On Reconfiguration-Oriented Approximate Adder Design and Its Application

Rong Ye[†], Ting Wang[†], Feng Yuan[†], Rakesh Kumar[§] and Qiang Xu[†]

[†]CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {rye, twang, fyuan, qxu}@cse.cuhk.edu.hk

[§]Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, USA
Email: rakeshk@illinois.edu

ABSTRACT

Approximate circuit designs allow us to tradeoff computation quality (e.g., accuracy) and computational effort (e.g., energy), by exploiting the inherent error-resilience of many applications. As the computation quality requirement of an application generally varies at runtime, it is preferable to be able to reconfigure approximate circuits to satisfy such needs and save unnecessary computational effort. In this paper, we present a reconfiguration-oriented design methodology for approximate circuits, and propose a reconfigurable approximate adder design that degrades computation quality gracefully. The proposed design methodology enables us to achieve better quality-effort tradeoff when compared to existing techniques, as demonstrated in the application of DCT computing.

1. INTRODUCTION

A large and growing number of applications are inherently error-tolerant, which do not require "strict" correctness but rather approximate correctness. Applications of such kind include multimedia, DSP, wireless communication, data mining and synthesis. They may process noisy data sets and the associated algorithms are stochastic or involve a human interface with limited perceptual capability. For these applications, *approximate computing*, being able to trade off computation quality (e.g., accuracy) and computational effort (e.g., energy), has attracted lots of attention recently [1–17].

As the computation quality requirement of an application may vary significantly at runtime, it is preferable to design quality-configurable systems (QCSs) that are able to tradeoff computation quality and computational effort on-the-fly according to application needs [15–17]. One of the key elements in a QCS would be the reconfigurable approximate hardware building blocks, which provide various quality-effort configuration points for the system to tune itself. Most existing approximate hardware designs (e.g., [10–14]), however, do not explicitly take such reconfiguration needs into consideration, which limits their applicability.

ICCAD 2013, November 18-21, 2013, San Jose, California, USA.
978-1-4799-1071-7/13/$31.00 ©2013 IEEE.

To tackle the above problem, in this work, we investigate how to design an approximate circuit that can better support system reconfiguration. To be specific, we explore the various configuration points of a QCS and evaluate its reconfiguration behavior according to a "quality-effort" curve that describes the relationship between the quality we can achieve and the effort we need to pay. Such "quality-effort" curve is able to guide us to develop more effective approximate circuits, i.e., we could achieve more benefits by conducting optimization towards a better "quality-effort" curve. As adders are the primary components for building many error-tolerant applications (e.g., DSP applications) and largely determine the performance and energy consumption in such systems, we propose a novel reconfiguration-oriented approximate adder design and apply it to a discrete cosine transform (DCT) computing platform. Experimental results demonstrate that our proposed approximate adder is able to provide much better quality-effort tradeoff when compared to existing designs.

The remainder of this paper is organized as follows. In Section 2, we present related work and motivate this work. The proposed accuracy-configurable approximate adder and its application in DCT computing platform are then detailed in Section 3 and Section 4, respectively. Next, Section 5 presents our experimental results. Finally, Section 6 concludes this paper and discusses potential work in the future.

2. RELATED WORK AND MOTIVATION

2.1 Related Work

There have been a large number of recent works in approximate computing (e.g., [1–17]), leveraging the inherent error-resilience of applications at various levels of design hierarchy.

Generally speaking, approximate hardware designs implement a slightly different yet more energy-efficient and/or faster Boolean function. There are a few attempts to systematically evaluate and/or generate approximate circuits. [7] proposed to construct an equivalent untimed circuit that represents the behavior of an approximate circuit and evaluate it by comparing to the original implementation. [8] investigated the behavior of approximate circuits under timing variations. [9] presented a systematic logic synthesis framework to generate functionally approximate circuits that satisfy a given quality constraint. In addition, various approximate designs for specific arithmetic components were presented in the literature, taking advantage of the structural properties of these components. In [10], a truncation-error-tolerant adder is used to

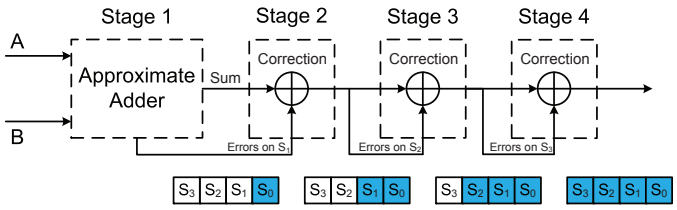


Figure 1: Accuracy-configurable adder in [17].

