Duplicated Control Unit Based Embedded Fault-masking Systems

György Györök, Bertalan Beszédes
Óbuda University/ Alba Regia Technical Faculty, Székesfehérvár, Hungary
e-mail: {gyorok.gyorgy, bertalan.beszedes}@amk.uni-obuda.hu

***Abstract*— Fault-masking architectures are classified into a few major categories. The first is the multiplication of the microcontroller, the other is a CON-MON architecture (not a full-fledged fault-masking system), there is of course, the multiplication of frequently failing units. In this article, the focus is on the different kind of solutions, how can a duplicated microcontroller based system, monitoring itself, and increasing the fault-tolerance level of the embedded system.**

I. INTRODUCTION

A control unit – as any other physical component (typically, the power supply unit [1]) [2] – can suffer a malfunction, and the consequences can range from discomfort to disaster. For example, an error in a phone line controlling unit can cause temporary line dripping, while an error in a transport vehicle's controlling unit can cause a serious accident. For this reason, the reliability of computers, is an important viewpoint during the development.

In a fault-tolerant system, which contains, redundant – for example, multiplicated – modules, to increase the fault tolerance level, need to contain a supervisor control. This can be a microcontroller, but this element should be more reliable, than the supervised modules – it need to be completely fault-tolerant, also called as hard-core. Let us see a few methods, how to ensure this feature.

II. THEORY

A. Fault-tolerant solutions

At the beginning of the fault tolerance history, the passive physical hardware solutions were the limit of a redundant system. The most common solution was to multiply the physical parts of the device, and hence increase the fault tolerance level.

The OAO (Orbiting Astronomical Observatory) satellites and controlling units of the early Apollo program was one of the last computers which had been built by discrete transistors. All of the units, four serially parallel connected transistors were used instead of one, to mask the fault if one transistor could cause if it is get out of order. [3] (OAO's are NASA satellites, they are made observations in UV range mainly at the 60s. With this, they are laid the basis of the astronomical observation in space, so that they are regarded as the predecessor of the Hubble Space Telescope.) Nowadays, there are more sophisticated solutions. A redundant system has added resources against a simple system.

Hardware redundancy is when extra hardware is added to the system, it is typically used for fault detection and in fault tolerant solutions. For example, multiplicated modules where each type of module has their well delimited task.

Software redundancy means that, the added extra software routines, giving the possibilities to detect the faults, next to the default functions of the original software. If it is possible, they should fix the faults. For example, timeout monitoring assigned to waitings.

Information redundancy is the extra information what is used to fault detection or fault correction, which would not be absolutely necessary for the default functions of the device. If it is possible, the added function should fix the faults. For example, using error detecting- or error correcting bits.

Time redundancy is the extra time what is used for fault detection or a fault tolerant feature. For example, running identical calculations multiple times and checking the consistency of the outcome.

From the above-stated it seems, depending on what kind of redundancy had been used, it has significantly impact to system performance, required power, weight, price and reliability. It is important to review the various methods to assess – the perspective of – the possibilities how to increase the reliability.

B. Hardware redundancy

Hardware redundancy is the most common form of redundancy. Electronic components become increasingly smaller and cheaper, therefore it is more and more acceptable compromise the use of hardware redundancy. [4]

Hardware redundancies can be divided into three groups:

- During passive redundancy is meant the procedures, which do not require intervention from the operator or the system, in case of fault, it is simply masking the faults.
- In case of active redundancy, it is used fault detection and fault correction methods, to increase reliability. The latter term is often referred to reconfiguration, and it meant the removal of the faulty hardware or an application with adaptive methods. The system is tries to adapt to the changed circumstances, so as to keep up or to improve the operation.
- The hybrid redundancy is combines the advantages of the two method. Hybrid redundancy is using commonly the tools of error

masking, error detecting and error recovery, to increase the fault tolerance feature of the system.

