b-HiVE: A Bit-Level History-Based Error Model with Value Correlation for Voltage-Scaled Integer and Floating Point Units

G. Tziantzioulis[†], A. M. Gok[†], S M Faisal[‡], N. Hardavellas[†], S. Ogrenci-Memik[†], and S. Parthasarathy[‡]

[†]Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL, USA
{getziadz, amg}@u.northwestern.edu
{nikos, seda}@northwestern.edu

[‡]Department of Computer Science
The Ohio State University, Columbus, OH, USA
{faisal, srini}@cse.ohio-state.edu

ABSTRACT

Existing timing error models for voltage-scaled functional units ignore the effect of history and correlation among outputs, and the variation in the error behavior at different bit locations. We propose b-HiVE, a model for voltage-scaling-induced timing errors that incorporates these attributes and demonstrates their impact on the overall model accuracy. On average across several operations, b-HiVE's estimation is within 1-3% of comprehensive analog simulations, which corresponds to 5-17x higher accuracy (6-10x on average) than error models currently used in approximate computing research. To the best of our knowledge, we present the first bit-level error models of arithmetic units, and the first error models for voltage scaling of bitwise logic operations and floating-point units.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development–*Modeling Methodologies;* B.8.1 [**Hardware**]: Performance and Reliability–*Reliability, Testing, and Fault-Tolerance*.

General Terms

Measurement, Reliability, Experimentation.

Keywords

Approximate Computing, Voltage Scaling, Error Modeling.

1. INTRODUCTION

Power has become a first-class design constraint [12] for both high-end and mobile systems due to the breakdown of Dennard's scaling and the advent of dark silicon [7]. Several techniques to reduce power have been proposed, including dynamic voltage and frequency scaling, near-threshold computing, sleep states, and specialized cores (accelerators). However, the impact of these techniques is hampered by the traditional worst-case design of systems. Advanced technology scaling leads to a probabilistic behavior in CMOS [1], forcing designers to add significant voltage margins and timing slack [13] to overcome the environmental fluctuations and design uncertainties. The resulting conservative guard-bands reduce both performance and power efficiency.

DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

ACM 978-1-4503-3520-1/15/06 $15.00
http://dx.doi.org/10.1145/2744769.2744805

In an attempt to improve power efficiency, recent research proposed techniques for error-tolerant execution [2,3,5,8,10,11,16,17] that detect and correct the occasional circuit errors, but prohibit the aggressive voltage scaling of functional units (FUs). Lately the research community took a step further and embraced approximate computing by allowing faults in execution while providing acceptable output quality by limiting the errors only to computations that can tolerate them [4,14,15]. These techniques allow the voltage to scale well beyond what error-tolerant techniques can afford.

Approximate computing via voltage scaling, at both the architectural and component design levels, requires thorough assessment with an accurate error model to explore the accuracy-power tradeoff. An incorrect error model could lead to the discarding of potentially useful tools and aversion from a fruitful path on improving the power efficiency of future computer systems. Prior works have used a variety of error models to correlate the energy savings to output quality of applications, from single bit-flip probabilities to uniform distribution models to random values [4,14,15].

However, these error models are not able to fully capture the error behavior of FUs. Current error models do not cover important families of FUs such as those that perform floating point and bitwise logic operations (e.g., xor), which behave drastically different than integer adders. Moreover, they ignore the impact of **computation history:** prior operations on a FU affect the result of the current computation, as switching a signal is typically harder than keeping it constant. Existing models also ignore the impact of **value correlation:** the values of consecutive operations in a real application are often similar, thus missing timing does not necessarily mean that the output is incorrect. Finally, current models uniformly apply a single error probability to all bits, ignoring the variability in error behavior across **bit locations**, which is often significant.

In this paper we address these shortcomings and propose b-HiVE, a novel error model for voltage-scaling-induced timing errors in FUs that incorporates computation history, value correlation, and awareness of bit location. b-HiVE attains at least 5x higher accuracy than current error models for integer operations and 6-10x higher accuracy on average across all modeled operations, while its predictions are within 1-3% of comprehensive analog simulations. We target the largest variety of components studied to date, and present the first error models for floating point and bitwise logic functional units. More specifically, our contributions are:

- We present a detailed characterization of the bit-level error behavior of industrial-strength integer, bitwise logic, and floating-point units across operation types and voltage levels.
- We analyze the impact of computation history on the error behavior of arithmetic units at reduced voltage and show that including the history of the immediately preceding operation (History-1) is necessary and sufficient to obtain high accuracy.

