Microelectronics Reliability xxx (xxxx) 115198

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

RESAC: A redundancy strategy involving approximate computing for error-tolerant applications

Padmanabhan Balasubramanian [a,*], Douglas L. Maskell [a], Krishnamachar Prasad [b]

[a] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[b] School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, New Zealand

ARTICLE INFO

Keywords:
Fault tolerance
Redundancy
Approximate computing
Arithmetic circuits
Logic design
Low power
High speed
CMOS

ABSTRACT

Given the continuing miniaturization of underlying transistors, electronic functional units (circuits/systems) become increasingly susceptible to high-energy radiation, encountered in applications like space. Hence, redundancy is employed as a radiation hardening by design strategy to cope with faults of functional units used in such applications and to maintain their correct operation. Triple modular redundancy (TMR), which is a subset of N-modular redundancy (NMR), that can mask any single fault or a faulty functional unit has been widely used. However, compared to a simplex implementation a TMR implementation requires two additional functional units and a majority voting logic therefore a TMR implementation's area and power overheads are greater by over 200 %. This is burdensome for resource-constrained applications like space where low power and energy efficiency are important considerations. This paper presents a new redundancy strategy involving approximate computing called RESAC for error-tolerant applications such as digital image/video/audio processing, which is used in space systems. We evaluated the feasibility of RESAC for an image processing case study and the results confirm the usefulness. For implementation using a 28-nm CMOS technology, RESAC achieves reductions in area, delay, and power by 22.3 %, 15.3 %, and 24.9 % compared to TMR. Nonetheless, RESAC can address any NMR.

1. Introduction

Due to the continuing miniaturization of underlying transistors, electronic functional units such as circuits or systems tend to become increasingly susceptible to high-energy radiation [1,2], especially when operating in harsh environments such as space. Several types of high-energy radiation can impact electronic systems, and some examples are solar flares, cosmic rays, Van Allen radiation belts, electromagnetic pulses, solar particle events, etc. Some of the effects of radiation on transistors and devices include total ionizing dose which is the accumulated radiation dose over time, single event effects which include single event upsets, single event latch-ups, and single event transients, displacement damage due to the collision of heavy ions in space with atoms in a semiconductor material, temperature/thermal recycling due to the extreme temperatures encountered in space and the inducing of mechanical stress on the device due to the orbiting of spacecraft around celestial objects leading to failures and reliability issues, etc. To mitigate the effects of high-energy radiation, spacecraft, and satellites are designed with protective measures [3] such as: (i) radiation hardening by using radiation-hardened materials such as SiGe, SiC, GaAs, etc., and techniques (for example, redundancy), (ii) shielding to reduce the penetration of radiation and minimize its impact on sensitive components, (iii) error-correction mechanisms to detect and correct errors caused by radiation-induced faults, and (iv) rigorous testing and qualification procedures to ensure that components can withstand the expected levels of radiation in space. Through these methods, the resilience of transistors and devices used in space systems can be enhanced ensuring their reliable operation throughout the mission duration.

In this paper, we focus on redundancy that is widely adopted as a radiation-hardening-by-design strategy in the design of functional units for safety-critical applications. N-modular redundancy (NMR), where N identical functional units are used and the respective outputs of the N functional units are majority voted is popular and well-known. In NMR, with N being odd (at least 3), $(N + 1) / 2$ functional units should operate correctly, and the arbitrary fault(s) of utmost $(N - 1) / 2$ functional units are tolerated. A fault may be temporary or permanent in nature, which may result in failure.

* Corresponding author.
E-mail address: balasubramanian@ntu.edu.sg (P. Balasubramanian).

https://doi.org/10.1016/j.microrel.2023.115198
Received 1 June 2023; Received in revised form 11 July 2023; Accepted 14 August 2023
0026-2714/

3-modular redundancy or triple-modular redundancy, abbreviated as 3MR or TMR, is the fundamental version of NMR, that uses three identical functional units which can tolerate the fault of any one functional unit. Higher-order NMR such as quintuple modular redundancy, septuple modular redundancy, etc. may be used to tolerate the faults of multiple functional units but those would involve significant overheads in the design metrics (delay, area, power, energy), and cost of hardware implementation. Hence, it is suggested to use higher-order NMR progressively [4] in a redundant implementation whereby TMR may be used widely and higher-order NMR may be used selectively.