III. HARDWARE REDUNDANCY

A. *Triplication and voting system*

For mission-critical systems, like automotive, avionics, railway-signaling controllers, medical devices and nuclear plant systems, a failure may be life-threatening [5][6]. These systems have multiple controlling unites (three or more – running parallelly) and use a complex, majority logic based voting system, to implement the fault-tolerant system. [7][8]

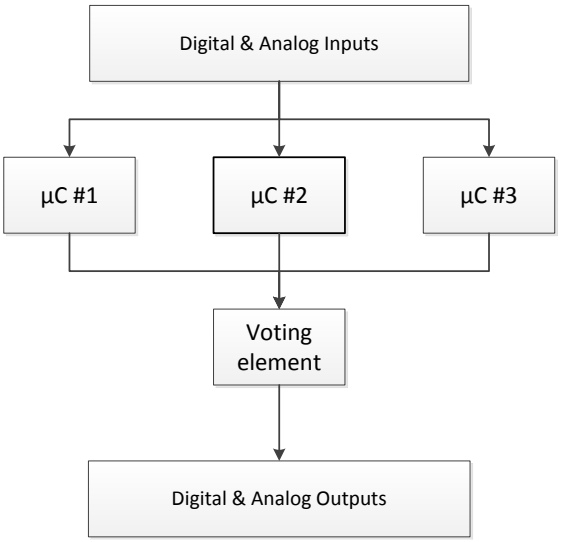

Figure 1. TMR architecture

An implementation of this kind of static hardware redundancy is the TMR (Triple Modular Redundancy) [9]. The system also contains a majority voting element, able to detect errors (Fig. 1.). If there is a difference between the output of controllers, the voting element will choose the two matching result, as a correct outcome, and mark the different controller as unreliable. [10]

Majority voters are critical parts of a redundant system [11]. A simple voting circuit is can be formed by four logic gates, – see Figure 2. – where, any two inputs are in high logic level, the output will become logical '1', and any two inputs are in low logic level, the output will become logical '0' – independently from the third input's logic value.

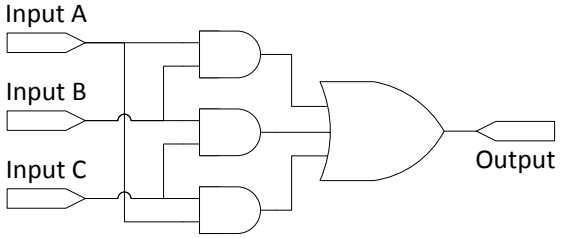

Figure 2. One-bit voter built by logic gates

This solution is reliable because of its simplicity and it is also easy to use. With parallel circuits, the voter's word-width can be extended, but if the input signals are not arriving at the same time, it can cause a false output, so it need to handle as a synchronous network. If in the operation area have high background radiation, the logic can be made by simple switching devices, like transistors, MOSFETs – called as RadHard (Radiation Hardened) solution. In the case of analogue signals, the logic can be made by using an operational amplifier in comparator mode. It could be a good solution, where the voting need to be done of multiple analogue sensors.

B. *CON-MON architecture*

The previous solution is an expensive architecture, but there is a cheaper way, which could be very useful and well usable for most areas, for a much lower price, the CON-MON architecture.

This is one of the unique architectures, frequently used in automotive, avionics and other systems. The name CON-MON stands for control-and-monitor microcontroller architecture. This architecture is not as fault-tolerant as the duplicated controller architecture, but it can make an alarm, when the main uC fails. It uses two uCs — the main controller, which fully handle the function of the system, and a small microcontroller, which monitors the main controller through a serial interface, especially when the WDT overflows (Fig. 3.).

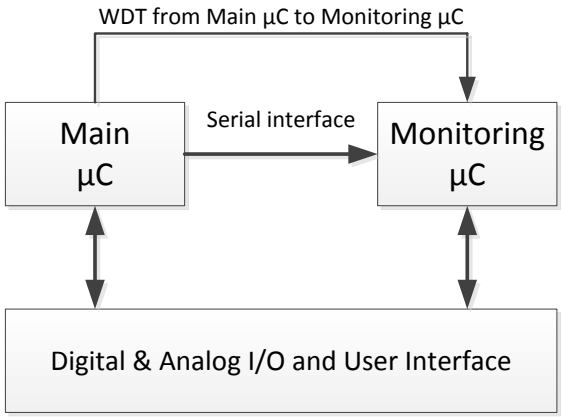

Figure 3. CON-MON architecture

The main advantage of this architecture is, to let developers and maintainers know, what circumstances the fault is generated. Imagine that the small microcontroller – the logging microcontroller – is not part of the system. When the main uC restarts due to a fault, the details about the environment, like the exact time, I/O states, system status, the actual subroutine, etc., are lost, and failure is known only when the main uC stops working again. Otherwise, in the CON-MON architecture, the smaller controller will log this data, apart from raising the alarm.

