Low-Power Variation-Aware Cores based on Dynamic Data-Dependent Bitwidth Truncation

Ioannis Tsiokanos, Lev Mukhanov and Georgios Karakonstantis
Institute of Electronics, Communications and Information Technology (ECIT), School of EEECS, Queen's University Belfast
Email: {itsiokanos01, l.mukhanov, g.karakonstantis}@qub.ac.uk

***Abstract*—Increasing variability of transistor parameters in nanoscale era renders modern circuits prone to timing failures. To address such failures, designers adopt pessimistic timing/voltage guardbands, which are estimated under rare worst-case conditions, thus leading to power and performance overheads. Recent approximation schemes based on precision reduction may help to limit the incurred overheads, but the precision is reduced statically in all operations. This results in unnecessary quality loss, since these schemes neglect the fact that only few long latency paths (LLPs) may be prone to failures, and such paths may be activated rarely. In this paper, we propose a variation-aware framework that minimizes any quality loss by dynamically truncating the bitwidth only for operands triggering the LLPs. This is achieved by predicting at runtime the excitation of the LLPs based on the processed operands. The applied truncation, which we implement by setting a number of least-significant bits to a constant value of zero, can effectively reduce the delay of the excited LLPs, providing sufficient timing slack to avoid failures without using conservative guardbands. To facilitate the adoption of such a scheme within pipelined cores and limit the incurred overheads, we also shape the path distribution appropriately for isolating the LLPs in a single pipeline stage. Additionally, to evaluate the efficacy of our framework, we perform post-layout dynamic timing analysis based on real operands that we extract from a variety of applications. When applied to the implementation of an IEEE-754 compatible double precision floating-point unit (FPU) in a 45nm technology, our approach eliminates timing failures under 8% delay variations with no performance loss. Our design comes at a cost of up-to 4.48% power and 0.34% area overheads, while the occasional operand truncation incurs minimal quality-loss in terms of relative error, up-to $4.1 \cdot 10^{-6}$. Finally, when compared to an FPU with pessimistic margins, our technique can save up-to 44.3% power.**

I. Introduction

The aggressive shrinking of transistor sizes has led to a 50% frequency variation [1] in advanced nanometer technologies. This trend renders circuits prone to failures and prevent them from meeting power and performance specifications [1], [2]. The quest for energy savings today and the scaling of supply voltage, which is one of the most effective technique to reduce power, further worsens variations and the resultant failures.

State-of-the-Art. One of the most popular variation-aware mechanisms is based on the adoption of timing guardbands that force the circuit to operate at a lower frequency or a higher voltage [1], [3], thus providing sufficient timing margins to mitigate variation-induced failures. However, such timing margins are considered to be overly pessimistic since they are estimated based on assumed rare operating conditions, thus incurring large power/performance overheads [2], [3].

To limit such overheads popular in-situ error detection and correction (EDAC) approaches were proposed that enabled operation beyond the always correct critical timing/voltage margins [4], [2], [5]. These design-centric schemes integrate special units/registers to monitor the critical long latency paths (LLPs) and either change the cycle time [4] or utilize different recovery mechanisms [2], [5] in case of detected errors. Although effective, such schemes make it difficult to meet the design constraints and may lead to large recovery overheads, especially if the activation probability of the error-prone LLPs is high [5]. Other recent work has tried to exploit the operand dependent dynamic path excitation [6] in pipelined cores. This work has shown the overly pessimistic estimation of timing paths by the conventional static timing analysis (STA) compared to the suggested use of dynamic timing analysis (DTA). However, the proposed clock frequency adjustment per instruction may be very challenging to be applied in practice.

Approximate computing has recently emerged as a concept that takes advantage of the error resilient properties of applications to tolerate a number of errors or approximate a number of less critical operations [7] for trimming down the traditional overheads. Recent schemes use precision scaling to reduce the energy consumption [8], while authors in [9] use it for mitigating the estimated delay increase over few years. Other approximate based schemes [10], [11], [12] truncate the bitwidth of all the input operands in simple data-paths, enabling power savings due to the reduced switching activity, while compromising accuracy. The common idea behind all the existing schemes lies on the static estimation of the potential delay increase under scaled voltage or variations and the reduction of the precision in arithmetic instructions at design time. By doing so, such schemes may incur unnecessary quality loss since they neglect the fact that in any circuit there are only few error prone LLPs [13], where the reduced precision can mitigate errors. In addition, such LLPs are excited by few rather rare combinations of input operands, further limiting the number of times that the precision scaling is necessary. In reality, there are many operands that activate only short latency paths (SLPs), which have inherently enough timing slack to address any potential delay increase and for which any reduced precision is redundant. More importantly, existing approximation schemes neglect the fact that the number of LLPs may vary across pipeline stages. This implies that the precision scaling level may need to be changed across stages to avoid an unnecessary accuracy loss, which may be critical for some operations [7].

