Energy-Efficient Approximate Least Squares Accelerator

A Case Study of Radio Astronomy Calibration Processing

G.A. Gillani, A. Krapukhin, and A.B.J. Kokkeler
Faculty of EEMCS, University of Twente, Enschede 7500 AE, Netherlands
{s.ghayoor.gillani,a.b.j.kokkeler}@utwente.nl

ABSTRACT

Approximate computing allows the introduction of inaccuracy in the computation for cost savings, such as energy consumption, chip-area, and latency. Targeting energy efficiency, approximate designs for multipliers, adders, and multiply-accumulate (MAC) have been extensively investigated in the past decade. However, accelerator designs for relatively bigger architectures have been of less attention yet.

The Least Squares (LS) algorithm is widely used in digital signal processing applications, e.g., image reconstruction. This work proposes a *novel LS accelerator* design based on a heterogeneous architecture, where the heterogeneity is introduced using accurate and approximate processing cores. We have considered a case study of radio astronomy calibration processing that employs a complex-input iterative LS algorithm. Our proposed methodology exploits the intrinsic error-resilience of the aforesaid algorithm, where initial iterations are processed on approximate modules while the later ones on accurate modules. Our energy-quality experiments have shown up to 24% of energy savings as compared to an accurate (optimized) counterpart for biased designs and up to 29% energy savings when unbiasing is introduced. The proposed LS accelerator design does not increase the number of iterations and provides sufficient precision to converge to an acceptable solution.

CCS CONCEPTS

• **Computing methodologies** → **Model development and analysis**; • **Hardware** → *Power and energy*; • **Computer systems organization** → Heterogeneous (hybrid) systems;

KEYWORDS

Least squares accelerator, iterative workloads, approximate computing, energy efficiency, radio astronomy.

ACM Reference format:
G.A. Gillani, A. Krapukhin, and A.B.J. Kokkeler. 2019. Energy-Efficient Approximate Least Squares Accelerator. In *Proceedings of CF'19, Alghero, Italy, April 30-May 2, 2019,* 8 pages.
https://doi.org/10.1145/3310273.3323161

CF'19, April 30-May 2, 2019, Alghero, Italy

ACM ISBN 978-1-4503-6685-4/19/05…$15.00
https://doi.org/10.1145/3310273.3323161

1 INTRODUCTION

Approximate computing has shown hardware efficiency benefits for processing of error-resilient applications such as machine learning, search engines, and multimedia digital signal processing [10, 20]. The hardware efficiency benefits refer to a lower chip-area, power consumption, latency, and energy consumption of a circuit as compared to that of the so-called accurate computing counterparts.

Error-resilient applications have one or more of the following characteristics: redundant/real-life inputs, iterative/statistical computations, and a set of acceptable outcomes [5]. To evaluate a target application for error-resilience, statistical[5, 9] approximation models are applied. These models inject errors during the execution of a target application on statistical bases to quantify the bearable error profile. For iterative applications, the analysis in [9] suggests that a certain number of initial iterations can be approximated while producing an acceptable outcome.

Being pivotal building blocks of DSP architectures, approximate multipliers and adders have been extensively researched for increased hardware efficiency [3, 11, 12, 14, 15, 18]. Accumulation based processing units like multiply-accumulate (MAC) [6] and square-accumulate (SAC) [8] have also shown higher power/energy efficiency as compared to their accurate counterparts. However, approximate accelerator designs for relatively bigger algorithms have been of less attention yet.

The Least Squares (LS) algorithm is widely utilized in digital signal processing applications like image reconstruction in radio astronomy [16, 19], medical [17], and synthetic aperture radar [4] domains. Despite its importance, no approximate least squares accelerator design has been investigated to the best of our knowledge.

Modern radio telescopes like Square Kilometer Array (SKA) [13] require highly power-/energy-efficient processing architectures to process terabytes of raw data per second. For instance, double-precision fused multiply-add operations will require 7.2MW of power consumption in the medium-frequency array of SKA if contemporary technology would be used [13]. In this work, we investigate an energy-efficient LS accelerator architecture based on a case study of radio astronomy calibration processing. The aforesaid processing employs an iterative LS algorithm to compute sensors' gains for a certain configuration of a radio telescope.

This paper presents a *novel Least Squares (LS) accelerator* design targeting the energy-efficiency. Our design methodology utilizes a heterogeneous architecture composed of two processing cores that differ in their precision of computation, namely an *accurate core* and an *approximate core* (Section 3). In Section 4, we show how a set of initial iterations can be processed in an approximate core, while the rest of the iterations in an accurate core to achieve an overall energy-efficiency increase. Finally, the conclusions of our work are discussed.

2 BACKGROUND

In order to understand the error resilience of an iterative Least Squares (LS) algorithm, here we elaborate on the concepts related to the error-resilience analysis of iterative workloads (applications). Moreover, as we employ approximate multiplication in our LS accelerator design, it is also briefly discussed in this section.

2.1 Error Resilience of Iterative Workloads

Chippa et al. [5] proposed a systematic scheme for Application Resilience Characterization (ARC). This scheme partitions a workload into sensitive and resilient parts and characterizes the resilient parts by utilizing approximation models that represent a wide spectrum of approximate computing techniques. In a high-level analysis, errors are introduced according to the statistical approximation model (SAM), which injects a normally distributed error profile based on three parameters: EM (error mean), EP (error predictability) and ER (error rate). Based on this analysis, an error-resilience profile of an application is generated that can be utilized for selecting promising approximation techniques for a given application [5].

