Combining Structural and Timing Errors in Overclocked Inexact Speculative Adders

Xun Jiao[†], Vincent Camus[‡], Mattia Cacciotti[‡], Yu Jiang[§], Christian Enz[‡], Rajesh K. Gupta[†]
[‡]Ecole Polytechnique Fédérale de Lausanne, Neuchâtel, Switzerland
[†]University of California, San Diego, USA
[§]Tsinghua University, Beijing, China
These authors contributed equally to this work:
xujiao@cs.ucsd.edu, vincent.camus@epfl.ch

***Abstract*—Worst-case design is used in IoT devices and high performance data centers to ensure reliability, leading to a power efficiency loss. Recently, approximate computing has been proposed to trade off accuracy for efficiency. In this paper, we use Inexact Speculative Adders, which redesign the adder architecture to shorten its critical path and improve performance, but introduces controlled structural errors. On the other hand, overclocking is used to reduce conservative timing guardbands but could normally introduce catastrophic timing errors, we thus apply a supervised learning model to overclock speculative adders and predict their timing errors. We build a methodology to combine both structural and timing errors and analyze how they interplay with each other to limit the overal errors.**

I. Introduction

Power efficiency has become the primary concern for IoT applications, both at the sensor node as well as on its cloud computing counterpart. Unfortunately, achieving high power-efficiency and robustness requires complex and conflicting design constraints and expensive safety margins. For low-power IoT devices, Process-Voltage-Temperature (PVT) variations are enormous, and designing for worst-case is disastrous for energy efficiency. For cloud computing, high-performance data centers are required to process in real-time the massive amount of data collected from the growing network of IoT devices. Fortunately, the inherent redundancy and noise of such data makes its processing resilient to errors and approximations.

Approximate computing is a recent design paradigm that allows to trade off a small amount of accuracy for significant power, area and delay savings. In its early stage, it has been facing resistance by going against the well-established and ultra-conservative exact computing. Now, as a consequence of approaching the end of Moore's Law, approximate computing has become a leading trend to keep improving energy efficiency and sustain the exploding demand of data processing of IoT and cloud applications. Approximate computing has been investigated at all levels of abstraction, from circuit level through voltage-precision scaling [1] up to application level with computation skipping [2].

In the case of arithmetic circuits, that are at the heart of DSPs, two approaches have shown prominent interests, namely inexact speculative circuits [3] and guardband-reduction with timing-error prediction [4]. First of all, they have independently shown substantial ability to relax timing and energy constraints. Particularly suited for arithmetic operators such as adders and multipliers, circuit-level speculation redesigns the architecture of circuits by shortening the circuit critical path, reducing delay, area and power consumption. In a desperate attempt to protect circuits from PVT variations, designers typically apply ultra-conservative—thus ultra-expensive—guardbands to avoid timing errors, timing-error prediction allows direct performance improvement by enabling operation under overclocking.

Those two approaches targeting different abstraction levels and introducing different types of errors in digital circuits could intuitively be the perfect combination to maximize circuit efficiency. As in analog circuits, in which contributions from different noise sources are fine-tuned to maximize SNR while minimizing power consumption, this work proposes to combine errors from speculative architectures and overclocking in order to minimize the overall error compared to what an exact and properly-clocked circuit would output.

In this work, we apply bit-level timing-error prediction [4] to predict timing errors of overclocked Inexact Speculative Adders (ISA) [3]. Our contributions are the following:

- We build a bit-level timing-error prediction model for overclocked ISA evaluating arithmetic effects of errors.
- We develop a methodology to combine both structural and timing errors and to show their joint contribution on the average relative error.
- We characterize the trade-off between structural and timing errors for different overclocked ISA designs.
- We analyze the distribution of both types of errors and how they interplay with each other in an overclocked ISA.

II. Inexact Speculative Adders

Since adders are the most common arithmetic blocks used in DSPs, many attempts have been made to design approximate versions of them [5]–[8]. To this purpose, *carry speculation* is an interesting technique, exploiting the fact that in additions, carry propagate sequences are typically short [5]. Hence, it is possible to estimate, more or less accurately, an intermediate carry using a limited number of previous stages. This allows to splits the carry chain into two or more shorter paths, relaxing the constraints over the entire design and pushing energy, delay and area beyond the limits imposed by traditional design.

