Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits

Wei-Ting J. Chan[1], Andrew B. Kahng[1,2], Seokhyeong Kang[1], Rakesh Kumar[3], and John Sartori[4]
[1]ECE and [2]CSE Departments, UC San Diego, La Jolla, CA 92093
[3]Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801
[4]ECE Department, University of Minnesota, Minneapolis, MN 55455
{wechan, abk, shk046}@ucsd.edu, rakeshk@illinois.edu, jsartori@umn.edu

***Abstract*—Aggressive requirements for low power and high performance in VLSI designs have led to increased interest in approximate computation. Approximate hardware modules can achieve improved energy efficiency compared to accurate hardware modules. While a number of previous works have proposed hardware modules for approximate arithmetic, these works focus on solitary approximate arithmetic operations. To utilize the benefit of approximate hardware modules, CAD tools should be able to quickly and accurately estimate the output quality of composed approximate designs. A previous work [10] proposes an interval-based approach for evaluating the output quality of certain approximate arithmetic designs. However, their approach uses sampled error distributions to store the characterization data of hardware, and its accuracy is limited by the number of intervals used during characterization.**

In this work, we propose an approach for output quality estimation of approximate designs that is based on a lookup table technique that characterizes the statistical properties of approximate hardwares and a regression-based technique for composing statistics to formulate output quality. These two techniques improve the speed and accuracy for several error metrics over a set of *multiply-accumulator* testcases. Compared to the interval-based modeling approach of [10], our approach for estimating output quality of approximate designs is 3.75× more accurate for comparable runtime on the testcases and achieves 8.4× runtime reduction for the error composition flow. We also demonstrate that our approach is applicable to general testcases.

***Keywords*—Approximate computation, error modeling**

I. Introduction

Modern VLSI designs face increasingly significant challenges in meeting the power and performance constraints demanded by present and future computing systems. Recently, approximate computing has been explored as a means of improving energy efficiency for noise-tolerant applications. While approximate computation circuits have been shown to be effective at improving energy efficiency at the expense of perfect functional correctness, modern CAD tools are inequipped to perform design automation for designs that contain approximate computation circuits. One key building block required for CAD tools that can create efficient approximate designs is the ability to quickly and accurately estimate the output quality of designs composed of approximate computation circuits. Such functionality is necessary for CAD tools that would minimize the energy of a design during synthesis, optimization, etc. while maintaining acceptable output quality, as specified by system designers. In this paper, we propose a flow that can analyze how errors originate and propagate in designs composed of approximate computation circuits to quickly and accurately estimate the output quality at nets in an approximate design. The following terminology is relevant to our treatment of approximate circuit design.

- *Error metric* (EM): a unit of measure that quantifies the deviation between the outputs produced by a functionally correct design and an approximate design. We review several commonly-used EMs from the existing literature in Section II-A below.
- *Approximate hardware module*: a hardware module that is functionally incorrect by design (e.g., approximate adders and multipliers).
- *Approximate circuit*: a circuit that contains one or more approximate hardware modules. Figure 1 compares an approximate circuit to its accurate counterpart.
- *Composed EM* ($EM_{composed}$): the estimated EM value at an output or internal net in an approximate circuit.
- *Pre-characterized EM* (EM_{char}): a sampled EM for an individual approximate hardware module that has been stored in a library. We measure EM_{char} using Gaussian random variables as inputs and propose composition rules for different EMs.
- *Composition function*: a function that maps EM_{char} to $EM_{composed}$.
- $D_{ref}(X)$: the output of a correct circuit (not approximate) for an input distribution X.
- $D_{appx}(X)$: the output of an approximate circuit for an input distribution X.

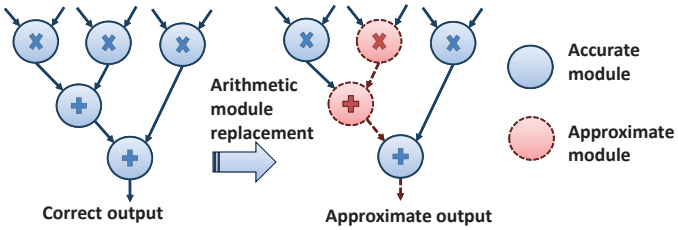


Fig. 1. Illustration of approximation module replacement, in which accurate hardware modules are replaced by approximate ones.

Given the above definitions, the *error metric composition* problem seeks to find a composition function for a composed EM as described in Equation (1), where EM^i_{char} denotes the EM_{char} of the i^{th} approximate module in an approximate circuit. Values of EM_{char} for different approximate hardware modules may be stored in a library for quick reference during computation of $EM_{composed}$.

$$EM_{composed} = f\left(EM^1_{char},\ EM^2_{char},\ \ldots,\ EM^n_{char}\right) \quad (1)$$

In this paper, we propose an automated methodology to estimate $EM_{composed}$ at the outputs and internal nodes of an approximate circuit. Our approach accounts for the relationship between EM behavior, input distribution statistics, and hardware characteristics of approximate hardware modules. We use lookup tables parameterized by approximate hardware module statistics to accelerate EM composition; we further incorporate regression-based models that capture how errors propagate through the topology of an approximate circuit. We demonstrate and validate our methodology with several randomly generated benchmark circuits with varying complexity as well as designs evaluated in previous work [9] (e.g., FIR filter).

978-1-4799-2987-0/13/$31.00 ©2013 IEEE

We make the following contributions.

