2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS) | 978-1-6654-3595-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/DDECS52668.2021.9417023

Design and Implementation Strategy of Adaptive Processor-Based Systems for Error Resilient and Power-Efficient Operation

Mitko Veleski[1], Michael Hübner[1], Milos Krstic[2,3] and Rolf Kraemer[1,2]

[1]BTU Cottbus-Senftenberg, Chair of Computer Engineering, Cottbus, Germany
[2]IHP - Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
[3]University of Potsdam, Potsdam, Germany
E-Mail: {mitko.veleski, huebner}@b-tu.de, {krstic, kraemer}@ihp-microelectronics.com

***Abstract*—The contemporary computing systems are facing two major challenges: excessive power consumption and susceptibility to faults. In order to take advantage of techniques that efficiently address these challenges, the classic ASIC design flow requires some modifications. In this paper, we present a simple and convenient strategy for design and implementation of processor-based systems using highly configurable, cross-layer framework that encompasses techniques such as Adaptive Voltage and Frequency Scaling (AVFS) and Triple Modular Redundancy (TMR). The proposed strategy augments the conventional design flow with two additional steps to integrate the framework's hardware building blocks into the system. Such system is then able to dynamically switch between low power and error resilient operation modes according to the current requirements. By following the proposed strategy, we were able to implement processor-based system that significantly reduces the power consumption / increases the soft error resilience while preserving the performance at negligible area overhead of less than 1%.**

I. Introduction

The constant reduction of the Integrated Circuit (IC) technology dimensions to nano-scale feature sizes has made the modern computing systems more vulnerable to faults. Especially common are transient faults such as Single Event Upsets (SEUs) which could lead to critical soft errors and hence, system failure. In order to prevent such undesirable outcome, it became necessary for the conventional ASIC design flow to undergo some modifications. For example, the systems intended to operate in harsh environments such as space, include standard fault-tolerant techniques known as *Radiation Hardnening By Design* (RHBD) in the design flow [1]. RHBD usually consists of making the registers robust to radiation by employing TMR and / or making the power grid stronger [2]. This approach, however, results in dramatically increased overheads w.r.t both area and power consumption.

On the other hand, the power consumption has become a first-class metric for design of modern computing systems. Traditionally, power optimization is considered at the RTL level, where most of the architectural decisions have already been made (e.g. use of clock-gating) [3]. However, it is frequently necessary to specify power-related implementation details requiring semantics unsupported by the Hardware Description Languages (HDLs). Such power-related specifications are referred to as *power intents* - often included in power-aware design flows. The power intents are used to describe power domains, voltages, power cells (switches, isolation, level shifters) and power rail connections within the design [4]. On the downside, implementation of some advanced low-power techniques such as AVFS is not possible using power intents only, as AVFS requires realization of a closed control loop in hardware.

The majority of today's computing systems are heavily based on (multi)processors. Frequently, a system needs to accommodate a broad spectrum of applications with distinct requirements. Moreover, the system will also need to achieve an optimal trade-off between non-trivial and non-complementary metrics such as error resilience and power consumption. Thus, system adaptivity is becoming increasingly important requirement for a modern computing system.

In this paper we propose a design and implementation strategy of (multi)processor-based systems using highly configurable and cross-layer framework for adaptive operation. As described in our previous works, the framework enables techniques such as AVFS, selective circuit-level TMR and clock-gating to be synergistically combined while the system is online [5], [6]. At negligible area and complexity overheads (less than 1%), the framework enables significant power reduction (up to 15%, depending on the configuration), thus overcoming the limitations and drawbacks related to the RHBD and power intent techniques. To be able to achieve such results, the basic building blocks of the framework implemented in hardware - the SWIELD FF and the Chameleon System Operation Management Unit (SOMU) need to be appropriately interfaced within the system. We introduce convenient and flexible integration procedure that fits well to the standard flow. Hence, by replacing only 5%-13% of the timing critical flip-flops in the processor with SWIELD FFs, the framework can significantly increase the resilience to soft errors. Finally, due to the framework's ability to predict and avoid timing errors, the system does not suffer performance losses [5], [6].