This work was supported by the NanoTera IcySoC project of the Swiss National Science Foundation (SNSF) and the Variability Expedition in Computing of the US National Science Foundation (NSF) under Award No. 1029783.

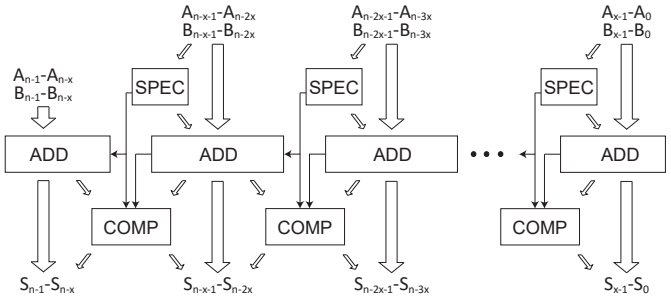


Fig. 1. General block diagram of an Inexact Speculative Adder (ISA). Each speculative segment consists of a carry speculator (SPEC), a regular adder (ADD) and an error compensation block (COMP).

Many speculative adders have been proposed in literature based on the ETAII concept [5]. Among them, the Inexact Speculative Adder (ISA) [3] has generalized and optimized the architecture for speculative compensated addition to minimize speculation overhead and by implementing a dual-direction compensation mechanism. Moreover, it has already been successfully verified and integrated in multiplier circuits [9]. The following subsections presents the ISA in details.

A. Inexact Speculative Adder

The block diagram of an ISA is depicted in Fig. 1. It splits the carry chain in multiple paths executed concurrently, each of them consisting of a carry speculator block (SPEC), a sub-adder block (ADD) and an error compensation block (COMP).

The SPEC generates a partial carry signal from a limited number of operand bits using a carry look-ahead approach. When a propagate chain covers the full block, the exact carry cannot be speculated from the partial product and the output carry is guessed. The ADD calculates local sums from the speculated carry. The COMP detects the speculation faults by comparing the carry generated from the SPEC with the carry-out coming from the previous ADD. It then compensates faulty sums either by attempting to correct a few bits of the local sum or by reducing relative error over a few bits of the preceding sum. This allows to avoid massive errors generated from an internal overflow caused by an inconsistent carry.

B. Error Compensation Technique

The achieved addition arithmetic is illustrated in Fig. 2. The COMP's error-correction technique consists in incrementing or decrementing only a small group of LSBs of the local sum to compensate for the erroneous speculated carry. In most cases, it can fully resolve carry errors, but if those stages are all in propagate modes, correction is impossible as it would lead to an internal overflow. In this situation, the uncorrected bits ensure a low relative error of the result, since they have a higher significance than the error bit. The COMP also uses error balancing to flip a small group of MSBs of the preceding sum to further reduce the relative error.

Thus, using the COMP block reduces simultaneously error rate and relative error. Moreover, as the correction hardware is executed concurrently to the local addition, this technique has a minimal impact on the critical path.

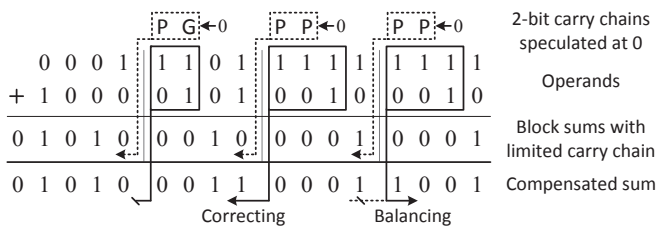


Fig. 2. Example of ISA addition arithmetic with 2-bit speculation, 1-bit correction and 1-bit error reduction. Faults only occur in the two right-hand paths. The 1st LSB of the central path can be corrected. The 1st LSB of the right path cannot be corrected, so the 1st MSB of the preceding sum is flipped.

C. Design Strategy

Inexact Speculative Adders can easily be designed with a delay-accuracy approach: the adequate delay tradeoff is obtained by sizing SPEC and ADD blocks, principal slack elements of the ISA, while the sizing of the COMP techniques can then be used to tune the mean accuracy and limit the worst-case error. Thanks to a custom sizing of each speculative path and each speculative path block, the ISA architecture allows very precise tuning of multiple error characteristics while optimizing circuit performance and efficiency.