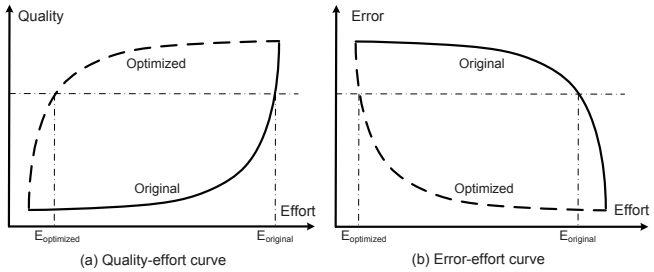


Figure 2: Quality-effort curve and its corresponding error-effort curve.

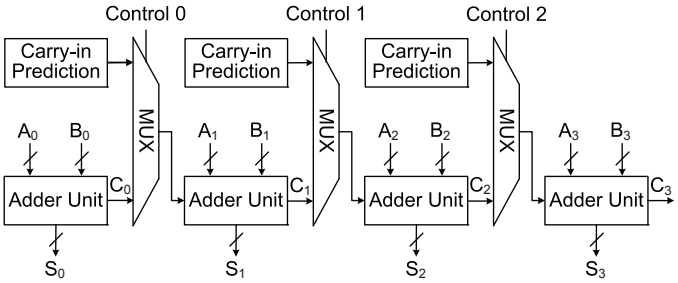


Figure 3: The proposed gracefully-degrading accuracy-configurable adder.

ease the restriction on computation accuracy and applied in DSP applications. In [11], simplified full adder cells are utilized to form approximate adders with reduced power. In [12], the authors try to find out a new optimal in a timing-starved adder by trading off error magnitude and error rate. In [13], a speculative carry select adder with error detection and recovery capability is proposed. In [14], a low-power approximate multiplier design is presented.

In practice, the computation quality requirement of an application may vary significantly under different circumstances. The above static approximate circuit designs with fixed approximation quality thus are likely to fail to meet application quality requirement or cause extra performance/energy overhead by providing unnecessarily high quality. Motivated by the above, the concept of scalable effort computing was studied in [15, 16], wherein reconfigurable systems are designed to achieve the maximum benefits under varying quality constraints.

One accuracy-configurable adder (denoted as *ACA adder*) is designed in [17] to adapt to varying accuracy requirements of different workloads. To be specific, ACA adder performs approximate addition first and then selectively turns on/off some of its correction stages to generate outputs with different levels of computation accuracy. As illustrated in Fig. 1, this example ACA adder has four stages. In the lowest accuracy mode, ACA adder shuts down all the correction stages (Stage 2, Stage 3 and Stage 4) and uses the output of approximate adder in Stage 1 directly as the final output, and only the least significant bits S_0 are guaranteed to be correct. If higher accuracy is required, ACA adder turns on some of the following correction stages. With all the stages turned on, ACA adder behaves as a fully accurate adder.

2.2 Motivation

The concept of scalable effort computing studied in prior works is interesting and promising, however, how to efficiently evaluate and optimize such a system has not been well studied. To guide QCS design, let us plot a "quality-effort" curve to represent the increase in computation quality with respect to computational effort such that we can traverse on the curve to achieve various quality-effort trade-off points and the corresponding error-effort curve, as shown in Fig. 2.

Consider the ACA adder presented in [17]. As discussed earlier, its correction scheme proceeds from the least significant bits (LSBs) to the most significant bits (MSBs). That means, if S_3 is required to be correct, we should first ensure all the LSBs (S_0, S_1, and S_2) are correct. Consequently, even if we only want a result with relatively high accuracy (correct S_3), we may have to turn on all the correction stages, which consumes much effort and dramatically weakens the benefit of accuracy reconfiguration. The "quality-effort" curve and "error-effort" curve of ACA adder thus follow the solid curve in Fig. 2 (denoted as "Original"). If, however, we can design an approximate adder with dotted "quality-effort" curve (denoted as "Optimized"), it is a much better QCS design. This is because, for the same quality, the optimized design consumes less effort than the original one. In other words, the optimized design degrades more gracefully with the decrease of computational effort.