The remainder of the paper is structured as follows. The

978-1-6654-3595-6/21/$31.00 ©2021 IEEE

related work in this area is reviewed in Section II. A comprehensive description of the proposed strategy is given in Section III. The practical implementation results are presented in Section IV. Finally, the paper is concluded in Section V.

II. Related Work

The previous works related to modified ASIC design flow move generally in two distinct directions: towards mitigation of soft errors or towards mitigation of timing errors. However, the common thread of all works is the usage of a particular "enhanced circuitry" that needs to be placed in the system or more often, replaced with some of the system's flip-flops / registers in order to achieve the desired goal. In the former case, TMR flip-flops are used as soft error protection circuits, while in the latter case, the timing errors are either detected / corrected or predicted by employing in situ monitors (ISMs). We give an overview of the related works for both TMR- and ISM-oriented approaches in Sections II-A and II-B respectively.

A. TMR-oriented Approaches

A method to improve the design robustness to soft errors without deviating from standard ASIC design flow is proposed in [7]. The method relies on specially designed TMR flip-flops that use standard, non-hardened flip-flops from the technology library as building blocks. Evaluation is performed on a test chip containing shift registers. As radiation hardness is the main concern in this work, no data regarding the power / area overhead is available. To reduce the circuit complexity while meeting specified reliability level, a methodology for automatic, selective TMR insertion is introduced in [8]. By analysing the circuit's topology and by using iterative optimization algorithm, a balance is maintained between two main design constraints - the reliability level and the area cost. However, the circuit type used for evaluation of the proposed methodology is not revealed. Similar approach is published in [9]. Namely, a framework for selective hardening to improve the Failure In Time (FIT) rate is formulated as a linear optimization problem. An unspecified IP core used extensively in safety-critical automotive microcontrollers serves as a test vehicle for experiments. The authors report a soft error robustness improvement of 32% at a price of 2% area increase. A comprehensive overview of techniques for selective hardening can be found in [10]. The obvious drawback of all these approaches is not taking the power consumption into account.

B. ISM-oriented Approaches

The ISMs can be placed at intermediate points or at the end points of the critical paths. To mitigate the variation- or aging-induced timing errors more efficiently, the ISMs are frequently placed at intermediate points. The internal nodes are shared between more critical paths, thus experience more transitions and higher activity compared to the end nodes. Therefore, the delay degradation can be detected faster [11]. On the other hand, the works that aim to employ some sort of adaptive regulation management in the system, usually place the monitors at the end of the critical paths.

1) ISMs at intermediate points: ISM insertion at intermediate points was first proposed in [12]. The goal is to reduce the number of ISMs in the design while efficient system health tracking can still be performed. Linear Programming (LP) is used to solve the ISM location selection problem. The ISM insertion is performed after the place and route phase. An evaluation using benchmarks for commercial processors showed that the total number of ISMs can be reduced by almost an order of magnitude if inserted at intermediate nodes. This comes at a price of additional 6% to 14% power overhead. The authors in [13], [14] propose an insertion flow for in situ monitoring of delay degradation based on dynamic monitor excitation. Thus, graph-based Static Timing Analysis (STA) is performed in order to determine the most critical paths and similarly to [12], the ISMs are inserted at intermediate points along those paths. The number of ISMs is decided during synthesis, but the actual insertion is performed after place and route. The evaluation results performed on ARM Cortex M0 processor showed that the number of ISMs can be decreased elevenfold compared to end point approaches. On the downside, power and area overheads can exceed 10% and 11% respectively.

2) ISMs at end points: Of course, the aging- / variation-induced delay degradation monitoring can be successfully implemented also by placing the ISMs at the end of the critical paths. The netlist-level sensor insertion flow for in situ monitoring of timing slacks in SoCs presented in [15] confirms the efficiency of such approach. However, the proposed method is not evaluated on entire system, rather individual processor units are used instead.