III. Guardband Reduction with Bit-level Timing-error Prediction

Circuit designers typically apply ultra-conservative guardband to protect circuits from timing errors. This guardband is computed from a multi-corner worst-case analysis at design time, which leads to operational inefficiency. Attempting to reduce such guardband, *Better-than-worst-case* (BTWC) approaches are proposed to reduce guardband by overclocking, and use recovery schemes to correct the timing errors caused by overclocking [10]–[12]. While effective, such techniques incur silicon overhead for online monitoring and recovery penalty.

To avoid such overhead, model-guided adaptive techniques have been proposed to predict timing errors in advance and then adjust guardband accordingly. Instruction-level models characterize the susceptibility of instructions to timing errors by measuring their maximum delay and guide the guardband reduction [13, 14]. WILD [15] further improves the accuracy of delay characterization by considering the effect of input workload.

These models focus on predicting timing error existence but not on the quantitative effect of timing errors on the arithmetic value due to their ignorance of bit-level information. In this paper, we use a modified supervised learning-based model which has been proposed to predict timing errors at bit-level.

A. Bit-level Timing-error Prediction Model

As proposed in [4], the bit-level timing-error prediction model for guardband reduction uses binary classification method to predict timing errors for a given clock reduction and input load. It captures the dynamic circuit path sensitization behaviors by learning the mapping relationship between input workload and bit-level timing errors. For each bit position, a

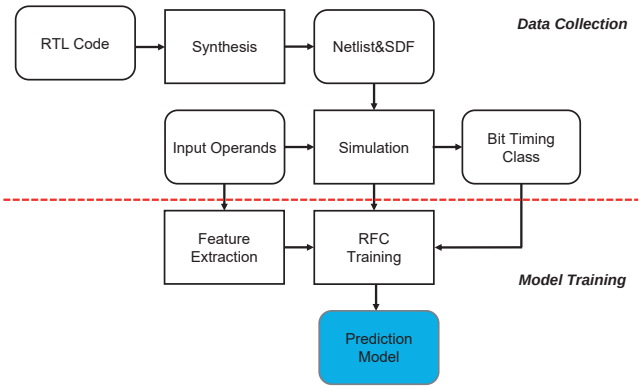


Fig. 3. Bit-level timing error prediction model construction flow.

binary classifier is trained to predict if it is timing-erroneous. The overall model construction flow, containing two parts, *Data Collection* and *Model Training*, is illustrated in Fig. 3.

Data Collection: First, we synthesize the RTL code into a netlist and extract the corresponding standard delay format (SDF) file. Second, using random data as input operands, we perform SDF-annotated gate-level simulations to generate output data at unsafe clock periods. At each cycle, a new input vector is fed into the simulation. We define $x[t]$ as the input vector, $y_{\text{RTL}}[t]$ as pure-RTL output and $y[t]$ as gate-level output, at cycle t. For an output bit position n at cycle t, we define the timing class of $y_n[t]$ as $c_n[t]$, which is one of two timing classes: {*timing-correct*, *timing-erroneous*} based on whether it has a timing error. If $y_n[t]$ matches $y_{\text{RTL}n}[t]$, then $c_n[t]$ is timing-correct, otherwise timing-erroneous.

Model Training: First, we extract the useful feature vectors from input data. At cycle t, the output $y[t]$ is jointly determined by current input $x[t]$ and preceding cycle input $x[t-1]$. Besides, we also consider output bit value $\{y_{\text{RTL}n}[t-1], y_{\text{RTL}n}[t]\}$ as input feature because the timing error (bit flip) can only occur when these two bits are different [16]. If these two bits hold same value, the latched value is correct to users even if the clock period does not meet the sensitized path delay. Thus, we consider $\{x[t], x[t-1], y_{\text{RTL}n}[t-1], y_{\text{RTL}n}[t]\}$ as our input feature and the $c_n[t]$ as output label. For each bit position, we train a binary classifier using supervised learning methods. In this paper, we use Random Forest tree Classification (RFC) as our learning method to balance the prediction accuracy and training cost. RFC is an ensemble method composed of a number of decision trees (DT), which learn a set of decision rules based on the pattern of input and their possible outcomes. DT considers the joints effects of different bit positions but could incur overfitting problem. RFC alleviates overfitting issue by developing more than one decision tree and use their average result as final prediction. It may lose the opportunity to learn some "irregular" patterns, overall it reduce the overfitting and boost performance.

