Design of Approximate Circuits by Fabrication of False Timing Paths: The Carry Cut-Back Adder

Vincent Camus, *Student Member, IEEE*, Mattia Cacciotti, *Student Member, IEEE*, Jeremy Schlachter, *Student Member, IEEE*, and Christian Enz, *Senior Member, IEEE*

***Abstract*— This paper introduces a novel method for designing approximate circuits by fabricating and exploiting false timing paths, i.e., critical paths that cannot be logically activated. This allows to strongly relax timing constraints while guaranteeing minimal and controlled behavioral change. This technique is applied to an approximate adder architecture, called the Carry Cut-Back Adder (CCBA), in which high-significance stages can cut the carry propagation chain at lower-significance positions. This lightweight approach prevents the logic activation of the carry chain, improving performance and energy efficiency while guaranteeing low worst-case errors. A design methodology is presented along with implementation, error optimization, and design-space minimization. The CCBA is proven capable of extremely high accuracy while displaying significant circuit savings. For a worst case precision of 99.999%, energy savings up to 36% are demonstrated compared with exact adders. Finally, an industry-oriented comparison of 32-bit approximate and truncated adders is carried out for mean and worst-case relative errors. The CCBA outperforms both state-of-the-art and truncated adders for high-accuracy and low-power circuits, confirming the interest of the proposed concept to help building highly-efficient approximate or precision-scalable hardware accelerators.**

***Index Terms*— Low-power digital circuits, timing optimization, false timing paths, approximate circuits, approximate adders, speculative adders.**

I. Introduction

PERFORMANCE, density and energy efficiency of integrated circuits have been increasing exponentially for the last four decades following Gordon Moore's remarkable prediction. However, power and reliability pose several challenges to the future of technology scaling. Power has definitely emerged as a critical concern due to the poor scaling of V_{th}, while transistor miniaturization reaching atomic scale has led to tremendous Process-Voltage-Temperature (PVT) variations. Unfortunately, achieving low power and robustness against variability requires complex and conflicting design constraints. For example, while power efficiency calls for voltage down-scaling and minimization of hardware, robustness demands higher voltage, larger transistors and additional correction or redundancy. As a result, designers are being pushed to seek new energy-efficient computing techniques to meet the increasing demand of data processing.

The concept of *error tolerance*, i.e. accepting error in a design to save resources, is well known in many abstraction layers and is already implicit in digital signal processing as the representation of real numbers is approximated due to the finite number of bits. Built on these ideas, *approximate computing* [1] has emerged as a promising candidate to improve performance and energy efficiency beyond technology scaling. Designing *approximate circuits* explores a new trade-off, not only by accepting unreliability, but by intentionally introducing errors to save area and power and overcome the limitations of traditional circuit design.

With the exploding amount of data being processed in the cloud and on mobile devices, a wide range of applications can trade accuracy without compromising the functionality or the user experience. In multimedia applications, a small proportion of errors stays imperceptible to humans. The growing demand for statistical algorithms such as data mining, search and recognition represents another opportunity to compute in an approximate way as the outcome of those applications is not required to be a single golden result, but an adequate match. Finally, iterative applications like vision and tracking are inherently resilient to errors since those can be compensated in the succeeding frames or steps.

Approximate computing has been investigated at different levels of abstraction, such as voltage-frequency-precision scaling at circuit level [2] or significance-driven computation at algorithmic level [3]. Another way consists in redesigning the architecture of digital circuits into an approximate version with smaller delay, silicon area or power consumption. This technique is particularly suited for arithmetic operators such as adders.

This paper introduces a novel concept to optimize arithmetic circuits by artificially inserting and exploiting false paths, and co-designing circuit implementation together with circuit functionality. This technique is demonstrated for the design of an approximate adder trading off arithmetic precision in a floating-point manner. In this adder, called the Carry Cut-Back Adder (CCBA) and briefly introduced in [4],

Manuscript received January 7, 2018; revised May 11, 2018; accepted June 1, 2018. This work was supported in part by the Nano-Tera IcySoC Project of the Swiss National Science Foundation and in part by the ARTEFaCT Project of the French Agence Nationale de la Recherche. This paper was recommended by Guest Editor A. Marongiu. *(Corresponding author: Vincent Camus.)*

The authors are with the Integrated Circuits Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail: vincent.camus@epfl.ch; mattia.cacciotti@epfl.ch; jeremy.schlachter@epfl.ch; christian.enz@epfl.ch).

Source code of the Carry Cut-Back Adder is available online at https://github.com/vincent-camus/carry-cut-back-adder.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2018.2851749

the carry-chain activation is prevented in order to relax timing constraints in the entire design, thus strongly improving circuit efficiency. With this approach, high-significance carry stages are monitored to cut the carry propagation chain at lower-significance positions, guaranteeing low relative errors of floating-point type.

In this work, the CCBA architecture is enhanced with a new *input-induced* cut mechanism, improving error control and outperforming previous implementation in high-performance and high-accuracy circuits. It is provided with a detailed design strategy for efficient CCBA design including circuit implementation, error optimization and design-space minimization.

An exhaustive and industry-oriented comparison of 32-bit CCBAs against 10 state-of-the-art approximate adders, including truncated exact adders, is performed for two target frequencies. All circuits are described at behavioral level and synthesized in an industrial design manner for a 65 nm commercial CMOS technology. Mean and maximal relative errors are used to assess the accuracy. Results show that the CCBA offers by far the best performance for worst-case relative errors. For mean relative errors, the CCBA is proven to be comparable to truncated adders among high-accuracy circuits, and to outperform truncation in low-speed configurations.

The organization of this paper is as follows: Section II gives an overview of the concept of circuit optimization by fabrication of false timing paths. Section III details the architecture, arithmetic principle and design strategy of the CCBA. Section IV finally presents the CCBA results and its comparison against truncated adders and state-of-the-art approximate adders.

The source code of the CCBA circuit is available online at https://github.com/vincent-camus/carry-cut-back-adder.

II. Approximate Circuit Design and Optimization by Fabrication of False Timing Paths

Digital circuits need to be designed to be functional in the worst-case scenario, i.e. when their *critical path* is activated. This is achieved by finding all potential critical paths and adopting on them conservative—thus expensive—timing margins. But sometimes, it happens that a critical path *can never be logically activated*, it is then called a *false path*, as it is unnecessary to apply conservative constraints over it.

False paths are traditionally unexpected byproducts of circuit design. Finding them and obtaining their information, known as delay constraints or timing exceptions, makes it possible to relax timing constraints on signal paths during the Static Timing Analysis (STA). It can enable the synthesis tool to achieve the desired design performance (e.g. power, area, or speed) or timing closure by focusing efforts on real paths instead of false paths. Thus, many scientific articles [5], [6] and patents [7]–[9] have described techniques to identify them in the circuit netlist by analytical or numerical ways.

A. False-Path Fabrication

The novelty and main interest of this new technique is to artificially introduce and exploit false paths to optimize the implementation of digital circuits. Preventing full activation of a delay-critical signal in a circuit by inducing a false path allows more relaxed timing constraints, resulting in lower circuit implementation cost, higher yield, or earlier signal arrival times. In some cases, if a signal path originally fails to fit the delay constraint, this technique can make it possible to fit the constraint without the need to redefine design specifications, or without costly methods such as upsizing cells and transistors, buffering, parallelizing or increasing pipelining.

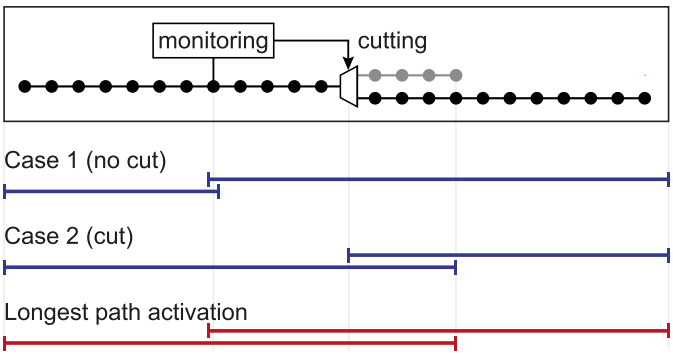

Fig. 1. Diagram illustrating a fabricated false path. The monitoring block tracks signals related to a stage of the original black-dotted signal path. When propagation is possible through this stage, it triggers the cutting multiplexer to select the gray path, resulting in a shorter effective propagation.

In order to create a false path or transform a signal path into a false path, we introduce two required logic elements:

- A *cutting* element multiplexing the signal path in its center, either to maintain the path itself, or to substitute it for an alternate path that is faster.
- A *monitoring* element triggering the cutting element to select the faster alternate path when it detects a possibility of full signal-path activation based on the monitoring of a few related nets.

