Scheduling of Iterative Computing Hardware Units for Accuracy and Energy Efficiency

Setareh Behroozi
University of Wisconsin–Madison
sbehroozi@wisc.edu

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

Hoeseok Yang
Ajou University
hyang@ajou.ac.kr

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

***Abstract*—Iterative computing, where the output accuracy gradually improves over multiple iterations, enables dynamic reconfiguration of energy-quality trade-offs by adjusting the latency (i.e., number of iterations). In order to take full advantage of the dynamic reconfigurability of iterative computing hardware, an efficient method for determining the optimal latency is crucial. In this paper, we introduce an integer linear programming (ILP)-based scheduling method to determine the optimal latency of iterative computing hardware. We consider the input-dependence of output accuracy of approximate hardware using data-driven error modeling for accurate quality estimation. The proposed method finds optimal or near-optimal latency with a significant speedup compared to exhaustive search and decision tree-based optimization.**

***Index Terms*—Arithmetic operations, approximate computing, low power, scheduling**

I. Introduction

The past decade has witnessed the ever-increasing demand for energy-efficient computing driven by the prevalence of power-demanding applications on power-constrained computing devices. As a promising solution to the challenge of computing energy efficiency, *approximate computing* has emerged. Approximate computing is to produce "just good enough" results where imprecise results are sufficient for its purpose at much lower power consumption [1]–[4]. Various error-resilient applications where quality can be gracefully traded off for energy efficiency, such as sensing, signal processing, and, more recently, machine learning [5], have greatly benefited from the new computing paradigm and new hardware techniques based on it [6], [7].

Iterative computing is one such approach that progressively improves the output quality over multiple iterations. Using iterative computing hardware, one can either achieve high accuracy by allowing for more latency or high power efficiency by finishing computation sooner. To take advantage of this approach, various iterative computing hardware designs have been proposed, mainly for arithmetic operations [8]–[13]. However, in spite of the benefit, the adoption of iterative computing hardware has been impeded by the lack of an efficient method to determine the optimal latency to meet the energy and quality requirements at the same time.

There are several major challenges in determining the optimal latency of iterative computing hardware. First, the optimal latency is heavily input-dependent because the accuracy of approximate computing hardware varies significantly dependent on their input. Moreover, most applications consist of multiple inter-dependent computations where the output of one computation is input to another computation, making the accuracy estimation even more challenging. Finally, the design space complexity is exponential to the application complexity, rendering naive design space exploration methods (e.g., exhaustive search) impractical. Unfortunately, there exist no prior approximate computing hardware optimization methods are not suitable for the problem of scheduling of iterative computing hardware. Some recent work [14], [15] proposed methods for optimizing the selection of hardware with different accuracy, but they do not consider the case where the same hardware has varying accuracy depending on latency. A gradient descent-based method proposed [16] is not directly applicable since it does not consider the case where the same hardware is used multiple times throughout the processing pipeline. A more recent work [17] optimizes the latency of iterative hardware for a given accuracy constraint, but it does not consider multiple units.

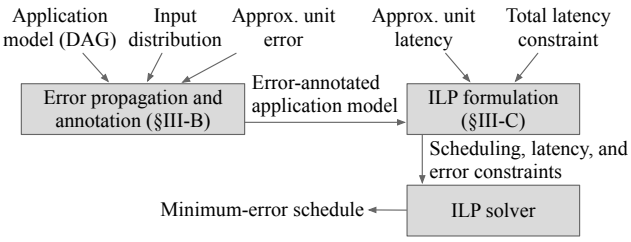


Fig. 1. Solving the scheduling problem for approximate hardware units using ILP.

In this paper, we propose an integer linear programming (ILP)-based scheduling method to address the aforementioned challenges. We consider the input dependence of output accuracy by using a data-driven error modeling of iterative computing hardware depending on its location within the application. The proposed error modeling enables accurate estimation of the final output quality for a given input distribution without running time-consuming simulations. We demonstrate that the proposed method successfully finds optimal or near-optimal scheduling solutions orders-of-magnitude faster than exhaustive search or decision tree-based optimization.

