A Simple Yet Efficient Accuracy-Configurable Adder Design

Wenbin Xu, *Student Member, IEEE*, Sachin S. Sapatnekar, *Fellow, IEEE*, and Jiang Hu, *Fellow, IEEE*

***Abstract*— Approximate computing is a promising approach for low-power IC design and has recently received considerable research attention. To accommodate dynamic levels of approximation, a few accuracy-configurable adder (ACA) designs have been developed in the past. However, these designs tend to incur large area overheads as they rely on either redundant computing or complicated carry prediction. Some of these designs include error detection and correction circuitry, which further increase the area. In this paper, we investigate a simple ACA design that contains no redundancy or error detection/correction circuitry and uses very simple carry prediction. The simulation results show that our design dominates the latest previous work on accuracy-delay-power tradeoff while using 39% lower area. In the best case, the iso-delay power of our design is only 16% of accurate adder regardless of degradation in accuracy. One variant of this design provides finer-grained and larger tunability than that of the previous works. Moreover, we propose a delay-adaptive self-configuration technique to further improve the accuracy-delay-power tradeoff. The advantages of our method are confirmed by the applications in multiplication and discrete cosine transform computing.**

***Index Terms*— Accuracy-configurable adder (ACA), approximate computing, delay-adaptive reconfiguration (DAR), low-power design.**

I. Introduction

POWER constraints are a well-known challenge in advanced VLSI technologies. Low-power techniques for the conventional exact computing paradigm have been already extensively studied. A comparatively new direction is approximate computing, where errors are intentionally allowed in exchange for power reduction. In many applications, such as audio, video, haptic processing, and machine learning, occasional small errors are indeed acceptable. Such error-tolerant applications are found in abundance in emerging applications and technologies.

A great deal of approximate computing research has been concentrated on arithmetic circuits, which are essential building blocks for most of computing hardware. In particular, several approximate adder designs have been developed [1]–[14]. One such design [2] achieves 60% power reduction for discrete cosine transform (DCT) computation without making any discernible difference to the images being processed. In realistic practice, accuracy requirements may vary for different applications. In mobile computing devices, different power modes may entail different accuracy constraints even for the same application. Specifically, arithmetic accuracy can be adjusted at runtime using methods, such as dynamic voltage and frequency scaling, to obtain the best accuracy-power tradeoff. The benefit of runtime accuracy adjustment is demonstrated in [3], but their approximation is realized by voltage overscaling, where errors mostly occur at the timing-critical path associated with the most significant bits (MSBs), i.e., errors are often large.

To reduce the overall error, a few approximate designs have been developed by intentionally allowing errors in lower bits with shorter carry chain in addition operation. In [4], a design that considers only the previous k inputs instead of all input bits can approximate the result with the benefit in half of the logarithmic delay. Reliable variable latency carry select adder shows a speculation technique that introduces carry chain truncation and carry select addition as a basis [7]. A series of error tolerant adders (ETAI, ETAII, and ETAIIM), which truncate the carry propagation chain by dividing the adder into several segments, have been proposed [8]–[10]. Correlation-aware speculative adder in [11] relies on the correlation between MSBs of input data and carry-in values. Another approximate adder that exploits the generate signals for carry speculation is presented [12]. These designs focus on static approximation, which pursue almost correct results at the required accuracy. However, in some applications such as image processing or audio/video compression, the required accuracy might vary during run time. To meet the need for runtime accuracy adjustment, a series of designs are developed to implement accuracy-configurable approximation, which could be reconfigured online to save more power.

A few accuracy-configurable adder (ACA) designs that use approximation schemes other than voltage overscaling have been proposed. An early work [15], called ACA, starts with an approximate adder and augments it with an error detection and correction circuit, which can be configured to deliver varying approximation levels or accurate computing. Its baseline approximate adder contains significant redundancy, and the error detection/correction circuit further increases area overhead. The ACA design [15] is generalized to a flexible

Manuscript received June 2, 2017; revised October 20, 2017 and December 25, 2017; accepted January 16, 2018. Date of publication February 28, 2018; date of current version May 22, 2018. This work was supported by NSF under Grant CCF-1255193, Grant CCF-1525749, and Grant CCF-1525925. *(Corresponding author: Wenbin Xu.)*

W. Xu and J. Hu are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: wbxu@tamu.edu; jianghu@tamu.edu).

S. S. Sapatnekar is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: sachin@umn.edu).

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/TVLSI.2018.2803081

framework GeAr in [16]. In both ACA and GeAr, the error correction must start from the least significant bits (LSBs), and hence, the accuracy improves slowly in the progression of configurations. The work of Accurus [17] modifies ACA/GeAr to overcome this drawback and achieves graceful degradation. However, in ACA, GeAr, as well as Accurus, the error correction circuit is pipelined, implying that the computation in accurate mode takes multiple clock cycles and causes data stalls.

An alternative direction of ACA design is represented by accuracy gracefully-degrading adder (GDA) [18] and RAP carry look-ahead adder (CLA) [19]. These methods start with an accurate adder and use carry prediction for optional approximation. As such, they no longer need error detection/correction and do not incur any data stall. In addition, they intrinsically support graceful degradation. The GDA design [18] is composed of accurate carry ripple adder (CRA) and extra configurable carry prediction circuitry, similar as the carry look-ahead part of CLA. Thus, its area is generally quite large. reconfigurable approximate carry look-ahead adder (RAP-CLA) [19] is based on accurate CLA design and reuses a portion of the carry look-ahead circuit as carry prediction. This leads to an overall area that is less than GDA but greater than CLA. In [19], the carry-prediction-based approach is shown to be superior to error-correction-based design [16].

In this paper, we propose a new carry-prediction-based accuracy configurable adder design: simple accuracy-reconfigurable adder (SARA). It is a simple design with significantly less area than CLA, which, to the best of our knowledge, has not been achieved in the past in ACAs. SARA inherits the advantages of all previous carry-prediction-based approaches: no error correction overhead, no data stall, and allowing graceful degradation. Compared to GDA [18], SARA incurs 50% less power-delay product (PDP) and can reach the same peak signal-to-noise ratio (PSNR). Moreover, SARA demonstrates remarkably better accuracy-power-delay tradeoff than the latest, and arguably the best, previous work RAP-CLA [19]. A delay-adaptive reconfiguration (DAR) technique is developed to further improve the accuracy-power-delay tradeoff. The proposed designs are also validated by multiplication and DCT computation in image processing.

II. Prior Works and Rationale of Our Design

We review a few representative works on ACA design and show the relation with our method. These designs can be generally categorized into two groups: error-correction-based configurations [15]–[17] and carry-prediction-based configurations [18], [19].

The main idea of an error-correction-based approach [15]–[17] is shown in Fig. 1. The scheme starts with an approximate adder (the dashed box), where the carry chain is shortened using separated subadders with truncated carry-in. In order to reduce the truncation error, the bit-width in some subadders contains redundancy. For example, *subadder2* calculates the sum for only bits 8 and 9, but it is an 8-bit adder using bit [9 : 2] of the addends, 6 bits of which are redundant. Even with the redundancy,

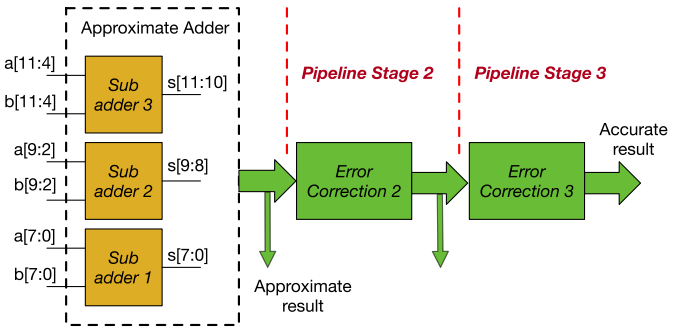


Fig. 1. Error-correction-based configurable adder.

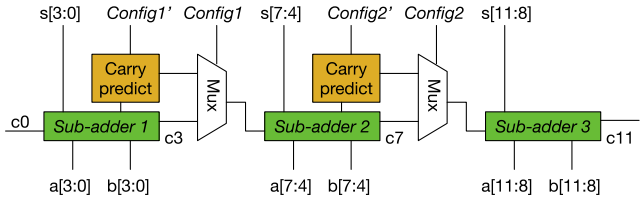


Fig. 2. Carry-prediction-based configurable adder.

there is still residual error which is detected and corrected by additional circuits. In Fig. 1, the errors of *subadder2* must be corrected by *error-correction2* before the errors of *subadder3* are rectified by *error-correction3*. As such, the configuration progression always starts with small accuracy improvements. The redundancy and error detection/correction incur large area overhead. Since the error correction circuits are usually pipelined, an accurate computation may take multiple clock cycles and could stall the entire datapath, depending on the addend values.

The framework of carry-prediction-based methods [18], [19] is shown in Fig. 2. These schemes start with an accurate adder design, which is formed by chaining a set of subadders. Each subadder comes with a fast but approximated carry prediction circuit. By selecting between the carry-out from subadder or carry prediction, the overall accuracy can be configured to different levels. Such an approach does not need error detection/correction circuitry. Moreover, the configuration of higher bits is independent of lower bits. This leads to fast convergence or graceful degradation in the progression of configurations. In GDA [18], the subadders are CRA designs, while the carry-prediction circuit is similar to the carry look-ahead part of CLA. Furthermore, its carry prediction can be configured to different accuracy levels. However, the complicated carry prediction induces large area overhead. The RAP-CLA scheme [19] uses CLA for its baseline, where the carry-ahead of each bit is computed directly from the addends of all of its lower bits. Its carry prediction reuses a part of the look-ahead circuit rather than building extra dedicated prediction circuitry, and hence is more area-efficient than GDA. But its baseline is much more expensive than GDA.