Fig. 1 illustrates an example of fabricated false path. The monitoring block tracks a portion of the circuit related to the original signal path in order to detect potential propagation across it. If such risk is detected, it generates a signal that triggers the cutting element into selecting a shorter signal path. Independently of all the other (non-monitored) parts of the signal path, a case analysis of the longest possible logic propagation along the signal path results in shorter path activation, thus lower path delay.

An important step is to manually exclude the generated false paths from STA. This is the case for the CCBA circuit described in the following section, for which signaling timing exceptions is crucial to get the correct implementation. Indeed, identifying false paths in a circuit is a non-trivial and computer-demanding task. Thus, omitting to signal them to the synthesis tool is likely to lead to a miss. In that case, the tool would unnecessarily attempt to meet delay constraints on them, losing all the benefits of the technique. Different types of timing exceptions are possible, depending on the synthesis tool, for instance *set max delay*, *set false path* or *set disable arc*.

B. Significance-Driven Cuts

It is important to note that inducing or fabricating a false path requires a careful co-design of circuit timing together with circuit behavior. Indeed, the non-complete path activation will prevent its full and (assumed) original behavior to establish, thus altering the overall circuit behavior. For this reason, this technique is best suited for the design of approximate circuits, for example by transforming an exact design into an approximated version lacking full functionality.

The greatest challenge is undeniably to find the right signal path and to implement a cut that guarantees minimal and controlled changes on the functionality, such as reduced precision or controllable arithmetic errors. An evident application of this technique is thus on arithmetic operators and computing datapaths. Indeed, their intrinsic bit and net significance can help to guide the selection of monitoring and cutting elements. Furthermore, in arithmetic circuits, the critical path generally spans from low to high significance signals. Thus, the monitoring can be configured to track one or more high-significance signals, while the cutting would occur at a lower-significance position.

III. Application to Approximate Circuits: The Carry Cut-Back Adder

A. State-of-the-Art Approximate Adders

Additions are the most common arithmetic units in digital systems. With the demand for higher speed and power efficiency, many attempts have been made to build them in an approximate manner.

The most common approach to build approximate adders is using the concept of *carry speculation* [10]. As carry propagation typically does not cover the entire length of the adder, it is feasible to guess relatively accurately an internal carry based on a small number of preceding stages. As a result, the carry propagation chain can be reduced or sliced into multiple shorter paths executed in parallel, enabling performance beyond theoretical bounds of exact adders.

Different speculative schemes have been explored in the literature, among which *segmented* [11], [12], *compensated* [13]–[15] and *timing-starved* adders [16], [17]. As they contain the full addition stages (simply unconnected to be executed in parallel), all these adders can easily and inexpensively be made dynamically configurable to allow exact computation or variable-latency correction [11], [14], [16], [17]. This work solely considers their core architecture without such features.

Other architectural techniques are based on simplifying LSBs of the addition [18], [19], either by replacing the low-significance Full Adder (FA) cells by an approximate counterpart, or by pruning low-significance gates after circuit synthesis.

1) Speculative Segmented Adders: Early speculative adder works are based on the Equal Segmentation Adder (ESA) [11]. The ESA simply slices the addition into multiple sub-adder blocks executed in parallel, without carry propagation amongst themselves. This segmented carry chain without any circuit overhead offers a high energy efficiency at the cost of numerous and uncontrolled errors.

To reduce the error rate, the Error-Tolerant Adder type II (ETAII) [12] complements the sub-adders with equally-sized carry generator sub-blocks to speculate more accurately the input carry of each sub-adder.

2) Speculative Compensated Adders: In order to reduce the error impact and limit the worst case, segmented adders have been coupled with multiplexer-based error compensation. The Error-Tolerant Balancing Adder (ETBA) [13], direct descendant of the ETAII, uses an error balancing technique based on multiplexers to mitigate the relative error on the preceding sub-adder block in case of incorrect carry speculation.

The Generate-Signals-Exploited Carry Speculation Adder (GCSA) [14] has a functionally similar carry scheme as the ETBA. It differs by introducing, in case of incorrect speculation, the error reduction on the current sub-adder block rather than on the preceding one.

The Inexact Speculative Adder (ISA) [15] is a generalized and optimal architecture of speculative adder. It minimizes the carry-generator overhead, reducing its large critical delay. It also optimizes error reduction with a dual-direction compensation on both preceding and current sub-adder blocks. The ISA improves accuracy while strongly outperforming other speculative adders in terms of speed and energy efficiency. It is worth noting that the ISA encompasses the state-of-the-art segmented and compensated adders, those being boundary cases.

3) Speculative Timing-Starved Adders: The Almost Correct Adder (ACA) [16] is the best-known timing-starved adder. It is composed of an array of overlap-ping and translated sub-adder blocks, so that each sum bit of the ACA is constructed using exactly the same amount of preceding carry stages (except the first ones, which require less). The critical-path delay is thus limited, but the circuit cost is fairly high.

As for the ACA, the Accuracy-Configurable Approximate Adder (ACAA) [17] is also composed of overlapping sub-adder blocks. But those sub-adders are translated by half of their bit-width. Thus, fewer blocks are required and the circuit complexity is reduced.

4) Adders with Simplified LSBs: The Lower-part-OR Adder (LOA) [18] divides the addition into two parts. The upper part computes precise addition of the MSBs, while some OR gates approximate the lower-part addition instead of conventional FA cells. An extra AND gate is used to generate the carry-in of the upper part addition from the preceding stage. Despite a high error rate, error values remain small, while the critical path is reduced to the carry chain of the upper-part adder only.

Gate-Level Pruning (GLP) [19] belongs to the class of CAD tools to automate the design of approximate circuits [19]–[22]. GLP removes low-significance gates, trading accuracy in exchange for area and power savings. It has been successfully applied to adders, for which it retains the gates required for accurate carry propagation while discarding those used for generating low-significance outputs.

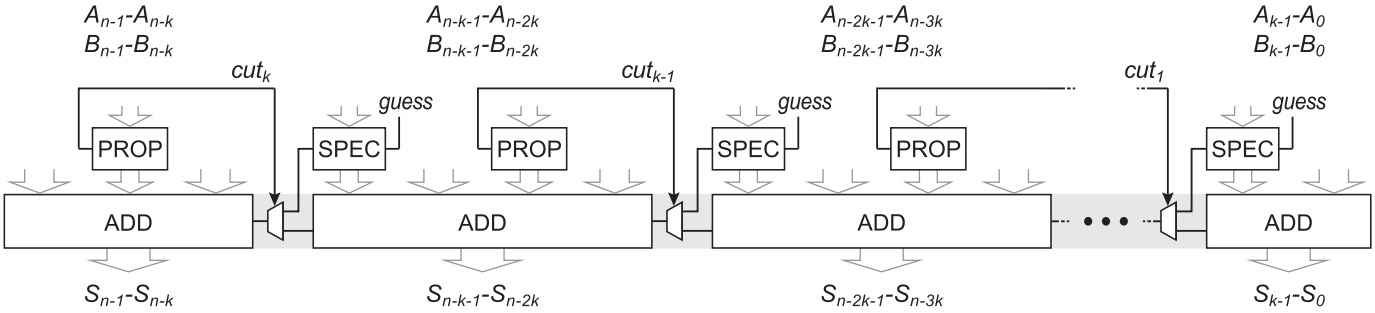

Fig. 2. General block diagram of the proposed Carry Cut-Back Adder (CCBA).

B. Proposed Architecture

The general structure of the proposed Carry Cut-Back Adder (CCBA) is depicted in Fig. 2. As presented in [4], the CCBA is built on an ordinary fixed-point adder composed of the chain of sub-adder blocks (ADD), with insertion of multiplexers that can cut the carry chain to shorten the effective critical path. The cut multiplexes the real carry with a carry speculated from a much shorter chain.

The decision to cut the chain is taken in the carry propagate block (PROP) that monitors a group of carry stages and generates the cut signal if those are all in propagate states. The cut always occurs at a lower-significance position than the PROP in the carry chain, guaranteeing low relative errors.

Note that this *cut-back* mechanism appears as a feedback between two carry-chain positions. But the PROP only uses local propagate signals, which are computed from the input operands and not from the carry chain. It is therefore not a recursive loop and cannot influence the circuit stability.

The speculated carry is generated in the optional carry speculator block (SPEC) with a *guess* value, identical if there are multiple cut-backs. Shorter than the exact carry path, this alternate path speculates the carry from a few preceding stages and propagates the *guess* if those stages are all in propagate mode. The *guess* is usually a hardwired '0' or '1', but it can be a dynamic value, such as a preceding-stage input operand to avoid a bias in the error distribution [13].