In this paper, we depart from the existing schemes and introduce an occasional data-dependent bitwidth truncation, which is applied only 'when and where' is needed by exploiting the dynamic excitation of paths in pipelined units.

To demonstrate the efficacy of our approach, we applied it to an IEEE-754 compliant floating-point unit (FPU) [14]. Such FPUs are excellent representatives of pipelined designs where any approximation has a direct impact on accuracy. The contributions of this work can be summarized as follows:

- We develop a framework for tolerating variation induced failures by dynamically truncating the bitwidth only for operands that trigger the LLPs. We achieve this by implementing a unit that predicts the LLPs excitation at runtime.
- We implement the occasional bitwidth truncation in pipelined cores by setting a number of least-significant bits (LSBs) of specific critical operands to a constant zero value. This allows us to effectively reduce the computation delay of the critical LLPs along with their excitation probability.
- We facilitate the adoption of the proposed scheme within pipelined units by shaping the path distribution, such that the number of the LLPs is significantly reduced and all these paths are isolated to a single pipeline stage. The advantages of such an approach are two-fold. First, it reduces the critical LLPs and thus the failure probability; and second, it limits the number of stages where any LLPs prediction unit need to be integrated, thereby limiting the incurred overheads.
- We estimate the bit error rate (BER) under potential worst-case delay increase levels for the original and the proposed FPUs using our developed DTA tool. Based on the BER, we also evaluate the quality loss quantified in terms of the relative error for various applications.
- We demonstrate that the proposed FPU can lead to 40.89% power savings on average at a cost of 0.34% area and up-to $4.1 \cdot 10^{-6}$ quality loss in terms of relative error compared to an error-free guardband based FPU.

The rest of the paper is organized as follows. Section II presents the background and the motivation of our work, while Section III discusses the proposed approach and the implemented design flow. In Section IV, we apply the proposed framework to the design of an FPU; and Section V presents the experimental results. Conclusions are drawn in Section VI.

II. Background and Motivation

Typically, a data-path consists of a set of N unique combinational paths $P = \{p_1, p_2 ... p_N\}$, which are characterized by their delays $D(p_i)$ for $i = 1, 2 ... N$. The longest path determines the clock period, such as:

$$CP_{STA} = \max_{\mathbf{p} \in \mathbf{P}} \{D(p)\} \tag{1}$$

The overall path set can be distinguished into a set of K SLPs, which we define as: $P_{SLP} = p_1^{SLP}, p_2^{SLP} ... p_K^{SLP} \subset P$ and a set of M LLPs: $P_{LLP} = p_1^{LLP}, p_2^{LLP} ... p_M^{LLP} \subset P$. Assume that P_{SLP} is excited by a set of operands O_{SLP} and can be completed within a time $T_{SLP} = \max\{D(P_{SLP})\}$, whereas P_{LLP} is activated by a set of operands O_{LLP} and requires more time than T_{SLP} (i.e. $T_{SLP} < D(P_{LLP}) \leq CP_{STA}$) to be completed. In case of a delay increase triggered by variations or scaled voltages, paths from P_{LLP} are more susceptible to failures, as they may not have enough time to be completed correctly (i.e. $D(P_{LLP}) > CP_{STA}$).

Approximation based schemes try to reduce the potential failures by reducing the precision and thus the time required

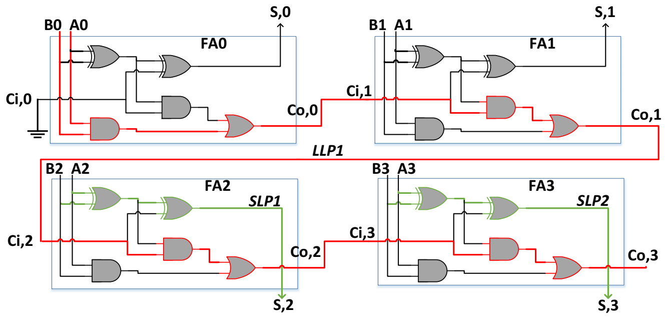


Fig. 1: Data-paths in a 4-bit ripple carry adder.

