Received July 4, 2018, accepted August 17, 2018, date of publication August 30, 2018, date of current version September 21, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2868036*

# Squash: Approximate Square-Accumulate With Self-Healing

**G. A. GILLANI[1], MUHAMMAD ABDULLAH HANIF[2], M. KRONE[1], S. H. GEREZ[1], MUHAMMAD SHAFIQUE[2], (Senior Member, IEEE), AND A. B. J. KOKKELER[1]**
[1]Faculty of EEMCS, University of Twente, Enschede 7500 AE, The Netherlands
[2]Faculty of Informatics, Vienna University of Technology (TU Wien), 1040 Vienna, Austria

Corresponding author: G. A. Gillani (s.ghayoor.gillani@utwente.nl)

This work was supported in part by the ASTRON and IBM Joint Project, DOME, through the Netherlands Organization for Scientific Research, in part by the Dutch Ministry of EL&I, and in part by the Province of Drenthe.

**ABSTRACT** Approximate computing strives to achieve the highest performance-, area-, and power-efficiency for a given quality constraint and vice versa. Conventional approximate design methodology restricts the introduction of errors to avoid a high loss in quality. However, this limits the computing efficiency and the number of pareto-optimal design alternatives for a quality-efficiency tradeoff. This paper presents a novel self-healing (SH) methodology for an approximate square-accumulate (SAC) architecture. SAC refers to a hardware architecture that computes the inner product of a vector with itself. SH exploits the algorithmic error resilience of the SAC structure to ensure an effective quality-efficiency tradeoff, wherein the squarer is regarded as an approximation stage, and the accumulator as a healing stage. We propose to deploy an approximate squarer *mirror pair*, such that the error introduced by one approximate squarer *mirrors* the error introduced by the other, i.e., the errors generated by the approximate squarers are approximately the additive inverse of each other. This helps the healing stage (accumulator) to automatically average out the error originated in the approximation stage, and thereby to minimize the quality loss. For random input vectors, SH demonstrates up to 25% and 18.6% better area and power efficiency, respectively, with a better quality output than the conventional approximate computing methodology. As a case study, SH is applied to one of the computationally expensive components (SAC) of the radio astronomy calibration application, where it shows up to 46.7% better quality for equivalent computing efficiency as that of conventional methodology.

**INDEX TERMS** Approximate computing, approximate multiplier, approximate squarer, multiply-accumulate, radio astronomy, self-healing, square-accumulate.

## I. INTRODUCTION

Approximate Computing has attained remarkable attention for its potential to increase computing efficiency in terms of power, area and performance [1]. It has shown significant benefits for error resilient applications like multimedia processing and machine learning, by leveraging a controlled in-accuracy in the overall output [2], [3]. In approximate computing the relaxed accuracy requirements are used in order to trade-off computational quality with efficiency [1]–[5]. The design target in approximate computing is to achieve the highest computing efficiency for a given quality constraint or to achieve the highest quality for a given efficiency requirement.

The state-of-the-art approximate design methodologies suggest to utilize *fail-rare*, *fail-small*, and *fail-moderate* approaches, where the approximations are restricted in terms of introducing errors and thereby limiting the computing efficiency. The fail-rare strategy suggests that the approximation technique should result in a low error rate but may exhibit high error magnitudes [6]. On the other hand, fail-small refers to introducing low error magnitudes with high error rates [6]. Fail-moderate suggests to utilize an additional design space of approximations, wherein the errors introduced may also exhibit moderate error rates and moderate error magnitudes [7]. It is important to note that the aforesaid strategies limit the design space as they do not allow approximations that introduce high error rates with high error magnitudes. The reason being obvious that this threatens the quality loss hugely in case of general algorithms, and if employed naively.

However, we argue that there are several algorithms like square-accumulate (SAC), multiply-accumulate (MAC) and Least Squares (LS), for which the approximations introducing simultaneous high error magnitudes and high error rates can also be utilized provided that the errors originated in various sub-components are potentially canceled out to minimize the overall quality loss (a *fail-balanced* approach) while achieving a higher efficiency. This increases the design space by introducing a higher number of pareto-optimal approximate design alternatives to help effectively exploit the quality-efficiency trade-off.

This paper presents a Self-Healing (SH) methodology that exploits the fail-balanced approach for an approximate square-accumulate (SAC) architecture. Specifically, the analysis and design of an approximate SAC is discussed. SAC refers to a hardware architecture that computes the inner product of a vector with itself. Therefore, it is a special case of a multiply-accumulate (MAC), where both inputs of the multiplier are equal. It is one of the computationally expensive components of Least Squares (LS) algorithm in general and radio astronomy calibration [8], [9] in particular. LS is also employed in other Digital Signal Processing (DSP) applications like medical [10], synthetic aperture radar [11] and radioastronomical [12] image-reconstruction.

## A. STATE-OF-THE-ART DESIGNS AND THEIR LIMITATIONS

Approximate designs for adders [13]–[20] and multipliers [21]–[33] have been researched for their indispensable role in DSP. Kulkarni *et al.* [22] presented a low power under-designed $2\times2$ multiplier and showed its efficacy in constructing higher order $(n\times n)$ multipliers, which can trade a bearable quality loss with improved computing efficiency. In order to achieve pareto-optimality, design space exploration of $n \times n$ approximate multipliers is performed in [25], which considers various $2 \times 2$ approximate multipliers to search for an optimized design.

Deploying truncated multiplication in a MAC architecture attracted researchers in the last decade, where the objective was to limit the bit-width of multipliers and lower the error introduced due to truncations [32], [33]. Recent design approaches for approximate MAC present the introduction of an offset to compensate the inaccuracies of the approximate multiplier stage [34] and the utilization of hybrid redundant adders [35]. The aforesaid works presented techniques to approximate the MAC architecture. However, no exploitation of the self-healing approach has been studied to the best of our knowledge. Moreover, despite the importance of the SAC architecture in DSP, approximate SAC designs have not been researched yet.

Radio astronomy studies celestial objects at radio frequencies. The Science Data Processing (SDP) pipeline takes sky visibilities as input and processes (calibration and imaging) them to compute sky images [36]. For modern and future radio telescopes like the Square Kilometer Array (SKA), power consumption—7.2MW when using double-precision fused multiply-add operations (medium frequency antenna array)—and throughput (up to 1.8 EOps/s) are key challenges [36]. Radio astronomy has potentially redundant real life data with low signal to noise ratio and iterative/healing algorithms of approximate nature (e.g. least squares), which motivate to study the quality-efficiency trade-off and employ approximate computing. In order to overcome the power and throughput challenges related to the radio astronomy SDP, FPGA based hardware accelerators [37] and graphics processor based designs [38] have been proposed. Despite the potential of error resilience, no quality-efficiency trade-off has been studied to the best of our knowledge.

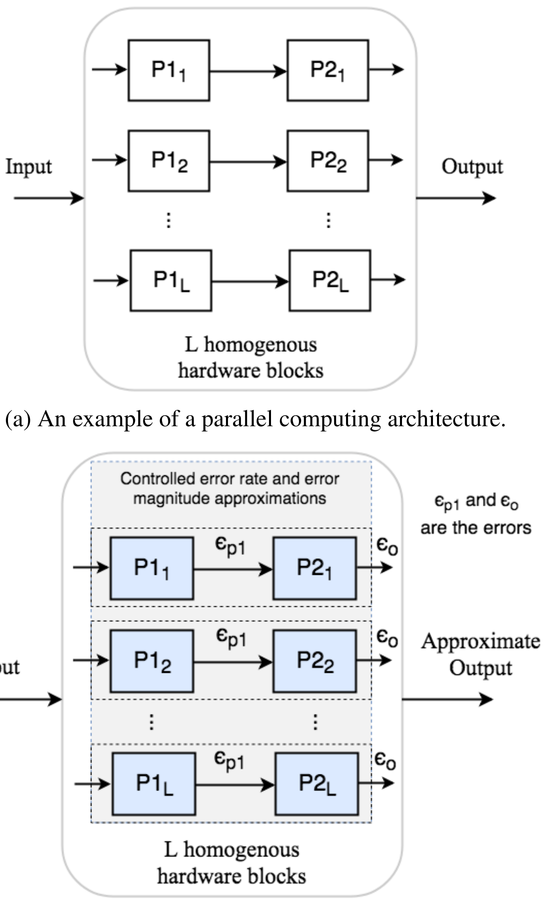
The key limitation of conventional approximate computing techniques deals with its restricted design space, which does not allow employing approximate design alternatives (circuit-, architectural-, algorithm-level) that produce high error rates and high error magnitudes simultaneously. This restriction on design alternatives limits the achievable computing efficiency gains, hindering the exploitation of the quality-efficiency trade-off effectively.

## B. NOVEL CONTRIBUTIONS AND OUTLINE

Consider an example of a parallel computing architecture, Fig. 1a, where the input stream is processed by L homogeneous hardware blocks to generate the output stream. Each hardware block consists of two processing elements, P1 and P2. The processing elements can be considered as arithmetic elements like multipliers and adders. The conventional way of approximating such an architecture is to employ approximate circuits for P1 and P2 in such a way that the error magnitudes and error rates are restricted to avoid an unacceptable loss in quality, Fig. 1b.

The primary contribution of this paper is a novel *Self-Healing* (SH) methodology that aims to utilize a fail-balanced approach wherein an architecture is divided into two types of stages, namely an *approximation stage* and a *healing stage*. Approximations are employed at the approximation stage in such a way that the resulting errors have a potential to be healed up at the healing stage. Therefore, we propose to approximate P1 elements in pairs, such that each P1 in a pair generates an error that is the *mirror* (approximately additive or multiplicative inverse) of the other, Fig. 1c. This minimizes the output error $\epsilon_0$ to zero in some cases and to a lower value (as compared to the conventional methodology) in the other, and provides an effective quality-efficiency trade-off. The hardware cost of a pair is considered to be twice as that of a single P1 element, but overall hardware cost of a parallel architecture remains the same as we require L/2 hardware blocks instead of L for the same throughput. To elaborate on the SH concept, this paper presents the following:

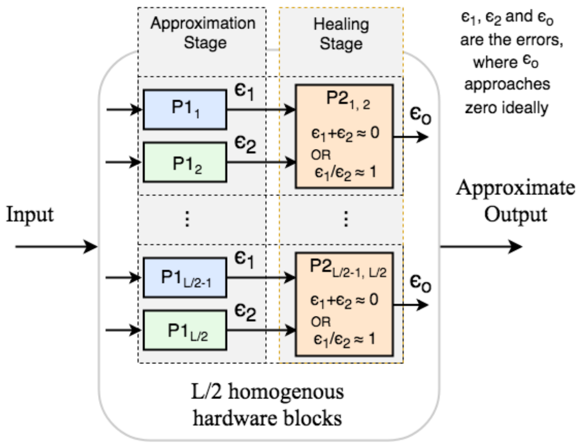
- Related terminology and application of SH for approximate square-accumulate (SAC) architecture design (Section II).
- Analysis of an approximate SAC utilizing a truncated squarer that establishes the foundation for the SH methodology (Section III).
- An $n \times n$ approximate-squarer (AxSq) *mirror pair*, wherein the error introduced by one AxSq ($\epsilon_{s1}$) is an additive inverse of the error introduced by the other ($\epsilon_{s2}$), i.e., $\epsilon_{s1} = -\epsilon_{s2}$ (Section IV).



Input
Output
P1₁
P2₁
P1₂
P2₂
P1_L
P2_L
L homogenous hardware blocks
Controlled error rate and error magnitude approximations
ϵp1 and ϵo are the errors
ϵp1
ϵo
Approximate Output

(a) An example of a parallel computing architecture.

(b) Conventional approximate computing methodology.



(c) Proposed approximate computing methodology.

