Modeling and Synthesis of Quality-Energy Optimal Approximate Adders

Jin Miao, Ku He, Andreas Gerstlauer, and Michael Orshansky
Department of Electrical & Computer Engineering, The University of Texas at Austin
{jinmiao, kuhe, gerstl, orshansky}@utexas.edu

Abstract

Recent interest in approximate computation is driven by its potential to achieve large energy savings. This paper formally demonstrates an optimal way to reduce energy via voltage over-scaling at the cost of errors due to timing starvation in addition. We identify a fundamental trade-off between error frequency and error magnitude in a timing-starved adder. We introduce a formal model to prove that for signal processing applications using a quadratic signal-to-noise ratio error measure, reducing bit-wise error frequency is sub-optimal. Instead, energy-optimal approximate addition requires limiting maximum error magnitude. Intriguingly, due to possible error patterns, this is achieved by reducing carry chains significantly below what is allowed by the timing budget for a large fraction of sum bits, using an aligned, fixed internal-carry structure for higher significance bits.

We further demonstrate that remaining approximation error is reduced by realization of conditional bounding (CB) logic for lower significance bits. A key contribution is the formalization of an approximate CB logic synthesis problem that produces a rich space of Pareto-optimal adders with a range of quality-energy trade-offs. We show how CB logic can be customized to result in over- and under-estimating approximate adders, and how a dithering adder that mixes them produces zero-centered error distributions, and, in accumulation, a reduced-variance error. We demonstrate synthesized approximate adders with energy up to 60% smaller than that of a conventional timing-starved adder, where a 30% reduction is due to the superior synthesis of inexact CB logic. When used in a larger system implementing an image-processing algorithm, energy savings of 40% are possible.

1. Introduction

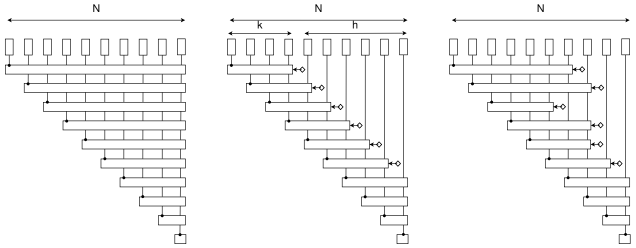
The possibility of trading computation accuracy for improved energy-efficiency has recently been investigated for use in digital systems that naturally tolerate errors, such as signal processing circuits. Error-permissive or approximate computations can be realized at several levels and via different mechanisms. Most of the existing research efforts have been aimed at solutions lying at system and algorithmic levels. In [9], energy is reduced by discarding algorithm steps or iterations that contribute less to the final quality. In [10], adaptive precision of the arithmetic unit output is used to save energy. Error-correction schemes have been proposed that use a main computing block running at a lower voltage and a simpler error-correcting block that runs at a higher voltage and is thus, error-free [2], [12]. Other techniques use the properties of specific algorithms to identify and skip unnecessary computations, e.g. [13], [5]. Transistor-level optimization of full adder cells is proposed in [1]. A general strategy for pruning out statistically-unimportant logic nodes is studied in [6], [3].

Elementary addition lies at the heart of many signal-processing applications, and attempts have been made to exploit error-energy trade-offs at the component level. At the circuit level, the primary mechanism for achieving energy savings is operating under a reduced timing budget via scaled V_{dd}, i.e., in the regime of timing starvation, and several efforts explored that possibility, including [11], [7], [15]. As of now, there is no formal answer to the question: Given a fixed amount of timing starvation, what is the optimal design strategy for approximate adders? In this paper, we develop a formal analysis for designing energy-optimal timing-starved approximate adders in the context of signal processing applications. Statistical properties of carry propagation are a key part of such an analysis. In addition, the carry path is a critical path and each bit is potentially impacted by the carry. Hence, under starvation, some output sum-bits become timing-inaccessible for a primary carry-in. The typical carry chain length is, however, much smaller than the maximum one. In fact, the likelihood of a long carry chain is quite small. For an N-bit adder, the expected worst-case carry length is close to $logN$ [4], and with extremely high probability is less than $logN + 12$ [11]. The lowest likelihood of errors is thus ensured even under large timing starvation if the sum-bits are each allowed to have their maximum possible carry chain [11], [7].

Maximally reducing error probability for each sum-bit minimizes overall error frequency. Yet, in many signal processing applications the relevant metric of approximation quality is a quadratic error measure, e.g. SNR/PSNR, that involves error magnitude as well as frequency. As we show in Section 2.2, due to possible error patterns, there is a fundamental trade-off between error frequency and error magnitude in a timing-starved adder. Points on a frequency-magnitude trade-off curve are generated by different arrangements of shortened carry segments, where the PSNR-optimal choice also depends on the statistics of operands. This drives the first key contribution of this paper: a formal proof that for signal processing applications assuming a uniform distribution of operands, reducing bit-wise error frequency is sub-optimal and that quality-optimal approximate addition is achieved by limiting maximum error magnitude while accepting a larger error frequency. This is realized by reducing carry chains significantly below what is allowed by the timing budget for a large fraction of sum-bits, using an aligned, fixed internal-carry structure for higher significance bits. Crucially, such a structure also allows for maximal sharing of logic across all aligned carry segments, thus resulting in an area- and energy-optimal design. To enable formal analysis, we introduce a model of timing-starved addition, which allows us to analyze error patterns and their frequency. The model is general and applies to ripple-carry as well as prefix/tree-type adders.

We further show that while maximum error is minimized by an aligned fixed internal carry adder for higher significance bits, it is crucial to further minimize average error. (In other words, just truncating the adder is a bad idea). This can be achieved by using LSB logic to produce an intentionally incorrect result that compensates for the error due to timing starvation. We introduce logic that generates LSB bits that cap their output when an error is generated

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012,
November 5-8, 2012, San Jose, California, USA



(a) Full-budgeted adder (b) Timing-starved RCA (c) Timing-starved CLA

Figure 1: Timing-starved adder model (TSAM)

in the MSB, i.e. conditionally. The key insight is that energy cost of such conditional bounding (CB) logic can be substantially reduced by realizing its *logically inexact* version. We formalize the inexact CB logic synthesis problem and demonstrate existence of a rich space of alternatives with different area/energy-error behavior. Note that while other instances of bounding approximate addition have been reported, e.g., [15], [14], they are introduced heuristically without the proof of optimality or formal synthesis methods. Finally, we demonstrate that both under- and overestimating approximate adders are possible. We introduce several implementations of *dithering approximate adders* which mix the under- and overestimating adders to produce a *zero-centered error distribution*. We demonstrate effectiveness of a dithering adder in reducing accumulation errors in consecutive additions by error averaging.