- We quantify the impact of value correlation in the error behavior of arithmetic operations.
- We derive robust error models using both machine learning and probabilistic techniques that accurately capture the error behavior of these FU designs on real applications.

2. RELATED WORK

There is a large body of work in detecting and correcting the timing errors that stem from voltage-overscaling or overclocking FUs. Razor [2,3] is a register transfer level technique that allows operation in sub-nominal voltage regimes and provides a mechanism for detecting and correcting timing errors in the pipeline. Ernst *et al.* [3] collected timing violation statistics with measurements from voltage-scaled FUs synthesized in FPGAs, a fabricated in-order processor, and SPICE-level analysis. Subsequently, Krimer *et al.* [10] adapted the Razor technology to GPUs and utilized this data to construct an error model for integer adders and multipliers based on an exponential relationship between the supply voltage scaling factor and the observed timing behavior in their data. In contrast to b-HiVE, this model ignores the impact of bit location and computation history, and only covers integer add and multiply operations.

Recently-proposed CAD techniques provide graceful degradation of modules under voltage-overscaling or overclocking, combined with correction for the occasional timing errors [5,11,15,16]. It has been shown that some instructions are more likely than others to stress the circuit's critical paths and produce timing errors, which gave rise to techniques that allocate extra cycles to critical instructions or substitute critical instructions with less critical ones [8,17]. b-HiVE can be used in that line of research to provide a high accuracy bit-level error model of a multitude of arithmetic units, coupled with the accurate error behavior of real applications.

More recent research has embraced approximate computing, where errors are allowed to occur at certain locations in a program, enabling even more aggressive voltage scaling. A programming framework and architecture were recently proposed [4,14] for marking individual instructions as imprecise and executing them in aggressively voltage-scaled FUs. Work in this space has relied on basic fault injection error models that perform single bit flips at the output with uniform distribution [14], or select random or previously seen values for the entire output [14], resulting in highly inaccurate error predictions. Subsequent works [4] rely on uniform bit-error models with per-component probabilities, but these models are not reproducible as no details are publicly available. Unlike these models, b-HiVE accounts for variability of behavior among bit locations and the past history of activity for an output bit, and also, for the first time, provides an error model of FUs executing logic and floating-point operations.

The impact of correlation between consecutive values in the behavior of an error-tolerant design has been considered with a limited study focusing on the carry propagation of integer adders [9]. Our work performs the first detailed study of this phenomenon considering the overall operation of several complete FUs, with SPICE-level analog simulations. In addition, b-HiVE captures the variation in error rate between different bits of the same operation at each voltage level. Unlike previous error models that represent the error rate for a given voltage with a scalar value for the entire FU output word, our findings suggest that a per-bit error rate is required for accurate modeling.

3. MODELING FRAMEWORK OVERVIEW

Our modeling framework is comprised of three phases: Characterization, Data Classification, and Model Generation. In the first phase we collect detailed information about the behavior of each component under supply voltage scaling. We extract and synthesize the full circuit blocks of each component and conduct detailed analog simulations in a range of supply voltages using both random-generated and real workload traces. The Data Classification phase entails pre-processing of the simulation data to expose information that represents the vital novel attributes of our model: bit-location awareness, computation history, and value correlation. Specifically, we classify output bits into five distinct classes that represent the expected states of the sampled value of the register at the output of a module. With these five classes we capture the behavior of each bit in the output of a functional unit when operating at a scaled voltage. Ultimately, in the Model Generation phase we develop error models that provide the probability with which each output bit is expected to belong to one of these five classes under a given operating condition, i.e., supply voltage level, operation type and bit location.

4. METHODOLOGY

4.1 Functional Unit Characterization

For our characterization we fully synthesize the integer and floating point FUs of the OpenSPARC T1 core using the Synopsys Design Compiler and the SAED90nm standard cell library. All modules are 64-bit, consistent with the OpenSPARC T1 datapath. We perform analog simulations of SPICE-level models of the standard cells in the netlists using the Synopsys VCS and HSIMplus co-simulation tools. We simulate the modules after generating the physical layouts, including clock tree synthesis when necessary using the Synopsys IC Compiler and VCS/HSIMplus (which we verified they provide faster simulation without loss of accuracy).