B. Prediction Model Evaluation

To evalutate the model prediction accuracy for a selected overcloking rate, we define the *average bit-level prediction error rate* ($ABPER$) as follows:

$$ABPER[clk] = \frac{\sum_{bit\ n}\left(\frac{\sum_{cycle\ t}|TC^{(pred)}_{clk,n,t} - TC^{(real)}_{clk,n,t}|}{||\#cycles||}\right)}{||\#bit\_positions||} \quad (1)$$

where $TC^{(pred)}_{clk,n,t}$ and $TC^{(real)}_{clk,n,t}$ are the predicted and real timing classes (0 for timing-erroneous and 1 for timing-correct) at a given clock period clk, bit position n and cycle t. This metric is a good indicator of bit-level model prediction accuracy.

IV. Combining Structural and Timing Errors

All previous works have discussed individual use of either approximate circuit design, such as speculative compensated architectures, or guardband-reduction (overclocking) timing-error prediction. But those two approaches targeting different abstraction levels could intuitively be the perfect and complementary combination to maximize circuit performances and robustness. Indeed, timing errors occur on the critical paths, which would be split into multiple shorter paths in a speculative circuit. The timing errors would thus be distributed among all outputs—instead of only degrading MSBs in conventional circuits—but at the cost of structural errors due to speculation. Parameters of the speculative structure and levels of guardband reduction can be adjusted together in order to find an optimum between timing and structural errors.

This study focuses on the case of binary addition based on the use of ISA adders synthesized for 3.3 GHz in a 65 nm technology. The methodology adopted is the following:

- Several ISA adders have been selected and implemented with design parameters optimizing error and circuit costs.
- Timing-error prediction has been adapted to predict timing errors on these circuits for different overclocking levels.

A. Error Combination

A new error model needs to be developed to distinguish and combine correctly the error contributions from both abstraction layers. First, at behavioral level, *structural errors* are caused by the design of the ISA architecture. Those deterministic errors vary with the selection of design parameters such as the selection of speculation, error-correction and error-reduction mechanisms. Structural errors are obtained by comparing the outputs from the designed circuit from exact addition results. Then, at gate level or below, *timing errors* occur when overclocking the ISA circuit, thus are obtained by comparing the over-clocked circuit to the same inexact but properly-clocked circuit. Those errors vary with different clock periods and are less predictable as they also depend from the previous circuit state or inputs.

To simplify the three type of output values used to compute those errors, we define the following types of output values:

- y_{silver}, the *silver output* obtained from the over-clocked ISA circuit, containing both structural and timing errors.
- y_{gold}, the *golden output* the expected value from the implemented circuit, containing the structural errors only.
- y_{diamond}, the *diamond output* ideal output value from an exact addition or conventional adder circuit.

Thus, we compute the *arithmetic error* (E) from each abstraction level as:

$$E_{\text{struct}} = y_{\text{gold}} - y_{\text{diamond}} \qquad E_{\text{timing}} = y_{\text{silver}} - y_{\text{gold}}\,, \quad (2)$$

whereas the *relative error* (RE), both contributions being calculated with respect to the exact result, is defined as:

$$RE_{\text{struct}} = \frac{y_{\text{gold}} - y_{\text{diamond}}}{y_{\text{diamond}}} \qquad RE_{\text{timing}} = \frac{y_{\text{silver}} - y_{\text{gold}}}{y_{\text{diamond}}}\,. \quad (3)$$

Despite this study only considers unsigned computations, it is important for arithmetic and relative errors to be kept signed. Indeed, if both error contributions are in the same directions, they would add to each other to increase the overall error, such as in Fig. 4:

output values			error contributions	
y_{diamond}	1000	8	RE_{struct}	$\frac{6-8}{8} = -\frac{2}{8}$
y_{gold}	0110	6	RE_{timing}	$\frac{4-6}{8} = -\frac{2}{8}$
y_{silver}	0010	4	RE_{joint}	$-\frac{2}{8} - \frac{2}{8} = -\frac{4}{8}$