II. ILP-based Scheduling of Iterative Computing Hardware

In this section, we define the scheduling problem of iterative computing hardware along with our system model. Then we describe the error propagation model and annotation, followed by the formulation of the scheduling problem in ILP. Fig. 1

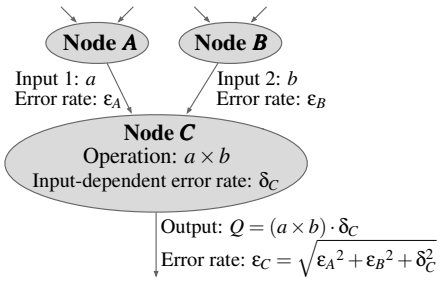


Fig. 2. Propagation of error through approximate multiplication in Node C.

illustrates the overall flow of scheduling described in this section.

We consider the hardware architecture composed of precise hardware units and iterative computing-based approximate hardware units. Precise hardware units have a fixed latency and always produce exact results. On the other hand, approximate hardware units have a finite number of discrete approximation levels to dynamically trade-off accuracy for latency. The implementation of iterative computing hardware can be based on various iterative approximation algorithms such as logarithmic operations [9] or Taylor approximation [10], [12], [13].

An application is modeled as a directed acyclic graph (DAG), $G = \langle V, E \rangle$, where operations are represented as nodes V, and dependencies between operations are presented as edges E between corresponding nodes. The distribution of the input is known at design time, which is common in application-specific system design. If not known, one can assume any statistical distribution. The scheduling problem is, given a total latency constraint, to find the optimal starting time and the latency of each operation that minimizes the error rate of the final output.

A. Error Propagation and Annotation

The output error of an approximate operation is a result of the input error rates and the error of the approximate operation, which can be modeled as the propagation of error [18]. The error rate of an approximate operation, δ, can be considered as a scaling factor that is multiplied to the precise result of the operation. That is, for an operation $f(a,b)$, the approximate output is

$$Q = f(a,b) \cdot (1+\delta), \tag{1}$$

and the error rate of Q is

$$\varepsilon = \sqrt{\varepsilon_{f(a,b)}{}^2 + \delta^2}, \tag{2}$$

where $\varepsilon_{f(a,b)}$ is the error rate when the operation $f(a,b)$ is precisely performed on erroneous inputs a and b. For example, consider the approximate multiplication in Node C in Fig. 2. When the inputs a and b have error rates of ε_A and ε_B, respectively, and the error rate of the approximate multiplication is δ_C, the error rate of the output of Node C is

$$\varepsilon_C = \sqrt{\varepsilon_A{}^2 + \varepsilon_B{}^2 + \delta_C{}^2}, \tag{3}$$

since the error propagation of multiplication $a \times b$ is

$$\varepsilon_{a \times b} = \sqrt{\varepsilon_A{}^2 + \varepsilon_B{}^2}. \tag{4}$$

TABLE I
NOTATION SUMMARY.

Notation	Definition
G	Application DAG $G = \langle V, E \rangle$
M_i	Set of approximation levels of Node i
P_i	Set of predecessors of Node i
L	Total latency constraint
$type_i$	Operation type of Node i
$mode_{i,k}$	If 1, approximation level of Node i is k; otherwise 0
$pr_{i,j}$	Nodes i and j share a hardware unit, and Node i precedes j
$d_i[k]$	Latency of Node i at approximation level k
$\delta_i[k]$	Error rate of Node i at approximation level k
t_i	Starting time of Node i
c_i	Latency of Node i
ε_i	Error rate of the output of Node i
i, j	Index of node
k	Index of approximation level

We estimate error rates for all approximate operations and all approximate levels ($\delta_i[k]$, described later in Table I). For that, we measure the error rate of a target operation (Node i) in all approximate levels (k) when all the other operations in the application are precise. We repeat this process for all operations within the application. Error rates for all approximate operations are normalized to reflect the amount of accuracy improvement for varying approximate levels in the error propagation model. Once all operations in the DAG are annotated with error rates, the error rate of the final node is the objective function to be minimized.