Improvements on the ARC framework were introduced in [9], where an adaptive statistical approximation model (ASAM) was utilized. In addition to the original three parameters of SAM (EM, EP, and ER), a new parameter was utilized, i.e., number of approximate iterations (NAI). The model allows dividing an iterative workload into exact and approximate iterations. The work in [9] applied ASAM to radio astronomy calibration processing that showed a significantly larger approximation space as compared to that of SAM analysis. For this application, it was demonstrated that the first 23% of iterations can be made approximate with certain EM, EP, and ER values, while the remaining iterations remain accurate.

Although the aforesaid SAM and ASAM approximation models provide a high-level error resilience analysis based on acceptable output quality, they do not quantify the hardware efficiency improvements that can be achieved by exploiting the intrinsic approximation space.

2.2 Approximate Multiplication

The simplest form of approximate multiplication is the truncation of inputs, where inputs of the multiplier are truncated for lower significant bits [1]. Another form of approximation is the truncation of partial products, where the less significant partial products are not processed to reduce the hardware costs [15, 18].

Hashemi et al. [11] proposed an approximate multiplier with a dynamic range selection scheme and an unbiased error distribution. The main idea of their method is to use an exact multiplier but with smaller operand widths. If the operands to multiply have a width n, they use a $k \times k$ bit multiplier ($k < n$) and choose the k bits from each operand by detecting the leading 1 in the bit pattern and selecting k bits starting from there. It means that instead of approximating the multiplication process, they approximate the operands while using an exact multiplier. The $2k$ bit result is then shifted to the left by a certain number of bits depending on the positions of the leading ones in the original operands to get a $2n$ bit result. To enable a near-zero mean-error profile, the LSB of the newly formed k bit operand is set to 1. This allows the multiplier to be unbiased and have a near-zero average error.

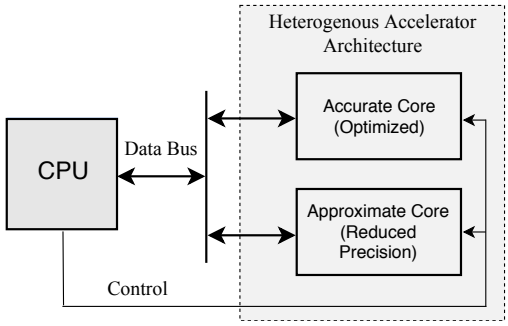


Figure 1: Our design methodology for an approximate Least Squares (LS) accelerator enables initial iterations to be processed on an approximate core (while the rest on an accurate core) to achieve an overall energy-efficiency.

3 DESIGN OF A HETEROGENEOUS LEAST SQUARES ACCELERATOR

Our design methodology for an approximate Least Squares (LS) accelerator is shown in Fig. 1. The accelerator architecture is composed of two cores that differ in computation precision, introducing heterogeneity in the architecture. The accurate core is optimized for the required precision for the LS algorithm. However, the approximate core introduces a reduced-precision in the computation to provide energy efficiency. In the proposed LS accelerator, the initial iterations are run on the approximate core, while the rest of the iterations on the accurate core. This brings an overall energy efficiency when a central processing unit (CPU) switches off the unused core. Nevertheless, using two cores instead of one brings area overhead. However, if the CPU can utilize both cores simultaneously for parallel processing of independent processes, this area overhead can be translated into increased throughput. In any case, the energy efficiency can be increased for processing the LS algorithm with or without area penalty.

In this section, we consider a case study of radio astronomy calibration processing. We demonstrate how to design an LS accelerator using the proposed methodology, wherein the accurate LS core and the approximate LS core are optimized to achieve an energy-efficient LS processing.

3.1 Radio Astronomy Calibration Processing

In radio telescope arrays, the gain of the main beam of each telescope and the phase difference between the telescopes have to be estimated to enhance the quality of astronomical sky images. This process is called gain calibration [19]. The gains combine the effects of atmospheric disturbances, telescope geometry, and receiver characteristics. Atmospheric disturbances can vary within minutes, hence the calibration of these arrays (like SKA) have to be done online. This process is computation-intensive and is energy hungry [9, 13].

Gain calibration can be performed by observing a known bright source in the sky for which a matrix of model visibilities M is known. Matrix V is a measured signal of this source. To estimate the gains, the difference in Eq. (1) has to be minimized by finding

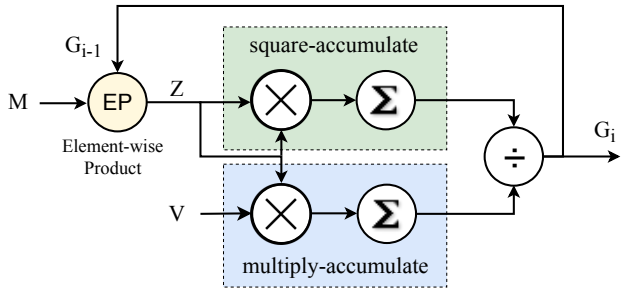


Figure 2: Least Squares (LS) algorithm for radio astronomy calibration processing [8].

the matrix of gains G,

$$||V - GMG||^2 \tag{1}$$

where $G = diag(g)$ represents complex antenna gains. In this work, we assume 124 antennas and 4 frequency channels. Therefore, the gains are 124 complex numbers, so the matrix G has size 124×124 with gains on the diagonal and zeros at other entries. V represents measured visibilities, a matrix with size 124×124×4 as there are four channels. M represents model visibilities having the same size as that of V.