- We analyze the interval-based approach in [9] [10], and explore the potential drawbacks of the approach.
- We propose composition rules for estimating the EM observed at any net within an approximate circuit.
- We develop an approach to build pre-characterized libraries for individual approximate hardware modules and demonstrate how to accelerate the computation of composed EMs using the libraries. Our approach reduces runtime for characterization and results in improved accuracy compared to previous works [9] [10].
- Compared to previous works, we improve the accuracy of EM estimation by 3.75× for the same runtime. We achieve 1.36× and 8.4× runtime improvements, respectively, for library characterization and error composition.
- We demonstrate that our proposed approach achieves accurate estimates for approximate FIR circuits as well as random artificial circuits with various topologies.

The remainder of the paper is organized as follows. Section II reviews related works according to different system abstraction levels and error sources. Section III describes the motivation of error metric composition and our search for rules that govern composition of error distributions. In Section IV, we show how to apply our composition rules and pre-characterized error libraries to analyze arbitrary circuit topologies. Section V concludes the paper and gives directions for our future work.

II. Related Works

A. Error Metrics

Definitions of EMs from the literature are given in Equations (2) to (7). Note that $E[\cdot]$ indicates the *expected value* of a random variable and $max[\cdot]$ indicates the *maximum value*.

$$ER = \sum_{X s.t. D_{appx}(X) != D_{ref}(X)} Pr(X) \tag{2}$$

$$ES = E[D_{appx}(X) - D_{ref}(X)] \tag{3}$$

$$ARES = E[(D_{appx}(X) - D_{ref}(X))/D_{ref}(X)] \tag{4}$$

$$MSE = E[|D_{appx}(X) - D_{ref}(X)|^2] \tag{5}$$

$$SNR = E[|D_{ref}(X)|^2/|D_{appx}(X) - D_{ref}(X)|^2] \tag{6}$$

$$MAXE = \max_X[|D_{appx}(X) - D_{ref}(X)|] \tag{7}$$

- *Error rate* (ER) [15] is used to evaluate the likelihood of correctness in arithmetic operations. An accurate estimation of ER is important in the case where approximate circuits spend additional cycles for error corrections.
- *Error significance* (ES) [27] addresses the magnitude of errors. We define *ES* as the signed average difference between correct and erroneous results.
- *Average relative error significance* (ARES) is used to measure the impacts of errors for image processing in [12] [29]. ARES is defined as the average absolute difference between correct and erroneous results, normalized to correct results. In digital signal processing (DSP) circuits, the magnitude of errors is important because small errors may be masked by other noise sources.

TABLE I. Categories of error analysis, propagation, and optimization works.

Category	(C1)	(C2)	(C3)	(C4)
Manipulated Elements	Logic cell	Arithmetic	Arithmetic	Multiple Levels
Error Source	Appx. HW	Rounding	Appx. HW	Over-scaled VDD
Probabilistic Errors	N	N	N	Y
Reference	[27] [21] [23]	[6] [24] [1] [2] [22] [17] [18] [14]	[8] [10] [9]	[13] [3] [4] [25]

- *Mean squared error* (MSE) in [5] [28] and *signal-to-noise ratio* (SNR) in [5] [7] are common metrics to measure signal degradation in communication and image processing systems.
- *Maximum error* (MAXE) is defined as the maximum absolute value of produced errors. In [9], the MAXE metric is used to evaluate approximate circuits.

B. Approximate Arithmetic Modules

Various approximate arithmetic modules have been proposed in previous works, where aggressive timing and power benefits are obtained by breaking critical paths in the approximate module. To achieve a bounded error significance or configurable error rate, several techniques have been applied to reduce the severity of errors in these approximate hardware modules. ETAI [29] limits the maximum error by detecting a carry propagation and setting all lower sum bits to "1". A similar compensation approach is used in Shin's approximate adder [23], which detects a carry propagation using a specially designed truth table. By using error compensation approaches, the error can be reduced compared to simply breaking the carry chain. ETAIIM [29], ACA-SD [12], Lu's adder [19], and ACA-X [26] use a *carry-look-ahead* (CLA)-based approach to shorten the longest carry propagation path in the adder. These adders are composed of CLA submodules, and the numbers and sizes of the submodules can be configured at design time. The error significance and error rate can be configured by changing the length of carry propagation paths. Kahng et al. [12] also show that the errors can be detected and corrected in each CLA block, and that the accuracy can be configurable during runtime.

C. Analysis and Composition of Errors

We categorize existing works on hardware error analysis into four categories as shown in Table I. In category (C1), the works focus on searching for useful approximations during logic synthesis. Venkataramani et al. [27] work with existing commercial synthesis tools and simplify logic based on *approximate don't-care* (ADC) information under a given error significance bound. Miao et al. [21] focus on a methodology to design more efficient adders by combining logic components to reduce the maximum error. In [23], Shin et al. provide a heuristic to search for useful approximations based on a truth table to study the tradeoff between the error rate and literal terms (hardware cost). Previous works in category (C2) address rounding errors between floating-point and fixed-point conversions. In these works, the rounding errors are determined by the wordlength of hardware, and so are different from the errors induced by approximate hardware. In category (C3), [9] and [10] use an interval-based approach (*interval arithmetic* or *affine arithmetic*) to propagate errors. The interval-based approach uses pre-characterized libraries for error estimations, but the runtime of characterization can increase when more intervals are required for large ranges of signals. In category (C4), existing works assign overscaled supply voltages to achieve a graceful accuracy degradation. Kedem et al. [13] analyze propagations of errors induced by the degraded supply

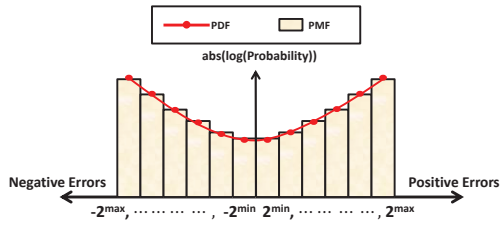


Fig. 2. Probability mass function (PMF) used in the interval-based approach is shown. Note that the error magnitudes are in log scale. The error magnitudes will be clamped to $\pm 2^{max}$ or $\pm 2^{min}$ if they are out of range.

voltage, and they simplify the analysis by assuming that no error cancellations occur between multiple adders. Venkatesan et al. [25] propose the *MACACO* flow to evaluate propagations of errors induced by overscaled supply voltage. They also apply this approach to characterize errors for different approximate adders. Chippa et al. in [3] [4] propose methodologies to analyze and optimize computing effort at different levels of abstraction, and also consider errors due to overscaled voltage supplies.

