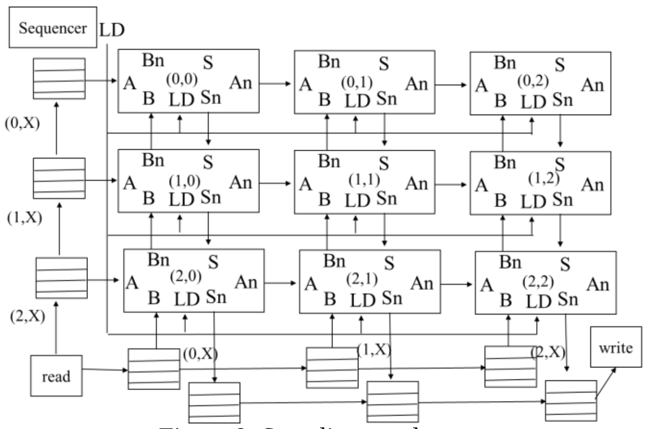

Figure 9. Systolic array layout

VII. Conclusion

To automatically classify and analyze customer complaints, we investigated whether it is possible to speed up the OSS syntax analyzer SyntaxNet with an FPGA, implemented FPGA offload, evaluated an actual machine, and obtained the following conclusions. We evaluated SyntaxNet with FPGA offload, and confirmed that SyntaxNet's execution performance was 661 words/s, almost achieved the target of 700 words/s, and processed sentences of 700 words in general size in about 1 second. The execution time of an FPGA offload machine was measured, and the measurement results confirmed that (i) the execution performance of SyntaxNet was 661 words/s (which almost achieved the target of 700 words/s) and (ii) sentences with size of 700 words (in general) could be processed in about 1 second. These results demonstrate that automatic text categorization and analysis can be immediately executed on a system with a reduced number of servers by speeding it up with power-saving FPGA acceleration. This FPGA offloading can be applied to all neural networks using matrix multiplication.

Acknowledgment

The author thanks Dr. Tsuyoshi Tanaka for his assistance in writing this paper.

References

[1] D. Lin, Introduction to Natural Language Processing (NLP) [0nline]. Available form: https://www.slideserve.com/jory/introduction-to-natural-language-processing-nlp 2014.3.12

[2] Akamai Technologies, Inc. New Study Reveals Impact of Travel Site Performance. [Online]. Available from:

https://www.akamai.com/us/en/about/news/press/2010-press/new-study-reveals-the-impact-of-travel-site-performance-on-consumers.jsp 2010.06.14

[3] Forbes. Do You Read Fast Enough To Be Successful?. [Online]. Available from: https://www.forbes.com/sites/brettnelson/2012/06/04/do-you-read-fast-enough-to-be-successful/#3b3dd025462e 2012.06.04

[4] D. Andor, C. Alberti, D. Weiss, and A. Severyn, "Globally Normalized Transition-Based Neural Networks," arXiv 1603.06042v2, March 2016.

[5] J. Chang, J. Seefried, S. Taylor, and A. Brandner, "SyntaxNet: Google's Open-sourced Syntactic Parser," Department of Computational Linguistics University of Tubingen, January 2018.

[6] Eigen is a C++ template library for linear algebra. [Online]. Available from: http://eigen.tuxfamily.org/index.php?title=Main_Page 2019.08

[7] Khronos Group. OpenCL Overview. [Online]. Available form: https://www.khronos.org/opencl/ 2019

[8] Intel, Inc. Intel FPGA SDK for OpenCL Software Technology. [Online]. Available from: https://www.intel.com/content/www/us/en/software/programmable/sdk-for-opencl/overview.html 2019

[9] Z. Wang, B. He, W. Zhang, and S. Jiang, "A Performance Analysis Framework for Optimizing OpenCL Applications on FPGAs" IEEE, 2016.

[10] T. Kudo and Y. Matsumoto, "Japanese Dependency Analysis Using Cascaded Chunking," , in Japanese, June 2002.

[11] S. Kurohashi and M. Nagao, "KN Parser : Japanese Dependency / Case Structure Analyzer," unknown, 1994

[12] S. Kurohashi and M. Nagao, "Building a Japanese Parsed Corpus while Improving the Parsing System," unknown, 1998

[13] Stanford University. Stanford CoreNLP – Natural language software. [Online]. Available from: https://stanfordnlp.github.io/CoreNLP/ 2019

[14] N. P. Jouppi, C. Young, N. Patil, D. Patterson, et.al "In-Datacenter Performance Analysis of a Tensor Processing Unit", ACM/IEEE 44th, 2017

Monitor for Safety-Critical Mirror Drivers of MEMS Micro-Scanning LiDAR Systems

Philipp Stelzer, Andreas Strasser, Philip Pannagger, Christian Steger
Graz University of Technology
Graz, Austria
Email: {stelzer, strasser, steger}@tugraz.at
pannagger@student.tugraz.at

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

***Abstract*—In future, more and more cars will be equipped with Advanced Driver-Assistance Systems (ADAS) like Adaptive Cruise Control (ACC), Collision Avoidance System and many more. Currently, the driver is responsible by law to perceive the environment and take over control if it is required. But in foreseeable future highly automated vehicles or even fully automated vehicles will appear on the road; where the vehicle is responsible for perceiving the environment, operating the vehicle and intervening in hazardous situations. At the latest then it will be necessary that systems shall not fail unnoticed. Therefore, it is mandatory to monitor safety relevant components. For instance, also Light Detection and Ranging (LiDAR) Systems like the 1D Micro-Electro-Mechanical System (MEMS) Micro-Scanning LiDAR, which will be part of intelligent sensor fusion in ADAS. Due to the fact that highly automated vehicles often have various safety monitors installed, our novel Monitor for the Safety-Critical MEMS Driver is an alternative approach to the well-known Built-In Self-Test (BIST). In this publication, we introduce a novel system architecture that is able to verify the correct functionality of internal control systems in MEMS-based LiDAR systems. To evaluate the effectiveness of our novel monitoring approach, we have implemented the procedure on a 1D MEMS Micro-Scanning LiDAR prototype platform.**

Keywords*–*ADAS; LiDAR; Signal Monitor; 1D MEMS Mirror; Safety Monitor

I. Introduction

Fully automated driving is gaining more and more attention. Therefore, industry and academia put a lot of effort into researching in the field of sensor fusion and functional safety for sensors in the automotive domain. Key enablers of highly automated vehicles will be robust Radio Detection and Ranging (RADAR) and Light Detection and Ranging (LiDAR) solutions with additional support from vision cameras. By fusion of sensor data and control functions it is intended to enable safe automated driving as well in rural as also in urban environments. In the project PRogrammable sYSTems for INtelligence in automobilEs (PRYSTINE) the consortium aims for a Fail-operational Urban Surround perceptION (FUSION) [1]. For years, various Advanced Driver-Assistance Systems (ADAS), such as Electronic Stability Control (ESC) and Anti-lock Braking System (ABS), have been mandatory in new cars in the European Union [2]. ESC and ABS are ADAS, which are active safety components in contrast to passive safety components, such as seat belts and airbags [3]. For highly automated vehicles, it is indispensable that ADAS

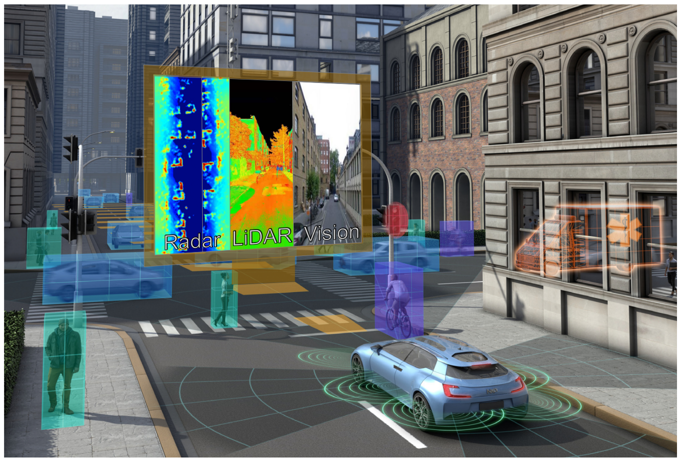

Figure 1. PRYSTINE's concept view of a Fail-operational Urban Surround perceptION (FUSION) [1].

are reliable and therefore to ensure the safety for the driver, passengers and all other road users. Due to quantity and reliability of such ADAS and integrated systems the Society of Automotive Engineers (SAE) has introduced six levels of driving automation. The higher the SAE level, the higher ranked is the driving automation of the vehicle. Due to the competences that the systems take over in the vehicle, it is possible to declare the SAE level of the vehicle [4]. No matter whether a vehicle, according to the manufacturer, would support higher automation levels, it is currently necessary in many countries that the driver continues to observe the environment and in an emergency can take over control [5]. For example, according to Article 8 of the Vienna Convention on Road Traffic, the driver must be able to control the vehicle continuously. The Vienna Convention on Road Traffic was ratified by the majority of EU member countries and several others. Large countries, such as the USA, China or England, are not among the signatories [6]. Due to legal and technical barriers, driving automation levels of vehicles are currently not beyond SAE level 2. In order to introduce vehicles with SAE Level 3 and higher in the future, it is imperative to adapt the law from a legal point of view and to develop ADAS with a higher level of safety, reliability and availability. In projects like PRYSTINE, it is the goal to develop components and systems for high reliable and safe ADAS [1]. To ensure the proper functionality of systems it is mandatory to monitor the system especially safety critical parts of it. In case of a malfunction the system has to be degraded and in worst case suspended. Hence, these safety monitors are essential for ADAS in vehicles of SAE level 3 and above. Misbehaviour of a system is only detectable if the system is monitored. Therefore, we have been engaged in monitoring the critical signals of a 1D MEMS Micro-Scanning LiDAR System.

