

Effective but Lightweight Online Selftest for Energy-Constrained WSNs

Ulf Kulau, Daniel Szafranski and Lars Wolf
 Institute of Operating Systems and Computer Networks
 TU Braunschweig
 Email: [kulau|szafranski|wolf]@ibr.cs.tu-bs.de

Abstract—In real world Wireless Sensor Network (WSN) applications nodes can be affected by soft errors that occur either accidentally or are introduced deliberately. Especially when WSNs are deployed outdoors harsh environmental conditions may impair the proper function of devices. However, soft errors might also be provoked intentionally to increase the energy efficiency of nodes when, e.g., undervolting is applied.

In any case the detection of an error is essential as small failures could propagate throughout the system and downgrade the efficiency and effectiveness of the entire network. Hence, this paper presents an approach to detect errors with minor overhead, independent from the used micro controller unit (MCU) and with a high effectiveness. To perform an extensive evaluation we implemented automatic test equipment to enable the investigation on different types of MCUs. The results show that our approach outperforms classical approaches while not being hardware-dependent.

I. MOTIVATION AND INTRODUCTION

In space application it is common practice to perform selftests to check the integrity of the entire system. The radiation outside the earth's magnetic field leads to single event upset (SEU) when ionized particles hit, e.g., the MCU and cause bit-flips at digital gates resulting in *soft errors*. It is essential to detect such soft errors as unpredictable malfunction of the entire system could be the consequence. For this reason mechanisms like Triple Modular Redundancy (TMR) are common practice to overcome this issue [1]. On earth the electronics do not suffer from radiation but soft errors do also occur. Thus, for critical applications like embedded devices for aviation, vehicular or medical engineering similar approaches are applied to enable high safety levels [2], [3], [4]. With regard to WSNs the presence of soft errors might play a secondary role as these networks are already known as rather unreliable [5]. Recent publications revealed that soft errors have a significant impact on the energy efficiency of WSNs, either negative or even positive [6], [7].

On the one hand there is a variety of sources of soft errors on WSN nodes that happen accidentally and decrease the energy efficiency of nodes and the entire network. Low-grade hardware as well as rough environmental conditions could cause a malfunction of the node's hardware. In [8] the authors reported that errors occurred due to high temperatures when a WSN is deployed outdoors. Moreover and with regard to long-term WSN deployments extreme temperatures accelerate the aging of components resulting in soft errors, too [9]. In the first instance the efficiency of the affected node decreases.

However, considering a WSN that implements adaptive routing such temporary failures are not locally bounded due to an increased control overhead. In our previous work [7] we investigated the impact of temporary node failures on the energy efficiency of WSNs using Routing Protocol for Low power and Lossy Networks (RPL). Depending on the scenario and topology, one malicious node can increase the total energy consumption by up to 8 % while individual nodes may consume 20 % more. Furthermore, resetting nodes multiple times can amplify this effect. Hence, a reliable and early detection of soft errors would be highly beneficial to mitigate an induced energy overhead.

On the other hand soft errors might also be evoked willfully to increase the energy efficiency of nodes when active undervolting is applied. With *IdealVolting* [6] we previously implemented an adaptive undervolting scheme for WSN nodes which decreases the power dissipation significantly but in exchange of a slightly increased risk of evoking soft errors. The actual voltage level at which a node or an MCU still works properly depends on the clock speed, the temperature, and its individual properties due to variations in the manufacturing process [10]. To adapt each node to its lowest usable voltage level, *IdealVolting* implements a control loop between an undervolted primary MCU and a tiny but reliable secondary MCU. A key factor is the periodic execution of a spot test to detect soft errors due to insufficient supply voltage of the MCU. Based on the test results, that are further validated by a secondary MCU, we can decide whether the voltage level can be further lowered or not.

In any case an adequate online selftest can help to increase the reliability, effectiveness and efficiency of nodes and the entire WSN, respectively, when soft errors are detected at an early stage. Hardware-based approaches like the TMR are not lightweight enough to be used in failure prone but energy constrained networks like WSNs. Thus, in this paper we present a platform-independent online selftest that matches the requirement of low overhead while detecting soft errors with a high probability. Therefore, we first review and discuss some related work in the following Section II. In Section III the online selftest is presented. To evaluate the selftest against other approaches and to show its platform-independent utilization, we implemented an automatic test equipment that supports different types of MCU. The test equipment is described in Section IV as well as a short evaluation of its characteristics.

With the aid of the test equipment an extensive evaluation is performed in Section V before Section VI summarizes the paper.

II. RELATED WORK

Already in 1984 Huang Kuang-Hua and J. A. Abraham described the occurrence of soft errors in computer systems [11]. They assert that the increasing complexity of computers increases the probability of faults on a physical layer. However, as mentioned in the introduction, there is a plethora of MCU-based applications, e.g., medical devices or vehicle systems, where reliability and safety have a high priority. Hence, research has been going on for many years and a lot of different approaches and procedures have been proposed. While redundancy-based methods further offer the opportunity for error correction, their overhead is significant and, therefore, not suitable for energy-constrained applications like WSN nodes. More lightweight solutions are based on software tests to detect errors and to check the integrity of the system.