Due to the extremely high number of possible input combinations and the excessive time requirements of analog simulations, we perform the simulations using semi-randomly generated inputs. Assuming a FU that accepts two 64-bit operands, we randomly generate a trace of N operand pairs by forming a 2D plane with the x- and y-axis in the range $[0, 2^{64}-1]$. We then tile the plane into N tiles by dividing each dimension into $\sqrt{N}$ intervals. We randomly choose one point within each tile, and consider its x-y coordinates as the values of the two input operands of the FU. This method results in a homogenous distribution over the 2D input space, guarantees that we cover the entire range of possible input operand values, and allows for value randomness, thus enables us to capture a significant range of the operation behavior through the simulations.

When scaling the voltage of a FU its output is a function of the current inputs and also the preceding operations' inputs and outputs (i.e., the computation history). This effectively means that we cannot generate an accurate model with a trace of randomly generated input operands, as this would represent a single "history" instance. To solve this problem, after generating a trace using tiling, we create a set of different permutations of all these inputs to emulate different history instances. Using this method we create 5-10 permutations of traces ranging from 2,025 to 10,000 operations, and use them to feed transistor-level simulations of the components we evaluate. To keep the simulation time within practical limits (a few days per trace per component) we use the shorter traces to simulate highly-complex components, such as the floating-point units.

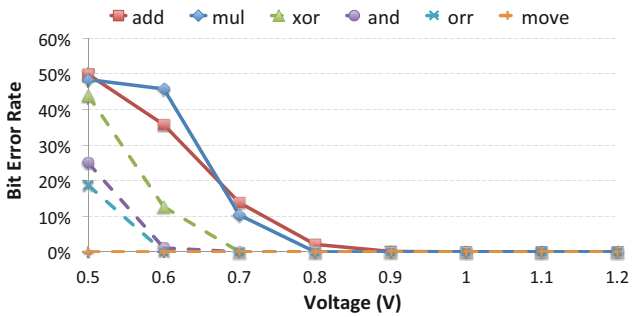

FIGURE 1. Bit error rate for integer ops (arithmetic: solid lines; bitwise: dashed) in analog simulation of a random trace.

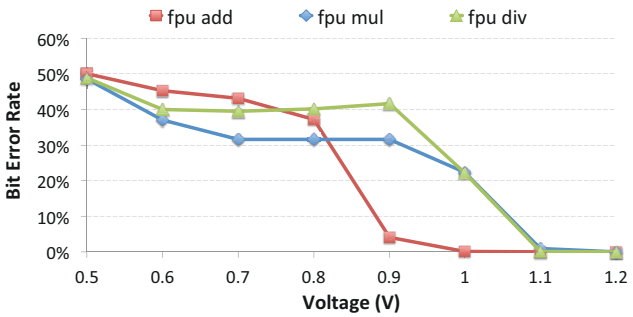

FIGURE 2. Bit error rate for floating point arithmetic operations in the analog simulation of a random trace.

For each component we determine the minimum possible clock period that results in no errors at the nominal supply voltage (V_{dd} = 1.2V) without considering process variation and wear-out. Then, keeping the clock cycle time constant, we simulate each component on a range of supply voltage levels from 0.5V to 1.2V (in increments of 0.1V) using the same trace for each voltage level.

4.2 Data Classification

After collecting the analog simulation data, we classify the behavior of each FU output bit. For each FU output bit, in addition to the result computed for a given operation, all results of the operations simulated prior to it are available in the simulation data. Our classification process exposes the history of past operations at each bit location in this data. For each bit location in the output of a FU, the values of the immediately preceding and current operations could be described with four 1-bit descriptors: Past Observed (PO) corresponds to the value of the bit for the preceding operation observed through the analog simulation, Past Correct (PC) corresponds to the expected (correctly computed) value of that bit, and similarly Current Observed (CO) and Current Correct (CC) correspond to the observed and correct bit values for the current operation.