With our paper contribution we:

- create a novel test opportunity for control loops,
- ensure the detection of malfunction during test run and
- enhance safety due to this diverse monitoring approach.

The remainder of the paper is structured as follows. The overview on related work of MEMS-based LiDAR systems is given in Section II. The architecture of a novel safety monitor for safety-critical signals in a MEMS-based LiDAR System will be presented in detail in Section III and the achieved results including a short discussion will be provided in Section IV. A summary and short discussion of the findings will conclude this paper in Section V.

II. Related Work

LiDAR technologies, which are currently available in the market are very bulky and cost intensive like the Velodyne HDL-64E [7]. Therefore, industry and academia put a lot of effort into the research of automotive qualified, long-range and low-cost LiDARs. Druml et al. have introduced a 1D MEMS Micro-Scanning LiDAR, which is able to perceive the environment up to 200m, shall cost less than 200$ and is qualified for automotive applications due to its robustness [8]. The functional principle of the 1D MEMS-based LiDAR by Druml et al. is depicted in Figure 3. Several lasers are shot on the 1D MEMS mirror. A vertical laser beam is deflected by the mirror into the scenery. This vertical line is moved horizontally across the Field-of-View(FoV) by oscillation of the mirror and the reflected light of the obstacle is captured by a stationary detector.

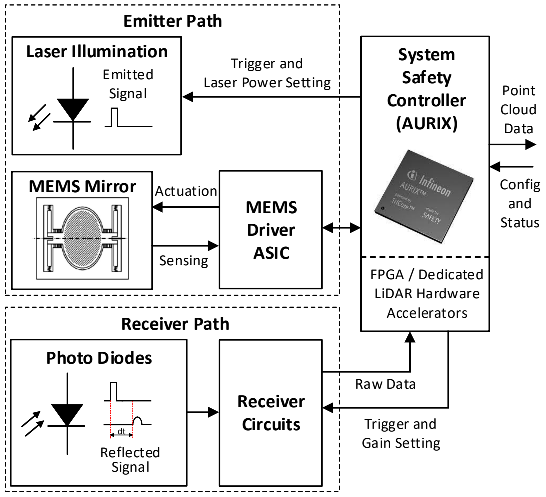

Figure 2. System concept of a 1D MEMS-based automotive LiDAR system by Druml et al. [8].

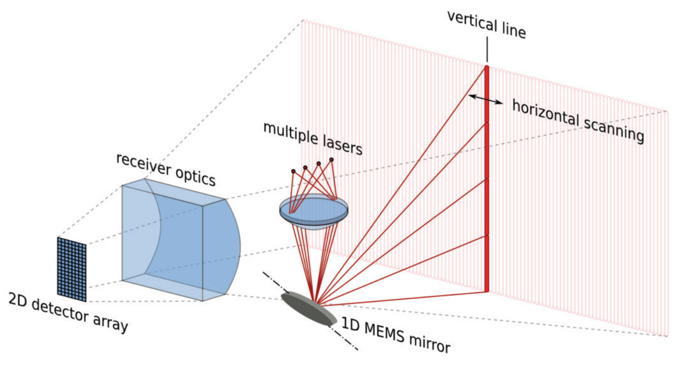

Figure 3. Functional principle of a 1D micro-scanning LiDAR [8].

A. 1D MEMS Micro-Scanning LiDAR

In this section, the 1D MEMS-based LiDAR System by Druml et al. is presented. The system concept of the MEMS-based LiDAR is depicted in Figure 2. Druml et al.'s system consists in general of an emitter path, a receiver path and the System Safety Controller (AURIX). In the emitter path are included a laser illumination unit, the MEMS mirror and the actuation and sensing unit of the mirror, the MEMS Driver ASIC. Within the receiver path are an array of photo diodes and the receiver circuits. The System Safety Controller is the central unit, which is responsible ,e.g., for monitoring, controlling and signal processing. According to the signal processing part, the task of the System Safety Controller is to compute and provide a 3D point cloud for dedicated ADAS [8]. Due to the dependence of correct position, direction and verification signals of the mirror, the Driver ASIC, which is responsible for the actuation and sensing of the MEMS mirror, is described in particular. The MEMS Driver is providing crucial signals to the System Safety Controller and therefore it is mandatory that the delivered information is reliable. By reference to the correctness of these crucial signals, the System Safety Controller will create with the raw data from the receiver circuits a plausible 3D point cloud. If the crucial signals were corrupted the 3D point cloud would be useless due to wrong assumptions of the reflected laser origin.

In Figure 4, the crucial signals are illustrated, which are provided by the MEMS Driver ASIC. These signals are needed to monitor during operation the current status of the MEMS mirror. The POSITION_L represents whether the mirror is

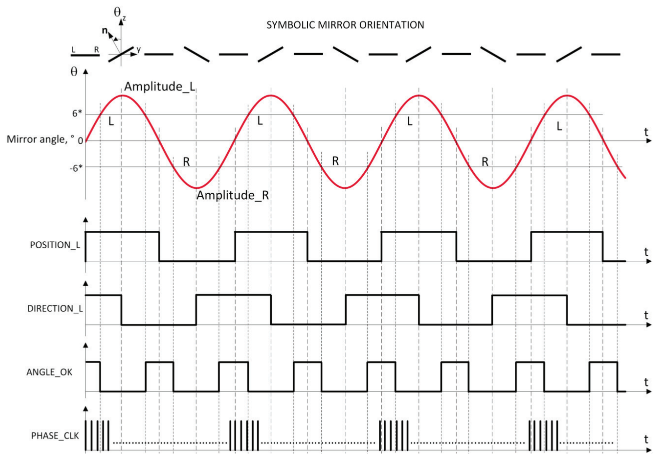

Figure 4. Crucial signals of the MEMS Driver ASIC from Druml et al.'s LiDAR system [8].

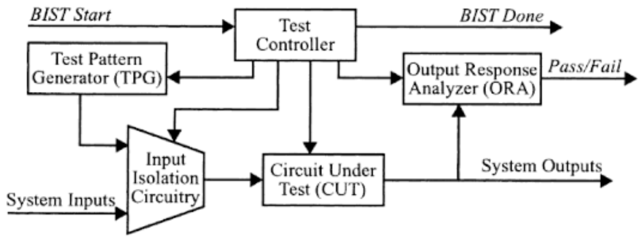

Figure 5. A basic Built-In Self-Test Architecture [9].

aligned to the left or to the right side; logical high means an alignment to the left and logical low to the right. DIRECTION_L indicates in which direction the movement is located; logical high means moving to left and logical low to the right. Precise and high-frequent phase information of the current mirror position is provided by a PHASE_CLK signal that counts from 0 to n_{max} in equal time steps during one mirror oscillation. Furthermore, an ANGLE_OK signal is available besides the tracking signals. This ANGLE_OK signal notifies to the System Safety Controller if the Driver ASIC operates according to the programmed specification (e.g., angle setpoint is reached). To be able to ensure functional-, eye-, and skin-safety this notification is mandatory: MEMS mirror's current position and MEMS Driver ASIC's internal position information must match to allow a laser shooting [8].

B. Test Facilities

One of the major objectives of the automobile industry is to evolve the individual traffic. The coexistence of partial automated, highly automated and fully automated cars will be the reality in the near future. In conventionally equipped vehicles, the driver is responsible for environment perception, operation of the vehicle and intervention in hazardous situations. In prospective automated cars more and more competences will move from the driver to the car. Based on information, which is obtained from ADAS, the vehicle will make decisions. Therefore, it is apparently necessary that this information is reliable. To ensure safe and reliable operation of ADAS and their embedded components like LiDAR, it is mandatory to test the behaviour for correctness. BISTs and a wide variety of safety monitors can be used for this purpose.

1) Built-In Self-Test:
A Built-In Self-Test (BIST) is thought to run simultaneously to the circuit and is monitoring or checking the output of a circuit to check its validity. The BIST needs a strategy for generating input signals for the circuit and has to know how to evaluate the correlated output. The circuit or device which is tested is called the Circuit Under Test (CUT). A basic BIST architecture is shown in Figure 5. A realization of a BIST fundamentally needs to implement four new functions to the existing system. First of all, there is the Test Pattern Generator (TPG), which is responsible for generating the input signals and for the test. The test pattern consists of multiple sets of test cases, which theoretically simulates all possible combinations of input signals. The complement to the TPG is the Output Response Analyzer (ORA). Its task is to know every correct output response of the CUT and decides whether the current output is faulty or valid. To create a meaningful and valid test it is important to isolate the test from any other input. Therefore, the Input Isolation Circuitry (IIC) is implemented. Its task is to decouple all input signals, which are commonly provided to the CUT and replace them with test-signal coming from the TPG. Last but not least, to synchronize the behaviour of the TPG, ORA and IIC the Test Controller is implemented. It first initializes a specific test then decouples the System Inputs and finally activates the ORA which then outputs a Fail or Passed signal [9][10].