Our design is a carry-prediction-based approach. Its subadders are CRA instead of expensive CLA as in RAP-CLA. Its carry prediction also reuses part of the subadders rather than having dedicated prediction circuitry. As such, it avoids the disadvantages of both GDA and RAP-CLA. A comparison

TABLE I
COMPARISON OF CHARACTERISTICS FOR DIFFERENT TECHNIQUES

Method	Baseline sub-adder	Error correction	Graceful degradation	Carry prediction
ACA [15]	Redundant CRA	Yes	No	No
GeAr [16]	Redundant CRA	Yes	No	No
Accurus [17]	Redundant CRA	Yes	Yes	No
GDA [18]	CRA	No	Yes	Stand-alone
RAP-CLA [19]	CLA	No	Yes	Reuse
SARA (ours)	CRA	No	Yes	Reuse

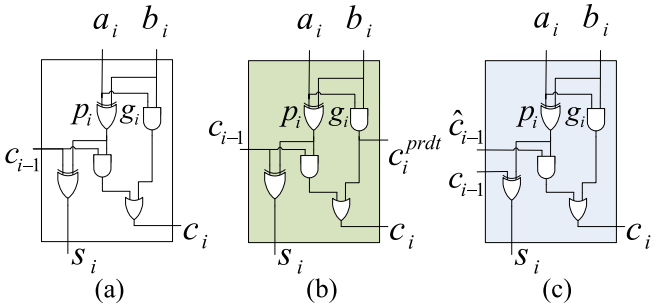


Fig. 3. (a) Conventional full adder. (b) Carry-out selectable full adder. (c) Carry-in configurable full adder.

among the characteristics of these different techniques is provided in Table I.

III. SIMPLE ACCURACY-RECONFIGURABLE ADDER

A. Preliminaries

An N-bit adder operates on two addends $A = (a_N, a_{N-1}, \ldots, a_i, \ldots, a_1)$ and $B = (b_N, b_{N-1}, \ldots, b_i, \ldots, b_1)$. For bit i, its carry-in is c_{i-1} and its carry-out is c_i. Defining the carry generate bit $g_i = a_i \cdot b_i$, propagate bit $p_i = a_i \oplus b_i$, and kill bit $k_i = \bar{a}_i \cdot \bar{b}_i$, the conventional full adder computes the sum s_i and carry c_i according to

$$s_i = p_i \oplus c_{i-1} \tag{1}$$

$$c_i = g_i + p_i \cdot c_{i-1}. \tag{2}$$

A gate level schematic of a conventional full adder is provided in Fig. 3(a). A CRA is used to chain N bits of conventional full adders together.

By applying (2) recursively, one can get

$$c_i = g_i + p_i g_{i-1} + \cdots + g_1 \prod_{k=2}^{i} p_k + c_0 \prod_{k=1}^{i} p_k. \tag{3}$$

This equation implies that c_i can be computed directly from g and p of all bits, without waiting for the c of its lower bits to be computed. This observation is the basis for CLA adder.

B. Simple Accuracy-Reconfigurable Adder Design

In SARA, an N-bit adder is composed of K segments of L-bit subadders, where $K = \lceil N/L \rceil$ (see Fig. 2). Each subadder is almost the same as CRA except that the MSB of a subadder, which is bit i, provides a carry prediction as

$$c_i^{\text{prdt}} = g_i. \tag{4}$$

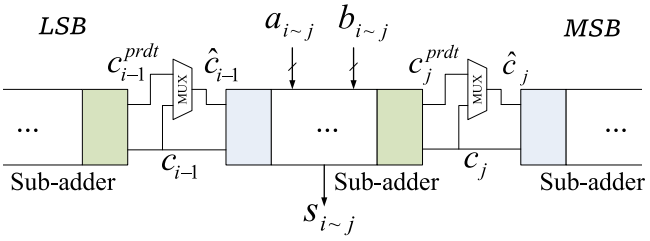


Fig. 4. Design of SARA.

For the LSB of the higher bit subadder, which is bit $i+1$, its carry-out c_{i+1} can be computed using one of two options: either by the conventional $c_{i+1} = g_{i+1} + p_{i+1} \cdot c_i$, or by using the carry prediction as

$$c_{i+1} = g_{i+1} + p_{i+1} \cdot c_i^{\text{prdt}} = g_{i+1} + p_{i+1} \cdot g_i. \tag{5}$$

The selection between the two options is realized using MUXes as in Fig. 4 and the multiplexer (MUX) selection result is denoted as $\hat{c}_i$. Comparing (5) with (3), we can see that the carry prediction is a truncation-based approximation to carry computation.[1] Therefore, $\hat{c}_i$ can be configured to either accurate mode or approximation mode, that is

$$\hat{c}_i \leftarrow \begin{cases} c_i^{\text{prdt}}, & \text{if approximation mode} \\ c_i, & \text{if accurate mode.} \end{cases} \tag{6}$$

It should be noted that the carry prediction c_i^{prdt} reuses g_i in an existing full adder instead of introducing an additional dedicated circuit as in [18] or Fig. 2. This prediction scheme makes a very simple modification to the conventional full adder, as shown in Fig. 3(b).

One can connect $\hat{c}_i$ to its higher bit $i+1$ to compute both carry c_{i+1} and sum s_{i+1}, as in GDA [18] and RAP-CLA [19]. We suggest an improvement over this approach by another simple change as in Fig. 3(c), where s_{i+1} is based on c_i instead of $\hat{c}_i$. Such approach can help reduce the error rate in outputs when an incorrect carry is propagated. Because the sum keeps accurate and the carry will not be propagated when addends are exactly the same. Moreover, out of all four configurations of sum/carry calculation by approximate/accurate carry-in, the most meaningful way is to have sum bit calculated by accurate carry and make carry bit configurable. Therefore, sum s_{i+1} is calculated directly by accurate carry c_i without the option of c_i^{prdt}. Applying this in SARA as in Fig. 4, in the approximation mode, computing s_{j+1} from c_j can still limit the critical path to be between c_{i-1}^{prdt} and s_{j+1}, but has higher accuracy than computing s_{j+1} from $\hat{c}_j$. Compared to sum computation in GDA and RAP-CLA, this technique improves accuracy with almost no additional overhead. Compared to CRA, the overhead of SARA is merely the MUXes, which is almost the minimum possible for configurable adders.

Although s_{j+1} is calculated by accurate carry c_j, its delay can still be reduced by approximate carry in lower subadder. In a multibit adder, the delay of sum bit depends on the carry

[1] A similar approximation is used in static approximate adder design [12].

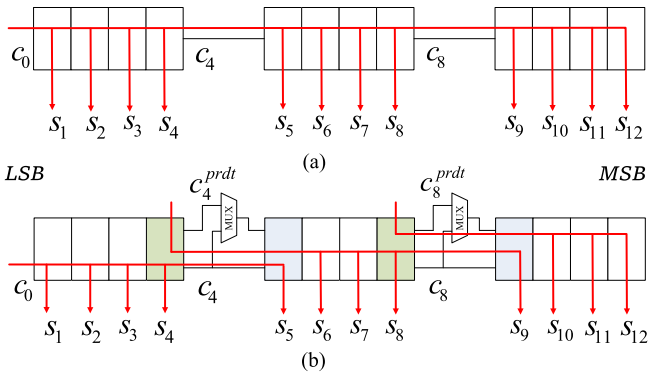


Fig. 5. Implementation of 12-bit adder in (a) CRA and (b) SARA.

chain propagated from its lower bits. In our SARA structure, even when accurate carry c_j is propagated at bit j, the carry chain might be truncated by approximate carry in other lower bits. In Fig. 4, when c_{i-1}^{prdt} is propagated, the delay of s_{j+1} is reduced as its path is shorten to be between bit $i-1$ and $j+1$. We can take the 12-bit adder in Fig. 5 as an example. For 12-bit SARA working in approximate mode, the sum s_9 uses the accurate carry c_8 from a lower subadder (bits 5 to 8). But c_8 is propagated from approximate carry c_4^{prdt} of another subadder (bits 1 to 4). As shown in the figure, the delay of s_9 in SARA is about six stages. Compared with the same bit in CRA, the delay of sum bit s_9 in SARA is reduced by three stages. Similar delay reduction can be observed in other sum bits (bits 6 to 12). For sums at bits 1 to 5, their delay is the same as CRA, because they are using an accurate carry c_0 from LSB. As a result, the maximum delay in 12-bit SARA is reduced, since for a multibit adder its maximum delay depends on the longest critical path.

C. Usage of SARA

When $\hat{c}_i$ is configured to be c_i for all K subadders, SARA operates very much like the CRA, where the critical path is along N-bit full adders. If all $\hat{c}_i$ are selected to be c_i^{prdt}, the critical path is shortened to roughly L-bit full adders. This large delay reduction can be translated to power reduction by supply voltage scaling. Voltage scaling (reducing supply voltage) on digital circuits will lead to increase in delay. Therefore, we can reduce the supply voltage on SARA to make its critical delay same as that of CRA under normal voltage. As the supply voltage decreases, the power consumption could be reduced. There can be 2^{K-1} different configurations. For two configurations with the same critical path length, obviously we only need the one with higher accuracy. Therefore, there are K effective configurations, with critical path lengths of L-bit, $2L$-bit, $\ldots, K \cdot L \simeq N$-bit full adders. The delay of such configurable design varies according to configured accuracy, which results in different power reduction by voltage scaling.

