MIPAC: Dynamic Input-Aware Accuracy Control for Dynamic Auto-Tuning of Iterative Approximate Computing

Taylor Kemp*
tkemp@wisc.edu
University of Wisconsin–Madison

Yao Yao
yyao69@wisc.edu
University of Wisconsin–Madison

Younghyun Kim
younghyun.kim@wisc.edu
University of Wisconsin–Madison

ABSTRACT

For many applications that exhibit strong error resilience, such as machine learning and signal processing, energy efficiency and performance can be dramatically improved by allowing for slight errors in intermediate computations. Iterative methods (IMs), wherein the solution is improved over multiple executions of an approximation algorithm, allow for energy-quality trade-off at run-time by adjusting the number of iterations (NOI). However, in prior IM circuits, NOI adjustment has been made based on a pre-characterized NOI-quality mapping, which is input-agnostic thus results in an undesirable large variation in output quality. In this paper, we propose a novel design framework that incorporates a lightweight quality controller that makes input-dependent predictions on the output quality and determines the optimal NOI at run-time. The proposed quality controller is composed of accurate yet low-overhead NOI predictors, generated by a novel logic reduction technique. We evaluate the proposed design framework on several IM circuits and demonstrate significant improvements in energy-quality performance.

KEYWORDS

Approximate computing, Quality control, Logic synthesis

ACM Reference Format:
Taylor Kemp, Yao Yao, and Younghyun Kim. 2021. MIPAC: Dynamic Input-Aware Accuracy Control for Dynamic Auto-Tuning of Iterative Approximate Computing. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3394885.3431551

1 INTRODUCTION

Emerging compute-heavy applications, such as deep learning, data mining, and multimedia processing, are highly resilient to errors and produce similar (often, the same) results even if the intermediate computations are not 100% exact. For such applications, *approximate computing* exploits this error resilience to greatly improve performance and energy efficiency by allowing for small errors [8, 14]. *Iterative methods (IMs)* are a common approximation technique that is widely adopted in many applications for large linear/non-linear systems of equations and combinatorial optimization problems [18]. IMs improve approximate solutions from an initial guess over multiple iterations at a much lower computational effort as compared to obtaining the exact solution, thus the accuracy of the approximation is dependent on the number of iterations (NOI): the more iterations, the more accurate the approximate result.

Including iterative logarithmic multipliers [1, 2] and Taylor approximation-based arithmetic logic units (ALUs) [3, 6, 15, 17], a variety of IM circuits have been proposed recently. While these IM circuits enable dynamic energy-accuracy trade-off, they also pose the new challenge of *determining optimal NOI*. In conventional IM circuit design, NOI is determined based on the average- or worst-case energy-accuracy trade-off characterized at design-time by offline analysis or profiling. The accuracy of the approximation, however, is not constant and largely input-dependent [13]. As a result, the offline characterization-based NOI control is inevitably input-agnostic, resulting in a wide variation of approximation error (even if the target average error is satisfied). If a worst-case error must be met, the wide-varying error must be over-compensated by extra iterations, thus more energy consumption.

To address this challenge, we propose *dynamic auto-tuning* of IM circuits using a novel *lightweight quality predictor*. The quality prediction aims to maintain computation quality by determining the level of approximation from the input, without comparing approximate results to exact results, which incurs extra performance and energy overheads. Conventional quality prediction methods are mainly detect-and-correct, where if an approximate module generates a large error, the exact module counterpart corrects the error by re-executing the corresponding portion of the code. They are generally designed to work at the algorithm or application level, which can afford multiple cycles for quality prediction and extra overhead for re-execution. However, for low-power IM circuits, which complete within a few cycles, such high-level quality prediction is prohibitively time- and energy-consuming. For such circuits, predictors should make actionable predictions to determine whether the approximation should terminate every cycle they are run.

Designing lightweight quality predictors for IM circuits offers several challenges and objectives. First, the prediction should start at the same time with the target IM circuit and must be completed before its earliest possible termination, which can be as short as a single cycle. Second, the predictor should generate an actionable output, i.e., optimal NOI, that can be directly used to control the target IM circuit, rather than just an estimated accuracy that incurs extra time and power overheads to control the IM circuit. Finally, energy overhead for accuracy prediction should be minimal to not offset energy savings. These requirements, unfortunately, render

*Currently at Facebook (taylorkemp@fb.com).