2) Safety Monitor Approaches:
Besides BISTs, there are also other monitors, which verify the behaviour of circuits and overall systems. Schuldt et al. [11], for example, are strive to test and validate ADAS efficiently by reference to systematically generated virtual test scenarios. The idea hereby is to identify the factors, which are affecting the assistance system. Hence, the test scenarios will be generated. By reference to the test scenarios a test will be executed and due to a variety of scenarios a evaluation of the results can be done. Another approach to monitor ADAS is presented by Mauritz et al. [12]. With this approach results obtained from simulations are transferred to the road. They ensure a consistently behaviour of the ADAS in both worlds due to a simulation of realistic driving conditions and by utilization of a set of runtime monitors. Furthermore, Meany [13] elucidated that in all modern safety-critical systems the Integrated Circuits (IC) are the root. According to Meany, besides redundant and diverse development, it is necessary to monitor the ICs to be fault-tolerant. There are several ways to monitor the IC during operation. Meany addresses in his paper several opportunities of IC diagnostics.

III. Core Concept and Architecture

In this section, we present our concept and architecture for a novel safety monitor of MEMS-based LiDAR systems. The reliability of the Driver is a sensitive topic, therefore it is indispensable to monitor and test the Driver extensively and diverse. Due to that we have introduced this novel procedure to be able to test and monitor the Driver in a new way. At first, the architecture modifications are highlighted and described. Furthermore, we go through the process flow of the monitoring and test period. With this new monitor there is another possibility to detect faults in the Driver module at an early stage and to take appropriate measures. Due to the diversity of the testing module it should be possible to prevent undetected faults even better.

In Figure 6, the modified block diagram is illustrated. In principle, it is a common phase-locked loop (PLL), which is essential for the MEMS mirror actuation, the System Safety Controller, the MEMS mirror and our novel Safety-Critical Mirror Driver Monitor (SCMDM). The HV(On/Off) signal sets the points in time in the internal schedule at which the High Voltage (HV) is switched on or off. This internal schedule is administrated by the Mirror Subtiming block. How fast or slow this schedule is processed is dependable from the PLL and thus, we aimed to test the PLL on its functionality. Due to this we have designed a SCMDM and also adapted the existing architecture and integrate our novel monitor into it. The core of the SCMDM is consisting of a mirror simulation part and a decision part. The decision part is responsible to decide when the test run is conducted and for notification to the System Safety Controller. With the start of the test run and the accompanying monitoring of the system, it is also necessary to decouple the Driver from the physical MEMS mirror. Hence, there were switches for the

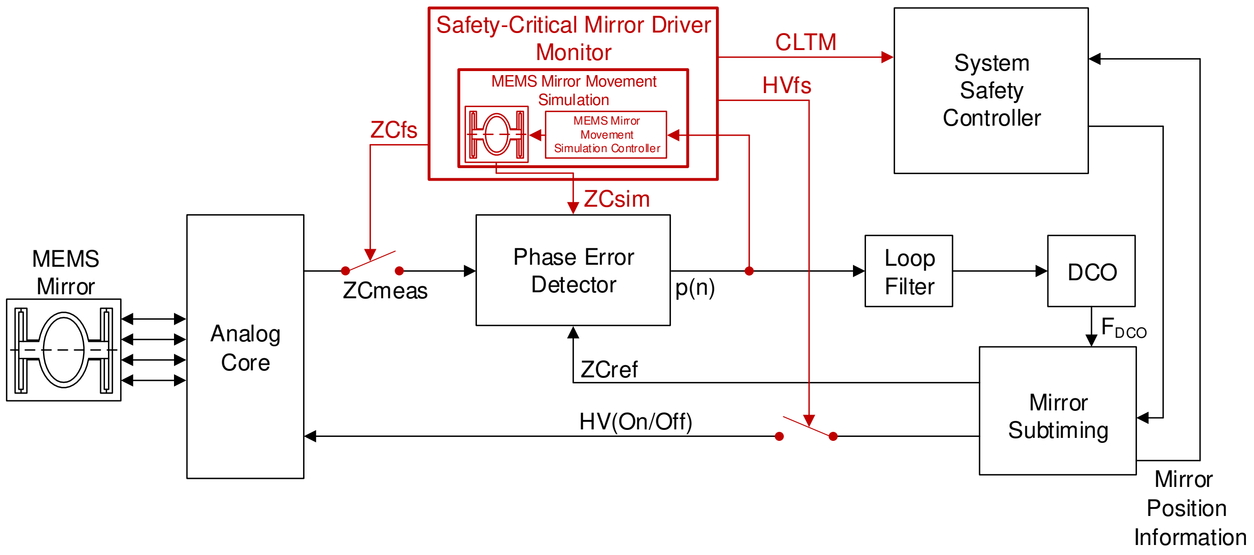

Figure 6. Block diagram of a PLL architecture with the novel adaptions to include a Safety-Critical Mirror Driver Monitor module in the system.

Zero-Crossing measured (ZCmeas) and High Voltage On/Off (HV(On/Off)) signals implemented. In case of test run started the SCMDM block disables the switch for ZCmeas signal by Zero-Crossing forwarding stop (ZCfs) signal and the switch for HV(On/Off) signal by High Voltage forwarding stop (HVfs) signal. Furthermore, the SCMDM notifies the System Safety Controller about a test run by the Control Loop Test Mode (CLTM) signal.

After a test run is started (can be started at at a vehicle start or when stopping in front of a traffic light) the Zero-Crossing simulated (ZCsim) signal is instead of the ZCmeas forwarded to the Phase Error Detector (PD) block. In case of a vehicle start, the frequency of the simulated MEMS mirror movement is set to a random but plausible frequency. Otherwise, the frequency is set to a different frequency than the actual mirror swing to test and monitor the behaviour of the MEMS Driver during control operation. To be able to adapt the simulated frequency of the Zero-Crossing (ZC) a MEMS Mirror Movement Simulation Controller (MMMSC) is implemented in the simulation part of the SCMDM. By reference to the PLL error this controller is adapting the simulated MEMS mirror frequency and works contrary to the PLL. Due to the characteristics of the MEMS mirror concerning acceleration and deceleration, the control loop of the simulation must take these into account. This is necessary to be able to emulate the physical MEMS mirror's behaviour after frequency increase respectively decrease. The acceleration of the mirror requires more energy effort than its deceleration. Thus, the integrator values have to be chosen accordingly to that fact. How the flow of this procedure looks alike is depicted in Figure 7. The test cycle and monitoring procedure is divided in the following steps:

1) **Checking for Driving Cycle**
In the background, it is continuously checked whether the vehicle is in the driving state or not. A stopped driving cycle is, for example, a vehicle stop before a traffic light or a vehicle start. A test cycle with subsequent mirror restart usually lasts much shorter than 1s. In both cases, traffic light stop and vehicle start, there is at least 1s time to perform the test and monitoring cycle. Hence, the SCMDM is started after a stop of the driving cycle is detected.

2) **Safety-Critical Mirror Driver Monitor Enable**
After the driving cycle check gives green light for the SCMDM the SCMDM is enabled and notifies the System Safety Controller via the CLTM signal about the test cycle. Next step is to adjust the frequency for the simulated mirror.

3) **Frequency Adjustment**
On the basis of a simulated mirror movement the adequate and orderly function of the MEMS Driver

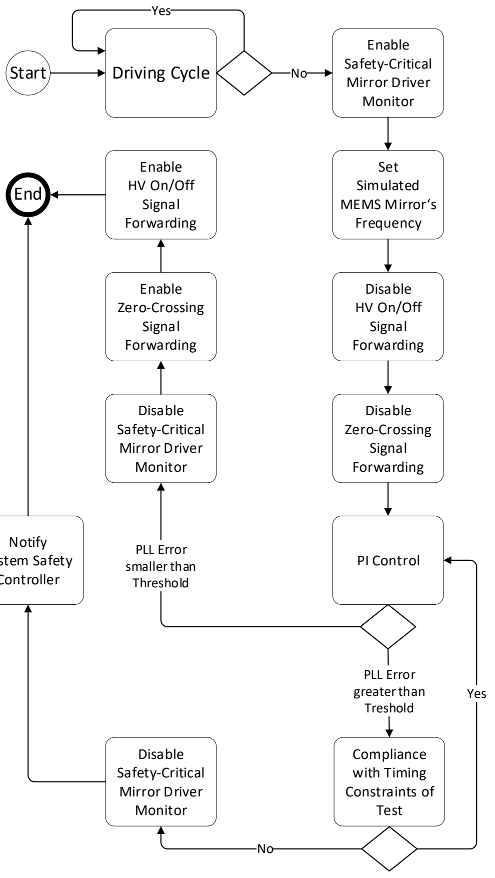
Yes
Start
Driving Cycle
No
Enable Safety-Critical Mirror Driver Monitor
Set Simulated MEMS Mirror's Frequency
Disable HV On/Off Signal Forwarding
Disable Zero-Crossing Signal Forwarding
PI Control
PLL Error greater than Treshold
Yes
Compliance with Timing Constraints of Test
No
Disable Safety-Critical Mirror Driver Monitor
Notify System Safety Controller
PLL Error smaller than Threshold
Disable Safety-Critical Mirror Driver Monitor
Enable Zero-Crossing Signal Forwarding
Enable HV On/Off Signal Forwarding
End

Figure 7. Process flow of the Safety-Critical Mirror Driver Monitor module.

