ApproxIt: An Approximate Computing Framework for Iterative Methods

Qian Zhang, Feng Yuan, Rong Ye and Qiang Xu
CUhk REliable computing laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {qzhang,fyuan,rye,qxu}@cse.cuhk.edu.hk

ABSTRACT

Approximate computing, being able to tradeoff computation quality (e.g., accuracy) and computational effort (e.g., energy) for error-tolerant applications such as media processing and the emerging Recognition, Mining, and Synthesis (RMS) applications, has gained significant traction in recent years. Many of these applications employ iterative methods for solution-finding, wherein a sequence of improving approximate solutions are generated before reaching the final converged solution. In this work, we propose ApproxIt*, a novel approximate computing framework for iterative methods with quality guarantees. To be specific, we present a lightweight quality estimator that is able to capture the solution quality of each iteration and use it to guide the selection of approximate computing mode in the next iteration. With the proposed dynamic effort scaling technique,* ApproxIt *is able to dramatically improve application energy efficiency under quality guarantees, as demonstrated in our experimental results.*

1. INTRODUCTION

Approximate computing is an emerging design paradigm that is able to tradeoff computation quality (e.g., accuracy) and computational effort (e.g., energy) by exploiting the inherent error resilience of a wide spectrum of applications [1, 2], such as multimedia, digital signal processing, and the broad class of applications referred to as Recognition, Mining, and Synthesis (RMS). For these applications, there is usually no specific "golden" output value that must be computed. Consequently, by relaxing the numerical equivalence between the specification and implementation of such applications, designers could use more energy-efficient approximate hardware building blocks for their applications and thus achieve significant energy savings and/or performance improvement.

Most existing works in approximate computing are static designs that replace fully accurate arithmetic components with approximate units to achieve various benefits with slight quality degradation. As the computation quality requirement of an application can vary significantly at runtime, however, it is preferable to design *quality-configurable systems* (QCSs) that can adapt themselves to the changing application needs on-the-fly [3–5]. The critical components in a QCS include: (i). *quality estimators* that monitor the output quality in each intermediate computation step; (ii). *control mechanisms* that tune approximation modes at runtime so that an optimized system consumes minimum computational efforts with quality guarantees. Chippa *et al.* [3] proposed to embed sensors at various levels for quality estimation and employ a proportional-integral-derivative (PID) controller to regulate computational effort levels based on sensor outputs. While interesting and inspiring, these sensors cannot tell the "true" computation quality at intermediate step, and more importantly, there is no guarantee for the quality of the final result.

It is very difficult, if not impossible, to develop an end-to-end approximate computing framework for all kinds of error-tolerant applications. In this work, we focus on applying approximate computing on *iterative methods* (IMs) [6], for the following reasons: (i). IMs are the most widely-used solutions for large linear/non-linear systems of equations and combinatorial optimization problems, which have a broad application in many fields [7]; (ii). the computation quality requirement in each iteration of IMs varies at runtime, which makes it a perfect candidate for QCS design. The proposed approximate computing framework for iterative methods is namely *ApproxIt* and its main contributions include:

- *ApproxIt* is an end-to-end solution with quality guarantees. To the best of our knowledge, this is the only approximate computing work in the literature that ensures the quality of final outputs at algorithm-level;
- We propose a lightweight quality estimator that is able to effectively capture the computation quality in each iteration of IMs;
- We present a novel optimization framework that dynamically adjust approximation modes to minimize computational efforts under quality guarantees.

The paper is organized as follows. In Section 2, we introduce prior works and motivation. Section 3 presents our framework and theoretical analysis. Online reconfiguration strategies and experimental results are then detailed in Section 4 and 5, respectively. Finally, Section 6 concludes this paper.

2. PRELIMINARIES AND MOTIVATION

2.1 Iterative Methods

In computational mathematics, an *iterative method* produces a sequence of approximate answers that, in the best case, converges to the solution of the problem. Typically, it involves a relatively simple calculation, performed repeatedly.

Thanks to their computational efficiency, IMs are widely used in solving large linear/non-linear systems of equations and combinatorial optimization problems, and hence they are playing a prominent role in a variety of applications (e.g., computational biology, chemistry, finance and aerospace). For example, the iterative-based finite difference and finite element

DAC'14, June 01 - 05 2014, San Francisco, CA, USA

methods give us perfect solutions, both in theory and practice, to tackle partial differential equations.