Compared to previous works, our work focuses on (1) the error propagation at arithmetic-level instead of gate-level computation, (2) the errors induced by approximate hardware as opposed to the overscaled supply voltage, and (3) awareness of both ER and ES. Furthermore, we simplify the composition of errors with a pre-characterized library and regression coefficients.

III. Analytical Studies on Error Estimation

A. Analysis of Existing Interval-based Approach

Huang et al. address the issue of error rate estimation for approximate circuits in [9] [10]. Their flow first characterizes approximate hardware modules by simulating the error probabilities for different input value intervals. Then, with given input operand distributions, they use *interval arithmetic* to estimate the *probability mass function* (PMF) of errors produced and propagated in an approximate arithmetic circuit. After propagating and composing errors with interval arithmetic, the error metrics are obtained from PMFs. Their interval-based approach samples the *probability distribution functions* (PDFs) or PMFs of errors to generate sampled PMFs. The height of each interval in the sampled PMF represents the probability of error. Figure 2 shows an example PMF which is used in the interval-based approach. Due to the limited number of intervals used for characterization, the error magnitudes will be clamped to $\pm 2^{max}$ or $\pm 2^{min}$ if they are out of range. As a result, the accuracy of the interval-based approach will be impacted if the range of characterization does not match the inputs.

We observe two drawbacks in the interval-based approach. First, there is a quantization error, since the approach represents multiple error values with a single interval. If the actual error distribution varies greatly within one interval, the estimation will be inaccurate. This may particularly be an issue for large intervals closer to $\pm 2^{max}$. For such large intervals, quantization error may be quite substantial. For example, an error value of $2^{max-1}+1$ is placed in the bin for 2^{max}, and the quantization error $(2^{max-1}-1)$ is essentially as large as the error value itself $(2^{max-1}+1)$. This example also illustrates another potential drawback. Unless the error value is an exact power of two, the quantization error for a large interval tends to be large. Perhaps counter-intuitively, error values that are very close to an interval value may cause very large errors. Second, the interval-based approach requires consecutive intervals to cover the range from maximum to minimum error magnitude ($\pm 2^{max}$ and $\pm 2^{min}$ in Figure 2). If the errors fall out of $\pm 2^{max}$ or $\pm 2^{min}$, the interval-based approach will clamp the estimated errors to the $\pm 2^{max}$ or $\pm 2^{min}$ values, and the estimation error will be saturated. If a large portion of errors or data experience this saturation issue, the estimation inaccuracy will be high. To address these drawbacks, the interval-based approach requires re-characterization of the libraries to increase the number of intervals, incurring significant runtime overhead. For better understanding of the strengths and weaknesses of the interval-based EM composition, we evaluate the EM estimation with a testcase shown in Figure 3(a). We vary the input distribution to evaluate accuracy for different input distributions and hardware configurations. We collect results from 100 combinations (10 Gaussian distributions with different standard deviations and 10 sets of ETAIIM configurations). Figure 3(b) shows the runtime of library characterization performed by the interval-based approach for different numbers of samples per interval. The accuracy results of the interval-based approach compared to Monte Carlo simulation are shown in the form of a correlation plot in Figure 3(c). From Figure 3(b) we notice that increasing the sample size to 18.5M requires 1.7 hours for library characterization, but estimation errors (offsets) are still observed in Figure 3(c). Possible reasons for the inaccuracy are (1) the use of discrete PMF and (2) inaccurate propagation of EMs from the pre-characterized library.

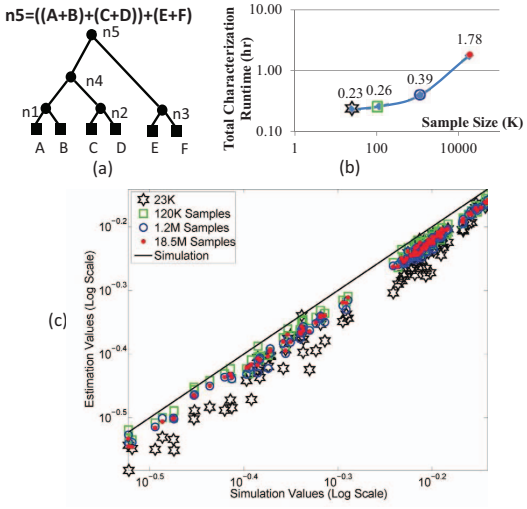


Fig. 3. (a) Five-node test case. (b) Runtime from interval-based approach for each sample size. (c) ER estimation results from interval-based approach. The results are generated from 100 testcases (10 hardware configurations and 10 combinations of input distributions).