ASIC's PLL shall be proved. Therefore, it is necessary to set a start frequency for this simulated mirror with a significant difference to the actual frequency of the physical MEMS mirror. In case of a vehicle start it is only necessary to choose a frequency within given limits of the physical MEMS mirror. If the MEMS mirror has already been in operation, the frequency to be set must then be selected within plausible limits and the selected frequency must also be sufficiently different from the actual mirror frequency. After the initial frequency of the mirror simulation is set the system has to be decoupled from the physical MEMS mirror during the test cycle.

4) **Decoupling**
Switches have been integrated into the existing architecture to decouple the system from the MEMS mirror. By means of HVfs the HV(On/Off) signal is decoupled from the physical mirror and thus prevents an unwanted mirror actuation. During the test phase, the mirror is actuated in a open loop mode with the HV(On/Off) value, which is configured before the test is started. In order to prevent a disturbance of the control loop in the test mode by the ZC of the physical mirror, the ZCmeas signal is switched off. Thus, only the ZCsim signal is forwarded to the PD block and the PLL is not affected of two different, actual and simulated ZC, signals.

5) **PI Control**
Then the control of the PLL and the simulated mirror frequency begins. The PLL is operating as usual and tries to match the internal adjusted frequency with the simulated mirror frequency. The simulated mirror is also adapting the frequency with respect to the specifics of the acceleration and deceleration of the physical mirror. By reference to the obtained PLL error the MEMS Mirror Movement Simulation (MMMS) part is informed whether an acceleration (frequency increase) or a deceleration (frequency decrease) has to be simulated. It is necessary to know whether the simulated mirror needs to be accelerated or decelerated because the integrator values of acceleration and deceleration are different. This is the case because there is a difference in energy consumption between acceleration and deceleration. This control happens until either the simulated mirror has the desired frequency or a time limit is reached.

6) **End of PI Control**

   a) **Control Success**
   After the control process was successful, the SCMDM is disabled and the physical MEMS mirror is integrated into the control system again instead of the simulated one. To re-integrate the MEMS mirror, the ZCmeas signal is forwarded to the PD block and the HV(On/Off) signal of the Mirror Subtiming block is forwarded to the Analog Core that connects to the physical mirror.

   b) **Control Abort**
   In the case that the control is aborted by reaching the time limit, the SCMDM is also disabled. In contrast to successful control, however, a notification of failure is transmitted to the System Safety Controller. The System Safety Controller is then responsible for what measures are taken. Such measures could possibly be a further test run or a degradation of the system.

7) **Encoupling**
After the test run is finished, the physical mirror is coupled back into the system. This works in principle similar to the start-up procedure. The physical mirror in open loop mode is put back into closed loop mode by activating the PLL. This completes the test run and the system continues to operate as before.

With this novel procedure there is another possibility to check the function of a control loop for MEMS-based LiDAR systems. Especially for safety-critical components in environmental perception systems, it is important that there is not only redundancy of tests and monitors but also diversity. The most important thing is to ensure the correct functioning of the systems that provide information for ADAS and other fusion components. The following section discusses and explains the results of the novel monitor approach.

IV. Results

In this section, we provide the measurement results of our novel monitoring procedure, which has been introduced in Section III.

Figure 8 shows the start of the novel monitor procedure. After 427 mirror half periods, the frequency of the simulated mirror is changed. The Angle_Ok signal can be used as an indicator for a frequency shift between mirror and driver because it indicates whether the angle setpoint is reached or not. At the beginning of the frequency mismatch this is also clearly visible in the ZC measurement. The red signal corresponds to the ZC reference signal of the MEMS mirror Driver and the blue one to the ZCsim signal. After the 427 mirror half period it is clearly visible that the reference and the simulated ZC signal are no longer synchronous. The exemplary course of the mirror is recorded at Mirror Angle. The red curve indicates the course of the mirror at the same frequency and the blue curve looks like the course when the new frequency is set for

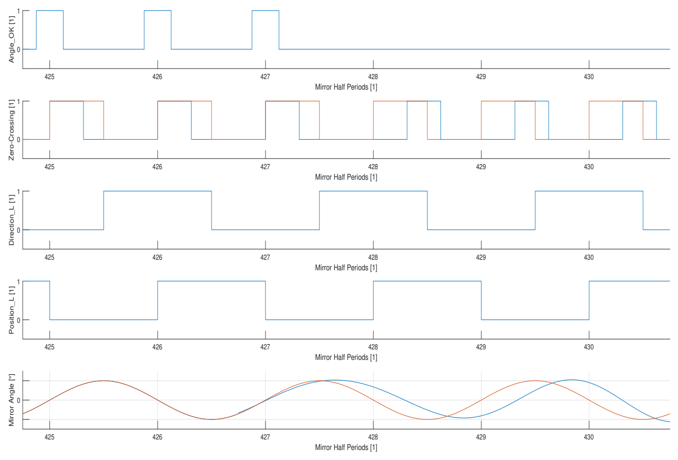

Figure 8. Measurement with the initial frequency adaption of the simulated MEMS mirror.

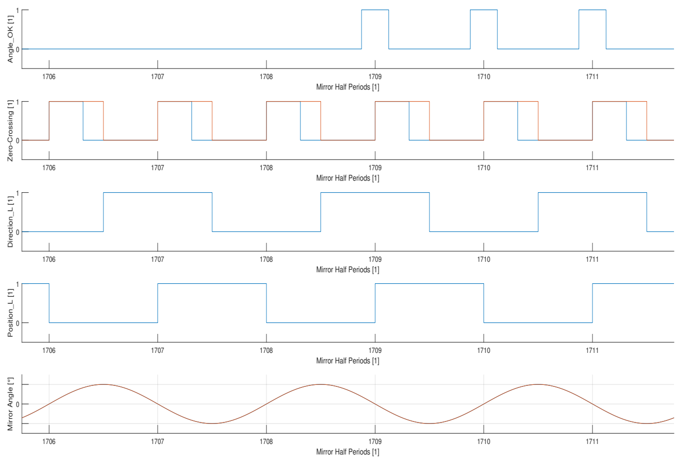

Figure 9. Measurement with the frequency match of the simulated MEMS mirror and the MEMS Driver.

the simulated mirror. Figure 9 shows that the frequency of the mirror has been adjusted again and that the angle setpoint has been reached again from the 1709 mirror half period onwards. Here the Angle_Ok signal is essential for detecting whether the angle setpoint has already been reached again. It looks as if the frequencies of mirror and Driver are equalized before the 1709th mirror half period. The exemplary courses of the mirror overlap almost completely and reference and simulated ZC signal also occur again almost simultaneously. For our measurement, the control required 1282 mirror half periods to adjust the frequencies. That was about 220ms at this frequency. Depending on the frequency difference between mirror and Driver, this control time can be extended or shortened. Finally, the results of the frequency adaption duration are summarised and shown in Table I.

TABLE I. MEASUREMENT RESULTS

	Begin	End	Time in ms
Duration of Frequency Adaption	427	1709	$\sim$ 220

V. Conclusion

In our paper, we have introduced a novel architecture for a Safety-Critical Mirror Driver Monitor. With this monitor a new possibility is created to test the control of a MEMS-based LiDAR system and to monitor the functionality of the Driver during the test cycle. The diversity of system monitor options is further increased with this new SCMDM, along with BIST and other diagnostic variants, further reducing the likelihood of malfunctions remaining undetected. With a duration of around 220ms, this test run is also well under 1s. So it is no problem to perform this procedure while the start of the vehicle or a vehicle stop in front of a traffic light. Even if the traffic starts to move again, not even 1s passes until the LiDAR system is operational again. Due to the speed at which the vehicle starts to move (usually a slow start), it is only a few centimetres at the most that the vehicle does not receive any information from the LiDAR. By further optimizing the parameters, the time required for the test run can probably be shortened considerably. Our intention was to show that in principle it is possible to simulate the mirror and thus create a further possibility for MEMS Driver monitoring by means of the novel monitor. Monitors such as these will be even more important in the future for highly automated vehicles than they already are in safety-critical vehicle components. The top priority is to ensure the safety and reliability of the ADAS in the vehicles and also to check whether this is the case.

Acknowledgment

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] 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.

