Supervised Learning Based Model for Predicting Variability-Induced Timing Errors

Xun Jiao[‡], Abbas Rahimi[‡], Balakrishnan Narayanaswamy[‡], Hamed Fatemi[*], Jose Pineda de Gyvez[*], and Rajesh K. Gupta[‡]
[‡]Department of Computer Science and Engineering, UC San Diego, [*]NXP Semiconductors
{xujiao, abbas, muralib, gupta}@cs.ucsd.edu {hamed.fatemi, jose.pineda.de.gyvez}@nxp.com

***Abstract*—Circuit designers typically combat variations in hardware and workload by increasing conservative guardbanding that leads to operational inefficiency. Reducing this excessive guardband is highly desirable, but causes timing errors in synchronous circuits. We propose a methodology for supervised learning based models to predict timing errors at bit-level. We show that a logistic regression based model can effectively predict timing errors, for a given amount of guardband reduction. The proposed methodology enables a model-based rule method to reduce guardband subject to a required bit-level *reliability specification*. For predicting timing errors at bit-level, the proposed model generation automatically uses a binary classifier per output bit that captures the circuit path sensitization. We train and test our model on gate-level simulations with timing error information extracted from an ASIC flow that considers physical details of placed-and-routed single-precision pipelined floating-point units (FPUs) in 45nm TSMC technology. We further assess the robustness of our modeling methodology by considering various operating voltage and temperature corners. Our model predicts timing errors with an average accuracy of 95% for unseen input workload. This accuracy can be used to achieve a 0%–15% guardband reduction for FPUs, while satisfying the reliability specification for four error-tolerant applications.**

I. Introduction

Variability in microelectronic circuits stems from different sources, including workload variability, dynamic variations in the operating conditions caused by temperature fluctuations and supply voltage ripples, and static process variations that are amplified as device dimensions shrink [6]. Designers typically handle variability by adding a safety margin as guardband. This guardband is computed from a multi-corner worst-case analysis at design time, which leads to overly conservative designs. The most immediate manifestation of reducing guardband is *timing error* that could lead to an invalid state being stored in a sequential element.

Error-tolerant applications can tolerate some degree of errors at the application-level, where multiple valid output values are permitted [7], [8], [9], [11], [14]. Conceptually, such error-tolerant programs have a possible set of 'elastic outputs', and if execution is performed approximately (due to e.g., numerical imprecision or timing errors), the program still appears to execute correctly from the users' perspective. For instance, Rely [7], is a language for expressing approximate computation that allows developers to define a *reliability specification*, which identifies the minimum required probability with which a program must produce an exact result. These relaxed specifications can be inferred from a domain expert developer or through a profiling phase [9], [14], [11], and allow departure from the overly conservative designs to enable more efficient execution. Chisel [11], further enhances the capabilities of Rely by providing combined *reliability* and/or *accuracy* specification. The accuracy specification determines a maximum acceptable difference between the approximate and exact result values, while the reliability specification specifies the probability that a computation will produce an acceptably accurate result. The former specification can be guaranteed through *unequal error protection* methods [5], or by careful partitioning the computation through reliable or unreliable mediums [8]. However, meeting the latter specification is a challenge for automatic model generation, since the model must provide reliable information about the possibility of an error occurrence, i.e., accurate *error prediction*.

We earlier used supervised learning for the error prediction only under hardware variations [12]. This paper makes three main contributions toward the error prediction for unseen variations in the input workload. **1)** We propose a methodology to construct automatic models for bit-level timing error prediction using supervised learning. A logistic regression based model can predict timing errors for each output bit for a desired amount of reduced guardband. The model uses a binary classifier for each output bit that captures the circuit path sensitization resulting in a functional model of the propagation of timing errors through the stages of the pipeline. **2)** We assess the robustness of our bit-level model for error prediction by varying two sets of parameters that significantly reshape the circuit under modeling. We vary structure and topology of the circuit by considering three single-precision pipelined FPUs: adder, multiplier, and square root (sqrt). We also change the electrical properties of the modeled circuit for various voltage and temperature corners. Across this space, considering a guardband reduction of $\{5\%, 10\%, 15\%\}$ our modeling exhibits a minimum accuracy of $\{99\%, 97\%, 94\%\}$ for the multiplier, $\{99\%, 98\%, 85\%\}$ for the adder, and $\{99\%, 83\%, 50\%\}$ for the sqrt. **3)** We use the proposed model to derive guardband reduction at the instruction-level for four error-tolerant applications based on their reliability specifications. Subject to satisfying the reliability specifications, the model provides a wide range of guardband reduction due to accurate error prediction: 10%–15% for the adder in matrix multiplication, 10%–15% for the multiplier in DCT, and 15% for the sqrt in sobel filter.