**FIGURE 1. An overview of the conventional and the proposed approximate computing methodologies for parallel architectures. The proposed Self-Healing (SH) methodology does not restrict the approximations based on an error profile but provides the opportunity for error cancellation to achieve an effective quality-efficiency trade-off.**

- Design of $2 \times 2$ approximate multiplier *mirror pair* and $2 \times 2$ AxSq *mirror pair* that constitute the $n \times n$ AxSq *mirror pair* (Section IV).
- Statistical error analysis of an AxSq *mirror pair* and design space exploration for pareto-optimal approximate SAC alternatives (Section V).
- Quality-efficiency trade-off comparison of the proposed self-healing and conventional methodologies for random input data and for radio astronomy calibration processing (Section V and VI).

## II. SELF-HEALING METHODOLOGY FOR APPROXIMATE SQUARE-ACCUMULATE (SAC)

This section elaborates the Self-Healing (SH) concept by discussing its utilization for an approximate SAC architecture. Wherein the squarer stage is regarded as *approximation stage*, and the accumulator as *healing stage*. First we define the terminology related to SH that will be used in the rest of the paper.

### A. TERMINOLOGY

We define the *Mirror Error Effect* (MEE) as an introduction of errors ($\epsilon_1$, $\epsilon_2$) in a pair of approximate components that has the potential of cancellation (completely or partially) at a subsequent healing stage. For instance, $\epsilon_1$ and $\epsilon_2$ have opposite signs (healing stage: adder) or a common factor (healing stage: divider). Such a pair of components is (proposed to be) called *absolute approximate mirror pair* provided that $\epsilon_1 = -\epsilon_2$ or $\epsilon_1/\epsilon_2 = 1$, otherwise *approximate mirror pair* where the errors are canceled out partially. It is to be noted that the word "absolute" is used in the term *absolute approximate mirror pair*, where it provides a notion that the errors are canceled out completely instead of partially.

In case of the SAC architecture where an accumulator is regarded as a healing stage, a pair of approximate multipliers complying to the MEE is referred to as an *Absolute Approximate Multiplier Mirror Pair* (AAMMP) if it exhibits $\epsilon_1 = -\epsilon_2$, otherwise it is called an *approximate multiplier mirror pair*. Similarly, a pair of approximate squarers complying to the MEE is referred to as an *Absolute Approximate Squarer Mirror Pair* (AASMP) if it exhibits $\epsilon_1 = -\epsilon_2$, otherwise it is called an *approximate squarer mirror pair*.

### B. EMPLOYING SELF-HEALING FOR APPROXIMATE SAC ARCHITECTURE

With reference to Fig. 1a, consider a parallel architecture that computes a Square-Accumulate (SAC) operation where each hardware block is referred to as a SAC architecture. Now P1 can be regarded as a squarer and P2 as an accumulator, illustrated in Fig. 2a. A SAC computes:

$$\sum_{i=1}^{N}(A_i * A_i) \tag{1}$$

where $A$ is an input vector of length $N$. We consider an even number of elements in the input vector to ease the discussion, i.e., $N \in 2\mathbb{Z}_{>0}$. To design an approximate SAC in a self-healing fashion, we propose to approximate the squarer by deploying a pair of approximate squarers (e.g., AASMP) that

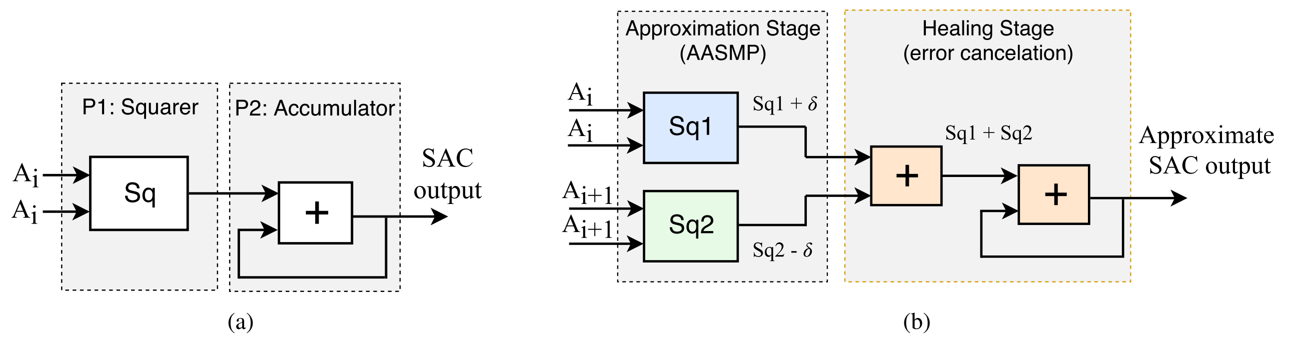


**FIGURE 2.** Square-accumulate (SAC) architectures; (a) accurate; (b) proposed approximate SAC utilizing an Absolute Approximate Squarer Mirror Pair (AASMP). Given the same input distribution for Sq1 and Sq2, the errors ($+\delta$ and $-\delta$) originated at the approximation stage are canceled out at the healing stage.

introduce errors, $+\delta$ and $-\delta$ at Sq1 and Sq2 respectively, and utilize an accurate accumulator to cancel out the errors introduced in squarer stage (Fig. 2b).

Equation (1) can be re-written as,

$$\sum_{i=1}^{N-1}\{(A_i * A_i) + (A_{i+1} * A_{i+1})\} \quad i \in 2\mathbb{Z}_{\geq 0} + 1 \tag{2}$$

where $i$ is an odd positive integer. Equation (2) shows a pair of squarers that can be regarded as Sq1 and Sq2 as illustrated in Fig. 2b. Given the same input distribution for Sq1 and Sq2 (ideal case), the error at the approximate SAC output will approach zero for an infinite number of inputs. However, for non-ideal (real-world) cases, the error is minimized due to partial cancellation. It can be noted that the proposed approximate SAC hardware block (Fig. 2b) doubles the circuit area as compared to a reference SAC (Fig. 2a) hardware block, however, the overall circuit area of a parallel architecture remains the same for the proposed architecture as we require L/2 SAC hardware blocks instead of L for the same throughput.

## III. ANALYSIS OF APPROXIMATE SAC UTILIZING TRUNCATED SQUARER

This section presents the mathematical analysis, and simulation of various truncated square-accumulate (SAC) alternatives that establish the basis for Self-Healing (SH). For a pair of approximate squarers, this paper presents two ways by which the MEE can be achieved. In this section we discuss truncated squaring and in Section IV we discuss logic pruning (reducing number of logic gates) as a mean of approximation.

### A. MATHEMATICAL ANALYSIS OF TRUNCATED SQUARING

We consider the SAC operation executed on $n$ bit words and the $t$ Least Significant Bits (LSBs) are truncated. Here we consider the signed numbers represented in 2's complement. Let $A$ be defined as a random number,

$$-2^{n-1} \leq A < 2^{n-1} \quad A \in \mathbb{Z}$$

For the input pairs $(A, A)$, we can describe all (but one) numbers in the $n$ bit range by either having $(+a, +a)$ for a positive input and $(-a, -a)$ for a negative input, where $a = |A|$. We have two possible cases for the SAC input pairs,

$$\begin{aligned}(A, A) = &(+a, +a)\\ &(-a, -a)\end{aligned}$$

Let $\alpha_t$ represent the input number with the $t$ LSBs truncated, as:

$$\alpha_t = A - m \quad 0 \leq m < 2^t, \ m \in \mathbb{Z}$$

For the input pair $(+a, +a)$ the product of the truncated factors is given by:

$$\begin{aligned}\alpha_t \cdot \alpha_t &= (a - m_1)(a - m_1) \quad 0 \leq m_1 < 2^t, \ m_1 \in \mathbb{Z}\\ &= aa - 2am_1 + m_1m_1 \end{aligned}\tag{3}$$

In (3) the product $aa$ is the accurate result of the squaring. The terms $2am_1$ and $m_1m_1$ are errors introduced by the truncation. Let's assume: $m_1 \ll a$, i.e., the truncation error is much smaller than the original value of the number that was truncated. This is plausible looking at the ranges of $a$ in comparison to $m_1$. The product can now be approximated by:

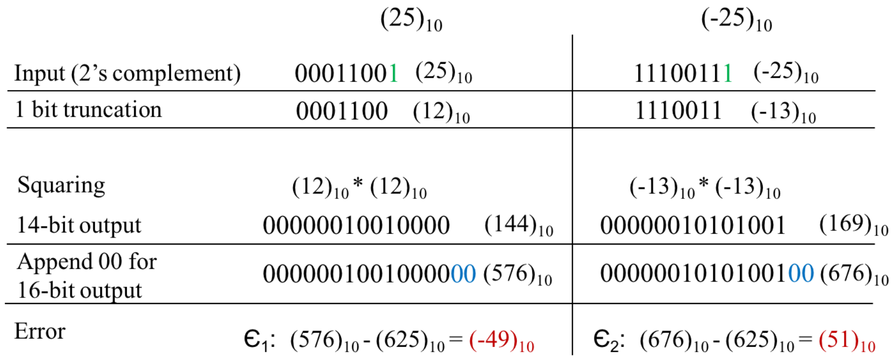
$$\alpha_t \cdot \alpha_t \approx aa - 2am_1 \tag{4}$$

For the input pair $(-a, -a)$,

$$\begin{aligned}\alpha_t \cdot \alpha_t &= (-a - m_2)(-a - m_2) \quad 0 \leq m_2 < 2^t, m_2 \in \mathbb{Z}\\ &= aa + 2am_2 + m_2m_2\\ &\approx aa + 2am_2\end{aligned}\tag{5}$$

Equations (4) and (5) show that one case introduces a negative error (approximately $-2am_1$) and the second case introduces a positive error (approximately $+2am_2$). Assuming that these two cases have equal probability and $m_1 \approx m_2$, we can conclude that the error of the first case approximately cancels the error of the second case at the accumulation stage.

This is an interesting property of the truncated squarer in a SAC architecture, for squaring signed numbers, where the errors higher and lower than zero have a potential to be canceled out at the healing stage (accumulator).



| | $(25)_{10}$ | $(-25)_{10}$ |
|---|---|---|
| Input (2's complement) | 00011001 $(25)_{10}$ | 11100111 $(-25)_{10}$ |
| 1 bit truncation | 0001100 $(12)_{10}$ | 1110011 $(-13)_{10}$ |
| Squaring | $(12)_{10} * (12)_{10}$ | $(-13)_{10} * (-13)_{10}$ |
| 14-bit output | 00000010010000 $(144)_{10}$ | 00000010101001 $(169)_{10}$ |
| Append 00 for 16-bit output | 0000001001000000 $(576)_{10}$ | 0000001010100100 $(676)_{10}$ |
| Error | $\epsilon_1$: $(576)_{10} - (625)_{10} = (-49)_{10}$ | $\epsilon_2$: $(676)_{10} - (625)_{10} = (51)_{10}$ |

**FIGURE 3.** Truncated squaring of a number as a positive integer and as a negative integer demonstrate the Mirror Error Effect (MEE). The subsequent accumulation can cancel out the errors partially and improves the overall output quality.

## B. QUALITY ANALYSIS OF VARIOUS TRUNCATION ALTERNATIVES

In case of squaring, the product is always a positive number. Therefore, we have a choice to invert the signs of inputs (multiplicands) without affecting the output. The MEE can be achieved by utilizing an approximate squarer pair (Sq1 and Sq2), where Sq1 squares the truncated input numbers that are made positive (before truncation) while Sq2 squares the truncated input numbers that are made negative (before truncation) to form a *mirror pair* as,

**Sq1:** $A \rightarrow ((|A|)_t)^2$
**Sq2:** $A \rightarrow ((-|A|)_t)^2$

where $A$ is an input, and the subscript $t$ denotes a truncation operation. The motivation behind this proposal is that on average the amount of positive number squaring equals negative number squaring which provides an opportunity to cancel out the truncation error.