Consider convex optimization is at the heart of machine learning problems and IMs are generally regarded as the only possible solution, without loss of generality, let us consider such a problem on $f : \mathbb{R}^n \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^n} f(x) \quad . \tag{1}$$

Let a random $x^0 \in \mathbb{R}^n$ be an initial iterate with $\nabla f(x^0) \neq 0$. To achieve progress, we need to proceed in a certain search direction $d^k \in \mathbb{R}^n$. The family of update rules parameterized by the search directions $\{d^k\}_{k \geq 0}$ and step sizes $\{\alpha^k\}_{k \geq 0}$ are:

$$x^{k+1} = x^k + \alpha^k d^k \ for \ k = 0, 1, ... \tag{2}$$

which shows that there are two kinds of computations in the whole procedure of iterative method. One is to get the moving direction, and the other is to update the current position. From this point of view, errors caused by approximate techniques are either from direction calculation (called "direction error") or update calculation (called "update error").

2.2 Related Work

Recent years have witnessed a large and growing number of applications that are inherently error-tolerant. Because these applications do not require "strict" correctness but rather approximate correctness, various approximate computing techniques, spanning from transistor-level designs [8, 9] to high-level programming languages [10], were proposed to exploit this feature to achieve improved performance and/or energy efficiency.

Generally speaking, approximate hardware designs implement a slightly different yet more energy-efficient and/or faster Boolean function, when compared to exact designs. Various approximate designs for specific arithmetic components were presented in the literature (e.g., [11–14]), taking advantage of the structural properties of these components. In addition, there are also a few attempts to systematically evaluate and/or generate approximate circuits [15–17]. The quality of the above approximate hardware designs were usually demonstrated with some specific case studies.

In practice, the application quality requirement can vary significantly at runtime under different datasets and computational steps. The concept of *dynamic effort scaling* (DES) was thus proposed in [3], wherein *scalable effort systems* (SES) are designed to achieve the maximum benefits under varying quality requirements. Due to the criticality of adder design in building up approximate circuits, accuracy-configurable adders are designed in [4, 5] to serve the changing computation requirement.

2.3 Motivation

To the best of our knowledge, [3] is the only work that presented an approximate computing framework for general error-tolerant applications, which tries to regulate scaling mechanisms to operate the hardware at the lowest possible effort while meeting the target quality constraints.

One of the most critical issues in such a DES system design is how to continuously monitor the output quality so that the system can adapt to changes in the degree of resilience. In [3], the authors proposed to conduct *sensor-based quality estimation*. In particular, they proposed to use some of the internal variables of the computation as *algorithm-level sensors* to estimate the output quality. While the concept is interesting and inspiring, because the quality of the final outputs is the real concern, the critical challenge of how to derive an effective quality measure at intermediate computation steps remains an open question. More importantly, such quality measures should be able to guide the reconfiguration of approximation modes so that the target quality constraints are satisfied at the end, which is not guaranteed with the PID control mechanism in [3].

Let us take the application of approximate computing on K-means clustering problem discussed in [3] as an example. K-means clustering is a well-known data mining technique that clusters a set of given data points into K groups. In [3], the authors proposed to use *mean centroid distance* (MCD), i.e., the average distance of a point from its corresponding centroid, as the algorithm-level sensor for quality evaluation, and employ a PID control mechanism to tune approximation modes based on sensor outputs. While intuitive, this design cannot provide a quality-guaranteed end-to-end solution. This is because: (i). such kind of metrics are rather ad-hoc and its effectiveness depends heavily on the datasets; (ii). the control mechanism does not take the error-tolerance capability of the application into consideration. Consequently, the iterative clustering scheme might be unstable, causing convergence difficulty (i.e., large changes in many iterations) or even false clustering results at the end.

From the above, we argue that it is extremely difficult, if not impossible, to have a general approximate computing framework that is applicable for all kinds of error-tolerant applications. This has motivated us to study particular types of algorithms and develop specific quality-guaranteed QCSs for them. Due to the broad application of iterative methods on many fields, we focus on it and propose the so-called *ApproxIt* framework, as detailed in the following sections.