to complete computation paths [10]. A representative method lies on setting some of the input LSBs to zero. To elucidate the impact of bitwidth truncation on the computational delay, we consider a simple 4-bit ripple carry adder (RCA), as shown in Fig. 1. This adder consists of 4 full adders (FA). FA is a logic circuit that adds two input operand bits (Ai,Bi) plus a Carry in bit (Ci,i) and generates a Carry out bit (Co,i) and a sum bit (S, i). In such a design, the most timing critical path LLP1 (emphasized in red) will be activated when the carry propagates all the way from Ci,0 to Co,3. If we define the gate delay as T, then the LLP1 requires a delay equal to $8T$ to be completed, since the LLP1 will have to travel from the AND gate (emphasized in red) in FA0 down to XOR gate (emphasized in red) in FA3. According to (1), the minimum CP_{STA} of this data-path is $8T$. Such a delay will be activated only in the case of the operands belonging to O_{LLP}. For instance, when A=1111 and B=0001, then the carry generated in the first bit position (i.e. LSB) is propagated all the way to the final bit position, exciting the LLP1. Under an assumed delay increase, this path will fail as $D(LLP1) > 8T$ and thus $D(LLP1) > CP_{STA}$. However, by modifying the inputs and inserting 0 in the last 2 bits, such as: A=1100 and B=0000, there is no carry propagation and thus only SLPs (SLP1 and SLP2 highlighted in green) will be excited. Essentially, by truncating the last 2 bits of the input operands, we reduce the delay of the excited paths down to $2T$.

Nonetheless, the bitwidth truncation comes at a quality loss, a degree of which may be tolerated by some applications [10], [11], but there is room to limit it substantially. Any reduction of the truncation induced quality loss may be beneficial for many applications, especially for those, where few operations play a significant role in determining the output quality [7]. The unnecessary quality loss is attributed to the fact that the majority of the existing approximation based schemes reduce the precision of all operands statically, neglecting the fact that only few operands from O_{LLP} are actually exciting the error-prone LLPs. Conversely, the majority of the proceed operands belongs to O_{SLP}, which have sufficient timing slack to handle any delay increase without obtaining errors. Hence, there is no need to apply bitwidth truncation to operands from O_{SLP} and inflate the quality loss. The challenge in limiting any unnecessary quality loss lies on the need to dynamically identify operands from O_{LLP} and apply bitwidth truncation or any other approximation only to them. Achieving the above at low cost is further complicated within pipelined designs, where the LLPs may be distributed within any stage, thus requiring the adoption of several path excitation detection units.

III. Proposed Approach and Design Flow

The main goal of this work is to limit the unnecessary quality loss incurred by traditional timing failure avoidance schemes through occasional bitwidth truncation. The first step of our approach is to identify at runtime operands from O_{LLP} by exploiting the dynamic data-dependent sensitization of combinational paths. Upon detection of such operands, we deliberately set to zero a number of LSBs, allowing LLPs to be completed under various potential delay increase levels. Note that in the case of O_{SLP}, there is a sufficient timing slack to deal with any delay variations, and thus no approximation is applied. Therefore, we apply the bitwidth truncation only to the rather rare operands from O_{LLP}, whereby we significantly reduce the incurred quality loss. Another contribution of our work lies on enabling the occasional bitwidth truncation within pipelined computational cores. This is achieved by applying a LLPs isolation step, which essentially facilitates the adoption of a LLPs prediction unit ($LLPPU$) only where is needed within the pipeline. Our approach can be used to address any delay increase, while minimizing the quality loss due to potential timing failures and the applied approximations. In the next paragraphs, we present the main steps of our approach.

A. *Data-Dependent Runtime Prediction of O_{LLP}*

The aim of this step is to take advantage of the dynamic sensitization of combinational paths and try to identify O_{LLP} at runtime. While considering arithmetic units such as the adder in Fig. 1, carry propagation is the main bottleneck in the performance/delay of these systems. As it is explained in the 4-bit RCA example in Section II, by implementing a low-complex circuit that monitors the carry propagate signal at the middle of such a data-path, i.e. at the 2rd Full-Adder (FA1), which is given by: $(A0 \oplus B0) \cdot (A1 \oplus B1)$, we can identify if any LLP is going to be sensitized. In general, we can monitor $m-n$ bits (with $m > n$) and detect if carry propagates across the m^{th} bit, using the following logic:

$$F(m,n) = (A_m \oplus B_m) \cdot (A_{m-1} \oplus B_{m-1}) \cdot ...(A_n \oplus B_n) \quad (2)$$

Only when F evaluates a value of 1 the carry bit propagates from the n^{th} to the m^{th} bit into the addition result. In the case of F equals to 0, $SLPs$ are activated as there is no carry propagation across the m^{th} bit and the effective computation time is the maximum of the following two delays: one from the 0 to m^{th} bit and the other from m^{th} to the most significant bit (MSB). In the above equation, there is a trade-off between m and n, area/power and the accuracy of $F(m,n)$. If we increase the number of bits being monitored, then the excitation probability of the $LLPs$, which we refer as $P(L)$, decreases. For instance, the carry propagation probability across the 1-bit FA1 in the 4-bit RCA example is $2(1-p')p'$, where p' denotes the input signal probability [4]. However, if we select to monitor the carry propagate signal over 3 FAs (e.g. from FA0 to FA2), then the carry propagation probability and, consequently $P(L)$ can be lowered down to $[2(1-p')p']^3$. As we will explain in section IV.B, we exploit such property by finding the right balance between $P(L)$ and the incurred area/power overheads. We note that a similar prediction scheme has been demonstrated in [4]. However, authors focus on simple integer arithmetic units rather than on pipelined FPUs.

