*Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023)*
*Edited by* Mário P. Brito, Terje Aven, Piero Baraldi, Marko Čepin and Enrico Zio
 *Published by* Research Publishing, Singapore.
doi: 10.3850/978-981-18-8071-1_P516-cd

# An all-digital hardware monitoring system at run-time to improve the reliability of electronic devices

Leonardo Fazzini

*Università degli Studi dell'Aquila – DISIM/DEWS, ITALY. E-mail: leonardo.fazzini1@student.univaq.it*

Luigi Pomante

*Università degli Studi dell'Aquila – DISIM/DEWS, ITALY. E-mail: luigi.pomante@univaq.it*

Marco Santic

*Università degli Studi dell'Aquila – DISIM/DEWS, ITALY. E-mail: marco.santic@guest.univaq.it*

This paper describes a use case developed in the context of a European research project. The main goal of such a use case is to show as it has been possible to exploit an all-digital hardware monitoring system at run-time to improve reliability with focus on verification activities. For such a purpose, a pacemaker has been developed in two versions. One version has been based on a Commercial Off-The-Shelf microcontroller, where some properties have been verified by means of a classical offline traces analysis approach. The other version has been based on a soft-core on Field Programmable Gate Array, where the same properties have been verified at run-time by means of the adopted all-digital hardware monitoring system. The comparison of the two verification approaches shows how it is possible (i) to reduce the time needed to perform verification and (ii) to provide the opportunity to verify more complex properties with respect to classical Built-In Self-Test approaches.

*Keywords*: Embedded Systems, Monitoring systems, Reliability, Verification, Microcontroller, FPGA

## 1. Introduction

The *monitoring action* on a system, from a general point of view, provides information about its state. These ones can be processed (e.g., filtered, interpolated) to obtain indications about parameters that can be useful also to consider reliability issues [1]. Referring to information collection, two types of monitoring can be identified: software and hardware.

Software monitoring systems are based on the exploitation of the same processors that are executing the application under exam also to collect data. There are various examples of software monitoring systems, that depend on the application: *Gprof* [2], the GNU statistical profiler, that is a tool able to count the number of times functions have been called and to measure the execution time of the routines. Another software monitoring tool is *Rapitime* [3] that, starting from the instrumentation of the code, takes timestamps of processor during execution of the application. Another branch of application of software monitoring systems is the monitoring of the correct behavior of the application under exam [4]. In general, software monitoring systems are cheap and fast to be integrated in the target system. As main drawback, they necessarily introduce some overheads on execution time, and it has some grade of statistical inaccuracy.

Hardware monitoring system are based on dedicated analog/digital hardware resources able to carry on the profiling action. This means that no source code instrumentation is needed and the software execution by the applicative processor is not altered, thus no overhead on execution time is introduced. For the same reason, hardware monitoring systems can guarantee the best accuracy in performance analysis. Various examples of all-digital (i.e., no analog components involved) hardware-based profiling approaches have been presented in literature. For

example, *SnoopP* [5] and *Airwolf* [6] are two all-digital hardware-based function-level profilers for software applications running on soft-core processors. However, hardware monitoring systems require more area/resources occupation for target system implementation.

The exploitation of all-digital hardware monitoring systems to improve reliability is a well-known practice in both the academic and industrial domains (e.g., [7][8][9]). The most common evaluated metrics are the *switching activity* and the *coverage*. The most common detected faults are related to *stuck-at* and, especially when an FPGA is exploited in the system, to *bit-flip*. In all the situations, the possibility to obtain meaningful information without affecting the nominal behavior of the system is of critical importance, i.e., the effectiveness and efficiency (e.g., non-intrusiveness and low area overhead) of the adopted monitoring system is of critical importance too.