C. *Duplicated control unit based fault-tolerant systems*

When the system has to be completely fault-tolerant, the Test and Intervener Controller still need to be redundant. In the duplicated controller architecture, microcontrollers are duplicated, so if one microcontroller

fails, the other microcontroller takes over. Duplicated controller based fault-tolerant systems are based on a combination of hardware and software. [12]

The fault-tolerant mechanism works on two essential features that the controllers have. The one is the watchdog timer and the second is a high-speed serial link between two microcontrollers. Fig. 4. shows how duplicated microcontroller based fault-tolerant systems are implemented. [13]

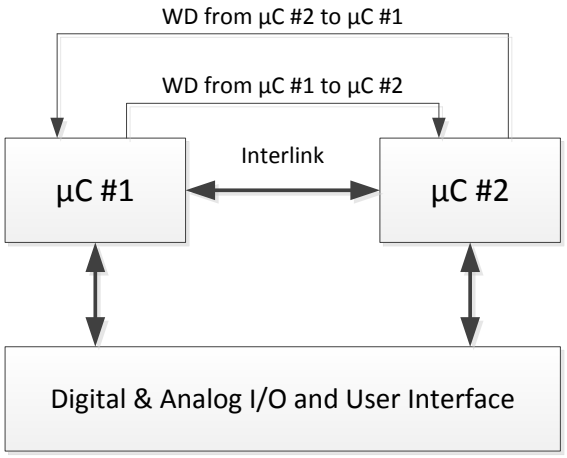

Figure 4. How duplicated uC based FT systems are implemented

One of the simplest cases of the self-diagnostic feature, is the watchdog timer (WDT). It does not require significant hardware and modern processors are including in an integrated form. It is a great advantage of the WDT, that it can be successfully used against both software and hardware failures.

The WDT The watchdog timer is usually much simpler as the units that is monitored with it, therefore it is more reliable as well.

The watchdog timer is an electronic hardware timer, that is used to detect controller malfunctions. During normal operation, the uC regularly restarts the WDT to prevent it from timing-out. If – due to a hardware fault or a program error – the controller fails to clear the content of the watchdog timer (which is incrementing automatically), the timer will overflow and will generate a time-out signal and an interrupt to initiate corrective action.

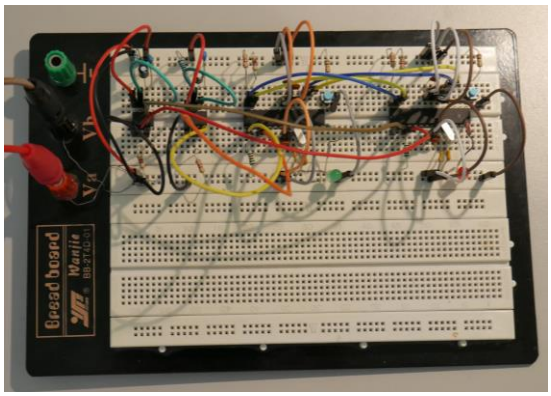

Figure 7. The model of the circuit

The watchdog timer does not detect all the problems, only confirms the unit's "viability". Therefore, in combination with other solutions. [14]

In applying the watchdog, the counter resetting instructions should be placed in the main cycle. And the length of the counting period should always be adapted to the specific characteristics of the application, for example: response time, tasks running time.

In this case, there are a few aspects that need to be know, to be fulfilled:

- It takes time for the good controller to take over the control from the faulty one (known as switch-over time)
- The system data and the user data need to be consistent between of the two controllers (data integrity)
- It need to provide a common bus for the two controllers (redundant controller bus interface)
- A built-in diagnostic, to identify and isolate problems in the system (built-in self-test)

Let us see how the system is implemented. Meanwhile, the faulty controller after a watchdog timer overflow, restarts and runs self-diagnostics to identify whether the problem is related to hardware or software. If the problem connected with hardware, it generates the error code and notify the supervisor system. Meanwhile, the redundant controller will take over the control and runs the system as usual, so that the main functionality of the system does not suffer from a shortage, as Figure 5 and Figure 6 shows.

It is the system software's responsibility to maintain the log [15], after the faulty unit has been replaced by a maintenance man with a good one and the system has returned to duplex mode. [16]

At this stage, it need to mention that dual uC implementation has two variations in their working, based on software implementation.