The above observation motivates us to design a novel accuracy gracefully-degrading adder (denoted as GDA adder) in this paper. To the best of our knowledge, this is the first work that explicitly uses quality-effort curve to guide QCS designs.

3. ACCURACY GRACEFULLY-DEGRADING ADDER DESIGN

In this section, we describe the proposed GDA adder design. Two mechanisms are utilized in GDA adder to reconfigure computation accuracy, as detailed in the following subsections.

3.1 Reconfigurable Sub-Adder Bit-Length

As illustrated in Fig. 3, our proposed GDA adder consists of some basic adder units, and each unit is a k-bit adder that can be implemented using any design scheme.

Our first accuracy reconfiguration mechanism is to make the bit-length of sub-adders configurable. Let us consider an N-bit GDA adder. Given two N-bit addends A and B, we partition them into segments (e.g., $A = (A_3, A_2, A_1, A_0)$ and $B = (B_3, B_2, B_1, B_0)$) with k bits in each[1], and use adder units to compute the segmented addends respectively. Adder units are connected using multiplexers, which selects carry-in from either the less significant adder unit or a carry-in prediction component equipped with each unit. If all the multiplexers select the carry-ins from prediction components, the delay to execute such an N-bit addition, attributing to k-bit adder, prediction component and multiplexers, will be much smaller than the original delay of N-bit conventional adder, however with some accuracy loss. Consequently, by setting up the control

[1] A_0 and B_0 are LSBs, while A_3 and B_3 are MSBs

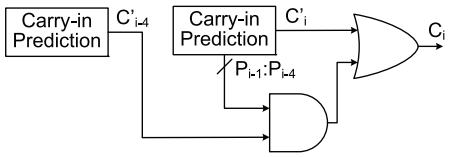


Figure 4: The proposed hierarchical prediction scheme.

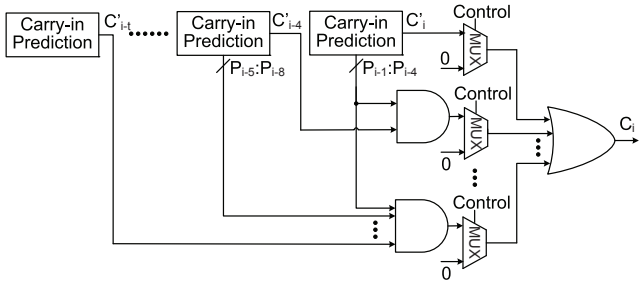


Figure 5: The proposed reconfigurable prediction scheme.

signals of multiplexers, we can determine which adder units are combined together as groups, operating as an individual sub-adder. In other words, the problem of reconfiguring GDA adder into a certain mode is just equivalent to the problem of combining adder units into certain sub-adders: the more effort we pay, the less accuracy loss we suffer from.

3.2 Reconfigurable Carry-in Prediction

The design of carry-in prediction logic is critical for determining computation accuracy of approximate adders, and the second mechanism for accuracy reconfiguration in GDA adder is to reconfigure the number of carry-in prediction bits.

The simplest design can be just a "0" or "1" signal. In fact, carry-in prediction can also be implemented using a complex but accurate mechanism. Although different types of prediction schemes are already proposed in prior works, their common idea is to generate carry-in based on the less significant bits of addends. Obviously, it is a tradeoff between the prediction accuracy and the hardware cost to build up such a prediction.

The two N-bit addends are represented by $A = (A_{N-1}, ..., A_i, ..., A_0)$ and $B = (B_{N-1}, ..., B_i, ..., B_0)$, where A_i and B_i are the i-th bit of A and B respectively. Then, we have the propagate P, the generate G, and the carry-in C as:

$$\begin{aligned} P_i &= A_i \oplus B_i \quad , \\ G_i &= A_i \cdot B_i \quad , \\ C_{i+1} &= G_i + P_i \cdot C_i \quad . \end{aligned} \tag{1}$$

Thus, we can develop the boolean equation that calculates carry-in C from less significant bits:

$$C_i = G_{i-1} + P_{i-1} \cdot G_{i-2} + \cdots + \prod_{j=i-1}^{i-t+1} P_j \cdot G_{i-t} + \prod_{j=i-1}^{i-t} P_j \cdot C_{i-t} \quad , \tag{2}$$

which indicates the basic principle of observing preceding t bits for carry-in prediction. If the carry-in C_{i-t} can be correctly given, the computation is accurate; otherwise, by assuming $C_{i-t} = 0$, we make a carry-in prediction for C_i.