If there is no SPEC block (equivalent to a 0-bit SPEC), the multiplexer can be simplified to a monotonic gate, as in the *AND-cut* of Fig. 3a where $cut = 0$ dictates the AND output regardless of its second input. Another solution, called the *input-induced* cut, is to induce the cut from the input operands themselves, as in Figs. 3b-c where both stage inputs are zeroed or given the same value to allow kill or generate states only. A different HDL description is required for those latter implementations.

C. Circuit Timing

The main advantage of this approach remains in its timing characteristic. In a regular adder, the critical path is only activated if all the stages are in propagate mode. This occurs with a low probability as the carry propagation is naturally broken by the distribution of input bits. The adder within the CCBA physically contains the entire carry chain (through ADD blocks and multiplexers), but *this path can never be activated*. By monitoring a few stages of the adder, the PROP detects such risk and calls the SPEC as shorter path to be used instead, ensuring that the design meets tighter timing constraints.

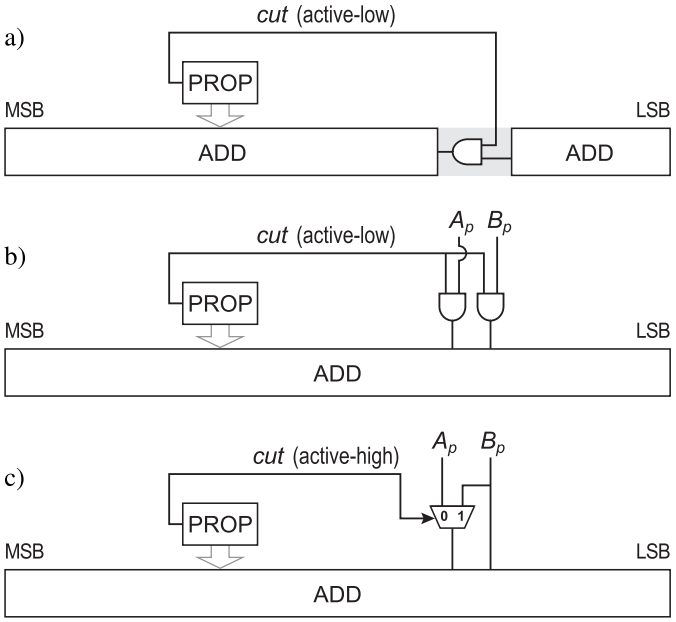

Fig. 3. Examples of CCBA without SPEC. (a) shows an AND-cut, equivalent to 0-bit SPEC with $guess = 1$. (b) and (c) show input-induced cuts. (b) sets the pth inputs to '0', forcing a kill state. (c) sets both inputs to the same value, allowing kill or generate states only.

Fig. 4 shows a case study of the longest propagation chains that can flow through a CCBA built with two OR-cuts enabled by active-high *cut* signals (example without SPEC for simplicity, but same reasoning in the general case). Each cut-back module splits the carry chain with two possibilities:

- *cut = 0:* The OR gate output follows the input, propagating the carry from one ADD block to the other. In this typical case, no intentional cut happens at the cut position, but the carry chain is naturally broken within the PROP among the ADD_2 stages. The critical path is therefore limited, as it cannot entirely cross over the PROP.
- *cut = 1:* All the stages within the PROP are in propagate mode. The carry necessarily propagates through the

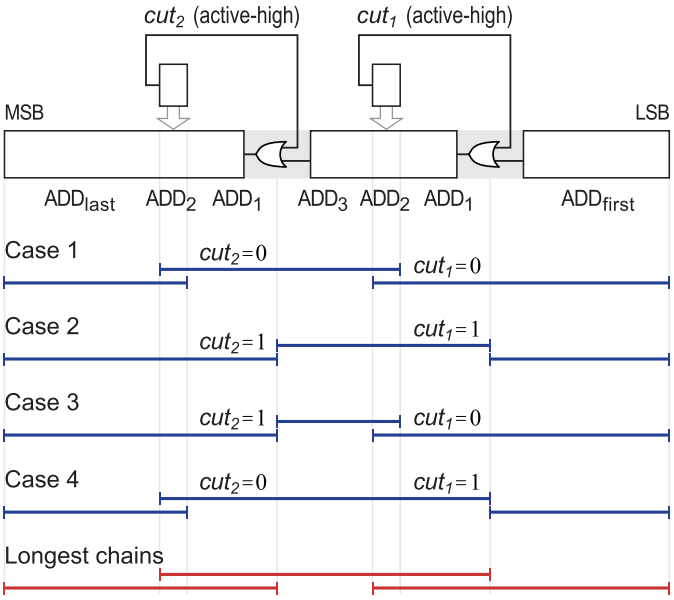

Fig. 4. Diagram of the longest carry chains and resulting effective critical paths in the example of an implementation of CCBA with two OR-cuts.

PROP and there is a risk of long critical-path activation if the other non-monitored stages are also in propagate mode. The active-high *cut* signal deliberately forces the OR output at '1' no matter the real carry coming from the preceding ADD block. The carry propagation is therefore interrupted at this position and its maximum length remains limited.

As explanation of Fig. 4, no intentional cuts occur in case 1, the carry chain being naturally broken within the two PROP blocks. Two deliberate cuts occur in case 2, artificially breaking the carry chain at the OR gate positions. Cases 3 and 4 both contain one naturally broken chain and one deliberately cut, shortening carry chains to various lengths.

Despite the fact that the full carry chain physically exists in the design, no input combination can activate it from the start to the end. It is therefore a *false path*, and it can be excluded from the timing optimization. The *effective* critical paths, in red in Fig. 4, sum up the longest propagate chains that can occur in the circuit among the different cases. Insertion of more carry cut-back modules, possibly overlapping each other, would lead to shorter effective critical paths.

D. Arithmetic and Errors

The CCBA addition arithmetic is illustrated in Fig. 5. Stages within PROP and SPEC blocks are indicated with their carry state: P, G and K, representing propagate, generate and kill states, respectively. Cuts and signals are drawn in dotted lines when cuts are inactive.

An error only occurs with the concurrence of three factors:

- A sequence of propagate signals spans the entire PROP bit-width, triggering the cut.
- A sequence of propagate signals spans the entire SPEC bit-width, making the exact carry prediction impossible with the SPEC stages only.

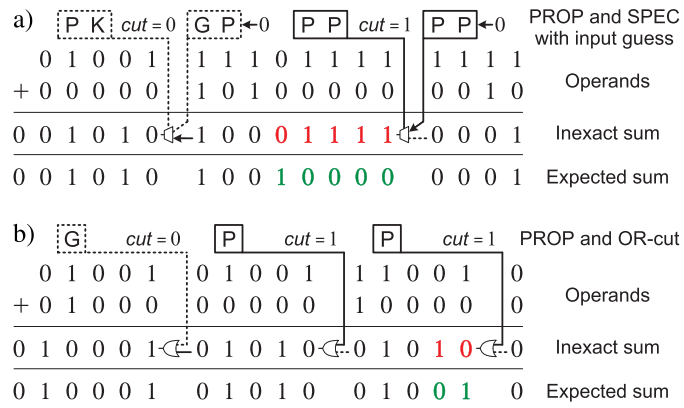

Fig. 5. Example of CCBA addition arithmetic for two circuit architectures: (a) two multiplexed cuts with 2-bit PROP and SPEC blocks and guess at '0', (b) three OR-cuts with 1-bit PROP blocks. Dotted lines depict inactive cuts.

- A wrong guess of the carry that inputs the SPEC (Fig. 5a) or that directly substitutes for the real carry (Fig. 5b).

Because of the simultaneous occurrence of the three aforementioned properties, an error occurs in the right-hand path of Fig. 5a. In the OR-cut implementation of Fig. 5b, the active-high *cut* signal is also the guess value due to the use of the OR gates. The first error condition is met for the two right-hand paths, directly triggering the cut since there is no SPEC. The guess value at '1' unintentionally follows the real carry in the central path and leads to a correct sum. But it happens to be wrong in the right-hand path and leads to a faulty sum.

Occurrence of an error implies that one or both operands have non-zero bits at the PROP position to drive those stages into propagate mode. As the error occurs at the cut position, at a lower-significance position, the expected sum is necessarily much larger than the introduced error. In the example of Fig. 5a, the absolute error is 16 while the expected sum is 43,265 so the relative error is 0.04 %. In the computation of Fig. 5b, the relative error is only 0.006 %. Such low relative errors are typical in speculative adders for calculations involving large value operands. However, it is the worst case that gives the upper-bound relative error and defines the minimum precision of an adder.

E. Worst-Case Error and Floating-Point Precision

1) Error Propagation: It is interesting to note from Fig. 5 that the error caused by the cut can propagate on many bits, but seems to keep the magnitude of the carry cut-back position, i.e. the first wrong bit. However, a series of erroneous bits can result in very different arithmetic errors due to compensation mechanisms. Thus, a careful demonstration has to be provided.