Obviously, just like TMR flip-flops, ISMs induce significant area and power overhead in the design unless proper strategy for insertion is not followed. In this regard, a selective replacement method for timing error predictive flip-flops named Canary FFs is introduced in [16]. To reduce area overhead, all flip-flops that violate the defined timing constraints are replaced with Canary FFs after synthesis and STA stages. An evaluation of the method on commercial processors showed area overhead between 2% and 12%. A power-aware, better-than-worst-case ISM insertion flow based on timing speculation is published in [17]. The ISMs are inserted at end points in the design following post-layout STA and critical path extraction. Although evaluated on a 32-bit multiplier only, power savings of 49% at a price of 5% performance loss is achieved. Another ISM insertion related work aiming to reduce the power overhead in the design caused by ISMs is introduced in [18]. Similar to [17], the ISMs are placed at the end points. Unlike [17], however, an evaluation is performed on LEON3 and by replacing only 20% of the critical paths in the processor's Integer Unit (IU), power reduction of 7% to 18% is reported. The performance degradation in this case is 16%. Details regarding the ISM insertion flow are omitted.

In contrast to the approaches discussed in this Section, we introduce a design and implementation strategy for a complete

adaptive (single- or multi-core) processor-based system using highly configurable and flexible framework that provides ability to tackle both soft and timing errors. A brief overview of the framework and an extensive description of the proposed strategy is given in Section III.

III. The Proposed Strategy

A complete and comprehensive report on the highly configurable and cross-layer framework is given in [5], [6]. The SWIELD FF and the Chameleon SOMU are two main hardware building blocks of the framework implemented at circuit and architecture layer of the system stack respectively. As described in [5] the SWIELD FF can operate in three distinct modes:

1) as a regular flip-flop;
2) as an ISM for prediction of timing errors as well as implementation of AVFS scheme;
3) as a TMR flip-flop.

Of course, some additional circuitry is required for realization of these enhanced functionalities. The Chameleon SOMU is responsible for managing the operation modes of the SWIELD FFs and is crucial for providing system adaptivity [5], [6]. In order to enable full utilization of the configurability and flexibility offered by the framework as well as proper implementation of the AVFS scheme, the SWIELD FF and the Chameleon SOMU need to be interconnected in a closed-loop fashion. This calls for some modifications in the traditional design flow. In this section we describe in detail the strategy for efficient and convenient integration of the framework in a complete processor-based system.

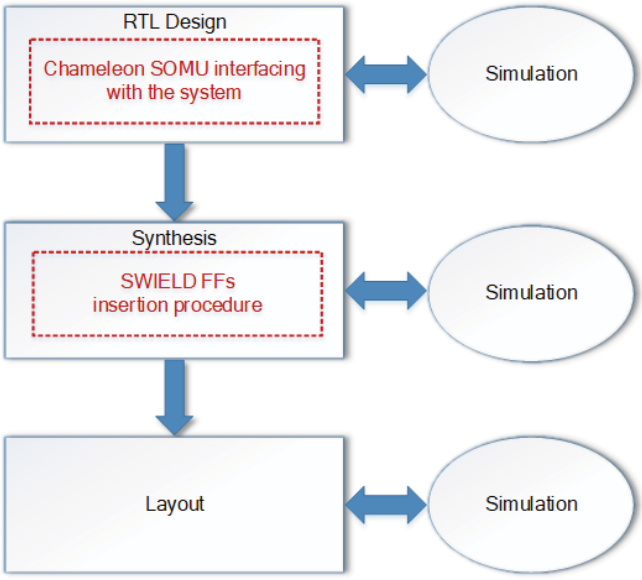

Fig. 1. The proposed design and implementation strategy flowchart. The blocks enclosed by solid lines (in blue) denote the classical design flow phases, while the internal blocks enclosed by dashed lines (in red) designate the additional steps in the proposed strategy. The location of the internal blocks points out the phases during which the additional steps need to be taken.

As depicted in Fig. 1, the proposed strategy consists of adding two major steps to the conventional design flow:

1) interfacing of the Chameleon SOMU with the system (performed on RTL level);
2) insertion of the SWIELD FFs in the design by replacing so-called critical flip-flops (intervention in the synthesized gate-level netlist is required).

