Improving Error Resilience Analysis Methodology of Iterative Workloads for Approximate Computing

G.A. Gillani 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

Assessing error resilience inherent to the digital processing workloads provides application-specific insights towards approximate computing strategies for improving power efficiency and/or performance. With the case study of radio astronomy calibration, our contributions for improving the error resilience analysis are focused primarily on iterative methods that use a convergence criterion as a quality metric to terminate the iterative computations. We propose an adaptive statistical approximation model for high-level resilience analysis that provides an opportunity to divide a workload into exact and approximate iterations. This improves the existing error resilience analysis methodology by quantifying the number of approximate iterations (23% of the total iterations in our case study) in addition to other parameters used in the state-of-the-art techniques. This way heterogeneous architectures comprised of exact and inexact computing cores and adaptive accuracy architectures can be exploited efficiently. Moreover, we demonstrate the importance of quality function reconsideration for convergence based iterative processes as the original quality function (the convergence criterion) is not necessarily sufficient in the resilience analysis phase. If such is the case, an additional quality function has to be defined to assess the viability of the approximate techniques.

CCS CONCEPTS

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

KEYWORDS

Error resilience analysis, iterative workloads, quality function, approximate computing, heterogeneous architectures

ACM Reference format:
G.A. Gillani and A.B.J. Kokkeler. 2017. Improving Error Resilience Analysis Methodology of Iterative Workloads for Approximate Computing. In *Proceedings of CF'17, Siena, Italy, May 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3075564.3078891

1 INTRODUCTION

Computational workload analysis provides the essential insights about the applications that help in the optimization process for mapping effectively on the contemporary heterogeneous architectures to achieve maximum performance and energy efficiency [9]. Approximate computing—sometimes coined as best-effort computing or soft computing—can be regarded as aggressive optimization because it allows legitimate inexactness and provides results with the bare minimum accuracy to improve run-time and/or power consumption. The aforesaid advantages have been demonstrated for error resilient applications such as multimedia digital signal processing, search engines and scientific computing [13, 27]. The major approximation techniques include loop/code perforation [5, 11], pattern reduction [21], thread fusion [22], approximate memoization [1], voltage over-scaling [8, 14] and using the approximate storage/arithmetic units/accelerators [3, 24, 25] to name some.

Error resilience is inherent to an application due to its possibly redundant/noisy real-time inputs, probabilistic or self-healing computational patterns, and/or a range of acceptable outputs [2]. However, there are error-sensitive parts or kernels within every error-resilient application. Therefore, it is important to analyse the applications for error resilience to separate the error-sensitive parts from that of error-tolerant parts and to get insights of promising approximation techniques before employing the implementation efforts [2, 11, 12, 18, 19, 23]. The procedure of the error resilience analysis is to apply approximations offline while monitoring the quality function to verify that the approximations produce acceptable results. In this way, approximate computing techniques are substantiated for a workload that can be employed in the implementation/online phase to achieve the desired benefits.

Heterogeneous architectures have the ability to handle various workloads efficiently while using different power and performance trade-off computing nodes, e.g. ARM's big.LITTLE architecture [6]. However, in the approximate computing domain, the definition of heterogeneous architecture extends further to include exact and inexact computing units, where control instructions and sensitive computational parts run at precise cores while the error resilient parts run at the error-prone cores to achieve the overall efficiency in speed and energy [7, 23]. Moreover, with the advent of accuracy-configurable architectures like adaptive voltage over-scaling [8] and adaptive processors comprised of configurable adder/multiplier blocks [25], where the quality-cost[1] trade-off can be controlled at run time, it is of remarkable importance to analyse a workload for Adaptive Error Resilience (AER) as well. We believe that AER has the potential to reveal approximation opportunities even in strict quality function (relatively less error resilient) workloads, where

[1]Cost refers to power consumption, latency and on-chip area required for maintaining the specific quality of output

they can be processed adaptively for varying approximation levels to gain energy/performance benefits.