Eghbal et al. [12] and Rohani et al. [13] investigated how hardware-faults affect different applications on commercial off-the-shelf (COTS) MCUs. For this purpose they injected transient faults in different components such as arithmetic logical unit (ALU), instruction register, program counter, register file and flag registers. The analysis showed how soft errors manifest in different scenarios and revealed that the ALU is one of the components that is most affected by soft errors. This is an important finding, as an error-free ALU is therefore a verifiable indicator for the reliability of an MCU. For this reason, many existing work focused on algorithms that require a high ALU participation, but can be implemented platform independent and in a lightweight manner. These characteristics are well established in Algorithm-based fault tolerance (ABFT) procedures, so that they form the basis of our selftest implementation.

The preliminary work for modern ABFT procedures was performed by Huang Kuang-Hua and J. A. Abraham [11]. They describe the increased susceptibility of computationally-intensive matrix operations to soft errors. Since many numerical methods such as „Addition“, „Multiplication“ or „LU-decomposition“ are based on matrix operations, they saw a need to develop a method for error detection and correction in matrix operations. Hence, they proposed a checksum-based method for fault-tolerant matrix multiplication, that is able to detect more than 99.9 % of all computational errors.

Based on this work, Panruo Wu et al. [14] extended the procedure by an online aspect. As a result, calculation errors could already be detected and often corrected directly after intermediate steps of a matrix multiplication.

In WSNs we have to deal with nodes that have low computational capabilities and are designed to be highly energy efficient. Hence, a suitable selftest should provide both, low overhead and a high probability of error detection. Our selftest combines properties of the ABFT and online features. Furthermore, we create a generic selftest implementation for the detection of soft errors on different hardware platforms.

III. ONLINE SELFTEST

Basically, our selftest is based on an online capable fault-tolerant matrix multiplication, as described in [11] and [14].

Therefore, two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times r}$ with $m, n, r \in \mathbb{N}$ are being used:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \in \mathbb{R}^{m \times n} \quad (1)$$

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,r} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,r} \end{pmatrix} \in \mathbb{R}^{n \times r} \quad (2)$$

In preparation for the error-detection mechanism, elements for checksums are added to the regular matrices A and B . In doing so A and B are extended by an extra row and column, respectively, resulting in A_c and B_c :

$$A_c = \begin{pmatrix} A \\ J_a \cdot A \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \\ a_{m+1,1} & a_{m+1,2} & \cdots & a_{m+1,n} \end{pmatrix} \in \mathbb{R}^{(m+1) \times n} \quad (3)$$

$$B_c = \begin{pmatrix} B \\ B \cdot J_b \end{pmatrix} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,r} & b_{1,r+1} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,r} & b_{2,r+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,r} & b_{n,r+1} \end{pmatrix} \in \mathbb{R}^{n \times (r+1)} \quad (4)$$

$$\text{with matrices of ones } J_a \in \mathbb{1}^{1 \times m} \text{ and } J_b \in \mathbb{1}^{r \times 1} \quad (5)$$

The function to calculate the checksums is arbitrary. In this case we tend to reduce the computational overhead so that a simple summation is implemented:

$$a_{m+1,j} = \sum_{i=1}^m a_{i,j} \quad \text{with } j = [1, \dots, n] \quad (6)$$

$$b_{i,r+1} = \sum_{j=1}^r b_{i,j} \quad \text{with } i = [1, \dots, n] \quad (7)$$

The multiplication of the checksum matrices $A_c \cdot B_c$ results in a checksum matrix $C_c \in \mathbb{R}^{(m+1) \times (r+1)}$. In addition to the actual result of the sole matrix multiplication $C = A \cdot B \in \mathbb{R}^{m \times r}$, C_c also contains the checksums in form of column and row sums of the matrix C :

$$A_c \cdot B_c = C_c = \begin{pmatrix} C & C \cdot J_b \\ J_a \cdot C & J_a \cdot C \cdot J_b \end{pmatrix} \in \mathbb{R}^{(m+1) \times (r+1)} \quad (8)$$

$$C_c = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,r} & c_{1,r+1} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,r} & c_{2,r+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,r} & c_{m,r+1} \\ c_{m+1,1} & c_{m+1,2} & \cdots & c_{m+1,r} & c_{m+1,r+1} \end{pmatrix} \quad (9)$$

This simple validity check reveals whether any errors have occurred in the multiplication.