[2] European Road Safety Observatory, “Advanced driver assistance systems,” https://ec.europa.eu/transport/road_safety/specialist/observatory/analyses/traffic_safety_syntheses/safety_synthesis_en, retrieved: October, 2019. [Online]. Available: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/ersosynthesis2018-adas.pdf

[3] M. Lu, K. Wevers, and R. V. D. Heijden, “Technical Feasibility of Advanced Driver Assistance Systems (ADAS) for Road Traffic Safety,” Transportation Planning and Technology, vol. 28, no. 3, 2005, pp. 167–187. [Online]. Available: https://doi.org/10.1080/03081060500120282

[4] SAE, “SAE International Standard J3016 - Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems,” SAE International, Standard, January 2014.

[5] C. Brünglinghaus, “Wie das Recht automatisiertes Fahren hemmt,” ATZ - Automobiltechnische Zeitschrift, vol. 117, no. 4, Apr 2015, pp. 8–13. [Online]. Available: https://doi.org/10.1007/s35148-015-0039-0

[6] United Nations Conference on Road Traffic, “19 . Convention on Road Traffic,” https://treaties.un.org/pages/ViewDetailsIII.aspx?src=TREATY&mtdsg_no=XI-B-19&chapter=11&Temp=mtdsg3&clang=_en, retrieved: October, 2019. [Online]. Available: https://treaties.un.org/pages/ViewDetailsIII.aspx?src=TREATY&mtdsg_no=XI-B-19&chapter=11&Temp=mtdsg3&clang=_en

[7] Velodyne LiDAR, “HDL-64E,” 2016.

[8] N. Druml, I. Maksymova, T. Thurner, D. Van Lierop, M. Hennecke, and A. Foroutan, “1D MEMS Micro-Scanning LiDAR,” in The Ninth International Conference on Sensor Device Technologies and Applications (SENSORDEVICES 2018), 09 2018.

[9] C. E. Stroud, A designers guide to built-in self-test. Springer Science & Business Media, 2006, vol. 19.

[10] E. J. McCluskey, “Built-In Self-Test Techniques,” IEEE Design Test of Computers, vol. 2, no. 2, April 1985, pp. 21–28.

[11] F. Schuldt, F. Saust, B. Lichte, M. Maurer, and S. Scholz, “Effiziente systematische testgenerierung für fahrerassistenzsysteme in virtuellen umgebungen,” 2013, retrieved: October, 2019. [Online]. Available: https://publikationsserver.tu-braunschweig.de/receive/dbbs_mods_00052570

[12] M. Mauritz, F. Howar, and A. Rausch, “Assuring the Safety of Advanced Driver Assistance Systems Through a Combination of Simulation and Runtime Monitoring,” in Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2016, pp. 672–687.

[13] T. Meany, “Functional Safety for Integrated Circuits,” July 2018, retrieved: November, 2019. [Online]. Available: https://www.analog.com/en/technical-articles/a54121-functional-safety-for-integrated-circuits.html

Challenges in Mitigating Soft Errors in Safety-critical Systems with COTS Microprocessors

Amer Kajmakovic*, Konrad Diwold*, Nermin Kajtazovic¶, Robert Zupanc¶

*Pro2Future GmbH & Institute of Technical Informatics, TU-Graz, Graz, AT
E-mail: (amer.kajmakovic, konrad.diwold)@pro2future.com
¶ Siemens AG, Graz, AT, E-mail: (nermin.kajtazovic, robert.zupanc)@siemens.com

***Abstract*—The number of Commercial-Off-The-Shelf (COTS) microprocessors and microcontrollers used in safety applications increased significantly over the last decade. In contrast to safety-certified microcontrollers, these microcontrollers are produced without integrated protection against memory soft errors, and limited in terms of available memory and computation power. However, due to the constant optimizations of the memory's physical size and the voltage margins, the probability that external factors, such as magnetic fields or cosmic rays, temporally alter a memory state (and thus cause a soft error) rises. Especially within safety-critical automation systems, it is crucial to address such errors and a wide range of error mitigation strategies have been proposed. In the context of established brownfield automation systems, the redesign and deployment of new hardware is usually not feasible. Therefore software-based strategies are required, which can be deployed on existing fail-safe architectures to further improve their performances, without requiring their rework or conceptual changes. This article identifies challenges associated with software-based soft error detection and correction strategies. Along with the challenges, a short overview of currently applicable software-based mitigation strategies is given and the strategies are evaluated.**

Keywords–soft errors; mixed-criticality; fail-safe; 1oo2D; embedded memory; hamming; parity bit; redundant parity.

I. Introduction

Given their ever-decreasing packaging size, semiconductors are becoming more susceptible to external influences such as alpha particles, cosmic rays or magnetic fields [1]. Figure 1 shows the correlation of error rates in semiconductors and technology/fabrication nodes (nm) size. It is noticeable, that the Soft Error Rate (SER) is increasing with decreasing node size, while the Hard Error Rate remains constant [2]. Given their low costs and good performances, Commercial-Off-The-Shelf (COTS) microcontrollers are increasingly deployed in safety applications [3]. In contrast to safety-certified microcontrollers, COTS microcontrollers are not produced with an integrated protection against soft errors. As a consequence, recent research proactively engages environmentally induced soft errors by developing new methods for error detection, mitigation, and data recovery [4]. This research direction has also yielded new challenges and requirements.

The importance of detecting and resolving soft errors is reflected by the numerous reports on soft error problems within safety-critical applications. Reports are coming from a wide range of industries, such as the automotive industry, space industry, or the medical industry. Duncan and Roche's analysis of semiconductor reliability in the context of autonomous driving [5] is devastating, as they conclude a (soft error induced) failure rate of 1 part per million per year. Given that a single-car implements approximately 8,000 semiconductors,

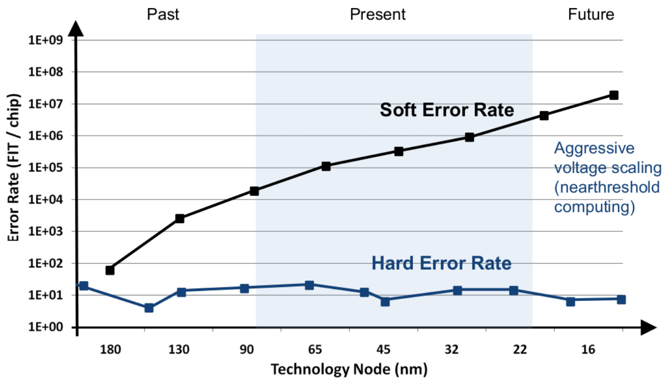

Figure 1. Software and hardware error rates in semiconductors [2].

the likelihood of a car exhibiting semiconductor induced errors within its lifespan (of 15 years) is around 12%. While the results of such failures are unclear while a car is operated, semiconductor-based soft errors can be resolved (fairly easy) by restarting the affected component. Not all safety-critical systems provide the luxury, of resolving an error by "turning it off and on again". Consider, for example, safety-critical nuclear power plant equipment, where restarting a device in the event of a soft error is not an option and could lead to fatalities.

Safety-critical applications usually exhibit different levels of criticality in terms of their underlying data. While a fraction of data is system critical (i.e., if affected by an error the consequences can be catastrophic), errors affecting non-critical data will not impact the safety of operation. This phenomenon is known as mixed-criticality. Incorporating mixed-criticality into the design of mitigation strategies, by devising and applying different detection and correction strategies on memory areas holding data of different levels of criticality, allows to further improve a system's availability while guaranteeing a correct treatment of system critical events.

The remainder of the paper is organized as follows: Section II presents an overview of the mitigation strategies. Section III defines the challenges and requirements for soft error software-based mitigation strategies in safety-critical applications. Section IV shows the evaluation of the two most suitable software-based mitigation strategies. Section V introduces how existing safety architectures can be improved with the software-based approach. In the last section, summary and future work of the paper are presented.

II. Mitigating soft errors

While soft errors represent the majority of memory errors, they can be prevented and/or corrected. To prevent soft errors, memories require fault-tolerance. Fault-tolerance denotes a systems' ability to handle faults in individual hardware or software components, power failures, or other forms of unexpected problems, while still meeting its specification [6]. There are different approaches to achieving fault tolerance.

Shielding constitutes one of the first approaches, that made components fault-tolerant. Shielding is applied during the production phase, where a specific particle-resistant layer was deployed over the component's package. The layer reduces exposure of the bare component/device and prevents environmental particles from influencing under-layers of the package. Resistance to the electrical charges can also be achieved by using specific designs and materials for critical points in the component (e.g., strengthening the gate of the transistors). Although techniques used during the production phase have shown very effective against soft-errors, they always require additional materials and significantly increase the cost of the design and production.

If early stage design protections are not available, which is often true for the COTS microcontrollers, then the system redundancy is a very common solution to establish a fault-tolerant system. Four main types of redundancy exist: hardware, software, information, and time redundancy. The first three types of redundancy are achieved by providing additional components, functions, or data items that are not required for fault-free operation, but function as a backup for the event of a fault. Timing redundancy denotes repeating computations and a comparison of computational results from different timings.