In summary, the paper makes the following contributions: (1) a timing-starvation model demonstrating that an optimal approximate adder reduces carry chains for a large fraction of sub-bits to a length significantly below what is allowed by the timing budget, using an aligned, fixed internal-carry (AFIC) structure for higher significance bits; (2) a theoretical analysis concluding that the CB logic is the optimal structure for realization of lower significance bits in conjunction with an AFIC adder for higher significance bits; (3) a set of models and algorithms to efficiently find Pareto-optimal realizations of inexact CB logic; and (4) a dithering approximate adder that mixes the under- and overestimating logic to produce a reduced-variance zero-centered error distribution.

2. Timing-Starved Addition: Properties and Optimality

In this section, we develop a timing-starved adder model as a tool for analyzing the key features of approximate addition. We use it to demonstrate a fundamental trade-off between error frequency and error magnitude in a timing-starved adder. We conclude that for signal processing applications in which a quadratic error measure is used, reducing bit-wise error frequency is sub-optimal and limiting maximum error magnitude is paramount.

2.1 Timing-Starved Adder Model

In order to formally study the error frequency and magnitude patterns in approximate addition, we introduce a timing-starved adder model (TSAM) as defined in Fig. 1. The model can represent a variety of actual adder implementations, including ripple carry and tree adders. For ease of presentation, we discuss the ripple carry (RCA) adder first. In TSAM model, the top-level blocks represent sum bits, the horizontal blocks represent logic to compute each sum bit S_i, and the rightmost point of each such segment defines the location of the farthest accessible internal carry under a given timing budget. Under a full timing budget, (Fig. 1(a)), all the sum-bits have access to the correct carry-in ($= 0$) at bit 0. Under a reduced timing budget (Fig. 1(b)) equivalent to $k < N$ bits, some sum-bits do not have enough time to be impacted by (do not "have access" to) the correct zero-bit carry-in. The actual accessible carry, given by the shifted rightmost point of each segment, in an operating adder depends on the value left on the carry node by the previous computation cycle and is treated as *unknown*. We represent this unknownness of the carry in a timing-starved adder by a diamond, see Fig. 1(b). Note that if Fig. 1 is used to model more complex adder structures, e.g. carry look ahead adders (CLAs) or prefix types, the segments will not be regular due to differences in paths for each bit. The carry will propagate to higher significance bits via carry-look-ahead bypass logic, whereas less significant bits may still need a regular propagation path. This shifts the adder critical path from the most significant bit (MSB) to the less significant bits (LSBs). Thus, when timing starvation occurs, the MSBs may, surprisingly, have an accessible internal carry that is further than even its right neighbor bits. This is illustrated in Fig. 1(c) for the example of a CLA.

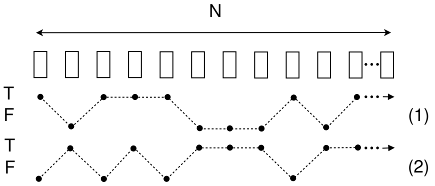


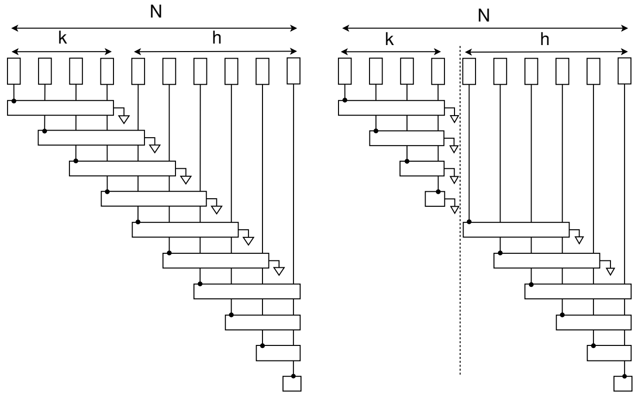
Figure 2: Error pattern waveforms.

The model allows studying the behavior of error frequency and magnitude with onset of timing starvation depending on the pattern of access of individual sum-bits to internal carries. Specifically, we show that depending on an arrangement of carry segments, a trade-off curve of maximum error magnitude and error frequency exists. The minimum error frequency solution is achieved by minimizing bit-wise error probabilities. Because of the low probabilistic likelihood of long carry chains, we conclude that to lower the bit-wise error occurrence frequency, we need to allocate the longest possible propagation chain for each bit position under the given timing budget. This is represented by an implementation that mimics the models in Fig. 1(b) and Fig. 1(c). While in a simple ripple-carry adder such behavior can be achieved by timing starvation directly, concerns about metastability or timing-closure may require breaking up the carry chain into over-lapping independent carry blocks. Furthermore, in tree adders with non-uniform default segment lengths, an independent implementation of identical blocks allows for capturing the maximum possible carry length in all sum bits. Several such implementations have been reported [11] and [7]. Intriguingly, we show below that this strategy is sub-optimal for many applications because of the nature of the trade-off between error frequency and maximum magnitude of error under a quadratic quality measure. Furthermore, implementations with independent carry blocks carry a larger area and hence energy overhead than what can be optimally achieved.

2.2 Error Frequency-Magnitude Trade-off

The trade-off between error frequency and maximum error magnitude is caused by patterns of possible errors. For a timing budget below k bits (see Fig. 1), any bit upto the MSB bit can be false. Thus, the largest possible error is 2^{N-1}. However, the maximum error is reduced if the false bit is followed by a *string* of false bits. Thus, surprisingly, forcing a set of bits to be false reduces the maximum error, and, if this is done for every pattern, the maximum *possible* error is reduced. Below we show how to achieve this effect and that its flip-side is the increased frequency of errors.