Furthermore, as iterative methods are common candidates for approximate computing like K-means [10], GLVQ training [2], and model predictive control [17]; we propose the Adaptive Statistical Approximation Model (Adaptive-SAM) to analyse the high-level approximation space of iterative workloads. Adaptive-SAM performs better than the Statistical Approximation Model (SAM) [2] by quantifying the number of approximate iterations (NAI) in addition to the statistical approximation space that will be discussed in Section 2, and therefore can better exploit the heterogeneous/accuracy-configurable architectures by assigning resilient iterations to the approximate computing cores/modes and the sensitive iterations to the exact counterparts.

In this paper, we elaborate the Adaptive-SAM analysis methodology (Section 3) and present its significance with a case study of the radio astronomy calibration application (Section 4). In practice, the error resilience or the approximation space of an application is quantified by injecting the errors defined by the approximation models (statistical or technique specific) and monitoring the overall output of the application in compliance with the quality function. The range of error injection within an approximation model for which the quality function is satisfied can be regarded as the approximation space of the application. Therefore, defining a quality function is a very deliberate task in the approximate computing domain. In Section 5, we demonstrate that the original quality function of an iterative process may not be necessarily sufficient in the error resilience analysis procedure, which requires defining an additional quality function to serve the purpose.

2 RELATED WORK

This section reviews some state-of-the-art approximate computing architectures that motivate statistical and adaptive-statistical error resilience analysis. Moreover, we discuss contemporary analysis methodologies/tools and the need of Adaptive-SAM analysis.

The quality-cost trade-off for an accurate multiplier (AccMul) and two approximate multipliers ($AxMul_1$ and $AxMul_2$) have been presented in [25]. These multipliers are 2x2 (input size=2 bits for each operand) that can construct the higher order multipliers e.g. 4x4, 8x8 and so on. $AxMul_1$ has better area and power costs as compared to AccMul with 1 error case (error magnitude=2) out of 16 possible cases. $AxMul_2$ is even more energy efficient as compared to $AxMul_1$. However, the error rate for $AxMul_2$ is 3 out of 16 possible cases (error magnitude=1). In short, $AxMul_2$ has higher error rate and lower error magnitude as compared to $AxMul_1$ while offering better energy efficiency. The selection from such design choices is based on workload's error resilience characteristics that whether the target workload can tolerate higher error rate or higher error magnitude. Moreover, this design space (number of alternatives) becomes larger for higher order multipliers that can have a number of such multipliers (approximate or accurate) and a number of adders (approximate or accurate) for the adder tree to compute the final higher order product [16]. For that matter, it is important to analyse a workload for statistical error distribution that can provide the error-resilience profile of an application based on statistical parameters: error mean (EM), error predictability (EP) and error rate (ER). EP and EM determine the magnitudes of the injected errors and correspond to the variance and mean of the normal error distribution while ER defines the rate at which errors are injected in the approximation analysis. The aforesaid error resilience profile helps to reduce the available design space in order to choose the best possible quality-cost design alternative.

The idea of adaptive accuracy or accuracy-configurable architectures composed of approximate accelerators has been proposed in [25]. These architectures contain accuracy-configurable operators, such as multipliers and adders, which can change the computation mode from accurate to approximate and vice-versa during the run-time. This helps in run-time adjustment of quality-cost trade-off based on the workload's error resilience. Moreover, the adaptive voltage over-scaling (AVOS) [8] has shown the improvements in power efficiency (25% to 30%) at negligible quality loss for texture decompression application. The aforesaid scheme reduces the supply voltage till the limited number of errors are introduced and can increase the voltage again to attain error-free operation. Therefore, AVOS can also adjust the quality-cost trade-off during run time. In this context, we argue that the aforementioned adaptive accuracy architectures can be better exploited provided that the error resilience profile of an iterative workload also quantifies the number of approximate iterations in addition to the statistical parameters (EM, EP and ER) in order to adaptively apply approximations during the run-time to gain target benefits.