3. THE PROPOSED FRAMEWORK

The proposed approximate computing framework for IMs, *ApproxIt*, is comprised of two stages: the offline characterization stage and the online reconfiguration stage, as demonstrated in Figure. 1. As each application has its unique error-resilient behavior, at offline stage, we try to capture those features that differentiate an application from others. This is achieved by identifying the error-resilient and error-sensitive parts of each application and characterizing the impact of approximate hardware building blocks on it. Then, at online stage, reconfiguration of approximation modes are performed at certain iterations considering the time-varying resilience requirements of the application.

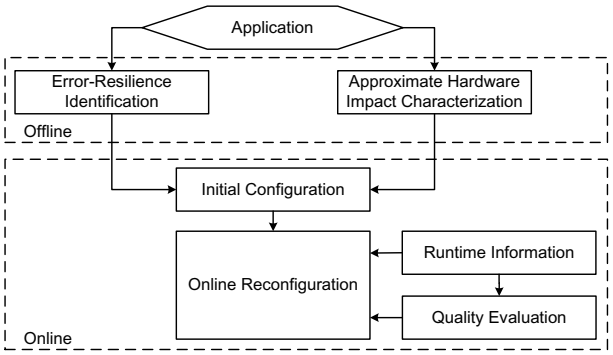


Figure 1: The proposed *ApproxIt* framework.

3.1 Offline Characterization

Even for error-tolerant applications, there exist error-sensitive parts (e.g., control flow) that using inexact computations for them may cause fatal errors or even crash the system. Consequently, a critical step at offline stage is to first extract those error-resilient computations that can be conducted on approximate hardware. In this work, we resort to the analysis and characterization techniques in [2] to achieve this objective.

Various metrics have been proposed in the literature (e.g., [5, 18]) to evaluate the quality degradation of approximate hardware, e.g., worst-case error (WCE), error rate (ER), and mean

error (ME). These low-level evaluation metrics, however, cannot be directly used to characterize the quality degradation at the application-level because of the error masking and/or error accumulation effects. To tackle the above problem, we introduce a new metric *quality error* for IMs to evaluate the impact of approximate hardware at high level:

Definition 1. Denoting the accurate and approximate results of one iteration of an IM application as $f(x)$ and $f'(x)$ respectively, the *quality error* ϵ is defined as the relative difference between $f(x)$ and $f'(x)$, i.e.,

$$\epsilon = \frac{|f(x) - f'(x)|}{f(x)} \quad .$$

Since iterative applications conduct the same computations in each iteration, error measurements at the iteration level can better reflect the computation quality than the above low level metrics. In our work, the quality errors of different approximate hardwares (or different approximation modes of a QCS) are pre-characterized at offline stage by simulating several iterations on representative workloads.

3.2 Online Reconfiguration

Needless to say, effective online reconfiguration is the key to achieve optimized QCS designs. There are two critical issues to resolve: (i). the quality requirement for the final result must be satisfied; (ii). the reconfiguration strategy should be effective in terms of computational effort. In the following, we detail how *ApproxIt* is able to provide quality guarantees for iterative methods. The detailed reconfigurable strategy is then discussed in Section 4.

While it is extremely difficult, if not impossible, to ensure the computation quality with approximate hardware for general applications, fortunately, this is achievable for iterative methods. This is because, it is known that the quality of IMs is guaranteed if the specific iterative algorithms can converge to a local minimum [6, 19]. Therefore, essentially, the problem of guaranteeing application quality for IMs is equivalent to the problem of ensuring the correct convergence of iterative algorithms. By theoretical analysis below, we acquire two useful criterion on *direction error* and *update error*, the two types of errors caused by approximate techniques (see Section 2.1 for details).