We represent error patterns by their F and T bit positions, which indicate whether a bit is incorrect (false) or correct (true), respectively. Bit sequences are given in the form of regular expressions, where '$*$' indicates consecutive repetitions. Graphically, er-



(a) Hardwired to zero (FIC-TS) (b) Aligned carries (AFIC-TS)

Figure 3: Timing-starved adder with fixed internal carries.

ror patterns can be represented as arbitrary waveforms of correct and incorrect bits (see Fig. 2). It can be shown that a timing-starved adder can produce an N-bit output in which any pattern of F^* and T^* is possible. Importantly, the maximum error magnitude of an adder is defined by the location of the first left-most possible occurrence of an F^* pattern; thus, the location of the first possible pair of F^*T^* bounds the maximum error magnitude of an adder. We call this the *FT transition*. Furthermore, an F^* pattern with a bitwidth of m, with a right-most bit in the pattern rooted at bit position r, can result in errors with only two magnitudes: $2^{m+r}-1$ or 2^r. In this case, whether the error pattern leads to a large or a small error depends both on the current adder inputs and the computational history for the internal carries. The key to our adder analysis is the realization that if we logically fix all the internal carries, conditions under which F^* would result in a large error (of magnitude $2^{m+r}-1$) cannot occur, i.e. F^* can only generate small errors with a magnitude of 2^r. Notice that internal carries can be fixed to either 0 or 1, leading to either lower- or upper-bounding of the result. (In what follows, we assume for the time being that the carries are fixed at 0). We term such an adder a fixed internal-carry timing-starved adder (FIC-TS), Fig. 3(a).

To reduce the maximum error magnitude, we are interested in shifting the FT transition to a lower bit position. In a FIC-TS adder, the FT transition can occur in the highest bit position and the maximum error is defined by the full length of the adder with a magnitude of 2^{N-1}. Since we cannot avoid errors in general, the only way to shift the FT transition within an F^*T^* pattern is to convert as many T bits as possible to F.

A bit j is F when the carry into its segment is incorrect, e.g., the correct carry is 1 while it's fixed to 0, and every downstream bit which is part of this segment has its propagate condition as true. In order to shift the FT transition by one bit, we need to ensure that if bit j is F, bit $j-1$ also becomes F, which can be made true if we ensure that the segment of bit $j-1$ also depends on the same incorrect carry-in. This can be achieved by aligning the right edges and hence inputs of the segments for bit j and bit $j-1$. Now the correctness/incorrectness of bits j and $j-1$ depends only on whether the accurate carry-in is zero (in which case both j and $j-1$ are T) or one (in which case both j and $j-1$ are F).

To shift the FT transition as far right as possible, we repeatedly apply the above conversion starting at bit $j=N$. This results in aligning a set of segments of downstream bits to that of bit N. Clearly, segments of lower bits are shorter than k (the length of the Nth segment). Hence, it is impossible to shift the FT transition beyond k. The segment length k is limited by the available timing budget, i.e. by the degree of timing starvation. However, the alignment of segments also means that the *effective carry chains* are reduced for the sum-bits below the MSB bit, and that increases the probability of individual and thus overall error. Fig. 4 shows the trade-off curve between maximum error and frequency that results from this exploration for an increasing number of aligned segments up to $k=7$ in a 16-bit RCA. The exact values of the Pareto-front depend on the statistics of adder operands, where results are shown both for an independent, uniform distribution as well as for input pairs that exhibit a small value distance across a uniformly distributed common magnitude range. Fig. 4 also shows the PSNR values that correspond to each configuration, which we discuss in the next section.

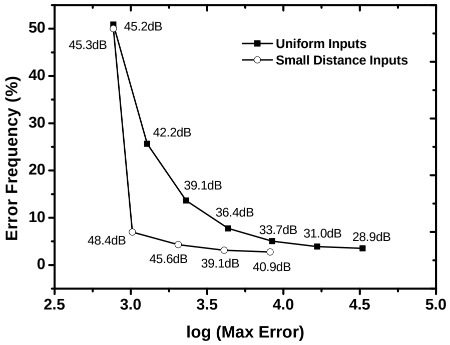


Figure 4: Error frequency vs. maximum error magnitude.

2.3 Optimal Approximate Addition under the PSNR Metric

Minimizing frequency of possible errors is justified in applications relying on error-correction. For other applications, such as in signal processing, it is the minimization of error magnitude that is more essential. In these applications, the quadratic error measure of adder error behavior, i.e. the quality of produced output, is most relevant. The specific metrics commonly used are the normalized mean squared error (MSE) and the related, peak signal-to-noise ratio (PSNR). PSNR depends on both magnitude and frequency of emerging errors. We now demonstrate that, for adder operands that are uniformly distributed, i.e. where all the input values are equally likely, PSNR is much more heavily influenced by the magnitude of the maximum error rather than error frequency. To do that, we consider a trade-off between error magnitude and frequency at a fixed PSNR value. We compare two adders A_1 and A_2, shown in Fig. 5, that produce errors of maximum magnitude $\delta_{1_{max}}$ and $\delta_{2_{max}}$ with a frequency of $f_{1_{max}}$ and $f_{2_{max}}$, respectively. We measure the quality loss in adder i as the sum of squared errors $SS_i = \sum_j^N \delta_{i_j}^2$ over N additions, which is proportional to the inverse of PSNR. Given maximum error magnitudes and their frequencies, we can bound quality losses from below and above as $Nf_{i_{max}}\delta_{i_{max}}^2 < SS_i < N\delta_{i_{max}}^2$ by assuming, in the best and worst case, that only maximum errors occur or that all additions lead to a maximum error, respectively. To understand when an adder A_1 has better quality than an adder A_2, we need to establish the conditions under which $SS_1 < SS_2$. Using the upper and lower bounds above for SS_1 and SS_2, respectively, this is the case if $SS_1 < N\delta_{1_{max}}^2 < Nf_{2_{max}}\delta_{2_{max}}^2 < SS_2$, i.e. $f_{2_{max}} > (\delta_{1_{max}}/\delta_{2_{max}})^2$.

In an adder, the maximum error magnitude is determined by the position m of the most significant bit in which an error can occur, and is equal to $\delta_{max} = 2^m$. If two adders differ by s bits in their maximum error location (see Fig. 5), we can conclude that an adder A_1 with smaller error magnitude will have better quality than an adder A_2 with larger error magnitude if the frequency of maximum errors in adder A_2 is at least $f_{2_{max}} > 1/4^s$. (Note that the inverse

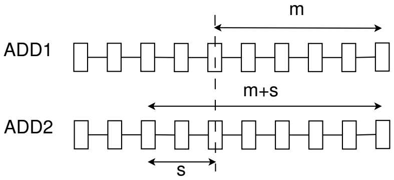


Figure 5: Adders A_1 and A_2 with locations m and $m+s$ and magnitudes 2^m and 2^{m+s} of maximum errors.

is not true, i.e. $f_{2_{max}}$ being below this bound does not necessarily imply that A_2 is better than A_1.) Thus, a larger error magnitude requires an exponential reduction in the frequency of such errors in order to remain below the quality budget set by an adder with lower error magnitude. This is confirmed in Fig. 4 that shows empirically collected PSNR values that correspond to each configuration. We see that for the uniform input distribution, the peak PSNR is indeed achieved for a solution with the smallest maximum error magnitude. This is not the case, however, for all distributions: if an adder processes input pairs that have similar values (small distance) then the peak PSNR is achieved at a different point.