IV. SARA Error Analysis

In this section, we give a theoretical analysis on the expected error of our SARA design and validate the results by numerical experiments. To make it easier for readers to follow the analysis, we list the parameters used in this section as Table II.

TABLE II
Definition of Parameters for Error Analysis

Parameter	Definition
p_i	propagate bit at bit i
g_i	generate bit at bit i
k_i	kill bit at bit i
c_i	accurate carry-out bit at bit i
c_i^{prdt}	approximate carry-out bit at bit i
$\hat{c}_i$	carry-in bit at bit $i+1$
ER_i^{prdt}	error rate of c_i^{prdt}
$\widehat{ER}_i$	error rate of $\hat{c}_i$

For any bit i in carry-out selectable full adder as in Fig. 3(b), an error in approximate carry-out occurs when $c_i^{\text{prdt}} \neq c_i$. There is only one situation where this error may happen: when $c_{i-1} = 1, p_i = 1, c_i^{\text{prdt}} = 0$, and $c_i = 1$. Then the error rate, or probability of such error, is given as

$$\begin{aligned} \text{ER}_i^{\text{prdt}} &= P\left(c_i^{\text{prdt}} \neq c_i\right) = P\left(c_i^{\text{prdt}} = 0, c_i = 1\right) \\ &= P(c_{i-1} = 1, p_i = 1) \\ &= P(c_{i-1} = 1)P(p_i = 1) \end{aligned} \tag{7}$$

where P indicates the probability and the last part assumes that c_{i-1} and p_i are independent of each other. Then, if the approximate/accurate carry-out can be selected by a MUX gate, the error rate of MUX output $\hat{c}_i$ is

$$\widehat{\text{ER}}_i = P(\hat{c}_i \neq c_i) = \begin{cases} \text{ER}_i^{\text{prdt}}, & \text{if } \hat{c}_i \leftarrow c_i^{\text{prdt}} \\ 0, & \text{if } \hat{c}_i \leftarrow c_i. \end{cases} \tag{8}$$

Let us consider a configuration of SARA in Fig. 4, which has both bit j and bit $i-1$ in approximate mode. For the subadder which calculates addends from bit i to j, its LSB (bit i) is using carry-in configurable full adder, while its MSB (bit j) is in carry-out selectable full adder. According to (7) and (8), the error rate of $\hat{c}_j$ is determined by the probabilities of $c_{j-1} = 1$ and $p_j = 1$

$$\widehat{\text{ER}}_j = P(c_{j-1} = 1)P(p_j = 1). \tag{9}$$

According to the logic of addition, the carry-out bit is calculated by the carry-in and addends. There are two cases which can result in $c_{j-1} = 1$: generate bit g_{j-1} should be 1 in case of carry-in $c_{j-2} = 0$; or kill bit k_{j-1} must be 0 when carry-in comes with $c_{j-2} = 1$. Then, the probability of $c_{j-1} = 1$ can be computed by the probability of $c_{j-2} = 1$ as

$$\begin{aligned} &P(c_{j-1} = 1) \\ &= P(c_{j-2} = 0, g_{j-1} = 1) + P(c_{j-2} = 1, k_{j-1} = 0) \\ &= P(c_{j-2} = 0)P(g_{j-1} = 1) + P(c_{j-2} = 1)P(k_{j-1} = 0) \\ &= [1 - P(c_{j-2} = 1)]P(g_{j-1} = 1) + P(c_{j-2} = 1)P(k_{j-1} = 0). \end{aligned} \tag{10}$$

Similarly, the probability of $c_{j-2} = 1, c_{j-3} = 1, \ldots, c_{i+1} = 1$ can be calculated using the same formula. For the probability of $c_i = 1$, it is little different because the carry-out c_i in our carry-in configurable full adder is based on predicted carry-in $\hat{c}_{i-1}$ instead of c_{i-1}. Considering that bit $i-1$ is configured in approximate mode, we have

$$P(\hat{c}_{i-1} = 1) = P(c_{i-1}^{\text{prdt}} = 1) = P(g_{i-1} = 1). \tag{11}$$

TABLE III
ERROR RATE OF SUBADDER WITH DIFFERENT WIDTHS

Sub-adder length L	Calculated error rate	Simulated error rate
1	1/8 = 0.125	0.1257
2	3/16 = 0.1875	0.1879
3	7/32 = 0.21875	0.2187
4	15/64 = 0.234375	0.2347
5	31/128 = 0.2421875	0.2424
6	63/256 = 0.24609375	0.2464

Then, the probability of $c_i = 1$ can be expressed as

$$P(c_i = 1) = [1 - P(g_{i-1} = 1)]P(g_i = 1) + P(g_{i-1} = 1)P(k_i = 0). \quad (12)$$

By expanding (10) recursively till bit i, the probability of $c_{j-1} = 1$ can be calculated by a function of generate bit and kill bit from bit $i-1$ to bit $j-1$

$$P(c_{j-1} = 1) = f\{P(g_{i-1} = 1), \ldots, P(g_{j-1} = 1) P(k_i = 0), \ldots, P(k_{j-1} = 0)\}. \quad (13)$$

Assuming that the inputs for adder are uniformly distributed random numbers, we have $P(g = 1) = 1/4$ and $P(k = 0) = 3/4$. As the length of subadders varies from 1 to 6, the error rates of $\hat{c}_j$ calculated by (9) are listed in the second column of Table III. Corresponding data from numerical simulation in MATLAB are also presented in the last column. The error rates calculated by our method match well with experimental results, which demonstrates the correctness of our mathematical analysis. We can also observe that as the length of subadder increases the error rate is bounded by 0.25. That is because when the length of subadder comes to infinite the probability of $c = 1$ will become 0.5, same as the normal carry in accurate adder.

Theorem 1: If $\mathcal{I}$ is the set of bits with MUX at output, the expected error of SARA for unsigned integers is

$$\sum_{\forall i \in \mathcal{I}} \widehat{\mathrm{ER}}_i \cdot P(p_{i+1} = 1) \cdot 2^{i+1}.$$

Proof: The overall expected error of SARA can be calculated by summing respective error introduced by every approximate bit from LSBs to MSBs. But the propagation of inaccurate carry bit may cause error in higher bit, which also be counted in the calculation of lower bit. Therefore, we need to exclude those errors to avoid overcalculation in the total error.

Let us consider the SARA design in Fig. 4 which have approximate configuration at both bit $i-1$ and bit j. Assuming that bit $i-1$ is the lowest bit configured in approximate mode, we know that all sum bits s_k ($k \in [1, i-1]$) as well as carry bit c_{i-1} are accurate

$$c_{i-1} = c_{i-1}^{\mathrm{acc}}. \quad (14)$$

Then, the probability that carry prediction at MUX output $\hat{c}_{i-1}$ mismatches with accurate carry c_{i-1}^{acc} should be the same as the error rate of MUX output $\hat{c}_{i-1}$

$$P\left(\hat{c}_{i-1} \neq c_{i-1}^{\mathrm{acc}}\right) = P(\hat{c}_{i-1} \neq c_{i-1}) = \widehat{\mathrm{ER}}_{i-1}. \quad (15)$$

According to the structure of carry-in configurable full adder [Fig. 3(c)], sum bit s_i calculated from c_{i-1} is always accurate; however, the carry-out bit c_i becomes conditionally accurate, which depends on both carry-in bit and propagate bit. As shown in (16), the scenario of accurate carry-out can be attributed to two conditions: when the carry-in is not accurate, the carry-out bit becomes accurate as the propagate bit is false; otherwise, it must be accurate no matter what kind of addends are given

$$P\left(c_i = c_i^{\mathrm{acc}}\right) = P\left(\hat{c}_{i-1} = c_{i-1}^{\mathrm{acc}}\right) + P\left(\hat{c}_{i-1} \neq c_{i-1}^{\mathrm{acc}}\right) P(p_i = 0). \quad (16)$$

Its complementary part, the probability of inaccurate carry c_i, can be expressed as

$$\begin{aligned} P\left(c_i \neq c_i^{\mathrm{acc}}\right) &= P\left(\hat{c}_{i-1} \neq c_{i-1}^{\mathrm{acc}}\right) P(p_i = 1) \\ &= P(\hat{c}_{i-1} \neq c_{i-1}) P(p_i = 1) \\ &= \widehat{\mathrm{ER}}_{i-1} \cdot P(p_i = 1). \end{aligned} \quad (17)$$

As a result, the approximation at bit $i-1$ would cause an inaccurate carry-in c_i at bit $i+1$, which introduces the magnitude of 2^i to the overall error in final result. Then, the expected error introduced by approximation at bit $i-1$ can be estimated by

$$E[e_{i-1}] = P\left(c_i \neq c_i^{\mathrm{acc}}\right) \cdot 2^i = \widehat{\mathrm{ER}}_{i-1} \cdot P(p_i = 1) \cdot 2^i. \quad (18)$$

Next, we consider the expected error introduced by approximation at bit j. As bit j is not the lowest bit in approximate mode, there is a chance that the propagation of inaccurate carry from bit $i-1$ induces error at bit j, while it has be taken into account in the error calculation of bit $i-1$. Then, the problem is whether the carry c_j is accurate when there is a mismatch between $\hat{c}_j$ and c_j. If not, we need to exclude the impact from lower bit when estimating the error at bit j. Let us answer this question in the following cases.