Let S_i, C_i and P_i denote the sum, carry-in and propagate signals of the ith stage addition, respectively. The sum and carry propagation are defined by:

$$S_i = P_i \oplus C_i \tag{1}$$

$$P_i = 1 \implies C_{i+1} = C_i. \tag{2}$$

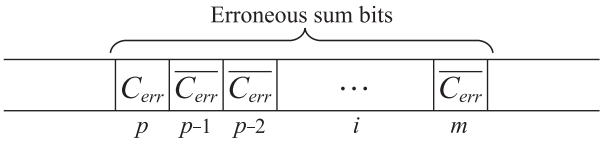

Fig. 6. Balanced error pattern.

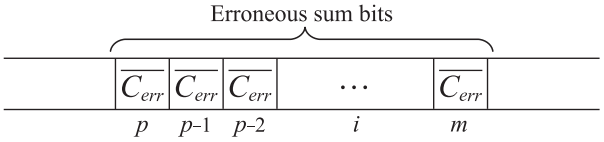

Fig. 7. Unbalanced error pattern.

Let assume a carry error at the ith bit of the adder, with an erroneous carry of value C_{err}. The sum bit and the carry-out depend on the value of P_i:

- If $P_i = 1$, (1) gives $S_i = \overline{C_{err}}$ instead of C_{err} for the expected sum bit, while (2) propagates the wrong carry C_{err} to the next stage, where the same formulae apply again.
- If $P_i = 0$, (1) gives $S_i = C_{err}$ instead of $\overline{C_{err}}$ for the expected sum bit, but the wrong carry is not propagated by (2), so the next stage addition is correct.

Assuming that the erroneous sum spreads from the mth to the pth stage, the error pattern appears as shown in Fig. 6. Just as in Fig. 5, where the error patterns are shown in red, the last faulty bit counterbalances the first ones, the absolute error has the significance of the first erroneous bit only:

$$2^p - 2^{p-1} - 2^{p-2} - \cdots - 2^m = 2^m. \tag{3}$$

This result is valid if the carry propagates normally. But some circuits can have more than one cut-back multiplexer. If the stages between two cut-backs are in propagate modes, the normal propagation driven by (2) could be disrupted. Thus, the previous result needs to be recomputed for this case.

Let assume the same carry error ($C_i = C_{err}$) in a propagating stage ($P_i = 1$, else there would be no carry-chain perturbation). If another cut-back happens to guess the same faulty carry C_{err}, it does not disrupt the normal propagation and the previous result holds (2). But if the carry cut happens in the opposite direction $\overline{C_{err}}$, it overrides (2) and reverses the carry error: the carry, that was false until now, comes back to the value of the expected addition. Thus, despite the cut, the current sum bit, determined by (1) with the exact carry, is correct, as well as the next stages.

The last erroneous sum bit being the preceding stage at value $\overline{C_{err}}$, the error pattern appears this time as in Fig. 7. All the erroneous bits are in the same direction and the absolute error is simply their sum:

$$2^p + 2^{p-1} + 2^{p-2} + \cdots + 2^m = 2^{p+1} - 2^m. \tag{4}$$

The magnitude of this error is much higher than in the first case, but it can only occur if several carry-cuts happen in opposite directions. Thus, to avoid such dramatic errors, the SPEC guess or the straight carry-cut must be chosen in the same direction for all the cut modules of the CCBA.

2) Worst-Case Relative Error: Having validated the fact that any error only has the magnitude of the bit position of the cut that caused it, the low impact of this error on the expected sum should be demonstrated.

The worst case happens when the error magnitude is the highest on the lowest expected calculated sum. Occurrence of an error implies that the three factors mentioned in section III-D are realized, thus the PROP and SPEC blocks intercept only propagate signals. All the non-zero operand bits producing those propagates add up to the expected sum:

- The PROP non-zero bits, which significantly contribute to maximizing the expected result and thus to minimizing the worst-case relative error.
- The SPEC non-zero bits, which contribute to a lower extent in increasing the sum by attenuating a portion of the magnitude of the error.
- If the SPEC guess or straight carry-cut is '0' (speculating a low carry), an error replaces a real carry necessarily at state '1' coming from a generate stage. Added to the SPEC non-zero bits, this stage further increases its sum to 2^m.

Thus, whenever an error occurs, while it keeps the magnitude of the cut bit significance, i.e. an arithmetic error of value 2^m, the sum is always expected to be greater than:

$$2^m + \sum_{k \in \text{PROP}} 2^k \quad \text{and} \quad \sum_{k \in \text{SPEC}} 2^k + \sum_{k \in \text{PROP}} 2^k, \tag{5}$$

leading to a relative error lower than:

$$\frac{2^m}{2^m + \sum_{k \in \text{PROP}} 2^k} \quad \text{and} \quad \frac{2^m}{\sum_{k \in \text{SPEC}} 2^k + \sum_{k \in \text{PROP}} 2^k}, \tag{6}$$

in the cases where the carry guess is at '0' and '1', respectively. This result holds also if multiple errors occur in different carry-cut modules, since the ratio of error over sum is preserved.

As a result, a floating-point precision can be achieved at design time through a proper sizing and positioning of PROP and SPEC blocks and selection of the right carry guess. It is easy to verify that the worst-case relative error is 7.7 % for the example in Fig. 5a and 12.5 % in Fig. 5b, corresponding to minimum precisions between 4 and 5 bits.

F. Design and Implementation Strategy

The CCBA allows considerable improvements concurrently in circuit performance and error control. This section describes how to exploit its architectural advantages.

1) Circuit Implementation: As the carry cut-back technique introduces hardware overhead to the adder, mainly for implementing PROP and SPEC blocks, their circuitry must be minimized. The easiest way is to limit their bit-widths to a few stages in order to reduce their area and delay, as those need to be computed at first to determine the activation and value of the cut. Executed in parallel, the delay overhead is limited by the slowest between PROP and SPEC.

Usually implemented with a fast and efficient carry-lookahead architecture, the area overhead of these functional blocks can fortunately be balanced. Indeed, PROP and SPEC

can be built with a similar architecture than the adder segments that they overlay. They could then share most of their circuitry, for instance with a Carry-Lookahead Adder (CLA), that uses small sum generators onto a carry-lookahead network.

It is worth mentioning that alike speculative adders, the CCBA can easily be made dynamically configurable to embed an exact computation mode or a variable-latency error correction mechanism. The cut-backs simply have to be disabled, the carry will propagate normally throughout the entire adder chain (with the original critical-path delay) and directly lead to the exact addition result.

2) Timing Constraints: To effectively benefit from the cut-back mechanism, it is necessary to manually exclude the generated false paths from STA. As mentioned in II-A, this ensures that the synthesis tool does not unnecessarily spend resources to meet delay constraints on them.

This can be achieved by providing the tool with a timing exception script based on the CCBA design information. An easy way to do so for a single arithmetic module is to set multiple delay constraints for each effective path instead of a single constraint for the entire design. They would correspond to the longest paths shown in red in Fig. 4 (not forgetting the SPEC if there is). For instance, on the CCBA implementation of Fig. 5a, delay constraints should be set on the three longest paths: from input to output bits [0..7], [2..15] and [10..17]. For the CCBA of Fig. 5b, there are this time four longest paths: [0..4], [1..9], [6..14] and [11..16].

3) Error Trade-Offs: The CCBA offers a large design space to co-design circuit timing together with functional errors. This allows maximizing circuit savings while minimizing the application quality loss. It also enables to dissociate arithmetic precision from dynamic range, usually fixed by the bit-width.

Firstly, the CCBA design parameters can independently set different error characteristics:

- The error rate depends on the number of cut-back modules and of the PROP and SPEC bit-widths.
- The maximum error can be adjusted mainly by sizing the cut length and the PROP bit-width, and to a lesser extent by modifying the SPEC bit-width and input guess.
- Adjusting other error metrics, such as Signal-to-Noise Ratio (SNR), Root Mean Square (RMS) error or any other statistical metric can be achieved using the similar models than those built for speculative adders [23], [24].

Secondly, the same design parameters affect circuit timing and costs in terms of area and power, intuitively:

- More cuts leads to a faster circuit, but at the cost of more and higher errors, as well as circuit overhead required for SPEC, PROP and multiplexing.
- Larger SPEC or PROP reduces the errors at the cost of delay, area and power overheads.
- Increasing the cut length lowers the error impact without hardware overhead. However, a longer cut induces a lower timing relaxation, leading to lower circuit savings.
- To optimize the timing budget, it can be interesting to equalize the delay of all the timing paths, by equally spacing cut-back modules and by adjusting the bit-widths of the first and last ADD blocks.

G. Design-Space Minimization

Due to its flexibility, the choice of the right set of parameters remains cumbersome. Fortunately, the dependency between errors and circuit characteristics can help limiting the design-space exploration to a low number of possible candidates.