One of the algorithms performing the calibration process is called StEFCal (statistically efficient and fast calibration) [19]. The StEFCal calibration algorithm iteratively solves a least squares problem. To compute the gain of the pth sensor in the ith iteration, all elements of $g^{[i-1]}$ are multiplied by elements of the pth column of the matrix M, i.e., $M_{:,p}$, element-wise to compute vector Z,

$$Z = M_{:,p} \odot g^{[i-1]} \tag{2}$$

Then, a linear least squares problem is solved to find a scaling of Z such that it is as close as possible to the pth column of V, i.e., $V_{:,p}$, to find $g_p^{[i]}$,

$$g_p^{[i]} = \frac{V_{:,p}^H . Z}{Z^H . Z} \tag{3}$$

The symbol H denotes the Hermitian transpose, which means that the vector is transposed, and the complex conjugate of each element is taken. This process is repeated 124 times to get the vector $g_p^{[i]}$ which is then used in the next iteration to compute a better estimate. The algorithm stops when the improvement between iterations is small enough, i.e., when a convergence criterion is reached. In StEFCal the convergence criterion is the norm of the difference of two solution vectors divided by the Frobenius norm of the current solution vector (4). Usually, this process takes around 100 iterations.

$$\text{Convergence} = \frac{||g^{[i]} - g^{[i-1]}||_F}{||g^{[i]}||_F} \leq 10^{-6} \tag{4}$$

As illustrated in Fig. 2, the algorithm can be divided into four stages: element-wise product (EP), square-accumulate (SAC), multiply-accumulate (MAC), and division. A datapath to which the algorithm can be mapped is shown in Fig. 3. It is to be noted that multiplication of two complex numbers requires four multiplications, one addition, and one subtraction, i.e.,

$$(a + ib)(c + id) = ac - bd + i(ad + bc) \tag{5}$$

To compute one gain in the current iteration, the first element of $g = a + ib$ is multiplied by the corresponding element from a column of matrix M, $m = c + id$. The obtained product $z = e + if$ is then multiplied by the corresponding element from matrix V, $v = h + it$, the real and imaginary parts are stored in two registers. Also, $z = e + if$ is multiplied by its complex conjugate, which is equal to $e^2 + f^2$, and the result is stored in one register as the imaginary part is eliminated. This process is repeated 496 times as there are 124 gains and 4 channels, and the running sum is stored in the registers. After the accumulation is complete, the numbers stored in the real and imaginary registers of the MAC part are divided by the real number computed by the SAC part to obtain the gain for the next iteration. To compute the next gain, 496 computations are done again with the next columns of M and V and the current vector g. The gains in the current iteration can be computed in parallel by 124 structures as in Fig. 3, or all the gains can be computed in a serial way by one such structure.

3.2 Optimization of Accurate Least Squares Core

To utilize a heterogeneous accelerator architecture (Fig. 1), we first optimize the accurate LS core. As the StEFCal algorithm is given in floating-point double-precision format, it has to be converted to fixed-point format because fixed-point hardware is more efficient in terms of area and power. This process consists of two parts. First, integer parts of the signals are determined by observing the ranges of all signals, i.e., how many bits are needed to avoid overflow. Secondly, the required fractional parts are determined to satisfy the precision requirements of the algorithm.

To find an optimal number of fractional bits for each signal, an approach described in [2] is used. In this approach, all signals are kept in floating-point format except one which is converted into fixed-point. The minimum fractional length of this signal which satisfies the output-quality is found. This process is repeated for all signals to get the minimum fractional length for each signal. Then, all signals are converted to fixed-point and if the output-quality is satisfied, the optimization is finished. However, typically the fractional widths have to be increased by 1-2 bits from their minimal found widths. For that, the method in [2] uses an exhaustive combinatorial algorithm which tries all possible combinations to find an optimal combination. It starts with increasing one signal by one bit, and all signals are tried. If the output-quality is not satisfied, it increases two signals by two bits, for all possible combinations. This approach requires a large number of simulations and is only feasible if the number of signals is not very large, not more than six as suggested in [2].

If the number of signals is larger, the related and similar signals can be grouped to simplify the search. As the StEFCal algorithm has a large number of signals, they can be grouped into six groups for the optimal bit width search. Each group can have its signals untouched, increased by one bit, or increased by two bits, so there are three options for each group. Assuming 6 groups, $3^6 = 729$ simulations are required, which is feasible for the StEFCal algorithm.

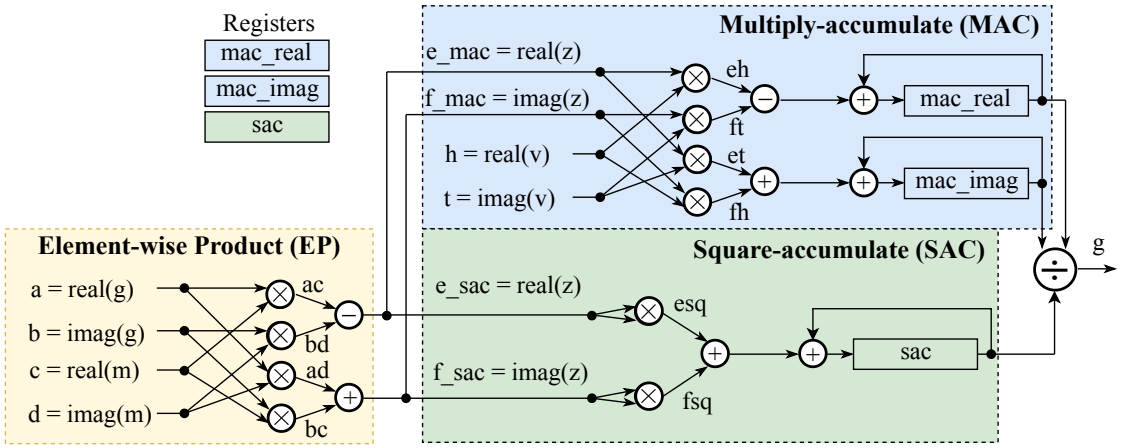


Figure 3: Signal flow of least squares algorithm in radio astronomy calibration processing.