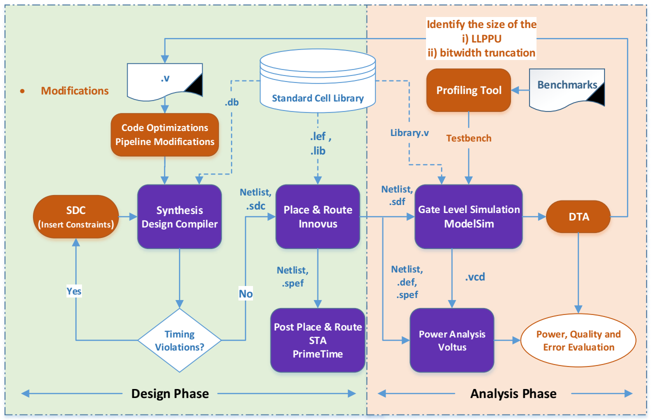


Fig. 2: Workflow of the proposed approach. The introduced modifications are highlighted in orange.

B. *LLPs Isolation in a Pipeline*

A straight-forward application of the proposed $LLPPU$ in pipelined designs is hindered by the optimizations performed in a performance-centric electronic design automation (EDA) flow. In fact, modern designs are optimized for power and area, subject to a frequency constraint: the LLPs are optimized by gate up-sizing, while the SLPs are allowed to become near-critical for recovering any area or power costs. This leads to a so-called ”timing wall” phenomenon, where many paths, across different stages, are close to the worst-case delay [13]. In such a path distribution, any prediction scheme would have to be applied to every stage to detect the LLPs excitation. To overcome such a phenomenon, we shape the path distribution appropriately for isolating P_{LLP} to a single stage.

C. *Design Flow*

The proposed approach is realized by utilizing state-of-the-art EDA tools as depicted in Fig. 2. The proposed flow consists of a design phase and an analysis phase.

1) Design Phase: To smooth the timing wall phenomenon, we set path-groups constraints during synthesis in the Synopsys Design Constraints (SDC) file. By introducing multiple path-group constraints, we force the synthesis tool to avoid optimizations that make naturally fast paths slower for saving area/power. These constraints may have an impact on the area and power consumption, but the resulting overheads can be kept small. Initially, we apply strict constraints to shift the timing wall away from the target CP_{STA}. If there are timing violations after synthesis, we relax the design constraints and re-run our method until the timing target is met. The above iterations are being implemented using tool command language (tcl) scripts, which after every synthesis round sort the paths into the two different sets, the P_{SLP} and the P_{LLP} based on the worst-case timing within the resulted groups. As a result, this procedure decreases the number of the LLPs, while ensuring all other paths are fast enough to tolerate timing failures. The synthesis step is followed by the place & route (PnR) step using the Innovus Tool from Cadence. Sign-off STA will follow with Synopsys PrimeTime.

2) Analysis Phase: To enable characterization of the data-dependent path activation, we use the post-place, gate-level simulation supported by ModelSim. Using this tool, we monitor the inputs and the outputs of all flip-flops in the design and generate a corresponding event log. To obtain this information, ModelSim requires a RTL netlist, a testbench and a standard delay format (SDF) file which describes the cell and interconnect delays. We obtain the RTL netlist and the SDF file after the PnR step. While our profiling tool feeds ModelSim with real FP operands, creating the required testbench. Providing that every set of operands under nominal conditions produces an error-free output, we define this simulation output as $Dgold$. Such an analysis phase also helps to extract BERs and an essential for power analysis value change dump (VCD) file. This file contains information about the switching activity and value changes occurred during the simulation.

IV. Case study: Application to an FPU

We apply our approach to a multicycle, IEEE-754 compatible double precision FPU, following the representation: $-1^S \times M \times 2^E$, where S : sign, E : exponent and M : mantissa. In a double precision FP number defined by the IEEE-754 Standard, the MSB indicates the sign, the next 11 bits represent the exponent and the mantissa consists of the last 52 bits. The applied FPU is a part of the Out-of-Order mor1kx MAROCCHINO pipeline, a 5-stage pipeline microprocessor based on the OpenRISC 1000 Instruction Set Architecture [15]. In our framework, the folowing FP instructions are implemented: addition/subtraction, integer to FP and FP to integer conversions and finally, comparison between FP numbers.