Fig. 4. Example of additive errors (exact output y_{diamond}, exemplary erroneous outputs y_{gold} and y_{silver} from ISA and over-clocked ISA, respectively)

But if two errors happening simultaneously are in opposite directions, they could compensate each other and reduce the overall error, such as in Fig. 5:

output values			error contributions	
y_{diamond}	1000	8	RE_{struct}	$\frac{6-8}{8} = -\frac{2}{8}$
y_{gold}	0110	6	RE_{timing}	$\frac{7-6}{8} = +\frac{1}{8}$
y_{silver}	0111	7	RE_{joint}	$-\frac{2}{8} + \frac{1}{8} = -\frac{1}{8}$

Fig. 5. Example of compensating errors

Fig. 6 depicts the flow used to combine ISA errors with timing errors in the case of arithmetic errors for example.

```
1  inputs: set of ISA architectures, input set, clock periods
2  outputs: mean arithmetic errors
3  foreach ISA ∈ ISA architectures do
4      foreach x ∈ input vectors do
5          compute y_diamond[x]
6          compute y_gold[x, ISA]
7          compute E_struct[x, ISA]
             = y_gold[x, ISA] − y_diamond[x]
8          foreach clk ∈ clock periods do
9              compute y_silver[x, ISA, clk]
10             compute E_timing[x, ISA, clk]
                 = y_silver[x, ISA, clk] − y_gold[x, ISA]
11             compute E_joint[x, ISA, clk]
                 = E_timing[x, ISA, clk] + E_struct[x, ISA]
12     compute means of |E_joint[x, ISA, clk]| over inputs
```

Fig. 6. Pseudo-code computing the mean arithmetic error of over-clocked ISAs with structural and timing errors.

B. Model Evaluation

Although the $ABPER$ is a good metric for bit-level prediction accuracy, it cannot exhibit the misprediction effect on the output arithmetic value of the adder. Thus, for this work, we define another metric using output arithmetic error values instead of bit timing classes, the *average value-level predictive error* ($AVPE$):

$$AVPE[ISA, clk] = \frac{\sum_{cycle\ t} \frac{|\, y_{\text{silver}}^{(pred)}[ISA, clk, t] - y_{\text{silver}}^{(real)}[ISA, clk, t]\,|}{y_{\text{silver}}^{(real)}[ISA, clk, t]}}{||\#cycles||} \quad (4)$$

where $y_{\text{silver}}^{(pred)}$ and $y_{\text{silver}}^{(real)}$ are the predicted and real arithmetic output values of a given ISA, clock period clk and at cycle t.

Note that the model does not directly generate arithmetic values, it only generates timing-class vectors, which are arrays of bit-flip positions, and deduces the corresponding y_{silver} compared to the expected output y_{gold}.

V. Experimental Results

A. General considerations

Twelve different ISA designs have been selected from [17], they are the best implementations fitting the 0.3 ns timing constraints. All ISA have regular structures with uniformly sized blocks (i.e. parallel paths of 2x16, 4x8, 8x4 bits only) and are denoted by quadruples of bit-widths: (block size, SPEC size, correction, reduction). They have been confronted to an exact adder, also constrained at 0.3 ns.

Approximate circuits are commonly characterized and validated through the simulation of random sets of inputs. As a matter of fact, the presented results are statistical estimations depending on the random sample distribution (occurrence of specific patterns initiates errors in specific adders). In this work, adders are characterized using a sample of ten million unsigned random inputs. The main metric considered is the Root Mean Square (RMS) of the relative error RE as it is independant of the adder bit-width and proportional to the SNR, which it interesting for many applications, particularly in multimedia processing.

Circuits have been synthesized with Synopsys Design Compiler in an industrial 65 nm technology from high-level descriptions in order to benefit from the compiler's optimization libraries and most favorable architecture choices. Delay-annotated gate-level simulations have been run with Mentor Modelsim in order to extract timing errors for three delays: 0.285 ns, 0.27 ns and 0.255 ns, corresponding to 5, 10 and 15 % of Clock-Period Reduction (CPR) from the safe-clock period of 0.3 ns. Machine learning methods used to construct the model come from the Scikit-learn Python package [18].