1) *Case 1:* If any propagate bit in subadder (bit i to j) equals 0, the error propagation by inaccurate carry will be paused. In another word, the error carried by inaccurate carry bit cannot be propagated to higher bit anymore, because the carry-out is independent of carry-in when propagate bit is false. In this case, the carry c_j should be always accurate regardless of the configuration at bit j.
2) *Case 2:* If all propagate bits of subadder equal 1, the value of inaccurate carry $\hat{c}_{i-1}$ (0 instead of 1) will be propagated to c_j. In this situation, the actual value of c_j propagated from bit $i-1$ must be 0, while the accurate value should be 1. Assuming that $\hat{c}_j$ mismatches with c_j, we can state that the value of $\hat{c}_j$ must be 1. However, it conflicts with the generation of c_j, because carry c_j is the logical conjunction of $\hat{c}_j$ and $p_j \cdot c_{j-1}$. So there should be no mismatch between $\hat{c}_j$ and c_j in this case.

In conclusion, when there is a mismatch between $\hat{c}_j$ and c_j, the value of carry c_j must be accurate. We can further conclude that the contributions of every approximate bit to the total error are independent to each other. Similar to bit $i-1$, the expected error at bit j can be estimated by

$$E[e_j] = \widehat{\mathrm{ER}}_j \cdot P(p_{j+1} = 1) \cdot 2^{j+1}. \quad (19)$$

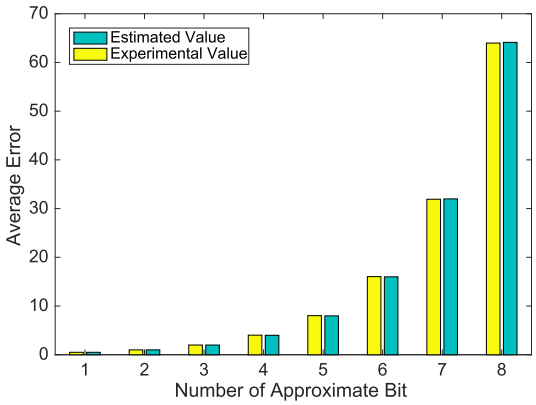


Fig. 6. Average error of 9-bit SARA in different configurations.

Thus, the total error can be obtained by summing up the errors, respectively, introduced by every approximate bit

$$E = \sum_{\forall i \in \mathcal{I}} E[e_i] = \sum_{\forall i \in \mathcal{I}} \widehat{\mathrm{ER}}_i \cdot P(p_{i+1} = 1) \cdot 2^{i+1}. \quad (20)$$

■

If input addends are random variables following uniform distribution, the expected error of SARA is given as

$$E = \sum_{\forall i \in \mathcal{I}} \widehat{\mathrm{ER}}_i \cdot 2^i. \quad (21)$$

We can verify (21) by numerical simulation of a 9-bit SARA design. In our experiment, SARA consists of nine subadders, whose width is 1 bit. The results are from 200 K run of Monte Carlo simulation with uniform distributed numbers as input. As shown in Fig. 6, there are two sets of data for comparison, experimental data are obtained directly in experiments, and estimated data are calculated by (21). The average errors from experiment are almost the same as the estimated values. According to the analysis described earlier, we can estimate the average error of SARA in any configuration, given the distribution of input numbers.

Since $|\mathcal{I}| = K-1$, the error of the worst case approximation mode increases with the number of subadders K. In addition, area overhead increases with K. On the other hand, a large K implies smaller L, and thus often facilitates shorter critical path and more power reductions. Therefore, K significantly affects the tradeoff among accuracy, power, delay, and area.

V. Delay-Adaptive Reconfiguration of SARA

Almost all previous works on ACA [15]–[19] reasonably assume that accuracy configuration is decided by architecture/system-level applications. We propose a self-configuration technique for the scenarios where architecture/system-level choice is either unclear or difficult. The simulation results show that SARA with the self-configuration outperforms several previous static approximate adder designs.

The main idea of self-configuration is based on the observation that the actual worst case path delay depends on addend values. Specifically, the actual path delay is large only when a carry is propagated through several consecutive bits.

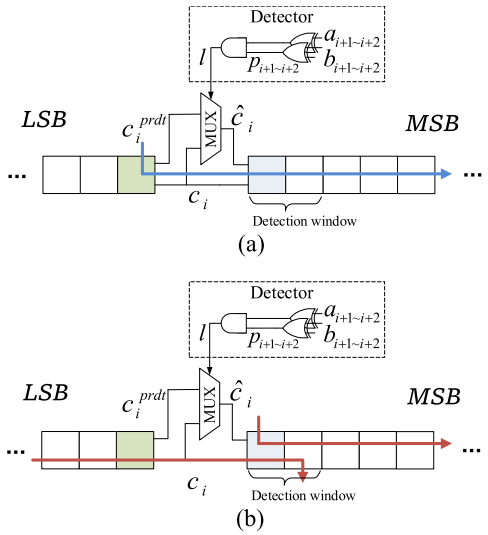


Fig. 7. Design of DAR for SARA operating in (a) approximate mode and (b) accurate mode.

Any false propagate bit from the addends results in a shorter carry propagation chain. When the actual carry propagation chain is short, there is no need to use approximation configuration, which is intended to cut carry chain shorter. We propose a **DAR** technique: the output of a MUX in SARA is set to approximation mode only when a potentially long carry chain is detected. Compared to the constantly approximate configuration, some errors for actual short carry chains are avoided, the actual long carry chain is cut shorter, and delay/power reduction can be still obtained.

The long carry chain detection and SARA-DAR design are shown in Fig. 7(a). When MUX is switched to accurate mode by any false propagate bit in detection window, the actual carry chain is retained by the position of false propagate bit. To obtain a shorter carry chain in accurate mode, the detection window for MUX at bit i in MSB should start from bit $i+1$. In Fig. 7, we use a detection window of 2 bits (p_{i+1} and p_{i+2}) to tell if there is a carry propagation across two subadders, and configure the MUX according to

$$\hat{c}_i \leftarrow \begin{cases} c_i^{\mathrm{prdt}}, & \text{if } p_{i+1} \cdot p_{i+2} \text{ is true} \\ c_i, & \text{otherwise.} \end{cases} \quad (22)$$

In approximation mode, the effective carry chain is represented by the blue line in Fig. 7(a) and its length is no greater than $L+1$ bits. When the MUX is set to accurate mode, the carry chain is indicated by the red lines in Fig. 7(b) and their lengths can be restrained to within $L+2$ bits. Since the propagate bits only depend on local primary inputs, we can reuse propagate bits in higher bits to save cost. Note that in this case the detection overhead here is almost the minimum possible, i.e., only one NAND gate for configuring each MUX.

In Fig. 7, we use 2-bit detection window, which can be generalized to W bit. Then, the error rate for MUX at bit i becomes

$$\widehat{\mathrm{ER}}_i^{\mathrm{dar}} = \mathrm{ER}_i^{\mathrm{prdt}} \cdot \prod_{j=1}^{W} P(p_{i+j} = 1). \quad (23)$$

The detection window size W decides the tradeoff between the accuracy and the effective carry chain length in accurate mode, which is $L + W$. When W increases, the error rate decreases while the critical path length in accurate mode increases.

VI. Experimental Results

A. Experiment Setup and Evaluation

Our SARA, SARA-DAR, and several previous designs are synthesized to 32-bit adders by Synopsys Design Compiler using the Nangate 45 nm Open Cell Library. The synthesized circuits are placed and routed by Cadence Encounter. The default supply voltage level is 1.25 V. To make fair comparisons across architectures, we describe all designs by structural modeling in Verilog to reduce the impact of synthesis and optimization. For comparison, we synthesize the accurate adder in behavioral modeling, which is described by expressional operator in Verilog. The netlist of such accurate adder should be automatically optimized by synthesizer in design complier, which is different from any man-craft gate-level design. In addition, we set the same supply voltage and no delay constraint on all designs for the same reason.

The evaluation of ACA designs can be subtle and therefore is worth some discussion.

1) *Area:* In the literature, the area sometimes refers to the part of the circuit working in a certain mode, e.g., the circuit for the accurate part is not included in area estimation when evaluating approximation mode. We report the routed layout area of each entire design.
2) *Delay:* Some configurable adders, such as ACA [15] and GeAr [16], implement error correction with pipelining, which sometimes takes multiple clock cycles to determine the complete result. The delay or performance evaluation of such designs is much more complicated than unpipelined designs. Our work is focused on unpipelined implementation, although it can be pipelined. Thus, the reported delay is the maximum combinational logic path delay obtained from Synopsys PrimeTime with consideration of wire delay.
3) *Power:* The power dissipation is estimated by Synopsys PrimeTime considering both static and dynamic power.
4) *Accuracy:* We use PSNR, where errors are treated as noise, as a composite accuracy metric for considering both error magnitude and error rate. In addition, the worst case error, which is equivalent to the maximum error magnitude [13], and error rate are also reported. Each error result is from 100K-run MATLAB-based Monte Carlo simulation assuming uniform distribution of addends.
5) *Tunability:* This means the range and granularity of runtime accuracy configurations. Sometimes, this can be confused with design-time flexibility.
6) *Tradeoff:* The tradeoff among the above-mentioned factors is complex and is difficult to capture in a simple picture. To this end, we use composite metrics including PDP, energy-delay product (EDP), and iso-delay power.

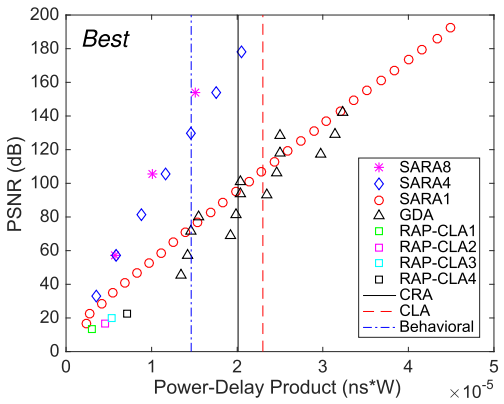


Fig. 8. SARA: PSNR versus PDP.