These four 1-bit descriptors <PC, PO, CC, CO> form 2^4=16 distinct cases that correspond to all possible value combinations, e.g., case 3 is PC=0, PO=0, CC=1, CO=1. Each case ties the circuit's behavior on the current operation to the history of the immediately preceding operation (we refer to this as **History-1**). Our experiments indicate that incorporating History-1 is necessary and sufficient for an accurate error model, and a longer history increases the distinct events required for classification without providing significantly higher accuracy. Ultimately, we classify these 16 cases under five main classes that capture the response of a given circuit to voltage scaling at a given bit location (see Table 1):

- The **correct class** captures the instances where the circuit meets timing and a correctly computed value is observed.
- The **previous observed class** captures instances where the circuit missed timing and the latched value matches the observed outcome of the immediately preceding operation.
- The **previous correct class** is similar to the previous observed class, but the latched value matches the correct value of the preceding operation.
- The **glitch class** captures the instances where a transition at the output bit is unnecessary (the previous and the current operation's correct bit values, and the previous observed bit value are all the same), however a transition occurs and it is latched.
- Finally, the **ambiguous class** captures the instances where the output is correct, but there is no way to know whether this is the result of a missed timing or correct operation (e.g. in case 0 we cannot distinguish between the circuit meeting timing vs. observing the previous observed or the previous correct value).

To compute the probability of each class for a specific bit we use the data from the classification process of the random traces. In Section 5 we demonstrate that constructing the model using data from the random traces is sufficiently accurate, while performing per-application training provides negligible additional improvements. This makes our model independent of the application it will be applied on, i.e., after constructing a model for a specific module no additional training is required for new applications.

4.3 Error Model Generation

We construct error models using two alternatives: decision trees, and the direct derivation of the frequency of occurrence of each case in Table 1 with the resulting error probability (b-HiVE).

4.3.1 Machine-Learning-Based Modeling

We derive error models by applying various machine learning techniques from the Weka [6] library of statistical tools (C4.5 Decision Trees, Logistic Regression, and Naïve Bayes classification) on the data collected from analog simulations of random traces. As the accuracy was similar across the resulting models, we pick the decision tree models due to their superior interpretability.

For the decision tree algorithms, the dataset is split 80:20, where 80% of the data is used for training and 20% for testing. In order to prepare the training data for modeling purposes, we perform a data selection step that removes data that are part of the ambiguous class (Section 4.2) since we cannot differentiate between correct operation and timing error. After the selection, our training sets for modeling purposes contain data related to the circuit level behavior of the voltage-scaled FUs. Specifically, our training set contains the operation type, bit position, voltage level and class information. We exclude operand values from the training dataset as the resulting decision trees become very deep with highly complex branching, but provide only negligible increase in accuracy.

TABLE 1. Data Classification.

Class	Timing Error	Included Cases
Correct	No	3, 4, 11, 12
Previous Observed	Yes	2, 5, 10, 12
Previous Correct	Yes	6, 9
Glitch	Yes	1, 14
Ambiguous	Unknown	0, 7, 8, 15

The decision tree algorithm builds the tree based on normalized information gain (change in entropy). At each point, the algorithm tries to reduce entropy in the resulting branches. To control the shape of the tree we modify a parameter that determines the minimum number of instances per leaf node of the tree. We see that varying this parameter does not affect the quality of our models, and, hence, we keep a single value across all experiments.

We refer to our decision-tree-based model including History-1 information as History-1-DecTree (**H1DT**). We also evaluate an alternative, NoHistory-DecTree (**NHDT**), which is constructed without taking into account how the history of consecutive computations affects the outcome of the more recent one. We present a comparison of the two models in Section 5. Our results show that including History-1 in the model enhances its accuracy significantly, supporting our claim that history correlation plays an important role in the construction of accurate error models. Including a deeper history improves the accuracy by at most 2.7%, hence we do not consider history deeper than 1.

4.3.2 Frequency-of-Occurrence-Based Modeling

In addition to the decision trees, we consider the alternative method of deriving the error model directly from the frequency of occurrence of each case in the analog simulation data of the random trace. Similar to the H1DT and NHDT models, we evaluate a model that excludes the ambiguous cases (Table 1) from the dataset (**b-HiVE** is the resulting model), and a variation of the model that classifies the ambiguous cases as correct ones (**b-HiVE-NE**).