In such a context, this paper describes a use case developed in the scope of the ECSEL IREL40 European research project [10]. The main goal of such a use case is to exploit an all-digital hardware monitoring system at run-time [11][12][13] to collect data related to system behavior, and to finally improve reliability with focus on verification activities. For such a purpose, a pacemaker has been developed in two versions: one based on a *Common Off-The-Shelf* (COTS) microcontroller on a *Printed Circuit Board* (PCB), where some properties are verified by means of a classical offline approach (i.e., script-based analysis of traces collected by means of a logical analyzer), and one based on a soft-core *General-Purpose Processor* (GPP) on a *Field Programmable Gate Array* (FPGA), where the same properties are verified at run-time by means of the integration of the adopted all-digital hardware monitoring system. The final goal is to compare the effectiveness and the efficiency of the two approaches by showing how it is possible (i) to reduce the time needed to perform verification and (ii) to provide the opportunity to verify more complex properties with respect to classical *Built-In Self-Test* (BIST) approaches.

This rest of this paper is structured as follows. The next section describes the development of the two pacemaker versions, the considered reliability properties, and an analysis of the obtained results with respect to the verification of such properties. Finally, Section 3 provides some conclusions and highlights future works on the topic.

## 2. The Proposed Use Case

As said before, the proposed use case targets the development of a pacemaker [14][15] (often considered in the literature due to its relevance, e.g., [16][17][18]) in two different versions: one based on a COTS microcontroller on PCB, where some properties are verified by means of a classical offline approach (i.e., script-based analysis of traces collected by means of a logical analyzer), and one based on a soft-core GPP on FPGA, where the same properties are verified at run-time by means of the integration of the adopted all-digital HW monitoring system. The final goals are (i) to reduce the time needed to perform verification, with respect to classical off-line approaches, and (ii) to provide the opportunity to verify more complex verification properties with respect to classical BIST approaches (by also allowing to perform verification during operative mode). So, the rest of this section provides more details about the reference pacemaker behavior, the development approaches adopted for the two pacemaker versions, the considered reliability properties, and obtained results with respect to the verification of such properties.

### 2.1. *Pacemaker Behavior*

The considered system is the *core* of a DDDR type pacemaker [16], i.e., the sub-system dedicated to the execute the algorithm related to the basic pacemaker functionality, i.e., without considering sensing/actuations sub-systems and extra functionality (e.g., actual heartrate estimation for dynamic adaptation, data logging and connectivity for internet-based remote monitoring, etc.). The behavior that the pacemaker core must exhibit to support the heart in case of failure is specified by the *Finite State Machine* (FSM) reported in Figure 1. It shows all the sequences that the pacemaker core must manage to preserve the correct heart functions.