In order to find the minimum design requirements to fit a given maximum relative error, a simple methodology can be derived by reversing equation (6). First, neglecting the terms associated to the SPEC gives the minimum *cut length*, composed of the bit-width of the PROP (l_{PROP}) and of the $\mathrm{ADD_1}$ ($l_{\mathrm{ADD_1}}$) blocks, as on Fig. 4:

$$l_{\mathrm{cut}} = l_{\mathrm{PROP}} + l_{\mathrm{ADD_1}} \geq \lfloor 1 - \log_2 (\mathrm{RE_{max}}) \rfloor \tag{7}$$

where $\mathrm{RE_{max}}$ is the maximum relative error requirement. For instance, $\mathrm{RE_{max}}$ constraints of 50 %, 25 % and 12.5 % would impose minimum overall cut lengths of 2, 3 and 4 bits. The rounding down is due to the fact that the PROP and SPEC can be used for error compensation as well. But the choice of a greater cut length would ensure the required accuracy without constraint on those blocks.

Fixing this integer term while reversing (6) allows expressing the minimum required PROP bit-width to further limit the error and fit the desired $\mathrm{RE_{max}}$:

$$l_{\mathrm{PROP}} \geq \left\lceil l_{\mathrm{cut}} - \log_2\left(2^{l_{\mathrm{cut}}} + 1 - \frac{1}{\mathrm{RE_{max}}}\right)\right\rceil \tag{8}$$

for a carry guess fixed at '0'. Note that the SPEC does not influence the maximum relative error in case of carry guessed at '0', as suggested by (6). Therefore, no other design parameter can help limiting the error, this is why the rounding up is necessary to ensure the required accuracy.

In the case where the carry guess is dynamic or fixed at level '1', the expression is changed to:

$$l_{\mathrm{PROP}} \geq \left\lfloor l_{\mathrm{cut}} - \log_2\left(2^{l_{\mathrm{cut}}} - \frac{1}{\mathrm{RE_{max}}}\right)\right\rfloor. \tag{9}$$

Choosing a higher integer would guarantee fitting the error constraint. However, for that case, the SPEC can finally complement the cut length and PROP for error compensation. The minimum required SPEC bit-width is expressed as:

$$l_{\mathrm{SPEC}} \geq \left\lceil -\log_2\left(1 + 2^{l_{\mathrm{cut}}} - 2^{l_{\mathrm{cut}} - l_{\mathrm{PROP}}} - \frac{1}{\mathrm{RE_{max}}}\right)\right\rceil. \tag{10}$$

For example, let assume a 7.7 % relative error constraint. Equation (7) directly implies a minimum cut length of $l_{\mathrm{cut}} = 4$. For the case of a fixed guess at '0', (8) simply gives a mini-mum PROP bit-width of $l_{\mathrm{PROP}} = 2$ (no requirement for SPEC). For the case of a dynamic or fixed guess at '1', (9) and (10) give requirements of $l_{\mathrm{PROP}} = 2$ and $l_{\mathrm{SPEC}} = 7$. A 7-bit SPEC seems unreasonable for circuit efficiency. Choosing a larger PROP (e.g. $l_{\mathrm{PROP}} = 3$) directly guarantees to fit the chosen $\mathrm{RE_{max}}$ without such unrealistic SPEC requirement. For both guess values, circuits with larger cut length (e.g. $l_{\mathrm{cut}} = 5$) fit the error straight away no matter PROP and SPEC bit-widths.

The number of cut-backs and their absolute positions do not influence the $\mathrm{RE_{max}}$. Thus, many implementations remain to be investigated to find the optimal circuit, but the design space would be reduced to a few dozen candidates.

IV. Results and Comparison

A. Methodology

1) Scope of the Exploration: In this work, 32-bit unsigned adders have been investigated in the frame of controlling and lowering the relative errors, particularly in its worst case, which delimits the floating-point error and thus the minimum precision of the operator.

To further narrow the design space, all the architectures of approximate adders have been kept uniform and regular for fair comparison, i.e. segmented and compensated adders using uniformly-sized blocks with identical properties and CCBA using uniformly-spaced and identical cut-back modules. They have all been considered with multiple types of carry generator guess: either fixed at '0' or '1', or dynamic using the preceding-stage input operand as introduced by [13].

More than 50,000 implementations of approximate adders with diverse error characteristics have been investigated to provide an overall picture of their performance.

2) Circuit Implementation and Characterization: All the approximate adders described in section III-A have been implemented in VHDL from the highest-level behavioral description. Therefore, the internal architecture of ADD, PROP and SPEC blocks is left to the compiler's optimization, in order to benefit from the most favorable architecture to fit the timing constraint. The exact adder used as reference has been instantiated from the Synopsys DesignWare IP library.

All designs have been synthesized using Synopsys Design Compiler in the UMC 65 nm process for two target frequencies: 800 MHz (low-power) and 3.3 GHz (high-performance). Post-synthesis delay, area, leakage and dynamic power have been extracted by Synopsys Design Compiler. Only implementations meeting the timing constraints have been considered.

Circuits have been assessed in terms of silicon area, Power-Delay-Area Product (PDAP), and energy per operation at the targeted frequency. This latter has been chosen not to disadvantage slower circuits yet fitting the timing constraint.

3) Error Characterization: The metrics used to characterize approximate adders in this work are based on the relative error (RE), which has the advantage of being independent of the size of the adder. It is defined as:

$$\mathrm{RE} = \left| \frac{S_{\mathrm{approx}} - S_{\mathrm{exact}}}{S_{\mathrm{exact}}} \right| \tag{11}$$

where S_{approx} and S_{exact} are the approximate and correct sums of an addition, respectively.

The main metric used to characterize the circuits for this study is the maximum of the relative error ($\mathrm{RE}_{\mathrm{max}}$), which states the worst case or minimum precision of the circuit. It is also essential for targeting commercial products. The mean relative error ($\mathrm{RE}_{\mathrm{mean}}$) is also taken into consideration as it is widely spread in the analysis of approximate circuits [25].

The approximate adders have been characterized through the simulation of two samples of five million unsigned random inputs. First, a log-uniform distribution, exhibiting a very large dynamic range, has been used to detect the worst-case error $\mathrm{RE}_{\mathrm{max}}$. Then, a uniform distribution has been utilized to estimate $\mathrm{RE}_{\mathrm{mean}}$.

However, five million test vectors is a very small subset of the exhaustive simulation, which contains 1.8×10^{19} combinations of two 32-bit operands. In order to assess the quality of each estimate, three additional samples of five million inputs have been used to measure the deviation of the statistical estimate. Other methods have recently been proposed to improve the estimation accuracy by dynamically adjusting the number of simulation vectors [26] or by using a formal approach to analyze errors [27].

As the error characteristics of adders spread over multiple orders of magnitude and are plotted on logarithmic scale, the Relative Standard Deviation (RSD) has been used (rather than the variance or the standard deviation) for measuring the dispersion of the results over the four random samples.

B. CCBA Results

Error characteristics and circuit costs are shown for a selection of CCBAs at 3.3 GHz in Fig. 8 and at 800 MHz in Fig. 9. Circuit costs are normalized to the exact adder represented on the left of the figures. Sorted by $\mathrm{RE}_{\mathrm{max}}$, those designs have been selected as they represent optimal and well-balanced circuits.

CCBAs are tagged by architectures, 'm' for multiplexed and 'ii' for input-induced, and denoted by quintuples with their most important design parameters: (*number of cuts*, *cut length*, PROP, SPEC, *guess*). Fixed guesses are indicated by '0' or '1', and dynamic ones by 'A' or 'B'. Input-induced cuts are expressed the same way although they override two input signals instead of the intermediate carry.

To first assess the quality of the error characterization, RE estimations are shown with Relative Standard Deviation (RSD) error bars in logarithmic scale, magnified by a factor 500. The largest variation is observed in the middle of Fig. 8, but only represents a RSD of 0.4 % (i.e. the standard deviation on the estimation accounts for 0.4 % of the estimated value). The typical variations observed for $\mathrm{RE}_{\mathrm{mean}}$ are in the order of 0.1 %, while around 10^{-3} % for $\mathrm{RE}_{\mathrm{max}}$, which confirms that five million vectors are sufficient for a correct estimation of $\mathrm{RE}_{\mathrm{mean}}$ and $\mathrm{RE}_{\mathrm{max}}$.

Figs. 8 and 9 highlight the large design flexibility and accuracy range enabled by the proposed architecture. The CCBA design parameters allow tuning the precision on almost seven orders of magnitude of $\mathrm{RE}_{\mathrm{max}}$ while exhibiting PDAP reductions up to 70 % . This precision tunability is three to four orders of magnitude higher than previously reported [4]. Those stunning results are enabled by the improvement of the HDL code and by the new input-induced cuts which mainly benefit to high-performance and high-accuracy adders (right of Fig. 8).