B. Timing-error Prediction Evaluation

Fig. 7 presents the $ABPER$ for each ISA at three CPRs: 5, 10 and 15 %. From this figure, we firstly observe that almost all $ABPER$ values are around or less than 1 %, demonstrating a high prediction accuracy of the model. Second, $ABPER$ at

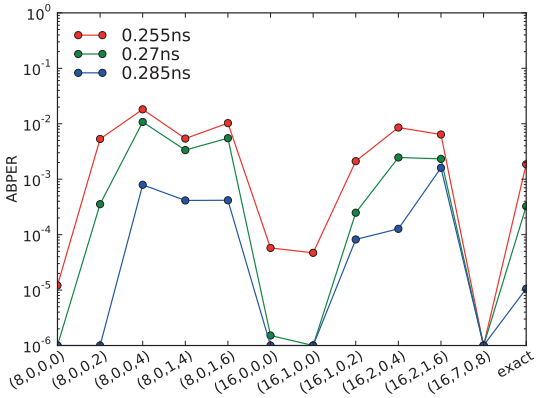


Fig. 7. Average bit-level prediction error rate (ABPER) under three CPR.

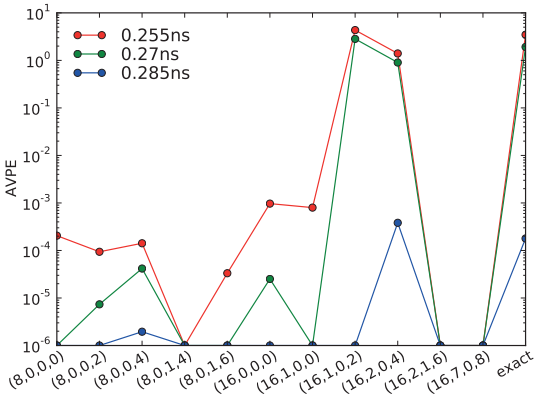


Fig. 8. Average value-level predictive error (AVPE) under three CPR.

higher CPR is always larger than that at less CPR. For example, the third ISA (8,0,0,4), has $ABPER$ around 0.1% at 0.285ns (5% CPR), and has $ABPER$ around 1% for 10% and 15% CPR. This is because more paths violating timing specification resulting in more timing errors, which makes model harder to track all path sensitization behaviors. Some $ABPER$ can reach 0 if there is no timing error, such as ISA (8,0,0,0) at 0.285 ns and 0.27 ns. We use 10^{-6} as $ABPER$ in this case.

Fig. 8 presents the $AVPE$ for each ISA at three CPRs. From this figure, we observe that although bit-level prediction accuracy are always good but the mispredicted bits could sometimes cause a large arithmetic error. For example, the eighth ISA (16,1,0,2) at 0.255 ns and 0.27 ns causes a $AVPE$ around 5. This is because many mispredicted bits are among most significant bits that can cause a large deviation up to 2^{32} from original value. While for most ISA, the third ISA (8,0,0,4) for example, has $AVPE$ less than 0.1% for all three CPRs, showing the misprediction effect on arithmetic value is negligible. Similar with Fig. 7, we neglect $AVPE$ value when it is lower than 10^{-6}. Overall, most $AVPE$ are lower than 10^{-2}, indicating that misprediction on arithmetic error is tolerable for most ISA designs.

C. Results of Error Combination

Fig. 9 shows the structural and timing relative error RMS as well as their resulting joint contribution for ISA designs under the three CPRs.

At the lowest overclocking rate of 5 % (Fig. 9a), we immediately observe that the exact adder circuit (rightmost of the figure) is subject to large timing errors which make it the worst adder of the group in terms of overall joint error RMS. We find that for most ISA adders, the joint error is dominated by the structural-error contribution coming from the speculative architecture. Low and medium-accuracy ISA circuits (on the left part of the figure) seem very robust to timing errors, having negligible timing errors compared to structural errors. Among the high-accuracy ISA designs, only ISA (16,2,0,4) has succumbed to a massive amount of timing errors. Though, if this specific ISA has a low sensitivity threshold to timing errors, it is still better than the exact adder in terms of joint error.

At 10 % CPR (Fig. 9b), timing-error contributions strongly increase, but stay lower than structural-error contributions for low-accuracy ISA adders. Two additional high-accuracy ISA circuits have fallen to timing errors: ISA (16,0,0,0) and (16,1,0,2) ISA circuits. Yet, they are still operating slighlty better than the exact adder, whose average error, entirely due to timing, has been multiplied by 3 compared to 5 % CPR.