On the one hand the calculation was correct when equations 10 and 11 apply.

$$\forall j \in [1, \dots, r+1] : c_{m+1,j} = \sum_{i=1}^m c_{i,j} \quad (10)$$

$$\forall i \in [1, \dots, m+1] : c_{i,r+1} = \sum_{j=1}^r c_{i,j} \quad (11)$$

On the other hand the following inequalities 12 or 13 imply that at least one soft error occurred and the MCU must be considered unreliable:

$$\exists j \in [1, \dots, r+1] : c_{m+1,j} \neq \sum_{i=1}^m c_{i,j} \quad (12)$$

$$\exists i \in [1, \dots, m+1] : c_{i,r+1} \neq \sum_{j=1}^r c_{i,j} \quad (13)$$

In order to adapt these calculations to energy-constrained MCUs of a WSN node, an optimized sequence is applied to reduce the overhead to compute C_c . In concrete terms, the row and column calculation is executed alternately. The main principle of this approach is shown in Figure 1. It enables an early detection of soft errors as checksums can be validated immediately. Furthermore, only square matrices with $n = m = r$ are recommended to simplify multiplication and implementation.

$$C_c = \begin{pmatrix} \begin{matrix} \xrightarrow{1} c_{1,1} & \xrightarrow{3} c_{1,2} & \cdots & \xrightarrow{5} c_{1,r} & \xrightarrow{2} c_{1,r+1} \\ \downarrow 2 & \downarrow 4 & & & \downarrow 6 \\ c_{2,1} & c_{2,2} & \cdots & c_{2,r} & c_{2,r+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,r} & c_{m,r+1} \\ \downarrow & \downarrow & & & \downarrow \\ c_{m+1,1} & c_{m+1,2} & \cdots & c_{m+1,r} & c_{m+1,r+1} \end{matrix} \end{pmatrix}$$

Figure 1: Optimized sequence for calculating the elements of a matrix multiplication, shown for the first steps.

In order to estimate the calculation effort, a overhead analysis was carried out for various n . Therefore, it is assumed that the first checksum calculation for the two initial matrices (A and B) is either done manually or represents a one-time and negligibly small overhead, which is not considered below.

The mathematical foundations show that for each element of the matrix multiplication, $n-1$ additions and n multiplications are necessary. Since the checksums are calculated by a sum function, $n-1$ further additions are needed for each checksum calculation. To estimate the exact calculation effort, a distinction must be made between several cases.

If no soft errors occur, the selftest terminates after the entire checksum matrix has been calculated. Therefore $(n+1)^2$ matrix elements and $2(n+1)$ checksums must be calculated, resulting in a total of $(n+1)^2 \cdot n$ multiplications and $(n+1)^2 \cdot (n-1) + 2(n+1) \cdot (n-1)$ additions.

However, if soft errors occur, the selftest can terminate earlier. In the best-case, an error in the first row or column, the necessary number of matrix element calculations is reduced to $(n+1)$ and the number of checksums to one. More precisely $(n+1) \cdot n$ multiplications and $(n+1) \cdot (n-1) + (n-1)$ additions are needed.

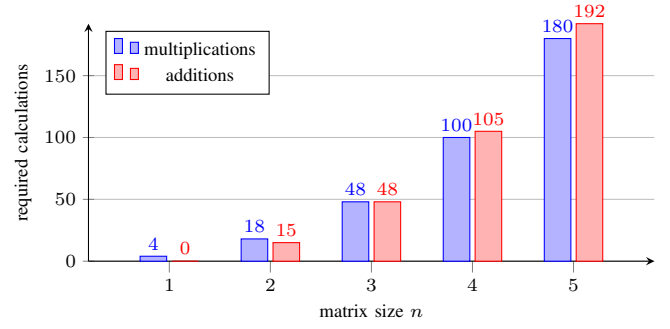


Figure 2: overhead for various n , in case no errors occur.

Since, ideally, the number of soft errors is relatively low and, thus, the matrix element calculations have a complexity of $O(n^3)$, n is the limiting factor for a lightweight calculation and must be kept as small as possible. To estimate the overhead, Figure 2 shows the number of additions and multiplications required for $1 \leq n \leq 5$.

IV. AUTOMATIC TEST EQUIPMENT

To perform an extensive evaluation of the previously described online selftest against classical test procedures, we developed an automatic test equipment that supports different MCUs as Device Under Test (DUT). A big challenge is the injection of failures on COTS MCUs to provoke soft errors. Therefore, the basic idea is to utilize fine grained undervolting to operate the DUTs at the edge of functionality and beyond (cf. Section I). Similar to the secure instance of the implementation in [6], we use a Control Unit (CU) for this purpose. The CU also represents the central component and user interface of the test equipment. A Single Board Computer (SBC) represents the CU, where in our case a Raspberry Pi 3 was used for this purpose. The CU is responsible for the entire test procedure including the adjustment of the voltage level, the generation of the input values as well as the validation of the corresponding output values of the DUT. Moreover it checks whether communication errors occurred and it is able to reset and restart the entire DUT if needed. The CU saves all results in a database for further evaluation. Beside the results of the selftests, the CU collects operating parameters from the DUT such as voltage level, current consumption and ambient temperature. By using a SBC the orchestration of test series is very convenient, even for long-term evaluations.

To support different types of MCUs and to provide a scalable voltage level, the CU is extended by a shield (cf. Figure 3). For the software-controllable supply voltage of the DUT (V_{MCU}) a very low-noise linear regulator TPS79101 [15] is used in conjunction with a high-resolution (10 bit) potentiometer AD5272 [16]. This enables fine grained and linear voltage regulation with a resolution of about 1 mV as a function of the digital potentiometer R_{dig} . For power monitoring, the current sensor INA219 [17] was used where a resolution of less than 0.1 mA can be achieved. Temperature measurement is performed using the temperature sensor LM75 [18] with a resolution of 0.5 °C.