II. Timing Error Analysis Framework

In this section, we describe our framework for the timing error analysis at the bit-level. Timing error information for a circuit is collected during simulations for a given input workload, operating voltage and temperature corners, and the clock speed. In the following, we describe our circuit modeling and analysis flow for simulation and modeling of the timing errors.

A. Floating-Point Pipelined Circuits

We focus on single-precision FPUs that provide complex circuits and require deep pipelining compared to their integer counterparts. These circuits are fully compatible with IEEE-754 standard and described at the register transfer level (RTL). The output of the pipelined FPUs is not only dependent on the current input data but also depends on the previous inputs. This requires a workload-dependent model to capture a notion of *history* to be able to track the timing errors at the output bits. To capture the history impact, we consider the current data input ($x_i[t]$) and the previous data input ($x_i[t-1]$) jointly as the features, $\{x_i[t-1],\ x_i[t]\}$ and directly

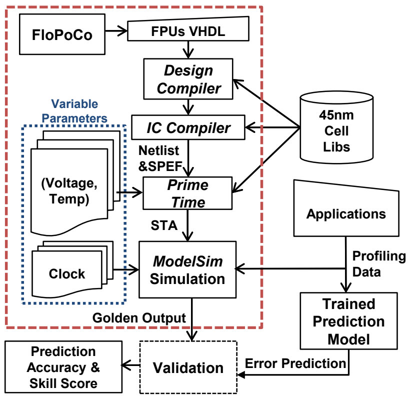


Fig. 1: Timing error extraction and model validation flow.

use the final output as the labels for model training. This per-FPU modeling leads to timing error prediction at the coarse granularity of the entire FPU. However, we have also studied a finer granularity of generating models per each stage, and then composing these per-stage models to predict the timing errors at the output of the FPU. This per-stage modeling exhibits a prediction accuracy as high as the per-FPU modeling while imposing higher computational cost. Hence, we focus on the coarse granularity of per-FPU modeling.

B. Timing Error Extraction

In order to extract and precisely characterize the circuit behavior in the presence of the timing errors, we use a standard ASIC flow to turn a RTL description of the FPUs into a post-layout netlist. We utilize tight synthesis and physical optimizations for timing closure, to ensure a well-optimized netlist for performance and power. The standard ASIC design flow uses TSMC 45nm technology with the *Synopsys Design Compiler* and the *Synopsys IC Compiler* as front-end and back-end design tools, respectively. *Synopsys PrimeTime* is used for voltage and temperature scaling. Next, we extract the optimized netlist in conjunction with the standard delay format (SDF) file related to the specific operating voltage and temperature corner. Finally, we perform a post-layout simulation with the SDF back-annotation in *Mentor Graphics ModelSim* to extract the bit-level timing error information.

In every cycle, a random number generator provides two single-precision numbers as the input for the FPUs, and the flow captures the history of input values of the first pipeline stage to build the training data $\{x_i[t-1], x_i[t]\}$ for modeling. The flow observes and characterizes the timing error at the output (y_i). When a timing error is detected for an output bit during the simulation, we mark the output bit position as 1, otherwise 0 meaning that there is no timing error. The timing error comes from the flip-flop setup/hold timing violation. Fig. 1 illustrates the overall flow for the timing error analysis.