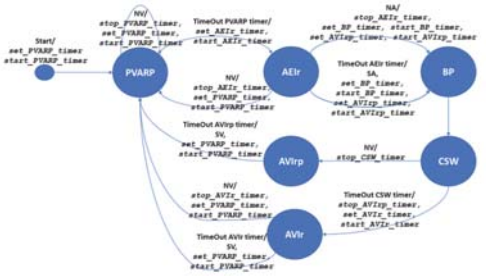


Figure 1. Pacemaker core behavior

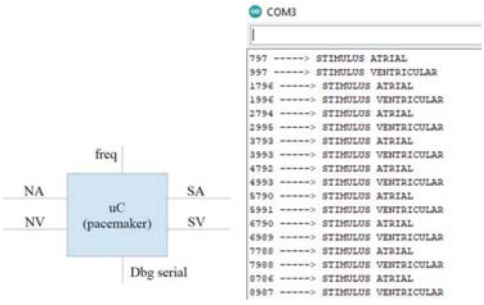


Figure 2. Baseline pacemaker core

The time origin is the ventricular event (natural or stimulated) that correspond to the start of the PVARP (*Post-Ventricular Atrial-Refractory Period*). During the PVARP the atrial catheter is inhibited (to avoid crosstalk phenomenon) while a ventricular event restarts the pacemaker cycle. After the PVARP, the atrial input is activated, and the pacemaker core waits for an atrial event. If this event doesn't occur, at the end of AEIr (*Atrial Escape Interval residue*) the pacemaker core sends an artificial atrial stimulus also starting the AVI (*Atrio-Ventricular Interval*). During the BP (*Blanking Period*) phase of the AVI, both the catheters are inhibited. At the start of the CSW (*Crosstalk Sensing Window*), the ventricular one is activated waiting for an event or for the AVIr (*AVI residue*). If the event doesn't occur before the end of AVIr the pacemaker core sends an artificial stimulus. If a ventricular event occurs during the CSW, the pacemaker core has to provide a precautionary stimulus to the ventricle after a given interval of time (i.e., 110 ms for 60 bpm) from the previous atrial event (natural or stimulated). In both cases the cycle starts again.

### 2.2. *Development approaches*

In order to show the benefits of the integration of an all-digital hardware monitoring system for the verification of properties related to reliability, it has been first developed a baseline pacemaker core, based on a COTS microcontroller (i.e., *Microchip Atmel ATMega328P*) on a PCB, for which verification relies on a classical approach. The baseline prototype is shown in Figure 2 (left).

The basic verification activity relies on a classical approach in which test vectors composed of natural atrial (NA) and/or natural ventricular (NV) stimulus have been manually generated and the output debug traces (Figure 2 right) have been extracted by means of a debug serial line and provided to a logic state analyzer for manual verification. Other than the basic verification activity, it is also possible to check for verification of more complex properties related to reliability issues. For example, it is possible to consider the following ones:

- *Unexpected Timer Fired* (UTF)
- *Unexpected States Sequence* (USS)

UTF is an event that signals when the timer activated at the exit from AEIr state (Figure 1), should be stopped in the CSW state (i.e., when going towards AVIr) but it is found already fired. This situation cannot happen in a correct design, but it can happen in the final implementation due to timing issues. USS is an event that signals when the pacemaker is moving along a state sequence that is not correct (i.e., not admissible) with respect to the possible ones defined by the related FSM.

Both the events can happen due to faults or system performance degradation, as a consequence of undetected manufacturing physical defects and/or aging [17]. Both the events are of critical importance and their notification to an external sub-system can be of great relevance. For example, the voting sub-system that manages the *Triple Module Redundancy* (TMR) typically used in a real

product could also consider such events, other than comparing the outputs, to determine which instance of the pacemaker core is the more reliable at a given time.

To verify the previously described properties (at design-time), test vectors have been generated and provided to the pacemaker by an advanced verification system. Figure 3 shows an additional microcontroller used to extract test vectors from real ECGs (Figure 4) and to provide them to the microcontroller implementing the pacemaker core. The outputs are then analyzed by looking at more expressive traces generated from the debug ones by means of some scripts (Figure 5).

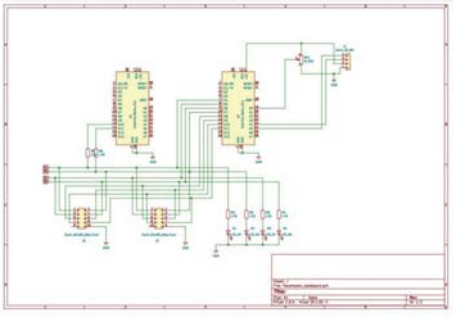

Figure 3. Advanced verification system

Figure 4. Test traces extraction from real ECGs

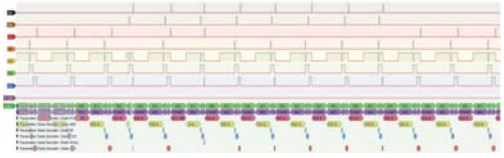

Figure 5. Script-based verification activity

The same pacemaker core has been then developed based on a soft-core GPP (i.e., *Xilinx Microblaze*, shown in Figure 6) on a FPGA (i.e., *Xilinx Artix7*), and integrated with the all-digital hardware monitoring system available in [19] to collect data related to system behavior and to verify, automatically and at run-time, the same properties considered above.