A motivational example of 1-bit truncated squaring of an 8-bit integer ($n = 8$, $t = 1$) is shown in Fig. 3, where the errors are shown for squaring the integer as a positive ($(+25)_{10}$) and a negative ($(-25)_{10}$) number after truncation. The truncated squaring produces the output as a 14-bit integer that is shifted left (2-bit) to achieve a 16-bit output. The errors ($\epsilon_1 = -49$ and $\epsilon_2 = 51$) show the mirror effect that have a potential to be approximately canceled out at the accumulator.

In the above example, $(00)_2$ is appended to the least significant position after squaring in order to produce the 16-bit output. However, any 2-bit combination can be hard-wired to the least significant position targeting the lowest error output. Below we consider design alternatives for 1-bit truncated squaring of 8-bit input integers ($n = 8$, $t = 1$) and compare their output quality in a SAC architecture.

**Actual_s**: A conventional approximate design that squares the truncated operands as they are originally fed (without changing sign) and appends 00 to form a 16-bit output.

**Pos_xx**: A design alternative that makes every input a positive integer before truncating and computing the square operation. The example (Fig. 3) illustrates that truncated squaring of positive integers produces an output smaller than (or in some cases equal to) the exact result. For appending two bits to the 14-bit squarer output, five options are considered: **Pos_00**, **Pos_01**, **Pos_10**, **Pos_11**, and **Pos_LL** that append $(00)_2$, $(01)_2$, $(10)_2$, $(11)_2$ and $(LL)_2$ respectively to produce a 16-bit output. Here $(LL)_2$ is the two times repetition of the truncated 1-bit.

**Neg_00**: A design alternative that makes every input a negative integer before truncating, and appends 00 to produce the 16-bit output. As the example (Fig. 3) shows that truncated squaring of negative integers produces output larger than (or in some cases equal to) the exact result, we do not consider appending 01, 10, 11, and LL because it will increase the error.

**MEEx**: Mirror error effect designs (MEE1 and MEE2) square the truncated operand in the proposed self-healing fashion, where half of the inputs are made positive integers and the other half are made negative integers before truncating and squaring. It is to be noted that Sq1 and Sq2 in MEEx designs provide errors in opposite signs but not in equal magnitudes. Therefore, swapping Sq1 with Sq2 brings different quality for finite-length input vectors. For that reason, we consider both design alternatives, **MEE1**: odd-indexed elements of an input vector are considered as positive and even-indexed as negative integers, **MEE2**: odd-indexed elements of an input vector are considered as negative and even-indexed as positive integers.

Fig. 4 shows a quality comparison for the above design choices in a SAC architecture. Inputs are considered as signed numbers represented in 2's complement. Two input distributions have been assessed: uniform (Uniform) and normal (Norm_x). Each distribution has 1000 vectors ($v = 1000$) of 10,000 elements each. We have utilized the Mean Square Error (MSE) metric [32], [33] to compute the error in dB as in [39],

$$\mathrm{MSE\ (dB)} = 10 * \log_{10}\Big[\sum_{i=1}^{v}(Ex_i - Ax_i)^2/v\Big] \qquad (6)$$

where $Ex_i$ is the exact SAC output and $Ax_i$ is the approximate counterpart for the $i^{th}$ vector; and $v$ is the number of test

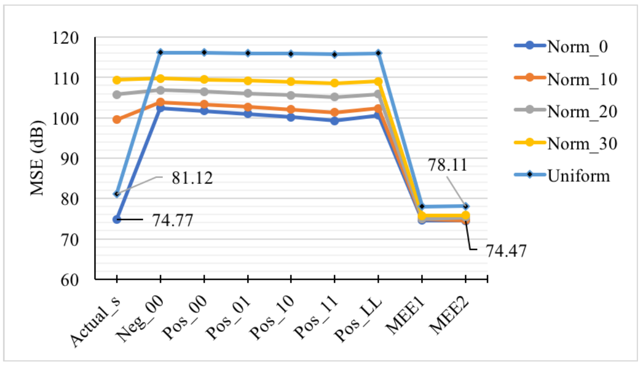


**FIGURE 4.** **Quality analysis for approximate SAC utilizing various truncated squaring strategies. For every considered input distribution, MEE designs are outperforming because of error cancellation at the accumulator stage.**

vectors. The normal distribution has been analyzed with various mean values: 0, 10, 20 and 30. For instance, Norm_0 refers to a normal distribution with 0 mean and Norm_30 to a mean of 30 in Fig. 4. It can be noted that self-healing based designs utilizing the MEE (MEE1 and MEE2) bring the best quality for all considered input distributions. In case of a Norm_0 input, MEEx designs have slightly better quality as compared to Actual_s design. However, as the mean is shifted away from zero (Norm_10, Norm_20, Norm_30), the advantage of MEEx designs is quite significant. This is because, a normal distribution with 0 mean (ideal case) is more likely to have an equal number of positive and negative integers which inherently produces the *mirror error effect* within a conventional design. However in general cases, where input distributions of various mean values can be possible, only self-healing based designs like MEE1 and MEE2 ensure efficient error cancellation to provide the best quality output.

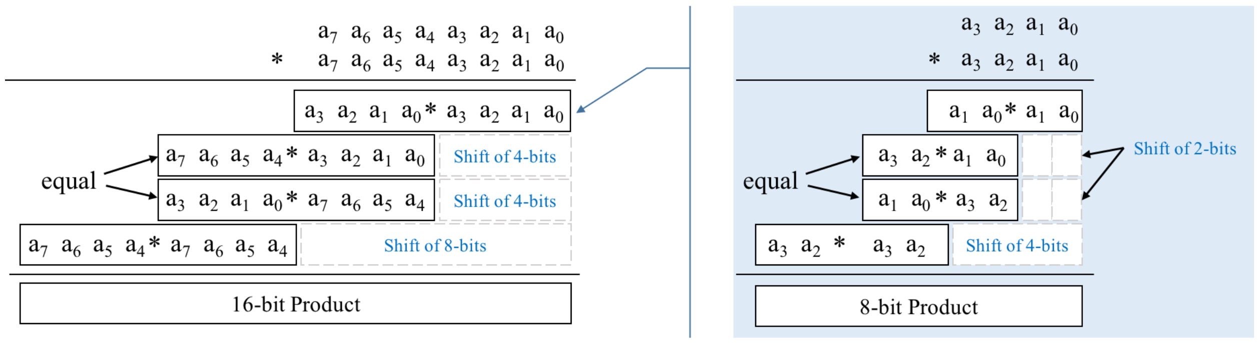
The above analysis provides the foundation of the self-healing methodology that shows better quality output as compared to the conventional way of applying approximations. However, it is important to note that for truncated designs, the MEE is achieved in the sign of errors, i.e., the errors have opposite signs, while the magnitudes are still unequal. Therefore, an *approximate mirror pair* can be formed that partially cancels out the error. Secondly, inverting signs of input operands may render hardware costs depending upon the data representation. In the following section, we discuss how to achieve the MEE for logic-pruned (approximate) $n \times n$ squarers that utilize $2 \times 2$ squarers and $2 \times 2$ multipliers in order to produce *absolute approximate mirror pairs*, which cancel out the error originated within the pairs completely.

## IV. ABSOLUTE APPROXIMATE SQUARER MIRROR PAIR (AASMP)

An $n \times n$ recursive multiplier can be constructed from $4^{m-1}$ elementary $(2 \times 2)$ multipliers, where $n = 2^m$ is the width of input operands A and B in bits [22], [25]. These $2 \times 2$ elementary multipliers form the partial products and summing the bit-shifted partial products produces the overall product by utilizing adder trees. Any number out of the $2 \times 2$ partial products and/or adders can be approximated to achieve an approximate $n \times n$ multiplier [25].

In case of an $n \times n$ squarer, the number of required elementary $(2 \times 2)$ modules is less than $4^{m-1}$ (where $n = 2^m$) due to the repetition of a few partial products with the same inputs. The only exception is a $2 \times 2$ squarer ($n = 2$). Without loss of generality, here we consider an $8 \times 8$ unsigned squarer (Sq8×8) to design an approximate *mirror pair*, wherein the error introduced by one approximate squarer is additive inverse (opposite in sign and equal in magnitude) of the other. Therefore, the pair is called AASMP. In order to achieve an approximate Sq8×8, we have employed approximations in $2 \times 2$ partial products constructs only, not for the adder trees.

First we discuss the construction of an accurate Sq8×8 module based on elementary $(2 \times 2)$ modules. Let Sq8×8 = A*A, where A = $a_7a_6$ $a_5a_4$ $a_3a_2$ $a_1a_0$ is an 8-bit unsigned number and $a_7$ and $a_0$ are the most significant and least significant bits respectively. Fig. 5 illustrates the Sq8×8 computation, which shows that a total of four $4 \times 4$ partial products are required. Out of the four $4 \times 4$ partial products, two compute 4-bit square (Sq4×4) operation and the other two compute 4-bit multiply (P4×4) operation.



$a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$
* $a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$
$a_3$ $a_2$ $a_1$ $a_0$* $a_3$ $a_2$ $a_1$ $a_0$
$a_7$ $a_6$ $a_5$ $a_4$* $a_3$ $a_2$ $a_1$ $a_0$
Shift of 4-bits
equal
$a_3$ $a_2$ $a_1$ $a_0$* $a_7$ $a_6$ $a_5$ $a_4$
Shift of 4-bits
$a_7$ $a_6$ $a_5$ $a_4$* $a_7$ $a_6$ $a_5$ $a_4$
Shift of 8-bits
16-bit Product
$a_3$ $a_2$ $a_1$ $a_0$
* $a_3$ $a_2$ $a_1$ $a_0$
$a_1$ $a_0$* $a_1$ $a_0$
$a_3$ $a_2$* $a_1$ $a_0$
Shift of 2-bits
equal
$a_1$ $a_0$* $a_3$ $a_2$
$a_3$ $a_2$ * $a_3$ $a_2$
Shift of 4-bits
8-bit Product

**FIGURE 5.** **Sq8×8 computation utilizing 4 × 4 partial products (left), each 4 × 4 partial product can be computed by deploying 2 × 2 partial products (right).**

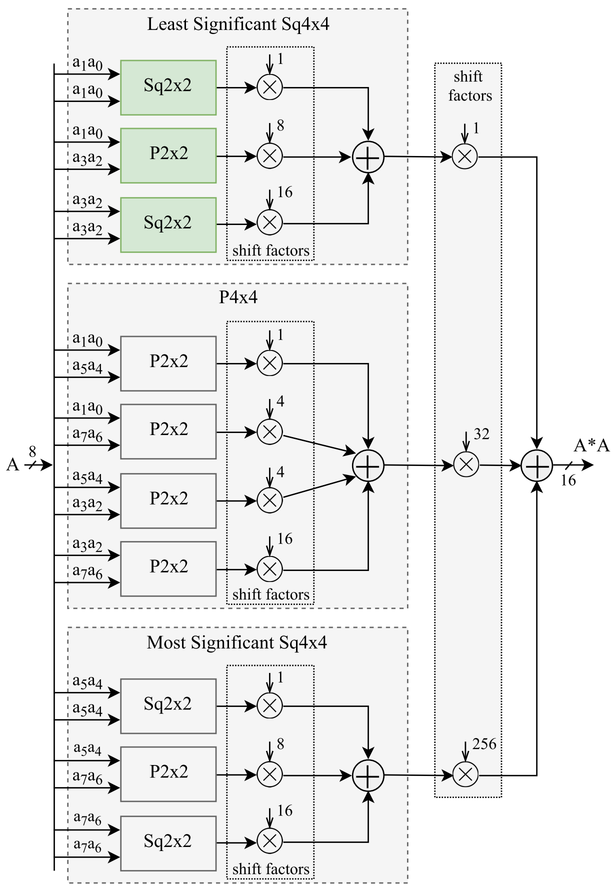


**FIGURE 6.** An 8 × 8 Squarer (Sq8×8) construction utilizing ten elementary (2 × 2) modules, with four squarers (Sq2×2) and six general multipliers (P2×2).