According to [20], we have a proposition related to "direction error" as follows:

Proposition 1. If there exists a d in $\mathbb{R}^n$ such that we have $\nabla f(x^k)^T d < 0$, then there exists an $\alpha_0 > 0$ such that $f(x^k + \alpha d) < f(x^k)$ for all $\alpha \in (0, \alpha_0)$.

where α and d are the same with relevant ones in Section 2.1. That means, since IMs (e.g., first-order Gradient Descent, second-order Newton's Method) are to minimize an approximation of function f at the current iterative $x^k \in \mathbb{R}^n$, we need to choose $x \in \mathbb{R}^n$ such that $\nabla f(x^k)^T d^k < 0$, where d^k is the current search direction. In particular, we transform into the following optimization problem:

$$\begin{aligned} &minimize && \nabla f(x^k)^T d^k \\ &subject\ to && \|d\|_2^2 \leq \|\nabla f(x^k)\|_2^2 \quad . \end{aligned} \tag{3}$$

After solving it, we obtain the d^k that satisfies $\nabla f(x^k) d^k < 0$, e.g., the steepest descent $d^k = -\nabla f(x^k)$ for gradient descent and $d^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$ for newton's method. By doing this, the move of IM iteration can be ensured towards the local minimum and hence guarantee the convergence. This is the first criteria to handle with "direction error". Interested readers may refer to [20, 21] for the proof.

Next, we have the second criteria considering the "update error" (denoted by ϵ) introduced by the update function in the following:

$$x^{k+1} = x^k + \alpha d^k + \epsilon^k \ for\ k = 0, 1... \tag{4}$$

where $\{\epsilon^k\}$ is a sequence of errors in $\mathbb{R}^n$. Similarly it is also theoretically provable [19] that the condition $\|\epsilon^k\| \leq \|x^k - x^{k+1}\|$ for scaled data must be satisfied, such that the algorithm would not deviate much from the standard gradient descent and can eventually converge.

4. RECONFIGURATION STRATEGY

This section presents two reconfiguration strategies that can be used in *ApproxIt* to change the operational modes of the approximate hardware at certain iterations.

4.1 Incremental Strategy

The basic idea for our incremental strategy is as follows: considering the fact that IMs generally start with random solutions with low quality requirement in the beginning while the quality requirements gradually increase, we always start with configuring approximate component at the lowest accuracy level. Each reconfiguration then updates the configuration to its adjacent one with higher accuracy level (no other reconfigurations are allowed), until we get to the fully accurate mode.

The convergence nature of iterative methods guarantees that the program will reach certain local minimum as long as the accurate mode is eventually applied. However, the error of approximate component may lead to the situation that searching direction deviates too much from the standard descent direction and even cause extra iterations. For saving the unnecessary energy consumption, we proposed to observe some runtime information and to perform light computation to guide the reconfiguration process.

We define three tightly related schemes, namely gradient, quality, and function scheme, respectively. Based on the criteria related to "direction error", which we have discussed in last section, gradient scheme considers the dot production of moving direction and current gradient is less than zero, to make sure the direction computed by approximate component is correct. The quality scheme takes the second criteria on "update error" into account, as shown in [19]. However, since the offline choice of impact characterization (such as ME) cannot represent all cases, our last function scheme provides a recovery mechanism when error happened.

Suppose we use the i^{th} approximate component with error influence character ϵ_i, the above three schemes are formally defined as:

- Gradient Scheme: Reconfigure whenever
$$\nabla f(x^{k-1})^T (x^k - x^{k-1}) > 0.$$
- Quality Scheme: Reconfigure whenever
$$f(x^k) - f(x^{k-1}) < \|x^k\| \epsilon_i.$$
- Function Scheme: Reconfigure and roll back one iteration whenever
$$f(x^k) > f(x^{k-1}).$$

Gradient and quality are error prevention schemes, and we perform a reconfiguration if the constraint defined by any one of these schemes is violated. The gradient scheme reconfigures whenever the momentum term and the negative gradient are making an obtuse angle. In other words we reconfigure when the momentum seems to be taking us in a bad direction, as measured by the negative gradient at that point. The quality scheme reconfigures whenever estimated error is bigger than the distance(ℓ_2 norm) of two iterations. If there is an error, the recovery based function scheme reconfigures and rolls back one iteration to avoid the extreme cases in which we are moving away from the optimum or falsely stopped (not truly converged) caused by approximation.

Note that, all the above quantities involved are already available along with conducting IMs, and hence the extra computations used in such incremental reconfiguration strategy is negligible.

4.2 Adaptive Angle-Based Strategy

The above iterative reconfiguration strategy is simple, but there is only one reconfigurable direction, i.e., from low-accuracy level to high-accuracy one. In the following, we present a more flexible adaptive angle-based reconfigurable scheme.