We focus in the rest of the section on the analysis of the uniformly distributed (equally likely) adder inputs. Based on the analysis of the trade-off curve, the adder with the smallest maximum error is realized by aligning a set of segments of downstream bits to that of bit N. We call the resulting approximate adder structure an aligned fixed internal-carry timing-starved (AFIC-TS) adder. Fig. 3(b) shows the AFIC-TS adder, where the FT transition is shifted to the dotted boundary at bit position $N-k$. This reduces the maximum error magnitude by a factor of $\frac{1}{2^{k-1}}$ to make the maximum possible error 2^{N-k}. Note that the structure for the higher significant bits (left of the dotted boundary in Fig. 3(b)) is logically equivalent to, and can be implemented as, a regular adder, e.g. a RCA or CLA, that spans the MSB segment length k with a fixed carry in.

Using this analysis, we can determine the conditions under which an AFIC-TS adder has better error behavior than a FIC-TS one. The adders differ in their maximum error magnitudes by $k-1$ bit positions. Hence, an AFIC-TS adder will be better if the maximum error frequency of the FIC-TS adder is greater than $1/4^{k-1}$. The maximum error in an FIC-TS adder occurs if an incorrect carry propagates into its MSB while all other output bits are correct (T^*). This is the case if a carry is generated in the $k+1$st bit from the leftmost bit ($N-k-1$) and all higher significant bits propagate (but not generate) while all lower significant bits are correct. For uniform inputs, the probability of a bit to propagate or generate is $1/2$ and $1/4$, respectively. Furthermore, the probability for the lower significant bits to be correct is at least $1/2$ [7]. Thus, the maximum error frequency of the FIC-TS adder is at least $1/2^{k+2}$. In order to guarantee that the AFIC-TS adder is better, we need to ensure that $1/2^{k+2} \geq 1/4^{k-1}$, i.e. $k \geq 4$. Since k is a function of the available timing, this condition holds in almost all practical cases where we allow for budgets of at least 4 bit delays. Overall, the proof establishes that a AFIC-TS or equivalent adder (such as ETA [15]) is guaranteed to be better than a FIC-TS or equivalent adder (such as the approximation adder in [7]), regardless of the logic for lower significance bits (on the right side of the dotted boundary).

The discussion thus far has focused on proofs of quality optimality. Depending on the implementability of various adder structures, there may be differences in logic complexity, area and hence energy. As such, a subset of non-quality-optimal adders can have a better energy than the quality-optimal structure and, thus, also be Pareto-optimal in the quality-energy space. However, importantly, since aligning of segments allows for sharing of their logic, a maximally-aligned AFIC structure is not only optimal from a quality perspective, but also minimizes logic complexity.

When designing such optimal AFIC adders, we have remaining choices in regards to the logic of the lower significant bits and the value to which we fix a carry-in into the higher significance segment. Fixing the carry-in to zero or one will result in errors being always negative or positive, respectively. Depending on the desired behavior, we can therefore synthesize adders that over- or underestimate the result. This also opens the possibility of creating structures that dither to produce a zero-centered and reduced-variance errors. This choice also dictates the synthesis of the desired upper or lower bounding logic in the lower significance bits, as will be discussed further in the following sections.

3. Synthesis of Conditional Bounding Logic

After the maximum error is minimized with the aligned fixed internal carry adder for higher significance bits, it is crucial to reduce *average* error. We achieve this by using LSB logic to produce an intentionally incorrect result to compensate the error due to timing starvation. We introduce logic that generates LSB bits that bound its correct output when an error is generated in the MSB, i.e. conditionally. We further show that energy cost of conditional bounding (CB) logic can be substantially reduced by realizing its logically inexact version without substantial extra quality loss. In fact, there exists a range of Pareto-optimal adder implementations in the quality-energy design space. In the following, we formalize this design space and develop a heuristic to synthesize adder implementations for different application requirements and target technologies.

We focus our discussion initially on the case when the timing budget (set by the MSB segment length k) is sufficient for the correct timing evaluation of the LSB bits, i.e., $h = N-k \leq k$. When $h = N-k > k$, our synthesis approach will be able to trade off optimality for meeting a given timing budget. We support a hierarchical strategy that partitions the entire LSB logic into several smaller segments that each individually meet their timing constraints. This requires the segments, however, to be isolated from each other with no carry propagation between them. As a result, this solution may come at a cost of further degradation in achievable PSNR value.

3.1 Conditional Bounding Logic Formalization

As discussed, depending on the value of the fixed (controlled) carry into the MSBs, an AFIC adder will always over- or underestimate the true result. An important observation is that a quality-optimal adder implementation can be achieved by designing matching, conditionally bounding LSB logic that further minimizes remaining errors. Without loss of generality, we first assume the design of an underestimating adder with internal carries fixed to zero for the following discussion. Let C be the carry out of the LSB and carry into the MSB logic that is discarded. If $C = 0$, both MSB and LSB logic are correct. If $C = 1$, the MSB logic is incorrect, but an unmodified LSB logic still produces a correct result. This will always lead to the largest possible, negative error of -2^h. With these observations in mind, the optimal LSB logic should have the following properties: (a) produce a correct result when $C = 0$, and (b) produce the largest possible value (i.e. $11\ldots1$) when $C = 1$ to compensate for the large negative error in the MSBs as much as possible. This behavior is equivalent to the following Boolean equation for the desired LSB logic:

$$S_i^{'} = S_i \vee C, \qquad (1)$$

where S_i is the true sum value for output i, C is the carry-out of the entire LSB block, and $S_i^{'}$ is the desired sum value for bit i.

As previously discussed, an alternative overall adder design possibility is to fix the carry into the MSB logic to one. In this case, the LSB logic should be reversed. It should produce a correct output when $C = 1$ and the smallest possible value (i.e. $00 \dots 0$) when $C = 0$, which is logically described as:

$$S_i^{'} = S_i \wedge C \tag{2}$$

This allows us to design adders that are either over- or underestimating while minimizing the overall quality loss.