The results of applying this approach to the StEFCal algorithm are provided in Table 1. All the signals can be seen directly in Fig. 3 or traced from the name of the signal. For example, the et_plus_fh signal corresponds to the output of the adder which computes the sum of the outputs of the two multipliers computing the et and fh signals.

In this work, the performance of the StEFCal algorithm is assumed to be satisfied if the number of iterations to converge is smaller or equal to the floating-point version and the relative difference in length between the fixed-point solution vector and the floating-point answer is not more than 10^{-5}, similar to [9].

$$\text{Diff\_rel} = \frac{||g_{\text{float}} - g_{\text{fixed}}||_F}{||g_{\text{float}}||_F} \leq 10^{-5} \tag{6}$$

The signal widths are presented in the WL.FL format which is used in the Fixed-point designer toolbox of MATLAB. In this format, the integer length (IL) is equal to the word length (WL) minus the fractional length (FL). It is to be noted that the *accurate LS core* is also referred to as *accurate core* in the subsequent sections.

3.3 Optimization of Approximate Least Squares Core

The *accurate core* of StEFCal specifies the minimum hardware requirements if exact arithmetic units are used. Simplifying it further will violate the iteration count criterion or the distance from the floating-point reference result. However, better energy-efficiency can be achieved by using approximate arithmetic units. As demonstrated in [9], the algorithm does not require uniform precision in the course of the computation. It suggests that some number of initial iterations can bear a certain level of approximations. Here we demonstrate how simplified reduced-precision hardware can be utilized to process the initial iterations in order to reduce energy consumption. In this regard, we discuss how to achieve a maximum possible energy-efficiency by exploring the design space of an approximate core.

3.3.1 Design Space Exploration. Application of approximations to the StEFCal algorithm (or any iterative workload) is not straightforward. It requires to determine which signals to further truncate (approximate), by how many bits (level of approximation), and for how many iterations. To simplify this problem, the element-wise

Table 1: Fixed-point optimization of StEFCal algorithm for accurate Least Squares (LS) core.

Signal	Min. IL	Min. FL	Optimal FL	Optimal WL.FL
a	9	13	14	23.14
b	8	13	14	22.14
c	-9	24	25	16.25
d	-10	24	25	15.25
h	6	11	12	18.12
t	6	11	12	18.12
ac	-2	23	25	23.25
bd	-4	23	25	21.25
ad	-3	24	26	23.26
bc	-2	24	26	24.26
e_sac	-2	21	23	21.23
f_sac	-2	21	22	20.22
esq	-6	26	28	22.28
fsq	-6	26	28	22.28
esq_plus_fsq	-6	26	28	22.28
sac	1	22	23	24.23
e_mac	-2	23	25	23.25
f_mac	-2	24	26	24.26
eh	3	23	25	28.25
ft	1	23	25	26.25
et	2	24	26	28.26
fh	1	24	26	27.26
eh_minus_ft	3	23	25	28.25
et_plus_fh	2	24	26	28.26
mac_real	7	16	18	25.18
mac_imag	6	16	18	24.18

IL, FL and WL represent integer-length, fraction-length, and word-length respectively.

product is kept accurate (in this work) with the parameters determined in Section 3.2, approximations are applied only to the MAC and SAC units that already cover above 60% of the computation load [9].

As already discussed, the role of an approximate core is to process the initial iterations, while the rest of the iterations are to be processed in the accurate core. To find the energy savings (E_s) due to the approximations employed in the approximate core, the following expression is used,

$$E_s = \frac{100 \times (E_{acc} - E_{ax}) \times N_{ax}}{E_{acc} \times N_{acc}}\% \quad (7)$$

where E_{acc} and E_{ax} refer to energy consumption (for one iteration) of accurate and approximate cores respectively, N_{acc} refers to the total number of iterations if only an accurate core is utilized, and N_{ax} refers to the number of iterations that are run in the approximate core. We have assumed an equal frequency of operation for accurate and approximate cores, which means an equal processing time for both cores for executing a single iteration. Therefore, Eq. (7) is reduced to power (P) numbers only, as in Eq. (8).

$$E_s = \frac{100 \times (P_{acc} - P_{ax}) \times N_{ax}}{P_{acc} \times N_{acc}}\% \quad (8)$$

Even approximation of only one signal has to be optimized. For example, if a signal can be truncated by 3 bits and 40 iterations or by 2 bits and 60 iterations, it has to be assessed which option saves more energy. This can be done by synthesizing the design using ASIC tooling and performing a power simulation. However, it is infeasible for such a complex architecture as it requires a significant amount of time for design, compilation, and simulation. For this reason, to analyze the effect of approximations and understand how much power is saved for each approximation, the unit gate model is used [21]. This gate model counts the number of gates required for each hardware configuration. In this model the NOT gate is equal to 0.5 gates, two-input gates AND, OR, NAND, NOR are assigned a value of 1, and the XOR gate has a cost of 2 gates. If an approximation is applied in such a way that all the logic after an approximated multiplier is simplified, these effects are also included in the model. The gate model outputs a single number which corresponds to the number of gates required by the whole datapath – multipliers, squarers, adders, and registers in the design. As the unit gate model is more relevant to area estimations and the optimization goal is the reduction of power consumption, an assumption is made that the smallest architecture will have the smallest power. In that sense the purpose of the model is to find the minimum, the real percentage of saving may be different, and will be reported by synthesizing the best designs and performing the power simulations in ASIC flow (see Section 4).