Generally speaking, gradually increasing the accuracy level at runtime works well when the parameter manifold is convex. However, the manifold can be much more complex, which means the error-tolerance capability of the application is not gradually decreasing, as shown in Figure. 2. To tackle such

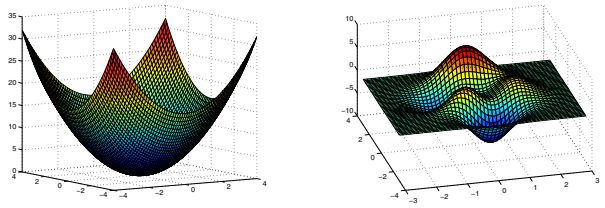


Figure 2: Parameter Manifold in 3D Space

kind of problem with maximum benefits, it is better to let the online reconfiguration strategy select the most suitable approximate components adaptively instead of simply changing it to the adjacent one with higher accuracy level.

We define the hyperplane that is perpendicular to the objective direction (i.e. z direction in Figure. 2) as our base plane, and denote the angle between the current point's tangent plane of parameter manifold and this base plane as α. When α gets bigger, the manifold will gets steeper. In the case that the angle is big, as long as the moving direction is correct, the point has more freedom to go to somewhere else below than itself. In another word, algorithm is more tolerant to approximation error when α get larger, thus we can use approximate mode with lower accuracy levels to get better performance and power saving. While when the angle is getting to zero, the computing is much likely to be converged and becoming more sensitive to errors, so components with higher accuracy levels are preferred.

It is easy to see that $\alpha \in [0, 90^\circ]$. The key issue in our adaptive angle based strategy is to form and keep a lookup table which contains the best α ranges for approximate components with different accuracy level, so that we can select appropriate component based on α in each iteration to minimize energy consumption. In the remaining part of this subsection, we first formulate this as an optimization problem, and then propose online lookup table updating method to make our reconfiguration strategy applicable to different dataset by learning runtime information.

4.2.1 Offline Lookup Table Initialization

We use $\Omega = \{\omega_0, \omega_1...\}$ and $J = \{j_0, j_1...\}$ to denote the angle percentage and energy consumption of different approximate components, respectively.

Our objective function is defined as minimizing energy consumption in one iteration with respect to that errors are under control, as follows.

$$\begin{aligned} min \quad & \Omega^T J \\ s.t. \quad & \sum_{i=0}^{n=4} \omega_i = 1,\ \omega_i > 0 \\ & \Omega^T \epsilon \leq E \end{aligned} \tag{5}$$

where E is the error which can be tolerated in this iteration. In the initialization step, $E = f(x^1) - f(x^0)$, and both of $f(x^1)$ and $f(x^0)$ can be obtained from the offline impact characterization procedure. Fortunately, this optimization problem can be effectively solved by introducing Lagrange multipliers, and the initialized lookup table is put into the framework to guide the selection of approximation modes.

4.2.2 Online F-Step Fixed Update

At runtime, we take f as the update parameter, which means the lookup table will be updated after every f steps. In this way, we give programmers more freedom to tradeoff computation quality and computational effort.

Suppose we are currently at $k = f$ iteration, Ω will be updated following Equation.6 with respect to $E = f(x^k) - f(x^{k-1})$. When $f = 1$, the framework always keeps the latest state of lookup table in every single iteration, and this means in every step, the strategy greedily select the most suitable component to maximize power saving with quality guarantees.

The complexity of updating this lookup table is decided by the number of approximation modes that are used. In our current implementation, this state space is only 5*1, which can be safely ignored when compared to the application complexity. In other words, the proposed adaptive angle-based reconfiguration strategy is lightweight even though the lookup table is updated at runtime.

5. EXPERIMENTAL RESULTS

In this section, we present experimental results of *ApproxIt* on various iterative methods with different datasets.

5.1 Experimental Setup

In our experimental results, the output generated with fully accurate component is denoted as the $Truth$. As our candidate approximate components, we implement four kinds of approximate adders (as illustrated in [5]), and denote the accuracy level of these approximate components by $Level = \{level1, ..., level4\}$, where the larger level index means higher the accuracy level. It should be noticed that our proposed framework is also applicable to other approximate component designs.