1) Hardware redundancy: Hardware system architectures can provide fault tolerance via hardware redundancy. Safety-critical systems often adopt an N-modular (where $N > 2$) architecture, where the components exist in certain redundancy n and perform the same computations in parallel. The correct result is established based on majority voting. If one of the modules fails, the majority voter masks the fault by identifying the result of the remaining fault-free modules [6]. N-modular systems can yield towards a higher Safety Integrity Level (SIL), as they provide inherent fault tolerance and consequently a low failure rate. SIL is a quality indicator for systems that fulfill safety requirements in accordance with the IEC61508 standard. Many safety systems use simple architectures such as 1oo1D (1-out-of-1 with diagnostics) and 1oo2D (1-out-of-2 with diagnostics) [7]. In some cases, a diagnostic system is realized with an additional CPU (i.e., lock-step architecture) or with an additional watchdog (i.e., challenge-response architecture) [8]. These architectures are also known as fail-safe where in the event of a specific type of failure, the system inherently responds in a way that will cause no or minimal harm to equipment, environment or people. The main advantage is that these architectures have a good balance between functional safety (i.e., achieving high safety integrity) and development process costs. A shortcoming of hardware redundancy is its requirement for additional hardware. In the context of memory, it will increase its cost, weight, size, power consumption, and thus, impacts its designs and tests. Moreover, additional hardware needs to be in-calculated from the first stage of chip design. It is therefore almost impossible to upgrade already existing systems with additional hardware without degrading its performances, making it unsuitable for brownfield automation.

2) Software redundancy: Software fault-tolerant techniques are also based on the redundancy, which is applied to procedures, processes, data or the whole execution code. The most common type of software redundancy in embedded systems is the multiplication of data. A simple way of doing this is to store a variable copy simply transformed (e.g. with hamming distance 4 or simple inverse function) in a different memory area. This helps detect (via comparison), mitigate or recover corrupted data. The main disadvantage of software redundancy is memory consumption because multiplication of data, code or processes requires additional memory space that is usually limited in embedded systems. Also, in some cases, the code execution time could be significantly increased [1], [4].

3) Informational redundancy: The most prevalent type of redundancy in the context of memories is *Informational redundancy*. It assumes the addition of extra information to the data, which allows verifying the soundness of the information. Usually, this additional information are codes, which are computed based on the data itself. Those codes (so-called Error Detection And Correction codes (EDAC)) were firstly used in communication [6] for data recovery, but nowadays they are widely used in the memories [9]. The family of EDAC codes is expanding constantly, so far the most popular EDAC codes are: Parity Codes (error detection without recovery) [10], Hamming Codes (2-bit detection and 1-bit recovery) [10], Reed-Solomon and Bose-Chaudhuri-Hocquengham Codes (for multiple bits error masking) [9]. Also, some works considered the implementation of other EDAC codes used in communication such as: LDPC codes [11], RS codes, Turbo codes [12].

Most of nowadays EDAC codes for memories are implemented with an additional chip which is used to encode and decode EDAC codes [13]. These additional chips increase the cost of memory by 10-20%. Also, the memory's die-area increases by around 20% and processing speed decreases by 3-4% [9]. To avoid an increase in chip size and hardware re-designs, software-based EDAC codes have been proposed [14], [15]. However, such an approach leads to a decrease of available memory as well as an increase of computation time, access time, and the complexity of the overall system and usual trade-offs between listed parameters need to be made. It is worth mentioning that some conventional micro-controllers are already offering embedded memories with EDAC codes (i.e., hamming code or parity bit chips that can protect data from faults [16], [17]).

EDAC codes have two main properties that need to be considered: speed and quality. Speed is defined as the time needed for encoding/decoding EDAC codes and this time extends the overall memory access time, while quality can be determined as a number of the faulty bits that the code can detect and correct. Naturally, there is a trade-off between quality and speed. For higher quality, more complex EDAC codes are required, which allow correcting multiple bit-flips. In this case, both, code magnitude as well as computing demand are increased due to these adaptations. Faster and less memory expensive correction schemes are limited in terms of the number of bits that can be corrected.

Based on EDAC codes, a new method called scrubbing was also developed. The idea behind scrubbing is to periodically re-write data in its original location and eliminating soft errors, if they are correctable through EDAC [18], or copy of original data [19]. With this approach, an accumulation of soft errors inside one region of memory can be avoided.

4) Timing redundancy: Another method that has been recently investigated is the so-called timing redundancy. It involves repeating a computation or data transmission two or more times and comparing results with previously-stored copies [6]. This type of redundancy is good when we need to distinguish between transient and permanent errors. If the fault is still there after repeating the test several times, then it's likely that error is a permanent one.

III. Challenges in mitigating soft errors

To overcome soft errors, and consequently lower their impact on the non-functional properties of the system, various methods for error detection, correction, and mitigation were introduced. Available methods can be distinguished into hardware- and software-based correction mechanisms. Hardware-based mechanisms provide error detection and correction on an architectural level and use specific hardware. As already stated, hardware approaches are not applicable in the brownfield i.e., existing devices or systems and they are usually demanding redesign and redeployment. For the already deployed systems or devices, software solution fits better because they can be deployed with a simple update or software patch and consequently costs are minimized. Software-based correction mechanisms operate on the memory itself without altering the underlying hardware or architecture. Depending on the application, an adequate correction quality is required, which denotes the fault magnitude a strategy is capable to detect, mitigate, and/or recover. Given that there is no such thing as a free lunch, soft error strategies require additional execution time and/or memory space, and therefore affect processor run-time and can cause memory overhead. In the next section, challenges for software-based solutions for soft-error mitigation will be discussed.

These observations lead to a general trade-off problem for the design and deployment of soft error detection and correction, as it is always required to balance the quality of detection (required by the underlying application) and the resources required to implement appropriate correction and detection strategies. Higher quality of the soft error correction will require more computation time, space, and sometimes additional hardware. Depending on the target system, this might lead to a violation of the system's requirements (in terms of cost, available memory space and computation time for the system's applications). In the following, the system's requirements are outlined in more detail.

1) Run-time performance: The development of methods, which provide sufficient error coverage, while keeping the impact on the system's run-time or memory overhead minimal is particularly important in the context of safety-critical systems. This is due to the fact that such systems have very strict timing requirements (i.e., norms in the field define specific timing limits here, such as Fault Tolerant Time Interval (FTTI) in ISO26262 or Process Safety Time (PST) in the IEC61508 standard). The FTTI constitutes the timespan between fault and hazard [20]. Faults must be detected and corrected within this interval. If a correction is not possible, the system must guarantee to reach a safe state within the FTTI. Therefore, the run-time performance of correction strategies plays a crucial role in the context of safety-critical systems as its application must not lead to a violation of the FTTI requirements.

2) Memory consumption: Many strategies require additional memory space for their implementation, which is used to store copies of data or code, or additional information (required by the method), such as Parity bits or Error Detection and Correction Codes (EDAC). From all software solutions, EDACs codes exhibit the smallest overhead because the ratio between additional bits required for protection and protected bits is always less than one while this is not the case for full redundancy. While in most cases EDAC codes can have a large memory footprint, parity bits constitute their most lightweight form. They allow monitoring the consistency of a memory region (with a defined length) based on a single bit, which denotes if the number of one-bits in the region is odd or even. With decreasing the size of the protected region this can lead to increased memory overhead. To give an example: the protection of 32bit word via hamming code will result in a 3.15% memory overhead). One-bit recovery of a 32-bit word, using Hamming code, would require additional 7 bits and thus result in a memory overhead of 22%.

3) Mitigation quality: The quality of a strategy is defined by its capability of detecting and correcting (recovering) faulty bits. Furthermore, detecting and correcting capabilities are expressed by the number of faulty bits that can be detected and corrected. The simplest EDAC code (Parity) can detect all odd bit flips but doesn't have recovering capabilities. On the other hand, a 2oo3 system can detect all bit flips and also can correct, but the complexity and consequently costs are higher.

In fail-safe systems, detection of an error is usually reflected with the safety feature because detection is enough to trigger activation of the safe state and prevents further safety issues. Between error detection and activation of a safe state, the system has a defined time for the recovery procedure. If recovery is not possible for any reason, the system will go into the safe state and availability will be affected.

4) Mixed criticality: When speaking about safety-critical memory one must distinguish between different levels of safety-criticality, which applies to the system data. Especially in safety-critical applications, some data may have a higher criticality level than the other. As already outlined, this phenomenon is known as ”mixed-criticality”. While adequate protection needs to be provided for the whole system, safety-critical data requires stronger protection. Several recent studies have investigated mixed-critically in memories, with a focus on data delivery and prioritization by data criticality [21].

Taking mixed-criticality into account when designing memory detection and correction strategies, allows enhancing the reliability and safety of the underlying system, as such strategies aim for increasing the protection of safety-critical memory parts. By treating different parts of the memory with a different criticality, the overhead of the correction strategies can be reduced (in contrast to the whole memory being subject to rigid correction/detection strategies). In addition, incorporating mixed-criticality can increase a system's availability, as faults in non-system critical memory areas will not necessarily lead to the halt of the system.