C. Model Training and Testing

Training: For each output bit, we consider a single binary classifier that takes in $\{x_i[t-1], x_i[t]\}$ as the input features, and predicts whether the output bit will face a timing error or not. Therefore, for a given circuit with *K-bit* output, a set of K binary classifiers is required to determine the erroneous output bits for any input data. We use supervised learning to train these binary classifiers. For the training data, two sets of data will be generated, input training data (x_i) and target data as labels (y_i). We use 50K random data – that can maximize the generalization of the model – as the training data.

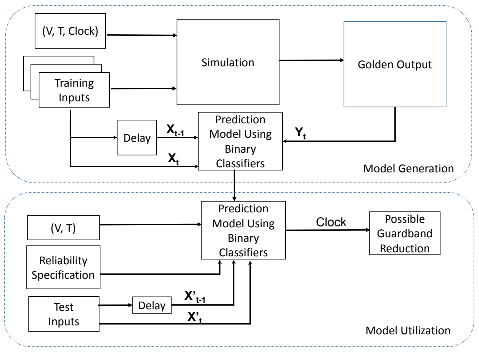


Fig. 2: Model generation and utilization for predicting timing errors.

The 50K input vectors are inserted into the post-layout simulations to generate the pipeline output (y_i). The input feature data, and the target data labels are presented as binary vectors, where a value of 1 indicates the position of the erroneous bit. At each bit position in the output, the training process tries to combine all the input information together to produce the output bit.

Testing: We use profiling to generate test data from real applications. We profile application execution and capture all the input data for different FPUs activated during execution: multiplier, adder and sqrt. These test data are inserted into the post-layout simulations to compute the FPUs output as the golden data. The golden data are compared with the output generated by the trained model for computing the prediction accuracy. Fig. 2 illustrates the process for model generation and model utilization. Based on the accuracy of the timing error prediction, the model can be utilized to reduce the guardband subject to the bit-level reliability specification provided by the application.

Since the timing error prediction can be modeled by binary classifiers, we evaluate different methods, including k-nearest neighbor (k-NN), support vector machine (SVM) and logistic regression (LR) [4]. We observe through extensive experiments that the k-NN algorithm cannot provide good prediction as its average accuracy is of less than 80%. The LR and the SVM reach almost the same high quality of the prediction. However, we have selected the LR due to its efficacy in the training time. In the LR, we learn weights (w) to compare the logic functions that perform well on the training data $\mathcal{D}$. In particular, for an input x we predict 1, or the timing error, if the ratio of $\frac{F(x)}{1-F(x)} >= 1$ where $F(x)$ is given by

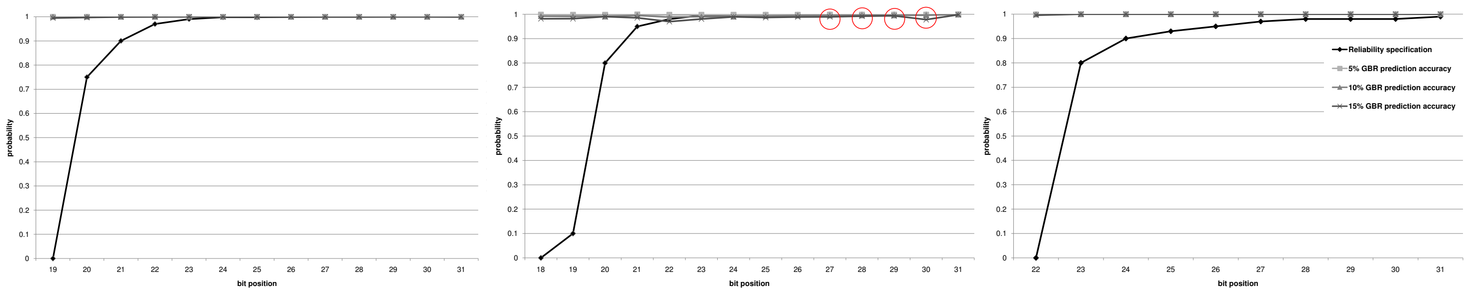
$$F(x) = \frac{1}{1+e^{-w \cdot x}} \tag{1}$$

Since the timing errors are relatively rare, the accuracy of many methods will be high and not illuminating. In particular, a trivial classifier that always predicts "no error" will have a "high" accuracy. To combat this, we evaluate the classifier using skill scores [10]. Skill scores typically normalize for the base rate or performance of trivial classifiers in different ways. That is, a skill score is of the form, skill score = (accuracy of classifier - accuracy for the trivial classifier) / (maximum achievable accuracy - accuracy for the trivial classifier). The skill score must be positive.