Based on above hardware platform, we perform experiments on a benchmark suite consisting 2 representative iterative methods in different application fields, and Table. 1 describes their detailed information. For the sake of fairness, each algorithm uses the same initialization for different datasets, and the quality of generated approximate results are evaluated by comparing with above true results. The quality evaluation metric (QEM) is dependent on application. The datasets and parameter settings for each application are illustrated in Table. 2.

5.2 Single Mode Results

Our first experiment is conducted for single mode configuration, that is, only one approximate component is applied through the entire algorithm lifetime. The numerical results for GMM are presented in Table. 3(a), where column "Configuration" indicates the applied approximation component, column QEM presents data of quality evaluation metric, and column "Iteration" and "Energy"(energy model in [22]) give the number of iterations and the normalized energy consumption on total approximate parts, respectively. The graphical clustering results on dataset "3cluster" are also presented in Figure. 3 as the visualized quality evaluation.

First of all, decreasing the accuracy level can bring benefit on energy saving. However, this can be the other way around if the error caused by approximation becomes out of control. As illustrated in the first row of rightmost column in Table.3(a), accuracy level 1 will give an output using 4 times

Table 1: Benchmark Suite Description

Benchmark	Representative Fields	Quality Evaluation Metric
Gaussian Mixture Models	Nonlinear Clustering and Classification, Convex Optimization	Hamming Distance
AutoRegression	Time Series, Regression Problems	Least Square Error with ℓ_2 Norm

Table 2: Dataset and Parameter Description

Dataset	Application	Samples	Source	MAX_ITER	Convergence	Adder Impact
3cluster	Gaussian Mixture Model	1000*2	Matlab	500	1e-10	Mean Value
3d3cluster	Gaussian Mixture Model	1900*3	Matlab	500	1e-6	Mean Value
4cluster	Gaussian Mixture Model	2350*2	Matlab	500	1e-6	Mean Value
HangSeng INDEX	AutoRegression	6694*10	Yahoo!	1000	1e-13	80% Confidence Space
NASDAQ Composite	AutoRegression	10799*10	Yahoo!	1000	1e-13	80% Confidence Space
S&P 500	AutoRegression	16080*10	Yahoo!	1000	1e-13	80% Confidence Space

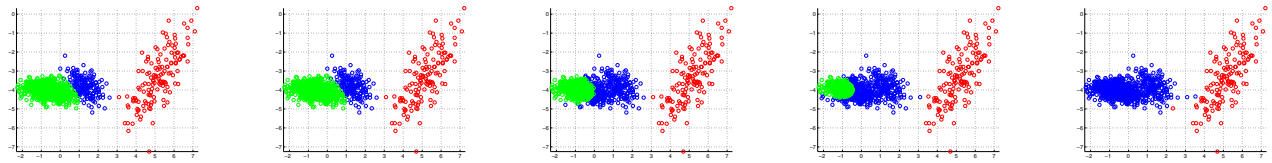
Table 3: Results on Gaussian Mixture Models

(a) Single Mode Results

Configurations	3cluster			3d3cluster			4cluster		
	Iteration	QEM	Energy	Iteration	QEM	Energy	Iteration	QEM	Energy
level1	4	695	0.0513	3	1796	0.0526	MAX_ITER	1309	4.43
level2	59	186	0.658	37	951	0.474	25	470	0.238
level3	63	137	0.718	45	764	0.579	47	245	0.476
level4	67	0	0.769	53	476	0.737	77	199	0.762
Truth	81	0	1	68	0	1	93	0	1

(b) Online Reconfiguration Results

Dataset	Incremental Reconfiguration							Adaptive Reconfiguration (f=1)						
	Steps on Single Components					Total	Error	Steps on Single Components					Total	Error
	level1	level2	level3	level4	acc			level1	level2	level3	level4	acc		
3cluster	3	8	12	30	0	53	0	4	13	19	15	0	51	0
3d3cluster	2	2	10	23	17	54	0	7	10	16	15	6	54	0
4cluster	1	17	10	28	2	58	0	7	23	12	13	3	58	0