By transforming Eq. 2 into different forms, we can have various implementations for the prediction component, e.g., t-bit adder proposed in [17] or carry-look-ahead (CLA) mechanism used by [1]. Actually, any design scheme of adder can be employed here to serve as carry-in prediction. The more bits we observe, the larger delay we have to suffer from. In this work, we propose a novel hierarchical prediction scheme that can observe more bits with less delay penalty, compared to a straightforward adder-like design.

Without loss of generality, let us simply assume 8 bits are required to be observed for enough accuracy. Based on Eq. 2 with $t = 8$ and $C_{i-t} = 0$, we have one of its equivalent transformations expressed by the following equations:

$$\begin{aligned} C_i &= C'_i + \prod_{j=i-1}^{i-4} P_j \cdot C'_{i-4} \quad , \\ C'_i &= G_{i-1} + P_{i-1} \cdot G_{i-2} + \cdots + \prod_{j=i-1}^{i-3} P_j \cdot G_{i-4} \quad , \\ C'_{i-4} &= G_{i-5} + P_{i-5} \cdot G_{i-6} + \cdots + \prod_{j=i-5}^{i-7} P_j \cdot G_{i-8} \quad . \end{aligned} \tag{3}$$

The above equations imply that C'_{i-4} will propagate to C_i only when

$$\prod_{j=i-1}^{i-4} P_j = P_{i-1} \cdot P_{i-2} \cdot P_{i-3} \cdot P_{i-4} = 1 \quad . \tag{4}$$

Consequently, in order to "look-ahead" 8 bits, we should observe two 4-bits separately and then combine them together with the case of Eq. 4 considered. As shown in Fig. 4, the two 4-bit prediction components are combined by control logic that detects the case of Eq. 4. Since the delay of this control logic is much smaller than that of 4-bit prediction component, the proposed prediction scheme has smaller delay than a conventional 8-bit prediction without any loss of prediction accuracy. More importantly, by combining more prediction components together in a similar way and adding some control logic (see Fig. 5), we can realize a reconfigurable prediction scheme, which controls how many prediction components are turned on and combined together, trading off power/delay and prediction accuracy.

4. DISCRETE COSINE TRANSFORM COMPUTING PLATFORM

Discrete cosine transform (DCT) serves as the basis in many international standards of static and dynamic images (e.g., JPEG, MPEG, etc.) [20]. By mapping input image into frequency domain, an image is expressed using a linear combination of weighted basis functions. The basis functions are frequency components and the weights are frequency coefficients. Because human visual system is more sensitive to changes in low frequency DCT components [21], DCT computation has inherent error-resilience for high-frequency components. For DCT coefficients of high frequency components, accuracy loss to a reasonable degree will not strongly degrade image quality. Consequently, a DCT computing platform that adapts its computation accuracy to accuracy requirements of different frequency components will be able to achieve benefits. In this section, we investigate how to apply our proposed accuracy gracefully-degrading adder in DCT computing platform and then justify its effectiveness in real application.

4.1 Accuracy Reconfigurable DCT Computing Platform

One-dimensional (1-D) DCT of an $n \times n$ data matrix X is defined as

$$Y = XC \quad , \tag{5}$$

where C is a normalized n-th order DCT matrix[2] and Y is the matrix of frequency coefficients. As shown in Fig. 6, the image is divided into 16 $n \times n$ blocks and then each block is transformed from image data to frequency coefficients.

A classic DCT computing platform that utilizes a memory look-up table to store some fixed data directly for fast computation is proposed in [19] and then developed by numerous following works (e.g., [18, 20]) to further reduce its hardware area. In this platform (see Fig. 7), data sequence $(x_{i,1}, x_{i,2}, \cdots, x_{i,n})$ is shifted sequentially with bit-parallel structure into Q shift registers, and then concurrently loaded into R shift registers. The data in R shift registers is concurrently shifted out through a butterfly stage in a bit-serial manner with LSB first. By feeding the bit patterns shifted out of R shift registers into ROM and Accumulator Components (RACs) (see Fig. 8), an output sequence $(y_{i,1}, y_{i,2}, \cdots, y_{i,n})$ is achieved. To realize an accuracy reconfigurable DCT computing platform, the adders inside RACs are replaced with the proposed gracefully-degrading adders, so that the DCT platform can adapt to the requirements of different frequency coefficients.