TMR is popular among NMR and has been used for many safety-critical applications [5]. A study analyzing the exposure of Xilinx Virtex FPGAs to proton and heavy-ion-induced radiation [6] reported that 96 % to 99 % of the total upsets corresponded to single-bit upsets and the remainder corresponded to multiple-bit upsets. The study showed the dominance of single-bit upsets in radiation environments, which can be overcome by employing TMR.

Although a standard TMR implementation can fully overcome the fault of any one functional unit, it requires >200 % area and power overheads compared to a single functional unit implementation, also called simplex implementation. This is because TMR uses two additional functional units and an extra majority voting logic compared to a simplex implementation, and this could be a premium for resource-constrained applications such as space where low power and energy efficiency are important.

This paper presents a new redundancy strategy utilizing approximate computing called RESAC, as an alternative to NMR, especially for inherently error-tolerant applications. Nevertheless, RESAC may be customized to suit non-error-tolerant applications. Many practical applications are inherently error-tolerant such as digital signal processing, neuromorphic computing, big data mining and analytics, low-power graphics processing, etc. [7]. For example, digital image, video, and audio processing are error-tolerant since minor distortions in an image or video frame or a feeble noise in audio may not be distinguished by humans due to the innate limitations of human perception. Also, digital image/video/audio processing is used in safety-critical applications like space. In this paper, we specifically consider a comparison between RESAC and TMR (as a representative of NMR).

In the rest of this paper, Section 2 surveys the existing literature on alternative redundancy approaches to TMR. Section 3 describes the proposed redundancy strategy RESAC and compares it with TMR. Section 4 presents the results of the digital image processing case study corresponding to TMR and RESAC. Section 5 gives the design metrics of simplex and redundant implementations of the functional unit considered. Section 6, finally, draws some conclusions.

2. Literature survey

To reduce the area and power overheads of a standard TMR implementation compared to simplex implementation, alternative redundancy approaches have been suggested by researchers [8–11].

Selective TMR (STMR) insertion [8] considers employing TMR in the select portions of a functional unit that are deemed to be critical. Thus, STMR would have reduced area and power overheads compared to TMR. However, differentiating the critical and non-critical portions of a functional unit may not be trivial for all practical applications. Moreover, any fault in the non-critical portion of a functional unit might cause undesirable consequences.

Approximate TMR (ATMR) [9] suggests the use of one original i.e., accurate functional unit, and two distinct approximate i.e., non-accurate versions of the original functional unit. The corresponding outputs of the functional units are voted on using accurate majority voters. Since two approximate functional units with reduced logic are used in ATMR, this would help to reduce its area and power compared to a counterpart TMR implementation. ATMR is realized such that the corresponding outputs of any two functional units (either accurate and approximate functional units or two approximate functional units) should match. Supposing that the accurate functional unit becomes faulty, a correct output cannot always be expected from an ATMR implementation. This also means that the complete tolerance of a single fault is not guaranteed in an ATMR implementation. Hence, ATMR may only provide partial fault tolerance.

Fully approximate TMR (FATMR) [9,10] was suggested as a further optimized version of ATMR. In FATMR, three distinct approximate versions of an original (accurate) functional unit are used and their corresponding outputs are voted on using accurate majority voters. While realizing FATMR, the corresponding outputs of any two approximate functional units are made to match. Since only approximate functional units are used throughout, FATMR could help to reduce all of the design metrics viz. area, delay, and power compared to TMR and ATMR. However, FATMR, being a weaker variant of ATMR, a correct output is not always guaranteed when an approximate functional unit may become faulty. Thus, FATMR like ATMR can only provide a partial fault tolerance. Nonetheless, to our knowledge, there has been no demonstration of the usefulness of ATMR or FATMR for any practical application.