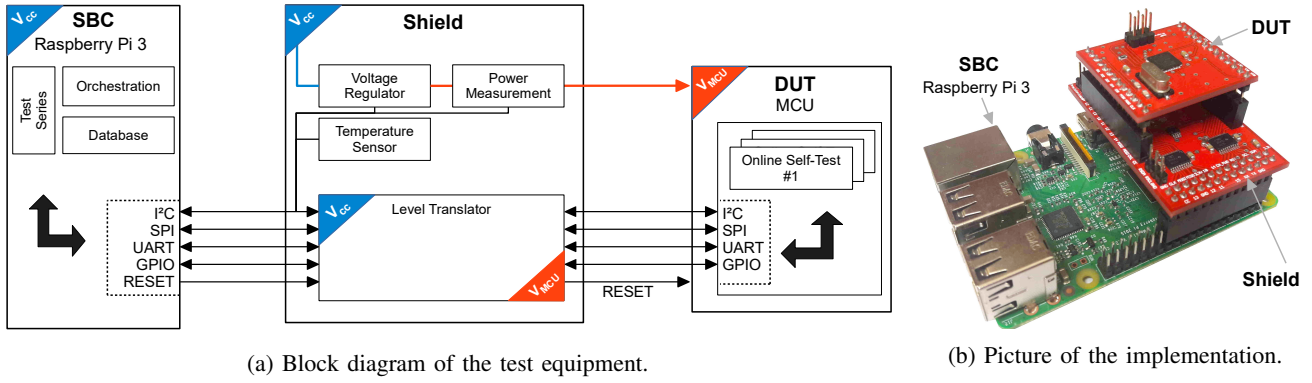


Figure 3: Automatic test equipment to evaluate the online selftest.

To show the accuracy of the automatic test equipment itself, we performed an evaluation to quantify its characteristics. For this purpose we connected a static load resistor $R_{test} = 992.5 \Omega$ to the scalable voltage supply and measured both, the quality of the voltage regulation as well as the accuracy of the power monitoring. The results are summarized in Figure 4. It can be seen that the relative error of the voltage conversion (mean error 0.259 %) as well as the relative error of the power monitoring (0.319 %) are fairly small.

To connect different types of MCUs the shield offers a header with a generic interface including: Voltage supply (V_{MCU}), Serial Peripheral Interface Bus (SPI), Inter-Integrated Circuit (I²C), Universal Asynchronous Receiver/Transmitter (UART) and General Purpose Input/Outputs (GPIOs). Two 8-bit TXB0108 [19] level shifters are inserted since a direct connection between the shield and the specific DUT is not possible due to different voltage levels.

V. EVALUATION

In order to evaluate our selftest, we implemented and tested it on two different types of 8-bit MCUs that are commonly used for WSN nodes. In particular, multiple instances of the Atmel Atmega1284p [20] and the Microchip PIC 18lf27j13 [21] are considered to obtain a meaningful evaluation.

The MCUs were undervolted and the selftest was executed 100 times per each of the $2^{10} = 1024$ voltage steps. The

amount of detected soft errors per voltage step was used to derive the relative detection ratio. Since there is no concrete ground truth about soft errors, the detection rate was used for later evaluation and comparison.

As the individual instances of the same MCU showed very similar results, in this paper we present one instance representative for each MCU.

A. Detection rate

Since the online selftest is a dynamic procedure, various parameters can be changed and adapted to special conditions, even at runtime. For example, when nodes with energy harvesting capabilities are considered, the varying energy budget could necessitate an adaption to lower the overhead. Hence, the impact of different parameters, e.g., the matrix size, on the detection rate of soft errors was investigated first. Therefore, the different MCUs were undervolted and the selftest was executed with changing parameters such as matrix sizes, sequence of calculations and various number ranges. It turned out that the decisive parameter, which affects the computational overhead as well as the detection rate of soft errors, is the used matrix size n . Figure 5a shows the results for the Atmel MCU and Figure 5b for the Microchip counterpart.

For all investigations, all instances of the MCUs showed very similar behavior. When the voltage is continuously reduced, no errors appear for a long time at first, even far below the recommended minimum voltage U_{min} . As soon as the voltage drops below a certain voltage U_{error} (here $U_{error,atmel} = 1.801V$ for the exemplary Atmel MCU and $U_{error,microchip} = 1.378V$ for the exemplary Microchip MCU), the selftest starts to reveal the first errors. The number of detected soft errors increases exponentially below U_{error} .

Due to voltage noise and a small amount of randomness in the occurrence of soft errors, the exact U_{error} and the graph characteristic may vary in small margin for every test run.

Nevertheless, it is already possible to detect errors consistently and repeatably even with a matrix size of $n = 1$. In general it can be seen that the detection rate of soft errors grows with n , which is in evidence for the Atmel MCU (cf. Figure 5a). In addition the sensitivity to reveal soft errors is also higher with bigger n which allows an earlier detection of errors.