However, the two P4×4 operations multiply the same (equal) inputs, Fig. 5 (left). Therefore, an Sq8×8 hardware requires a P4×4 and two Sq4×4 blocks along with an adder tree.

Similarly, each 4 × 4 partial product can be computed by utilizing four 2×2 partial products, Fig. 5 (right). In case of an Sq4×4 operation, two out of the four 2 × 2 partial products compute 2-bit square (Sq2×2) operation and the other two compute 2-bit multiply (P2×2) operation. However, both P2×2 multiply the same (equal) inputs, Fig. 5 (right). Therefore, an Sq4×4 hardware block requires a P2×2 and two Sq2×2 elementary modules along with an adder tree. On the other hand, a P4×4 hardware block requires four P2×2 elementary modules along with the adder tree. This explains why we require less number of elementary (2 × 2) modules for an $n \times n$ squarer as compared to a general multiplier that requires $4^{m-1}$ elementary modules. Fig. 6 illustrates the construction of an Sq8×8 architecture that requires ten elementary (2 × 2) modules. Out of the ten elementary modules, four compute 2 × 2 square operation (Sq2×2) and 6 compute 2 × 2 multiply operation (P2×2), see Appendix A for the details.

In order to achieve an approximate Sq8×8, any number out of the ten elementary modules can be approximated based upon the error tolerance of an application. To design an

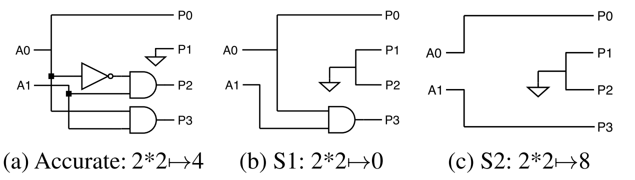


**FIGURE 7.** Logic diagrams of 2 × 2 squarers; (a) accurate design; (b) proposed approximate design S1; (c) proposed *absolute approximate mirror* of S1.

8 × 8 AASMP, we propose to utilize approximate P2×2 and Sq2×2 modules in one Sq8×8 and to utilize the *mirror* approximate counterparts in the other Sq8×8 to form a pair. Therefore, in order to design an 8×8 AASMP, we first discuss how to design 2 × 2 AASMP and 2 × 2 AAMMP.

### A. 2 × 2 AASMP DESIGN

We propose an approximate 2 × 2 squarer: Sq2×2 (S1) which introduces an error for one out of 4 possible input combinations: 2*2≈0, instead of 4 ($\epsilon_{s1} = -4$). An *absolute approximate mirror* of S1 is S2, which computes 2*2≈8 ($\epsilon_{s2} = +4$). The logic diagrams of the accurate Sq2×2, S1 and S2 are shown in Fig. 7.

### B. 2 × 2 AAMMP DESIGN

Consider M1, an approximate 2 × 2 multiplier (P2×2) that introduces an error for one out of 16 possible input combinations: 3*3≈7, instead of 9 [22], for the error: $\epsilon_{m1} = -2$. To design M2, which is an *absolute approximate mirror* of M1, we propose 3*3≈11, $\epsilon_{m2} = +2$. Therefore, combining M1 and M2 in a parallel pair produces a 2 × 2 AAMMP. The logic diagrams of the accurate P2×2, M1 and M2 are shown in Fig. 8.

### C. DESIGN SPACE OF A 2 × 2 ABSOLUTE APPROXIMATE MIRROR PAIR

Several choices can be made to design an Sq2×2 AASMP or a P2×2 AAMMP. Keeping in view the truth tables, any output (except zero) can be approximated with a $\pm\delta$ error within a pair, i.e., $+\delta$ for one approximate 2 × 2 module and $-\delta$ for the other to form an *absolute approximate mirror pair*. For instance, a P2×2 AAMMP can utilize more design choices such as: 2*1≈1 and 3 ($\epsilon = \pm 1$), 3*2≈5 and 7 ($\epsilon = \pm 1$), or 4 and 8 ($\epsilon = \pm 2$). Comprehensive design space exploration to get an optimal 2 × 2 *absolute approximate mirror pair* is beyond the scope of this paper, we only intend to show here how it can be designed.

### D. 8 × 8 AASMP DESIGN

As discussed earlier, an Sq8×8 architecture is composed of six P2×2 and four Sq2×2 elementary modules (Fig. 6). Approximate elementary modules like M1, M2, S1 and S2 can be utilized to design an approximate Sq8×8. However for an 8×8 AASMP design, we propose to utilize the M1 and S1 modules for one Sq8×8 ($\epsilon_1 = -\delta$), and the M2 and S2 modules for the other Sq8×8 ($\epsilon_2 = +\delta$) to form a pair.

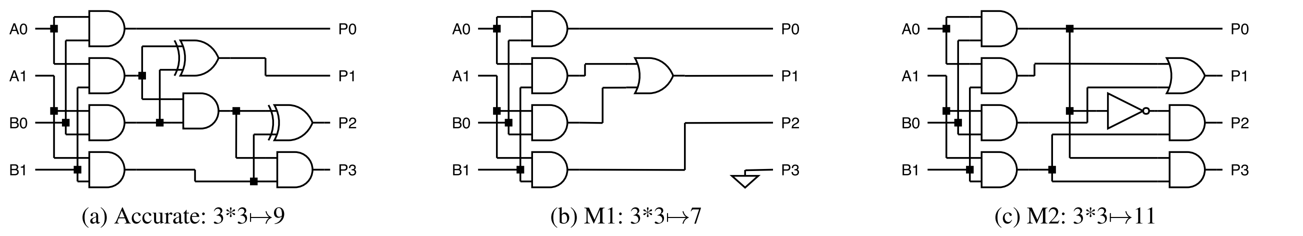


**FIGURE 8.** Logic diagrams of 2 × 2 multipliers; (a) accurate design; (b) an approximate design M1 [22]; (c) proposed *absolute approximate mirror* of M1.

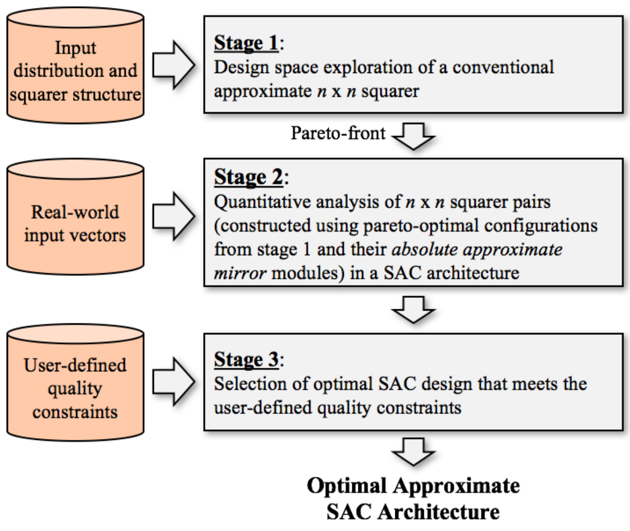

Input distribution and squarer structure
Stage 1:
Design space exploration of a conventional approximate n x n squarer
Pareto-front
Real-world input vectors
Stage 2:
Quantitative analysis of n x n squarer pairs (constructed using pareto-optimal configurations from stage 1 and their absolute approximate mirror modules) in a SAC architecture
User-defined quality constraints
Stage 3:
Selection of optimal SAC design that meets the user-defined quality constraints
Optimal Approximate SAC Architecture

**FIGURE 9.** Design methodology for building an optimal approximate Square-Accumulate (SAC) architecture.

### E. $n \times n$ AASMP DESIGN

We have elaborated on the design of an $8 \times 8$ AASMP so far. In a similar fashion, an $n \times n$ AASMP can be designed by utilizing the negative error elementary $2 \times 2$ designs ($\epsilon = -\delta$, like M1 and S1) for one $n \times n$ squarer and the positive error elementary designs ($\epsilon = +\delta$, like M2 and S2) for the other $n \times n$ squarer to form a pair.

## V. DESIGN SPACE EXPLORATION OF AN APPROXIMATE SAC ARCHITECTURE AND COMPARISON OF SELF-HEALING WITH CONVENTIONAL METHODOLOGY

In this section, we present a design flow for building an optimal approximate square-accumulate (SAC) architecture. Fig. 9 illustrates the proposed design methodology. As can be seen in the figure, the first step involves a design space exploration of a conventional approximate $n \times n$ squarer using statistical analysis (Section V-A). The pareto-optimal design configurations from the first stage are then fed to the second stage where $n \times n$ squarer pairs are formed by employing *absolute approximate mirror* modules of the pareto-optimal configurations. Quantitative analysis of the pairs is then performed using exhaustive simulations (covered in section V-B), to quantify the performance of the pairs in real world scenarios that involve random input vectors of finite (limited) length. We present these simulations for both self-healing and conventional methodologies to compare their overall quality-efficiency trade-off. Based on the quantitative analysis in Stage 2 and the user-defined quality constraints, an approximate configuration can be selected that provides optimal efficiency while satisfying the user-constraints.

### A. DESIGN SPACE EXPLORATION USING ERROR PROBABILITY MASS FUNCTION (EPMF) OF AN APPROXIMATE SQUARER

First of all we present the statistical error analysis for the design space exploration of a single approximate squarer (composed of elementary $2 \times 2$ multiplier and squarer modules) designed in a conventional way (Stage1). By conventional way, we mean that we are only using M1 as approximate $2 \times 2$ multiplier and S1 as approximate $2 \times 2$ squarer, without the *mirror* approximate designs (M2 and S2). To evaluate the quality of all the possible configurations of an approximate squarer, we propose an algorithm that can be used to compute the probability mass function of error (EPMF) [19], [28] of an approximate squarer configuration, provided an input distribution. As the computed EPMF can be used for evaluating most of the error metrics (like Mean Error (ME), Mean Error Distance (MED), and Mean Square Error (MSE)), the proposed algorithm can be used for comparing all the possible configurations of an approximate squarer for a defined error metric.

Let $n$ be the number of input bits. Based upon the fact that the elementary modules used in this work are $2 \times 2$ multipliers and $2 \times 2$ squarers, the input can be divided into pairs of bits, starting from the LSB, as shown in Fig. 10. Each input pair can attain one of three possible states, which defines whether the particular state will/can/will not lead to error(s) in the corresponding $2 \times 2$ approximate elementary module(s). The states are defined as follows:

- State 0 : If in this state, the input pair will not generate rather kill any possibility of error(s) in the corresponding elementary module(s).
- State 1 : If in this state, the input pair will generate an error in the corresponding approximate $2 \times 2$ squarer module only.
- State 2 : If in this state, the input pair can only generate an error in the corresponding approximate

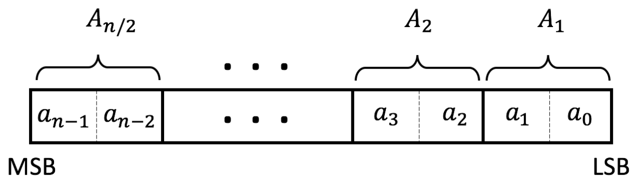


**FIGURE 10.** Pairing of bits for an $n$-bit input. $A_1, A_2, \ldots, A_{n/2}$ are utilized by elementary $2 \times 2$ modules in an $n \times n$ squarer.

$2 \times 2$ multiplier module(s), depending on the state of the other pair(s) of bits being used by the corresponding elementary module(s).

It can be inferred from the designs of the approximate elementary modules (M1 and S1), presented in section IV, that an input pair will be considered in state 0 if it has either $(00)_2$ or $(01)_2$ value, as in all the designs these values do not lead to any approximation error. An input pair is considered in state 1 if it has $(10)_2$ value, as this will always generate an error in case of an approximate $2 \times 2$ squarer (S1) being used at the corresponding location. Similarly, an input pair is considered in state 2 if it has $(11)_2$ value, as this might generate an error in a $2 \times 2$ approximate multiplier (M1) module, depending on the state of the other pair of bits being input to the module.