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431551

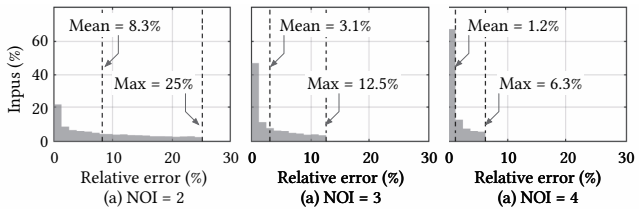

Figure 1: Error distribution of Taylor approximation-based division showing wide variation in output quality depending on the input.

conventional high-level predictors requiring multiple iterations of complex operations (e.g., multiply-and-add) unsuitable for the quality control of IM circuits.

In this paper, we aim to address these challenges to provide a general auto-tuning quality controller for iterative approximate computing. The following is a summary of the novel contributions of this work:

- We propose the design of a lightweight quality controller which determines the optimal NOI for a given input to achieve optimal energy-accuracy trade-off providing fine-grained dynamic accuracy configuration.
- We present an efficient design flow including a novel approximate logic minimization technique to generate accuracy quality predictors under strict power constraint.
- We apply the proposed method to multiple IM circuits and demonstrate improvement in energy efficiency and accuracy distribution in comparison to decision tree (DT)-based quality control.

2 BACKGROUND

In this section, we introduce prior work on approximate computing methods and quality prediction for approximate computing.

2.1 Approximate Circuits

Various approximate circuits have been introduced to improve the performance and energy efficiency of emerging error-resilient applications [9, 12]. Among others, approximate ALUs have received much attention since they are the key building blocks for implementing energy-efficient computing systems. Early approximate ALUs, such as [4, 5], aim to reduce power consumption by simplifying hardware at the circuit or logic level. In these approximate circuits, however, the level of approximation is fixed by hardware design at design-time, which makes run-time energy-quality reconfiguration impossible. Because the actual accuracy is not constant but heavily input-dependent, and also because the accuracy level required by the application varies over time, the fixed-configuration approximate circuits often fail to deliver the optimal energy-quality trade-off the application needs. Recent approximate circuits attempt to address this limitation by adopting IMs [2, 3, 6, 17]. These circuits adjust the level of approximation by adapting NOI at run-time based on off-line quality profiling. Even these solutions, however, do not consider the underlying issue that quality is input-dependent and rely solely on off-line characterization between NOI and average- or worst-case error.

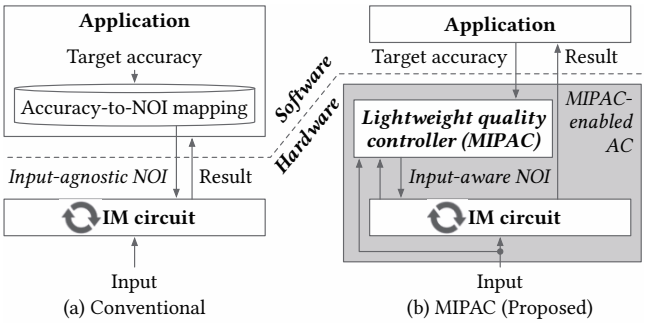

Figure 2: Comparison of NOI determination in conventional and proposed IM-based approximations.

2.2 Run-time Quality Prediction

Quality prediction is crucial for an approximate computing system to meet an energy-quality target in the presence of input-dependent quality variation. Rumba [7] is one of the first attempts at predicting and controlling the quality by detecting quality violations and correcting them by re-computing using the exact counterpart. They consider application-specific approximate computing, such as k-means clustering, JPEG, etc. and use high-level predictors such as linear models and DTs. Other approaches focus on improving prediction accuracy through the implementation of neural network (NN) and ensemble models, which are computationally heavier [10, 16, 18]. These quality predictors are computationally complex and take multiple cycles to generate prediction output. They are suitable to the algorithm- and application-level approximate computing where the time and energy overheads of quality prediction are relatively negligible. However, the overheads are prohibitively high for the quality prediction of IM circuits that should be completed within one to a few cycles.