A majority-voting based reduced precision redundancy (MVRPR) adder was presented in [11], targeting inherently error-tolerant applications. However, MVRPR can be generalized as an alternative redundancy technique to TMR. In [11], according to MVRPR, an accurate adder (representing a functional unit) is split into two equal-sized parts namely the most significant part (MSP) and the less significant part (LSP) both of which comprise accurate logic. The outputs of MSP and LSP are jointly considered as the primary output. The MSP and LSP are connected through accurate carry logic. TMR is applied to the MSP while the LSP is kept as simplex. Therefore, MVRPR guarantees only moderate fault tolerance since the LSP is not protected because it is simplex but MVRPR could lead to savings in area and power compared to TMR. The hypothesis behind the MVRPR adder design is that the sum output of the MSP would dominate the sum output of the LSP, and so even if the carry output of the LSP, which is given as the carry input to the MSP, may become erroneous due to a single-bit upset, it may only have a negligible impact on the sum output of the MSP. But, based on our experimentation we noted that this hypothesis may not universally hold well particularly when small to medium size inputs are added since the impact on the overall sum output was not analyzed. Also, the possibility of sum bit(s) in the LSP of MVRPR getting affected, since they are unprotected, was not analyzed in [11]. Further, the usefulness of an MVRPR adder was not demonstrated for any practical application.

3. TMR and proposed redundancy (RESAC)

Before describing the proposed redundancy strategy, TMR is briefly discussed. Representative architectures of TMR and RESAC are shown in Fig. 1a and b respectively. According to TMR, a functional unit is triplicated into three identical functional units say, A, B, and C, all of which are accurate. The functional units are shown in lavender boxes in Fig. 1a. Let us assume that each functional unit produces 3 outputs say, P, Q, and R. The respective outputs of the functional units are voted on using accurate majority voters 1, 2, and 3, shown in rose boxes, which yield the primary outputs M1, M2, and M3. Assuming X, Y, and Z are the inputs of a majority voter, its output (say V) is given by $V = XY + YZ + XZ$. Different designs of a 3-input majority voter are presented in [12], and any of those may be used to physically realize a majority voter.

The proposed redundancy strategy utilizing approximate computing (RESAC) is ideally suited for inherently error-tolerant applications, promising potential savings in all the design metrics (area, delay, power, and energy) compared to conventional NMR while enabling acceptable output quality. However, RESAC may be customized to suit non-error-tolerant applications depending on the accuracy of output