To ensure the high accuracy of the error model, we take the varying distribution of input of each node into consideration. Even for the same approximate operation and the same approximate level, the average error rate can be different depending on where in the DAG the operation is performed because input distribution is not the same. Input distribution is obtained by running the application with a given dataset without approximation and statistically characterizing the input distributions at each node.

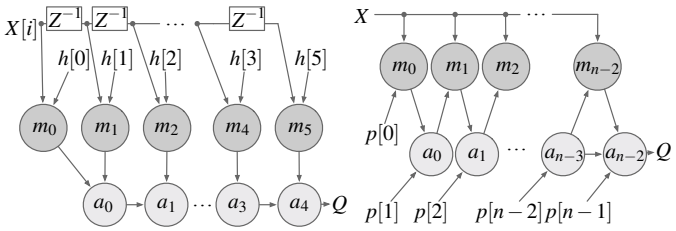
B. ILP Problem Formulation

From the error-annotated DAG, we generate ILP constraints. Table I lists the notations we use to describe the ILP formulation in this section. *Inputs to ILP-based scheduling* are (i) error-annotated application DAG generated in Section II-A, (ii) latencies of approximate hardware units at each approximation level, and (iii) total latency constraint L. The scheduling problem is to find the starting time (t_i) and the latency (c_i) of each node that *minimizes the error rate of the final node*, with respect to the constraints formulated in what follows.

a) Scheduling constraints: Since a single hardware unit is available per operation type, an arbitrary pair of nodes of the same operator type should not be overlapped in execution time. That is, $\forall i, j$ such that $type_i = type_j$,

$$\begin{cases} t_i + c_i \le t_j & \text{if } pr_{i,j} = 1 \\ t_j + c_j \le t_i & \text{otherwise} \end{cases}, \tag{5}$$

where $pr_{i,j}$ is a binary variable that is 1 if and only if the execution of v_i precedes that of v_j. In addition, the



(a) n-tap FIR (b) n-th order PoE

Fig. 3. Data dependency as a DAG diagram for (a) n-tap FIR and (b) n-th order PoE.

TABLE II
BENCHMARKS USED FOR EVALUATION.

Benchmark	FIR-5	FIR-16	FIR-32	PoE-6	PoE-16	PoE-32
Adders	5	16	33	5	15	31
Multipliers	6	17	32	5	15	31
Min. latency	7	18	34	10	30	62
Max. latency	13	34	65	15	45	93
No. of solutions	64	131,072	>8M	32	32,768	>2M

execution dependencies specified in E should be satisfied in the scheduling decision. Thus, $\forall(v_i, v_j) \in E$, the following inequality should hold:

$$t_i + c_i \leq t_j. \tag{6}$$

The latency of each node, c_i, is determined by the following equation, i.e., for each $v_i \in V$,

$$c_i = \sum_{k=1}^{|M_i|} mode_{i,k} \cdot d_i[k], \tag{7}$$

where $mode_{i,k}$ is a binary variable that is 1 if v_i is chosen to operate at the k-th approximation level selected from M_i, a set of approximation levels of v_i, and $d_i[k]$ denotes the latency of v_i at the k-th approximation level. Finally, each node has a single approximation level chosen, i.e., for all $v_i \in V$.

$$\sum_{k=1}^{|M_i|} mode_{i,k} = 1. \tag{8}$$

b) Total latency constraint: A valid scheduling solution should always be completed by the given total latency constraint L, i.e.,

$$\max_{v_i \in V}(t_i + c_i) \leq L. \tag{9}$$

c) Error constraints: The error rate of an approximate operator node is determined based on the errors transferred from input operands and the error generated from the node itself as shown in (3). In order to keep the linearity of the formulation, both left and right sides of (3) are squared:

$$\varepsilon_i{}^2 = \sum_{\forall v_j \in P_i} \varepsilon_j{}^2 + \sum_{k=1}^{|M_i|} mode_{i,k} \cdot \delta_i[k]^2, \tag{10}$$

where P_i denotes a set of predecessor nodes of v_i, i.e., $\forall v_j \in P_i$, $\exists(v_j, v_i) \in E$. Note that this linearization (squared error rate representation) does not jeopardize the optimality of the solution as ε is non-negative, and minimizing ε is equivalent to minimizing ε^2.