3 INPUT DEPENDENCY OF APPROXIMATION ERROR

We first present an example of the wide variation of IM output quality to motivate the need for proper quality prediction and control. Figure 1 shows the distribution of relative error for an approximate divider based on iterative Taylor approximation [3] for NOI of 2, 3, and 4. While it is evident that the overall accuracy improves as NOI increases, significant accuracy variation is observed for all NOIs, and many inputs are not accurately characterized by the mean error. The mean relative error (MRE) when NOI=2 is 8.3%, and the worse-case error (WCE) is as high as 25%. When NOI increases to 4, the MRE achieves 1.2%, and the WCE reduces to 6.3%. Both the MRE and WCE drop gradually as NOI increases, but a huge gap consistently exists between the MRE and WCE, which is indicative of the wide variation in the output quality. Therefore, if NOI is set to a fixed number irrespective of the input, the application will suffer from a large variation in output quality.

4 PROPOSED APPROACH

As a solution to the above-mentioned challenge, we present *Minimum Iteration Predictive Accuracy Control (MIPAC)*, a lightweight quality controller design for low-power IM circuits and a design

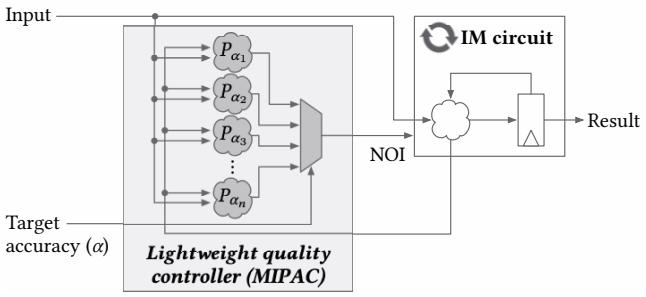

Figure 3: Hardware architecture of MIPAC-enabled IM circuit including the proposed lightweight quality controller.

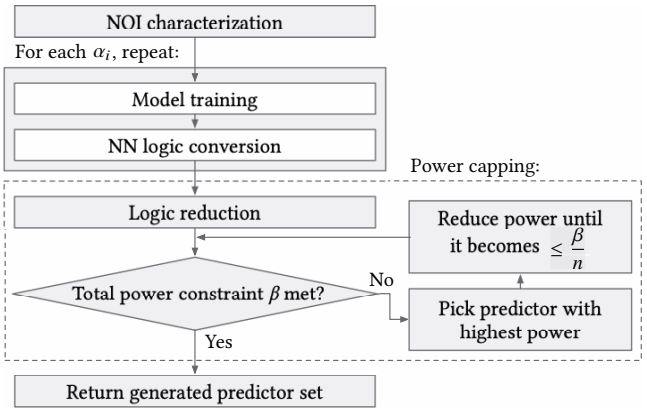

Figure 4: Overall design flow for optimizing MIPAC under accuracy requirements and power constraint.

framework for generating the controller. MIPAC addresses the issue of input-dependent error magnitude by moving the functional mapping of accuracy to NOI closer to the IM circuit. In conventional systems (Figure 2(a)), the application (to be more precise, the compiler) is responsible for determining the NOI for each IM circuit operation that achieves the desired accuracy. Because the actual input values are not available at compile-time, it determines the NOI solely based on the pre-characterized accuracy-to-NOI mappings of the IM circuit. However, the accuracy in the pre-characterized mappings is usually the average- or worst-case error that is not input-aware, and thus it results in a significant output quality variation as we discussed in Section 3. On the other hand, in the MIPAC-based system (Figure 2(b)), the accuracy-to-NOI mapping is performed by the IM circuit itself with the aid of the proposed lightweight quality controller. Because the inputs are directly visible at runtime, the optimal NOI can be derived from the input and/or some input-dependent signals from the IM circuit. Since the quality controller is more tightly integrated as hardware with the IM circuit, it provides the benefits of i) improved stability of the output quality due to input-aware accuracy-to-NOI mapping, and ii) decoupling of the application and compiler from a specific IM circuit, without any hardware modification to the circuit.