B. Analysis for Computation of Error Metrics

We analyze an ETAIIM adder to understand the error generation of approximate modules. ER of the ETAIIM adder is given in Equation (8). N is the total bit width of the adder; bits-per-block (BPB) is the size of carry-look-ahead (CLA) blocks; k is the number of connected CLA blocks, an architectural parameter used to control error magnitudes. From Figure (4), we observe that the errors are related to the input values of CLA blocks because errors occur when all input bits of the CLA block are in carry-propagate state. For example, if most of the input values are small, then the probability of generating larger errors will be small. This observation regarding ETAIIM motivates us to study the sensitivity of EMs to input distributions.

$$ER_{ETAIIM} = 1-(1-\frac{1}{2^{BPB}}\frac{2^{BPB}-1}{2^{BPB+1}})^{\frac{N}{BPB}-2-k} \times(1-\frac{1}{2^{(BPB\cdot k)}}\frac{2^{(BPB\cdot k)}-1}{2^{(BPB\cdot k)+1}}) \quad (8)$$

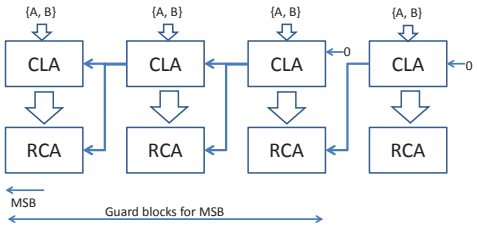


Fig. 4. The structure of an ETAIIM approximate adder. CLAs are carry-lookahead sub-adders. RCAs are ripple-carry sub-adders.

C. Proposed Approach to Estimate EMs

The analytical expression in Equation (8) is based on the assumption that distributions of the input values are uniform and the ranges cover from the MSB to LSB. However, this is not always the case, and we need to consider input distributions for the accurately-estimated EMs. To analyze the relationship between input distributions and EMs, we use 24-bit ETAIIM adders and simulate the EMs for different *BPB* and k. In this motivating experiment, both operands are assumed to have the same standard deviation for simplicity. Figure 5 shows each simulated EM value (y-axis) with respect to the standard deviation of input data (x-axis).

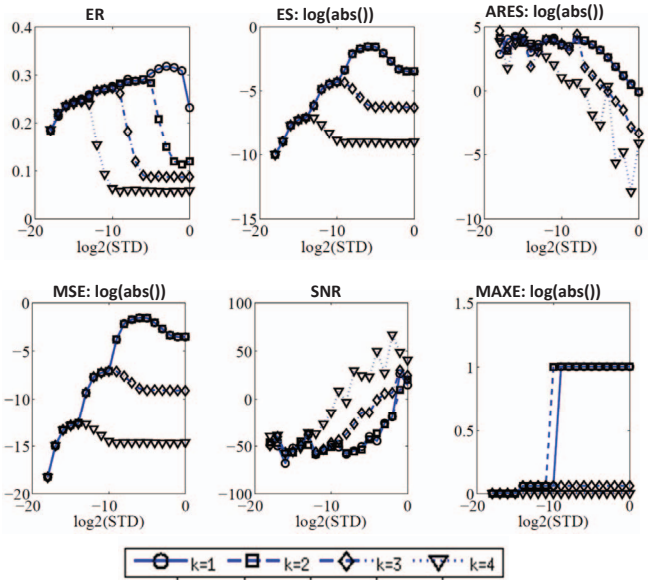


Fig. 5. The simulated EM results for input distributions. A 24-bit signed ETAIIM adder is simulated in the analysis. 20 bits are used for the fractional part, and the MSB guard block size k takes on values from one to four. For simplicity, both operands are assumed to have the same standard deviation.

Figure 5 shows that EM values change with respect to both the standard deviations of input values and hardware configuration (k). Based on the results, we construct lookup tables to model the error metric of approximate modules instead of using analytical expressions. Modeling with lookup tables is preferred since it is difficult to derive an analytical expression if input values are not uniformly distributed.

Figure 7 illustrates our EM formulation. To estimate the output EM (EM_Z), we consider *intrinsic* EM values (EM_{in}) which are generated by the approximate module itself, and *propagated* EM values (EM_A, EM_B) which come from the previous stages. We propose a lookup table (LUT)-based approach to consider different input distributions. The lookup tables for different hardware configurations

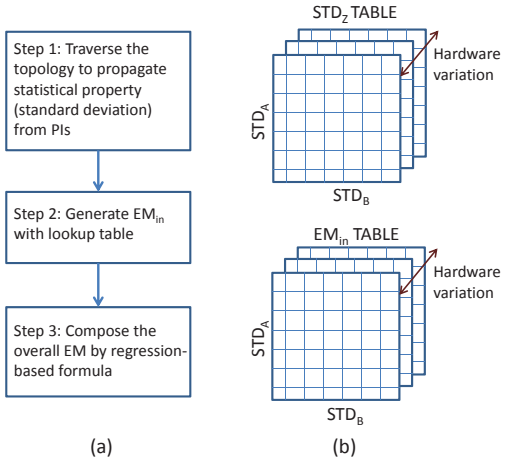


Fig. 6. (a) Our proposed approach for error estimation, and (b) the lookup tables in the pre-characterized libraries for EM_{in} and STD_Z.

are merged to become the pre-characterized library. We construct two types of lookup tables, as illustrated in Figure 6(b). The EM_{in} and STD_Z tables respectively contain (intrinsic) EM values and output standard deviations with respect to the input standard deviations.

Our LUT-based approach can be divided into three steps as described in Figure 6(a).

Step 1: Value distribution propagation in the circuit topology. We generate statistical properties with pre-characterized libraries. To obtain the statistical property of each node in the circuit, we traverse all the nodes in the circuit in a topological order from primary inputs to a primary output. During the traversal, we look up the statistical property (standard deviation) from a pre-characterized table (STD_Z), and annotate the standard deviation values at all nodes. The results in the upper-left plot in Figure 8 demonstrate the feasibility of this approach.

Step 2: EM estimation for approximate modules. With standard deviations of the internal nodes, we estimate EM values using a pre-characterized table (EM_{in}) for each internal node. The lookup table, EM_{in}, is characterized by simulating EM values as shown in Figure 5. We generate the LUTs for different approximate modules to estimate intrinsic error metric (EM_{in}), which is generated by modules themselves without input errors. By combining Steps 1 and 2, we can estimate the EM_{in} of each node in any circuit topology.