The targeted FPU consists of 6 pipeline stages and the micro-architecture of this unit is presented in our previous work [16]. At Stage 1, an Order Control Buffer and a Pre-Normalize block are implemented, which detect data dependencies and adjust the exponent and mantissa, respectively. Stage 2 executes the pre-addition/subtraction alignment. Stage 3 performs the multiplexing and shifting of the operands. Mantissa addition and exponent update are performed at Stage 4; while rounding is implemented in the last two stages.

A. Redesigned FPU

We first apply the typical EDA flow to the original unmodified design. This design is implemented using the typical corner of the CCS NanGate 45 nm cell library (@1.1V) [17]. After performing post-PnR STA, we built the path distribution, within each pipeline stage, as shown in Fig. 3a. The obtained distribution implies that the conventional performance-centric flow incurs the timing wall. Fig. 3a also reveals that the timing wall occurs in 4 out of the 6 stages; thereby the likelihood of timing failures at these stages is very high.

To reduce the impact of the timing wall, we impose timing constraints by grouping paths based on their latency. Such a method reduces the number of the $LLPs$, as shown in Fig. 3b. By combining the above constraints with simple RTL optimizations, we managed to isolate the LLPs to a single stage. In particular, rounding, which takes place at Stages 5 and 6 poses a bottleneck, as it is applied to the result of all FP operations, and thus many paths in these stages belong to P_{LLP}. To this end, we optimize some large conversions to

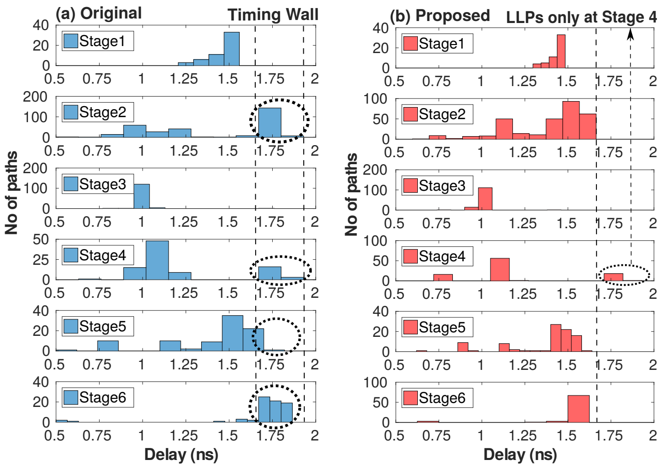


Fig. 3: (a) Original and (b) proposed path distribution across stages.

negative number occurring at Stage 6. After applying these micro-architectural changes, we isolate the LLPs to Stage 4 (see Fig. 3b). In other words, the few remaining LLPs are isolated in such a way that they can be triggered only by FP addition/subtraction instructions at Stage 4.

B. Implementation of the $LLPPU$

Since all the LLPs are restricted to Stage 4, we deploy the $LLPPU$ only at Stage 3. To predict the LLPs activation in clock cycle i (CC_i), the $LLPPU$ should indicate from the previous clock cycle (CC_{i-1}) if the input set for Stage 3 will trigger a LLP at Stage 4. Specifically, the $LLPPU$ produces a signal (F) at Stage 3 which is handled at Stage 4. If this signal is 1, then the in-flight instruction and related operands activate LLPs. In this case, we enable the bitwidth truncation of the involved operands. Otherwise, a SLP is activated, and we use the original data without truncation.

Given that Stage 4 implements the 52-bit mantissa addition, we deploy the $LLPPU$ that is explained in Section III.A. The F signal that denotes the latency type of the operands is determined by (2). The probability of a carry propagation across the m^{th} bit (i.e. $F(m, n) = 1$) is equivalent to the $P(L)$. As we discussed, $P(L)$ can be decreased by increasing the number of monitored bits. However, a higher number of monitored bits is accompanied by an increase of the $LLPPU$ area and high power overheads. Table I provides the $P(L)$ that we quantified using simulations with different sets of m,n and 40000 real FP operands. Following this table, we have chosen to monitor 7 bits (i.e. bit 27 to 33) for each operand, since $P(L)$ is only 1.1% and the area/power overheads are negligible for this choice. Note that the overheads incurred by the $LLPPU$, as well as the applications used for extraction of the operands, are discussed in the next section.