A general concern is that in either of these cases, consistent over- or under-estimation can result in errors that accumulate and grow when chaining several successive additions, as is the case, for example, in many applications that use accumulations. For applications that are sensitive to error accumulation, we introduce a structure that *alternates* between both types of logic in a dithering-style scheme in which statistical averaging reduces error variance in accumulation. This solution may come at an increased area cost, but due to our ability to synthesize reduced-area approximate bounding logic with the opportunity to share logic between both types, the area penalty is typically small. Furthermore, since at any given time only one block will be actively switching, there is very little energy overhead. A dithering adder is realized by a logic expression:

$$S_i^{'} = (D \wedge S_i \wedge C) \vee (\overline{D} \wedge (S_i \vee C)), \tag{3}$$

where D is an external control signal. This external signal allows dithering to be controlled by the application, e.g. to exploit knowledge about input data statistics or required error behavior. Furthermore, we can design simple, general control schemes that achieve averaging by driving the signal from a regularly alternating clock or through a history register that records whether a mismatch between hardwired and actual carry occurred and, if so, triggers the opposite bounding logic in the next addition in order to compensate.

As an alternative to external dither control, we can consider implementations in which the choice between over- and under-estimating, and hence between upper or lower bounding LSB logic, is generated internally based on other, regular adder inputs. Crucially, we can observe that an approximate AFIC adder will always produce a correct result iff the hardwired carry into the MSB matches the carry-out that would be produced by a regular LSB logic. Hence, if the LSB carry can be easily predicted from other inputs, and if the choice between different MSB carries and corresponding LSB bounding logic can be adapted accordingly, error frequency can be further minimized.

A low-overhead carry prediction can be performed based on adder inputs A_{h-1} or B_{h-1} at the partition boundary bit position $h - 1$. If both of these inputs are zero or one, the carry-out of the LSBs will also be zero or one, respectively, independent of any LSB-internal carry propagation. Hence, dithering can be controlled via the exclusive-or of those two inputs. In all unpredictable cases, we aim to randomly alternate for statistical averaging. For that, we can combine both cases and simply control the choice of MSB carry and LSB logic based on the value of one of the two inputs. Thus, the LSB logic expression for a $h-1$-dithering adder can be written as:

$$S_i^{'} = (A_{h-1} \wedge S_i \wedge C) \vee (\overline{A}_{h-1} \wedge (S_i \vee C)), \tag{4}$$

where A_{h-1} is the $h - 1$ bit of input A.

We refer to the logic defined in Eq. (1), Eq. (2) and Eqs. (3)/(4), as Conditional Upper Bounding (CUB), Conditional Lower Bounding (CLB) and Conditional Dithered Bounding (CDB) logic, respectively. In general, Conditional Bounding (CB) logic, where every sum output depends on the carry out of the complete LSB block, is more complex than that of a correct adder. An important part of our synthesis strategy is the idea that we can implement a logical approximation of $S_i^{'}$, given that ultimately the entire adder will still produce errors even if the CB logic implements $S_i^{'}$ exactly. By implementing a logical (Boolean) approximation to $S_i^{'}$, we can achieve significant area and energy reduction with only slightly worse error behavior. We expect a wide range of possible approximations of $S_i^{'}$ with different energy and quality values, from which a Pareto-optimal set can be found.

Table 1: Row-based function changes and distances for 2-bit CUB logic.

Input				CUB	D_1	D_2
A_1	B_1	A_0	B_0	$\langle S_1^{'} S_0^{'} \rangle$	$\langle S_1^{''} S_0^{''} \rangle$	$\langle S_1^{''} S_0^{''} \rangle$
0	0	1	0	01	00,10	11
0	0	1	1	10	01,11	00
0	1	1	0	11	10	01
0	1	1	1	11	10	01
1	0	0	0	10	01,11	00
1	0	0	1	11	10	01
1	1	0	0	11	10	01
0	0	0	0	00	01	10

3.2 Bounding Logic Synthesis

We are ultimately interested in an Pareto-optimal set of solutions in terms of MSE/PSNR and energy. However, a direct optimization seeking optimal points in this space appears intractable at the moment. Instead, we propose a heuristic approach that adopts a principle fundamental to logic synthesis: the number of literals in the logical expression is a proxy for the complexity, and thus area and energy (ignoring differences in switching activity), of the realization of a logic function. Formally, the Pareto-optimal set is generated by the solution to the following approximate logic synthesis problem:

$$min\ L(f^{'}) \quad s.t. \bigtriangleup (f^{'}, f) \leq \bigtriangleup_{target} \tag{5}$$

where $L(f^{'})$ is the number of literals in function $f^{'}$ and $\bigtriangleup(f^{'}, f)$ is the distance between the two functions $f^{'}$ and f. Notice that setting the distance to zero, $\bigtriangleup(f^{'}, f) = 0$, would make the problem equivalent to the traditional (exact) logic minimization problem.

The problem above introduces a proxy distance metric in lieu of PSNR allowing a more efficient implementation of the optimization problem. The distance definition we choose is closely coupled with a heuristic optimization we implement. The algorithm acts directly on a specification of the Boolean function in terms of the list of its ON-set/OFF-set minterms, i.e. on its truth table.

Without loss of generality, we continue the discussion based on CUB logic synthesis using Eq. (1) as function f to approximate. Synthesis of CLB and CDB logic is analogous and an identical algorithm can be applied. Consider the truth table for the CUB logic $S_i^{'}$ of a 2-bit adder in terms of input operands A_i and B_i (Table 1). The distance measure needs to capture the difference between the desired exact function and its approximation in a way that captures the characteristics of the PSNR error metric. Assuming uniform input distributions, each row in the truth table is equally likely. Therefore, the number of rows in which a change in function output is considered captures the frequency of errors. For a given row, we measure its *decimal distance (D)* as the decimal difference in the binary output values between the desired and inexact CUB logic. By flipping output bits within rows, we may produce inexact outputs with different decimal distances. Due to the quadratic nature of PSNR, the total distance (*TD*) is the sum of squared decimal distances over all rows in which bits have been

Algorithm 1 Inexact CUB synthesis

```
for #row_flips $r = 1 \ldots r_{max}$ do
    for each row in each subset of rows of size $r$ do
        for $D_j = D_{min} \ldots D_{max}$ do
            for each output $\langle S_i^{'} \rangle$ and all $\langle S_i^{''} \rangle = \langle S_i^{'} \rangle \pm D_j$ do
                Replace $\langle S_i^{'} \rangle$ by $\langle S_i^{''} \rangle$;
                Run 2-level Boolean minimization;
                Record $min\ (literal, TD)$ pairs;
            end for
        end for
    end for
end for
```

changed (r):

$$TD = \sum_{j}^{r} D_j^2 \tag{6}$$

where D_j is the decimal distance due to a change in row j. This procedure is illustrated in Table 1, which shows a partial truth table with decimal distances $D_1 = 1$ and $D_2 = 2$ for each row.