Step 3: Error composition with EMs of each approximate module. With the generated EMs (EM_{in}) of each approximate module, we apply a regression approach to find the composed EM values in the primary output. The error rate (ER) can be computed by multiplying pass rate (1-ER), and the composed ER is generated with Equation (9), where ER_Z is the composed ER, ER_A and ER_B are the propagated ERs to the inputs in Figure 7, ER_{in} is an intrinsic ER, and $\alpha_{\{in,P\}}$ are regression coefficients. Other EMs (ES, ARES, MSE, SNR and MAXE) are amplitude-based error metrics, and we generate the composed EM from Equation (10), where $\alpha_{\{in,P,C\}}$ are regression coefficients. These composition rules are developed for adders, and their generalization to multiplication and other arithmetic operations is a subject of future work.

$$ER_Z = 1 - 10^{\alpha_C} \cdot (1 - ER_{in})^{\alpha_{in}} \cdot ((1 - ER_A) \cdot (1 - ER_B))^{\alpha_P} \quad (9)$$

$$EM_Z = \alpha_{in} EM_{in} + \alpha_P (EM_A + EM_B) + \alpha_C. \quad (10)$$

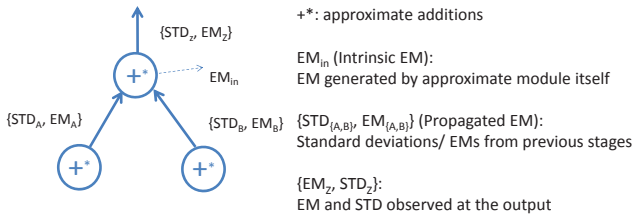


Fig. 7. EM estimation at a given node (approximate module) considering intrinsic and propagated EMs.

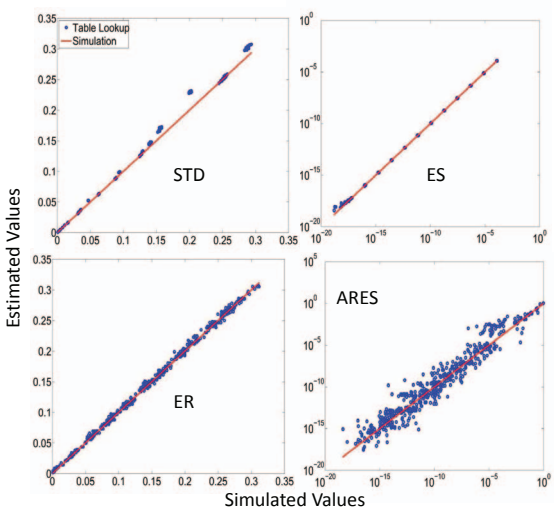


Fig. 8. Estimated STD_A or STD_B and EM_{in} values obtained from lookup tables. The x-axes are simulated values and y-axes are estimated values. The red lines show the ideal estimations and the blue dots show the estimated results from our proposed method.

To verify the correctness of our table lookup method in Step 2, we estimate the standard deviation (STD) and EM_{in} as shown in Figure 8. We test with 10 combinations of hardware configurations and 10 combinations of different input distributions (Gaussian distribution with different standard deviations). Figure 8 shows the correlations between the estimated and simulated STD/EM values from all internal nodes. The results show that we can obtain correct STD values from the lookup table with the topology traversal. With the estimated STD, we observe correct estimation for EM_{in} (ER and ES). We find that ARES results are less accurate compared to the results of ER and ES. This is because ARES measures error relative to input data. If the magnitude of input data is small (near zero), the range of the ARES value will be large. In such a context, accurate estimations are difficult, given the limited number of grids in the lookup table.

Table II shows our regression results for improved EM estimations. The upper part (a) of the table shows regression coefficients derived with different hardware configurations. The lower parts of the table show (b) the *estimation inaccuracy* and (c) the *absolute estimation inaccuracy*, as defined in Equation (11) and (12), where R_c and R_e are the simulated and estimated results, respectively. The two inaccuracy metrics are shown for both "without regression" and "with regression" cases.

$$Estimation\ inaccuracy\ =\ |R_c - R_e|/|R_c| \tag{11}$$

$$Absolute\ estimation\ inaccuracy\ =\ |R_c - R_e| \tag{12}$$

Without regression, we report the results with $\alpha_{IN} = \alpha_P = 1$ and $\alpha_C = 0$ for the coefficients in Equations (9) and (10); this is a pessimistic assumption (i.e., that there are no overlap effects from the composition). To obtain the coefficients in Equation (9) and (10), we simulate a single approximate adder with different operating conditions, which we model by changing the input distributions (Gaussian distributions with zero mean and different standard deviations), and applying artificial errors. The artificial errors are also assumed to have Gaussian distributions with zero mean and different standard deviations. Using (i) the EM_A, EM_B, and EM_Z measured from the simulation, as well as (ii) the EM_{in} values obtained from lookup tables, we generate the regression coefficients using the linear regression toolbox in Matlab [20].

After obtaining the regression coefficients, we apply them in EM estimations and report the two inaccuracy metrics of EM results. We observe from Table II that the regression coefficients help improve absolute inaccuracy of estimated ER, ES, ARES and SNR. However, the absolute inaccuracy slightly increases for MSE, and increases over 50% for MAXE. One possible reason could be that the data points that dominate MAXE are outliers for linear regression. Compared to absolute inaccuracy, the benefit of regression for estimation inaccuracy degrades for ES, MSE, and MAXE. This is because the linear regression applied to Equation (10) implies minimizing $|R_c - R_e|$, and $|R_c|$ in the denominator of Equation (11) is not considered. Improving the regression model to address both inaccuracy metrics mentioned above is one of our ongoing works.