TABLE I
EXCITATION PROBABILITY OF THE LLPs ($P(L)$)

(m, n)	(27, 34)	(27, 33)	(28, 33)	(28,32)
$P(L)(\%)$	0.62	1.1	2.71	4.98

C. Application of Dynamic Operand Truncation

After redesigning the FPU and identifying the operands that activate the few $LLPs$, we set constant 0 values to the LSBs of O_{LLP} in order to minimize the $LLPs$ excitation probability. The number of the truncated bits needs to be carefully selected as the quality loss grows with the number of truncated bits, while path delays reduce accordingly with the number of zeroed LSBs. Based on DTA with different number of truncated bits, we found that setting the last 40 LSBs of the long latency operands to zero provides a balanced option between delay reduction and quality degradation.

V. Evaluation Results

In this section, we evaluate the efficacy of our approach in eliminating the timing failures under various clock reduction (CR) levels, which are listed in Table II. We reduce the clock period from 1.85ns to 1.65ns in steps of 0.05ns, representing potential degrees of the worst-case path delay increase [18]. We also measure the area and power overheads incurred by our design ($Prop$) compared to the original unmodified FPU ($Orig$). To estimate the quality gains of the proposed dynamic truncation, we apply the static truncation of 40 LSBs of the FP operands to the original FPU (refer to as $Trunc$). Finally, we estimate any possible gains by exploiting the available timing slacks observed in our design compared to the FPU using the guardband based scheme (refer to as $Guard$).

TABLE II
CLOCK REDUCTION LEVELS (WORST-CASE DELAY INCREASE)

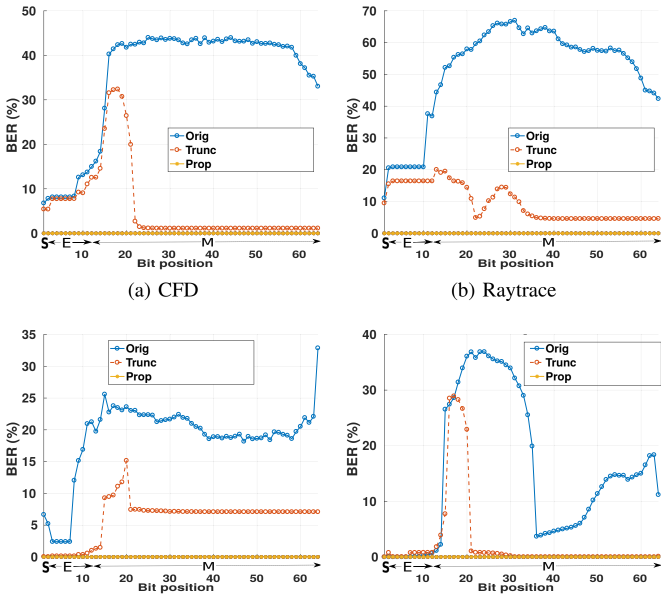
CR	CR0	CR1	CR2	CR3	CR4
clock, (ns)	1.85	1.80	1.75	1.7	1.65
reduction, (%)	0	2.7	5.4	8	10.8

A. Application Profiling

To evaluate the efficacy of our approach using realistic FP operands, we modified a profiling tool [19] to extract program traces from various applications. We use the K-means, CFD and Heartwall benchmarks from the Rodinia suite; and the Raytrace benchmark from the Parsec suite. These benchmarks represents a variety of algorithms that covers a wide range of domains, i.e. Data Mining, Fluid Dynamics, Medical Imaging and Computer Graphics. To obtain the program traces, we profile all benchmarks on an ARM A7 based system, Odroid-Xu3. Using such a tool, we extract 10000 operands from the most frequently executed FP instructions for each application, which we feed to the DTA tool to estimate the BERs.

B. Evaluation of BER and Quality

1) BER: We quantify the efficacy of our approach in eliminating the timing failures by estimating the output BER, which depends on the input data and the assumed worst-case delay increase. Fig. 4 shows the BER in the sign, mantissa and exponent bits of $Orig$, $Trunc$ and $Prop$ at CR3. We estimate the BER by comparing the simulated output and the error-free output $Dgold$. We make several interesting observations in that figure. Firstly, distinct bit positions incur different BERs. This happens because different input operands activate different paths. Secondly, the original design exhibits significantly high BERs, ranging from 1.15% up-to 66.9% in the mantissa and up-to 37.6% in the exponent. If we statically set 40 LSBs to 0 in the original FPU, the BERs are reduced. However, we observe that the BER of the MSBs of mantissa remains considerably high (up-to 32%), while the BER of the exponent bits ranges from 0% up-to 15%. The fact that in both $Orig$ and



(a) CFD (b) Raytrace

(c) K-means (d) Heartwall

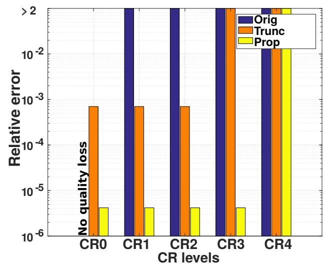
Fig. 4: BERs under CR3 across (a) CFD, (b) Raytrace, (c) K-means and (d) Heartwall benchmarks.