As discussed in Section 1, the quality prediction should be made within a single cycle. This is to prevent the IM circuit, which may produce an acceptable result after the first iteration, from running more than needed. Therefore, we generate each predictor as simple combinational logic, which, unlike prior quality predictors that return the outcome after many cycles of computation, completes within the first cycle of execution. Figure 3 shows the hardware architecture of MIPAC integrated with an IM circuit. The quality controller incorporates *n quality predictors*, which of each determines the minimum NOI that achieves a specific accuracy. Based on the target accuracy α, the output of the quality predictor that corresponds to the target accuracy is selected. For example, P_{α_1} is the predictor that produces the minimum NOI for a given input that achieves the accuracy α_1. The prediction is made based on the input and internal signals from the target IM circuit.

We adopt combinational logic-based neural networks (NNs) for solving this highly nonlinear yet error-tolerable quality prediction problem. The quality predictor is implemented by training a NN, generating a look-up table (LUT) from the NN, and realizing the LUT as combinational logic after applying approximate logic minimization. The combinational logic-based NN implementation offers the power of nonlinear decision making and the complexity that NNs offer, without the need for ALUs and other complex units commonly associated with NN. Our method allows us to learn the weighted importance of features and remove circuit signals that have a low correlation with predicted NOI, hence, vastly reducing logic complexity. We describe the design and implementation of MIPAC in more detail in the next section.

5 DESIGN AND IMPLEMENTATION

In this section, we first describe the overview of MIPAC, followed by the design of its key component, the quality predictor. Finally, we present a novel logic minimization scheme to efficiently realize the quality predictor.

5.1 Design Flow

The design of MIPAC requires a joint optimization over multiple design parameters, such as the number of quality predictors to be included in the controller and the accuracy-power trade-off of each predictor among others. Due to the large design space, a systematic design methodology is needed to efficiently derive an accurate yet low-overhead controller. The predictors for accurate quality control incur extra power overhead, and the more accurate, the more power consuming the predictors. Therefore, we present a design framework to maximize the prediction accuracy of MIPAC under minimal power overhead constraints.

NOI characterization. Figure 4 illustrates the offline process to design the MIPAC controller. A set of MREs, $\alpha_1, \alpha_2, \ldots, \alpha_n$, that the controller needs to support and a power constraint β that the controller can afford are specified by the designer. In order to satisfy the MRE targets α_i, we first determine how long to run each input to achieve them. We do this by annotating the minimum NOI needed to satisfy a variable WCE γ_i and pick γ_i such that the MRE becomes α_i. Once we have determined a suitable γ_i for each target MRE, we generate a set of annotated samples for each predictor to train on, using individual signals as input features.

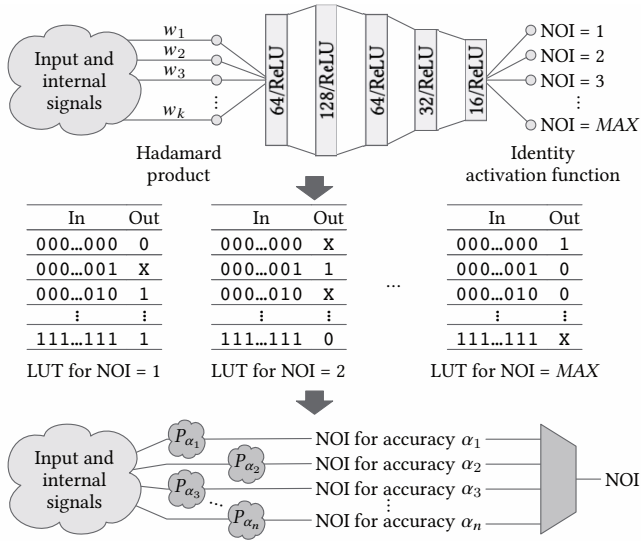

Figure 5: NN model is converted into a set of LUTs for each NOI setting. These LUTs are then further reduced into combinational logic.