TABLE II. (A) REGRESSION COEFFICIENTS DERIVED WITH DIFFERENT HARDWARE CONFIGURATIONS, (B) ESTIMATION INACCURACY WITH AND WITHOUT REGRESSION, AND (C) ABSOLUTE ESTIMATION INACCURACY WITH AND WITHOUT REGRESSION.

Regression Parameters						
	ER	ES	ARES	MSE	SNR	MAXE
α_{IN}	1.03E+00	1.00E+00	2.42E-02	1.00E+00	3.46E-01	9.40E-01
α_P	1.26E+00	9.98E-01	9.76E-01	1.00E+00	7.15E-02	7.98E-01
α_C	-5.85E-03	5.74E-08	-5.92E-03	-5.55E-09	-1.27E+00	8.65E-05
Estimation Inaccuracy						
w/o Reg.	4.15E-02	7.77E-02	8.38E+02	1.08E-01	1.35E+02	1.28E-01
with Reg.	7.40E-03	5.55E-01	2.09E+02	4.44E+04	4.04E-01	1.88E+01
Absolute Inaccuracy						
w/o Reg.	4.01E-02	2.90E-05	1.09E+01	2.24E-07	2.96E+03	9.37E-04
with Reg.	7.17E-03	2.71E-05	1.46E-01	2.90E-07	1.31E+01	1.52E-03

IV. EXPERIMENTS AND RESULTS

To evaluate the accuracy and performance of our EM estimation approach, we perform several experiments. (For pessimistic evaluation, we use estimation inaccuracy in Equation (11) in this section unless otherwise specified.) First, we demonstrate that our approach can be applied to a four-tap finite impulse response (FIR) filter. In the FIR experiment, the accuracies of six error metrics are evaluated. Second, we use *multiply-accumulator* (MAC) circuits with different sizes to compare the accuracy and runtime between our approach and the interval-based approach. Finally, we evaluate the accuracy of estimated results for randomly-generated topologies. In the experiments, we use 64-bit ETAIIMs with different k parameters. The adders are assumed to have 60 fractional bits.

FIR filter. To demonstrate that our approach is applicable to realistic computation circuits, we estimate EMs for the FIR filter design illustrated in Figure 9(a). Lookup table characterization for each error metric and standard deviation is performed for 12×12 different combinations of standard deviations (2^0, 2^{-2},... 2^{-22}). For each entry in the tables, we use 90K samples to obtain standard deviations and EMs. The runtime for building this set of lookup tables is 1.37 hours on a 2.8GHz Intel Xeon E5-2640 Linux workstation with 128GB of memory. With our lookup tables, we implement the flow in Figure 6 with Matlab [20].

Table III shows inaccuracy results of the estimations for each EM. We assume that the constant multipliers are accurate, and the adders

in the FIR filter are approximate modules. In the second column (error type), "IN" means an intrinsic EM value generated by the approximate modules themselves, and "P" means a propagated EM value composed from the EMs in previous stages. Based on the results in Table III, our approach provides accurate EM estimations for ER, ES, MSE and MAXE metrics. For the same testcase, the inaccuracies of the interval-based approach are 17.6% and 60.2% for ER and ES, respectively.

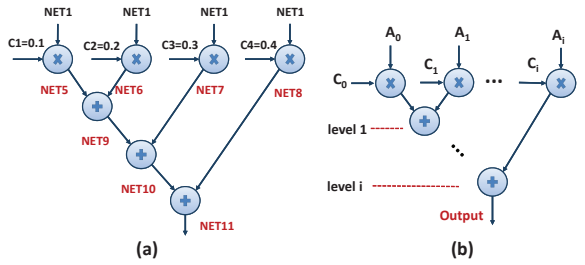


Fig. 9. Configuration of (a) FIR filter and (b) multiply-accumulator (MAC) circuits used in the experiments.

TABLE III. ESTIMATION INACCURACY OF A FOUR-TAP FIR FILTER SHOWN IN FIGURE 9(A).

		Estimation Inaccuracy					
Net	Type	ER	ES	ARES	MSE	SNR	MAXE
NET9	IN	0.3%	6.4%	17.0%	6.4%	19.1%	0.0%
NET10	IN	1.3%	2.6%	61.9%	3.3%	10.7%	0.0%
NET11	IN	1.0%	6.3%	419.6%	6.2%	6.1%	0.0%
NET11	P	13.4%	5.8%	692.3%	5.8%	436.4%	0.7%

MAC circuits. We test the accuracy and runtime of our approach against the interval-based approach for the MAC circuits shown in Figure 9(b), which are the general case of the FIR filter. We use 280 MAC circuits, having 14 different levels and 20 different configurations (parameters of each adder, constant values C_i, and input distributions).[1] We estimate EMs for the MAC circuits using our approach and the interval-based approach. Figures 10 and 11 show correlation plots for ER and ES, respectively. For ER, we observe that our approach achieves 1.28× better accuracy than the interval-based approach with 8.4× faster runtime. For ES, we observe that the estimated results from the interval-based approach are clamped to -2^{-20} on the right end. This is due to the saturation issue mentioned in Section III-A. For the same testcases, our approach is not affected by the saturation problem because the estimates of ES are interpolated or extrapolated from the lookup tables.