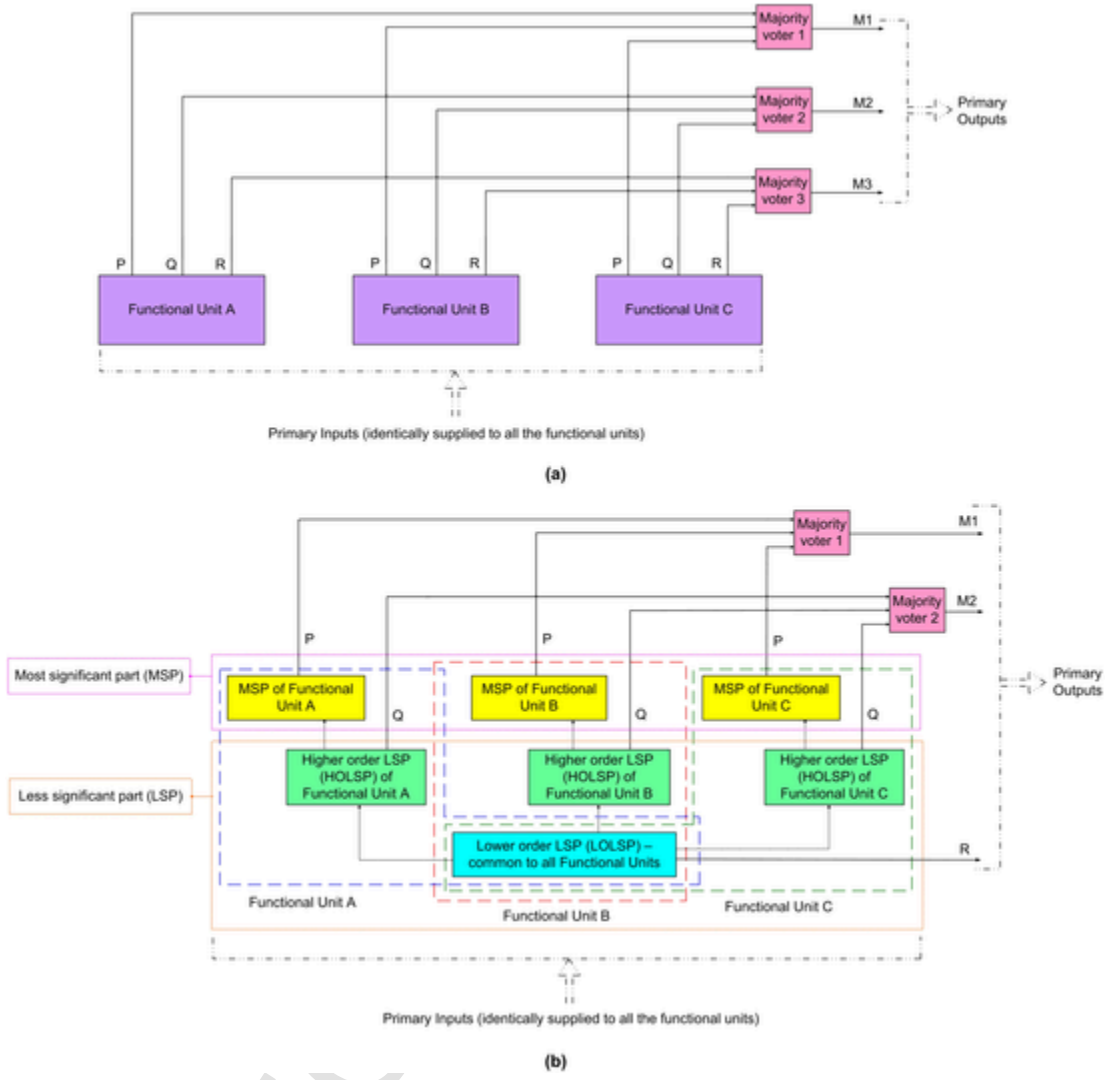

Fig. 1. Architecture of (a) TMR, and (b) proposed redundancy, RESAC (here, a counterpart of TMR is shown).

that may be deemed acceptable for a practical application. Although RESAC can address any NMR, in this paper we consider a 3-tuple version of RESAC to draw a comparison with TMR.

As an alternative to TMR, in RESAC, the same three functional units A, B, and C are considered, as shown in Fig. 1b. But the constituents of the functional units are somewhat modified in RESAC and are not entirely the same as in TMR. However, the modified functional units are identical in RESAC. In RESAC, each functional unit is first split into two parts viz. a most significant part (MSP) that is highlighted by the pink rectangular box in dotted lines in Fig. 1b, and a less significant part (LSP) that is highlighted by the orange rectangular box in dotted lines. The classification of the parts is based on the significance associated with the outputs produced by these parts in relation to the primary output, and the outputs of both these parts are jointly considered as the primary output.

In RESAC, the MSP of each functional unit is realized accurately like the original functional unit while the LSP is realized approximately. The size of the MSP and the LSP and the manner of approximation of the LSP are decided based on an application's requirement to ensure acceptable output quality. This is typically determined by performing an error analysis based on a comparison with the output of an accurate functional unit to ensure the error metrics are bounded within the limits that would be acceptable for an application.

Adders, multipliers, and other arithmetic circuits, as well as data paths containing arithmetic functions like the discrete cosine transform, finite/infinite impulse response filter, and the sum of absolute difference, etc. typically feature varying levels of significance in their output bits. This characteristic can be utilized to construct these functional units by dividing them into MSP and LSP, according to RESAC. Hence, arithmetic functions are well-suited for the implementation of RESAC. However, RESAC may not be suitable for the implementation of control logic. The RESAC architecture depicted in Fig. 1b is generic, and the nature and extent of approximation to be incorporated in the LSP of a functional unit should be determined based on the target application.

In RESAC, the LSP is divided into two sub-parts viz. a higher order LSP (HOLSP) and a lower order LSP (LOLSP), where HOLSP has greater significance than LOLSP. HOLSPs and LOLSP are portrayed by the green and turquoise boxes respectively in Fig. 1b while MSPs are depicted by the yellow boxes. In RESAC, TMR is applied to the MSP and HOLSP while the LOLSP is implemented as simplex. So, LOLSP is common to all functional units. The constituents of functional units A, B, and C in RESAC are shown within the blue, red, and green boxes appearing in dashed lines in Fig. 1b. MSP and HOLSP of the functional units are fully protected against any single fault while LOLSP is unprotected. Hence, the sizes of HOLSP and LOLSP may be decided based on the presumption that the LOLSP may be entirely affected and despite this, the required output quality should not be compromised for a target applica-

tion. This would help ensure a RESAC implementation's reliability against single or multiple-bit upsets.