B. Results of Tradeoff for Different Configurations

In this subsection, we mainly compare the following accuracy configurable adder designs.

1) GDA [18]: We use the same design as in [18], where each subadder has 4 bits. This design can be configured by choosing accurate or predicted carry-out for each subadder. The carry prediction at each segment can also be configured to different accuracy levels by using the different number of lower bit addends.
2) RAP-CLA [19]: We implement four different designs with carry prediction bit-width from 1 to 4 bits, which is reflected in the name. For example, RAP-CLA2 means each of its carry prediction is from its two lower bits. As in [19], each design can be configured to either only one approximation mode or accurate mode.
3) SARA: This is our proposed design, and we evaluate subadder bit-width of 1, 4, and 8 bits, referred to as SARA1, SARA4, and SARA8, respectively.

The main result is shown in Fig. 8, where each point is from one configuration of one design. The computation accuracy is evaluated by PSNR, while the conventional design objectives are characterized by PDP. A design and configuration is ideal if it has large PSNR but low PDP, i.e., northwest in the figure. PDPs of two classic accurate designs, CRA and CLA, are indicated by the two vertical lines as their PSNR is near infinity. The result of SARA working in completely accurate mode is unable to be presented in the figure, because its infinite PSNR cannot be displayed as a single dot in the plot. Evidently, the best solutions are from SARA4 and SARA8. At 100-dB PSNR, the PDP of SARA4 and SARA8 is about a half of GDA or CRA. The solutions from RAP-CLA, the latest previous work, are also largely dominated by SARA in PSNR-PDP tradeoff. An interesting case is SARA1. Its tradeoff is similar as GDA and not as good as SARA4 or SARA8. However, its runtime tunability is superior to all the other designs. It has the largest tuning range, the finest tuning granularity, and very smooth tradeoff.

Figs. 9 and 10 show the tradeoff between error magnitude and PDP. Ideally, a better design or configuration has smaller average error or worst case error with lower PDP, which can be marked in the lower left corner of the figure.

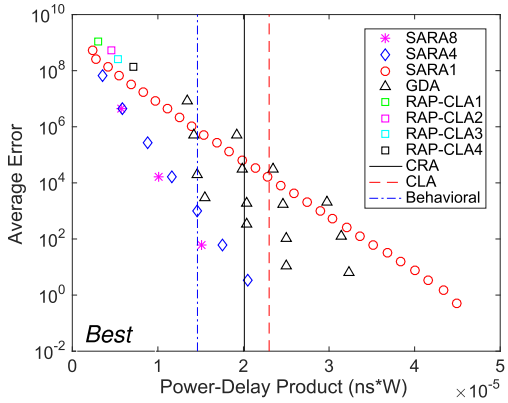


Fig. 9. SARA: average error versus PDP.

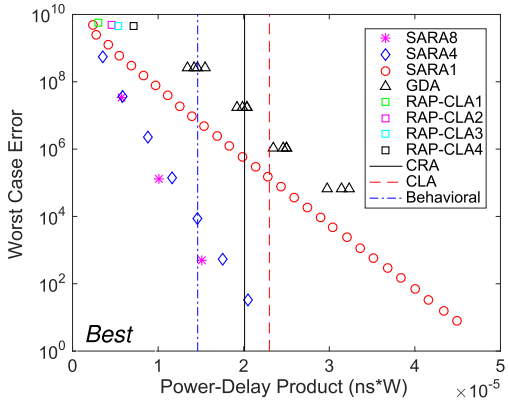


Fig. 10. SARA: worst case error versus PDP.

In Fig. 9, SARA4 and SARA8 dominate other designs in average error-PDP tradeoff. For each configuration, SARA4 and SARA8 have almost the lowest average error at a certain PDP level. Although SARA1 cannot achieve superior average error and PDP tradeoff to GDA, it shows fine-grant tunability in a large range same as PSNR-PDP tradeoff. Fig. 10 depicts the worst case error versus PDP and confirms the trend observed in the PSNR-PDP tradeoff. All SARA designs even for SARA1 have lower worst case error than previous work at the same PDP level. In addition, the result of SARA working in accurate mode cannot be found in the plot. That is because the y-axis is in logarithmic scale and zero error will be converted into infinite which cannot be displayed as a single dot.

EDP is another metric to efficiently evaluate tradeoffs between circuit level power saving techniques for digital designs. Figs. 11–13 illustrate accuracy versus EDP, which have similar trend in accuracy-PDP tradeoff. Most configurations of SARA4 and SARA8 have lower EDP than accurate adder CRA and CLA. At a certain EDP level, SARA4 and SARA8 still dominate GDA and RAP-CLA with larger PSNR, smaller average error, or worst case error. SARA1 in different configurations covers the range from lowest to highest EDP, which provides finest tunability in accuracy-energy tradeoff among different architectures.

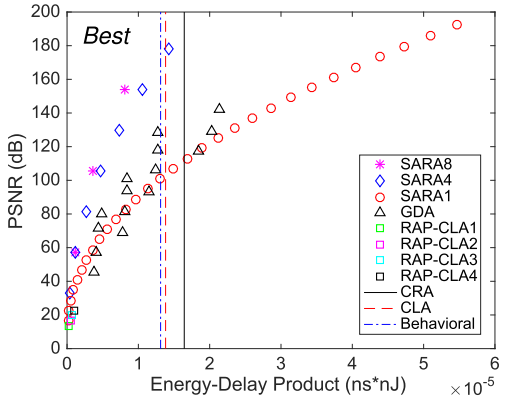


Fig. 11. SARA: PSNR versus EDP.

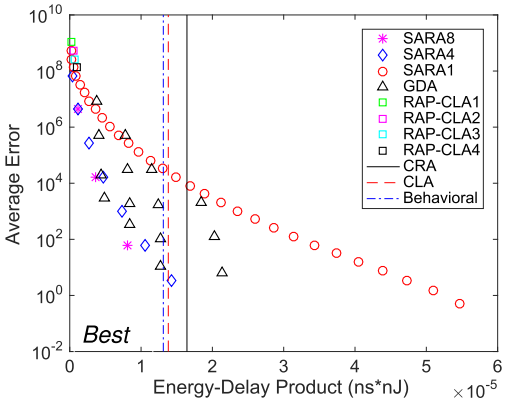


Fig. 12. SARA: average error versus EDP.

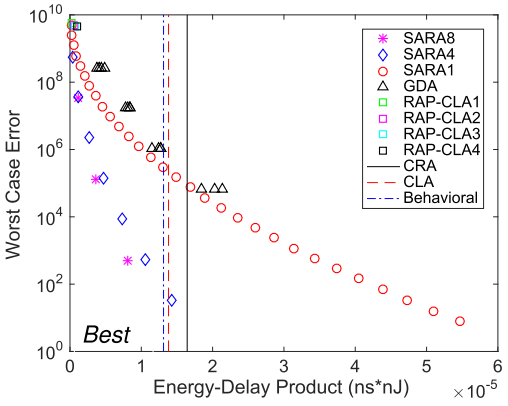


Fig. 13. SARA: worst case error versus EDP.

C. Results of Tradeoff for Delay-Adaptive Reconfiguration

This part is to evaluate the SARA-DAR design, where the configuration decision has already been made. Hence, it makes sense to additionally compare with static approximate adders, where no configuration is needed. Static approximate adder designs including ETAII [8], FICTS [13], and AFICTS [13] are implemented in the experiment. In addition, CRA-based approximate designs CRA-trunc$i$ implemented by ignoring lowest i bits in addends are presented, which is a simple but good baseline for comparison. Seven SARA4-DAR designs are obtained based on seven configurations of SARA4 with

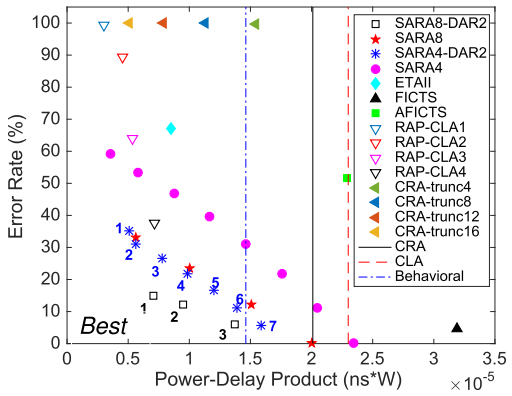


Fig. 14. SARA-DAR: error rate versus PDP.

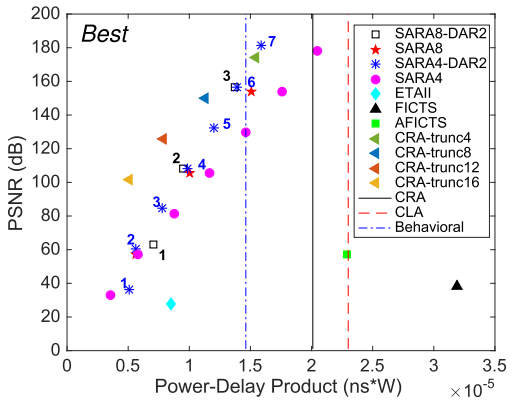


Fig. 15. SARA-DAR: PSNR versus PDP.