Model training. Next, an NN is trained for each γ_i. The NN is trained using squared hinge loss under Adam Optimizer with a learning rate of 4e-4. The batch size was set to 50 and the model was trained for 50 epochs. After model training, we convert the NN into a set of LUTs, one for each NOI setting. Each LUT determines a mapping from the selected IM circuit signals (NN input) to a particular NOI setting. If for a given key the LUT output is 1, this indicates the IM should run for the NOI corresponding to that LUT. In the event that multiple LUTs have an output of 1 for a given input, the NOI will default to the highest one. If none of the LUTs have a corresponding output of one, the predictor will default to the average NOI, rounded up, determined when generating the training data. In order to determine the output mapping, the majority vote of all inputs that map to a given key is used. Figure 5 illustrates the process.

NN logic conversion. The number of possible keys is very large for even a simple IM circuit. Therefore, we reduce the number of possible input keys in the LUTs by merging keys that are likely to map to similar outputs. During training, features that have higher importance when they go high will have larger corresponding weights in the Hadamard product. If the weight is small, pruning out that particular input will result in negligible changes to the overall model output. We pick and prune the n inputs with smallest associated importance iteratively increasing the n until the accuracy is maximized. As there are much fewer samples than possible key combinations, the output is undefined (don't-cares) for the majority of the key combinations, and low pruning levels will result in a high miss rate for the LUT. As we prune out unimportant features, however, we will observe fewer misses and as a result an improvement in the LUTs prediction accuracy, leading to overall accuracy improvements. We continue to prune out features until the maximum accuracy is obtained. Further pruning will begin to decrease accuracy as LUT inputs begin to be mislabeled.

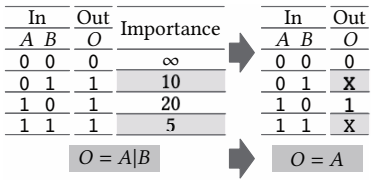

Figure 6: Example approximate logic minimization problem.

Power capping. After all predictors are converted to hardware LUTs, the controller is synthesized and the power consumption is checked against the constraint β. If the constraint is violated, beginning with the predictor with the highest power consumption, approximate onset logic minimization is performed to reduce the power to below the average. The importance of correctly specifying each input is determined based on the average improvement in error by increasing the NOI prediction to the average NOI. In this way, input keys in the LUT that will have larger error improvements should be set to zero first during approximate logic minimization. If fully pruning all bits is required to meet the β, the predictor is replaced by the fastest fixed NOI predictor satisfying the average error constraint α_i. Approximate onset logic reduction is explained further in Section 5.2.

5.2 Power Capping through Approximate Onset Logic Minimization

In this section, we further discuss power capping to ensure that the power overhead of the quality controller meets the power constraint β. In order to reduce power consumption beyond what can be attainable by conventional logic minimization, we leverage that the logic minimization of quality predictors does not have to exactly conserve the functional mapping since the consequence of the errors is the occasional misprediction of quality, thus non-optimal NOI. Therefore, by a novel approximate onset logic minimization algorithm, we trade quality prediction accuracy for power reduction.

Consider a partially specified LUT in which each input-output mapping has corresponding importance. The goal is to produce a minimum sum-of-products (SOP) of the onset where we can tolerate some ϵ threshold of error over the error-weighted onset. Error is incurred by setting the output for a corresponding input in the initial LUT from 1 to 0. Inputs with larger importance should be set to 0 with lower priority than those with less. Additionally, we do not consider the don't-care set as unspecified inputs are assumed to map to 0. This is because LUT misses in the predictor may correspond to other NOI predictions. At the algorithm level, we specify some overall tolerance ϵ which we may violate which determines the aggressiveness of approximation. Setting an output from logic one to zero introduces a corresponding error annotated during LUT construction.