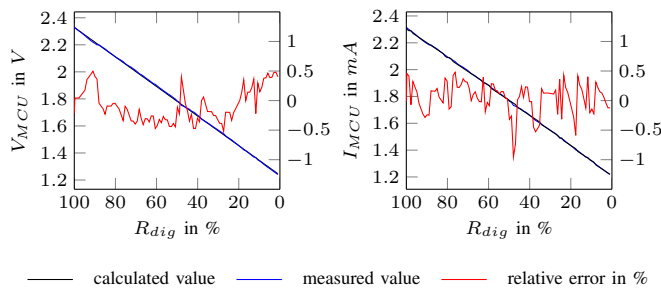
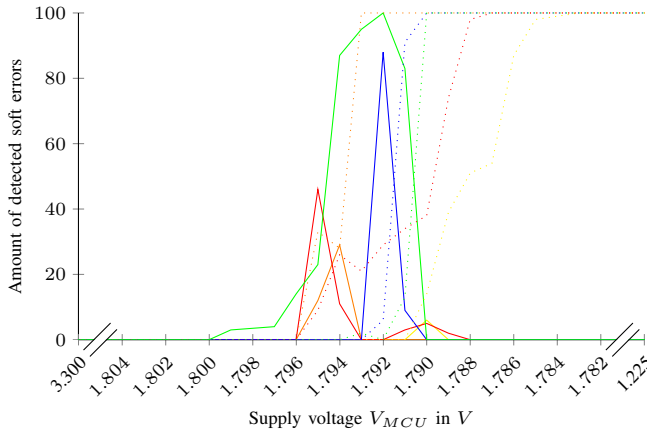
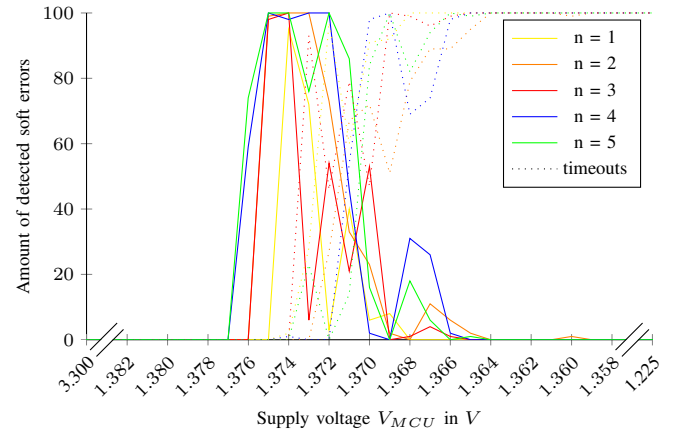
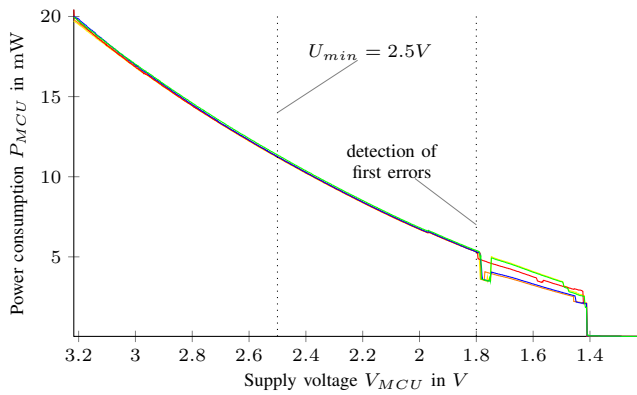
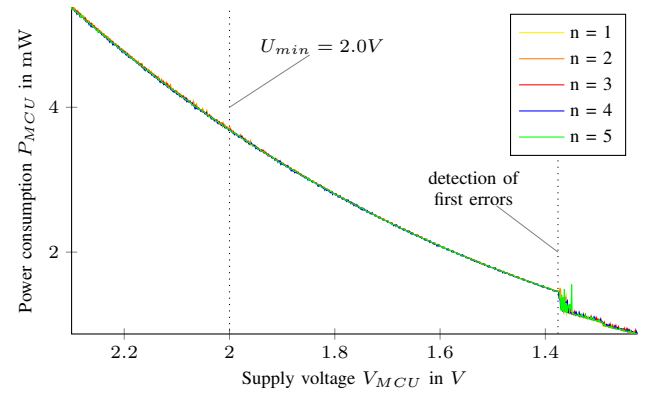


Figure 4: Evaluation of the characteristics of the automatic test equipment.

(a) Atmel ATmega 1284p, $f_{CPU} = 8\text{MHz}$, temperature = 20°C (b) Microchip PIC 18f27j13, $f_{CPU} = 8\text{MHz}$, temperature = 20°C .Figure 5: Detection rate of our selftest with different matrix sizes n on different MCUs(a) Atmel ATmega 1284p, $f_{CPU} = 8\text{MHz}$, temperature = 20°C .(b) Microchip PIC 18f27j13, $f_{CPU} = 8\text{MHz}$, temperature = 20°C .Figure 6: Power consumption of the MCUs during our selftest for different matrix sizes n

Besides the soft errors, also isolated resets of the MCUs and timeouts can be observed. When the undervolting proceeds, the percentage of MCU timeouts increases sharply until it reaches 100% and a communication with the MCU is no longer possible. This non-functional state remains in all cases until the minimum voltage $V_{MCU,min} = 1.225\text{V}$ of the system is reached.