Furthermore, we create a third variation, **b-HiVE-<app>Trained**, which is a b-HiVE model trained on the analog simulation data of a trace extracted from a real application workload. Ultimately, for any model used, the behavior of each bit location in the output of a FU is represented by the probability of four possible outcomes, which are determined by the operation, bit location, and voltage level: $P_{correct} + P_{prevCorrect} + P_{prevObserved} + P_{glitch} = 1$.

5. ANALYSIS

In this section we present the results of the functional unit characterization and the analysis of the different models. We also demonstrate the effectiveness of b-HiVE in comparison to alternative error models from existing literature.

5.1 Hardware Characterization

Using the fully synthesized hardware modules and the random input traces created through the process described in Section 4.1 we perform a series of simulations to characterize the behavior of each component at scaled supply voltage. Figure 1 presents the bit error rate (the percentage of incorrect output bits over the total

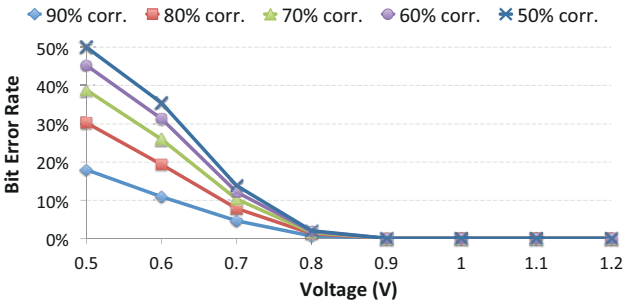

FIGURE 4. Bit error rate for 64-bit integer addition in the analog simulation of a random trace with value correlation.

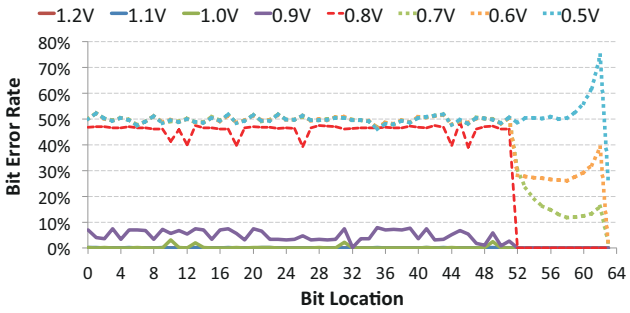

FIGURE 3. Per-bit error rate for double-precision floating point addition in the analog simulation of a random trace.

number of output bits in the simulation data) for the integer ALU (add, move, and, or, xor) and the integer multiplier. Similarly, Figure 2 presents the bit error rates for the floating point (FP) adder, multiplier and divider.

The nominal supply voltage (V_{dd}) for all components is 1.2V. We do not present the results of simulations for the [0.1-0.4V] range as we observe a complete breakdown of the circuit's behavior. In both Figures 1 and 2 it is clearly seen that different operations exhibit vastly different error behavior. Hence, all prior modeling work that treats all operations under the same lumped error rate will suffer from inaccuracies. Logic and move operations, in particular, exhibit a significantly smaller error rate than addition and multiplication (e.g., move operations have practically no errors even for 0.5V). This is expected, as the critical paths excited for these operations are simpler, reducing the chances of timing errors.

Figure 3 illustrates the error behavior of each of the 64 individual output bits of the FP adder. We observe that the exponent's bits are more resilient than the mantissa's even at low voltages. This is an interesting finding as it indicates that even though the exact number may be corrupted, the magnitude (exponent value) of the operation will likely remain correct down to 0.8V, and exhibit smaller variation than the mantissa at 0.7-0.6V. Such behavior could be exploited through hardware/software co-design to improve the fault tolerance of applications.