In RESAC, MSPs and HOLSPs may or may not be connected, and, HOLSPs and LOLSP may or may not be connected depending upon an application requirement hence these connections are shown in dotted lines in black in Fig. 1b. Moreover, the logic connecting MSPs and HOLSPs may not be accurate and rather simplified, implying that the critical data path of a RESAC implementation would be governed by the delay encountered in the MSP. This would mean the operating speed of RESAC is likely to be potentially higher than TMR. HOLSPs and LOLSP may be isolated but even if they may be connected, it is unlikely to result in a critical data path. Data processing in HOLSPs and LOLSP could occur in parallel when they are isolated. The outputs of MSPs of the functional units are voted on using majority voter 1, and the outputs of HOLSPs of the functional units are voted on using majority voter 2. Both majority voters 1 and 2, shown by the rose boxes are accurate. The output of LOLSP i.e., R directly serves as a part of the final output, which may or may not be equal to output M3 of TMR, depending upon the input supplied. M1, M2, and R denote the primary outputs of RESAC.

4. Case study and results

We considered digital image processing as a case study to evaluate TMR and RESAC comparatively. For this, we considered some grayscale images from [13]. An image (8-bit grayscale with a 512 × 512 spatial resolution) was first transformed into a matrix form and then processed by applying the fast Fourier transform (FFT). The image was reconstructed by applying inverse FFT (IFFT). The addition involved in FFT and IFFT computations was performed accurately (according to TMR) and approximately (according to RESAC), separately. Multiplication was performed accurately. A 32-bit adder was considered for addition and it was ensured that no data loss or overflow occurred in the computations. The adder was treated as the functional unit, and it was implemented redundantly according to TMR and RESAC.

An accurate 32-bit adder was used for TMR, and an approximate 32-bit adder, M-HERLOA [14] was used for RESAC whose architecture is shown in Fig. 2. Here, N-bits represents the adder size, (N-K) bits is the size of the precise adder part which is shown in blue, and K bits is the size of the imprecise adder part which is shown in brown. The precise adder part is more significant than the imprecise adder part. The addition is performed accurately in the precise part and inaccurately in the imprecise part. In Fig. 2, A and B represent the adder inputs and SUM represents the adder output. The sum bits produced by the precise and imprecise adder parts are concatenated to generate the final sum which is of size (N + 1) bits that include the carry overflow from the addition.

The adder portions corresponding to MSP, HOLSP, and LOLSP of the RESAC architecture (shown in Fig. 1b) are depicted within the red, green, and pink vertical dashed lines respectively in Fig. 2. As seen in Fig. 2, MSP contains accurate adder logic while HOLSP and LOLSP contain inaccurate adder logic. To correlate Fig. 2 with Fig. 1b, sum bits SUM_N up to SUM_K of Fig. 2 represent the output P of Fig. 1b, SUM_{K-1} up to SUM_{K-4} of Fig. 2 represents the output Q of Fig. 1b, and SUM_{K-5} up to SUM_0 of Fig. 2 represent the output R of Fig. 1b.

The approximate adder for RESAC comprises a 22-bit MSP and a 10-bit LSP. This partitioning was determined to be optimum based on trial-and-error experimentation with many images and error analysis. The 10-bit LSP was subdivided into a 4-bit HOLSP and a 6-bit LOLSP, and this partitioning was determined to be optimum based on error analysis in [14] which was confirmed through experimentation with many images.

In the case of RESAC, two simulation instances were considered – one assuming no bit upset in LOLSP, and another assuming the occurrence of worst-case bit upsets in LOLSP since LOLSP of RESAC is unprotected (as seen in Fig. 1b). The results of the image processing corresponding to TMR and RESAC are shown in Fig. 3.

The peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) were calculated to quantify the quality of the processed images. While PSNR is a generic figure of merit used in digital signal processing [15], SSIM is a specific figure of merit relevant to digital image processing [16]. The ideal PSNR value is ∞ and the ideal SSIM value is unity, which corresponds to TMR. Based on RESAC, a PSNR > 30 dB and an SSIM close to unity are achieved for the processed images while considering no bit upset and worst-case bit upsets occurring in LOLSP of RESAC. This has been confirmed by experimentation with many images, and the subjective quality of all the images in Fig. 3 appears to be the same. The values of PSNR and SSIM for the images processed using RESAC are acceptable, thus validating the usefulness of RESAC.