Given an input probability distribution, the probabilities of each state can be computed as follows:

- State 0 :

$$P(s(A_i) = 0) = P(A_i == (00)_2 \,||\, A_i == (01)_2) \quad (7)$$

- State 1 :

$$P(s(A_i) = 1) = P(A_i == (10)_2) \quad (8)$$

- State 2 :

$$P(s(A_i) = 2) = P(A_i == (11)_2) \quad (9)$$

Here $s(A_i) \in \{0, 1, 2\}$ defines the state of $i^{th}$ input pair of bits ($A_i$), where $i \in \{1, 2, \ldots, n/2\}$.

Using the probabilities of individual states of the input pairs, assuming that they are independent, the probability of a combination of input pairs can be computed as:

$$\begin{aligned} &P(\{s(A_1), s(A_2), \ldots, s(A_{n/2})\} = \{s_1, s_2, \ldots, s_{n/2}\}) \\ &\quad = P(s(A_1) = s_1, s(A_2) = s_2, \ldots, s(A_{n/2}) = s_{n/2}) \\ &\quad = \prod_{i=1}^{n/2} P(s(A_i) = s_i) \end{aligned} \quad (10)$$

To compute the probability of a combination of input pairs more precisely, a modified version of the equation presented in [19] can be used. The equation for the given scenario can be reformulated as:

$$\begin{aligned} &P(\{s(A_1), s(A_2), \ldots, s(A_{n/2})\} = \{s_1, s_2, \ldots, s_{n/2}\}) \\ &= P(s(A_1) = s_1, s(A_2) = s_2, \ldots, s(A_{n/2}) = s_{n/2}) \\ &= \sum_{\{a_{n-1},\ldots,a_0 | s(A_{n/2})=s_{n/2} \wedge \ldots \wedge s(A_1)=s_1\}} P((a_{n-1}\ldots a_0)_2) \end{aligned} \quad (11)$$

Note that as (11) allows us to consider the interdependency between pairs, therefore, the results generated using (11) are more accurate than the results generated using (10). Although at first it seems that (11) will result in significant computational overheads, this is not the case for design space exploration. This is because of the fact that the probability of the combinations of input pairs remain the same for all the configurations of the approximate squarer for a given input probability distribution, and is required to be computed only once. Therefore, the overhead when distributed over all the configurations results in insignificant overhead.

The error corresponding to the combination of input pairs can be computed by identifying the elementary module(s) that will generate error(s) and then adding the errors from all the modules together. Note that here by the error of an elementary module we mean the approximation error multiplied by the significance (shift factor) of the module. Algorithm 1 presents the pseudo-code for computing the error value ($v_c$) for a given combination of input pairs.

**Algorithm 1** Pseudo-Code for Computing $v_c$

```
Input:
    c : State combination that defines the state of each pair of input bits
    Config : Configuration of the approximate squarer
Initialize:
    $v_c = 0$

1:  for $i = \{1, \ldots, length(c)\}$ do
2:      if $c(i)$ == 1 & corresponding $2 \times 2$ sq. is approx. then
3:          $v_c = v_c +$ approximation error $\times$ significance of the corresponding module
4:      else if $c(i) == 2$ then
5:          for $j = \{i+1, i+2, \ldots, length(c)\}$ do
6:              if $c(j) == 2$ then
7:                  $v_c = v_c +$ approximation error $\times$ significance of the corresponding module
8:              end if
9:          end for
10:     end if
11: end for
12: RETURN $v_c$
```

The EPMF for a given configuration can be computed by iterating over all the possible combinations of the input pairs. Algorithm 2 presents the pseudo-code for computing the EPMF of a given configuration of a squarer, provided an input distribution. The arrays $V$ and $P_v$, returned by the algorithm, represent the EPMF where $V$ stores the error values and $P_v$ stores the corresponding error probabilities. Note that, although the computational complexity of the proposed algorithm is exponential, this results in significant reduction in the design space exploration time because it reduces the number of states per input pair from 4 (total number of input

**Algorithm 2** Pseudo-Code for Computing EPMF

```
Input:
    $n$ : Number of input bits
    $Config$ : Configuration of the approximate multiplier
    $P(a_i)$ : Probabilities of individual bits of the input
Initialize:
    $V$ : Array for storing error magnitudes
    $P_v$ : Array for storing error probabilties

1:  Compute $P(s(A_i) = s_i)\ \forall\ A_i \in \{1, 2, \ldots, n/2\}$ and $s_i \in \{0, 1, 2\}$ using Eq. 7, 8, and 9.
2:  Find $C$, i.e., All possible combinations of states of the pairs of input bits
3:  for $c = \{1, \ldots,$ number of combinations in $C\}$ do
4:      Compute $v_c$ (error value of the combination) using Algo. 1
5:      Compute $\rho_c$ (probability of the combination) using Eq. (10) or (11)
6:      if $v_c \in V$ then
7:          $P_v(v_c) = P_v(v_c) + \rho_c$;
8:      else
9:          Append $v_c$ to $V$ and place $\rho_{v_c}$ in $P_v(v_c)$
10:     end if
11: end for
12: RETURN $V$ and $P_v$
```

combinations, i.e., $2^2$) to 3 (because of 3 states) which thereby reduces the computational time significantly specifically for larger approximate squarers.

Note that, as mentioned earlier, Algorithm 2 provides the EPMF of a configuration which can be used to compute almost all the commonly used error metrics. Therefore, a complete design space, covering all the possible configurations of approximate squarer, can be generated for a given input distribution to identify the pareto-optimal configurations which provides an optimal trade-off between a defined error metric and an efficiency target (area/power/performance). We employed the algorithm to perform the design space exploration of an approximate $8 \times 8$ squarer (Sq$8\times8$). Fig. 11 illustrates the design space of an Sq$8\times8$ plotted against area and Mean Error (ME) for uniform and normal input distributions. Here we assume the efficiency target is area which is computed by adding the areas of individual $2 \times 2$ multiplier/squarer modules, synthesized at 1.43GHz for TSMC 40nm Low Power (TCBN40LP) technology as shown in Table 1. It is to be noted that the ME is an important error metric for the output-quality analysis of an approximate squarer in a SAC architecture, because the subsequent accumulator produces a small overall error for a low ME input by canceling out most of the errors. Therefore, we have utilized ME metric and is computed as follows,

$$ME[normalized] = ME/(2^{2n} - 1) \tag{12}$$

where $2n$ is the number of output bits. Based on the available design space, the pareto-optimal configurations (that provide

**TABLE 1.** Area of elementary $2 \times 2$ multipliers (P$2\times2$) and squarers (Sq$2\times2$).

| **Multiplier/Squarer** | **Area** ($\mu m^2$) |
|---|---|
| Accurate P2x2 | 9.65 |
| Conventional Approx P2x2 (M1) | 7.05 |
| Accurate Sq2x2 | 3.53 |
| Conventional Approx Sq2x2 (S1) | 2.12 |

an optimal trade-off between area and the quality metric as shown in Fig. 11) are selected for later stages (Stage2 and Stage3) of the design process. Note that, to have more accurate analysis for both of the cases, we used (11) for computing the probability in Algorithm 2.

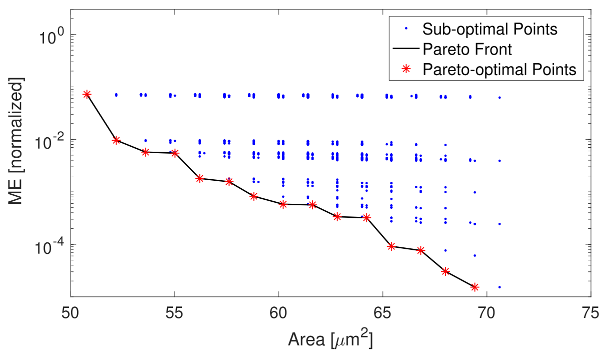
Although, the aforementioned analysis provides us with the configurations that offer optimal quality-efficiency trade-off, it is to be noted that a significant quality degradation will occur if we employ higher error rate and higher error magnitude approximations in the elementary $2 \times 2$ modules (aiming at higher efficiency). However, if we introduce approximations in the proposed self-healing manner such that the mean error of two modules forming a *mirror pair* are additive inverse of each other, then we have:

$$P(ME_{total}) = P(ME_1) + P(ME_2) = 0 \tag{13}$$

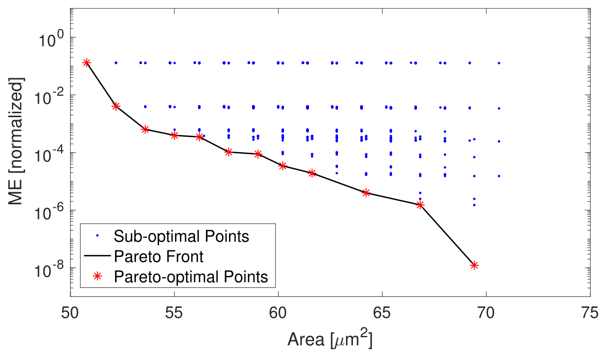
where $P(ME_1)$ and $P(ME_2)$ are the expected ME values of the two squarers in a *mirror pair*, and $P(ME_{total})$ is the overall expected ME value of the approximate *mirror pair*. Equation (13) shows that the expected ME value of a squarer *mirror pair* is zero, which will result in an overall zero error after the accumulation stage in a SAC architecture. This will also hold true for high error rate and high error magnitude approximate elementary $2 \times 2$ designs in a *mirror pair*. However, $P(ME_{total})$ is not necessarily zero in case of random *finite-length* input distributions for the self-healing case as in (13). This brings the importance of finite-length random input analysis for quality-efficiency trade-off evaluation and comparison of the self-healing methodology with the conventional methodology, and is presented in the following subsection.

## *B. QUALITY-EFFICIENCY TRADE-OFF OF SQUARER PAIRS FOR RANDOM FINITE-LENGTH INPUT VECTORS AND COMPARISON OF SELF-HEALING WITH CONVENTIONAL METHODOLOGY*

Here we evaluate SAC architectures utilizing approximate Sq$8\times8$ pairs with random finite-length input vectors and compare the quality-efficiency trade-off of the proposed self-healing methodology with the conventional methodology. We consider an exhaustive simulation method (Stage 2) where inputs are fed to the selected pareto-optimal designs (from Stage 1) and a quantitative analysis is done. We have utilized uniform and normal ($\mu = 128$ and $\sigma = 22.5$) distributions, where each distribution has two finite-lengths: a small data size with 100 vectors of 124 elements each and a large data size with 1000 vectors of 10000 elements each.



(a) ME vs. Area for uniformly distributed inputs.



(b) ME vs. Area for normally distributed inputs.

**FIGURE 11. Design space of an approximate Sq8×8 module constructed using elementary 2 × 2 modules: accurate P2×2, M1, accurate Sq2×2 and S1. The normal input distribution has $\mu = 128$ and $\sigma = 22.5$. Pareto-optimal points are chosen that provide best efficiency for a given quality constraint and vice versa.**

**TABLE 2. Area of 2 × 2 multiplier (P2×2) and squarer (Sq2×2) pairs.**

| Multiplier/Squarer Pair | Area ($\mu m^2$) |
|---|---|
| Accurate P2x2 | 19.29 |
| Conventional Approx P2x2 (M1 and M1) | 14.11 |
| Mirror Approx P2x2 (AAMMP) (M1 and M2) | 15.52 |
| Accurate Sq2x2 | 7.05 |
| Conventional Approx Sq2x2 (S1 and S1) | 4.24 |
| Mirror Approx Sq2x2 (AASMP) (S1 and S2) | 2.82 |

Area is considered as efficiency target as in Section V-A. However, here we consider a pair of 8 × 8 squarers utilizing 2 × 2 multiplier (P2×2) and 2 × 2 squarer (Sq2×2) pairs in a conventional and proposed way. In case of the conventional P2×2 pair, both P2×2 are M1 (3*3≈7). But when we make an *absolute approximate multiplier mirror pair* (AAMMP), one of the two P2×2 is M1 (3*3≈7) and the other is M2 (3*3≈11). Likewise, both Sq2×2 of a conventional Sq2×2 pair are S1, while an *absolute approximate squarer mirror pair* (AASMP) utilizes S1 and S2 to form a proposed Sq2×2 pair. Table 2 shows the area of aforesaid pairs, synthesized at 1.43GHz for TSMC 40nm Low Power (TCBN40LP) technology. The area cost of each 8 × 8 squarer design is estimated by adding the areas of 2 × 2 constructs only, not the adder trees. This estimation is plausible for comparison purpose because the adder trees remain accurate in all designs. However, in Section VI we will present some synthesized designs including the adder trees.