Over the past few years, many tools have been presented to assess the applications for intrinsic error resilience. Quality of service (QoS) profiling utilized loop perforation as a compiler pass to skip some iterations in order to identify sub-computations that can be replaced with less accurate counterparts [11]. Intel's open-source approximate computing toolkit (iACT) assesses the scope of approximations within the applications using programmer annotated pragmas to analyse the programmer guided parts of the code [12]. iACT has the ability to apply static approximate transformations such as precision scaling, run-time approximations like memoization, and the noisy hardware effects during the error resilience analysis simulations. Automatic sensitivity analysis for approximate computing (ASAC) applies statistical perturbation of program data to study the overall effects on the quality of workload's output [18]. Program analysis for approximation-aware compilation (PAC) provides a relatively faster way to study the degree of accuracy required by each component in an application to achieve an overall quality of output [19]. Approximate C compiler for energy and performance trade-offs (ACCEPT) is another open-source tool that applies a conservative approach to perform safe approximate relaxation analysis within a workload [23]. The aforesaid tools are limited in assessing the error-resilience of a workload as they do not cover all the approximation strategies and they do not provide a statistical error resilience profile to reduce the available design space (alternatives) as discussed earlier.

On the other hand, the application resilience characterisation (ARC) framework [2] is a state-of-the-art methodology that includes the statistically distributed error injection model to generate the statistical profile of a workload. The overview of the ARC methodology is shown in Fig. 1. The first step is to identify the dominant kernels based on their run-time share. The kernels that run for at least 1% of the total execution time are selected to perform analysis.

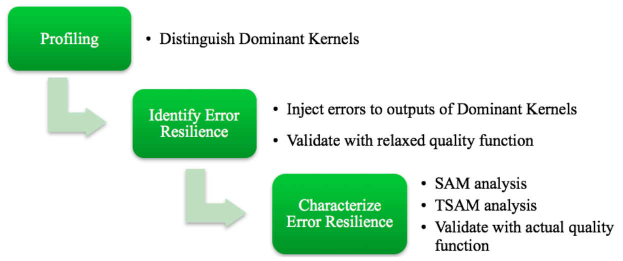

Profiling
• Distinguish Dominant Kernels
Identify Error Resilience
• Inject errors to outputs of Dominant Kernels
• Validate with relaxed quality function
Characterize Error Resilience
• SAM analysis
• TSAM analysis
• Validate with actual quality function

Figure 1: Error resilience analysis methodology - ARC

Secondly, the error resilience is identified by injecting random errors in the outputs of the dominant kernels and the overall output of the application is compared with a relaxed quality function to distinguish potentially resilient kernels from that of the sensitive ones. Relaxed quality function means that the error injected output is only checked for relatively bigger errors rather than the actual quality required by the application. Finally, high-level and Technique Specific Approximation Models (TSAM) are applied to characterize the resilience by using the actual quality function. The high-level model is also termed as Statistical Approximation Model (SAM) because it injects errors based on the statistical (Gaussian) distribution. This defines a high-level approximation space of an application by providing a quality profile based on statistical parameters and can help to narrow down technique-specific approximation choices such as arithmetic operations, data representation and algorithm level approximations [2]; for instance, choosing AxMul$_1$ or AxMul$_2$ as discussed earlier. However, in order to better utilize the adaptive accuracy architectures, we need Adaptive-SAM analysis of iterative workloads that can quantify the adaptive resilience by identifying the number of approximate iterations in addition to the statistical parameters.

3 ADAPTIVE STATISTICAL APPROXIMATION MODEL (ADAPTIVE-SAM)

As discussed earlier, our aim is to improve the high-level error resilience analysis of iterative workloads in order to better exploit accuracy configurable and heterogeneous architectures. In this regard, we present Adaptive-SAM that can replace SAM in the error resilience analysis methodology shown in Fig. 1 to provide the number of approximate iterations (NAI) in addition to statistical parameters of approximation space (EM, EP and ER).

To elaborate the Adaptive-SAM methodology, Algorithm 1 shows the pseudo code of an iterative workload example. Although such workloads may have any number of inputs and outputs, we assume two inputs (x_1, x_2) and one output (K_op) for the sake of simplicity. The algorithm iterates to improve the output by utilizing the inputs and previously computed result (last iteration), where i is the current iteration and N is the maximum number of iterations. The algorithm also uses an intermediate variable (im_var) that is computed by calling a kernel: function_im, having input arguments as x_1 and the computed output of previous iteration $K\_op(i-1)$. Then the output is computed by calling another kernel: function_Kop that has input arguments as x_2 and im_var. Subsequently, the convergence metric ($convergence\_met$) is computed by calling function_conv kernel that considers the current output $K\_op(i)$ and the previously computed output $K\_op(i-1)$ to generate the convergence metric. Iterative workloads may use different arithmetic operations within function_conv, but the aim is generally to compute the improvement in result within two consecutive iterations. Finally, Algorithm 1 checks the convergence metric for the allowed tolerance limit (tol) based on the quality function of the iterative workload. If the convergence is reached, the iterative process is terminated to provide the final outcome.