Using the TD metric defined above, we implemented the optimization heuristic shown in Algorithm 1 to find Pareto-optimal solutions in the literal-TD space. We empirically verified, as will be shown in the next section, that the proposed proxy metrics provide good fidelity while allowing tractable optimization. We synthesized solutions using Design Compiler to find true area and used behavioral simulation to extract PSNR values, creating a mapping between the literal-TD domain and the area-PSNR domain for a range of functions.

It is possible to make the above algorithm more efficient by avoiding considering all possible D_j in each row. The improvement is based on the conjecture that the Pareto set of total distance (TD) vs. number of literals (L) solutions of the inexact functions is to be found among solutions produced by only considering D_{min} combinations for a given number of row changes (flips) r. We can justify the conjecture by the following argument: (1) For any given TD value at a fixed r, the minimum achievable L is a function of the number of possible solutions to explore; (2) Due to the smaller number of possible flips of a row for larger D_j, the number of solutions $Num(D_j)$ decreases with an increase in individual distances being considered, i.e. $Num(D_i) > Num(D_j)$ for any $i < j$; (3) It follows that TD vs. L for a fixed r is monotonic and rising with increasing distance D_j. Hence, the Pareto set is among modifications formed by $D_{min} \times D_{min} \ldots D_{min}$ combinations and only these solutions need to be explored. This conjecture is verified empirically, where experiments for LSB adder logic of size 2 and 3 both confirmed the trend in (3).

Even with the described simplification, it is not feasible to use the algorithm for more than about $r_{max} = 8$ row flips. Whether this is sufficient depends on the number of rows in the truth table, which is exponential in the width h of the LSB block. With $r_{max} = 8$, we can explore a sufficient range of Pareto points for $h \leq 3$. To enable synthesis of larger adders, we adopt a hierarchical optimization strategy that partitions the LSB block into smaller segments that are synthesized and optimized independently and separately (Fig. 6). Segments are isolated from each other and there is no carry propagation between them, leading to sub-optimal approximations of the desired logic. However, by recursively applying the same CB synthesis approach to each segment with discarded output carry, the accrued errors are kept bounded, and we found results to be acceptable. Also, as alluded to earlier, the described hierarchical partitioning of the LSB logic can be used to meet reduced LSB timing budgets. The hierarchical approach reduces the runtime from $O(N \cdot 4^{2N})$ to $O(N)$ at the cost of reduced solution density. The actual accuracy loss is limited: we found that the gap is no larger than 3dB in the worst case.

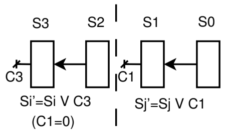
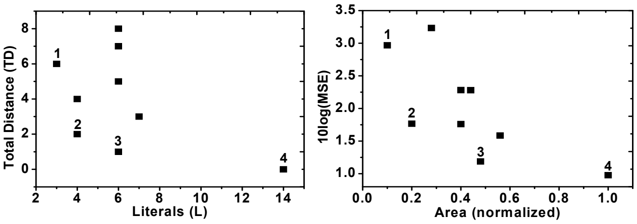


Figure 6: Hierarchical approach for partitioning of LSB logic and recursively applying CB synthesis to each segment.



(a) TD vs. L

(b) Area vs. quality

Figure 7: Synthesized inexact solutions for $h = 2$ CUB block.

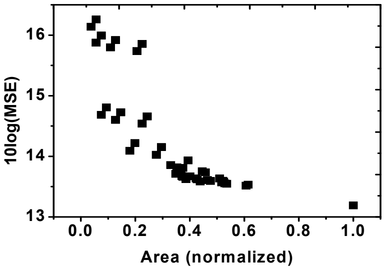


Figure 8: Inexact CUB synthesized hierarchically for 5-bit CUB block ($h = 5$).

Hierarchical exploration proceeds by first constructing the L vs. TD Pareto fronts for LSB logic of bitwidth $h = 1 \ldots 3$. To construct Pareto solutions for larger bitwidths, we explore all possible concatenations of smaller adders and their different design points to find the best overall L vs. TD solutions.

4. Experimental Results

We first demonstrate the results of approximate LSBs block synthesis using our algorithm with Espresso [8] as the internal 2-level Boolean minimization engine. Fig. 7(a) shows both the Pareto-optimal solutions and selected other design points explored by our algorithm for a 2-bit LSBs block in the TD vs. L space. The Boolean expressions for each of those solutions were synthesized with Synopsys DesignCompiler using the 45nm OSU PDK. Quality was estimated via simulation of the LSB block for 10,000 random, uniformly distributed input samples. The final area and quality values are shown in Fig. 7(b). Overall, we observe good fidelity: the points that are on the Pareto front in the TD vs. L space are also Pareto-optimal in the quality-area space. Points at the extreme high and low ends of the L/area range thereby correspond to exact and minimum-area realizations of the desired CUB logic, respectively. Furthermore, Fig. 8 shows the set of solutions for the approximate realization of a 5-bit LSB block produced by the hierarchical synthesis approach. We can observe a wide range of trade-offs, with some solutions having 1/5th of the area of the exact CUB logic at a moderate quality loss.

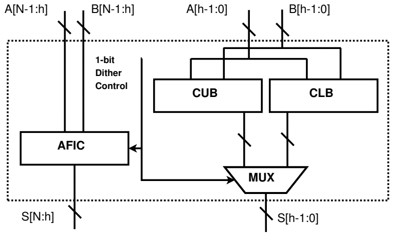


Figure 9: Dithering approximate adder.