To evaluate the effect of value correlation on the error rate, we generate controlled input vector traces for two integer ALU operations (ADD and OR) that produce a specific correlation rate between outputs. For example, 80% value correlation means that the correct results of two consecutive operations have on average 80% of their bits be identical, while the remaining 20% differ. Our random trace generation algorithm produces an 80% value correlation by flipping an 80/20 biased coin for each output bit to decide if the bit remains the same or switches value, selects operand inputs that

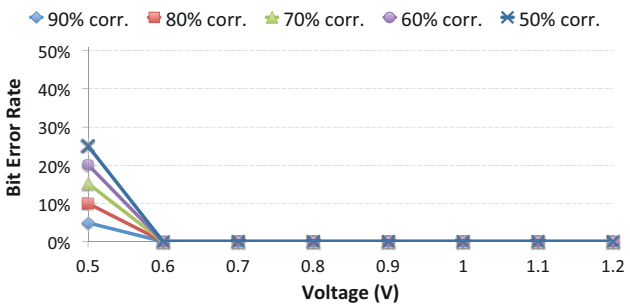

FIGURE 5. Bit error rate for 64-bit bitwise OR in the analog simulation of a random trace with value correlation.

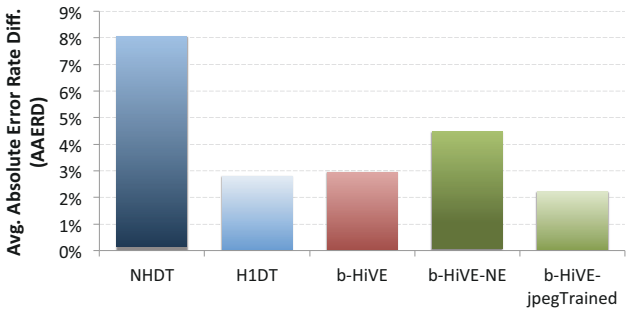

FIGURE 6. Average absolute bit error rate difference over the analog simulation of the JPEG decoder trace.

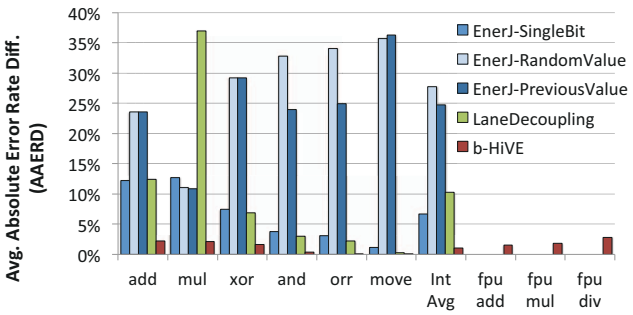

FIGURE 7. Average absolute bit error rate difference over the analog simulation of a random trace.

would produce that output had the operation been correct, and then includes this operation in the trace and repeats the process. This trace is then simulated in our analog simulation infrastructure, and we observe the output bits of each operation to calculate the bit error rate. Figures 4 and 5 present the results of these simulations for a range of value correlation levels, and clearly demonstrate that higher value correlation leads to significantly smaller error rate. This justifies and highlights the importance of incorporating value correlation into the error modeling of functional units.

5.2 Evaluation of b-HiVE

We evaluate five alternative error models that we developed: NHDT, H1DT, b-HiVE, b-HiVE-NE, and b-HiVE-<app>Trained (Section 4.3). We evaluate these models on a trace from a JPEG image decoding kernel. We define the quality of each model using the average absolute error rate difference metric (AAERD), calculated using the formula below, where $W_{X,V,i}$ is the value (0 or 1) of bit i for model X at voltage V, and S is the analog simulation:

$$AAERD_X = \sum_{V} \frac{\sum_{\text{bit i}} \frac{\left| \sum_{\text{trace ops}} \frac{|W_{X,V,i} - W_{S,1.2,i}|}{\|\#\text{ops in trace}\|} - \sum_{\text{trace ops}} \frac{|W_{S,V,i} - W_{S,1.2,i}|}{\|\#\text{ops in trace}\|} \right|}{\|\#\text{bit positions}\|}}{\|\#\text{voltage levels}\|}$$

Figure 6 presents a comparison of the five models for an execution trace of the JPEG decoder kernel. NHDT achieves an AAERD of 8%. By taking history correlation into account, H1DT's accuracy improves by 3x and its AAERD drops to 2.7%. Thus, history correlation is an important factor in error modeling. A designer's interpretation of a 2.7% AAERD is that the impact of incorporating a longer history into the model, or even the ideal case of infinite history (analog simulation), could improve accuracy by only 2.7%. Thus, History-1 provides sufficient accuracy and need not be extended. b-HiVE is an error model of almost identical accuracy to