D. Swapping microcontrollers

At the first case, the system is fully-controlled by one of the uC and the other uC takes over the control when the active one fails. This is known as hot-stand-by architecture. At this solution, the software is simple. There is two critical part, that need to be handled. One is the take-over part of the software, the other is the notification of the supervisor system or the user. In both cases, last task is logging and updating the data.

E. Parallel tracking

The second case is a more complex in software, and a more precise solution, the parallel tracking. In this implementation, both uCs parallel execute the tasks in the system, but only the active uC controls the system. The status of the system is almost identical. This architecture helps in mission-critical systems for faster take-over.

Of course, there is the possibility that two of the controllers make the same mistake, and only one gives the correct result, but for this, the probability is much lower, than the previous mentioned case. The only case that causes a problem if two of the three controller makes a fault, and these ones are different. At this situation the voting element, cannot decide which is the correct one. One solution to secure the system is to use more than three

parallel running controllers, but it will significantly increase the price.

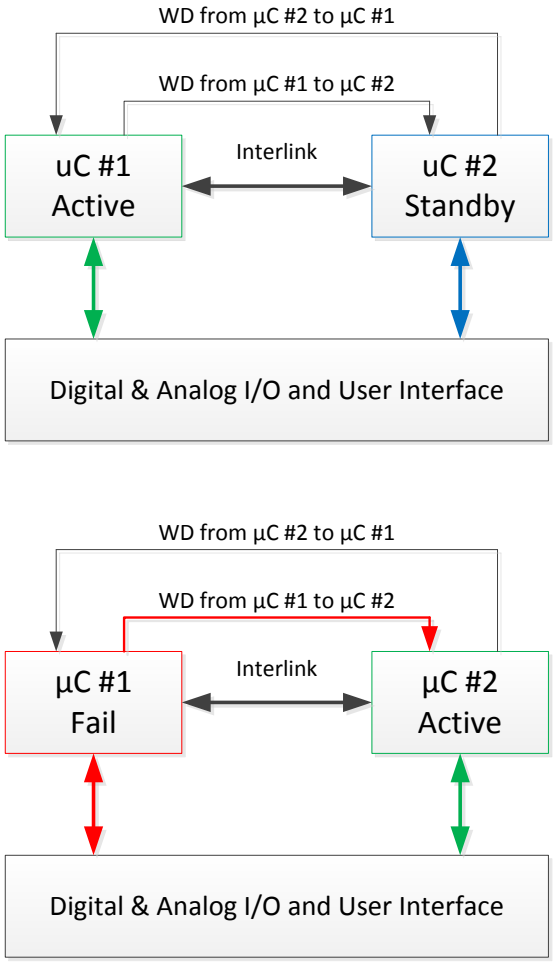

Figure 5. Sequence of events 1st case

Reliability is the probability that the system is operating correctly in a defined period [t_0; t] – if the system is operating correctly in t_0. In practice, it is estimating the chance, that the unit is functioning well, during that time. The failure willingness F_t, the reliability R_t, these are complementary events, i.e. the sum of these two is always one:

$$R_t + F_t = 1, \tag{1}$$

$$F_t = 1 - R_t, \tag{2}$$

$$R_{duplex} = R^2 + 2R(1-R), \tag{3}$$

$$R_{TMR} = \left[R^3 + 3R^2(1-R)\right]R_V, \tag{4}$$

where, R_V is the reliability of the voting element.

The reliability of the voting element is crucially important for the proper functioning, it need to be especially reliable – that is why so called hard core. This solution is well protects against static and permanent errors, but it is expensive.

IV. REALIZATION

The realization (Fig. 7.) is approaching the hard core solution, the block diagram is shown in Fig 4. The controllers are swappable, so the code is need to be independent from the devices. At the time only one of the controller – the main controller – supervises the system, the safety one's task are to follow the main controllers program, recalculate the actual subroutine's results, comparing the results, sending back the compared results to the main controller, logging and take over the control if necessary. To take over the control, the triggering event is the watchdog overflow signal.