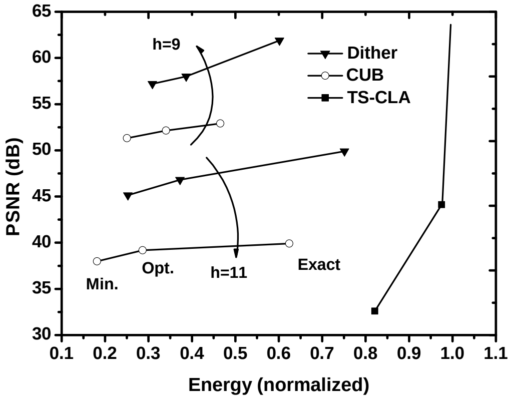


Figure 10: Quality-energy tradeoffs for different 16-bit CLAs.

As discussed previously, we can also realize adders that combine over- and underestimating behavior. Fig. 9 shows the conceptual design of the proposed dithering approximate adder. Its bounding behavior is controlled by an additional input signal, which determines both the carry into the MSB as well as the matching choice between CLB and CUB logic for the LSBs. In reality, we can synthesize the dithering LSBs as a combined CUB/CLB block with logic sharing. Overall, the overhead for a dithering-capable structure is low and its complexity remains well below that of a conventional adder. Note again that the dithering selection can be externally or internally controlled, either using more complex, adaptive schemes driven by the application or, simply, by a purely random signal, an alternating clock, based on carry-history or as a function of other inputs. After logic synthesis, a 24-bit RCA-based clock dithering adder with $h = 10$ has a 34% area overhead compared to a standalone, minimum-area AFIC-CUB design. For an internally controlled $h - 1$-dithering adder, the area overhead compared to a plain CUB RCA is around 30%. With increasing base complexity, this relative overhead reduces to 11% and 7.8% for CLA and Kogge-Stone based designs, respectively.

Fig. 10 shows the achievable quality-energy tradeoffs of various 16-bit approximate CLAs using an AFIC structure with LSB lengths $h = 9$ & $h = 11$ under varying minimum-area, optimal-tradeoff, exact CUB and $h - 1$-dithering realizations of the LSB block. We compare them against a conventional timing-starved CLA design. We assume energy reductions through V_{DD} scaling to be proportional to CV^2. For delay scaling, we utilize a curve-fitted model of HSPICE-simulated gate delays at different V_{DD} values. Energy of AFIC adders was estimated assuming a timing budget and V_{DD} value set by each nominal adder delay. Energy results are normalized to the base energy of the unscaled, original full-width CLA. Quality was measured by simulating adder results under scaled V_{DD} for 10,000 random inputs.

Results show that the conventional timing-starved adder experiences a sharp drop in quality once their timing budget is exceeded. For AFIC adders, the base quality level as well as the timing budget is set by the LSB and MSB widths h and $N - h$. Due to its ability for preemptively predicting the correct error compensation behavior purely from current adder inputs, a $h-1$-dithering scheme can in all cases significantly improve quality compared to its non-dithered counterparts. However, the added complexity comes at the cost of increased area and hence energy.

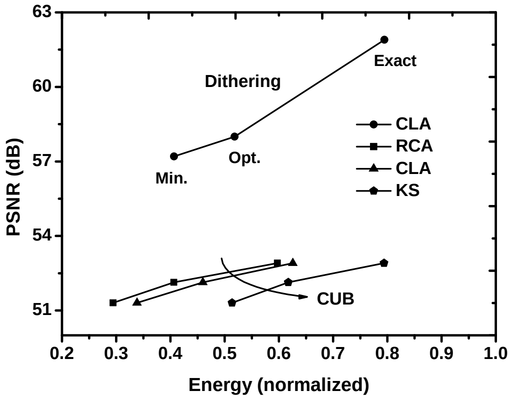
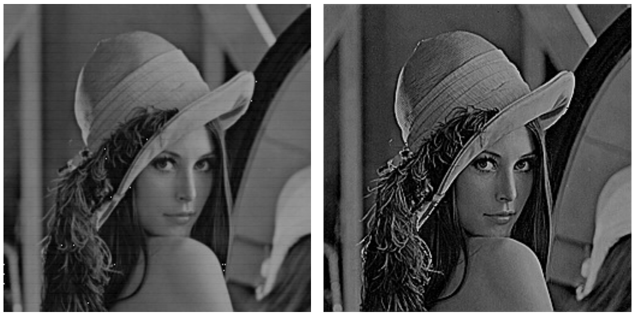


Figure 11: Quality-energy of 16-bit AFIC adders with $h = 9$.

Table 2: 16-bit CLA with $h = 9$ and varying LSB logic.

LSB Type	PSNR (dB)	Rel. Err. Full	Rel. Err. Small	Area (μm^2)	$\mathrm{V_{DD}}$ (V)	Energy
Original	∞	0%	0%	413.9	1.00	100%
Truncated	38.9	1.1%	100%	128.1	0.83	21.3%
$\mathrm{CUB_{min}}$	49.6	0.4%	23.5%	150.6	0.83	25.1%
$\mathrm{CUB_{opt}}$	51.2	0.4%	18.2%	204.6	0.83	34.1%
$\mathrm{CUB_{exact}}$	52.9	0.3%	0%	227.6	0.92	46.5%
$h - 1_{\mathrm{min}}$	57.2	0.2%	23.4%	185.8	0.83	30.8%
$h - 1_{\mathrm{opt}}$	58.6	0.2%	17.8%	232.4	0.83	38.7%
$h - 1_{\mathrm{exact}}$	61.9	0.2%	0%	264.7	0.97	60.2%



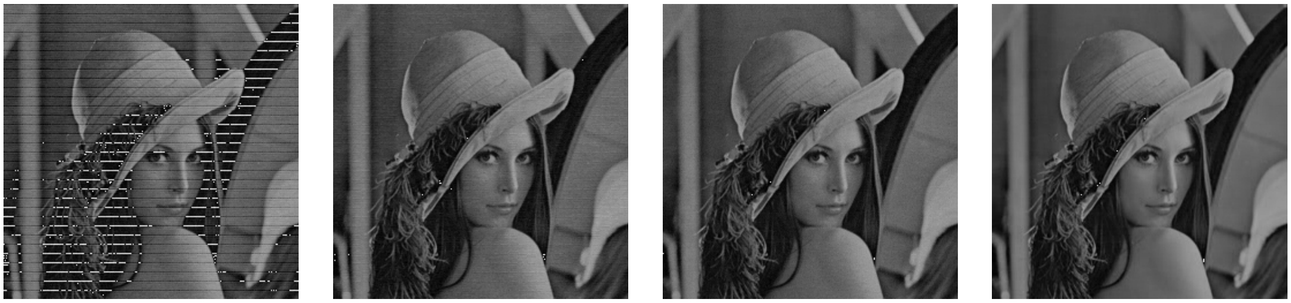
(a) IDCT w/ AFIC-CUB ($h = 8$)
PSNR=34.57dB, Energy=0.67

(b) Filter w/ AFIC-CUB ($h = 12$)
PSNR=23.7dB, Energy=0.60

Figure 12: Approximate adders in image processing applications.