As expected from III-F, the *cut length* (addition of ADD_1 and PROP bit-widths, as on Fig. 4) is the main parameter regulating the relative error. It ranges from 3 to 17 bits in Fig. 8, corresponding to $\mathrm{RE}_{\mathrm{max}}$ from 25 % down to 2.0×10^{-3} %, and reaches 27 bits in Fig. 9 for a $\mathrm{RE}_{\mathrm{max}}$ of only 1.5×10^{-6} %.

$\mathrm{RE}_{\mathrm{mean}}$ follows the same trend as $\mathrm{RE}_{\mathrm{max}}$ for implementations with multiple cuts (left of Figs. 8 and 9), for which

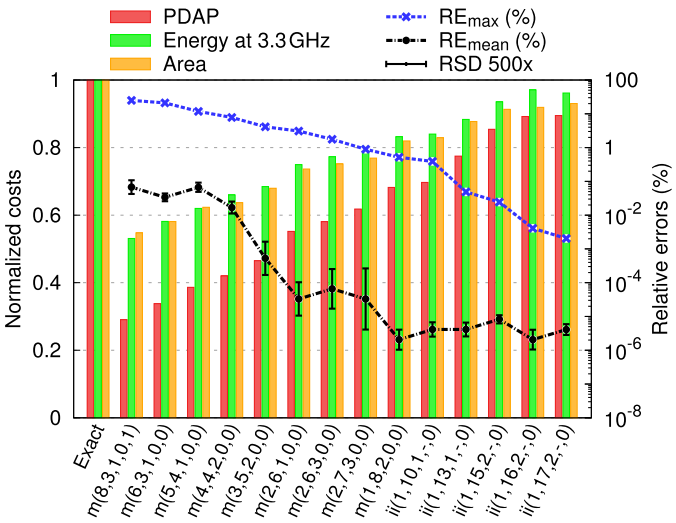

Fig. 8. Relative errors and normalized circuit costs of various 32-bit CCBAs (on the horizontal axis) synthesized at 3.3 GHz in a 65 nm technology.

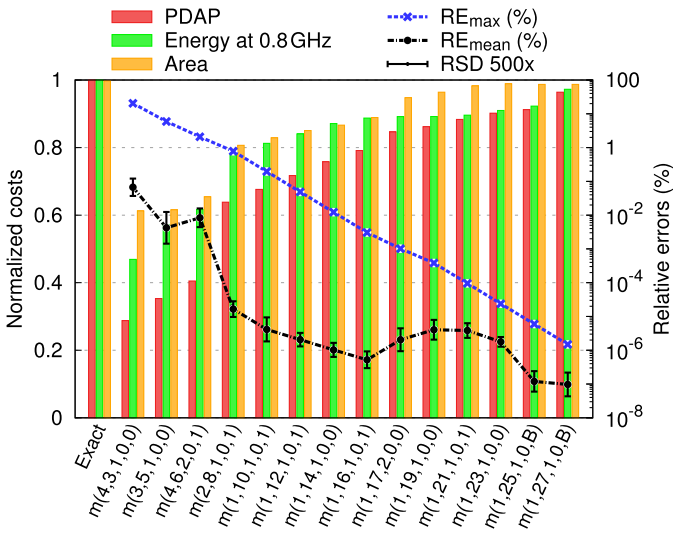

Fig. 9. Relative errors and normalized circuit costs of various 32-bit CCBAs (on the horizontal axis) synthesized at 800 MHz in a 65 nm technology.

a small change in their structure, repeated over many cuts, strongly impacts the overall error rate and mean. It scales in a lower degree among adders having a single cut and with higher accuracy (right of Figs. 8 and 9).

Target delay has a significant influence on the results. Low-speed implementations show better normalized savings than high-performance ones at equivalent precision. At 2 % RE_{max}, 3.3 GHz CCBAs achieve up to 25 % energy savings and 40 % PDAP reductions against around 40 % and 60 % for 800 MHz circuits.

Fig. 9 presents a sharp drop in circuit efficiency around 1 % RE_{max}. This corresponds to the precision from which the 800 MHz design becomes delay-constrained. Indeed, higher precision demands wider PROP, SPEC or greater cut length, which all lie in the effective critical path of the CCBA. This does not appear for 3.3 GHz adders, which are always tightly constrained.

Another irregularity occurs in the right-hand side of Fig. 9, where savings in high-accuracy adders appear to come exclusively from energy reduction. As for speculative adders, CCBA circuit savings usually come from the use of slower but more efficient topologies for the adder sub-blocks, enabled by the shortening of the critical path (e.g. Ripple-Carry instead of Han-Carlson, or Han-Carlson rather than Kogge-Stone). However, for very high accuracy, the large size of the sub-adders does not allow the compiler to select more area-efficient architectures, but it is still possible to relax timing and reduce dynamic power consumption compared to the exact adder.

Note that all the shown implementations have either a 0-bit SPEC or no SPEC at all, i.e. the penultimate design parameter being 0 or undefined, respectively. This is indeed appearing as the best circuit trade-off for lowering RE_{max}, sole criteria by which those designs have been selected. This is also true for the PROP bit-width, ranging from 1 to 3 bits only. Out of the scope of this paper but interesting for future studies, the use and focus on other metrics, such as error rate or arithmetic errors, would lead to optimization strategies employing bigger SPEC and PROP blocks.

C. Comparative Study

This section makes a comprehensive and exhaustive comparison of CCBAs and state-of-the-art approximate adders. All the adders described in III-A are represented: ESA [11], ETAII [12], ETBA [13], GCSA [14], ISA [15], ACA [16], ACAA [17], LOA [18] and pruned adders [19]. Exact truncated adders are also given for reference.

Figs. 10 and 11 show dot plots of the accuracy-efficiency Pareto frontiers achievable by approximate adders at 3.3 GHz and 800 MHz, respectively. Error characteristics are measured on horizontal axes, compared by RE_{mean} on left subfigures and by RE_{max} on right ones. Circuit costs are measured on vertical axes, regarding energy consumption for top subfigures and normalized PDAP for bottom ones (PDAP is normalized to the exact 32-bit adder). The best designs are towards the bottom-left corners of subfigures.

Before going into the detailed results, the quality of the statistical estimations has been assessed by computing their variation over four samples of input vectors. The median RSD observed is in the order of 0.1 % for RE_{mean} and 10^{-4} % for RE_{max}. Only a handful of designs exhibits a high variation, over 5 %, mainly among high-accuracy ACA and ACAA adders. This is due to their large overlapping sub-adders inducing extremely low error rates [26]. The most critical case shows 28 % RSD for RE_{mean} or 4 % RSD for RE_{max}. In regard to the wide distribution of adder errors, covering many decades of the logarithmic scale, such variability does not challenge the viability of the study.

1) Mean Relative Error: Figs. 10a-b and 11a-b compare approximate adders in regard to the RE_{mean}. Those circuits do *not* correspond to the ones in Figs. 8 and 9, they show a different Pareto-optimal set solely considering RE_{mean}.

At high speed (Figs. 10a-b), truncated adder and LOA show the best circuit efficiency at a given RE_{mean}. On the contrary, ACA, ETBA and GCSA display the highest energy and PDAP costs. These are also unable to achieve high accuracy due to their direct dependency between sub-adder bit-