Fig. 12 shows the quality-efficiency trade-off of approximate Sq8×8 pairs in a SAC architecture for random finite-length inputs. We have utilized the pareto-optimal approximate Sq8×8 design configurations suggested by statistical analysis (Stage 1, Fig. 11). Here we show the error metric (MSE) at the y-axis, which is computed at the output of the accumulator using (6). As expected, the quality-efficiency trade-off of the conventional methodology in Fig. 12 follows the pareto-front of Fig. 11 both for uniform and normal distribution. However, for normally distributed inputs, eleven pareto-optimal points (designs) have been shown in Fig. 12 for conventional methodology while they are twelve in Fig. 11. This is because the lowest approximation level (maximum area) pareto-optimal point in Fig. 11 has generated zero error for finite-length input due to absence of the error case, which brings MSE(dB) value of $-\infty$ ($\log_{10}(0) = -\infty$). Therefore, this design point is not shown in Fig. 12.

It is to be noted that a low MSE value and a small area are desired. Therefore, a pareto front that is near to the origin of the graph is a better option. It follows from the quality-efficiency trade-off illustrated in Fig. 12 that self-healing improves the pareto front (the optimal designs) for almost all considered input cases. For large data sizes (large vector inputs), the pareto front of the proposed self-healing methodology completely outperforms the conventional counterpart, because the large random vectors have more tendency towards an ideal input distribution (uniform or normal in this case). However, in case of smaller data sizes (small vector inputs), randomly generated vectors have a relatively higher deviation from ideal distribution, which results in less error cancellation in case of the proposed self-healing methodology. Nevertheless, self-healing still improves the trade-off for smaller data sizes by introducing several better pareto-optimal designs as shown in Fig. 12. It should be noted that the smallest area (highest approximation level) of the proposed self-healing methodology is greater than the smallest area of the conventional methodology. This is due to the fact that an *absolute approximate mirror pair* of a 2×2 multiplier has a larger area as compared to an approximate conventional pair as shown in Table 2. Specifically, in case of a conventional approximate multiplier pair, we have two M1, while in case of an AAMMP, we have M1 and M2. As the area of M2 is larger than M1, we have more area cost for a 2×2 AAMMP. Moreover, because of a lower probability of error at a 2×2 multiplier, the pareto-optimal designs suggested by statistical analysis tend to utilize more approximate 2×2 AAMMPs as compared to that of 2×2 AASMPs, therefore the higher order (n×n) approximate squarer pairs also have a higher area cost for self-healing methodology as compared to conventional methodology. However, as we see the

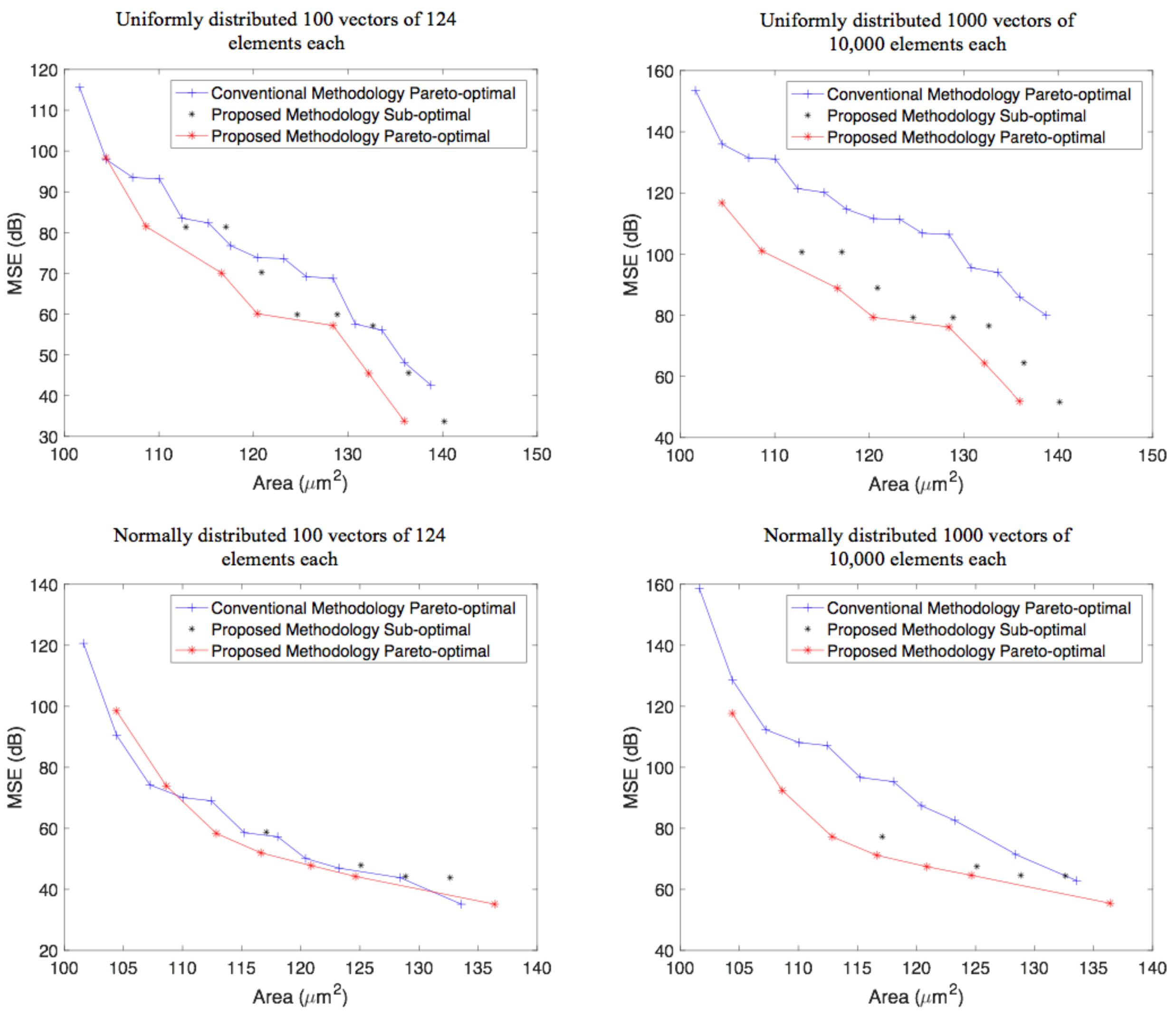


**FIGURE 12.** **Exhaustive simulation of pareto-optimal designs obtained from Stage1 for various finite-length randomly distributed inputs. The proposed self-healing methodology improves the quality-efficiency trade-off for all considered input distributions, which is quite significant in case of larger vector inputs.**

complete trade-off, the proposed self-healing methodology increases the design space (i.e., offers additional pareto-optimal designs), and provides overall more effective quality-efficiency trade-off.

It is to be noted that other efficiency targets can also be considered, e.g., power, performance, or energy to find the pareto-optimal designs from Stage 1 and Stage 2 (Fig. 9), where the relevant cost functions can be utilized like power consumption, delay, or power-delay-product respectively. Subsequently, while having a clear quality-efficiency trade-off (as in Fig. 12), an optimal design can be chosen based on user-defined quality constraints for the given input distribution, which accomplishes Stage 3 of the design process.

## VI. EXPERIMENTAL SETUP AND RESULTS

In order to compare the conventional and the proposed self-healing designs, a quality-efficiency trade-off study based on estimated area has been discussed in Section V-B. In this section, we consider some selected designs to show the synthesis results to accurately quantify the efficiency benefits of self-healing over the conventional methodology. We present results of quality analysis and hardware synthesis of the proposed and conventional approximate Sq8×8 designs deployed in square-accumulate architectures for random finite-length input vectors. Moreover, for the radio astronomy calibration processing case study, we analyze the quality impact of self-healing and compare it with an equivalent efficiency design utilizing the conventional methodology.

### *A. EXPERIMENTAL SETUP FOR QUALITY-EFFICIENCY TRADE-OFF STUDY*

Fig. 13 shows our experimental setup to study the quality-efficiency trade-off. Quality analysis has been performed in Matlab utilizing behavioral models of approximate

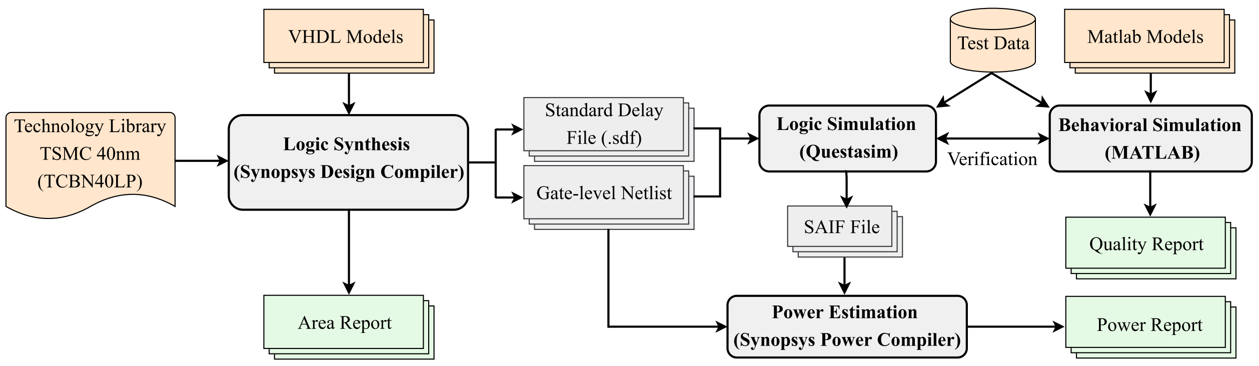

VHDL Models
Technology Library TSMC 40nm (TCBN40LP)
Logic Synthesis (Synopsys Design Compiler)
Standard Delay File (.sdf)
Gate-level Netlist
Area Report
Test Data
Matlab Models
Logic Simulation (Questasim)
Verification
Behavioral Simulation (MATLAB)
SAIF File
Quality Report
Power Estimation (Synopsys Power Compiler)
Power Report

**FIGURE 13.** Our experimental setup for quality-efficiency trade-off study.

multiplier/squarer designs. We used Synopsys tools: Design Compiler and Power Compiler to assess hardware costs i.e., area and power, for the TSMC 40nm Low Power (TCBN40LP) technology library. Questasim has been utilized to verify the functionality of the synthesized designs (gate-level netlists) and to generate the related SAIF (Switching Activity Interchange Format) files based on the respective standard delay file (.sdf) and test data. The aforesaid SAIF files and gate-level-netlists are utilized by Synopsys Power Compiler for power estimation. For efficiency comparison, we assume a higher efficiency for a lower computational cost (chip-area and power consumption) and vice versa.

## B. QUALITY-EFFICIENCY TRADE-OFF OF Sq8×8 PAIRS IN A SAC ARCHITECTURE

Here we present the comparison of some design alternatives of Sq8×8 pairs in a SAC architecture to quantify the efficiency benefits. The following designs are considered,

### a: ACCU

An accurate Sq8×8 pair, where both Sq8×8 in a pair are composed of accurate 2 × 2 elementary modules (P2×2 and Sq2×2).

### b: CONVENT

An Sq8×8 pair designed utilizing conventional approximation approach where a least significant P2×2 (shown in green in Fig. 6) is approximated as M1 (3*3≈7) [22] for both Sq8×8s in a pair.