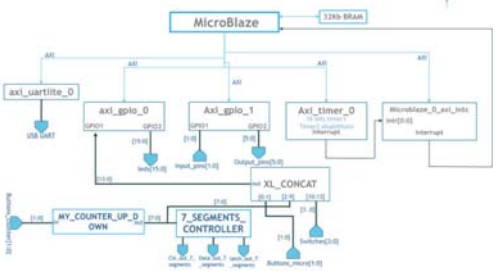


Figure 6. Soft-core based pacemaker core

### 2.3. *Results analysis*

The comparison of the two verification approaches has shown how it is possible (i) to reduce the time needed to perform verification and (ii) to provide the opportunity to verify more complex properties with respect to classical BIST approaches. In fact, during the development, it has been measured that the time needed to perform verification, with respect to the classical off-line approach (i.e., script-based analysis of output traces provided to a logic state analyzer) is reduced of 30% when considering the time needed to write and run the scripts, and analyze the results provided results. It is worth noting that the saved time would increase when considering more properties. Moreover, thanks to the proposed verification approach, it is possible to verify complex properties during operative mode, as it is not possible by means of classical BIST approaches. Finally, in order to compare the effectiveness and efficiency of the two verification approaches from a more general point of view it is possible to also consider that:

- the proposed approach allows to monitor internal system components while the classical one is limited to the observability offered by the debug serial line;
- the proposed approach allows to reduce overhead with respect to SW instrumentation-based ones without affecting the nominal behavior of the system.

## 3. Conclusions

For a lot of COTS digital devices, current reliability assessment is based on offline time-consuming analysis of traces collected while providing as input proper test vectors, and available BIST approaches shall be performed in a non-operative mode and are limited in the complexity of the verifiable properties. By exploiting an all-digital hardware monitor integrated into the system it is possible to collect, at run-time, data related to system behavior for the verification of properties related to reliability in a more effective and efficient way. The adopted all-digital HW embedded monitor [10][12][13] outperforms current SoA with respect to flexibility, size (i.e., cost) and energy consumption. For all this, it can be exploited to build next-generation embedded systems that need non-intrusive run-time monitoring actions (e.g., safety-critical systems like pacemakers), to improve HW/SW co-design methodologies [20][21], and to support training activities [22].

**Acknowledgement**

This work has been partially funded by the ECSEL Joint Undertaking (JU) under grant agreement No 876659. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Slovakia, Sweden, Finland, Belgium, Italy, Spain, Netherlands, Slovenia, Greece, France, Turkey. In addition, Germany including the Free States of Saxony and Thuringia, Austria, Belgium, Finland, France, Italy, the Netherlands, Slovakia, Spain, Sweden, and Turkey provide national funding.

**References**

[1] G. Kornaros and D. Pnevmatikatos. 2013. A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. ACM Trans. Des. Autom. Electron. Syst. 18, 2 (Apr. 2013).

[2] Susan L. Graham, Peter B. Kessler, and Marshall K. Mckusick. 1982. Gprof: A call graph execution profiler. In Proc. of the 1982 SIGPLAN symposium on Compiler construction (SIGPLAN '82).

[3] Rapita Systems. RapiTime, https://www.rapitasystems.com/products/rapitime, 2023.

[4] Nelissen G. et al., "A Novel Run Time Monitoring Architecture for Safe and Efficient Inline Monitoring". In Proc. Ada-Europe International Conference on Reliable Software Technologies, 2015.

[5] L. Shannon, P. Chow, "Using reconfigurability to achieve real-time profiling for hardware/software codesign," in Proc. ACM/SIGDA 12th Int. Symp. Field programmable gate arrays, Monterey, CA, 2004, pp. 190-199.