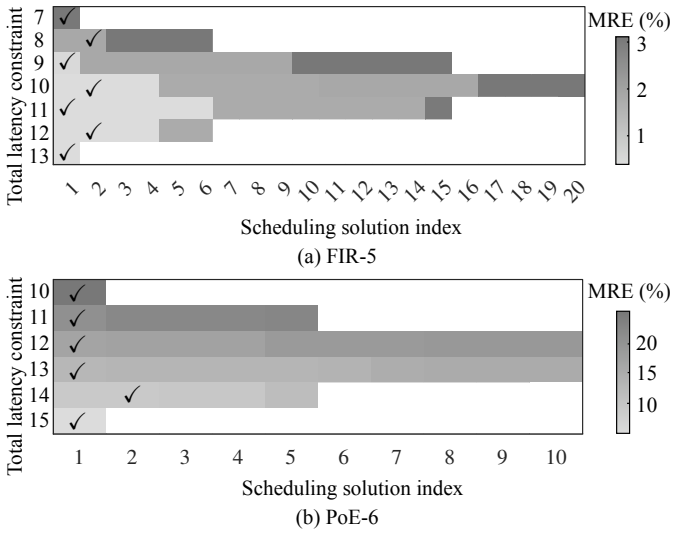


Fig. 4. MRE of all possible scheduling solutions of (a) FIR-5 and (b) PoE-6. The proposed scheduler's solutions are highlighted with checkmarks (✓).

III. Experimental Results

We evaluate the proposed ILP-based scheduler in terms of accuracy and optimization execution time.

A. Experimental Setup

We consider two types of benchmark applications: (i) finite impulse response (FIR) filters applied to electrocardiogram (ECG) signals and (ii) polynomial evaluation (PoE) applied to normally distributed random inputs. Fig. 3 illustrates the dependency in a n-tap FIR and a n-th order PoE benchmark in a DAG. Table II shows the number of addition and multiplication operations in the benchmarks and the number of all possible scheduling solutions. The 16-bit TIM multiplier [9] is used for approximate multiplication. Its average accuracy for uniformly distributed inputs is 94.77% in the first cycle and 98.45% in the second cycle. For the accuracy evaluation, we compare the mean relative error (MRE) of the schedules obtained by the proposed scheduler and the true optimal solution obtained by exhaustive search, or decision tree-based optimization if the exhaustive search is not tractable due to the large number of possible solutions. As mentioned in Section I, we use these generic optimization methods as the baselines since there exists no prior optimization method for this problem. The execution time is measured on a PC with the 3.4 GHz quad-core Intel Core i5 CPU with 16 GB of RAM. We use Gurobi [19] as the ILP solver.

B. Optimality of Scheduling Solutions

We first evaluate the proposed ILP-based scheduler by comparing the accuracy of the scheduling solutions obtained by ILP to that of all possible scheduling solutions. Because of the sheer number of the all possible solutions, exhaustive search is not tractable even for medium-size benchmarks, thus we use the smallest benchmarks for demonstration. Figures 4(a) and 4(b) show the accuracy of all scheduling solutions of FIR-5 and PoE-6, respectively. Each cell represents one scheduling