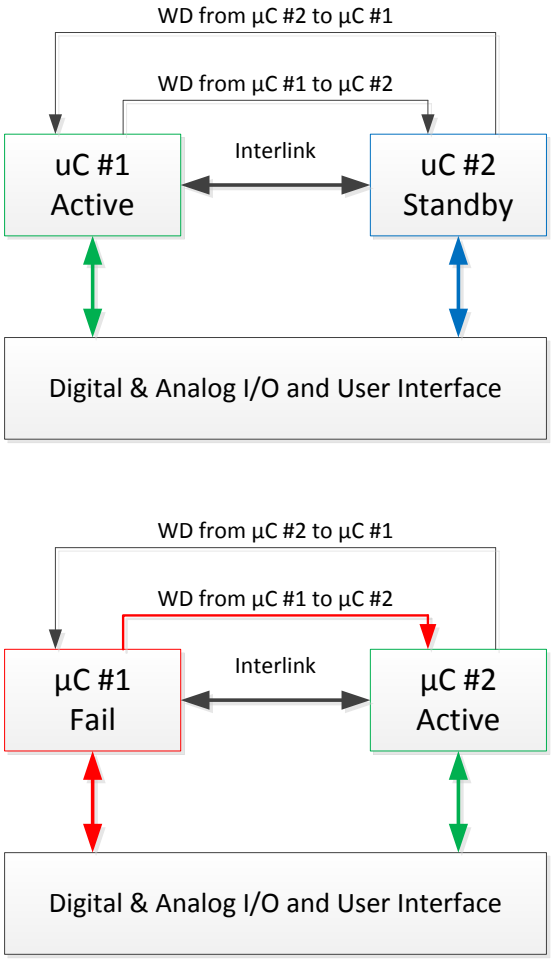

Figure 6. Sequence of events 2nd case

The microcontroller's program is separated to subroutines. Each subroutine starts with the question "Am I the Master controller?", and continuing depends of the answer. The master always in charge, he is who starting the communication with the Slave, the Slave need to wait with his program till the Master asks the results or giving an order.

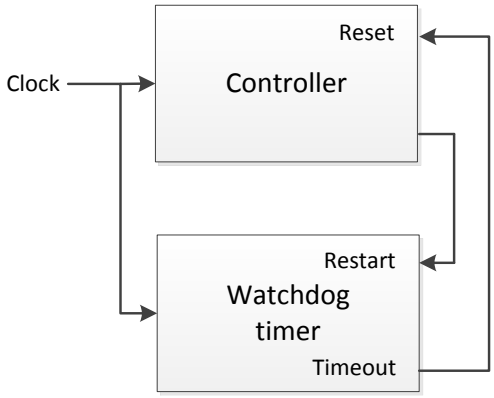

Figure 11. External watchdog timer architecture

In every subroutine, in the end – after the main function of the subroutine, when the results available –, there is the comparing function (Fig. 8.). If the results of the controllers are not matching, they need to recalculate the subroutine from the beginning. They can do this multiple time, until the results are matching. Beside logging the events, it is possible to decide, which controller is more reliable. If, the results are not matching, after multiple recalculation, the correct result will be the more reliable controller's result – and a supervisor system need to be notified (also if, there is a lot of mismatching result). If the controllers are equally reliable, the program is continuing with the results of the controller who is actually the master.

The microcontrollers have got two hardware I2C peripheries, the hard core interlink connection is implemented with these two separated channel. I2C is a

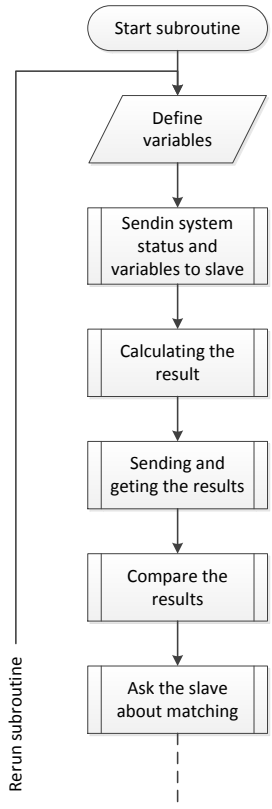

Figure 8. Subroutine validating

Master-Slave based protocol, the algorithm is built on the basis of criteria, the flow chart is shown in Fig. 9.

In the communication, every time, the controllers are sending the messages three times. If the three identical message is not the same, the receiver asks the other side to resend the message. It is also using an error detecting code under the I2C protocol, if the receiver found an error in the message, it asks the other side to resend the message, the flow chart is shown in Fig. 10. This method is executed on both channels, the sent messages are compared to each other, and if they are matching, then it can be acceptable as a correct message – it can be a data or a command.