### c: SH1

An Sq8×8 pair designed utilizing the proposed self-healing approximation approach where one least significant P2×2 is approximated as M1 (3*3≈7) [22] in one Sq8×8, and one least significant P2×2 is approximated as M2 (3*3≈11) in the other Sq8×8. This forms an AASMP.

### d: SH3

In addition to SH1 approximations, two least significant Sq2×2 (shown in green in Fig. 6) are approximated as S1 (2*2≈0) for one Sq8×8, and approximated as S2 (2*2≈8) for the other Sq8×8 to form an AASMP. Therefore, approximating the three least significant 2 × 2 elementary modules in a self-healing fashion.

### e: CONVENT3

This is a conventional counterpart of SH3, where one least significant P2×2 is approximated as M1 and two least significant Sq2×2 are approximated as S1 for both Sq8×8s in an approximate pair.

### f: SH7

In addition to SH1 and SH3 approximations, all four elementary modules (P2×2) of P4×4 (see Fig. 6) are approximated as M1 for one Sq8×8 and as M2 for the other Sq8×8 to form an AASMP. Therefore, approximating seven least significant 2 × 2 elementary modules in self-healing fashion.

We compare the computational costs in terms of chip-area and power consumption of the above Sq8×8 pairs at the operating frequencies: 1 GHz (Area_1, Power_1) and 1.43GHz (Area_2, Power_2). Normally distributed input vectors have been utilized to estimate power consumption. Table 3 shows an increase in computational efficiency ($E$) as the approximations are increased, i.e., a higher number of elementary 2 × 2 modules are approximated. For instance, we can see a minimum area for SH7 and a maximum for Accu. Table 3 can be summarized as,

$$E_{SH7} > E_{Convent3} \approx E_{SH3} > E_{Convent} > E_{SH1} \quad (14)$$

For a quality comparison, Mean Square Error (MSE) is computed at the SAC output as in (6). The result is shown in Fig. 14. We analyzed Uniform (Unif_x) and normal (Norm_x) 8-bit unsigned input distributions, namely:

**TABLE 3.** Computational cost comparison of accurate and approximate Sq8×8 pairs. Convent and Convent3 are based on conventional approximation methodology; while SH1, SH3 and SH7 are self-healing based approximate designs.

| Parameters | Accu | Convent | SH1 | SH3 | Convent3 | SH7 |
|---|---|---|---|---|---|---|
| Area_1 ($\mu m^2$) | 642 | 617 | 626 | 599 | 582 | 436 |
| Power_1 ($\mu W$) | 460 | 437 | 445 | 423 | 426 | 399 |
| Area_2 ($\mu m^2$) | 1000 | 925 | 935 | 897 | 855 | 729 |
| Power_2 ($\mu W$) | 1079 | 1051 | 1048 | 979 | 977 | 795 |

Area_1 and Power_1 at 1GHz, and Area_2 and Power_2 at 1.43GHz.

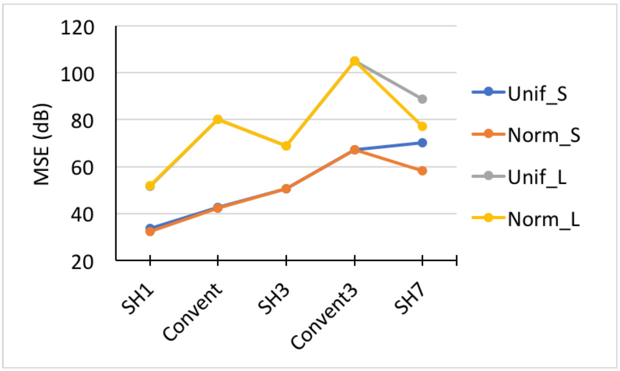


**FIGURE 14.** Quality comparison for various SAC designs utilizing Sq8×8 pairs. SH3 provides better quality as compared to conventional equivalent efficiency design (Convent3), while SH7 outperforms Convent3 both in quality (mostly) and efficiency.

Unif_S, Unif_L, Norm_S, Norm_L, where S stands for a small data size (100 vectors of 124 elements each) and L stands for a larger data size (1000 vectors of 10,000 elements each). We consider Norm_x distributions with $\mu = 128$ and $\sigma = 22.5$. In case of small data sizes (Unif_S and Norm_S), SH1 provides better quality as compared to the conventional approach (Convent). Even though the computational efficiency ($E$) of SH1 is lower than that of Convent design, this introduces an additional design alternative in the trade-off. Moreover, SH3 brings higher $E$ at lower quality as compared to Convent, and also contributes to an additional design alternative. Interestingly, SH3 and SH7 provide better quality output (mostly) as compared to Convent3 with almost equivalent and higher computing efficiencies respectively, this is due to the error cancellation opportunity brought by the self-healing approximation methodology.

For large data sizes with uniform (Unif_L) and normal (Norm_L) distributions, self-healing designs SH1 and SH3 show better quality than the conventional design Convent. Interestingly, SH3 deploys higher area- and power-efficiency as compared to Convent and still provides better quality due to error cancellation. Similar to small data sizes, self-healing designs SH3 and SH7 provide better quality as compared to the Convent3 design for large data sizes. SH7 shows better efficiency (up to 25% better area efficiency and up to 18.6% better power efficiency, see Table 3) as compared to the conventional design (Convent3) with a better quality output.

Therefore, we can conclude that in comparison to the conventional design approach, the self-healing approach increases the design space in some cases and brings better (higher efficiency and higher output quality) designs in the other cases to provide an effective quality-efficiency trade-off.

### C. RADIO ASTRONOMY CALIBRATION PROCESSING – A CASE STUDY

Radio astronomy calibration processing, known as StEFCal [8], employs a Least Squares (LS) algorithm that takes sky Visibilities (V) as input and utilizes the Model visibilities (M) to estimate sensor gains (G) for a given radio telescope configuration. It has three computationally expensive components: square-accumulate (SAC), multiply-accumulate (MAC) and element-wise product (EP), where M, V, Z, G ∈ ℂ (Fig. 15). Here we only present the simulation results for SAC architecture that computes,

$$\sum_{j=1}^{N}\{(Zr_j * Zr_j) + (Zi_j * Zi_j)\} \tag{15}$$

where $Zr$ is the real part and $Zi$ is the imaginary part of a complex vector $Z$. Each multiplication in (15) can be rewritten as a multiplier pair to employ self-healing,

$$\sum_{k=1}^{N-1}\{[(Zr_k * Zr_k) + (Zr_{k+1} * Zr_{k+1})] + [(Zi_k * Zi_k) + (Zi_{k+1} * Zi_{k+1})]\} \quad k \in 2\mathbb{Z}_{\geq 0} + 1 \tag{16}$$

Table 4 shows the quality analysis of various design choices as discussed in Section VI-B (Accu, Convent, SH1, SH3, Convent3). For the MSE (SAC error), both self-healing designs (SH1 and SH3) provide better quality as compared to their conventional counterparts (Convent and Convent3 respectively). It is important to note that the conventional designs produce higher error because of lack of error cancellation. However, the self-healing based designs can cancel out the error partially or fully (based on the input distribution). For that matter, we can see that SH3 produces 46.7% better quality as compared to the conventional counterpart (Convent3) while providing approximately the same computing efficiency as discussed in Section VI-B.

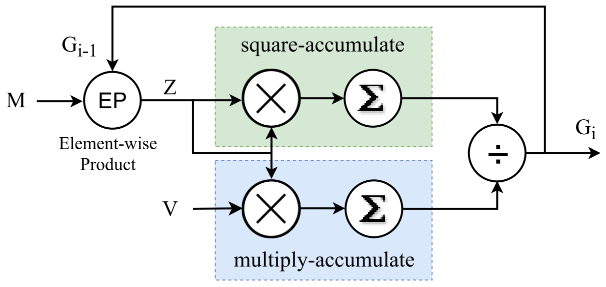


**FIGURE 15.** Least squares algorithm for radio astronomy calibration, where one of the computationally expensive components is square-accumulate (SAC).

**TABLE 4.** Quality analysis of radio astronomy calibration for employing various approximate SAC alternatives. SH3 and Convent3 are almost equal efficiency designs, however SH3 brings 46.7% better quality.

| Design Alternatives | SAC Error (MSE) |
|---|---|
| Accu | 0 |
| Convent | 7.8341e-07 |
| SH1 | 7.7470e-07 |
| Convent3 | 2.25e-02 |
| SH3 | 1.20 e-02 |

$E_{Convent3} \approx E_{SH3} > E_{Convent} > E_{SH1}$

## VII. CONCLUSION

A novel Self-Healing (SH) methodology to enable efficient and systematic approximate computing has been presented. Our analysis has shown that exploiting healing stages of an algorithm in general, and of the square-accumulate (SAC) architecture in particular, provides an effective quality-efficiency trade-off. We have shown how SH can be employed to truncation and logic pruning approximate computing techniques. We discussed the statistical error analysis, randomly distributed finite-length input analysis and a case study of radio astronomy calibration processing for an approximate SAC architecture that showed a more effective quality-efficiency trade-off utilizing SH as compared to conventional approximation methodology. Nevertheless, comprehensive design space exploration—based on input distribution and quantified error resilience—is required to ensure the highest efficiency for a given quality constraint within radio astronomy calibration processing. We have shown how *absolute approximate mirror pairs* are designed for unsigned logic-pruned multipliers and squarers, and their utilization in SAC architectures for an effective quality-efficiency trade-off. However, the utilization of SH for the signed multiplier case and Multiply-Accumulate (MAC) architectures for attaining an effective quality-efficiency trade-off are indicated as future directions of research.

## APPENDIX
### 8 × 8 SQUARER CONSTRUCTION

As discussed in Section IV, an 8 × 8 squarer (Sq8×8) computes A*A operation, where A is an 8-bit unsigned number. Let $A = a_7a_6\ a_5a_4\ a_3a_2\ a_1a_0$, where $a_7a_6$ are the most significant two bits, and $a_1a_0$ are the least significant two bits. Therefore,

$$\begin{aligned}\text{Sq}8\times8 &= a_7a_6\ a_5a_4\ a_3a_2\ a_1a_0 * a_7a_6\ a_5a_4\ a_3a_2\ a_1a_0\\ &= a_3a_2\ a_1a_0 * a_3a_2\ a_1a_0 + 16(a_3a_2\ a_1a_0\\ &\quad * a_7a_6\ a_5a_4) + 16(a_7a_6\ a_5a_4 * a_3a_2\ a_1a_0)\\ &\quad + 256(a_7a_6\ a_5a_4 * a_7a_6\ a_5a_4) \qquad (17)\end{aligned}$$

Equation (17) shows that the total of four 4×4 partial products compute the Sq8×8 operation as illustrated in Fig. 5 (left). However, two of the 4 × 4 partial products multiply the same (equal) inputs. Therefore, (17) can be re-written as,

$$\begin{aligned}&= a_3a_2\ a_1a_0 * a_3a_2\ a_1a_0 + 32(a_3a_2\ a_1a_0 * a_7a_6\ a_5a_4)\\ &\qquad + 256(a_7a_6\ a_5a_4 * a_7a_6\ a_5a_4) \qquad (18)\end{aligned}$$

which means that three 4 × 4 partial products can compute the Sq8×8 operation, where the factors 32 and 256 are implemented as bit shifts as shown in Fig. 6. Moreover, each 4 × 4 partial product can be further decomposed into basic 2 × 2 partial product constructs as shown in Fig. 5 (right). Therefore, (18) becomes,

$$\begin{aligned}&= a_1a_0 * a_1a_0 + 4(a_1a_0 * a_3a_2) + 4(a_3a_2 * a_1a_0)\\ &\quad + 16(a_3a_2 * a_3a_2) + 32[a_1a_0 * a_5a_4 + 4(a_1a_0 * a_7a_6)\\ &\quad + 4(a_5a_4 * a_3a_2) + 16(a_3a_2 * a_7a_6)] + 256[a_5a_4 * a_5a_4\\ &\quad + 4(a_5a_4 * a_7a_6) + 4(a_7a_6 * a_5a_4) + 16(a_7a_6 * a_7a_6)] \qquad (19)\end{aligned}$$

By combining the same (equal) 2 × 2 partial product operations,

$$\begin{aligned}&= a_1a_0 * a_1a_0 + 8(a_1a_0 * a_3a_2) + 16(a_3a_2 * a_3a_2)\\ &\quad + 32[a_1a_0 * a_5a_4 + 4(a_1a_0 * a_7a_6 + a_5a_4 * a_3a_2)\\ &\quad + 16(a_3a_2 * a_7a_6)] + 256[a_5a_4 * a_5a_4\\ &\quad + 8(a_5a_4 * a_7a_6) + 16(a_7a_6 * a_7a_6)] \qquad (20)\end{aligned}$$

Equation (20) shows total of ten 2 × 2 partial products, wherein four are Sq2×2 and six are P2×2 (also depicted in Fig. 6). The adders and shift factors are implemented in the higher order blocks like P4×4, Sq4×4 and Sq8×8 shown in Fig. 6.