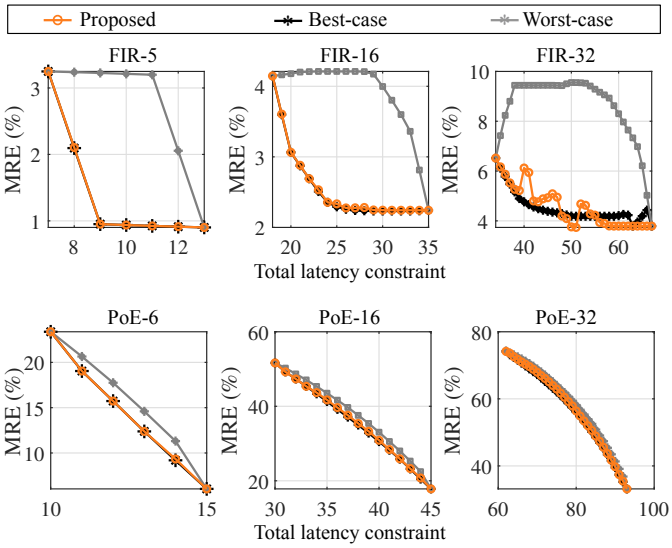


Fig. 5. Comparison of proposed scheduling with baselines. Exhaustive search is used as baselines for FIR-5 and PoE-6; decision tree for FIR-16, FIR-32, PoE-16, and PoE-32.

solution. Each row is a set of solutions with the same latency, sorted by accuracy in ascending order, ranging from the minimum (all multipliers run for one cycle) to the maximum (all multipliers run for two cycles). The left-most solution is the optimal solution for each latency constraint. It exhibits a wide variation in accuracy between scheduling solutions, not only by latency but also within the same latency. For each latency constraint, cells highlighted by checkmarks (✓) represent the scheduling solutions obtained by the proposed scheduler. The results show that the proposed scheduler is able to find the optimal solutions in most latency constraints in both benchmarks, and even the solutions that are not optimal are still as almost accurate as the optimal.

Next, we evaluate the accuracy for larger benchmarks in Fig. 5. We use exhaustive search as the baseline for FIR-5 and PoE-6 that are small, and decision tree for others. We compare the proposed scheduler's solution to the best-case and worst-case schedules obtained by the baseline schedulers. For FIR-5 and PoE-6, the proposed scheduler successfully finds the optimal or near-optimal solutions in all cases, which is also seen in Fig. 4. The proposed scheduler solutions also achieve good accuracy for larger benchmarks as well at a much lower execution time as will be shown in Section III-D.

C. Optimality of Scheduling Solutions under Input Variation

Error modeling in Section II-A relies on the distribution of input known at design time, but the actual input at run time may be different and may have a different optimal schedule. To investigate the impact, we evaluate the optimality of scheduling solutions that are optimal for one dataset when applied to another dataset. We use two sets of data: (i) a train dataset, based on which a scheduling solution is optimized, and (ii) a test dataset, to which the solution is applied to. For FIR benchmarks, we use two different sets of ECG signals as the train and test datasets. For PoE benchmarks, we use two sets of normally distributed random numbers with different mean values ($\mathcal{N}(0,2^8)$ and $\mathcal{N}(2^4,2^8)$). We find the optimal scheduling solution for the test dataset, and if it is different from that of the train dataset, we compare the accuracy by both scheduling solutions. All benchmarks result in the same or only slightly different optimal solutions as shown in Table III in spite of the different input distributions and error models. The results confirm the need for input-aware error modeling and also demonstrate the robustness of scheduling solutions under input variation.

TABLE III
NUMBER OF SAME LATENCIES IN THE OPTIMAL SCHEDULES FOR TRAIN AND TEST DATASETS, AND APPLICATION ACCURACY DEGRADATION OF SCHEDULING UNDER INPUT VARIATION.

Benchmark	FIR-5	FIR-16	FIR-32	PoE-6	PoE-16	PoE-32
Same latencies	2/6	18/18	31/34	4/6	2/16	2/32
MRE increase	0.13%	0.00%	1.29%	1.03%	0.57%	0.31%

TABLE IV
EXECUTION TIME IN SECONDS. (†: ESTIMATED FROM TIME FOR EVALUATING ONE SOLUTION AND THE NUMBER OF SOLUTIONS.)