For the definition of Wto, it is the maximum amount of time, that the watchdog timer can count before it needs to be reset [17]. Most of the microcontroller models, the Wto is not long enough for the tasks. In practice, often have serial processes that run longer than the maximum of the Wto. It is possible to use incorporate patting into the code, but when using external libraries, it could be cumbersome. Figuring out all of the possibilities and putting *wdt_reset()* calls in the right spot is difficult and with some serial routines, impossible, so to use an internal watchdog timer is problematic.

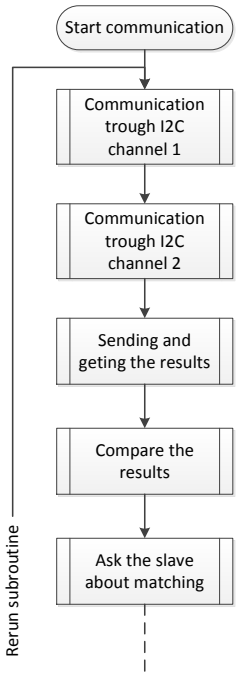

Figure 9. Communication with I2C channels

External watchdog timer is an independent timer that is separate from the controller entirely, as Fig. 11. shows. An external watchdog timer has a much higher reliability, than an internal one. There is no way that the controller's internal software, shut off an external watchdog timer from doing its job, but an internal watchdog timer can be easily shut off by software [18]. (Of course, if the external WDT need to be shut off via software, it could be done using a GPIO pin to control a transistor, but generally, it is not needed.)

In brownout conditions an internal WDT cannot work correctly, an external WDT can keep hitting the device until it does come back or the power levels are less brown. With an external WDT also can stretch out the Wto to a

lot more than 16 seconds, to cover all the possible – for example bootup – sequences.

Watchdog timers can improve the reliability of an embedded system [20]. Internal watchdogs have limitations, but can still improve the system's reliability. As a better solution an external watchdog timer can handle well the trouble-maker issues, such as brownouts, power loss, coding errors and radio frequency interferences.

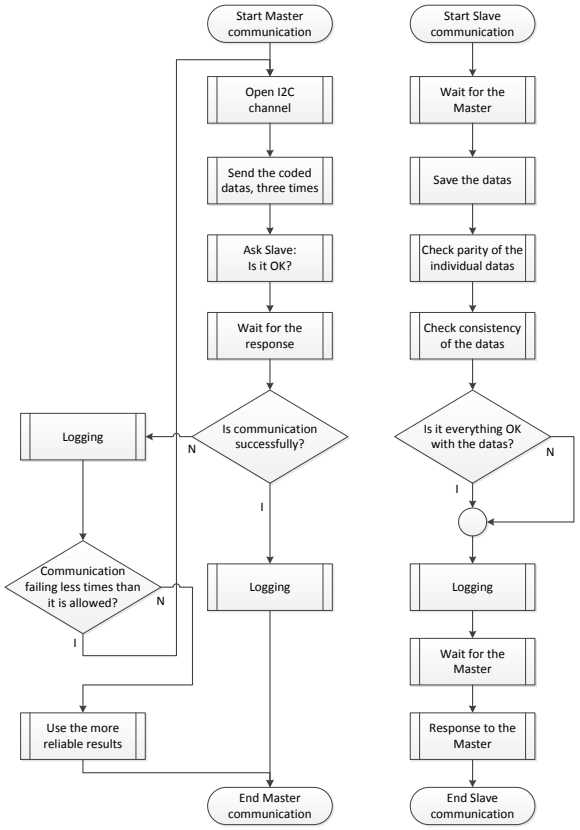

Figure 10. I2C communication

CONCLUSION

In this paper, is showed a realization of a redundant microcontroller based system, with different reliable levels. By the mentioned solutions, the error-free running time and error detecting features can be increased, as showed in the implementation of the system. We believe that the presented methods can be used in several applications.

REFERENCES

[1] Gy. Györök. A-class amplifier with FPAA as a predictive supply voltage control. In: 9th International Symposium of Hungarian Researcherson Computational Intelligence and Informatics (CINTI2008). 2008. 361–368. p.

[2] Györök György, Bertalan Beszedes Fault tolerant power supply systems In: Orosz Gábor Tamás 11th International Symposium on Applied Informatics and Related Areas (AIS 2016). Konferencia helye, ideje: Székesfehérvár, Magyarország, Budapest: Óbudai Egyetem, 2016. 68-73. pp. (ISBN:978-615-5460-92-0)

[3] David A. Rennels. Fault-tolerant Computing – Concepts and Examples. In: IEEE Transactions on Computers. December 1984. 1116-1129 pp.