Algorithm 1 An Iterative Workload Example.

```
Input: $x_1, x_2$
Output: $K\_op$
1: Initialize $K\_op(0)$
2: for $i = 1, 2, ..., N$ do
3:     $im\_var$ = function_im($x_1, K\_op(i-1)$);
4:     $K\_op(i)$ = function_Kop($im\_var, x_2$);
5:     $convergence\_met$ = function_conv($K\_op(i), K\_op(i-1)$);
6:     if ($convergence\_met \leq tol$) then
7:         break;        // convergence reached
8:     end if
9: end for
```

As discussed in Section 2, the error resilience methodology identifies the dominant kernels in the first place. These kernels are selected based on the percentage of their run-time or floating point operations (FLOPs) relative to the overall workload. The kernels that have the higher share are regarded as dominant kernels as it is likely to attain desired benefits (area, power or latency) while approximating them. In Algorithm 1, we assume that function_Kop is a dominant kernel to explain Adaptive-SAM analysis. Fig. 2 shows the signal flow of Adaptive-SAM. We assume N number of iterations/stages where i is the current iteration ($1 \leq i \leq N$). The output of iteration i of a dominant kernel ($K\_op(i)$) is added to a Gaussian error ($Error(i)$) if the randomized errors ($ER\_rand(i)$) and approximate iterations flag (ax_iter_flag) allow error injection to this stage. This flag is controlled via a parameter: NAI, which is the number of initial iterations to be instrumented with errors. It is important to note that $Error(i)$ is a function of the normally distributed EP and EM. Finally, the approximate (Ax) kernel output is assessed for quality function compliance to test the validity of the approximation space. Therefore, Adaptive-SAM quantifies the high-level error resilience of an application based on the acceptable range of normally distributed errors for the quantified approximate iterations. This helps to assign the approximate iterations to the inexact cores (fast and/or energy efficient) with the specified approximation space while running the exact iterations on the exact cores (slow and/or high power) to better exploit the heterogeneous architectures. Similarly, in case of accuracy configurable architectures, the approximate iterations can run in approximate (error prone) hardware modes at higher energy efficiency and/or performance.

4 EXPERIMENTAL SETUP AND RESULTS

We have applied SAM and Adaptive-SAM on an iterative workload, which is the radio astronomy calibration algorithm (StEFCal) [20].

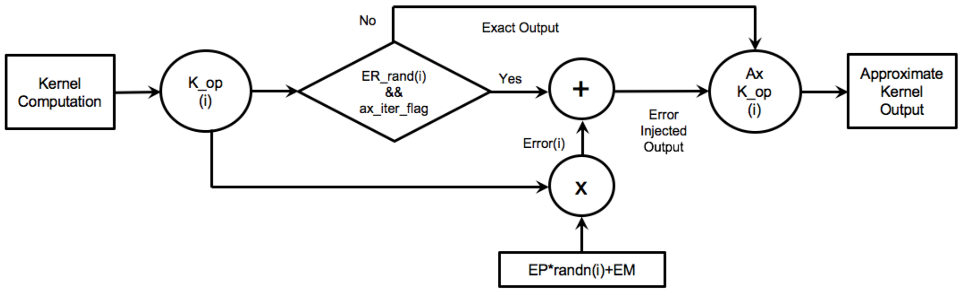


Figure 2: Proposed high-level error resilience analysis model - Adaptive-SAM

The said algorithm has been instrumented with the run-time statistical error injection in Matlab. The quality function compliance has also been checked at run-time to determine the statistical error tolerance limits. In this way, we have quantified the high-level error resilience for StEFCal by using SAM and Adaptive-SAM models.