Benchmark	FIR-5	FIR-16	FIR-32	PoE-6	PoE-16	PoE-32
Exhaustive search	2,760	4.0E6†	4.9E11†	47	1.5E5†	1.5E10†
Decision tree	861	14,455	37,169	23	730	15,672
Proposed ILP	0.1	1.3	19.4	0.1	0.8	5.1

D. Scheduling Optimization Execution Time

Finally, we evaluate the execution time of the proposed scheduling method against the measured or estimated execution time of other scheduling methods. As shown in Table IV, using the exhaustive search or decision tree-based optimization is not tractable due to the large number of possible solutions. The proposed scheduling method dramatically reduces the execution time by more than a billion times for the complex FIR-32 and PoE-32 benchmarks, making the scheduling of approximate hardware units feasible for complex applications.

IV. CONCLUSION

An ILP-based scheduling is proposed to determine the optimal latency of iterative approximate hardware to fully exploit the energy-accuracy trade-off. The proposed scheduling method benefits from the novel data-driven error modeling of iterative computing hardware to ensure the optimality of the solutions. The presented error model reflects the effect of input variation on approximation accuracy to better assist with the scheduling. Based on the error modeling, we presented an ILP formulation for the scheduling problem. The evaluation results show that the proposed scheduler achieves optimal or near-optimal scheduling solutions in significantly less execution time compared to enumerating approaches such as exhaustive search or decision tree.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proceedings of the IEEE European Test Symposium (ETS)*, 2013, pp. 1–6.

[2] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proceedings of the Design Automation Conference (DAC)*, 2015, pp. 120:1–120:6.

[3] M. Alioto, "Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, pp. 127–132.

[4] Y. Kim, J. San Miguel, S. Behroozi, T. Chen, K. Lee, Y. Lee, J. Li, and D. Wu, "Approximate hardware techniques for energy-quality scaling across the system," in *Proceedings of the International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–5.

[5] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 701–706.

[6] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.

[7] M. Alioto, V. De, and A. Marongiu, "Energy-quality scalable integrated circuits and systems: Continuing energy scaling in the twilight of Moore's law," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 653–678, 2018.

[8] Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocessors and Microsystems*, vol. 35, no. 1, pp. 23–33, 2011.

[9] S. E. Ahmed and M. Srinivas, "An improved logarithmic multiplier for media processing," *Journal of Signal Processing Systems*, vol. 91, no. 6, pp. 561–574, 2019.

[10] S. Behroozi, J. Li, J. Melchert, and Y. Kim, "SAADI: A scalable accuracy approximate divider for dynamic energy-quality scaling," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 481–486.

[11] H. Saadat, H. Javaid, A. Ignjatovic, and S. Parameswaran, "WEID: Worst-case error improvement in approximate dividers," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 593–598.

[12] J. Melchert, S. Behroozi, J. Li, and Y. Kim, "SAADI-EC: A quality-configurable approximate divider for energy efficiency," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2680–2692, 2019.

[13] D. Wu, T. Chen, C. Chen, O. Ahia, J. San Miguel, M. Lipasti, and Y. Kim, "SECO: A scalable accuracy approximate exponential function via cross-layer optimization," in *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[14] C. Li, W. Luo, S. S. Sapatnekar, and J. Hu, "Joint precision optimization and high level synthesis for approximate computing," in *Proceedings of the Design Automation Conference (DAC)*, 2015, pp. 1–6.

[15] M. A. Hanif, R. Hafiz, O. Hasan, and M. Shafique, "PEMACx: A probabilistic error analysis methodology for adders with cascaded approximate units," in *Proceedings of the Design Automation Conference (DAC)*, 2020, pp. 1–6.

[16] A. Raha and V. Raghunathan, "Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system," in *Proceedings of the Design Automation Conference (DAC)*, 2017, pp. 1–6.

[17] T. Kemp, Y. Yao, and Y. Kim, "MIPAC: Dynamic input-aware accuracy control for dynamic auto-tuning of iterative approximate computing," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 248–253.

[18] I. Hughes and T. Hase, *Measurements and their uncertainties: a practical guide to modern error analysis*. Oxford University Press, 2010.

[19] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com