For example, Figure 6 maps a set of 2-bit input signals A and B to a single bit output. We determine a corresponding importance for correctly specifying each row in the table and want to produce a minimum SOP for the onset. If we are unable to incorrectly specify any rows the solution will be $A|B$. However, if we are allowed to

Algorithm 1 Generate Onset Primes

```
procedure GenerateOnsetPrimes(onset)
    mergings = {}
    primes = {}
    for (y, dc') ∈ onset s.t. dist(y ∘ dc', x ∘ dc) = 1 do
        add (x, dc ∘ ¬(x ⊕ y)) to mergings
    end for
    if did not add x to a merging then
        add (x, dc) to primes
    end if
    return GenerateOnsetPrimes(mergings) ∪ primes
end procedure
```

violate some of the outputs in the onset, we can find a simpler solution. For example, if we can violate at most 15 total importance, the most we can violate is rows two ($AB = 01$) and four ($AB = 11$). In this case, violating only row two provides the simplest approximate solution of A. It is worth mentioning here that in general optimal solutions will consist solely of prime cubes.

The proposed approximate logic minimization for quality predictors can be formulated as a simple integer linear problem (ILP). For the ILP construction, we consider an e-implicant to be any input x_i mapping to logic 1 with associated error e. A cube is defined to be a pair $x, dc \in \{0,1\}^n$ s.t. $\{y : y \circ dc = x \circ dc, f(y) = 0\}$ is empty. We define a prime p to be a cube that is maximal, i.e., there is no cube that contains p as a strict subset.

In order to generate the set of all primes based on the onset we consider Algorithm 1 which performs repeated distance one merges within the onset.

The ILP formulation for finding a minimum SOP for ϵ-approximate logic minimization is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_i c_i p_i \\ \text{subject to} \quad & \sum_{i:x_j \in cube(p_i)} p_i \geq 1 - y_j, \forall j \\ & \sum_j e_j y_j \leq \epsilon \\ & p_i, y_j \in \{0,1\} \end{aligned}$$

where p_i is 1 if we include $cube(p_i)$, the i^{th} prime cube for the onset, and 0 otherwise; c_i is the associated hardware cost for adding the i^{th} prime cube and is larger for smaller cubes; x_j is the j^{th} implicant in our LUT and e_j is its corresponding error, the inverse of the average error improvement, larger improvements have a smaller e_j and should be set to 0 first; and y_j is one if the j^{th} implicant is set to zero and zero if it is correctly labeled in the reduced LUT. In this way, we may leave at most ϵ worth of implicants uncovered when finding a minimum SOP within the onset. We note that the exact solution to this ILP is optimal. Furthermore, if ϵ is zero, this will give the solution for exact onset logic minimization.

6 EXPERIMENTAL RESULTS

In this section, we demonstrate that the proposed quality controller effectively improves the output quality of various IM circuits.

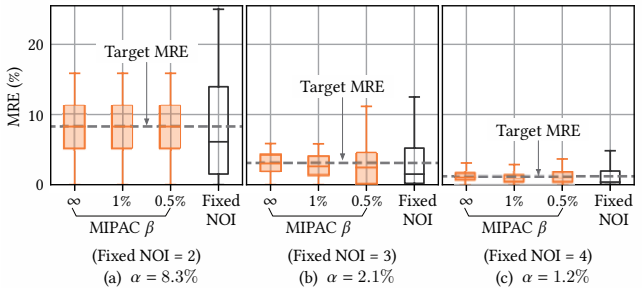

Figure 7: Error distribution of *div* with MIPAC constrained by various target MRE (α) power constraints (β) in comparison to the fixed-NOI baseline.

6.1 Experimental Setup

We evaluate the accuracy and latency of MIPAC implemented in Verilog HDL and present the energy-accuracy trade-off for various design-time and runtime parameters. We restrict the overall power consumption of the MIPAC controller to 10% of the target IM power consumption. MIPAC controller is generated for 8 quality settings for each application to provide fine-grained tunability. IM circuits and quality predictors are synthesized using Synopsis Design Compiler, targeting the NanGate 45 nm CMOS Technology. We compare and apply MIPAC to three IM circuits: a 16-bit multiplier (*mult*) [2], a 16-bit divider (*div*) [3], and an 8-bit exponentiation function (*exp*) [17]. Both *div* and *exp* are based on Taylor approximation, and *mult* is based on Mitchell's algorithm [11], which are widely used iterative approximation methods. As a baseline, we implement a DT-based NOI predictor similar to [7]. We assume uniformly distributed input data.