We evaluate runtime and accuracy for increasing circuit complexity by increasing the number of circuit levels in Figure 9(b). Figure 12(a) shows how runtime scales with circuit complexity. We observe that the runtime of error composition increases linearly for both our approach and the interval-based approach. Our approach is 8.4× faster than the interval-based approach. Figure 12(b) shows inaccuracy results. Our approach demonstrates improved accuracy compared to the interval-based approach, especially for the ES metric. The inaccuracy is reduced by 3.75× compared to the interval-based approach excluding saturation.[2]

Randomly generated topologies. To study the accuracy of EM estimation with respect to the size and topology of testcases, we use randomly generated testcases as in [11]. We use the following three components to generate the random testcases; (1) primary inputs (PI) with different standard deviations, (2) adders with different hardware configurations, and (3) arbitrary connections among adders and constant multipliers. We generate 50 artificial testcases with different numbers of nodes (adders or constant multipliers). The number of nodes ranges from 10 to 30 with a step size of five. The accuracy results for each EM are plotted in Figure 13. We evaluate the estimated results from our approach with the regression coefficients generated from the model in Section III-B. In the plot, inaccuracy results from 10 different topologies are averaged for each circuit size.

For randomly generated circuits, we observe that ER, ES, MSE and MAXE show relatively accurate results with 4.18%, 8.30%, 12.2% and 12.9% inaccuracy, respectively. Moreover, the accuracy does not degrade as circuit complexity (number of nodes) increases. The estimates of ARES and SNR are inaccurate (1.28×10^3 and 1.35×10^2). Inaccuracy in these metrics arises because they measure error relative to input data, and accurate estimation is difficult, as we have discussed in Section III-C. Methods that would accurately handle their composition are obvious directions for our future work.

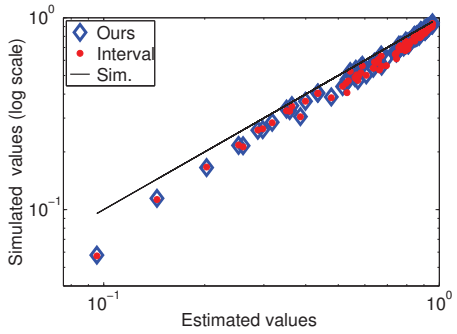


Fig. 10. Comparison of ER metrics between our approach and the interval-based approach.

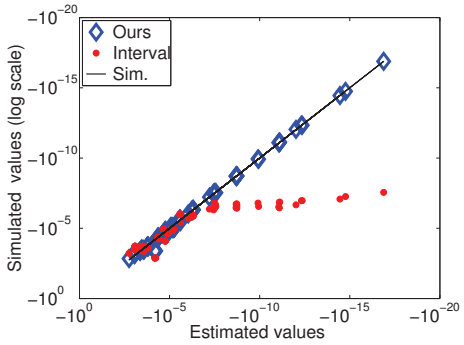


Fig. 11. Comparison of ES metrics between our approach and the interval-based approach. The inaccuracy on the right side is (2×10^9).

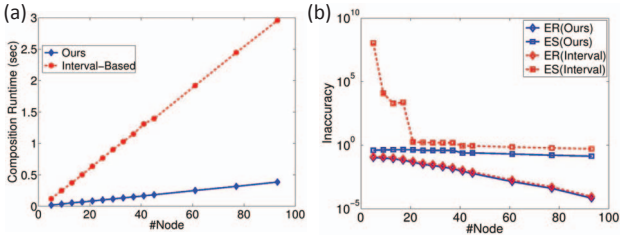


Fig. 12. Comparison of (a) runtime for error composition and (b) inaccuracy of EM estimation for the MAC circuits with different testcase sizes. Our average inaccuracy improvement against the interval-based approach is 3.75× excluding saturation.

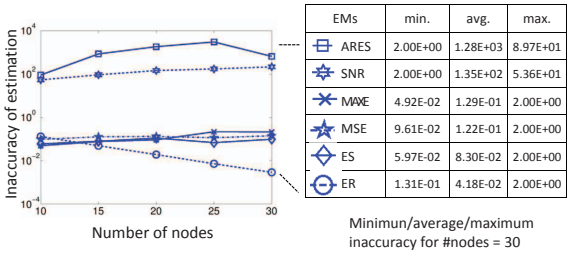
[1]Note that the multipliers are assumed to be accurate, so they only change the distribution of data but do not increase the number of errors.

[2]Note that when the number of nodes is small (the left side of the figure), the magnitude of estimation errors tends to be large relative to the magnitude of data, and the inaccuracy of the interval-based approach is very high due to the saturation issue.

V. Conclusions

We propose an approach for output quality estimation of approximate designs. Our LUT-based approach characterizes the statistical properties of approximate hardware modules and a regression-based technique improves the accuracy of EM estimation. With our composition approach, we achieve 1.36× and 8.4× runtime improvements for library characterization and error composition, respectively. We also achieve 3.75× accuracy improvement for ES compared to [9] [10] on a set of MAC circuits. We also demonstrate that our approach is applicable to general designs using the randomly generated testcases with up to 30 nodes in the configuration.

Our ongoing work seeks to improve the accuracy of EM estimation for relative error metrics (e.g., ARES and SNR). We will also extend our approach to other approximate modules, including multipliers. In addition, we are working to develop a synthesis flow for approximate circuits using our EM estimation approach. We further anticipate broadening our current studies to include more approximate arithmetic units and different input distributions. Currently, we assume that the input distributions are given; however, different applications have different distributions. Our follow-on work will seek (1) approaches that track the change of input distributions and reconfigure the hardware during runtime, in order to adapt the distributions such that error metric requirements are maintained; and (2) approaches that construct error metrics by decomposing arbitrary distributions into combinations of some basis distributions.