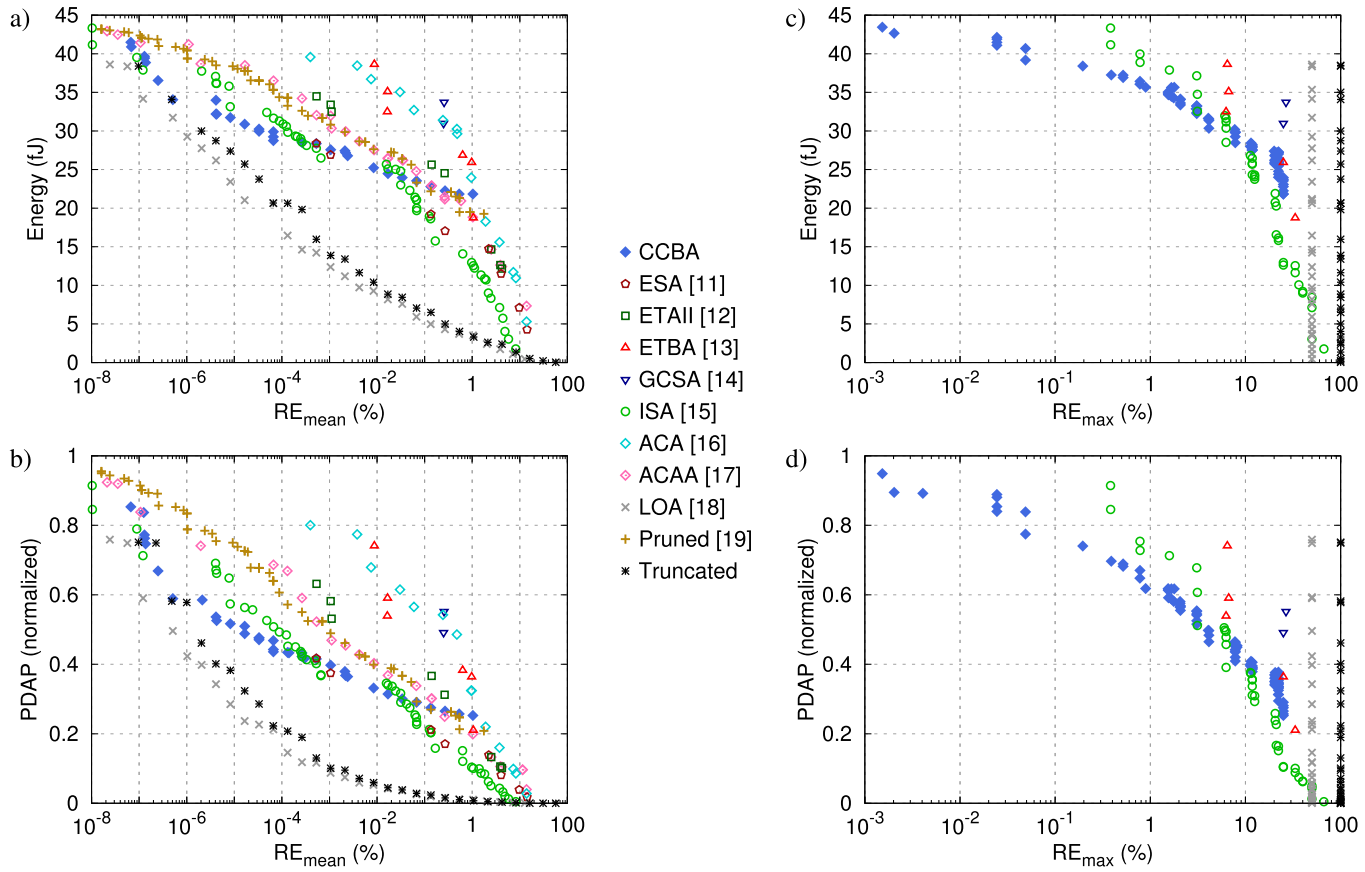

Fig. 10. Comparison of relative errors and circuit costs of approximate adders synthesized at 3.3 GHz in a 65 nm technology.

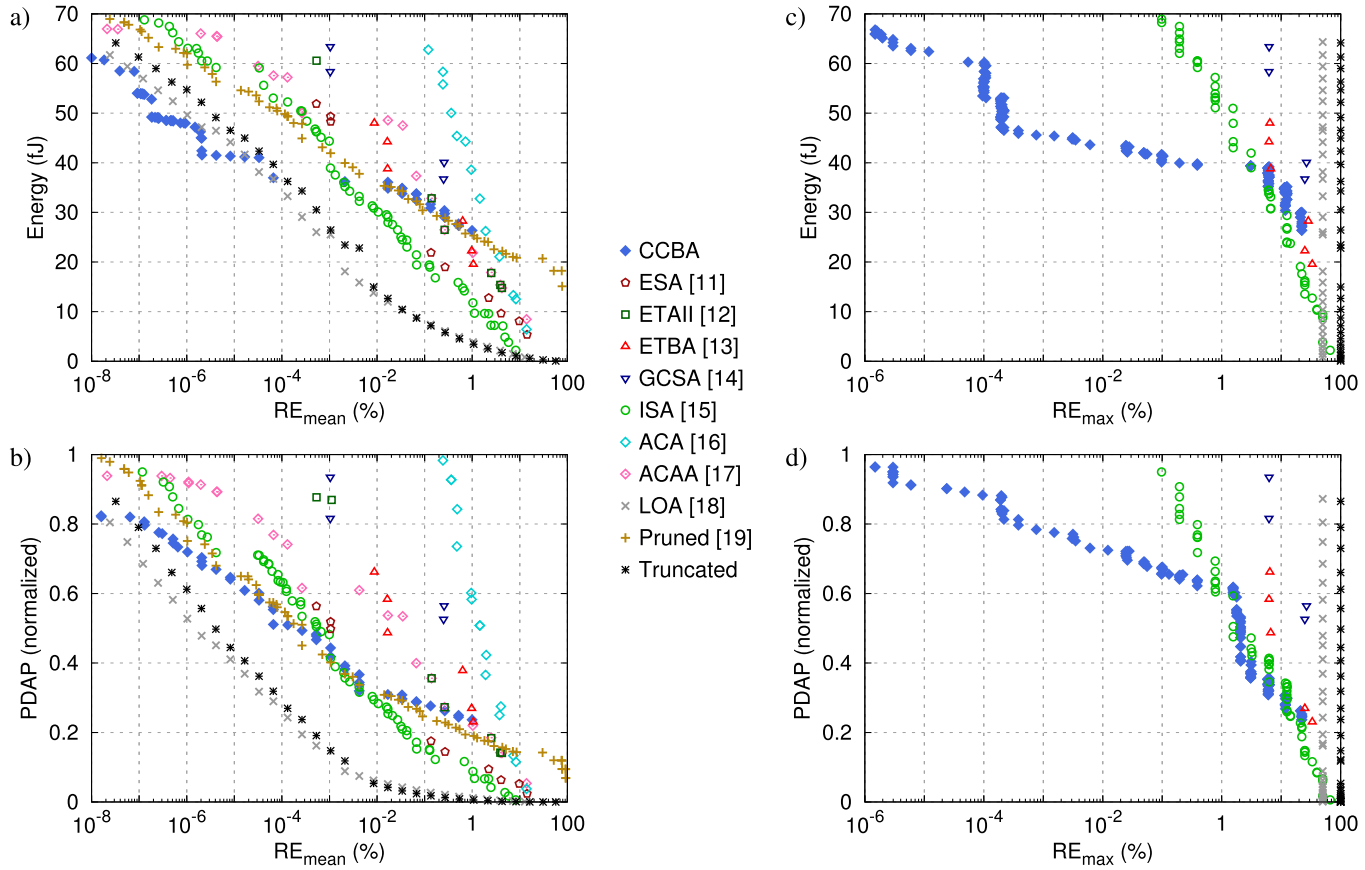

Fig. 11. Comparison of relative errors and circuit costs of approximate adders synthesized at 800 MHz in a 65 nm technology.

width and error compensation, which makes it difficult to find an implementation optimizing both circuit and functionality. Outperformed by truncated adder and LOA at low accuracies, ISA and CCBA tend to catch up for RE_{mean} below 10^{-6} %, in terms of energy consumption as well as in PDAP.

At lower speed (Figs. 11a-b), the pruned adder displays moderate savings, with a closer trend to speculative circuits, particularly with the ISA. The latter shows a slightly better efficiency for RE_{mean} above 10^{-6} %, while pruning takes over below 10^{-6} %, validating the results obtained in [28]. LOA and truncated adder still outperform state-of-the-art adders, particularly for low RE_{mean}. However, at high accuracy, the CCBA surpasses both of them in energy efficiency. This confirms the trend identified in IV-B that high-accuracy CCBA circuits exhibit better energy savings than area reductions.

2) Maximum Relative Error: Figs. 10c-d and 11c-d compare approximate adders regarding their ability to limit the worst-case precision, i.e. RE_{max}. As this study focuses on lowering the relative errors, only adders able to limit RE_{max} below 100 % have been represented: truncated adder, LOA, ETBA, GCSA, ISA and CCBA.

For both high-performance and low-power scenarios, the overall trend is similar. With its OR gates on the lower-part addition, the LOA cannot compensate errors, though it preserves the overall magnitude of the computed sum, leading to a 50 % worst-case error. Disregarding low-significance operand bits and underestimating the sum result, truncated adders have exactly a RE_{max} of 100 %.

ETBA and GCSA show moderate savings, but their error compensation schemes are limited by their dependency between block size and error compensation, as for mean errors. The ISA, which has an optimized architecture dissociating critical path and error control, offers a larger range of RE_{max} with more decent energy and PDAP reductions. For both 3.3 GHz and 0.8 GHz, the ISA offers the best trade-off for worst-case errors in the 7-50 % range.

The CCBA outperforms all other adders for RE_{max} below 7 % for energy and PDAP in both low-power and high-speed cases. It is also the only adder capable of high accuracy, with RE_{max} below 0.1 %, while still displaying considerable savings compared to exact adders. This is particularly emphasized for low-power circuits: at 10^{-3} % RE_{max}, the CCBA enables PDAP reductions up to 22 % or energy savings up to 36 %.