This behavior can also be recognized in the power consumption of the MCUs. Figure 6 shows characteristics in the power curves of the MCUs when the voltage level is lowered. As expected, the power consumption drops in a square of the supply voltage V_{MCU} . But it is remarkable that inconsistencies and unpredictable behavior shows up as soon as the first soft errors occur. If V_{MCU} drops even further, the power consumption also flattens further until it finally becomes 0.

Furthermore, it can be seen that the matrix size n has no visible influence on the power consumption of the MCU. Indirectly, however, the matrix size has quite an influence on the overall energy requirement $E_{selftest} = P_{selftest} \cdot t_{selftest}$ of the procedure since its execution time $t_{selftest}$ also increases with n (cf. Section V-B2). For the following considerations,

therefore, n is set to $n = 5$, as this showed the best compromise between high error detection and acceptable overhead.

B. Comparison with other methods

In order to fully evaluate the selftest, it was compared with known methods for soft error detection in terms of detection rate and overhead. Due to their popularity, Software-based TMR and a Class B test (as defined by IEC 60730 standard) were considered for this comparison.

The basis of the software-based TMR is a redundantly executed function. The following arbitrary function $f(x)$ was selected to implement the method:

$$f_i(x) = x^2 + 2 \cdot x + \frac{x}{2} \quad (14)$$

$$f_1(x) = f_2(x) = f_3(x) \quad (15)$$

The function is called $i = 3$ times in a row with the same parameter x . The results are then compared according to Equation 15. If the outputs do not match, a soft error is assumed. However, it should be mentioned that the voter which

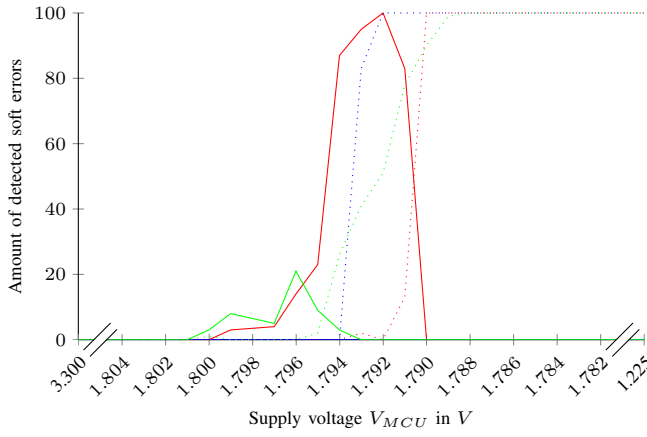
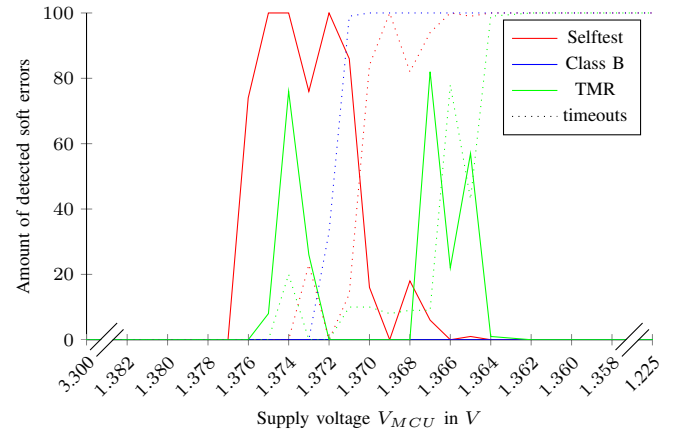
(a) Atmel ATmega 1284p, $f_{CPU} = 8MHz$, temperature = $20^{\circ}C$ (b) Microchip Pic 18f27j13, $f_{CPU} = 8MHz$, temperature = $20^{\circ}C$

Figure 7: Comparing the detection rate of our selftest with other methods

matches the specific outputs is a single point of failure and could also be affected by soft errors.

Since the complete implementation of a Class B test, like presented in [22], is very platform-dependent and the overhead is extremely high, an adjustment was made for this evaluation. To meet the requirement of genericity and low overhead, it was limited to the Static Random-Access memory (SRAM) only. The actual implementation iterates through the entire SRAM of the MCU while each byte is read and buffered. Then each bit of the byte in the SRAM is overwritten with a logical 0 and 1. The MCU then checks whether the write and read processes have been performed correctly. If any discrepancies occurred, a soft error is assumed.

1) *Detection rate*: The first comparative evaluation regarding the detection rate revealed the results shown in Figure 7.

The adapted Class B test was not able to detect soft errors in any case, neither on the Atmel nor on the Microchip MCUs. Thus, it can be assumed that the SRAM on the tested MCUs is not or only hardly affected by soft errors and this method is, therefore, not suitable for detecting soft errors.