6.2 Experimental Results

We first evaluate the improvement in error distribution achieved by the proposed quality controller as a function of the power constraint β. Figure 7 shows the distribution of MRE of *div* for three target MREs under three different power constraints β. We vary β as a percentage of the target IM circuit's power consumption: 0.5% (most constrained), 1%, and ∞ (unconstrained). The 'Fixed NOI' is the baseline without quality control, equivalent to $\beta = 0$. The three target MREs 8.3%, 3.1%, and 1.2% correspond to those of fixed NOIs of 2, 3, and 4, respectively. The results show that MIPAC can significantly reduce the distribution of output quality to close to the target MRE. With only a 1% power constraint ($\beta = 1\%$), the output quality improvement is as substantial as unconstrained case ($\beta = \infty$). More specifically, when β is set to 1%, the maximum MRE is improved by up to 53%. The results confirm the efficacy of the proposed quality controller with only a minimal power overhead as well as the proposed design framework.

Next, we evaluate the output quality improvement by MIPAC in comparison to the DT-based controller and the baseline without any quality control. Both MIPAC and the DT-based controller are generated with the same WCE γ-values, determined to satisfy the corresponding target α-values. At each node in the DT, we consider the specified bit signals from the IM circuit as individual features and choose the split based on information gain. For iso-power comparison, we impose the same power constraint on the DT as MIPAC,

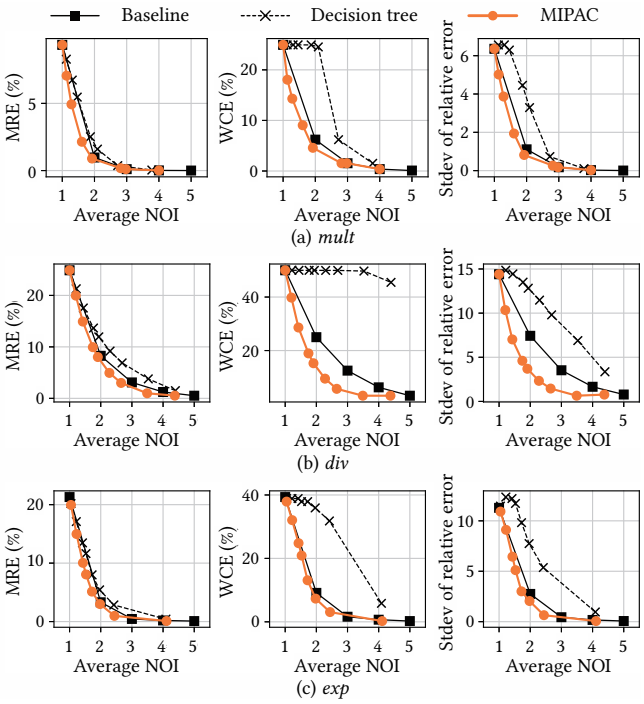

Figure 8: MRE, WCE, and standard deviation of relative errors without any control (Baseline) and with DT- and MIPAC-based quality control.

by restricting its maximum depth such that the overall power consumption for the set of DT predictors does not exceed 10% of the target IM power consumption. From Figure 8, we observe that MIPAC requires fewer average NOI for most quality settings, leading to additional energy savings. For the same average NOI, MIPAC also presents tighter average error variation than the baseline and DT. Although DT is unable to improve the output quality, it does provide finer-grained energy-accuracy control.

Finally, to compare the trade-off between energy efficiency and accuracy, we show the total energy consumption (including the quality controller's energy overhead) versus target MRE in Figure 9. With the input-dependent quality control, MIPAC exhibits significantly better energy-accuracy trade-off than the baseline and DT in almost all target MRE range. This is because MIPAC selectively runs each input for the minimum NOI needed to satisfy the overall target MRE, improving both energy and latency MIPAC achieves finer-grained energy-accuracy trade-off, which is essential for better energy-quality scaling of the application. This is possible because MIPAC incorporates input signals, allowing it to make input-dependent rather than input-agnostic NOI predictions.