(a) Accurate (b) Level4 (c) Level3 (d) Level2 (e) Level1

Figure 3: GMM Result: Single Mode Approximation on 3cluster

Table 4: Results on AutoRegression

(a) Single Mode Results

Configurations	HangSeng INDEX			NASDAQ Composite			S&P 500		
	Iteration	QEM	Power	Iteration	QEM	Power	Iteration	QEM	Power
level1	4	1.8842	0.461	244	2.5912	0.51	644	2.1246	0.65
level2	MAX_ITER	1.5716	126	263	2.1609	0.607	698	1.2306	0.772
level3	5	0.8229	0.647	271	2.0032	0.639	717	1.1093	0.813
level4	5	0.6772	0.667	349	0.6923	0.846	798	0.5601	0.928
Truth	7	0	1	387	0	1	802	0	1

(b) Online Reconfiguration Results

Dataset	Incremental Reconfiguration							Adaptive Reconfiguration (f=1)						
	Steps on Single Components					Total	Error	Steps on Single Components					Total	Error
	level1	level2	level3	level4	acc			level1	level2	level3	level4	acc		
HangSeng INDEX	1	1	1	2	2	7	0.0017	0	1	2	2	1	6	0.0017
NASDAQ Composite	17	44	35	99	187	382	0.0402	31	58	79	132	38	338	0.0099
S&P 500	23	63	110	276	327	799	0.0021	49	92	191	298	109	739	0.0011

energy consumption than the accurate mode. Second, over approximation may induce unacceptable quality degradation or even fail algorithm. As shown in these contents, when the dataset is *3cluster*, which means the GMM is applied to group these 2D data into 3 different clusters, the algorithm computed by accurate design finally converged with a clustering result after 81 iterations; while, the approximate design at accuracy level 1 falsely stops the algorithm after only 4 iterations, offering a completely wrong result that has only 2 clusters as shown in Figure. 3(e). And when we use dataset *4cluster* on the same algorithm, the application even cannot be converged within the maximum iteration using level1 approximate components. In addition, we also notice that the dataset variation has a significant impact on the algorithm quality. For example, accuracy level 4 is able to generate exactly the same result with *Truth*, while it will put 476 data samples to wrong clusters on *3d3cluster* and 199 on *4cluster*, respectively. From the above observation, using approximate components directly to applications cannot guarantee output correctness and output quality. To solve this problem, it is

preferable to observe runtime information and to conduct reconfiguration dynamically. Similar results can be also found in Table. 4.

5.3 Online Reconfiguration Results

Our second experiment is performed to evaluate the effectiveness of our proposed online configuration strategies in terms of output quality and energy efficiency. Table.3(b) lists the reconfiguration statistics and quality evaluation on Gaussian Mixture Model for incremental and adaptive online reconfiguration strategies, respectively. Result in "level" and "acc" columns indicate the iteration numbers used in computing mode with the corresponding accuracy level, and "acc" represents the fully accurate mode.

Compared to the single mode results, approximate computing with both incremental and adaptive online reconfiguration can provide zero-error (hamming distance) result by capturing application's runtime details. Similar results can be also found in Table 4.

Figure.4 show GMM's total energy consumption on approximate part and energy consumption per iteration on approximate part till the application get converged. By using incremental reconfiguration, we can get 52.4%, 25.0% and 33.6% energy benefit on these three datasets, while adaptive strategy brings us 63.8%, 28.4% and 44.0%, respectively. We can also observe the energy benefit on other benchmarks in Table 4.

The angle-based reconfiguration strategy is better than the incremental one, and the amount of energy savings depend on the respective case (see Figure.4). This is because, the former one use an optimization-based strategy to automatically provide the system a proper guidance to select the most suitable accuracy level, so the runtime information will have more impact on decision making, while the incremental method can be easily influenced by offline adder characterization.

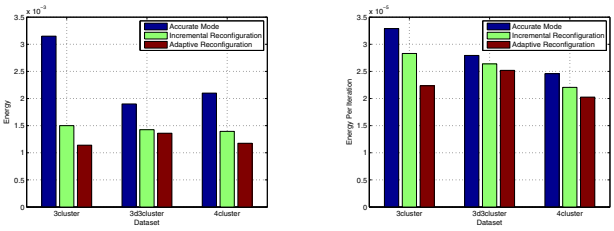


Figure 4: GMM:Comparison on Energy Consumption

6. CONCLUSIONS

Approximate computing is a promising solution to tradeoff computation quality and computational effort. Most existing works are static designs that replace accurate arithmetic components with approximate units to achieve various benefits with slight quality degradation. In this work, we propose *ApproxIt*, a novel approximate computing framework for iterative methods with quality guarantees. To be specific, we present a lightweight quality estimator that is able to capture the solution quality of each iteration and use it to guide the selection of approximation mode in the next iteration. The efficacy is demonstrated with experimental results on various applications. To the best of our knowledge, this is the only approximate computing work in the literature that ensures the quality of final outputs at algorithm-level.