$Trunc$ the failing bits occur in the exponent and the MSBs of the mantissa may result in a catastrophic quality loss. By contrast to the original FPUs, $Prop$ eliminates the BERs and minimizes any quality loss.

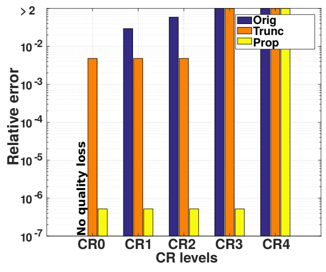
2) Quality: To measure the quality loss incurred by the random timing failures as well the static and the dynamic truncation, we estimate the average relative error (RE) of our design and compare it with the RE of $Orig$ and $Trunc$ for all the applications. The average relative error is defined as:

$$RE = \frac{\sum_{i=1}^{K} \left| \frac{D_{gold}(i) - O_{sim}(i)}{D_{gold}(i)} \right|}{K} \tag{3}$$

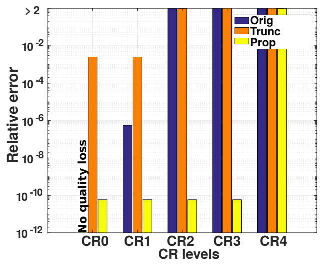
where D_{gold}(i) and O_{sim}(i) denote the error-free output value and the output value obtained by the simulation using our DTA tool for a specific (i) FP instruction and the associated operands, respectively. The O_{sim}(i) value is extracted by the output register of the considered FPU after simulating $Orig$, $Trunc$ and $Prop$ under a specific CR and truncation level. For these experiments, we extract 10000 FP instructions for each benchmark and thus i varies from 1 up-to K = 10000. The effect of different CR levels and the bitwidth truncation on the RE before and after applying our approach is depicted in Fig. 5. As shown, the original design without any truncation under the nominal clock period (CR0) introduces no quality degradation since no failures have been manifested. In the case of CFD, we can observe that $Orig$ leads to unacceptable (> 2) RE levels even under a small worst-case delay increase (i.e. CR1). Static truncation helps to reduce the RE to approximately 0.001 under CR1 and CR2, while $Prop$ limits this quality loss to $4 \cdot 10^{-6}$ under CR2 and CR3. Similar results are obtained for Raytrace, where the proposed FPU introduces a negligible RE ($5 \cdot 10^{-7}$) for the vast majority of the CR levels. In the case of K-means, we observe the lowest RE among all benchmarks ($6 \cdot 10^{-11}$) under CR1, CR2 and CR3 for the proposed design, whereas for $Orig$ and $Trunc$, the



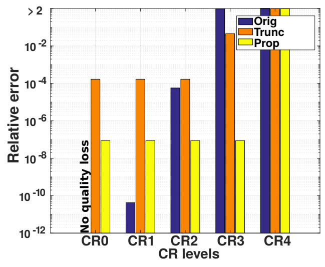
(a) CFD



(b) Raytrace



(c) K-means



(d) Heartwall

Fig. 5: Relative errors under different CR levels across (a) CFD, (b) Raytrace, (c) K-means and (d) Heartwall benchmarks.

quality degrades significantly after CR1. Finally, in the case of Heartwall, we can see that $Orig$ incurs a lower RE than the static or the dynamic truncation under CR1. Nonetheless, under CR2 and CR3 our approach obtain at least 99.9% lower RE when compared to $Orig$ and $Trunc$.

Overall, $Prop$ exhibits a significantly lower quality loss than $Orig$ under CR1, CR2 and CR3. Compared to $Trunc$, $Prop$ achieves up-to $4.2 \cdot 10^7 \times$ lower RE across the considered benchmarks. In addition, beyond the CR3 level, we notice a significant quality degradation in the original and the proposed designs (see Fig. 5). In particular, Fig. 3 shows that there are many paths with delay more than 1.65ns; this outcome implies that the probability of massive timing violations and thus quality degradation under CR4 is high.

C. Power and Area Penalty

The LLPs isolation step and the integration of the $LLPPU$ introduce area and power overheads, which we quantify using our design flow. Results show that $Prop$ incurs 0.34% area and 4.48% on average power overheads compared to $Orig$.