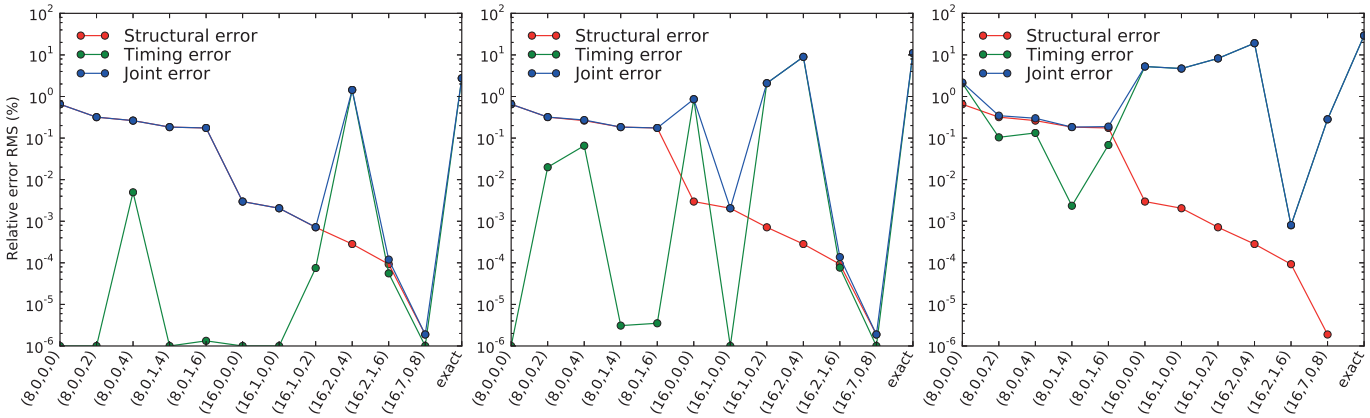
At the highest CPR of 15 % (Fig. 9c), all the selected high-accuracy ISA designs have fallen to timing errors. Yet, some of these designs still exhibits decent overall accuracy such as ISA (16,2,1,6). This latter relegates to the second place ISA (16,7,0,8), which has a more accurate architecture but is found less resilient to aggressive overclocking. Understanding this variability in timing-error robustness as well as the difference of threshold between structural and timing errors could be highly beneficial to low-power and time-constrained circuit design. This would require a deeper analysis combining more speculative designs to better cover the design space offered by inexact speculative circuits.

For low-accuracy ISA overclocked with 15 % CPR (left part of Fig. 9c), it is particularly interesting to note the high balance between timing and structural errors. This compromise between the two error contributions gives generally a better overall accuracy than adders designed with high structural accuracy.

D. Structural and Timing Error Balance

In order to better understand how the two types of errors interplay with each other, Fig. 10 displays the internal distribution of structural and timing errors within the example of 15 %-overclocked ISA (8,0,0,4) since this configuration shows the best balance between errors (c.f. Fig. 9c). Arithmetic structural errors have been translated into their equivalent bit-level positions. Note that the timing errors distribution is not as regular as the structural errors distribution. While it is easy to distinguish when several arithmetic speculative errors occur simultaneously on different speculative paths and translate independent errors into bit positions, timing errors might span over various outputs.

Structural errors are immediately recognizable on three speculative paths (the first speculative path, operating from the



(a) 5% (b) 10% (c) 15%

Fig. 9. Relative error RMS of ISAs under 5 %, 10 % and 15 % overclocking.

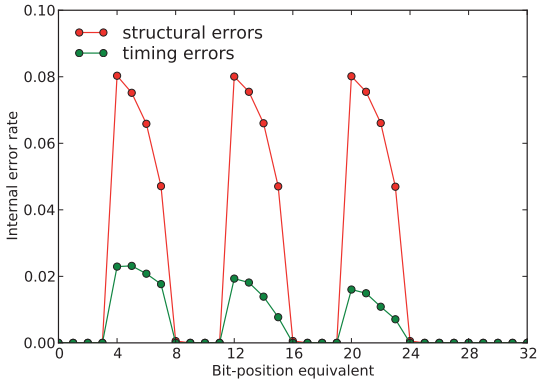


Fig. 10. Bit-level-equivalent error distribution in ISA (8,0,0,4) under 15 % CPR.