EMs	min.	avg.	max.
ARES	2.00E+00	1.28E+03	8.97E+01
SNR	2.00E+00	1.35E+02	5.36E+01
MAE	4.92E-02	1.29E-01	2.00E+00
MSE	9.61E-02	1.22E-01	2.00E+00
ES	5.97E-02	8.30E-02	2.00E+00
ER	1.31E-01	4.18E-02	2.00E+00

Fig. 13. Comparison of inaccuracy with respect to the number of nodes in randomly generated circuits.

Acknowledgments

We thank Dr. Jiawei Huang, Professor John Lach, and Professor Gabriel Robins for providing the source code and scripts used in reference [10].

References

[1] M. L. Chang and S. Hauck, "Precis: A Design-Time Precision Analysis Tool", *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002, pp. 229-238.

[2] M. L. Chang and S. Hauck, "Variable Precision Analysis for FPGA Synthesis", *Proc. NASA Earth Science Technology Conference*, 2003.

[3] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy and S. T. Chakradhar, "Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency", *Proc. DAC*, 2010, pp. 555-560.

[4] V. K. Chippa, A. Raghunathan, K. Roy and S. Chakradhar, "Dynamic Effort Scaling: Managing the Quality-Efficiency Tradeoff", *Proc. DAC*, 2011, pp. 603-608.

[5] I. S. Chong, H.-Y. Cheong and A. Ortega, "New Quality Metrics for Multimedia Compression Using Faulty Hardware", *Proc. International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006.

[6] C. F. Fang, R. A. Rutenbar and T. Chen, "Effcient Static Analysis of Fixed-Point Error in DSP Applications via Affine Arithmetic Modeling", *Proc. ICCAD*, 2003, pp. 275-282.

[7] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan and K. Roy, "IMPACT: Imprecise Adders for Low-Power Approximate Computing", *Proc. ISLPED*, 2011, pp. 409-414.

[8] J. Huang and J. Lach, "Exploring the Fidelity-Efficiency Design Space Using Imprecise Arithmetic", *Proc. ASP-DAC*, 2011, pp. 579-584.

[9] J. Huang, J. Lach and G. Robins, "Analytic Error Modeling for Imprecise Arithmetic Circuits", *Proc. SELSE*, 2011.

[10] J. Huang, J. Lach and G. Robins, "A Methodology for Energy-Quality Tradeoffs Using Imprecise Hardware", *Proc. DAC*, 2012, pp. 504-509.

[11] A. B. Kahng and S. Kang, "Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions", *Proc. ISPD*, 2012, pp. 153-160.

[12] A. B. Kahng and S. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs", *Proc. DAC*, 2012, pp. 820-825.

[13] Z. Kedem, V. J. Mooney, K. K. Muntimadugu, K. V. Palem, A. Devarasetty and P. D. Parasuramuni, "Optimizing Energy to Minimize Errors in Dataflow Graphs Using Approximate Adders", *Proc. CASES*, 2010, pp. 177-186.

[14] A. B. Kinsman and N. Nicolici, "Finite Precision Bit-width Allocation Using SAT-Modulo Theory", *Proc. DATE*, 2009, pp. 1106-1111.

[15] D. Koes and T. Chelcea, "Adding Faster With Application Specific Early Termination", *CMU Research Showcase*, 2005.

[16] P. Kulkarni, P. Gupta and M. D. Ercegovac, "Trading Accuracy for Power With an Underdesigned Multiplier Architecture", *Proc. International Conference on VLSI Design*, 2011, pp. 346-351.

[17] K. Kum and W. Sung, "Combined Word-length Optimization and High-level Synthesis of Digital Signal Processing Systems", *IEEE TCAD* 20(8) (2001), pp. 921-930.

[18] D.-U. Lee, A. A. Gaffar, O. Mencer and W. Luk, "MiniBit: Bitwidth Optimization via Affine Arithmetic", *Proc. DAC*, 2005, pp. 837-840.

[19] S.-L. Lu, "Speeding Up Processing With Approximation Circuits", *IEEE Computer* 37(3) (2004), pp. 67-73.

[20] *Mathworks MATLAB*, http://www.mathworks.com/products/matlab/

[21] J. Miao, K. He, A. Gerstlauer and M. Orshansky, "Modeling and Synthesis of Quality-Energy Optimal Approximate Adders", *Proc. ICCAD*, 2012, pp. 728–735.

[22] A. Nayak, M. Haldar, A. Choudhary and P. Banerjee, "Precision and Error Analysis of MATLAB Applications during Automated Hardware Synthesis for FPGAs", *Proc. DATE*, 2001, pp. 722-728.

[23] D. Shin and S. K. Gupta, "Approximate Logic Synthesis for Error Tolerant Applications", *Proc. DATE*, 2010, pp. 957-960.

[24] M. Stephenson, J. Babb and S. Amarasinghe, "Bitwidth Analysis With Application to Silicon Compilation", *Proc. PLDI*, 2000, pp. 108-120.

[25] R. Venkatesan, A. Agarwal, K. Roy and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing", *Proc. ICCAD*, 2011, pp. 667-673.

[26] A. K. Verma, P. B. and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design", *Proc. DATE*, 2009, pp. 1250-1255.

[27] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "SALSA: Systematic Logic Synthesis of Approximate Circuits", *Proc. DAC*, 2012, pp. 796-801.

[28] Z. Wang and A. C. Bovik, "Mean Squared Error: Love it or Leave it? A New Look at Signal Fidelity Measures", *IEEE Signal Processing Magazine* 26(1) (2009), pp. 98-117.

[29] N. Zhu, W. L. Goh, W. 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", *IEEE TVLSI* 18(8) (2010), pp. 1225-1229.

[30] D. R. Kelly, B. J. Phillipsy and S. Al-Sarawi, "Approximate Signed Binary Integer Multipliers for Arithmetic Data Value Speculation", *Proc. DASIP*, 2009, pp. 97-104.