4.1 Radio Astronomy Calibration – StEFCal

Radio astronomy studies celestial objects at radio frequencies. The sky spatial intensity distributions are estimated over the field of view in a radio telescope to generate the sky images [15]. Perhaps, the signal processing pipeline in radio astronomy can be considered as error-tolerant application because it reflects the attributes of the inherent resilience explained in Section 1; namely: real-life/redundant data input, and approximate/statistical/self-healing computation patterns. Therefore, we analyse a radio astronomy application (calibration) as a case study to quantize error resilience based on SAM and Adaptive-SAM techniques. The calibration algorithm, also known as StEFCal (statistically efficient and fast calibration), is a strict quality-of-service alternating direction implicit (ADI) method that estimates complex antenna gains (g_p) for the P sensors in a radio telescope [20]. The algorithm computes a g_p vector based on a measured signal/array covariance matrix (R) and the model covariance matrix (M), where each ADI iteration (i) computes linear least squares problems as:

$$g_p^{[i]} = \frac{R_{:,p}^H . Z_{:,p}^{[i-1]}}{(Z_{:,p}^{[i-1]})^H . Z_{:,p}^{[i-1]}} \tag{1}$$

Where $R_{:,p}^H$ is the hermitian transpose of array covariance matrix's pth column, and $Z_{:,p}^{[i-1]}$ is the element-wise product of $g_p^{[i-1]}$ and the model covariance matrix's pth column ($M_{:,p}$). It should be noted that $g_p^{[i-1]}$ is the antenna gains vector computed in the previous iteration. In our experiments, representative input data (R and M matrices) of the LOFAR facility [26] has been utilized (for $P = 124$).

The convergence criterion of StEFCal is based on the relative difference in length of consecutive iterations' solution vectors in the Euclidean space [20],

$$\text{Convergence} = \frac{||g^{[i]} - g^{[i-1]}||_F}{||g^{[i]}||_F} \leq 1.10^{-6} \tag{2}$$

In our initial experiments, we defined our quality function solely based on the convergence criterion. However, this proved to be insufficient in the approximate computing domain as explained in Section 5. For that matter, we have defined an additional quality metric: Diff_rel, which is the relative difference in length between the exact (ex) and approximate (ax) solution vectors,

$$\text{Diff\_rel} = \frac{||g_{ex}^{[i]} - g_{ax}^{[i]}||_F}{||g_{ex}^{[i]}||_F} \leq 1.10^{-5} \tag{3}$$

The tolerance limit in Eq. 3 is higher than that of Eq. 2 due to assumed acceptable error for approximately computed output. Furthermore, we assume that the quality acceptance range is satisfied (QARS), if and only if both the convergence (Eq. 2) and Diff_rel (Eq. 3) criteria are satisfied.

4.2 High-level Error Resilience Analysis

To apply SAM and Adaptive-SAM, the initial steps are the same as elaborated in Section 2, i.e. distinguishing dominant kernels that run at least 1% of the total execution time and identifying error resilience by injecting the random errors in the outputs of the dominant kernels. Three dominant kernels have been identified in StEFCal by using [4], which are Z vector computation (27% of the computational load) and the two dot products (72% of the computational load) as shown in Eq. 1. We have analysed the aforesaid three kernels for high-level error resilience. As the response of the dot products is almost similar to Z, we only present the simulation results for Z computation here that are sufficient to demonstrate the comparison between SAM and Adaptive-SAM outcomes.

Fig. 3 shows the response of StEFCal for SAM analysis. It can be seen that Diff_rel is increased as we increase the error mean (EM) and error rate (ER), while it has no remarkable effect due to changes in error predictability (EP). However, the convergence limit is only achieved at minimum EP and maximum ER. In practice, there can be some cases where the benefits of energy efficiency/performance are achieved even if the approximate computation runs for more iterations than the exact computation [17]. Nonetheless, for the sake of simplicity, we assume that the convergence limit is satisfied when the number of iterations required to converge for error-injected computation is less or equal to that of exact computing counterpart. While quantifying the error resilience intrinsic to StEFCal for SAM analysis, QARS is achieved for $EM \leq 0.002\%$, $EP \leq 2.10^{-4}$ at $ER = 100\%$. This shows a very small approximation space.