The Chameleon SOMU is designed to be flexible, portable and scalable to minimize the effort for interfacing with the target processor-based system. In addition to being responsible for managing the operation mode of the SWIELD FFs, the Chameleon SOMU is meant to serve as a system clock generator and driver of a voltage regulator (required for implementation of AVFS) [5], [6]. Therefore, the interfacing comes down to instantiation of the Chameleon SOMU in the top RTL level of the design and proper connection of its input / output signals to the rest of the system components. Note that the Chameleon SOMU contains internal registers that should be accessible to the processor(s) through the system bus. Hence, an adequate address decoding needs to be performed. Ultimately, the Chameleon SOMU acts as a peripheral unit in the processor-based system with memory-mapped registers.

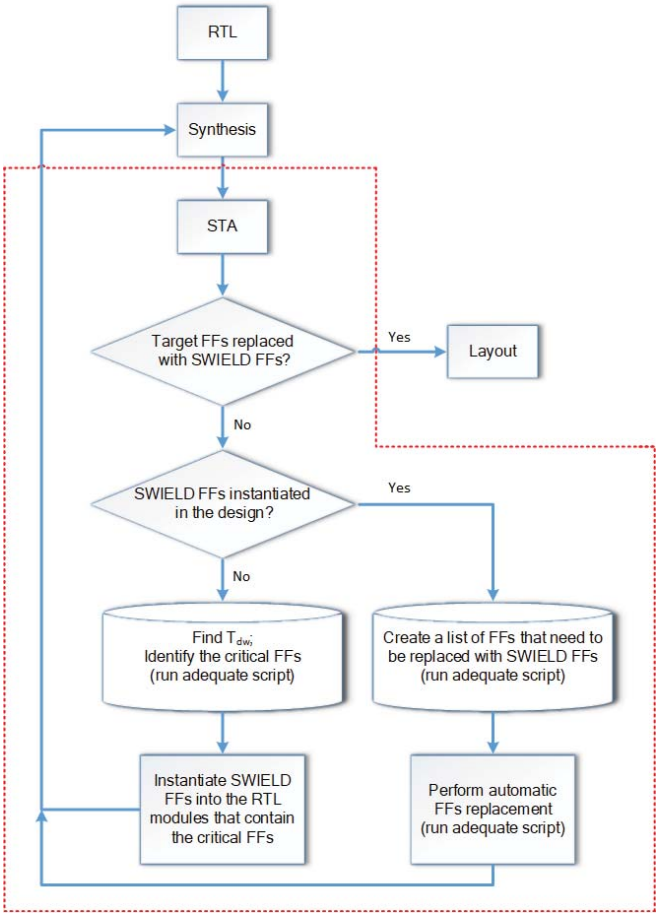

Fig. 2. A flowchart providing detailed display of the SWIELD FFs insertion procedure as a part of the design flow. The blocks surrounded by the red dashed line denote the necessary individual steps of the procedure.

On the other hand, the procedure for insertion of the SWIELD FFs in the system is a bit more complicated and demanding. Additionally, it requires some careful analyses before deciding the number and location of the SWIELD FFs to be inserted. Fig. 2 shows all the steps that need to be taken (some of them more than once) during the insertion procedure.

After the Chameleon SOMU is successfully integrated into the system, a synthesis is performed. In this phase, the gate-level netlist and the STA reports are obtained. The timing reports help to identify the so-called *critical* flip-flops, that is, the flip-flops which are candidates for replacement with SWIELD FFs. We use a metric called *pre-error detection window* (T_{dw}) to determine whether a given flip-flop will be classified as critical. A pre-error detection window is a particular time period between the alternative and the active clock edge. This metric is of key importance for efficient implementation of the AVFS loop. Namely, the SWIELD FF is capable of predicting timing errors by observing its input data transitions. If an input data transition occurs during T_{dw}, a timing pre-error warning signal is sent to the Chameleon SOMU. Based on the intensity of the pre-error warnings, the Chameleon SOMU adjusts the supply voltage / operating frequency in the system. Therefore, it is crucial to pick a length for the T_{dw} as accurate and as robust as possible [5], [6], [19]. The Algorithm 1 shows how this length is determined. Finally, all flip-flops with slack less than T_{dw} are considered critical.