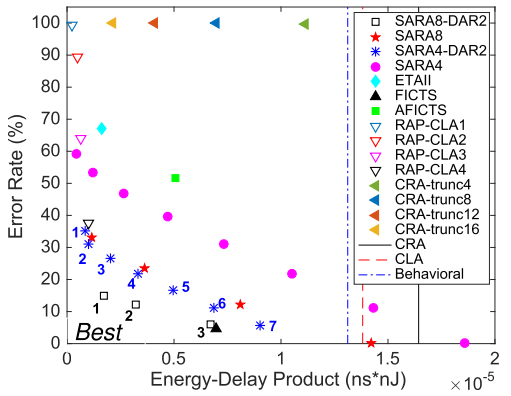


Fig. 16. SARA-DAR: error rate versus EDP.

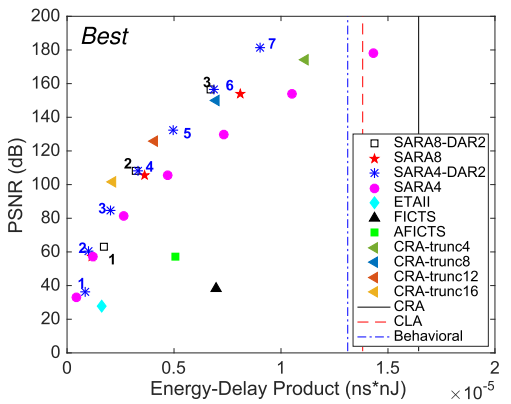


Fig. 17. SARA-DAR: PSNR versus EDP.

a detection window of 2 bits, while three SARA8-DAR are based on different configurations of SARA8. That is, if a MUX at bit i is configured to accurate carry in SARA4/SARA8, bit i of corresponding SARA-DAR is hard-wired to accurate carry without using MUX. When bit j in SARA4/SARA8 is in approximation mode, bit j of corresponding SARA-DAR uses the DAR.

Fig. 14 shows the error rate versus PDP tradeoff. The dot of SARA4-DAR2 labeled with "1" represents the counterpart of SARA4 when all the MUXes are controlled by DAR. When we remove the MUX at the highest bit to propagate accurate carry and keep others in DAR, another SARA4-DAR2 design could be obtained (another dot labeled with "2" in the figure). If we go on to remove more MUXes in MSB, a series of SARA4-DAR2 designs shown as dots with label "3" to "7" can be obtained. Three SARA8-DAR2 designs are created in the same way. According to the figure, the error rate of SARA is mostly lower than RAP-CLA. By using DAR, SARA-DAR often has less error rate and PDP than SARA. SARA-DAR also greatly outperforms the static approximate adders in both error rate and PDP. Moreover, in Fig. 14, we can observe those dots right on the x-axis, which represent SARA4 and SARA8 working in accurate mode. Both of them achieve zero error rate. PDP of accurate SARA8 is about $2 \times 10^{-5} ns \cdot W$, while PDP of accurate SARA4 is almost $2.3 \times 10^{-5} ns \cdot W$. In Fig. 15, SARA-DAR also demonstrates better PSNR-PDP tradeoff than other designs, except for comparing with CRA-trunc at some low PSNR levels. However, CRA-trunc has almost 100% error rate since it dismisses lower bits in addends, which is the worst among all static approximate adders. Figs. 16 and 17 show tradeoff between accuracy and EDP. At a certain EDP level, SARA-DAR has almost the same PSNR as CRA-trunc, which is the best among all static approximate adders.

D. Impact of Detection Window in Delay-Adaptive Reconfiguration

This subsection shows the impact of detection window in the tradeoff for DAR. According to (23), the error rate of MUX output can be reduced by DAR. As the length of detection window increases, the error rate would decrease because there are less probability that MUX is configured in approximate mode. As a result, the overall error rate varies with the size of detection window. Figs. 18 and 19 show the changes of error rate and PSNR of SARA4-DAR with different detection windows. As the size of detection window increases from 1 to 3, the error rate decreases compared to its SARA counterpart. However, we can observe that the gap of error rate between SARA4-DAR2 and SARA4-DAR1 varies with different configurations. Although the change in error rate for individual MUX of SARA-DAR is proportional to the size of detection window [as shown in (23)], the overall error rate in output results might not show linear change. When the size of detection window increases by 1,

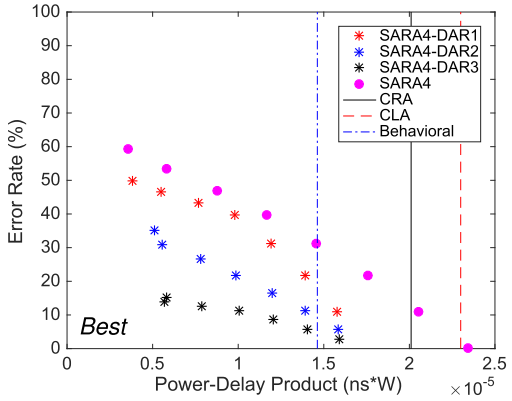


Fig. 18. Error rate of SARA4-DAR with a different detection window.

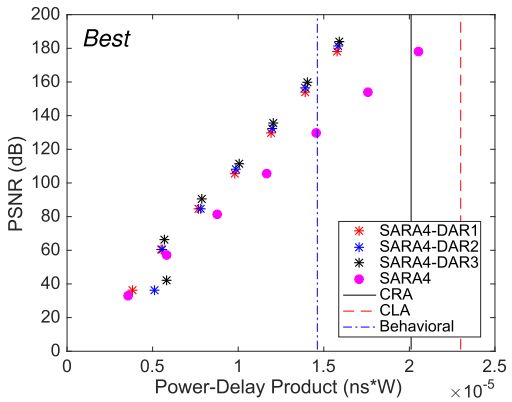


Fig. 19. PSNR of SARA4-DAR with a different detection window.

PSNR of SARA4-DAR increases by about 3 dB on average. We can also find that the PDP gap between SARA4-DAR2 and SARA4-DAR1 varies with different configurations in both figures. The change of PDP between SARA4-DAR2 and SARA4-DAR1 in most configurations is very small, while it is larger in the first configuration (which is presented as the first dot of SARA4-DAR in the left of the figures). It is mainly attributed to unproportioned change in delay between different configurations.

E. Results of Iso-Delay Power and Area

Although PDP results have been shown in Sections VI-B and VI-C, the tradeoff between power and delay is still unclear. The power-delay tradeoff can be obtained by different accuracy configurations or varying supply voltages. Different combinations of configurations and voltages may lead to overwhelming volume of results, which are difficult to interpret, especially when implication to accuracy is involved at the same time. Thus, we indicate the tradeoff by investigating the iso-delay power, which is the power of each circuit tuned to the same critical path delay (0.82 ns) by voltage scaling. The results are shown in Fig. 20. In general, SARA4, SARA8, and SARA4-DAR can achieve much lower power than CRA. Although GDA and RAP-CLA seem to provide low power, their PSNR is much less than our designs. Compared at the same iso-delay power level, SARA has more than 20-dB increase in PSNR than RAP-CLA, while GDA has more than 70-dB decrease than SARA designs.

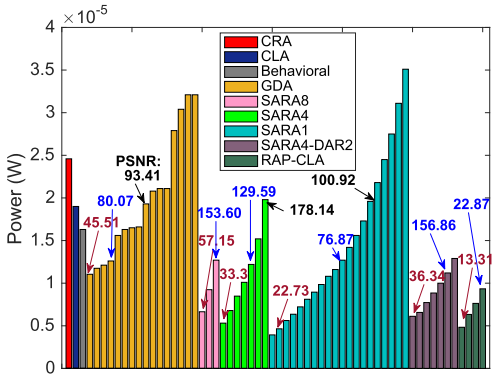


Fig. 20. Iso-delay power comparison. The numbers are PSNR.

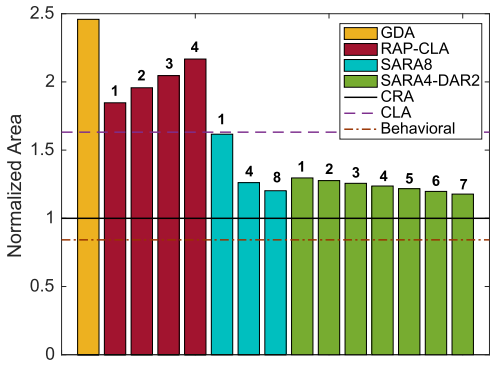


Fig. 21. Area comparison.

SARA1 shows a large range of iso-power tuning, which could reach the lowest and highest power among all adders. We do not have iso-delay power for approximate adders working in accurate mode, because the delay of such case is larger than CRA due to induction of MUXes, which cannot provide sufficient room for reducing supply voltage.

Last but not least, we compare area of these designs in Fig. 21. Same as our expectation, GDA and RAP-CLA have greater area than CLA, while area of SARA4 or SARA8 is significantly smaller than CLA. SARA1 has almost the same area as CLA due to MUXes in every bit which aids the accuracy configuration. On average, the area of SARA is 39% smaller than that of RAP-CLA and 50% smaller than that of GDA.

VII. Applications

A. Extension to Multiplier

In complicated datapath system, multiplier is considered as a much bigger component in power consumption. Our carry-prediction-based approximation uses generate bit to predict the carry from lower segments. The critical delay can be restrained to a smaller value with shorter critical path in carry propagation. Further extension of our technique to multiplier depends on the multiplication structure used in hardware implementation. There is a variety of hardware designs for multiplication, according to the structures of reduction tree. In this section, we apply our technique on three kinds of multiplication structures including array multiplier, Wallace multiplier, and Dadda multiplier.