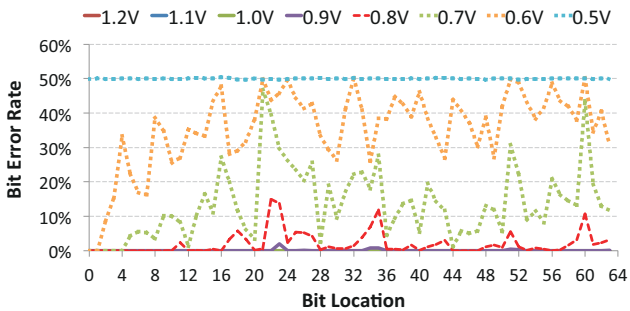

FIGURE 8. Per-bit error rate of 64-bit integer addition in the analog simulation of a random trace.

the decision tree but significantly simpler, thus complex machine learning techniques are not justifiable. We also evaluate b-HiVE-NE, a model trained on the full raw data by including the ambiguous cases as correct. b-HiVE-NE exhibits 52% lower accuracy than b-HiVE, emphasizing the importance of detecting and excluding the ambiguous cases. Finally, b-HiVE-jpegTrained, derived by training b-HiVE directly on the JPEG trace, improves accuracy by just 0.7%, supporting our argument that a generic FU-based model trained on random traces can work well for real-world applications.

5.3 Comparison to Previous Error Models

As the majority of previous work focuses on integer addition and multiplication modules, there is no accurate model for bitwise logic or FP operations. The variability in error rates for different operations (Figures 1 and 2) clearly shows that using the model of one operation to predict the behavior of another could lead to significant inaccuracies. Figure 7 compares the accuracy between the analog simulation of random traces and the predictions of 5 models: b-HiVE, three error models from EnerJ [14], and the model used in Lane Decoupling [10] with its error rate uniformly distributed among the bits. For the bitwise logic operations that Lane Decoupling presents no model, we apply the integer add model as it exhibits the lowest error. We cannot offer a comparison with scaled FP units as it was not presented in any of the above works. However, we present the accuracy of b-HiVE as a reference point.

b-HiVE exhibits at least 5x higher accuracy than competing error models (EnerJ-SingleBit and LaneDecoupling for the xor operation are the closest any of these models ever gets to b-HiVE), and up to 17x lower accuracy for LaneDecoupling (integer mul) and 10x for EnerJ (integer add). The gap between these models and b-HiVE is even more pronounced for the logic operations. More importantly, b-HiVE is within only 1-3% of comprehensive analog simulations.

Another limitation of previous error models is the use of scalar values to characterize the behavior of all bits at a voltage level. Fig-

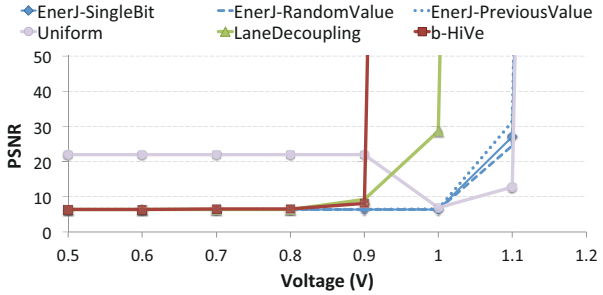

FIGURE 9. PSNR (higher is better) of the decoded JPEG image under various error models.

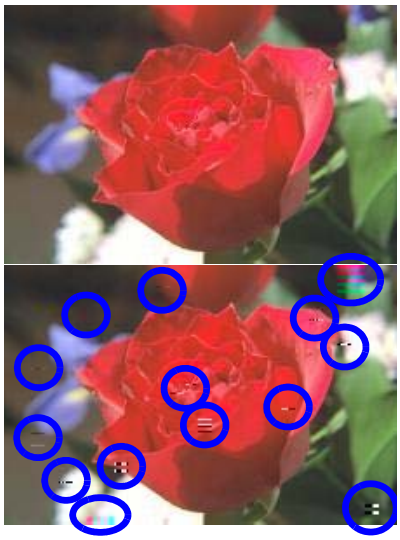

FIGURE 10. Decoded JPEG images of analog simulations.

ures 3 and 8 expose the problem of this approach, clearly showing that the error rate across bits can have significant fluctuations. Error models should capture this behavior to accurately evaluate design techniques that may impact the fidelity of a module.