Algorithm 1: Find T_{dw} and identify the critical FFs.

```
1  Set the default value of T_dw to T_dw = T_clk/2;
2  while T_dw > 0 do
3      Identify the critical flip-flops and their number;
4      Perform replacement;
5      Check timing;
6      Run simulation;
7      if mode(SWIELD_FF)=TMR ∧ simulation=OK
         then
8          break;
9      else
10         Gradually reduce T_dw (E.g. for 5% or 10%);
11 Report T_dw;
12 Report critical flip-flops;
```

The initial value of the pre-error detection window is set to $T_{dw} = T_{clk}/2$ (line 1). However, assigning such large value to T_{dw} (setting too long pre-error detection window) has several non-negligible drawbacks. Concretely, longer T_{dw} means that more critical flip-flops need to be replaced with SWIELD FFs which implies higher area overhead, often too high to be considered. Furthermore, the SWIELD FFs contribute to prolonging of the critical paths due to the additional circuitry required for operation in TMR mode [5]. Thus, too many SWIELD FFs may cause the system to crash when the SWIELD FF operation mode is set to TMR [6]. Finally, too long T_{dw} results in more timing pre-error warnings which leads to reduced power savings [19] provided by the AVFS scheme. Therefore, the value for T_{dw} is gradually reduced until a correct system operation with SWIELD FFs in TMR mode is assured (line 10). As soon as the value for T_{dw} that satisfies the condition in line 7 is found, a list and the number of critical flip-flops that correspond to the determined T_{dw} can be obtained (lines 11 and 12). However, this doesn't have to be the final set of critical flip-flops to be replaced with SWIELD FFs. For example, if so many SWIELD FFs in the system introduce higher area overhead than some predefined constraint, the number of critical flip-flops (and SWIELD FFs) can be further decreased by reducing T_{dw} until acceptable value is reached.

The identification of the critical flip-flops (line 3) and the replacement with SWIELD FFs (line 4) is automated by using in-house developed scripts. To find the number and the names of the critical flip-flops in the system, we use a script that parses the STA reports according to the determined value of T_{dw}. The list of the retrieved critical flip-flops together with the synthesized gate-level netlist are inputs to the script for automatic flip-flop replacement. This script locates the specified critical flip-flops in the system and rewires SWIELD FFs to their inputs and outputs. The output of the script is a modified gate-level netlist that contains SWIELD FFs instead of the critical flip-flops which can, but don't have to be located at the end of the critical paths.

Finally, the modified system is synthesized and timing check is performed. If the timing constraints are met, a post-synthesis, gate-level timing simulation is run to confirm the correct system operation. Afterwards, the system design flow can proceed towards the next stage. In case of timing violations, one can always reduce the T_{dw}, the operating frequency or both. Needless to say, the frequency reduction will result in performance degradation.

IV. Practical Implementation Results

The strategy presented in Section III was followed to implement adaptive processor-based systems (both single- [6] and multi-core) containing the highly configurable cross-layer framework. In both implementations, the systems are based on the LEON2 processor core [20]. However, any other general-purpose processor can be also used. The systems include several standard on-chip peripherals such as memory and interrupt controllers, timers, UARTs and programmable GPIO ports. Interconnection of the individual components is realized through Advanced Microcontroller Bus Architecture (AMBA) bus [21]. The RTL code of the Chameleon SOMU is adapted according to the AMBA specification and appropriate address decoding is performed, thus its internal registers could be easily accessed by the processor core(s). A voltage regulator (presented as a behavioural model only) able to scale the supply voltage level from nominal 1.2 V down to 0.8 V with 20 discrete steps of 20 mV is used to power the core(s). Incorporation of a voltage regulator in the system is of key importance for implementation of the AVFS scheme.

We implemented several (multi)processor-based system instances with different number of critical flip-flops replaced with SWIELD FFs by following the procedure described in Section III (see Fig. 2 and Algorithm 1). In all cases, the critical flip-flops were found to reside in the Integer Unit (IU) of the processor cores.

To confirm the benefits of the proposed strategy, we use the results of numerous experiments conducted on the implemented systems and compare them against works from Section II. Since similar trends were observed in the cases of both single- and multi-core systems, here we show only the results from the former case.