5. Implementation and design metrics

Since the adder is alone different between TMR and RESAC for the case study considered, we structurally described the following adders in Verilog HDL for synthesis: (i) an accurate 32-bit carry look-ahead adder (CLA) [17] comprising eight 4-bit CLAs representing a simplex implementation, (ii) a TMR implementation of the accurate 32-bit CLA, and (iii) the RESAC implementation of a 32-bit approximate adder [14] incorporating the accurate CLA for the MSP. The MSP of RESAC is 22 bits; hence, it is realized using five 4-bit CLAs and one 2-bit CLA. Synopsys tools were used for synthesis, simulation, and estimation of the design metrics (area, delay, and power) of adders. The adders were synthe-

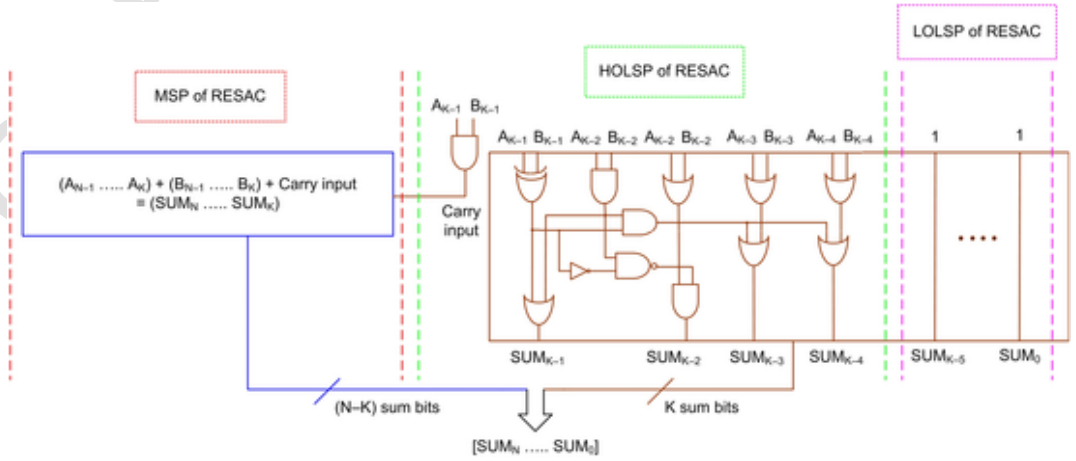

Fig. 2. Architecture of approximate adder used for the proposed redundancy strategy (RESAC) in this work.


TMR
RESAC
(no bit upset in LOLSP)
(PSNR:33.1088 dB)
(SSIM:0.95586)
RESAC
(worst-case bit upsets in LOLSP)
(PSNR:33.8989 dB)
(SSIM:0.95723)
TMR
RESAC
(no bit upset in LOLSP)
(PSNR:32.821 dB)
(SSIM:0.94687)
RESAC
(worst-case bit upsets in LOLSP)
(PSNR:33.7512 dB)
(SSIM:0.94546)
TMR
RESAC
(no bit upset in LOLSP)
(PSNR:32.9572 dB)
(SSIM:0.94228)
RESAC
(worst-case bit upsets in LOLSP)
(PSNR:33.8542 dB)
(SSIM:0.94383)
TMR
RESAC
(no bit upset in LOLSP)
(PSNR:32.8586 dB)
(SSIM:0.96374)
RESAC
(worst-case bit upsets in LOLSP)
(PSNR:33.7891 dB)
(SSIM:0.96375)
TMR
RESAC
(no bit upset in LOLSP)
(PSNR:32.8431 dB)
(SSIM:0.96481)
RESAC
(worst-case bit upsets in LOLSP)
(PSNR:33.7242 dB)
(SSIM:0.96577)

Fig. 3. Digital images processed using TMR and RESAC. In RESAC, no bit upset and worst-case bit upsets in LOLSP (which is unprotected) are considered.

sized by Design Compiler using a 28-nm CMOS standard cell library [18]. The total area of the adders including the areas of cells and interconnects was estimated. A default wire load model was considered for synthesis, and fanout-of-4 drive strength was assigned to all the output ports i.e., the sum bits of the adders.

A typical case library specification having a supply voltage of 1.05 V and an operating temperature of 25 °C was considered for synthesis. Functional simulations of simplex and redundant adders were performed by supplying a test bench comprising over a thousand random inputs at a latency of 2 ns (i.e., 500 MHz). The switching activity information obtained from functional simulations was used to estimate the total power using PrimePower. PrimeTime was used to estimate the critical path delay. The design metrics estimated are given in Table 1.