D. Comparison with a Guardband Based Scheme

In this paragraph, we compare $Prop$ with $Guard$. According to the conventional paradigm, $Orig$ adopts enough timing guardbands by up-scaling the voltage to avoid failures and obtain an error-free output. Using the available fast corner cell library (@1.25V) and the extracted VCD files, that allows us to consider the switching activity and dynamic path activation, we estimate the power consumption across all applications. Operation at such a voltage provides the necessary margins to avoid all the dynamic timing failures. However, as shown in Table III, it comes at a cost of up-to 44.3% power overhead when compared to $Prop$ (@1.1V), which, as depicted in Fig. 4, leads also to an error-free output.

TABLE III
POWER SAVINGS ACROSS ALL BENCHMARKS IN THE PROPOSED DESIGN COMPARED TO A GUARDBAND BASED FPU

Benchmarks	K-means	CFD	Heartwall	Raytrace
Power gains (%)	41.73	38.78	44.3	38.76

VI. Conclusion

In this paper, we presented a framework to avoid variation-induced failures in pipelined cores by dynamically truncating the bitwidth of the operands that activate the error-prone $LLPs$. To this end, we adopt a unit that detects the operands triggering the LLPs and effectively reduces the path-delay by setting some LSBs only of these operands to a constant value of 0. To facilitate the adoption of such a scheme within pipelined cores and limit the incurred overheads, we modify the path distribution appropriately to reduce the $LLPs$ and constrain such paths to a single pipeline stage. The evaluation of our approach indicates that the proposed design eliminates timing failures under potential delay variations with negligible overheads. When compared to the conventional guardband approach, our design saves 40.89% of power on average with a 0.34% area overhead and no performance loss. Our results show that the occasional truncation of the 40 LSBs of specific operands introduces a quality loss of up-to $4 \cdot 10^{-6}$ RE, which is significantly less than the quality loss (up-to $5 \cdot 10^{-3}$ RE) incurred by the static 40 LSBs truncation of all the operands.

Acknowledgments

The presented research effort is partially supported by the European Community Horizon 2020 programme under grant no. 688540 (UniServer) and grant no. 732631 (OPRECOMP).

References

[1] P. Gupta *et al*, "Underdesigned and opportunistic computing in presence of hardware variability," *TCAD*, 2013.
[2] D. Bull *et al.*, "A power-efficient 32 bit arm processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation," *JSSC*, Jan 2011.
[3] G. Karakonstantis *et al.*, "Containing the nanometer "pandora-box": Cross-layer design techniques for variation aware low power systems," *IEEE JETCAS.*, vol. 1, no. 1, pp. 19–29, 2011.
[4] S. Ghosh *et al.*, "Voltage scalable high-speed robust hybrid arithmetic units using adaptive clocking," *TVLSI*, 2010.
[5] Y. Zhang *et al.*, "irazor: Current-based error detection and correction scheme for pvt variation in 40-nm arm cortex-r4 processor," *JSSC*, 2018.
[6] J. Constantin *et al.*, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *DATE*, 2015, pp. 381–386.
[7] V. K. Chippa *et al.*, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC*, 2013, pp. 1–9.
[8] G. Tagliavini *et al.*, "A transprecision floating-point platform for ultra-low power computing," in *DATE*, 2018, pp. 1051–1056.
[9] H. Amrouch *et al.*, "Towards aging-induced approximations," in *54th ACM/EDAC/IEEE DAC*, June 2017, pp. 1–6.
[10] S. Narasimhan *et al.*, "Healing of dsp circuits under power bound using post-silicon operand bitwidth truncation," *IEEE TCAS I*, 2012.
[11] J. Schlachter *et al.*, "Design and applications of approximate circuits by gate-level pruning," *IEEE TVLSI*, vol. 25, no. 5, pp. 1694–1702, 2017.
[12] I. Tsiokanos *et al.*, "Minimization of timing failures in pipelined designs via path shaping and operand truncation," in *IOLTS*, 2018, pp. 171–176.
[13] A. B. Kahng *et al.*, "Slack redistribution for graceful degradation under voltage overscaling," in *ASP-DAC*, 2010.
[14] IEEE 754-2008 Standard for Floating-Point Arithmetic.
[15] OpenRISC, "OpenRISC 1000 architecture manual".
[16] I. Tsiokanos *et al.*, "Variation-aware pipelined cores through path shaping and dynamic cycle adjustment: Case study on a floating-point unit," in *ISLPED*, 2018, pp. 52:1–52:6.
[17] NanGate FreePDK45, http://nangate.com.
[18] X. Jiao *et al.*, "Slot: A supervised learning model to predict dynamic timing errors of functional units," in *DATE*, 2017, pp. 1183–1188.
[19] L. Mukhanov *et al.*, "Alea: Fine-grain energy profiling with basic block sampling," in *PACT*, 2015.