5) Memory organization: Because of the environmental changes, occurrences of soft errors in memory are not continuous. The chance of a cell being hit by an error is randomly distributed. Therefore, errors can appear at any time and in any type of memory or memory part. This can aggravate memory protection and detection mechanisms as they are type-dependent. One can distinguish between two types of memory in embedded systems: non-volatile and volatile memory. Non-volatile memory sustains stored information during a loss of power (e.g., flash memory), while volatile memory needs constant power to retain stored data (e.g., SRAM) [22].

Embedded memory exhibits different regions: program memory, data memory, registers and I/O ports [23]. Also, from the software point of view, the memory layout of C/C++ programs consist of the different sections, that are saved in different memory regions. Typical memory representation of C/C++ programs consists of a code segment, data segment, uninitialized data segment (bss), stack and heap. All of this can impact the design of the correction/mitigation mechanism.

IV. Evaluation of EDAC codes

As shown in the last section, it is crucial to estimate the performance and overheads of soft-error mitigation strategies in order to identify those appropriate for one's problem domain and underlying system requirements. This section demonstrates how such an assessment could be performed, by calculating and comparing memory consumption and run-time performances of the Parity Bit (PB) and Extended Hamming Code (EHC).

The evaluation is performed for varying lengths of protected data (as strategies scale different with these). For the representation of the codes a common annotation (n, k) was used, where n denotes the number of total bits and k the number of the protected data bits. The number of required check bits can be easily calculated as $n - k$. Utilizing these parameters, memory consumption (mc) is calculated in (1) and exhibited on the Figure 2

$$mc[\%] = (n - k)/k \cdot 100\% \qquad (1)$$

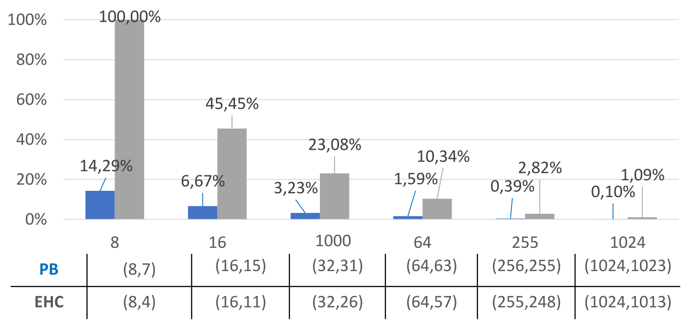

Figure 2. Memory overhead for different types of Parity Bit (PB) and Extended Hamming Code (EHC), where the x axis denotes total length of word and y denotes percentage of the memory overhead.

The run-time performance of a given strategy is closely connected with the complexity of the underlying algorithm. A good indicator of the algorithm's complexity is the number of logical XOR operators it requires for its implementation.

In the context of PB, a calculation stemming from [24] was used. The algorithm is based on the consecutive application of shift and XOR operators. Alternatively, a lookup table could be used to calculate the parity bits of 8-bit words. While using a look-up table will slightly increase the memory consumption of the algorithm it will decrease its complexity by 3 XORs.

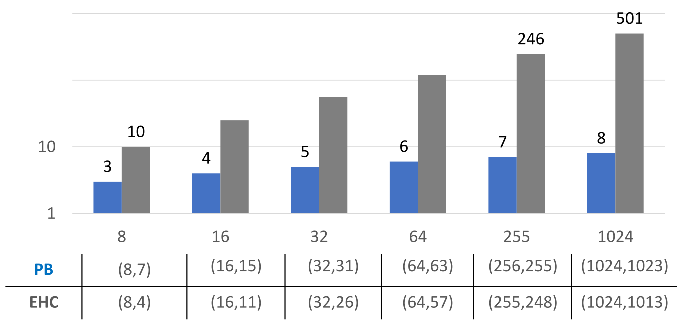

Figure 3. Number of XORs for encoding process for different types of PB(n, k) and EHC(n, k), where y-axis denotes the number of total XORs gates and x-axis the number of the protected data bits.

The number of the XORs for EHC was calculated according to (2).

$$XORs(k) = 2^{k+1} - k - 3 \qquad (2)$$

where parameter k can be derived from the following form of hamming code annotation $H(2^k, 2^k - k - 1)$. The equation (2) stems from [24] where it was calculated for the EHC recursive encoding computation. Figure 3 shows the number of the XOR operator for varying lengths of protected bits.

In terms of mitigation quality, a big difference between PB and EHC can be observed. While PB is only capable to detect errors with an odd number of bit-flips (including single-bit errors), EHC can detect 2 errors and correct only one flipped bit. In the context of safety-critical systems, this low mitigation quality has a big impact on availability and safety.

In [25] a detailed report on the number of soft errors in SRAM memory (512K x 8-bit) is given, which were observed in space. Errors were recorded in a nanosatellite that was circulating the Earth's orbit. During the 2510 days of recording, four different types of 247593 soft errors occurred. The majority of these errors were single-bit errors (i.e. a total of 244150 errors constituting 98.6% of the recorded errors), while only 2996 errors (i.e., 1.21% of the recorded errors) were double-bit errors. Multiple bit (> 2) errors occurred at an even lower rate (corresponding to a total of 217 errors (0.08%)), while the remaining errors (230 (0.09%)) were classified as severe errors.

If the capability of presented algorithms was considered in this example in addition to considering the safety-critical scenario, PB would detect all single-bit errors and some of the multiple bit errors, leading to a detection rate of 98.75%. PB detection alone is not enough and would not increase the availability of the system, because without recovery the sole identification of an error would lead to the system being put into a safe-state as it is not safe to continue calculations. Using EHC, 99.8% error would be detected and 98.6% would be corrected. This means that the system's availability could be increased significantly as it would only be stopped (put in a fail-safe state) for 1.4% of the errors. This leads to the conclusion that (on its own) EHC is significantly better when it comes to safety and availability, however, this is also associated with the higher memory overhead and complexity (as shown before). Also, one should keep in mind that the SRAM used was relatively old (approximately 20 years old), and thus exhibits a lower probability for multiple bit errors because of the higher technology node. With newer memories (utilizing smaller technologies) the distribution of the error is very likely to be different (i.e. more multiple-bit errors are to be expected).

In the context of safety-critical systems, the application of specific fail-safe architectures with hardware redundancy is very common. The next section will introduce a widely used fail-safe architecture and demonstrate how the application of simple EDAC codes can further improve its availability.

V. Enhancing an 1oo2D safety architecture

A typical representative of a fail-safe system is a 1oo2 architecture where the hardware, including sensor inputs, is independently implemented twice. This leads to a multi-core architecture similar to the one described in [26]. The output of these parallel lines is checked and selected by a voter [27]. Therefore, when the two outputs differ, the result leading to a safe and non-critical state is preferred and opted for by the voter.

From a memories' point of view, a 1oo2D architecture provides independent memories for each parallel line of computation. Two independent parallel memories ensure system hardware and software redundancy. This means that (besides memory specific data which is required for synchronization) identical data can be found on both memories (Figure 4 depicts the memory model in a 1oo2D architecture).

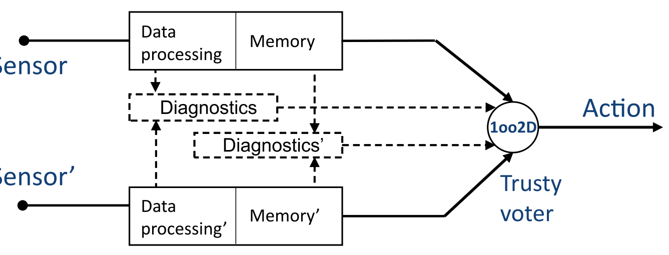

Figure 4. 1oo2D safety architecture.

All regions are equally exposed to the faults, however, different kinds of protection can be applied to different regions. Experts advise that protection should be implemented in the form of periodical tests run over data. As an exemplary guide, we can refer to the Safety manual [28] provided by STMicroelectronics for their micro-controllers. For soft errors, STMicroelectronics advises using redundancy for all safety-relevant variables. Usual solutions provide a copy of original data on the same memory chip or an additional (redundant) chip. The data is periodically compared with the original to detect the presence of errors [29]. When an error is detected, it is not clear which memory (or part of the memory) was affected, therefore such a solution leads to the detection but not to the correction and will result in the system transitioning into a safe-state.

A solution for overcoming this problem is to add mechanisms (on top of the existing architecture), which allow recovering faulty data and extend uptime of the system. Recovery mechanisms in this context are usually EDAC codes based. Adding additional hardware to the system is not feasible, as this would require redesigning the system from scratch. Another option is to apply software-based EDAC codes approaches.

Given that 1oo2D already provides the possibility to detect memory errors, the question arises on how existing architectures (i.e., 1oo2D) can be combined with software-based approaches.