7 CONCLUSIONS

Iterative approximation methods, such as Taylor approximation and Mitchell's algorithm, can dramatically improve the power efficiency of basic arithmetic operations. However, the lack of proper input-aware quality control has resulted in a wide input-dependent variation of output quality. In this paper, we presented a novel

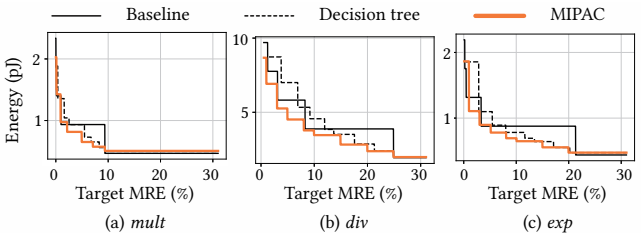

Figure 9: Comparison of energy-accuracy trade-off.

quality controller design called MIPAC, which adjusts the number of iterations depending on the input to better satisfy the desired quality. We also provided a design framework that generates the controller under a designer specified power constraint. By decoupling the underlying approximate hardware from compiler-driven control, MIPAC achieves high accuracy, low total energy consumption, and fine-grained quality control with only a minor power overhead. We apply MIPAC to an approximate divider, multiplier, and exponentiation function and demonstrated up to 53% worst-case output quality improvement compared to the baseline with only 1% energy overhead.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation under award CNS-1845469.

REFERENCES

[1] Syed Ershad Ahmed and M. B. Srinivas. 2019. An improved logarithmic multiplier for media processing. *Journal of Signal Processing Systems* 91, 6 (2019), 561–574.
[2] Zdenka Babić et al. 2011. An iterative logarithmic multiplier. *Microprocessors and Microsystems* 35, 1 (2011), 23–33.
[3] Setareh Behroozi et al. 2019. SAADI: A scalable accuracy approximate divider for dynamic energy-quality scaling. In *ASP-DAC*. 481–486.
[4] Vaibhav Gupta et al. 2011. IMPACT: Imprecise adders for low-power approximate computing. In *ISLPED*. 409–414.
[5] Jiawei Huang et al. 2012. A methodology for energy-quality tradeoff using imprecise hardware. In *DAC*. 504–509.
[6] Mohsen Imani et al. 2019. ApproxLP: Approximate multiplication with linearization and iterative error control. In *DAC*. 1–6.
[7] D. S. Khudia et al. 2015. Rumba: An online quality management system for approximate computing. In *ISCA*. 554–566.
[8] Younghyun Kim et al. 2020. Approximate hardware techniques for energy-quality scaling across the system. In *ICEIC*. 1–5.
[9] Vasileios Leon et al. 2018. Walking through the energy-error Pareto frontier of approximate multipliers. *IEEE Micro* 38, 4 (2018), 40–49.
[10] Divya Mahajan et al. 2016. Towards statistical guarantees in controlling quality tradeoffs for approximate acceleration. In *ISCA*. 66–77.
[11] John N Mitchell. 1962. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers* 4 (1962), 512–517.
[12] Vojtech Mrazek et al. 2018. Design of quality-configurable approximate multipliers suitable for dynamic environment. In *AHS*. 264–271.
[13] Arnab Raha and Vijay Raghunathan. 2017. Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system. In *DAC*. Article 74.
[14] Swagath Venkataramani et al. 2015. Approximate computing and the quest for computing efficiency. In *DAC*. 1–6.
[15] Liang-Kai Wang and Michael J Schulte. 2007. A decimal floating-point divider using Newton–Raphson iteration. *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* 49, 1 (2007), 3–18.
[16] Ting Wang et al. 2016. On effective and efficient quality management for approximate computing. In *ISLPED*. 156–161.
[17] Di Wu et al. 2019. SECO: A scalable accuracy approximate exponential function via cross-layer optimization. In *ISLPED*. 1–6.
[18] Qian Zhang et al. 2014. ApproxIt: An approximate computing framework for iterative methods. In *DAC*. 1–6.