In Eq. (8), P_{acc} is a constant number because the accurate core is not modified. N_{acc} is also a constant, which is equal to the number of required iterations if only an accurate core is utilized (92 in our case). Every approximation applied to the approximate core will change P_{ax} to some number smaller than P_{acc}, and it will also change N_{ax}. There is a trade-off between P_{ax} and N_{ax}, as smaller P_{ax} corresponds to more coarse approximations, leading to smaller number of iterations which can survive this approximation. Therefore, the optimization goal is to find an approximate architecture which minimizes P_{ax} and maximizes N_{ax} in such a way that the savings in Eq. (8) are maximized.

To determine the energy-efficiency offered by an approximation technique, the following steps are applied: (1) Apply the approximation technique. (2) Run the algorithm with all iterations on the approximate core (maximum possible savings). If the results are not acceptable, decrease the number of approximate iterations and map the other iterations to the accurate core. The number of approximate iterations is decreased until the results become acceptable. (3) Compute the savings using Eq. (8), where power (P) values are determined by the unit gate model. (4) Increase the level of approximation and repeat steps 1-3. Select the approximate design which provides the maximum E_s. (5) Implement the selected architecture in a hardware descriptive language, synthesize the design and perform the power simulation to obtain the synthesized area and power numbers.

4 EXPERIMENTAL RESULTS

To quantify the reduction in energy-consumption (or increase in energy efficiency) offered by our proposed Least Squares (LS) accelerator design, we have conducted an energy-quality study based on the StEFCal (LS) algorithm. In the following subsections, we discuss our experimental setup, and the results obtained for accurate and approximate cores.

4.1 Experimental Setup

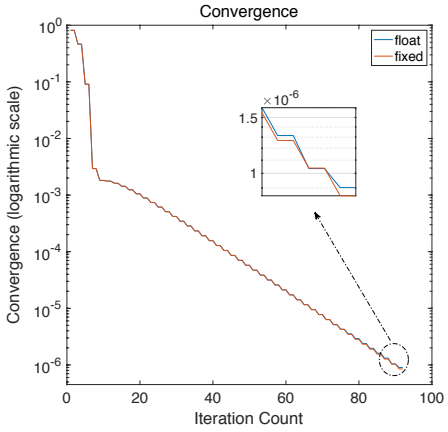
The optimized fixed-point version (accurate core) and the approximation techniques (for approximate core) are implemented in the MATLAB for quality analysis. Similar to [8], chip-area and power estimations have been obtained by synthesizing the designs in Synopsys ASIC flow (Design Compiler and Power Compiler) for the TSMC 40nm Low Power (TCBN40LP) technology library.

In our hardware analysis, an operating frequency of 50 MHz is selected in order to have enough timing-slack left for each design. A reasonable slack in each design helps to avoid different levels of optimizations provided by the tool for different designs, e.g., the synthesis tool may use a ripple-carry adder in one design and a fast adder structure requiring more power (for example a carry-lookahead adder) in another design, which brings different area/power results complicating comparisons and savings estimations. Therefore, fixing the frequency limit to 50MHz brings a legitimate comparison of various designs. However, our proposed design can also be synthesized at higher frequencies as per the design requirements.

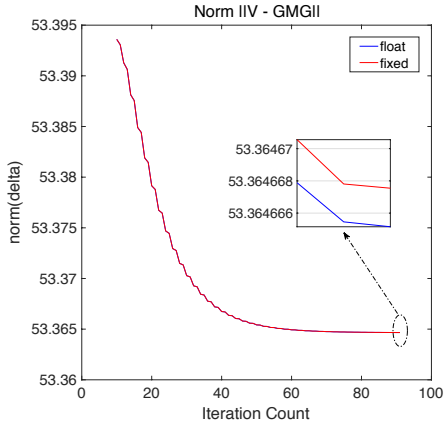
4.2 Accurate Core Results

The accurate design is implemented without using behavioral descriptions for the multipliers and squarers in VHDL. They are implemented by constructing the partial product matrix consisting of AND gates which are then summed by an adder tree selected by the synthesis tool. So the approximations can be applied directly to the partial product matrix for an approximate core. The power consumption and chip area for the accurate core are 3.5530 mW and 27023 μm^2 respectively.