5.4 Impact on Application Output Quality

The choice of error model has a measurable and visible impact on the quality of the final output of an application. Figure 9 presents the peak signal-to-noise ratio (PSNR) of the output of a JPEG image decoder when simulated using 6 error models across voltages. Figure 10 presents the different outputs of simulating the JPEG image decoder at 1.0V under the b-HiVE and Lane Decoupling [10] models, where we circle the additional artifacts that the Lane Decoupling model introduces. These figures demonstrate that previous models may overestimate or underestimate the error rate, resulting in mispredicted behavior.

6. CONCLUSIONS

Error models are at the core of the design of power efficient and reliable systems, hence their accuracy is of paramount importance. We show that awareness of bit location, history correlation and value correlation play an important role in building accurate error models, all of which have been largely ignored until now. We propose b-HiVE, a detailed error model for integer, bitwise logic, and FP units at bit-level granularity that incorporates history and value correlation, leading to substantially higher accuracy over typically used error models. b-HiVE is robust and application independent, hence it can be used with any application, and can be easily incorporated in architectural simulators to extract quality and power measurements orders of magnitude faster than full analog simulation at the cost of only 1-3% loss in accuracy.

7. ACKNOWLEDGMENTS

This work is supported by NSF CCF-1218768 and CCF-1217353.

8. REFERENCES

[1] A. Asenov, G. Slavcheva, A. Brown, J. Davies, and S. Saini. Increase in the random dopant induced threshold fluctuations and lowering in sub-100 nm MOSFETs due to quantum effects: a 3-D density-gradient simulation study. *IEEE Transactions on Electron Devices*, 48(4):722- 729, Apr 2001.

[2] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. A self-tuning DVS processor using delay-error detection and correction. *IEEE Journal of Solid-State Circuits*, 41(4):792- 804, 2006.

[3] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *36th Annual IEEE/ACM International Symposium on Microarchitecture*, 2003.

[4] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture support for disciplined approximate programming. In *17th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2012.

[5] B. Greskamp, L. Wan, U. Karpuzcu, J. Cook, J. Torrellas, D. Chen, and C. Zilles. Blueshift: Designing processors for timing speculation from the ground up. In *15th IEEE Int'l Symp. on High Performance Computer Architecture*, 2009.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10- 18, Nov. 2009.

[7] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6- 15, 2011.

[8] G. Hoang, R. B. Findler, and R. Joseph. Exploring circuit timing-aware language and compilation. In *16th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2011.

[9] S. H. Kim, S. Mukohopadhyay, and W. Wolf. Experimental analysis of sequence dependence on energy saving for error tolerant image processing. In *2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, 2009.

[10] E. Krimer, P. Chiang, and M. Erez. Lane decoupling for improving the timing-error resiliency of wide-simd architectures. In *39th Annual Int'l Symp. on Computer Architecture*, 2012.

[11] J. Miao, A. Gerstlauer, and M. Orshansky. Multi-level approximate logic synthesis under general error constraints. In *IEEE/ACM Int'l Conf. on Computer-Aided Design*, 2014.

[12] T. Mudge. Power: A first-class architectural design constraint. *Computer*, 34(4):52- 58, Apr. 2001.

[13] V. J. Reddi and M. S. Gupta. *Resilient architecture design for voltage variation*. Synthesis Lectures on Computer Architecture. Morgan and Claypool Publ., San Rafael, 2013.

[14] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. EnerJ: Approximate data types for safe and general low-power computation. In *32nd ACM Conf. on Programming Language Design and Implementation*, 2011

[15] J. Sartori, J. Sloan, and R. Kumar. Stochastic computing: Embracing errors in architecture and design of processors and applications. In *14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2011.

[16] A. Tiwari, S. R. Sarangi, and J. Torrellas. Recycle: Pipeline adaptation to tolerate process variation. In *34th Annual International Symposium on Computer Architecture*, 2007.

[17] J. Xin and R. Joseph. Identifying and predicting timing-critical instructions to boost timing speculation. In *44th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, 2011.