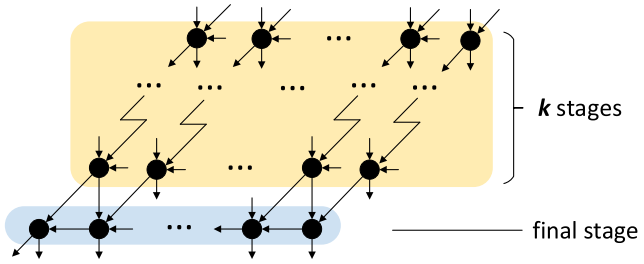


Fig. 22. Basic structure of multiplier.

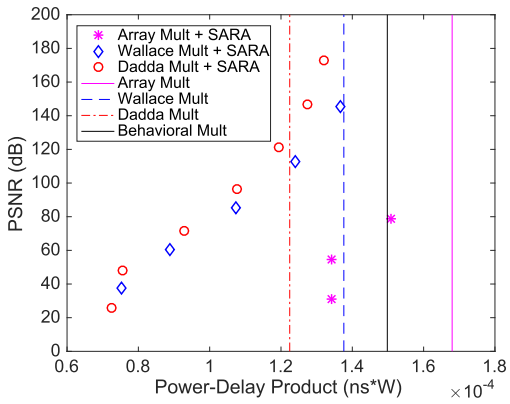


Fig. 23. Multiplier: PSNR versus PDP.

As shown in Fig. 22, the basic structure of multiplier employs a three-step process to multiply two integers.

Step 1) Generate all partial products by using an AND gate array.

Step 2) Combine the partial products in k stages by layers of half/full adder until the matrix height is reduced to two. Different types of structures depend on the reduction tree used to reduce the number of partial products in this step.

Step 3) Sum the resulting numbers in the final stage by a conventional adder.

In array multiplier, the carry bits in one stage are propagated diagonally downward, which follows the basic shift-and-add multiplication algorithm. Wallace multiplier based on Wallace tree combines the partial products as early as possible, which makes it faster than array multiplier [20]. Dadda's strategy is to make the combination take place as late as possible, which leads to simpler reduction tree and wider adder in final stage [20]. Thus, we can design approximate multipliers by using our SARA design instead of CRA in the final stage.

Three types of 16 × 16 multipliers (array multiplier, Wallace multiplier, and Dadda multiplier) as well as behavioral multiplier are synthesized and implemented by using Nangate 45-nm Open Cell Library. Their error data are obtained from 100-K-run Monte Carlo simulation with uniform distribution of operands. In approximate multiplier, the final stage uses SARA4, which consists of subadders with a bit-width of 4 bits, while the accurate one uses CRA. Figs. 23 and 24 present the tradeoff between error and PDP. Most of approximate multipliers configured in approximate mode have better PDP

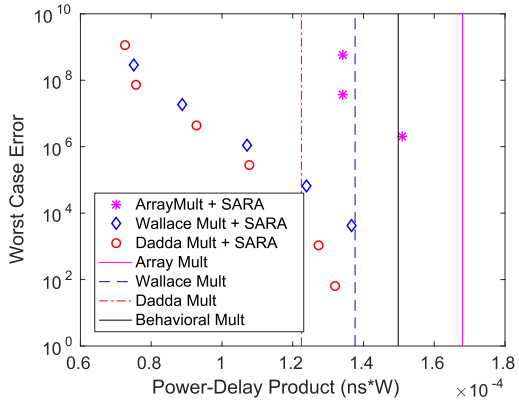


Fig. 24. Multiplier: worst case error versus PDP.

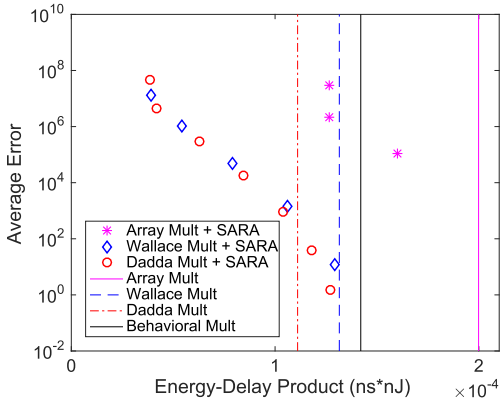


Fig. 25. Multiplier: average error versus EDP.

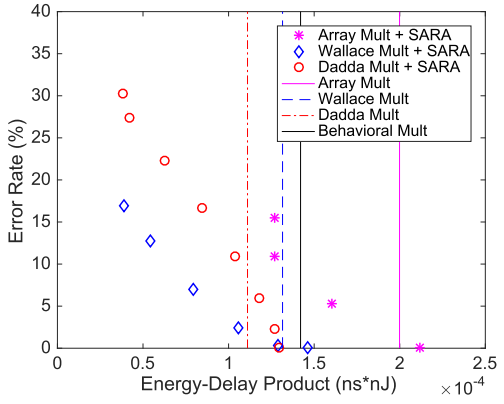


Fig. 26. Multiplier: error rate versus EDP.

compared with the accurate multipliers. The variance of error between different approximate modes in approximate multiplier has similar trend as SARA. Total error increases as more bits are configured in approximate mode. Approximate array multiplier shows larger error than approximate Wallace/Dadda multiplier at the same PDP level. It is because array multiplier has larger critical delay from internal stages in step 2 than Wallace/Dadda multiplier.

Figs. 25 and 26 show the error versus EDP for both accurate and approximate multipliers. As more MUXes are set to propagate approximate carry, the average error in output

Fig. 27. Comparison of image lenna. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

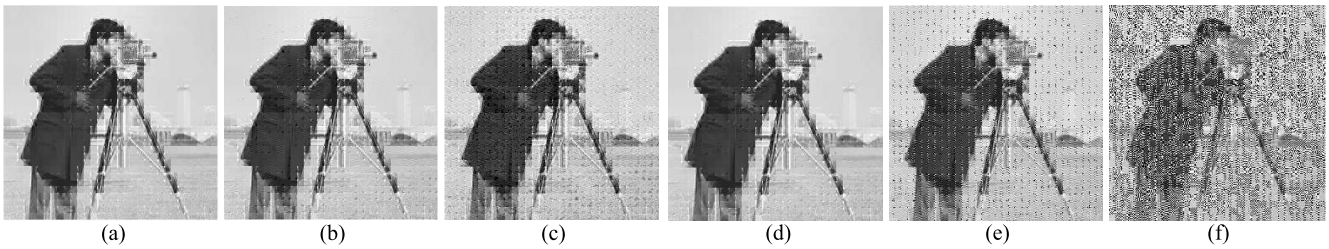


Fig. 28. Comparison of image cameraman. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

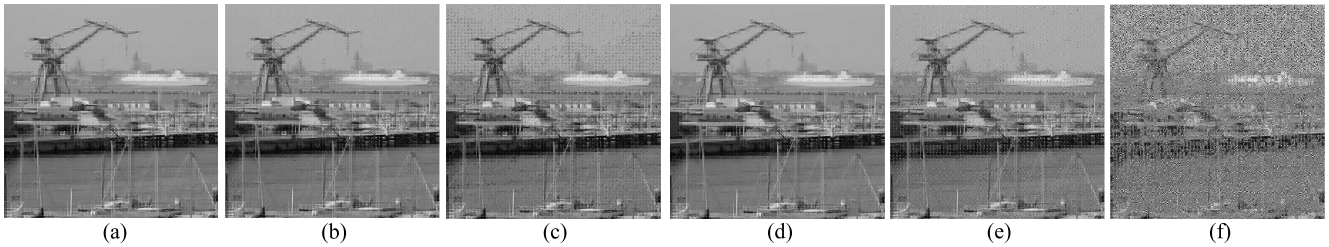


Fig. 29. Comparison of image kiel. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

increases to about 10^7, which as well achieves the best EDP. The worst case error rate of approximate Dadda multiplier is about 30%, while it comes to about 17% for approximate array multiplier and Wallace multiplier. As shown in Fig. 26, when approximate multipliers are working in completely accurate mode (error rate equals 0), EDP is larger than that of their accurate counterpart. In summary, the experimental results show that our technique can be successfully extended to high-speed multiplier designs. And due to the simple but effective structure of SARA, it provides an easy way for us to convert conventional multiplier into approximate design.

B. DCT Computation in Image Processing

DCT has been recognized as the basic in many transform coding methods for image and video signal processing. It is used to transform the pixel data of image or video into corresponding coefficients in frequency domain. Since human visual system is more sensitive to the changes in low frequency, the loss of accuracy in high-frequency components does not heavily degrade the quality of image processed by DCT. In addition, those components in different frequency have different tolerances to the degradation in the original data. It is a good example to show the reconfigurability of our design by applying them in VLSI implementation of DCT, computing in JPEG image compression.

The 2-D DCT is implemented by the row–column decomposition technique, which contains two stages of 1-D DCT [21]–[23]. The 2-D DCT of size $N \times N$ could be defined as

$$Z = C^t X C \tag{24}$$

where C is a normalized Nth-order matrix and X is the data matrix. Generally, the image is divided into several $N \times N$ blocks and each block is transformed by 2-D DCT into frequency domain components. The VLSI implementation of DCT computing contains a set of ROM and accumulator components, which can be implemented by multipliers and adders [21]–[23]. In this application, we use approximate adders to replace those accurate ones in CRAs to implement an imprecise, but low-power circuit for image processing that contains DCT computing.

We replace the adders in circuits with different configurations of SARA, SARA-DAR, GDA, as well as RAP-CLA. The results are obtained by numerical simulations on four images (Fig. 27–30) in MATLAB. As we know, after DCT process, data in different frequency domain have a different level of error tolerance. As shown in Fig. 31, matrix components in the top-left corner correspond to lower frequency coefficients that are sensitive to human vision, while those components in bottom-right corner might allow more errors.