TMR implementation incorporates two additional functional units and an extra majority voting logic compared to a simplex implementation, which increases its area occupancy by 2.3× and power dissipation by 2.2× in comparison. But the simplex implementation is not fault-tolerant. Due to the extra majority voter found in the critical data path of TMR, it has somewhat increased delay than the simplex implementation. RESAC incorporates approximate functional units where the MSP is realized accurately but the LSP is realized approximately using reduced logic. Also, TMR is applied only to MSP and HOLSP of RESAC whereas the LOLSP is implemented as simplex. Therefore, RESAC facilitates reductions in area and power compared to TMR. Since the critical path delay of RESAC is governed by the maximum propagation delay of the MSP alone (here, a 22-bit accurate adder delay in RESAC compared to the 32-bit accurate adder delay in simplex and TMR implementations), therefore RESAC implementation reports a reduced delay than both simplex and TMR implementations.

The theoretical critical path delays of simplex, TMR, and RESAC implementations, excluding interconnect and parasitic delays, are modeled by Eqs. (1) to (3). In Eqs. (1) to (3), D_{XOR2}, D_{AND4}, D_{OR4}, D_{AO21}, and D_{AO222} represent the typical propagation delays of a 2-input XOR gate, a 4-input AND gate, a 4-input OR gate, an AO21 complex gate, and an AO222 complex gate respectively.

$$D_{Simplex} = (D_{XOR2} + D_{AND4} + D_{OR4}) + (6 \times D_{AO21}) + (D_{AO21} + D_{XOR2}) \quad (1)$$

$$D_{TMR} = (D_{XOR2} + D_{AND4} + D_{OR4}) + (6 \times D_{AO21}) + (D_{AO21} + D_{XOR2}) + D_{AO222} \quad (2)$$

$$D_{RESAC} = (D_{XOR2} + D_{AND4} + D_{OR4}) + (3 \times D_{AO21}) + (D_{AO21} + D_{XOR2}) + D_{AO222} \quad (3)$$

In Eq. (1), on the right side, the first term denotes the delay encountered in the first 4-bit CLA, the second term denotes the delay encountered in six intermediate 4-bit CLAs, and the third term denotes the delay encountered in the last 4-bit CLA. Eq. (2) is like Eq. (1) but the extra fourth term present in Eq. (2) represents the delay of a majority voter. Eq. (3) expresses the critical path delay of the MSP of RESAC since the HOLSP and MSP are connected by just a 2-input AND gate whose output serves as the carry input to the MSP, as seen in Fig. 2. In Eq. (3), on the right side, the first term represents the delay encountered in the first 4-bit CLA (since the delay of the 2-bit CLA is subsumed in the 4-bit CLA delay), the second term represents the delay encountered in three intermediate 4-bit CLAs, the third term represents the delay encountered in the last 4-bit CLA, and the fourth term represents the delay of a majority voter. Based on the typical delay information given in the datasheet [18], the theoretical values of $D_{Simplex}$, D_{TMR}, and D_{RESAC} are calculated as 0.781 ns, 0.887 ns, and 0.698 ns respectively. The theoretical delays exhibit a similar trend as the practical delay estimates given in Table 1 for the adders.

Table 1
Design metrics of simplex and redundant implementations synthesized using a 28-nm CMOS standard cell library.

Implementation	Delay (ns)	Area (μm²)	Power (μW)
Simplex	1.13	527.45	91.7
TMR	1.24	1752.43	291.8
RESAC	1.05	1362.21	219.0

In the RESAC implementation, MSP and HOLSP are triplicated but LOLSP is maintained as simplex. Moreover, HOLSP and LOLSP of RESAC contain reduced logic due to approximation (as seen in Fig. 2). In comparison, an accurate functional unit (here an accurate adder) is fully triplicated in TMR. Further, RESAC requires one majority voter less than TMR. Hence, RESAC achieves a reduction in area, and thus a reduction in power compared to TMR, as seen in Table 1. Compared to the simplex implementation, RESAC achieves a 7 % reduction in delay, occupies 1.6× more area, and dissipates 1.4× more power. Compared to TMR, RESAC achieves reductions in area, delay, and power by 22.3 %, 15.3 %, and 24.9 % respectively while enabling an acceptable output quality (as seen in Fig. 3).