The results of Adaptive-SAM analysis of StEFCal are shown in Fig. 4 for various number of approximate iterations (NAI) and error

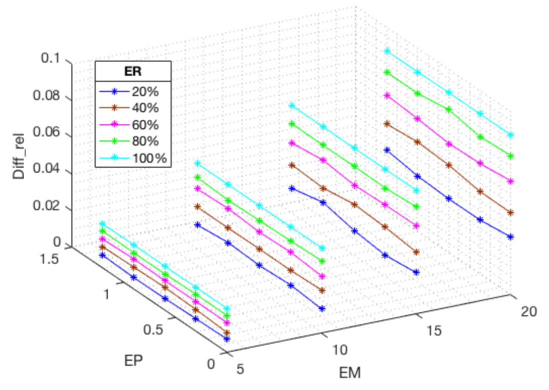


Figure 3: StEFCal response for SAM analysis

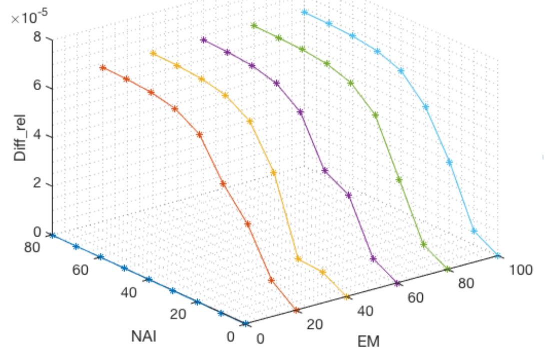


Figure 4: StEFCal response for Adaptive-SAM analysis

mean (EM) values. In this case, error predictability and error rate are fixed to the values that allow the solution to converge (EP=0 and ER=100%). As expected, the Diff_rel decreases with the decrease in NAI and EM. This suggests that the more the number of exact computing iterations, the better the quality of output and vice versa. While quantifying the error resilience, QARS is achieved for $NAI \leq 23\%$, $EM \leq 12\%$, $EP \leq 0.2$ at $ER = 100\%$; this shows the availability of an approximation space for 23% of the total iterations. Therefore, Adaptive-SAM reveals additional error resilience opportunities by quantifying the number of approximate iterations in addition to EM, EP and ER for the iterative workloads.

5 SIGNIFICANCE OF QUALITY FUNCTION RECONSIDERATION

In error resilience analysis, the approximation models are applied during run-time and are validated using the quality function. This makes the selection of a quality function very crucial as it can possibly reject or accept some approximation strategies unfairly. Therefore, in order to utilize the original quality metric of an iterative application (convergence criterion) in the approximate computing domain, it has to be reconsidered rigorously to attain reliable insights about the approximation space. In our case study of an iterative workload (StEFCal), the convergence criterion is utilized for the exact computing case. This computes the relative distance in Euclidean space between the current and previous solution vectors and is assumed to be satisfied if the improvement within two consecutive iterations is less or equal to 1.10^{-6}, which means that the solution has already been converged and no further precision can be achieved by computing more iterations. However, in the error resilience analysis process, it can not be guaranteed that the acceptable solution is achieved when it satisfies convergence. Perhaps, the solution is converged in the wrong plane, which is very distant from that of acceptable solution plane. We have observed this phenomenon during the error-resilience identification phase. As discussed in Section 2, random errors are injected in the resilience identification analysis while using a relaxed quality function. Two cases are shown in Fig. 5 to illustrate the problem. The first case is randomly skipping of computations (Fig. 5a and 5b), while the second case is an arbitrary error injection in the first element of the Z vector output (Fig. 5c and 5d). For the first case, we can see a comparable response of approximate and exact computing for gains outputs, albeit with no quality improvement beyond a specific number of iterations for the convergence metric. However in the second case, although the approximate solution converges quickly (Fig. 5c), it produces unacceptable gains (Fig. 5d). This shows that the original quality metric of StEFCal (the convergence criterion) is not sufficient in the resilience analysis process. For that matter, we introduced an additional quality parameter: Diff_rel (Eq. 3) that can ensure the convergence of approximate solution within an acceptable distance to that of exact solution in the Euclidean space.