III. Experimental Results

A. Experimental Setup

The synthesizable VHDL codes for the FPUs are generated by FloPoCo [2], and then synthesized and placed-and-routed by



(a) multiplier (b) adder (c) sqrt

Fig. 3: Reliability specification and prediction accuracy for sobel filter at (0.85V, 50°C). The red circles refer to violations where accuracy of prediction is less than the reliability specification.

the ASIC flow described in Section II-B. Afterwards, the static timing analysis is done through *Synopsys PrimeTime* for SDF files generation. The training data is randomly generated, while the test data comes from the profiling of the application on Multi2Sim [3], a cycle-accurate heterogeneous system simulator. Both training and testing data sets are simulated using *Mentor Graphics Modelsim* in gate-level simulations with the back-annotated delays to produce the timing error information at the final outputs. The machine learning module is provided by Scikit learning module [13] in Python, where various learning approaches are included.

B. *Bit-Level Reliability Specification*

The error-tolerant applications exhibit enhanced error resilience at the application-level when multiple valid output values are permitted. Instead of a single output value, the output value is associated within an application-specific quality metric, such as peak signal-to-noise ratio (PSNR). Therefore, if execution is not numerically precise, the application can still appear to execute correctly from the users' perspective. We focus on error-tolerant applications mainly from image processing domain, including sobel filter, gaussian filter, and DCT. In image processing applications, a PSNR larger than 26dB is generally considered as acceptable to users. For other computational applications, we use the average relative error between the elements of the outputs of the error-free and erroneous executions. In particular, we set the desired quality of PSNR to a minimum 26dB and the average relative error to a maximum of 10% which is commensurate with other work on quality trade-offs [9], [14], [11].

We then compute a bit-level reliability specification through a profiling phase [14]. This profiling is done through fault injection testing using a modified version of Multi2Sim simulator to flip a single bit among the 32-bit output for three frequently activated FPUs, the multiplier, the adder, and the sqrt. For example, flipping the 20th bit in the multiplier product with probability of 0.3 results a PSNR of 34dB. From this, we estimate that a reliability specification of 70% is sufficient for the 20th bit. We increase the fault injection probability until the PSNR drops to 26dB. This probability is referred as cutoff fault injection probability. The reliability specification can then be computed as 1-cutoff probability. Fig. 3(a) shows the reliability specification requirement for the multiplier used for sobel filter at operating condition of (0.85V, 50°C) starting at 19th bit, since the reliability specifications for the previous 18 bits are 0. Fig. 3(b) and Fig. 3(c) illustrate the reliability specifications of the adder and the sqrt respectively, starting from the bit position where the reliability specification is higher than 0. As shown, and as expected, higher significant bits require a larger reliability, when compared to the lower significant bits.

C. *Model Prediction Accuracy*

We generate 50K random data for training while the test data is collected from profiling the applications with real world input sets. For the image processing applications we use images in Caltech-UCSD Birds 200 vision dataset [1], and for matrix multiplication and DCT we use 200K random input data. The data are placed into the timing error extraction module, with ASIC flow at two operating corners of (0.72V, 0°C) and (0.85V, 50°C). The gate-level simulation is done at 5%, 10% and 15% guardband reduction to extract the timing erroneous information. The flow generates the training data for LR training and the test data will be used for computing the prediction accuracy and the skill score. Finally, the golden output data from the gate-level simulations is compared with the prediction results to validate the robustness of model as well as the skill score. Fig 1 shows this overall flow. Table I summarizes the range of prediction accuracy at two operating corners. The skill score range is positive, indicating that our prediction is better than the trivial classifier.