The comparison between the double-precision floating-point (float) and optimized fixed-point (fixed) versions can be seen in



(a)



(b)

Figure 4: Comparison between the double-precision floating-point (float) and optimized fixed-point (fixed) StEFCal processing, the latter is referred to as the *accurate core.*

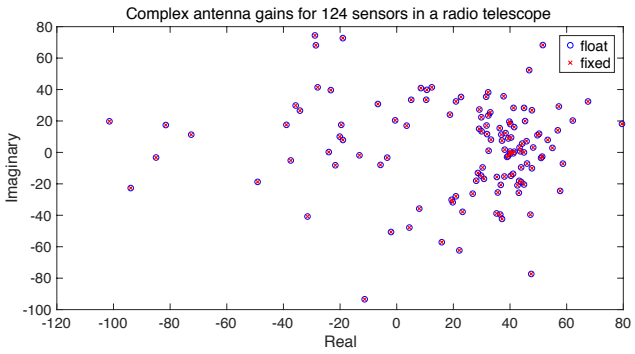
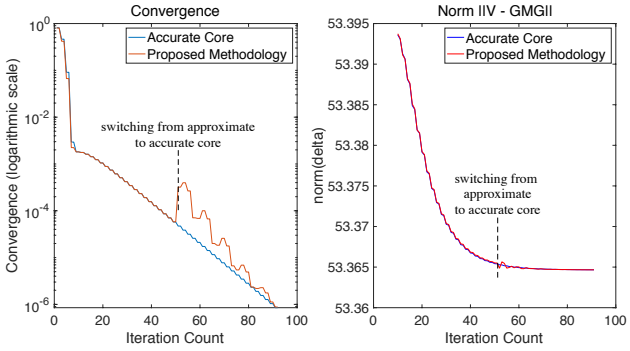
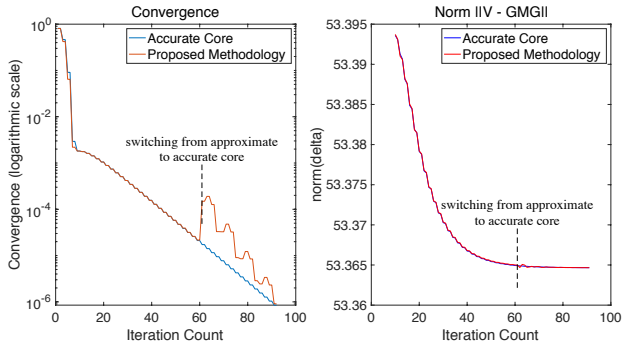


Figure 5: Gains computed by double-precision floating-point (float) and optimized fixed-point (fixed) processing.



(a) Biased



(b) Unbiased

Figure 6: Effect of partial product truncation for biased and unbiased cases.

Fig. 4, where the latter refers to the *accurate core*. Fig. 4a shows that both versions converge to 10^{-6} in 92 iterations. Fig. 4b shows the difference ||V-GMG||. The comparison between the computed gains is shown in Fig. 5, demonstrating identical results for the 124 complex gains.

4.3 Approximate Core Results

To optimize an approximate core, three approximations are applied in this work, namely: truncation of partial products, truncation of inputs, and DRUM [11].

The approximations are applied to the four multipliers of the MAC unit and to the two squarers of the SAC unit. First, the multipliers and squarers are approximated one by one, while the others remain accurate. This way an optimal approximation level for each of the units individually is determined. Then, the multipliers and squarers are approximated in pairs, around the determined numbers. This approach does not guarantee to find the optimal approximate designs, but it allows to reduce the number of time-consuming simulations and provides a near-optimal solution for each method. The synthesized results for area and power, and the quality analysis of the chosen (near-optimal) designs for each approximation method are discussed in the following subsections.

4.3.1 Truncation of Partial Products. The power consumption and chip area for the approximate core (utilizing the truncation of partial products) are 2.2280 mW and 22523 μm^2 respectively. The

quality comparison between this design and the optimized fixed-point (accurate core) design is shown in Fig. 6. In the proposed architecture, the first 51 (61 for the unbiased case) iterations are computed on the approximate core. After switching to the accurate core, two phenomena can be noticed: (1) the precision-oriented metric, i.e., convergence, experiences jumps because of the increase (change) in computation precision, see Fig. 6-left. (2) the deviation from the accurate solution gradually decreases, see Fig. 6-right. Overall, the solution converges to an acceptable value in the same number of iterations.

The truncated architecture has a negative bias as all the deleted partial products are equal to zero. The unbiasing is performed by placing three different initial values in each of the three accumulators, see Fig. 3. These values are determined by performing two simulations in parallel, one with the optimized accurate structure, and one with the approximate structure. The mean of the differences between the corresponding values is computed for each accumulator and the value is added to the corresponding accumulator. The effect of unbiasing is that the StEFCal is able to withstand the truncation of partial products for 61 iterations, ten iterations more compared to the biased case.

4.3.2 Truncation of Inputs. The power consumption and chip area for the approximate core (utilizing the truncation of inputs) are 2.0821 mW and 20604 μm^2 respectively. The following number of input bits have been truncated (as compared to Table 1) to obtain the approximate core: 8 bits for e_sac, f_sac, and e_mac; and 12 bits for f_mac. However, no further truncation was possible for h and t signals. The quality comparison between this design and the optimized fixed-point (accurate core) design is shown in Fig. 7. Similar to the truncation of partial products, the unbiasing technique within input-truncation increases the number of approximate iterations (by 12) as compared to that of the biased ($N_{ax} = 52$) case.