V. Conclusion

This paper has introduced a novel concept to optimize approximate circuits by fabricating false timing paths, i.e. critical paths that can never be logically activated. Co-designing circuit timing together with functionality, this method proposes to monitor and cut critical paths to transform them into false paths. This allows to relax timing constraints, resulting in lower circuit costs or higher performance. Implementing the cuts on low-significance nets of arithmetic circuits can guarantee minimal behavioral changes, such as reduced precision or controllable arithmetic errors.

This technique has been applied to an approximate adder circuit, called the Carry Cut-Back Adder (CCBA), in which high-significance stages can cut the carry propagation chain at lower-significance positions, guaranteeing a high accuracy. This lightweight approach prevents the carry-chain activation, therefore relaxing timing constraints and strongly improving circuit efficiency. A design methodology has been presented in order to tune the accuracy, to optimize and correctly implement the timing constraints, as well as to reduce the design-space exploration.

An industry-oriented comparison for a 65 nm commercial CMOS technology has been carried out against 10 state-of-the-art approximate adders, including truncated exact adders. The CCBA architecture allows tuning the accuracy over almost seven orders of magnitude while exhibiting power-delay-area reductions up to 70 % for low-speed implementations. It greatly outperforms all other approximate adders for worst-case errors, and most of them for mean errors. Finally, it even improves upon exact truncated adders in terms of energy in the case of high-accuracy low-power designs. For a worst-case accuracy of 99.999 %, energy savings up to 36 % or power-delay-area reductions up to 22 % are demonstrated compared to low-power conventional designs.

This work has proven the considerable advantage of exploiting false timing paths on adder circuits. This novel approach could benefit larger arithmetic circuits, such as multipliers [29], as well as bigger datapaths, like CORDIC [30] and FPU [31]. Its extremely lightweight circuit implementation could help building highly-efficient configurable or precision-scalable hardware accelerators, with a better predictable and controllable impact on their functionality.

References

[1] C. M. Kirsch and H. Payer, "Incorrect systems: It's not the problem, It's the solution," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2012, pp. 913–917.

[2] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "DVAFS: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *Proc. IEEE Conf. Design, Automat. Test Eur. (DATE)*, Mar. 2017, pp. 488–493.

[3] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. 14th ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2009, pp. 195–200.

[4] V. Camus, J. Schlachter, and C. Enz, "A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision," in *Proc. 53rd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2016, pp. 127:1–127:6.

[5] D. H. C. Du, S. H. C. Yen, and S. Ghanta, "On the general false path problem in timing analysis," in *Proc. 26th Conf. Design Autom.*, Jun. 1989, pp. 555–560.

[6] P. C. McGeer and R. K. Brayton, "Efficient algorithms for computing the longest viable path in a combinational network," in *Proc. DAC*, Jun. 1989, pp. 561–567.

[7] A. Magdy, Z. Jing, and B. Jayanta, "Design analysis tool for path extraction and false path identification and method thereof," U.S. Patent 8 627 249 (B1), Aug. 15, 2002.

[8] B. Siarkowski, "Method and system for false path analysis," U.S. Patent 7 958 470 (B1), May 2007.

[9] R. Solaiman, J. Mayank, R. Solaiman, and J. Mayank, "Method for modeling and verifying timing exceptions," U.S. Patent 7 650 581 (B2), Nov. 20, 2008.

[10] T. Liu and S.-L. Lu, "Performance improvement with circuit-level speculation," in *Proc. 33rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2000, pp. 348–355.

[11] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *Proc. IEEE Conf. Exhib. Design Automat. Test Eur. (DATE)*, Mar. 2011, pp. 1–6.

[12] N. Zhu, W.-L. Goh, and K.-S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. 12th IEEE Int. Symp. Integr. Circuits (ISIC)*, Dec. 2009, pp. 69–72.

[13] M. Weber, M. Putic, H. Zhang, J. Lach, and J. Huang, "Balancing adder for error tolerant applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 3038–3041.

[14] J. Hu and W. Qian, "A new approximate adder with low relative error and correct sign calculation," in *Proc. IEEE Conf. Exhib. Design, Automat. Test Eur. (DATE)*, Mar. 2015, pp. 1449–1454.

[15] V. Camus, J. Schlachter, and C. Enz, "Energy-efficient inexact speculative adder with high performance and accuracy control," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 45–48.

[16] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Autom. Test Eur.*, Mar. 2008, pp. 1250–1255.

[17] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2012, pp. 820–825.

[18] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[19] J. Schlachter, V. Camus, C. Enz, and K. Palem, "Automatic generation of inexact digital circuits by gate-level pruning," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 173–176.

[20] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, Jun. 2012, pp. 796–801.

[21] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApproxSb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. IEEE Conf. Design, Automat. Test Eur. (DATE)*, Mar. 2017, pp. 258–261.

[22] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-Bit multipliers accomplished," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 416–423.

[23] C. Liu, J. Han, and F. Lombardi, "An analytical framework for evaluating the error characteristics of approximate adders," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1268–1281, May 2015.

[24] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2012, pp. 728–735.

[25] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 60:1–60:34, Aug. 2017.

[26] J. Bonnot, V. Camus, K. Desnos, and D. Menard, "CASSIS: Characterization with adaptive sample-size inferential statistics applied to inexact circuits," in *Proc. 26th IEEE Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018.

[27] C. Yu and M. Ciesielski, "Analyzing imprecise adders using BDDs—A case study," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 152–157.

[28] V. Camus, J. Schlachter, and C. Enz, "Energy-efficient digital design through inexact and approximate arithmetic circuits," in *Proc. IEEE 13th Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2015, pp. 1–4.

[29] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[30] M. Franceschi, V. Camus, A. Ibrahim, C. Enz, and M. Valle, "Approximate FPGA implementation of CORDIC for tactile data processing using speculative adders," in *Proc. IEEE New Gener. CAS (NGCAS)*, Sep. 2017, pp. 41–44.

[31] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *Proc. 42nd IEEE Conf. Eur. Solid-State Circuits (ESSCIRC)*, Sep. 2016, pp. 465–468.

Vincent Camus (S'15) received the Engineering degree from the Grenoble Institute of Technology (Grenoble INP), France, and the joint M.Sc. degree in micro and nanotechnologies for integrated systems from Grenoble INP, France, the Polytechnic University of Turin (Polito), Italy, and the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2013.

He is currently pursuing the Ph.D. degree in approximate and error-resilient circuits for low-power or high-performance systems with the Integrated Circuits Laboratory, EPFL. He has co-authored a dozen of scientific articles or patents and has contributed to the organization of several conferences and public scientific events. His research interests include digital circuits, electronic design automation, embedded systems, and machine learning.

Mattia Cacciotti (S'17) received the B.Sc. degree in biomedical engineering from the Polytechnic University of Turin (Polito), Italy, in 2013, and the joint M.Sc. degree in micro and nanotechnologies for integrated systems from Polito, the Grenoble Institute of Technology, France, and the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2015.

He is currently pursuing the Ph.D. degree in electronic engineering with the Integrated Circuits Laboratory, EPFL, focusing on approximate computing for energy-efficient digital circuits and systems. His research interests include FPGA, hardware accelerators, embedded systems, and hardware-software co-design.

Jeremy Schlachter (S'15) received the B.Sc. degree in physics and electrical engineering in 2011, the M.Sc. degree in micro and nanoelectronics from the University of Strasbourg, France, in 2013, and the Ph.D. degree in electrical engineering from the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2017.

He joined the Integrated Circuits Laboratory, EPFL, in 2013. His research focuses on approximate digital circuit design for energy-efficient and error-tolerant computing systems. His research interests include low-power electronics, digital circuit design, and electronic design automation.

Christian Enz (S'83–M'84–SM'11) received the Ph.D. degree from the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 1989.

He is currently a Professor at EPFL, where he is also the Director of the Institute of Microengineering and the Head of the Integrated Circuits Laboratory. Until 2013, he was the Vice President of the Swiss Center for Electronics and Microtechnology (CSEM), Neuchâtel, Switzerland, where he was the Head of the Integrated and Wireless Systems Division. Prior to joining CSEM, he was a Principal Senior Engineer at Conexant (formerly Rockwell Semiconductor Systems), Newport Beach, CA, USA, where he was responsible for the modeling and characterization of MOS transistors for RF applications. His expertise is in low-power analog and RF integrated circuits, wireless sensor networks, and semiconductor device modeling. He is the developer of the EKV MOS transistor model. He has co-authored over 250 scientific papers and has contributed to numerous conference presentations. He is an individual member of the Swiss Academy of Engineering Sciences. He was an elected member of the IEEE Solid-State Circuits Society (SSCS) AdCom from 2012 to 2014, and he is the Chair of the IEEE SSCS Chapter of Switzerland.