LSB, uses directly the adder carry-in so doesn't have errors). As this ISA only has 4-bit error reduction (no error correction), it only introduces errors on the preceding sub-adder sums, that is why structural-error peaks are slightly shifted on the left of the figure.

In a conventional adder, overclocking would dangerously degrade MSBs. In this ISA, despite causing structural errors, the 4-path speculative structure leads to a split of critical path, distributing the timing errors over those paths instead of the MSBs. Those errors mainly occur on the 4-bit error reduction block, last logic element in the critical path. This trade-off between structural and timing errors demonstrates the good resilience of ISA architectures against conventional circuits.

VI. Conclusion

In this paper, we build a bit-level timing-error prediction model for overclocked Inexact Speculative Adders. We develop a methodology to combine their structural and timing errors and show their joint effect on the average relative error. We show that speculative adders are more resilient to overclocking than conventional adders. We show that the combination of speculation and overclocking is a perfect and complementary combination to maximize circuit robustness. Indeed, the speculative structure causes structural errors, but it induces a split in the critical path that balances timing errors and distribute them along all outputs instead of only degrading MSBs in conventional circuits.

References

[1] B. Moons *et al.*, "An energy-efficient precision-scalable convnet processor in 40nm CMOS," in *Journal of Solid-State Circuits (JSSC), IEEE*, 2017.

[2] E. Nogues *et al.*, "A DVFS based HEVC decoder for energy-efficient software implementation on embedded processors," in *International Conference on Multimedia and Expo (ICME), 2015 IEEE*, 2015.

[3] V. Camus *et al.*, "Energy-efficient inexact speculative adder with high performance and accuracy control," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium*, 2015, pp. 45–48.

[4] X. Jiao *et al.*, "Supervised learning based model for predicting variability-induced timing errors," in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, 2015, pp. 1–4.

[5] N. Zhu *et al.*, "An enhanced low-power high-speed adder for error-tolerant application," in *Integrated Circuits (ISIC), 2009 12th International Symposium*, 2009, pp. 69–72.

[6] J. Schlachter *et al.*, "Near/sub-threshold circuits and approximate computing: The perfect combination for ultra-low-power systems," in *Symposium on VLSI (ISVLSI), 2015 IEEE Comput. Society Annual*, 2015.

[7] V. Camus *et al.*, "A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision," in *Design Automation Conference (DAC), 2016 53rd ACM/EDAC/IEEE*, 2016.

[8] J. Schlachter *et al.*, "Automatic generation of inexact digital circuits by gate-level pruning," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium*, 2015, pp. 173–176.

[9] V. Camus *et al.*, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *European Solid-State Circuits Conference (ESSCIRC)*, 2016, pp. 465–468.

[10] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Microarchitecture (MICRO-36), 2003 36th Annual IEEE/ACM International Symposium*, 2003.

[11] M. Fojtik *et al.*, "Bubble razor: An architecture-independent approach to timing-error detection and correction," in *Solid-State Circuits Conference (ISSCC), 2012 IEEE International*, 2012, pp. 488–490.

[12] R. Ragavan *et al.*, "Adaptive overclocking and error correction based on dynamic speculation window," in *Symposium on VLSI (ISVLSI), 2015 IEEE Computer Society Annual*, 2016, pp. 325–330.

[13] J. Constantin *et al.*, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *Design, Automation, Test in Europe (DATE), 2015 IEEE Conf.*, 2015.

[14] S. Roy *et al.*, "Predicting timing violations through instruction-level path sensitization analysis," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, 2012.

[15] X. Jiao *et al.*, "Wild: A workload-based learning model to predict dynamic delay of functional units," in *Computer Design (ICCD), 2016 IEEE 34th International Conference on*, 2016.

[16] H. Cherupalli *et al.*, "Graph-based dynamic analysis: Efficient characterization of dynamic timing and activity distributions," in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*, 2015.

[17] V. Camus *et al.*, "Energy-efficient digital design through inexact and approximate arithmetic circuits," in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, 2015, pp. 1–4.

[18] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," in *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830.