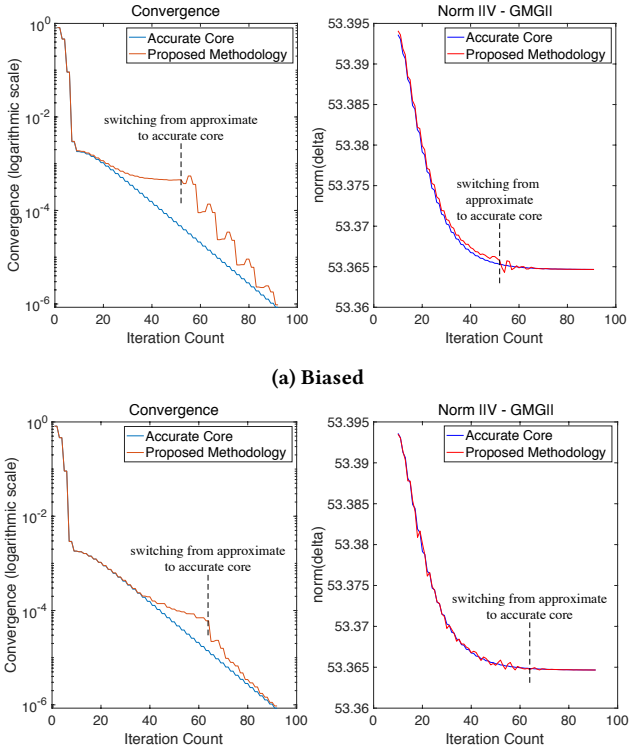
4.3.3 DRUM. We have also utilized a Dynamic Range Unbiased Multiplier (DRUM) technique because it provides a near-to-zero mean error profile [11]. Unbiasing by the accumulator initialization is not effective for this method, presumably because it is already unbiased, as can be suggested by the fact that it can be applied for 60 iterations, more than the truncation methods in the biased cases.

The power consumption and chip area for the approximate core (utilizing the DRUM methodology) are 2.7114 mW and 20835 μm^2 respectively. The quality comparison between this design and the optimized fixed-point (accurate core) design is shown in Fig. 8.

4.4 Overall Energy Savings

The overall energy-savings achieved by the proposed heterogeneous LS accelerator design by utilizing three approximation techniques are presented in Table 2. These savings have been calculated using Eq. (8), which compares our proposed accelerator design (a two core architecture) with that of a single core architecture and assumes that only one core is switched on at a certain period of time in our proposed design.

Table 2 also shows the number of iterations that can run on the approximate core (N_{ax}) are improved (increased) while using unbiased (UnB) design techniques. For example, in case of truncation of partial products, the first 51 ($N_{ax} = 51$) iterations can be processed



(a) Biased

(b) Unbiased

Figure 7: Effect of the truncation of inputs for the biased and unbiased cases.

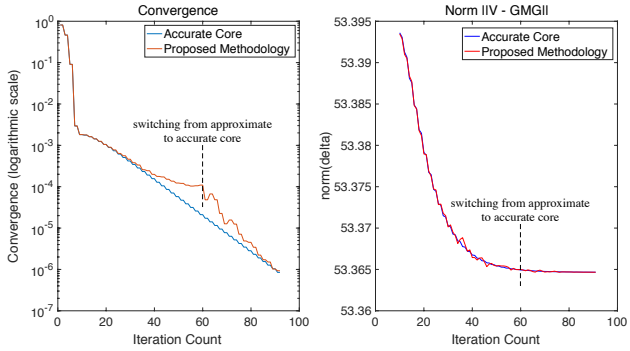


Figure 8: Effect of the DRUM technique.

in the approximate core, while the rest (92-51=41) are processed in the accurate core. However, when the unbiasing technique is utilized, the first 61 ($N_{ax} = 61$) iterations can be processed in the accurate core. Therefore, the unbiased case brings better energy savings. The truncation of inputs is the most effective method to apply to the StEFCal algorithm. It allows saving more than the other methods while having the smallest area overhead.

Table 2: Energy savings based on employed approximations in the proposed heterogeneous Least Squares (LS) accelerator. The proposed accelerator design provides up to 24% and 29% of energy improvements for biased and unbiased truncation techniques respectively.

Approximation Method	N_{ax}		AO %	Energy Savings %	
	Bias	UnB		Bias	UnB
Truncation of partial products	51	61	83	20	25
Truncation of inputs	52	64	76	**24**	**29**
DRUM	-	60	77	-	15

N_{ax} represents the number of iterations that can run on the approximate core. Bias and UnB represent biased and unbiased approximations.
AO is area overhead, which may be translated into throughput increase as discussed in Section 4.5.

4.5 Discussion and Future work

Table 2 compares our proposed accelerator design (a two core architecture) with that of a single core architecture. Therefore, we can see area overhead (AO), which is due to the addition of the approximate core. It is to be noted that a CPU (see Fig. 1) may also utilize both cores simultaneously for different processes, e.g., processing two calibration operations in parallel. This may translate the area overhead to throughput increase while having the same energy efficiency benefits for each process.

It is to be noted that the case study of radio astronomy calibration (StEFCal) processing has been discussed to show how to employ the proposed heterogeneous accelerator architecture by using a single time slot LOFAR [7] data. Nevertheless, an increased data set would better provide the allowable number of iterations to run on an approximate core, therefore, a better estimate of energy savings that can be achieved using the proposed accelerator architecture.