6 CONCLUSIONS

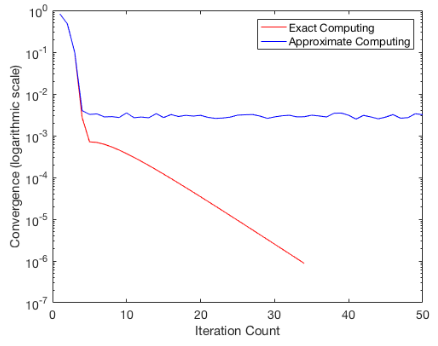
Our proposed Adaptive-SAM has shown improvements in the error resilience analysis of iterative workloads by quantifying the number of resilient iterations (23% of the total iterations in our case study) in addition to high-level analysis parameters. Moreover, we have shown that the quality function must be reconsidered in the error resilience analysis process as the original quality metric of an iterative workload (convergence criterion) might not be necessarily sufficient. In which case, an additional quality metric has to be defined for reliable quality assessment to establish the promising approximate computing strategies.

ACKNOWLEDGMENTS

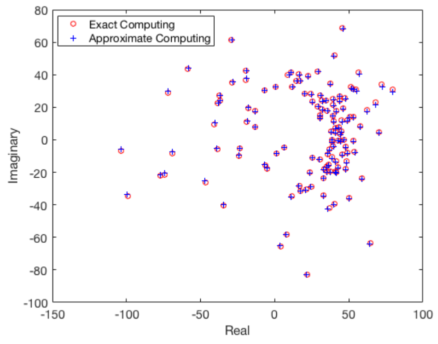
This work was conducted in the context of the ASTRON and IBM joint project, DOME, funded by the Netherlands Organization for Scientific Research (NWO), the Dutch Ministry of EL&I, and the Province of Drenthe. Authors are also thankful to Dr. S. H. Gerez for the valuable discussions.

REFERENCES

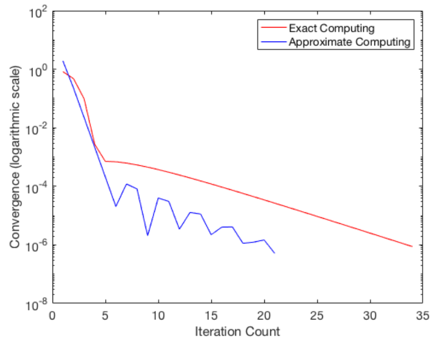
[1] Swarat Chaudhuri, Sumit Gulwani, Roberto Lublinerman, and Sara Navidpour. 2011. Proving programs robust. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM, 102–112.
[2] Vinay K Chippa, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan. 2013. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 113.
[3] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. 2012. Neural acceleration for general-purpose approximate programs. In *Proceedings of the*



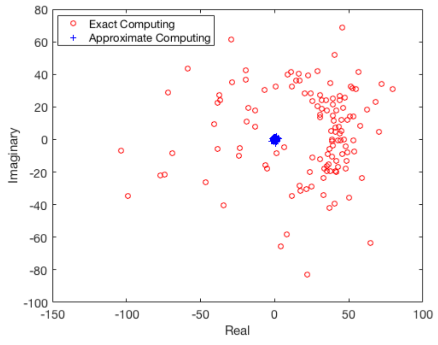
(a) Case 1: Convergence



(b) Case 1: Complex sensor gains (P=124)



(c) Case 2: Convergence



(d) Case 2: Complex sensor gains (P=124)

Figure 5: Error resilience identification; convergence (logarthmic scale) w.r.t the number of iterations (a) and (c); complex sensor gains for P=124 (b) and (d)

2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 449–460.

[4] Qian Hang. 2015. Counting the Floating Point Operations (FLOPS). In *MATLAB Central File Exchange*. No. 50608, Ver. 1.0, Retrieved Oct 7, 2016.