On the contrary, the TMR procedure was able to detect soft errors on any instances of the MCUs. The detection rate was rather low with about 40% on the Atmel MCUs and about 80% on the Microchip MCUs. Hence, the TMR method is basically suitable for the detection of soft errors, but the detection rate has shown an inconstant behaviour under different MCUs.

As expected our online selftest detected soft errors on any instance of the tested MCU reliably. Furthermore, the detection rates are significantly higher compared to the TMR, reaching up to a maximum of about 100% on any MCU. In sum, our approach does not show a device specific behaviour and offers highest detection rate of soft errors with the selected configuration ($n = 5$).

2) *Overhead*: To compare the efficiency of the methods, their overhead was determined. For this purpose, their execution time $t_{execution}$, additional energy requirement E_{method} and memory usage were measured. These measurements were done with the Atmel MCUs exemplarily and are representative

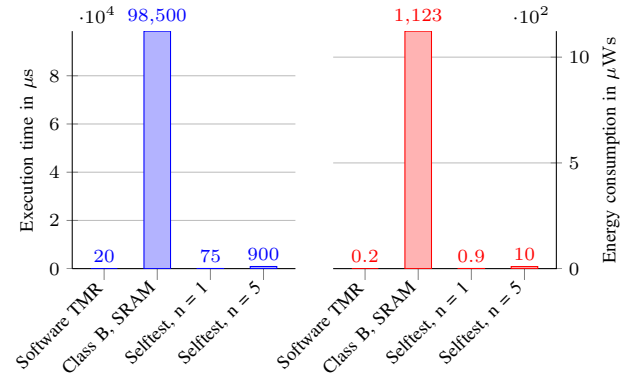


Figure 8: Computational overhead of one iteration.

for similar 8bit Reduced Instruction Set Computer (RISC) MCUs. To gather $t_{execution}$, the state of a pin was changed before and after the execution of a procedure. This time span was then measured using a digital oscilloscope. The additional energy requirement $E_{overhead}$ can be calculated on the basis of the execution time according: $E_{overhead} = P_{MCU}(U_{min}) \cdot t_{execution}$ where $P_{MCU}(U_{min})$ is available from the previous evaluation (cf. Section V-A).

The memory overhead has been divided into program memory and data memory. Since our approach is intended to be generic and universally applicable, the procedures were implemented in the programming language C. The actual memory requirements are therefore strongly dependent on the compiler and the settings used. In our case, the `avr-gcc` compiler was used with its standard settings and compilation flags. Of course, we took care that no necessary variables were optimized away.

Figure 8 shows the computational overhead for the different types of selftests while Figure 9 depicts the memory footprint.

It can be seen that the memory requirements of the Class B test with 416 bytes of program memory and only 4 bytes of data memory are relatively small. However, this method

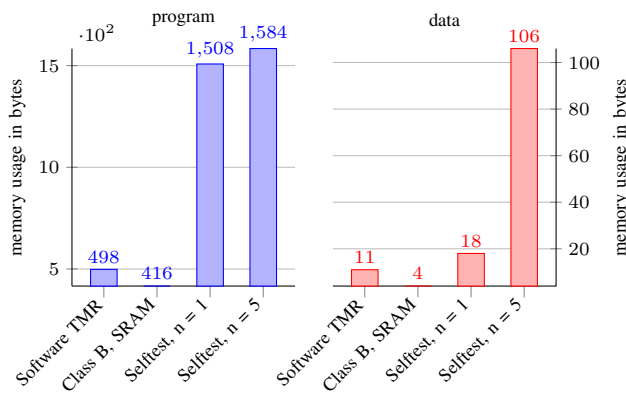


Figure 9: Memory requirements of different methods.

consumes by far the most power and loads the MCUs with an execution time of 98.5ms. The energy requirement of the Class B test is therefore disproportionately high at about 1.1 mWs per iteration without detecting any soft error.

The software-based TMR has, due to the choice of a relatively simple function $f(x)$, a low overhead. In case of memory consumption, this method is midfield with 498 bytes for program and 11 bytes for data. The execution time is rather short with 20 μ s, resulting in a low energy requirement of 0.2 μ Ws which qualify this method to be used on energy constrained MCUs.

The online selftest has an increased memory load of 1584 bytes of program memory and 106 bytes of data memory compared to the other methods. In terms of power consumption and execution time (900 μ s per iteration), this method is still very efficient. At 10 μ Ws, the energy consumption of the online selftest is quite higher than the software-based TMR but still well suitable for low-power devices used on WSN nodes. Moreover, considering the much better failure detection characteristics the slightly higher energy consumption seem to be easily acceptable.

VI. CONCLUSION

In this paper we presented an effective but lightweight approach to detect soft errors on energy-constrained WSN nodes. Beside the presentation and discussion of our online selftest, we implemented an automatic test equipment to perform an extensive test series on different types of MCU. We have shown that our online selftest offers consistently high detection rates of soft errors independent from the specific MCU. Moreover, the evaluation against other test methods showed that our approach offered the highest probability to detect failures. In sum, our selftest clearly represents the best compromise between energy requirement and detection rate of soft errors on various MCUs. Thus, the selftest is well suitable for energy constrained MCUs of WSN nodes.