Fig. 11 compares quality and energy of RCA, CLA and Kogge-Stone (KS) based designs of AFIC structures with $h = 9$, where energy is normalized against the base energy of an unscaled, regular RCA. Overall, even more so than the base adder structure, the partition boundary h or the timing budget, the choice of logic in the LSBs has a large effect on the area of the design and hence on the maximal achievable energy savings. Savings vary by 30% to 40% depending on the LSB logic style. This confirms the significance of exploring the CUB/CLB design space when designing families of approximate adders.

Table 2 summarizes results for a 16-bit AFIC CLA with $h = 9$ and different non-dithering and dithering LSB realizations. For comparison, we include a truncating adder with an empty LSB



(a) IDCT w/ Truncated Adder ($h = 10$)
PSNR=16.9dB, Energy=0.61

(b) Clock-dithering adder ($h = 10$)
PSNR=33.15dB, Energy=0.62

(c) History-dithering adder ($h = 10$)
PSNR=35.52dB, Energy=0.63

(d) $h - 1$-dithering adder ($h = 10$)
PSNR=36.92dB, Energy=0.62

Figure 13: IDCT quality and energy of a truncated adder (a), and different dithering schemes (b-d).

block. Results show that a significant difference in achievable quality between different synthesized CUB designs. Specifically, dithering adders improve PSNR considerably. Some CUB designs show very poor relative error metric which can be an important metric for realistic DSP systems. We show relative errors for two different uniform distributions of input with a full and a reduced range of magnitudes. For inputs that are smaller than the partition boundary, the design of the LSB logic has a large influence. In contrast to other realizations, an exact CUB realization will be error-free for such inputs. Overall, depending on application requirements, there exists a non-trivial tradeoff in finding a good compromise between quality and energy, as realized by the optimal $\mathrm{CUB}_{\mathrm{opt}}$ instance for the uniform input case.

To demonstrate feasibility for practical scenarios, we applied adder concepts to an IDCT image decompression and an image sharpening design, where the latter realizes a high-pass filter as a 2D convolution operation in the pixel domain. Fig. 12 shows the images and quality-energy tradeoffs under scaled V_{DD} when replacing a conventional 24-bit RCA in both designs with our minimal-area AFIC-CUB structure. Results are compared to the original IDCT and sharpening designs with a normalized energy of 1.0 and a PSNR of 44.6dB and 23.9dB, respectively.

While significant energy reductions can be achieved for a commonly accepted image quality above 30dB in the IDCT, error accumulations in the AFIC-CUB design lead to visual artifacts in the form of horizontal stripe patterns. By contrast, application of various dithering schemes provides both a much better PSNR as well as perceived quality. As shown in Fig. 13, by increasing the partition boundary h and hence decreasing the timing budget, this quality gain can be traded off for further energy savings. We compare our designs against a traditional approach that works with reduced precision (i.e. truncation) to achieve similar energy savings. Both from a PSNR and subjective image quality standpoint, the $h - 1$-dithering scheme is superior to truncation and any randomized external control. Dithering also leads to a reduction in the variance of observed errors. For the IDCT, we measured the error distributions. For an AFIC-CUB adder it has a mean of -0.95 and a variance of 5.16. By contrast, the clock-dithering adder produces errors with a mean of -0.1 and a variance of 0.94. Distribution of errors is not a concern in the sharpening filter. Here, a simpler AFIC-CUB adder with $h = 12$ already achieves similar results (Fig. 12(b)). In both cases, around 40% energy savings can be achieved while maintaining good image quality.

5. Conclusions

In this paper, we presented a theoretical approach for analysis and synthesis of approximate adders. Our approach is general and we formally prove the existence of optimal AFIC adder structures in which higher significance bits are implemented using regular, aligned carry additions. Within the space of AFIC adders, we further demonstrated that a rich set of design alternatives at varying quality-energy tradeoffs can be synthesized. This includes variants with overestimating, underestimating or dithering approximation behavior for use within different classes of application requirements. Our results show that energy savings of up to 60% are possible at the individual adder level. Integrating the developed approximate adders into realistic image processing designs allows more than 40% total energy savings while maintaining excellent image quality.

References

[1] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy. IMPACT: imprecise adders for low-power approximate computing. In *ISLPED*, 2011.

[2] R. Hedge and N. R. Shanbhag. Soft digital signal processing. *TVLSI*, 9(6), 2000.

[3] Z. Kedem, V. Mooney, K. Muntimadugu, and K. Palem. An approach to energy-error tradeoffs in approximate ripple carry adders. In *ISLPED*, 2011.

[4] D. R. Kelly and B. J. Phillips. Arithmetic data value speculation. In *ACSAC*, 2005.

[5] F. Kurdahi, A. Eltawil, K. Yi, S. Cheng, and A. Khajeh. Low-power multimedia system design by aggressive voltage scaling. *TVLSI*, 18(5), 2010.

[6] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *DATE*, 2011.

[7] S.-L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3), 2004.

[8] P. McGeer, J. Sanghavi, R. Brayton, and A. Vincentelli. Espresso-signature: A new exact minimizer for logic functions. In *DAC*, 1993.

[9] S. H. Nawab, A. V. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig. Approximate signal processing. *VLSI Signal Processing*, 15, 1997.

[10] A. Sinha and A. P. Chandrakasan. Energy efficient filtering using adaptive precision and variable voltage. *ASIC SOC Conference*, 1999.

[11] A. K. Verma, P. Brisk, and P. Ienne. Variable latency speculative addition: a new paradigm for arithmetic circuit design. In *DATE*, 2008.

[12] L. Wang and N. R. Shanbhag. Low-power filtering via adaptive error-cancellation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 51(2), 2003.

[13] T. Xanthopoulos and A. Chandrakasan. A low-power dct core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization. *J. Solid-State Circuits*, 35(5), 2000.

[14] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo. Enhanced low-power high-speed adder for error-tolerant application. In *ISOCC*, 2010.

[15] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *TVLSI*, 18(8), 2010.