[5] Henry Hoffmann, Sasa Misailovic, Stelios Sidiroglou, Anant Agarwal, and Martin Rinard. 2009. Using code perforation to improve performance, reduce energy consumption, and respond to failures. Massachusetts Inst. Technol. (MIT), Cambridge, MA, USA, Tech. Rep. MIT-CSAIL-TR-2009-042 (2009).

[6] Shubham Kamdar and Neha Kamdar. 2015. big. LITTLE Architecture: Heterogeneous Multicore Processing. *International Journal of Computer Applications* 119, 1 (2015).

[7] Ulya R Karpuzcu, Ismail Akturk, and Nam Sung Kim. 2014. Accordion: Toward soft near-threshold voltage computing. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. IEEE, 72–83.

[8] Philipp Klaus Krause and Ilia Polian. 2011. Adaptive voltage over-scaling for resilient applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 1–6.

[9] Jim kukunas. 2015. *Power and Performance: Software Analysis and Optimization*. Morgan Kaufmann, Waltham, MA 02451, USA.

[10] Jiayuan Meng, Srimat Chakradhar, and Anand Raghunathan. 2009. Best-effort parallel execution framework for recognition and mining applications. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 1–12.

[11] Sasa Misailovic, Stelios Sidiroglou, Henry Hoffmann, and Martin Rinard. 2010. Quality of service profiling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. 25–34.

[12] Asit K Mishra, Rajkishore Barik, and Somnath Paul. 2014. iACT: A software-hardware framework for understanding the scope of approximate computing. In *Workshop on Approximate Computing Across the System Stack (WACAS)*.

[13] Sparsh Mittal. 2016. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 62.

[14] Debabrata Mohapatra, Vinay K Chippa, Anand Raghunathan, and Kaushik Roy. 2011. Design of voltage-scalable meta-functions for approximate computing. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 1–6.

[15] Shahrzad Naghibzadeh, Ahmad Mouri Sardarabadi, and Alle-Jan van der Veen. 2016. Radioastronomical image reconstruction with regularized least squares. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 3316–3320.

[16] Semeen Rehman, Walaa El-Harouni, Muhammad Shafique, Akash Kumar, and Jörg Henkel. 2016. Architectural-space exploration of approximate multipliers. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 80.

[17] Antonio Roldao-Lopes, Amir Shahzad, George A Constantinides, and Eric C Kerrigan. 2009. More flops or more precision? Accuracy parameterizable linear equation solvers for model predictive control. In *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*. IEEE, 209–216.

[18] Pooja Roy, Rajarshi Ray, Chundong Wang, and Weng Fai Wong. 2014. Asac: Automatic sensitivity analysis for approximate computing. In *ACM SIGPLAN Notices*. ACM, 95–104.

[19] Pooja Roy, Jianxing Wang, and Weng Fai Wong. 2015. PAC: program analysis for approximation-aware compilation. In *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. IEEE Press, 69–78.

[20] 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.

[21] Mehrzad Samadi, Davoud Anoushe Jamshidi, Janghaeng Lee, and Scott Mahlke. 2014. Paraprox: Pattern-based approximation for data parallel applications. In *ACM SIGARCH Computer Architecture News*. 35–50.

[22] Mehrzad Samadi, Janghaeng Lee, D Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. 2013. Sage: Self-tuning approximation for graphics engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 13–24.

[23] Adrian Sampson. 2015. *Hardware and software for approximate computing*. Ph.D. Dissertation. University of Washington.

[24] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. 2014. Approximate storage in solid-state memories. *ACM Transactions on Computer Systems (TOCS)* 32, 3 (2014), 9.

[25] Muhammad Shafique, Rehan Hafiz, Semeen Rehman, Walaa El-Harouni, and Jörg Henkel. 2016. Invited: Cross-layer approximate computing: From logic to architectures. In *Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*. IEEE, 1–6.

[26] MP Van Haarlem, MW Wise, AW Gunst, George Heald, JP McKean, JWT Hessels, AG De Bruyn, Ronald Nijboer, John Swinbank, Richard Fallows, and others. 2013. LOFAR: The low-frequency array. *Astronomy & Astrophysics* 556 (2013), A2.

[27] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. 2016. Approximate computing: A survey. *IEEE Design & Test* 33, 1 (2016), 8–22.