7. ACKNOWLEDGEMENT

This work was supported in part by the Hong Kong SAR Research Grants Council under General Research Fund No. CUHK418111 and No. CUHK418112.

8. REFERENCES

[1] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Proc. IEEE European Test Symposium(ETS)*, 2013.

[2] V. K. Chippa, S. T Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *Proc. IEEE/ACM Annual Design Automation Conference(DAC)*, no.113, 2013.

[3] V. K. Chippa, K. Roy, S. T. Chakradhar, and A. Raghunathan. Managing the Quality vs. Efficiency Trade-off Using Dynamic Effort Scaling In *ACM Transactions on Embedded Computing Systems(TECS)*, 12(2s), No.90, 2013.

[4] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proc. IEEE/ACM Design Automation Conference(DAC)*, pp.820-825, 2012.

[5] R. Ye, T. Wang, F. Yuan, R. Kumar and Q. Xu. On Reconfiguration-Oriented Approximate Adder Design and Its Application. In *Proc. IEEE/ACM International Conference on Computer-Aided Design(ICCAD)*, pp.48-54, 2013.

[6] C.T. Kelley. *Iterative methods for optimization*, volume 18. SIAM, 1999.

[7] C. L. Byrne In *Applied Iterative Methods*. A K Peters/CRC Press, 2007.

[8] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. Low-power digital signal processing using approximate adders. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 32, number 1, pp.124-137, 2013.

[9] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, H. Yang. Memristor-based approximated computation. In *IEEE International Symposium on Low Power Electronics and Design(ISLPED)*, pp.242-247, 2013

[10] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. Enerj: Approximate data types for safe and general low-power computation. In *ACM SIGPLAN Notices*, volume 46, pp.164-174, 2011.

[11] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy. Impact: imprecise adders for low-power approximate computing. In *2011 International Symposium on Low Power Electronics and Design(ISLPED)* , pp.409-414, 2011.

[12] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *Proc. of the International Conference on Computer-Aided Design(ICCAD)*, pp.728-735, 2012.

[13] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *24th International Conference on VLSI Design(VLSI Design)*, pp.346-351, 2011.

[14] N. Zhu, W. L. Goh, W. J. Zhang, K. S. Yeo, and Z. H. Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. In *IEEE Transactions on Very Large Scale Integration Systems*, volume 18, number 8, pp.1225-1229, 2010.

[15] R. Venkatesan, A. Agarwal, K. Roy, A. Raghunathan, MACACO: modeling and analysis of circuits for approximate computing. In *Proc. IEEE/ACM International Conference on Computer-Aided Design(ICCAD)*, pp.667-673, 2011.

[16] M. Dehbashi, G. Fey, K. Roy, A. Raghunathan, On Modeling and Evaluation of Logic Circuits Under Timing Variations. In *Proc. Euromicro Conference on Digital System Design*, pp.431-436, 2012.

[17] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, A. Raghunathan, SALSA: systematic logic synthesis of approximate circuits. In *Proc. IEEE/ACM Design Automation Conference(DAC)*, pp.796-801, 2012.

[18] J. H. Liang, J. Han, and F. Lombardi. New metrics for the reliability of approximate and probabilistic adders. In *IEEE Transactions on Computers*, 12:0946, 2011.

[19] Z. Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. In *Annals of Operations Research*, volume 46, number 1, pp.157-178, Springer, 1993.

[20] S. P. Boyd and L. Vandenberghe. In *Convex optimization*. Cambridge university press, 2004.

[21] D. P. Bertsekas, In *Nonlinear programming*. In Athena Scientific, 1999.

[22] N. Weste, D. Harris. In *CMOS VLSI Design: A Circuits and Systems Perspective*. 4th edition, Addison Wesley, March 2010.