We compare the reliability specification versus the prediction accuracy, to assess whether our model can meet the bit-level reliability specification. Fig. 3(b) shows that at (0.85V, 50°C) the adder model violates the reliability specification at 27th, 28th, 29th and 30th bit position. The largest gap, 0.0011, between the reliability specification and our model prediction accuracy occurs at bit position 30, where reliability specification is 0.9998. No violation is shown in the multiplier and the sqrt because our models can always exhibit a prediction accuracy higher than the bit-level reliability specification. The highest prediction accuracy among these three models is 0.9987.

D. *Guardband Reduction Subject to Reliability Specification*

1) Bit-Level Guardband Reduction: As shown in Fig. 2, our model can be utilized to reduce the conservative guardband, while guaranteeing the reliability specification therefore generating the acceptable output result. The model is analyzed at two voltage/temperatute corners with three levels of guardband reduction (GBR): 5%, 10% and 15%, meaning that we increase the clock speed by 5%, 10% and 15%. Table II shows the possible GBR, as a percentage, for the multiplier. This GBR is computed per bit position subject to meeting the reliability specification of the applications as described in Section III-B. For each bit position, the maximal GBR is presented. For example, at 29th bit in Table II, 15/15 means 15% GBR can be achieved for the bit position 29 at the corner (0.72V, 0°C) as well as (0.85V, 50°C). This is because our bit prediction accuracy is higher than the particular bit reliability specification under 15% GBR. A pair of 10/10 means that we can only increase the

TABLE I: Prediction accuracy (minimum, average, maximum) at two corners.

	(0.72V, 0°C)			(0.85V, 50°C)		
	adder	multiplier	sqrt	adder	multiplier	sqrt
Sobel filter	(0.855,0.975,1)	(0.952,0.994,1)	(0.806,0.961,1)	(0.875,0.990,1)	(0.970,0.995,1)	(0.501,0.914,1)
Gaussian filter	(0.868,0.975,1)	(0.977,0.996,1)	(-,-,-)	(0.850,0.993,1)	(0.968,0.995,1)	(-,-,-)
Matrix multiplication	(0.873,0.934,1)	(0.998,0.999,1)	(-,-,-)	(0.851,0.993,1)	(0.987,0.997,1)	(-,-,-)
DCT	(0.854,0.957,1)	(0.941,0.991,1)	(-,-,-)	(0.880,0.974,1)	(0.947,0.990,1)	(-,-,-)

TABLE II: Bit-level guardband reduction (%) for the multiplier at two corners: (0.72V, 0°C)/(0.85V, 50°C).

	0	—	23	24	25	26	27	28	29	30	31
Sobel filter	15/15	—	15/15	15/15	15/15	15/15	15/15	15/15	10/10	0/5	15/15
Gaussian filter	15/15	—	15/15	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
Matrix multiplication	15/15	—	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
DCT	15/15	—	15/15	15/15	10/15	15/15	15/15	15/15	15/15	15/15	15/15

TABLE III: Bit-level guardband reduction (%) for the adder at two corners: (0.72V, 0°C)/(0.85V, 50°C).

	0	—	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Sobel filter	15/15	—	15/15	10/15	10/15	10/15	15/15	15/15	15/15	10/5	10/5	10/5	10/5	10/0	10/0	10/0	10/0	15/15
Gaussian filter	15/15	—	15/15	15/15	15/15	15/15	15/15	10/10	5/5	10/5	10/5	10/5	10/5	10/5	10/5	5/5	0/0	10/5
Matrix multiplication	15/15	—	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/10	15/10	15/10
DCT	15/15	—	15/15	15/15	15/15	15/15	15/15	15/15	10/10	15/10	15/10	10/10	15/5	15/5	10/5	10/10	0/5	15/15

clock speed by 10% maximally otherwise our prediction accuracy becomes smaller than the bit reliability specification which leads to the unacceptable output result. A GBR of 0 means we cannot provide a prediction accuracy while satisfying the bit reliability specification. From Table II, we can see that almost all the bits can achieve moderate GBR using our model.

Table III shows the GBR for the adder. There are also some bit positions where our model cannot meet the reliability specification at any GBR. For example, no GBR can be applied at 30th bit position for the adder in gaussian filter. The GBR for the sqrt reaches 15% for its entire 32bits.