All results are obtained through post-synthesis, gate-level timing simulations. The synthesis is performed using the IHP 130 nm technology library [22] and the clock period is set to T_{clk} = 21 ns. Basically, experiments for power consumption estimation and soft error resilience evaluation are performed. In order to estimate the power savings provided by the AVFS scheme, the same set of experiments were also conducted on *referent systems* - systems which do not contain the framework, otherwise identical. A referent system maintains the nominal supply voltage throughout the entire operation, while the system with integrated framework adjusts the supply voltage according to the AVFS scheme. In this work, we only analyse the results from aspect of the proposed strategy. More details regarding the techniques and tools used for conducting of the experiments can be found in [6].

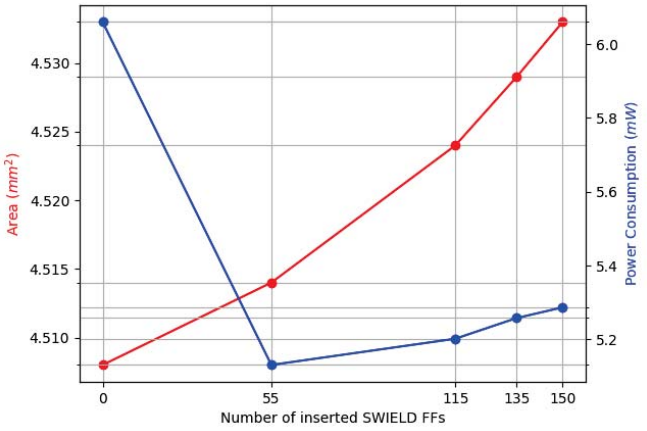

Fig. 3. Area overhead (red line) and power consumption (blue line) as functions of the number of inserted SWIELD FFs in the system. The points with coordinates (0, y) indicate the area and the power consumption of the referent system. Note that a straight line between two points in the plot corresponds to approximation rather than to linear relationship.

Fig. 3 shows the dependence of the area overhead and the power consumption on the number of SWIELD FFs inserted in the system according to the proposed strategy. Within the previously described experimental setup, the `while` loop in the Algorithm 1 could only be exited after reducing T_{dw} to 40% of the default T_{dw}. This corresponds to 150 critical flip-flops or around 13.5% of overall 1106 flip-flops in the IU. To observe the trends, we also implemented systems with T_{dw} reduced to 30%, 20% and 10% of the default T_{dw} which corresponds to 135, 115 and 55 critical flip-flops respectively. In the figure, it can be easily noticed that the design area compared to the referent system increases with the number of inserted SWIELD FFs. However, even in the case of 150 SWIELD FFs, the area overhead is less than 1% (more specifically, 0.55%), including the Chameleon SOMU. The power consumption, on the other hand, is significantly reduced due to the implementation of the AVFS scheme. Depending on the number of SWIELD FFs in the system, power savings between 13% (150 SWIELD FFs) and 15% (55 SWIELD FFs) can be achieved.

When the SWIELD FF is put into TMR operation mode, it is able to provide protection against soft errors [5], [6]. To investigate the error resilience in such case, numerous fault injection (FI) campaigns are performed. Essentially, faults of type SEU are injected at random times into the IU of the processor. The number of injected faults is correlated to the execution time. Concretely, we use a metric called *mean FI rate* defined as $\overline{FI_r} = 10\ faults/execution\_time$ as a basis for the FI campaigns. Since the used commercial FI tools (to the best of our knowledge) lack built-in functionality for correlation of the FI rates to the supply voltage, we provide various FI rates by doubling / halving and quadrupling / quartering $\overline{FI_r}$. Therefore, we assume that a higher FI rate during a given execution time corresponds to lower supply voltage. Table I summarizes the results of the FI campaigns.

TABLE I
FI RESULTS. THE STATUS OF THE INJECTED FAULTS IS DENOTED AS: D - DETECTED; M - MASKED; P - PROPAGATED. THE EXECUTION TIME IS APPROXIMATELY 17670 CYCLES LONG.