[4] Bray W. Johnson. Design and Analysis of Fault-Tolerant Didital Systems. 1989. Addison-Wesley Publishing

[5] Gy Györök, M Seebauer, T Orosz, M Makó, A Selmeci Multiprocessor Application in Embedded Control System In: Szakál A (szerk.) 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012, Subotica, 2012, September, 20-22. Konferencia helye, ideje: Subotica, Szerbia, 2012.09.20-2012.09.22. Piscataway: IEEE, 2012. pp. 305-309. (ISBN:978-1-4673-4751-8)

[6] Gy Györök, T Orosz, M Makó, T Treiber To Achieve Circuit Robustness by Co-operation of FPAA and Embedded Microcontroller In: Szakál Anikó (szerk.) IEEE 8th International Symposium on Applied Computational Intelligence and Informatics: SACI 2013. Konferencia helye, ideje: Timisoara, Románia, 2013.05.23-2013.05.25. (IEEE) New York: IEEE, 2013. pp. 315-320. (ISBN:978-1-4673-6397-6)

[7] Gy. Györök. Embedded hybrid controller with programmable analog circuit. In: Intelligent Engineering Systems (INES), 2010 14th International Conference on. IEEE, 2010.

[8] Gy. Györök. The FPAA realization of analog robust electronic circuit. In: Computational Cybernetics, 2009. ICCC 2009. IEEE International Conference on. IEEE, 2009.

[9] Subhasish Mitra, Edward J. McCluskey. WORD VOTER: A New Voter Design for Triple Modular Redundant Systems. In: VLSI Test Symposium. 465-470 pp. 2000.

[10] Behrooz Parhami. Voting Algorithms. In: IEEE Transactions on Reliability. Vol. 43. No. 4. 1994. December

[11] Triple Module Redundancy Design Techniques for Virtex FPGAs. In: Xilinx Corporation Application Notes. 2006. http://www.xilinx.com/bvdocs/appnotes/xapp197.pdf

[12] Gy Györök Embedded hybrid controller with programmable analog circuit In: Szakál A (szerk.) 14th International Conference on Intelligent Engineering Systems: Proceedings. Konferencia helye, ideje: Las Palmas, Spanyolország, 2010.05.05-2010.05.07. Budapest: IEEE Hungary Section, 2010. pp. 1-4. (ISBN:978-1-4244-7651-0)

[13] Gy. Györök. Self configuration analog circuit by FPAA. In: 4th Slovakien – Hungarien Joint Symposium on Applied Machine Intelligence (SAMI2006), 2006. 34–37. p.

[14] Behrooz Parhami. Hardware Design Methodes. In: Fault-Tolerant Computing UCSB egyetemi jegyzet. 2006. http://www.ece.ucsb.edu/Faculty/Parhami/pres_folder/f33-ftcomputing-lec13-hardw.pdf

[15] Gy. Györök, L. Vokorokos, L. Hluchý. Crossbar network for automatic analog circuit synthesis. In: 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2014). IEEE Computational Intelligence Society. Szerk.: J. Fodor. Budapest. 2014. ISBN:978-1-4799-3441-6, 263–267. p.

[16] K. Lamár, J. Neszveda. Average probability of failure of aperiodically operated devices. In: Acta Polytechnica Hungarica, 10.(8.). 2013. 153–167. p.

[17] György Györök, Bertalan Beszédes Artificial Education Process Environment for Embedded Systems In: Orosz Gábor Tamás (szerk.) 9th International Symposium on Applied Informatics and Related Areas - AIS2014. Konferencia helye, ideje: Székesfehérvár, Magyarország, 2014.11.12 Székesfehérvár: Óbudai Egyetem, 2014. pp. 37-42. (ISBN:978-615-5460-21-0)

[18] Gy. Györök, M. Makó. Configuration of EEG input-unit by electric circuit evolution. In: 9th International Conference on Intelligent Engineering Systems (INES2005), 2005. 1–7. p.

[19] Gy. Györök, M. Makó, J. Lakner. Combinatorics at electronic circuit realization in FPAA. In: Acta Polytechnica Hungarica, Journal of Applied Sciences, 2009. 6(1). 151–160. p.

[20] Gy. Györök. The function-controlled input for the IN CIRCUIT equipment. In: 8th Intelligent, Engineering Systems Conference(INES2004), 2006. 443–446. p.