6. Conclusions

This paper presented a novel redundancy strategy based on approximate computing called RESAC that can be an alternative to TMR for inherently error-tolerant applications. RESAC is found to facilitate significant optimization in the design metrics compared to TMR while enabling an acceptable output quality. Arithmetic functions with varying significance attached to their output bits are ideal candidates for a RESAC-based implementation. We demonstrated the usefulness of RESAC for a digital image processing application in this work where the processed images have PSNR > 30 dB and SSIM close to unity, which are acceptable. For the physical implementation of a functional unit (here adder), RESAC achieves a 36.5 % reduction in the power-delay product (that is representative of energy efficiency) compared to TMR, and a 50.6 % reduction in the power-delay-area product (that is representative of design efficiency) compared to TMR.

CRediT authorship contribution statement

Padmanabhan Balasubramanian : Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Validation, Visualization, Writing – original draft, Writing – review & editing. **Douglas L. Maskell :** Formal analysis, Funding acquisition, Investigation, Project administration, Resources, Supervision. **Krishnamachar Prasad :** Formal analysis, Methodology, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research was partially funded by the Singapore Ministry of Education (MOE) Academic Research Fund under grant numbers Tier-1 RG48/21 and Tier-1 RG127/22.

References

[1] R.C. Baumann, Radiation-induced soft errors in advanced semiconductor technologies, IEEE Trans. Device Mater. Reliab. 5 (2005) 305–316.
[2] N.N. Mahatme, et al., Terrestrial SER characterization for nanoscale technologies: a comparative study, in: Proc. IEEE International Reliability Physics Symposium, 2015, pp. 4B.4.1–4B.4.7.
[3] T. Iniewski, Radiation Effects in Semiconductors, CRC Press, Boca Raton, FL, USA, 2011.
[4] T. Ban, L. Naviner, Progressive module redundancy for fault-tolerant designs in nanoelectronics, Microelectron. Reliab. 51 (2011) 1489–1492.
[5] I. Koren, C.M. Krishna, Fault-tolerant Systems, Morgan Kaufmann Publishers, Burlington, MA, USA, 2007.
[6] H. Quinn, et al., Radiation-induced multi-bit upsets in SRAM-based FPGAs, IEEE Trans. Nucl. Sci. 52 (2005) 2455–2461.
[7] S. Mittal, A survey of techniques for approximate computing, ACM Comput. Surv. 48 (2016), 62 (1–33).
[8] O. Ruano, J.A. Maestro, P. Reviriego, A methodology for automatic insertion of selective TMR in digital circuits affected by SEUs, IEEE Trans. Nucl. Sci. 56 (2009) 2091–2102.
[9] I.A.C. Gomes, et al., Exploring the use of approximate TMR to mask transient faults in logic with low area overhead, Microelectron. Reliab. 55 (2015) 2072–2076.
[10] T. Arifeen, et al., Input vulnerability-aware approximate triple modular redundancy: higher fault coverage, improved search space, and reduced area overhead, Electron. Lett. 54 (2019) 934–936.
[11] A. Ullah, et al., Majority voting-based reduced precision redundancy adders, IEEE Trans. Device Mater. Reliab. 18 (2018) 122–124.
[12] P. Balasubramanian, N.E. Mastorakis, Power, delay, and area comparisons of majority voters relevant to TMR architectures, in: Proc. 10th International Conference on Circuits, Systems, Signal and Telecommunications, 2016, pp. 110–117.
[13] https://imageprocessingplace.com/root_files_V3/image_databases.htm. (Accessed 13 February 2023).
[14] P. Balasubramanian, R. Nayar, D.L. Maskell, An approximate adder with reduced error and optimized design metrics, in: Proc. 17th IEEE Asia Pacific Conference on Circuits and Systems, 2021, pp. 21–24.
[15] Al Bovik, Handbook of Image and Video Processing, 2nd edition, Academic Press, Orlando, FL, USA, 2005.
[16] W. Zhou, et al., Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (2004) 600–612.
[17] P. Balasubramanian, N.E. Mastorakis, High-speed and energy-efficient carry look-ahead adder, J. Low Power Electron. Appl. 12 (2022) 1–11 (46).
[18] Synopsys SAED_EDK32/28_CORE Databook, Revision 1.0.0, January 2012.