	Number of injected faults per execution time														
	$\overline{FI_r}/4 \approx 3$			$\overline{FI_r}/2 = 5$			$\overline{FI_r} = 10$			$2 * \overline{FI_r} = 20$			$4 * \overline{FI_r} = 40$		
	Status														
Number of SWIELD FFs	D	M	P	D	M	P	D	M	P	D	M	P	D	M	P
55	0	3	0	1	3	1	3	6	1	5	14	1	10	24	5
115	1	1	1	2	3	0	6	3	1	8	11	1	14	16	10
135	1	1	1	2	3	0	2	7	1	7	11	2	13	19	8
150	1	2	0	2	3	0	5	5	0	11	7	2	20	11	9

The faults which are randomly injected directly into one of the constituent flip-flops within the SWIELD FF are considered *detected*, because the SWIELD TMR structure outvotes such faults. On the other hand, all faults observed at the system primary outputs are classified as *propagated* faults. It is crucial to note that the detected faults are actually potential propagated (and failure-causing) faults in a processor-based system that does not contain the framework. Furthermore, during some of the FI campaigns, there were no propagated faults to the system primary outputs, that is, the executions finished failure-free. The shaded cells in Table I emphasize such FI campaigns. In summary, by replacing at most 13% of the flip-flops classified as critical with SWIELD FFs according to the proposed strategy, the resilience of the system to soft errors is considerably increased.

Finally, let us analyse how our work compares to the works overviewed in Section II. We establish a comparison criteria relying on several standard metrics (area / power overhead,

TABLE II
COMPARISON OF THIS WORK AGAINST RELATED WORKS FROM SECTION II.
THE MINUS SIGN PRECEDING SOME OF THE VALUES IN THE CELLS FROM THE COLUMN POWER OVERHEAD, ACTUALLY INDICATES POWER SAVING.
THE CELLS MARKED WITH * INDICATE THAT THE LOCATION OF THE TMR FF / ISM DEPENDS ON THE SOLUTION OF THE SELECTION ALGORITHM.
THE CELLS MARKED WITH ** INDICATE THAT NO EXPLICIT AREA OVERHEAD IS SPECIFIED, HOWEVER,
SOME AREA REDUCTION COMPARED TO OTHER APPROACHES IS REPORTED.

	Type of Error Mitigation	Timing Error Mitigation Method	Location of TMR flip-flop / ISM	Area Overhead	Power Overhead	Performance Degradation	Test Vehicle
[7]	Soft errors	/	Everywhere	N.A.	N.A.	N.A.	Shift registers
[8]	Soft errors	/	*	**	N.A.	N.A.	N.A
[9]	Soft errors	/	*	2%	N.A	N.A.	Unspecified IP Core
[12]	Timing errors	Prediction	Intermediate points	N.A.	6% - 14%	N.A.	Commercial processors
[13], [14]	Timing errors	Prediction	Intermediate points	11%	10%	N.A.	ARM Cortex M0
[16]	Timing errors	Prediction	End points	2% to 12%	N.A.	N.A.	Toshiba MeP & Renesas M32R
[17]	Timing errors	Prediction	End points	**	-49%	5%	32-bit multiplier
[18]	Timing errors	Detection	End points	N.A.	-7% to -18%	16%	LEON3 IU
This work	**Soft & Timing errors**	**Prediction**	*	**0.55%**	**-13% to -15%**	**No**	**LEON2-based system**

performance degradation), but also on some features which we consider important from aspect of this paper (such as the type of the errors targeted for mitigation, type of test vehicle and so on). The comparison criteria-relevant data gathered from the different works is summarized in Table II. It is easily noticeable that our work outperforms the related works in almost every aspect: at the expense of negligible area overhead, our strategy enables implementation of processor-based systems able to significantly reduce the power consumption and improve the soft error resilience while preserving the performance. The approaches proposed in the related works achieve favourable results in some segments of the comparison criteria, but none managed to implement an entire system able to synergistically tackle all of the considered critical metrics.

V. CONCLUSION

In the previous sections we presented a simple and convenient strategy for design and implementation of adaptive processor-based systems for power-efficient and error resilient operation using highly configurable and cross-layer framework. The strategy consists of adding two main steps to the traditional design flow: interfacing of the Chameleon SOMU with the system and replacing the flip-flops classified as critical with SWIELD FFs. As the framework provides dynamic switching between low power and soft error resilient operation modes, we show in Section IV that a system implemented according to the proposed strategy is able to significantly reduce the power consumption and increase the soft error resilience while preserving the performance at the price of negligible area overhead.