[6] J. Tong, M. Khalid, "Profiling Tools for FPGA-Based Embedded Systems: Survey and Quantitative Comparison", J. Comput., vol. 3, no. 6, pp. 1-14, June 2008.

[7] Philip M. Wells, Koushik Chakraborty, and Gurindar S. Sohi. 2009. Mixed-mode multicore reliability. In Proceedings of the 14th international conference on Architectural support for programming languages and operating systems (ASPLOS XIV). Association for Computing Machinery, New York, NY, USA, 169–180.

[8] J. C. Smolens, B. T. Gold, J. Kim, B. Falsafi, J. C. Hoe and A. G. Nowatryk, "Fingerprinting: bounding soft-error-detection latency and bandwidth," in IEEE Micro, vol. 24, no. 6, pp. 22-29, Nov.-Dec. 2004.

[9] D. Atienza et al., "Reliability-aware design for nanometer-scale devices," 2008 Asia and South Pacific Design Automation Conference, Seoul, Korea (South), 2008, pp. 549-554.

[10] K. Pressel et al., "The H2020-ECSEL Project "iRel40" (Intelligent Reliability 4.0)," 2021 24th Euromicro Conference on Digital System Design (DSD), Palermo, Italy, 2021.

[11] G. Valente et al., "Hardware Performance Sniffers for Embedded Systems Profiling," in 12th Workshop Intelligent Solutions Embedded Syst., 2015, pp. 29-34.

[12] G. Valente et al., "A flexible profiling sub-system for reconfigurable logic architectures", in Proceedings of PDP 2016, 2016.

[13] Giacomo Valente, Tiziana Fanni, Carlo Sau, Tania Di Mascio, Luigi Pomante, and Francesca Palumbo. 2021. A Composable Monitoring System for Heterogeneous Embedded Platforms. ACM Trans. Embed. Comput. Syst. 20, 5, Article 42 (September 2021), 34 pages.

[14] Siva K. Mulpuru, Malini Madhavan, Christopher J. McLeod, Yong-Mei Cha, Paul A. Friedman. Cardiac Pacemakers: Function, Troubleshooting, and Management: Part 1 of a 2-Part Series, Journal of the American College of Cardiology, Volume 69, Issue 2, 2017.

[15] Malini Madhavan, Siva K. Mulpuru, Christopher J. McLeod, Yong-Mei Cha, Paul A. Friedman, Advances and Future Directions in Cardiac Pacemakers: Part 2 of a 2-Part Series, Journal of the American College of Cardiology, Volume 69, Issue 2, 2017.

[16] C. Bolchini, L. Pomante, F. Salice, and D. Sciuto "Reliability Properties Assessment at System Level: A Co-Design Framework," Tech. Rep. 2001-85, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 2001.

[17] E. A. Rambo et al., "The Self-Aware Information Processing Factory Paradigm for Mixed-Critical Multiprocessing," in IEEE Transactions on Emerging Topics in Computing.

[18] Minjun Seo and Fadi Kurdahi. 2019. Efficient Tracing Methodology Using Automata Processor. ACM Trans. Embed. Comput. Syst. 18, 5s, Article 80 (October 2019).

[19] JOINTER, https://github.com/alkalir/jointer.git

[20] D. Ciambrone, V. Muttillo, L. Pomante, and G. Valente, "HEPSIM: An ESL HW/SW co-simulator/analysis tool for heterogeneous parallel embedded systems," 2018 7th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2018.

[21] Vittoriano Muttillo, Giacomo Valente, Luigi Pomante, Vincenzo Stoico, Fausto D'Antonio, and Fabio Salice. 2018. CC4CS: an Off-the-Shelf Unifying Statement-Level Performance Metric for HW/SW Technologies. In Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18).

[22] T. Di Mascio, L. Laura, and M. Temperini, "A Framework for Personalized Competitive Programming Training," 2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET), Olhao, Portugal, 2018.