2) Instruction-Level Guardband Reduction: By analyzing GBR at the bit-level, the instruction-level GBR can also be derived across different FPUs. This is achieved by taking the minimal GBR across all the bits. For example, a 5% GBR can be achieved for multiplier in sobel filter at corner (0.85V, 50°C) since the minimal GBR is limited by the 30th bit. For gaussian filter, a better GBR of 10%, for both corners is achieved for the multiplier. However, no GBR can be achieved at corner (0.72V, 0°C) for the multiplier in sobel filter from Table II. Since the role played by multiplier in gaussian filter is different from sobel filter and they receive statistically different input data, their bit-level reliability specification is different.

TABLE IV: Instruction-level guardband reduction (%) at two corners: (0.72V, 0°C) / (0.85V, 50°C).

	multiplier	adder	sqrt
Sobel filter	0 /5	10 /0	15 /15
Gaussian filter	10 /10	0 /0	—
Matrix multiplication	15 /15	15 /10	—
DCT	10 /15	0 /5	—

Table IV can guide a guardband reduction mechanism at the instruction-level during design time or runtime. For example, given a sobel filter under the operating corner of (0.72V, 0°C), we can reliably reduce the guardband 0%, 10% and 15% for the multiplier, the adder, and the sqrt, respectively. For a single instruction type across all the applications and the operating corners, we can also benefit from the GBR. For instance, the multiplier can achieve at least 5% and up to 15% GBR at (0.72V, 0°C).

IV. Conclusion and Future Work

Our proposed methodology generates a functional model for predicting the timing errors at the bit-level for a given amount of reduced guardband. The model is trained by logistic regression method through random input sequences, while the testing data is extracted from the actual execution of the applications to validate the prediction accuracy of our model. The model exhibits an average accuracy of 95% for the timing error prediction with positive skill score for various voltage/temperature corners and unseen workload. We verify the effectiveness of our model for reducing guardband while satisfying the reliability specification for the error-tolerant applications. Using this binary classifier-based model, the guardband can be reduced in the range of 0%–15% during matrix multiplication execution depending on the type of instructions and operating corner. Sobel filter and DCT benefit from the same range of guardband reduction, while gaussian filter limits it to maximum 10%. Our ongoing work concerns efficiency of model building.

References

[1] Caltech-UCSD birds 200. http://www.vision.caltech.edu/visipedia/CUB-200.html.

[2] FloPoCo: Floating-point cores generator. http://flopoco.gforge.inria.fr/.

[3] Multi2sim: A heterogeneous system simulator. https://www.multi2sim.org/.

[4] C. M. Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[5] S. Borade, et al. Unequal error protection: An information-theoretic perspective. *IEEE Trans. on Information Theory*, 55(12):5511–5539, Dec 2009.

[6] S. Borkar, et al. Parameter variations and impact on circuits and microarchitecture. *Proc. DAC*, pages 338–342, June 2003.

[7] M. Carbin, et al. Verifying quantitative reliability for programs that execute on unreliable hardware. *Proc. OOPSLA*, pages 33–52, 2013. ACM.

[8] H. Cho, et al. Ersa: Error resilient system architecture for probabilistic applications. *IEEE Trans. on CAD*, 31(4):546–558, April 2012.

[9] H. Esmaeilzadeh, et al. Neural acceleration for general-purpose approximate programs. *Proc. MICRO*, pages 449–460, Dec 2012.

[10] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal. of the American Statistical Assoc.*, 102(477):359–378, 2007.

[11] S. Misailovic, et al. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. *Proc. OOPSLA*, pages 309–328, 2014. ACM.

[12] A. Rahimi, et al. Hierarchically focused guardbanding: An adaptive approach to mitigate pvt variations and aging. *Proc. DATE*, pages 1695–1700, 2013.

[13] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, pages 2825–2830, 2011.

[14] A. Rahimi, et al. A variability-aware openmp environment for efficient execution of accuracy-configurable computation on shared-fpu processor clusters. *Proc. CODES+ISSS*, pages 1–10, Sept 2013.