ACKNOWLEDGMENT

This research was partially supported by the German Research Council (DFG) under the grant no. BU 3282/2-1.

REFERENCES

- [1] R. Ginossar, "Survey of processors for space," *Data Systems in Aerospace (DASIA)*, Eurospace, pp. 1–5, 2012.
- [2] R. Hammett, "Design by extrapolation: an evaluation of fault tolerant avionics," *IEEE aerospace and electronic systems magazine*, vol. 17, no. 4, pp. 17–25, 2002.
- [3] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant platforms for automotive safety-critical applications," in *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*. ACM, 2003, pp. 170–177.
- [4] Z. Latifi and A. Karimi, "A tmr genetic voting algorithm for fault-tolerant medical robot," *Procedia Computer Science*, vol. 42, pp. 301–307, 2014.
- [5] Holger Karl and Andreas Willig, *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.
- [6] U. Kulau, F. Büsching, and L. Wolf, "IdealVolting – Reliable Undervolting on Wireless Sensor Nodes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 2, April 2016.
- [7] U. Kulau, S. Müller, S. Schildt, A. Martens, F. Büsching, and L. Wolf, "Energy Efficiency Impact of Transient Node Failures when using RPL," in *Proceedings of the 18th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, ser. WoWMoM 2017, 2017.
- [8] M. Mousa and C. Claudel, "Water level estimation in urban ultrasonic/passive infrared flash flood sensor networks using supervised learning," in *Proceedings of the 13th international symposium on Information Processing in Sensor Networks*, (IPSN 2014), Poster Abstract. IEEE, 2014.
- [9] A. H. Dehwah, M. Mousa, and C. G. Claudel, "Lessons learned on solar powered wireless sensor network deployments in urban, desert environments," *Ad Hoc Networks*, vol. 28, 2015.
- [10] U. Kulau, F. Büsching, and L. Wolf, "Undervolting in WSNs: Theory and Practice," *Internet of Things Journal, IEEE*, vol. 2, no. 3, pp. 190–198, June 2015. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6991505>
- [11] K.-H. Huang *et al.*, "Algorithm-based fault tolerance for matrix operations," *IEEE transactions on computers*, vol. 100, no. 6, pp. 518–528, 1984.
- [12] A. Eghbal, H. R. Zarandi, and P. M. Yaghini, "Fault tolerance assessment of pic microcontroller based on fault injection," in *Test Workshop, 2009. LATW'09. 10th Latin American*. IEEE, 2009, pp. 1–6.
- [13] A. Rohani and H.-R. Zarandi, "An Analysis of Fault Effects and Propagations in AVR Microcontroller ATmega103(L)," in *IEEE International Conference on Availability, Reliability and Security*, ser. ARES 2009, 2009, pp. 166–172.
- [14] P. Wu, C. Ding, L. Chen, F. Gao, T. Davies, C. Karlsson, and Z. Chen, "Fault tolerant matrix-matrix multiplication: correcting soft errors online," in *Proceedings of the second workshop on Scalable algorithms for large-scale systems*. ACM, 2011, pp. 25–28.
- [15] Texas Instruments, "Datasheet: Tps79101-ep, ultralow noise, high psrr, fast rf, 100-ma low-dropout linear regulators," 2008. [Online]. Available: <http://www.ti.com/lit/ds/slvs325c/slvs325c.pdf>
- [16] Analog Devices, "Datasheet: Ad5272, 1024-/256-position, 1% resistor tolerance error, i2c interface and 50-tp memory digital rheostat," 2013. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD5272_5274.pdf
- [17] Texas Instruments, "Datasheet: Ina219, zero-drift, bidirectional current/power monitor with i2c interface," 2015. [Online]. Available: <http://www.ti.com/lit/ds/symlink/ina219.pdf>
- [18] Maxim Integrated, "Datasheet: Lm75, digital temperature sensor and thermal watchdog with 2-wire interface," 2009. [Online]. Available: <https://datasheets.maximintegrated.com/en/ds/LM75.pdf>
- [19] Texas Instruments, "Datasheet: Txb0108, 8-bit bidirectional voltage-level translator with auto-direction sensing and ± 15 -kv esd protection," 2014. [Online]. Available: <http://www.ti.com/lit/ds/symlink/txb0108.pdf>
- [20] Atmel, "Datasheet: Atmega1284p, 8-bit microcontroller with 128k bytes in-system programmable flash," 2009. [Online]. Available: <http://www.atmel.com/images/doc8059.pdf>
- [21] Microchip, "Datasheet: Pic18f27j13, pic18f47j13 family 28/44-pin, high-performance microcontrollers with xlp technology," 2016. [Online]. Available: <http://ww1.microchip.com/downloads/en/DeviceDoc/30009974C.pdf>
- [22] *Application Note: Atmel AVR1610: Guide to IEC 60730 Class B Compliance with XMEGA*, Atmel, 2013. [Online]. Available: <http://www.atmel.com/images/doc42008.pdf>