To utilize this feature for better energy-accuracy tradeoff, we make following configuration for different designs.

1) *SARA4:* SARA4 with 4, 3, 2, 1 consecutive segments working in accurate mode are used to compute components in S_1, S_2, S_3, and S_4, respectively.

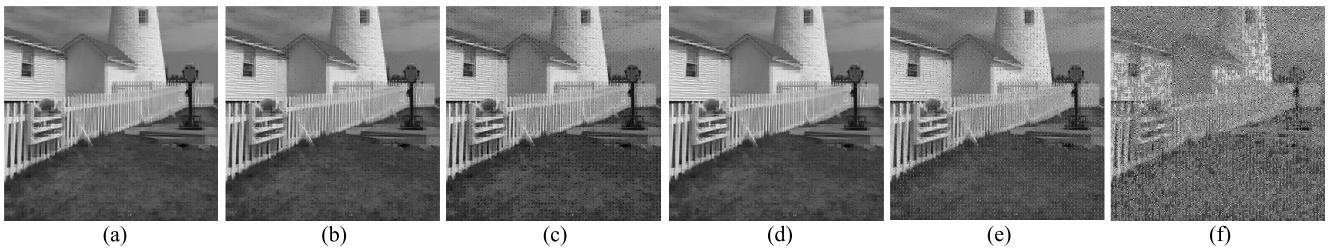


Fig. 30. Comparison of image house. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

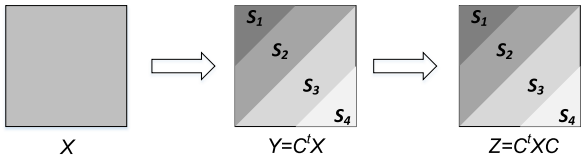


Fig. 31. 2-D DCT.

TABLE IV
IMAGE QUALITY COMPARISON IN PSNR

	lenna	cameraman	kiel	house	AVERAGE
Accurate	39.85	38.23	37.68	37.35	38.27
SARA4	38.32	37.50	36.83	36.53	37.30
SARA8	35.33	35.07	34.92	34.81	35.03
SARA4-DAR2	39.45	37.90	37.43	37.00	37.97
GDA	34.53	34.55	34.88	34.20	34.54
RAP-CLA	33.38	33.44	33.51	33.39	33.43

2) *SARA8:* SARA8 with one segment in accurate mode is used to compute components in S_1 and S_2, while another configuration with all segments in approximate mode are for S_3 and S_4.
3) *SARA4-DAR2:* DAR counterpart of SARA4 with a detection window of 2 bits.
4) *GDA:* $GDA_{4,1}$, $GDA_{3,1}$, $GDA_{2,1}$, and $GDA_{1,1}$ (same notation as [18]) are used to compute components in S_1, S_2, S_3, and S_4, respectively.
5) *RAP-CLA:* Since RAP-CLA can work in one approximate mode, we use RAP-CLA with window sizes of 20, 16, 12, and 8 to compute components in S_1, S_2, S_3, and S_4.

The image processing results are shown in Table IV. PSNR in the table is defined via the mean squared error (MSE). Given an $m \times n$ image I and its restored image K, MSE, and PSNR are defined as

$$\text{MSE} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2 \tag{25}$$

$$\text{PSNR} = 20 \cdot \log(\text{MAX}_I) - 10 \cdot \log(\text{MSE}) \tag{26}$$

where MAX_I is the maximum pixel value of the image. SARA4-DAR2 has the highest PSNR for every image among all configurable adders, which is close to the quality of accurate adder. Comparing SARA8 with GDA, they have similar PSNR and similar delay, but SARA8 has less power consumption according to the analysis in Section VI. SARA4-DAR2 achieves better image quality than SARA4, but might result in more power due to additional logics for self-configuration. The image quality for different adders in DCT computing can also be demonstrated in Figs. 27–30. According to human vision, SARA and its DAR counterpart show better image quality than GDA and RAP-CLA in JPEG compression processing.

VIII. CONCLUSION

In this paper, we propose an SARA design. It has significantly lower power/EDP than the latest previous work when comparing at the same accuracy level. In addition, SARA has considerable lower area overhead than almost all the previous works. The accuracy-power-delay efficiency is further improved by a DAR technique. We demonstrate the efficiency of our adder in the applications of multiplication circuits and DCT computing circuits for image processing.

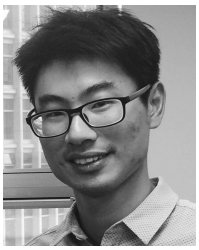
ACKNOWLEDGMENT

The authors would like to thank Dr. D. M. Walker from Texas A&M University for helpful discussions.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. IEEE Eur. Test Symp.*, May 2013, pp. 1–6.
[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.
[3] V. Chippa, A. Raghunathan, K. Roy, and S. Chakradhar, "Dynamic effort scaling: Managing the quality-efficiency tradeoff," in *Proc. Design Autom. Conf. (DAC)*, Jun. 2011, pp. 603–608.
[4] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, Mar. 2004.
[5] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, 2013, pp. 1367–1372.
[6] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, 2012, pp. 1257–1262.
[7] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, 2008, pp. 1250–1255.
[8] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. Int. Symp. Integr. Circuits*, 2009, pp. 69–72.
[9] 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," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.
[10] N. Zhu, W. L. Goh, and K. S. Yeo, "Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications," in *Proc. Int. SoC Design Conf. (ISOCC)*, 2011, pp. 393–396.
[11] G. Liu, Y. Tao, M. Tan, and Z. Zhang, "CASA: Correlation-aware speculative adders," in *Proc. Int. Symp. Low Power Electron. Design*, 2014, pp. 189–194.

[12] J. Hu and W. Qian, "A new approximate adder with low relative error and correct sign calculation," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, 2015, pp. 1449–1454.
[13] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proc. Int. Conf. Comput-Aided Design (ICCAD)*, 2012, pp. 728–735.
[14] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel, "An area-efficient consolidated configurable error correction for approximate hardware accelerators," in *Proc. Design Autom. Conf. (DAC)*, 2016, pp. 1–6.
[15] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. Design Autom. Conf. (DAC)*, 2012, pp. 820–825.
[16] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. Design Autom. Conf. (DAC)*, 2015, pp. 1–6.
[17] V. Benara and S. Purini, "Accurus: A fast convergence technique for accuracy configurable approximate adder circuits," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2016, pp. 577–582.
[18] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2013, pp. 48–54.
[19] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "RAP-CLA: A reconfigurable approximate carry look-ahead adder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, to be published.
[20] W. J. Townsend, E. E. Swartzlander, and J. A. Abraham, "A comparison of Dadda and Wallace multiplier delays," *Proc. SPIE*, vol. 5205, pp. 552–560, Dec. 2003.
[21] S. Yu and E. E. Swartziander, "DCT implementation with distributed arithmetic," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 985–991, Sep. 2001.
[22] M.-T. Sun, T.-C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16×16 discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 36, no. 4, pp. 610–617, Apr. 1989.
[23] D. Gong, Y. He, and Z. Cao, "New cost-effective VLSI implementation of a 2-D discrete cosine transform and its inverse," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 4, pp. 405–415, Apr. 2004.



Wenbin Xu (S'17) received the B.S. and M.S. degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2008 and 2011, respectively. He is currently working toward the Ph.D. degree in computer engineering at Texas A&M University, College Station, TX, USA.

He is currently a Research Assistant at the Department of Electrical and Computer Engineering, Texas A&M University. His current research interests include computer-aided design, approximate computing, and hardware security.

Sachin S. Sapatnekar (S'86–M'93–F'03) received the B.Tech. degree from IIT Bombay, Mumbai, India, the M.S. degree from Syracuse University, Syracuse, NY, USA, and the Ph.D. degree from the University of Illinois at Urbana–Champaign, Champaign, IL, USA.

He was with Iowa State University, Ames, IA, USA, from 1992 to 1997. He has been with the University of Minnesota, Minneapolis, MN, USA, since 1997, where he holds the Distinguished McKnight University Professorship and the Robert and Marjorie Henle Chair.

Dr. Sapatnekar is a Fellow of the Association for Computing Machinery (ACM). He was a recipient of seven conference best paper awards, the Best Poster Award, two ICCAD 10-year Retrospective Most Influential Paper Awards, the SRC Technical Excellence Award, and the SIA University Researcher Award.

Jiang Hu (F'16) received the B.S. degree in optical engineering from Zhejiang University, Hangzhou, China, in 1990 and the M.S. degree in physics and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1997 and 2001, respectively.

He was with IBM Microelectronics, Austin, Texas, from 2001 to 2002. In 2002, he joined the Faculty of Electrical Engineering, Texas A&M University, College Station, TX, USA. His current research interests include optimization for large-scale computing systems, especially very-large-scale integration circuit optimization, adaptive design, hardware security, resource allocation, and power management in computing systems.

Dr. Hu was a Technical Program Committee Member for DAC, ICCAD, ISPD, ISQED, ICCD, DATE, ISCAS, ASP-DAC, and ISLPED. He was a recipient of the Best Paper Award at the ACM/IEEE Design Automation Conference in 2001, the IBM Invention Achievement Award in 2003, and the Best Paper Award from the IEEE/ACM International Conference on Computer-Aided Design in 2011. He received a Humboldt Research Fellowship in 2012. He was a General Chair for the 2012 ACM International Symposium on Physical Design, and an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (2016–2011). He serves as an Associate Editor of the *ACM Transactions on Design Automation of Electronic Systems*.