A method, for enhancing existing 1oo2 hardware architecture, was proposed in our work [30]. This method constitutes an extension for mixed-critical real-time systems with an underlying 1oo2 architecture. We refer to it as Redundant Parity (RP). Figure 5 explains the basic concepts of the RP method. The method relies on 1oo2's ability to detect soft-errors and uses parity bits to establish the location of the error. Initially, the method generates parity bits for data that needs to be protected (i.e., data in redundant memories). When bit flips occur and 1oo2 comparator detects different bits in redundant data, the usual way is to generate a signal that will trigger the safe-state of the device. In contrast to that, the proposed method calculates new parity bits for both protected parts of the memories. In the next step, old parity bits are compared with newly calculated parity bits to establish the fault source. When the algorithm distinguishes between healthy and faulty data, the recovery phase is activated. Recovery is performed by simply copying healthy over the faulty data. To sum it up, the method uses 1oo2 architecture's inherent capability to detect bit flip in combination with parity bit to detect which of the redundant words is faulty and, in the end, it uses redundancy for recovery.

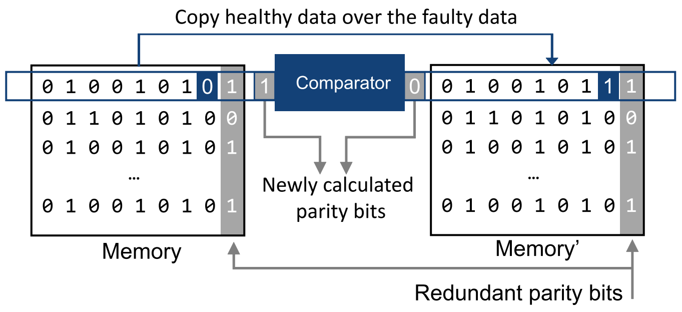

Figure 5. Redundant parity method.

The method allows for correcting single-bit soft errors (the majority of occurring soft-errors). In addition, odd multiple bits soft errors can be corrected and even multiple bits can be detected. In the context of the error data presented in the Section IV, this method would detect 100% of the errors and correct 99.4% of the errors. Memory overhead and complexity would be equal to double memory overhead and double complexity of the parity bit. Furthermore, the RP method provides separated detection and recovery phases, leading to less recovery time than in other EDAC methods. In addition, the proposed method is completely independent of the software architecture as it focuses on the memory's word level rather than on the variables or structures [31]. However, the results also show that the application of the approach is limited to a 1oo2 architecture, which already provides the required data redundancy as well as self-tests to detect errors in the data.

VI. Conclusion

The main goal of this work was to review software-based mitigation strategies for mixed-critical memories and identify challenges, that need to be considered. Soft errors, induced by external environmental factors, constitute a problem in memory operation. As safety certificated microcontrollers are expensive and complex industry is often utilizing COTS microcontroller.

To increase availability and reliability within COTS memories, a certain level of fault tolerance is required. Current safety-critical applications rely on simple fail-safe architectures like 1oo1D or 1oo2D (which were outlined in Section V). The reliability and availability of fault-tolerant systems can be further improved if such simple fail-safe architectures are extended with software-based recovery techniques such as EDAC codes. In addition, deployment of the software-based EDAC codes does not require additional hardware or a redesign of the underlying architecture.

When deciding on a method to be implemented on existing hardware, one must be aware of the overhead costs, which are associated with a respective method, as it will likely increase run-time and/or reduce the available memory space. This aspect can be incorporated in strategy design, by directly addressing mixed-criticality of data within the correction and detection strategies, and differentiating among memory regions. The article tried to outline how such an assessment could be performed, by calculating and comparing memory consumption and run-time performances of different strategies, which can then be linked to the existing requirements of existing safety architectures, such as 1oo1D or 1oo2D.

The comparison of PB and EHC showed that, while PB exhibits less complexity and run-time overhead it will not increase availability per se, as detection will not lead to correction (in contrast to EHC). However, when PB is combined with existing 1oo2 safety architectures, a mitigation approach (named redundant parity) can be established, which is able to both detect and correct most of the soft-error occurring in memories, and thus significantly improve availability.

The method utilizes 1oo2's inherent capability of soft error detecting (achieved by a simple comparison test) and adds the mechanism of parity bits to distinguish between faulty and healthy data. In case an error is detected, the innate redundancy of the 1oo2 architecture is used to recover the error by copying healthy over faulty data.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the Austrian Research Promotion Agency (FFG) (#6112792).

REFERENCES

[1] J. Vankeirsbilck, H. Hallez, and J. Boydens, “Soft error protection in safety critical embedded applications: An overview,” in 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), November 2015.

[2] H. Iwashita, “International standards adopted by itu-t to address soft errors affecting telecommunication equipment,” ITU-T International Telecommunication Union - Telecommunication Standardization Sector, Geneva, CH, Standard, 2018.

[3] H. Forsberg and K. Karlsson, “Cots cpu selection guidelines for safety-critical applications,” in 2006 IEEE/AIAA 25TH Digital Avionics Systems Conference, Oct 2006.

[4] V. Thati, J. Vankeirsbilck, J. Boydens, and D. Pissoort, “Data error detection and recovery in embedded systems: a literature review,” Advances in Science, Technology and Engineering Systems Journal, 2017.

[5] M. Duncan and P. Roche, “Paving the way towards autonomous driving — tackling soft errors to security challenges,” in 2017 IEEE International Reliability Physics Symposium (IRPS), April 2017.

[6] D. Elena, Fault-Tolerant Design. KTH Royal Institute of Technology, Krista, Sweden: Springer, 2013.

[7] F. Handermann, “Process safety architecture system neutral solution comparison,” Chemical Engineering Transactions, April 2016.

[8] R. Mariani and P. Fuhrmann, “Comparing fail-safe microcontroller architectures in light of iec 61508,” in IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007), 2007.

[9] A. Mukati, “A survey of memory error correcting techniques for improved reliability,” Journal of Network and Computer Applications, 2011.

[10] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Application. New York, NY, USA: Wiley-Interscience, 2006.

[11] S. Jeon, E. Hwang, B. V. K. V. Kumar, and M. K. Cheng, “Ldpc codes for memory systems with scrubbing,” in 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, Dec 2010.

[12] B. Tahir, S. Schwarz, and M. Rupp, “Ber comparison between convolutional, turbo, ldpc, and polar codes,” in 2017 24th International Conference on Telecommunications (ICT), May 2017.

[13] M. Restifo, P. Bernardi, S. De Luca, and A. Sansonetti, “On-line software-based self-test for ecc of embedded ram memories,” in IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, October 2017.

[14] N. Maruyama, A. Nukada, and S. Matsuoka, “Software-based ecc for gpus,” Symposium on Application Accelerators in High Performance Computing, January 2009.

[15] D. Dopson, “Softecc: a system for software memory integrity checking,” Ph.D. dissertation, Institute of Technology. Dept. of Electrical Engineering and Computer Science, Massachusetts, 2007.

[16] Intel® Embedded Memory User Guide, STMicroelectronics.

[17] MWCT101xS Safety Manual, NXP Semiconductors.

[18] G. Mayuga, Y. Yamato, T. Yoneda, M. Inoue, and Y. Sato, “An ecc-based memory architecture with online self-repair capabilities for reliability enhancement,” in 20th IEEE European Test Symposium (ETS), 2015.

[19] R. Santos, S. Venkataraman, A. Das, and A. Kumar, “Criticality-aware scrubbing mechanism for sram-based fpgas,” in 24th International Conference on Field Programmable Logic and Applications, 2014.

[20] IEC, “International Standard 61508 Functional safety: Safety related Systems,” International Electrotechnical Commission, Geneva, CH, Standard, 2005.

[21] J. S. Miguel and N. E. Jerger, “Data criticality in network-on-chip design,” in Proceedings of the 9th International Symposium on Networks-on-Chip, ser. NOCS ’15. New York, NY, USA: ACM, 2015.

[22] K. Itoh, “Embedded memories: Progress and a look into the future,” IEEE Design Test of Computers, January 2011.

[23] Reference manual for STM32 applications, Intel.

[24] L. Zhengrui, L. Sian-Jheng, and H. Honggang, “On the arithmetic complexities of hamming codes and hadamard codes,” 2018.

[25] H. Caleb and B. Vipin, “Error detection and correction on-board nanosatellites using hamming codes,” Journal of Electrical and Computer Engineering, 2019.

[26] F. Reichenbach and A. Wold, “Multi-core technology – next evolution step in safety critical systems for industrial applications?” in 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, September 2010.

[27] C. Preschern, N. Kajtazovic, and C. Kreiner, “Built-in security enhancements for the 1oo2 safety architecture,” in 2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), May 2012.

[28] STM32F4 Series safety manual - user manual, STMicroelectronics.

[29] Handling of soft errors in STM32 applications, Intel.

[30] A. Kajmakovic, N. Kajtazovic, K. Diwold, R. Zupanc, and G. Macher, “Flexible soft error mitigation strategy for memories in mixed-critical systems,” in 2019 ISSREW: International Workshop on Software Hardware Interaction Faults, Oct. 2019.

[31] A. Kajmakovic, R. Zupanc, S. Mayer, N. Kajtazovic, M. Höffernig, and H. Vogl, “Predictive fail-safe improving the safety of industrial environments through model-based analytics on hidden data sources,” in Proceedings of the 13th IEEE International Symposium on Industrial Embedded Systems. IEEE Press, June 2018.