## REFERENCES

[1] Q. Xu, M. Todd, and S. K. Nam, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[2] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proc. 52nd Annu. DAC ACM/EDAC/IEEE*, San Francisco, CA, USA, Jun. 2015, pp. 1–6.

[3] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62-1–62-33, Mar. 2016.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited: Cross-layer approximate computing: From logic to architectures," in *Proc. 53rd Annu. DAC ACM/EDAC/IEEE*, Austin, TX, USA, Jun. 2016, pp. 1–6.

[5] R. Nair, "Big data needs approximate computing: Technical perspective," *Commun. ACM*, vol. 58, no. 1, p. 104, 2015.

[6] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annu. DAC ACM/EDAC/IEEE*, Austin, TX, USA, May/Jun. 2013, pp. 1–9.

[7] E. Nogues, D. Menard, and M. Pelcat, "Algorithmic-level approximate computing applied to energy efficient HEVC decoding," *IEEE Trans. Emerg. Topics Comput.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/7517349/, doi: 10.1109/TETC.2016.2593644.

[8] S. Salvini and S. J. Wijnholds, "Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications," *Astron. Astrophys.*, vol. 571, p. A97, Nov. 2014.

[9] G. A. Gillani and A. B. J. Kokkeler, "Improving error resilience analysis methodology of iterative workloads for approximate computing," in *Proc. CF*, Siena, Italy, 2017, pp. 374–379.

[10] V. T. Olafsson, D. C. Noll, and J. A. Fessler, "Fast spatial resolution analysis of quadratic penalized Least-Squares image reconstruction with separate real and imaginary roughness penalty: Application to fMRI," *IEEE Trans. Med. Imag.*, vol. 37, no. 2, pp. 604–614, Feb. 2018.

[11] M. D. Buhari, G. Y. Tian, R. Tiwari, and A. H. Muqaibel, "Multicarrier SAR image reconstruction using integrated MUSIC-LSE algorithm," *IEEE Access*, vol. 6, pp. 22827–22838, 2018, doi: 10.1109/ACCESS.2018.2817359.

[12] S. Naghibzadeh, A. M. Sardarabadi, and A. J. van der Veen, "Radioastronomical image reconstruction with regularized least squares," in *Proc. ICASSP IEEE*, Shanghai, China, Mar. 2016, pp. 3316–3320.

[13] A. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. DATE*, Munich, Germany, 2008, pp. 1250–1255.

[14] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy, "Impact: IMprecise adders for low-power approximate computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, Fukuoka, Japan, Aug. 2011, pp. 409–414.

[15] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Annu. DAC ACM/EDAC/IEEE*, San Francisco, CA, USA, Jun. 2012, pp. 820–825.

[16] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[17] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. 52nd Annu. DAC ACM/EDAC/IEEE*, San Francisco, CA, USA, Jun. 2015, p. 86.
[18] M. K. Ayub, O. Hasan, and M. Shafique, "Statistical error analysis for low power approximate adders," in *Proc. 54th Annu. DAC ACM/EDAC/IEEE*, Austin, TX, USA, Jun. 2017, pp. 1–6.
[19] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel, "Probabilistic error modeling for approximate adders," *IEEE Trans. Comput.*, vol. 66, no. 3, pp. 515–530, Mar. 2017.
[20] B. S. Prabakaran *et al.*, "DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems," in *Proc. DATE*, Dresden, Germany, 2018, pp. 917–920.
[21] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. EDSSC*, Hong Kong, China, 2010, pp. 1–4.
[22] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. VLSI Des. IEEE*, Chennai, India, Jan. 2011, pp. 346–351.
[23] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilience systems," in *Proc. ISQED*, Santa Clara, CA, USA, 2014, pp. 263–269.
[24] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.
[25] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-space exploration of approximate multipliers," in *Proc. ICCAD*, Austin, TX, USA, 2016, pp. 1–8.
[26] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.
[27] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *Proc. NANOARCH IEEE/ACM*, Beijing, China, Jul. 2016, pp. 191–196.
[28] S. Mazahir, O. Hasan, R. Hafiz, and M. Shafique, "Probabilistic error analysis of approximate recursive multipliers," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1982–1990, Nov. 2017.
[29] S. Ullah *et al.*, "Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators," in *Proc. 55th Annu. DAC ACM/EDAC/IEEE*, San Francisco, CA, USA, Jun. 2018, p. 159.
[30] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 421–430, Mar. 2018.
[31] M. B. Sullivan and E. E. Swartzlander, "Truncated error correction for flexible approximate multiplication," in *Proc. ASILOMAR*, Pacific Grove, CA, USA, 2012, pp. 355–359.
[32] S. R. Kuang and J. P. Wang, "Low-error configurable truncated multipliers for multiply-accumulate applications," *Electron. Lett.*, vol. 42, no. 16, pp. 904–905, Aug. 2006.
[33] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 6, pp. 1312–1325, Jun. 2010.
[34] D. Esposito, A. G. Strollo, and M. Alioto, "Low-power approximate MAC unit," in *Proc. PRIME*, Giardini Naxos, Italy, 2017, pp. 81–84.
[35] S. Dutt, A. Chauhan, R. Bhadoriya, S. Nandi, and G. Trivedi, "A high-performance energy-efficient hybrid redundant MAC for error-resilient applications," in *Proc. IEEE VLSI Design*, Bangalore, India, Jan. 2015, pp. 351–356.
[36] R. Jongerius, S. Wijnholds, R. Nijboer, and H. Corporaal, "An end-to-end computing model for the square kilometre array," *Computer*, vol. 47, no. 9, pp. 48–54, Sep. 2014.
[37] Q. Wu, Y. Zhu, X. Wang, M. Li, J. Hou, and A. Masoumi, "Exploring high efficiency hardware accelerator for the key algorithm of square kilometer array telescope data processing," in *Proc. FCCM*, Napa, CA, USA, 2017, p. 195.
[38] B. Veenboer, M. Petschow, and J. W. Romein, "Image-Domain gridding on graphics processors," in *Proc. IPDPS*, Orlando, FL, USA, May/Jun. 2017, pp. 545–554.
[39] B. Barrois, O. Sentieys, and D. Menard, "The hidden cost of functional approximation against careful data sizing: A case study," in *Proc. DATE*, Lausanne, Switzerland, 2017, pp. 181–186.

**G. A. GILLANI** received the B.Sc. degree (Hons.) in electrical engineering from the University of Engineering and Technology at Taxila, Taxila, Pakistan, and the M.Eng. degree in computer science and technology from Northwestern Polytechnical University, Xi'an, China. He is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands.

His research interests include energy-efficient computing, low-power signal processing, and approximate computing. He was a recipient of the Chinese Government Scholarship for the M.Eng. degree.

**MUHAMMAD ABDULLAH HANIF** received the B.Sc. degree in electronic engineering from the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI), Pakistan, and the M.Sc. degree in electrical engineering with a specialization in digital systems and signal processing from the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan. He is currently a University Assistant with the Department of Informatics, Institute of Computer Engineering, Vienna University of Technology (TU Wien), Austria.

He was also a Research Associate with the Vision Processing Lab, Information Technology University, Pakistan, and as a Lab Engineer with GIKI, Pakistan. His research interests are in brain-inspired computing, machine learning, approximate computing, computer architecture, energy-efficient design, robust computing, system-on-chip design, and emerging technologies. He was a recipient of the President's Gold Medal for the outstanding academic performance during the M.Sc. degree.

**M. KRONE** received the B.Sc. degree (Hons.) in electrical engineering from the University of Twente, Enschede, The Netherlands, in 2017, where he is currently pursuing the M.Sc. degree in electrical engineering with the Faculty of Electrical Engineering, Mathematics, and Computer Science.

His research interests are in the fields of electronics design including digital, analog, and mixed-signal designs.

**S. H. GEREZ** received the M.Sc. degree (Hons.) in electrical engineering and the Ph.D. degree in applied sciences from the University of Twente, The Netherlands, in 1984 and 1989, respectively. He has been an Assistant Professor with the University of Twente since 1990 (part-time starting from 2001), focusing on research and education in the fields of implementation of digital signal processing, digital integrated-circuit design, and design automation.

From 2001 to 2009, he was with the Cordless Telephony Division of National Semiconductor (called Sitel Semiconductor after 2005 and currently part of Dialog Semiconductor). Since 2009, he has been running his business Bibix, which offers consultancy services in his mentioned fields of interest. He has authored the book *Algorithms for VLSI Design Automation* (Wiley, 1998).

**MUHAMMAD SHAFIQUE** (M'11–SM'16) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT) in 2011. Before, he was with Streaming Networks Pvt. Ltd., where he was involved in the research and development of advanced video coding systems for several years. He is currently a Full Professor with the Department of Informatics, Institute of Computer Engineering, Vienna University of Technology (TU Wien), Austria. He is also the Director of the Group on Computer Architecture and Robust, Energy-Efficient Technologies.

His research interests are in computer architecture, power- and energy-efficient systems, robust computing covering various aspects of dependability and fault-tolerance, hardware security, emerging computing trends like neuromorphic and approximate computing, neurosciences, emerging technologies and nanosystems, self-learning and intelligent/cognitive systems, FPGAs, MPSoCs, and embedded systems. His research has a special focus on cross-layer analysis, modeling, design, and optimization of computing; and memory systems covering various layers of the hardware and software stacks, and their integration in application use cases from Internet of Things, cyber-physical systems, and ICT for development domains.

He has authored over 180 papers in premier journals and conferences. He holds one U.S. patent. He is a member of the ACM, SIGARCH, SIGDA, SIGBED, and HiPEAC, and a Senior Member of the IEEE and IEEE Signal Processing Society. He received the Prestigious 2015 ACM/SIGDA Outstanding New Faculty Award, six gold medals in educational career, several Best Paper Awards and nominations at prestigious conferences like DATE, DAC, ICCAD, and CODES+ISSS, the Best Master Thesis Award, and the Best Lecturer Award. He has given several invited talks, tutorials, and keynotes. He has served as the TPC Co-Chair of ESTIMedia and LPDC, the General Chair of ESTIMedia, and the Track Chair of DATE and FDL. He has served on the program committees of several IEEE/ACM conferences like ICCAD, ISCA, DATE, CASES, FPL, and ASPDAC. He has served as a Guest Editor for the IEEE *Design & Test Magazine* and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.

**A. B. J. KOKKELER** was a System Engineer with Ericsson for over six years and the Scientific Project Manager of the Netherlands Foundation for Research in Astronomy for over eight years. In 2003, he joined the University of Twente, where he is currently appointed as an Associate Professor.

He is currently involved in research projects, sponsored by the Dutch, and the European Governments and industry. He has a background in telecommunication, mixed signal design, and signal processing. His current main interests lie in the areas of the design of low-power architectures for telecommunications and computationally intensive applications.