5 CONCLUSIONS

A novel heterogeneous architecture for Least Squares (LS) acceleration has been presented targeting the energy-efficiency. We have shown how a combination of optimized-precision (accurate) and reduced-precision (approximate) computing cores can be utilized to provide acceptable quality output while reducing energy consumption as compared to that of an accurate optimized architecture. Our design methodology exploits the inherent error-resilience of an iterative workload to leverage an approximate computing core for processing the initial iterations of the LS algorithm. A case study of radio astronomy calibration processing has shown up to 24% (and 29%) of energy savings as compared to that of the accurate counterpart for biased (and unbiased) design. However, it is to be noted that the proposed methodology is independent of the application, provided that the computation pattern is iterative in nature. We have utilized input truncation, partial product truncation, and DRUM multipliers as the means of approximations within an approximate core. However, our methodology can utilize any approximation technique that is promising for the target application.

ACKNOWLEDGMENTS

This work was conducted in the context of the ASTRON and IBM joint project, DOME, funded by the Netherlands Organization for Scientific Research (NWO), the Dutch Ministry of EL&I, and the Province of Drenthe.

REFERENCES

[1] Benjamin Barrois, Olivier Sentieys, and Daniel Menard. 2017. The hidden cost of functional approximation against careful data sizing—A case study. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE)*. 181–186.
[2] Shuvra S Bhattacharyya, E.F Deprettere, Rainer Leupers, and Jarmo Takala. 2018. *Handbook of signal processing systems*. Springer.
[3] M.A. Hanif S. Ullah G. Mazaheri A. Kumar B.S. Prabakaran, S. Rehman and M. Shafique. 2018. DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems. In *Proceedings of the DATE*. IEEE, 917–920.
[4] M.D. Buhari, G.Y. Tian, R. Tiwari, and A.H. Muqaibel. 2018. Multicarrier SAR Image Reconstruction Using Integrated MUSIC-LSE Algorithm. *IEEE Access* 6 (2018), 22827–22838. DOI: http://dx.doi.org/10.1109/ACCESS.2018.2817359
[5] Vinay K Chippa, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan. 2013. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual DAC*. ACM, 113.
[6] D. Esposito, A.G.M. Strollo, and M. Alioto. 2017. Low-power approximate MAC unit. In *Proceedings of the 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. 81–84.
[7] Van Haarlem et al. 2013. LOFAR: The low-frequency array. *Astronomy & Astrophysics* 556 (2013), A2.
[8] G.A. Gillani, M.A. Hanif, M. Krone, S.H. Gerez, M. Shafique, and A.B.J. Kokkeler. 2018. SquASH: Approximate Square-Accumulate With Self-Healing. *IEEE Access* 6 (2018), 49112–49128. DOI: http://dx.doi.org/10.1109/ACCESS.2018.2868036
[9] G.A. Gillani and A.B.J. Kokkeler. 2017. Improving Error Resilience Analysis Methodology of Iterative Workloads for Approximate Computing. In *Proceedings of the Computing Frontiers Conference*. ACM, 374–379.
[10] Jie Han and Michael Orshansky. 2013. Approximate Computing: An Emerging Paradigm for Energy-Efficient Design. In *European Test Symposium (ETS'13)*. IEEE, 1–6.
[11] S. Hashemi, R.I. Bahar, and S. Reda. 2015. DRUM: A Dynamic Range Unbiased Multiplier for approximate applications. In *2015 International Conference on Computer-Aided Design (ICCAD)*. IEEE/ACM, 418–425.
[12] Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, and Jie Han. 2017. A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13, 4 (2017), 60.
[13] R. Jongerius, S. Wijnholds, R. Nijboer, and H. Corporaal. 2014. An End-to-End Computing Model for the Square Kilometre Array. *Computer* 47, 9 (Sep. 2014), 48–54. DOI: http://dx.doi.org/10.1109/MC.2014.235
[14] Jin Miao, Ku He, Andreas Gerstlauer, and Michael Orshansky. 2012. Modeling and synthesis of quality-energy optimal approximate adders. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '12)*. IEEE, 728–735.
[15] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han. 2018. Scalable Construction of Approximate Multipliers With Formally Guaranteed Worst Case Error. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 11 (Nov 2018), 2572–2576. DOI: http://dx.doi.org/10.1109/TVLSI.2018.2856362
[16] S. Naghibzadeh, A.M. Sardarabadi, and A. van der Veen. 2016. Radioastronomical image reconstruction with regularized least squares. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3316–3320.
[17] V.T. Olafsson, D.C. Noll, and J.A. Fessler. 2018. Fast Spatial Resolution Analysis of Quadratic Penalized Least-Squares Image Reconstruction With Separate Real and Imaginary Roughness Penalty: Application to fMRI. *IEEE Transactions on Medical Imaging* 37, 2 (Feb 2018), 604–614. DOI: http://dx.doi.org/10.1109/TMI.2017.2768825
[18] Nicola Petra, Davide De Caro, Valeria Garofalo, Ettore Napoli, and Antonio GM Strollo. 2010. Truncated binary multipliers with variable correction and minimum mean square error. *IEEE Transactions on Circuits and Systems I: Regular Papers* 57, 6 (2010), 1312–1325.
[19] Stefano Salvini and Stefan J Wijnholds. 2014. Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. *Astronomy & Astrophysics* 571 (2014), A97.
[20] Muhammad Shafique, Rehan Hafiz, Semeen Rehman, Walaa El-Harouni, and Jörg Henkel. 2016. Invited: Cross-layer approximate computing: From logic to architectures. In *Proceedings of the Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*. IEEE, 1–6.
[21] A. Tyagi. 1993. A reduced-area scheme for carry-select adders. *IEEE Trans. Comput.* 42, 10 (Oct 1993), 1163–1170. DOI: http://dx.doi.org/10.1109/12.257703