REFERENCES

[1] O. Schrape, A. Breitenereiter, S. Zeidler, and M. Krstic, "Aspects on timing modeling of radiation-hardness by design standard cell-based tmr flip-flops," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pp. 639–642, IEEE, 2019.

[2] A. Simevski, O. Schrape, C. Benitoy, M. Krstic, and M. Andjelkovic, "PISA: Power-robust multiprocessor design for space applications," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–6, IEEE, 2020.

[3] F. B. Muslim, A. Qamar, and L. Lavagno, "Low power methodology for an ASIC design flow based on high-level synthesis," in *2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 11–15, IEEE, 2015.

[4] G. Panić, *A methodology for designing low power sensor node hardware systems*. PhD thesis, BTU Cottbus-Senftenberg, 2015.

[5] M. Veleski, R. Kraemer, and M. Krstic, "SWIELD: An in situ approach for adaptive low power and error-resilient operation," in *2019 IEEE East-West Design & Test Symposium (EWDTS)*, pp. 1–6, IEEE, 2019.

[6] M. Veleski, M. Hübner, M. Krstic, and R. Kraemer, "Highly configurable framework for adaptive low power and error-resilient system-on-chip," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pp. 24–28, IEEE, 2020.

[7] V. Petrovic and M. Krstic, "Design flow for radhard TMR flip-flops," in *2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, pp. 203–208, IEEE, 2015.

[8] O. Ruano, J. Maestro, and P. Reviriego, "A methodology for automatic insertion of selective TMR in digital circuits affected by SEUs," *IEEE Transactions on Nuclear Science*, vol. 56, no. 4, pp. 2091–2102, 2009.

[9] P. V. Torvi, V. Devanathan, and V. Kamakoti, "Framework for selective flip-flop replacement for soft error mitigation," in *2015 28th International Conference on VLSI Design*, pp. 381–386, IEEE, 2015.

[10] I. Polian and J. P. Hayes, "Selective hardening: Toward cost-effective error tolerance," *IEEE Design & Test of Computers*, vol. 28, no. 3, pp. 54–63, 2010.

[11] S. Sadeghi-Kohan, A. Vafaei, and Z. Navabi, "Near-optimal node selection procedure for aging monitor placement," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pp. 6–11, IEEE, 2018.

[12] L. Lai, V. Chandra, R. C. Aitken, and P. Gupta, "Slackprobe: A flexible and efficient in situ timing slack monitoring methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 8, pp. 1168–1179, 2014.

[13] H. Ahmadi Balef, H. Fatemi, K. Goossens, and J. Pineda de Gyvez, "Effective in-situ chip health monitoring with selective monitor insertion along timing paths," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pp. 213–218, 2018.

[14] H. A. Balef, K. Goossens, and J. P. de Gyvez, "Chip health tracking using dynamic in-situ delay monitoring," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 304–307, IEEE, 2019.

[15] M. Sadi, L. Winemberg, and M. Tehranipoor, "A robust digital sensor IP and sensor insertion flow for in-situ path timing slack monitoring in SoCs," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2015.

[16] Y. Kunitake, T. Sato, H. Yasuura, and T. Hayashida, "A selective replacement method for timing-error-predicting flip-flops," *Journal of Circuits, Systems, and Computers*, vol. 21, no. 06, p. 1240013, 2012.

[17] S. M. Londoño and J. P. de Gyvez, "A better-than-worst-case circuit design methodology using timing-error speculation and frequency adaptation," in *2012 IEEE International SOC Conference*, pp. 15–20, IEEE, 2012.

[18] C.-M. Huang, T.-T. Liu, and T.-D. Chiueh, "An energy-efficient resilient flip-flop circuit with built-in timing-error detection and correction," in *VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, IEEE, 2015.

[19] M. Wirnshofer, *Variation-aware adaptive voltage scaling for digital CMOS circuits*. Springer, 2013.

[20] GaislerResearch, "LEON2 processor user's manual," *Version 1.0.30, XST Edition, July*, 2005.

[21] ARM, "AMBA specification rev. 2.0," *http://www.arm.com*, 1999.

[22] https://www.ihp-microelectronics.com.