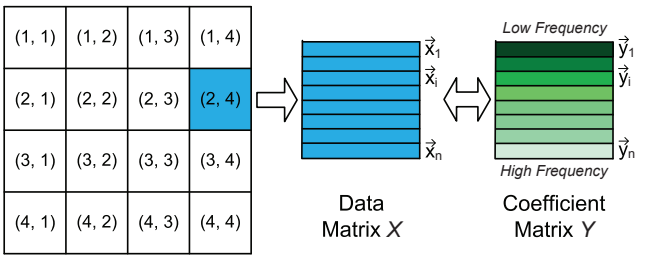


Figure 6: An example of image that is divided into 16 $n \times n$ blocks and its DCT process.

4.2 Rescheduling Computation Order

As discussed earlier, DCT has inherent error-resilience that allows low computation accuracy for high frequency components. Therefore, the elements of Y matrix $\{y_{ij}\}$ with larger i-values and j-values can tolerate accuracy loss with limited image quality degradation. Because each time DCT platform computes one row vector of Y:

$$\vec{y_i} = (y_{i1}, y_{i2}, \cdots, y_{in}) \quad , \tag{6}$$

we set up different accuracy requirements at different time points to explore this temporal variation of accuracy requirement. However, if we would like to utilize this accuracy requirement changed at every clock cycle, we, therefore, have to frequently reconfigure the computation capability accordingly to guarantee enough computation accuracy, which is not practical. To tackle this problem, we propose to reschedule the order of DCT computations, so that only infrequent accuracy mode changes is needed.

Conventionally, when applying DCT in image compression, the image is divided into blocks of size $n \times n$ and then individually processed by DCT. For each block, we will obtain an $n \times n$ matrix of DCT coefficients. As an example shown in Fig. 6, the image is divided into 16 blocks and DCT processes these blocks one by one from upper-left to bottom-right. Within each block, the data matrix X is processed one row vector after another to achieve one row vector of output matrix Y each time. In this work, we reschedule the order of DCT computations. Since the i-th row vector $\vec{x}_i$ of every block has the same accuracy requirement, we bundle the i-th row vectors $\vec{x}_i$ of all the blocks and process them in the same accuracy mode. After that, we turn to next bundle of row vectors $\vec{x}_{i+1}$ of all the blocks. By doing so, the computation with the same accuracy will last for a long time, because an image is usually divided into many $n \times n$ blocks, typically with $n = 8$. In that case, we only have 8 bundles of row vectors and hence the accuracy mode needs to be changed only 8 times for each image.

[2]Please refer to [18–20] for the details of DCT.

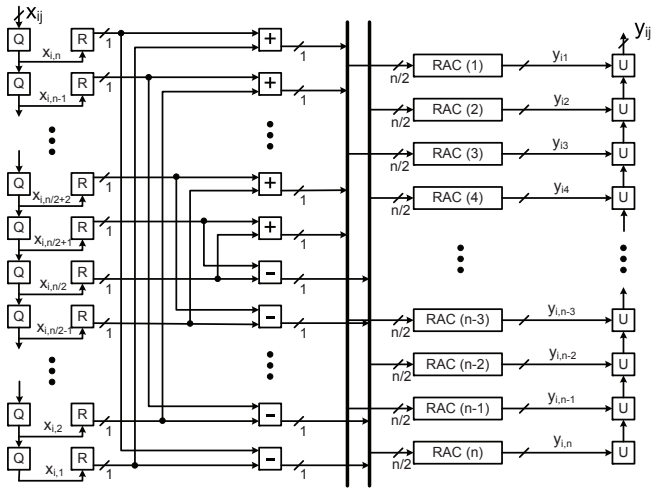


Figure 7: DCT computing platform in [19].

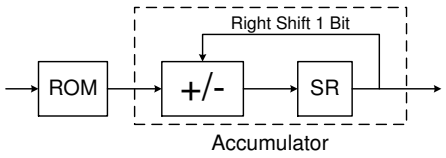


Figure 8: An example RAC design.

5. EXPERIMENTAL RESULTS

In this section, we present experimental results to demonstrate the effectiveness of the proposed GDA adder by comparing it against existing approximate adders.

5.1 Static Approximate Adder Comparison

We examine the static behavior of the proposed GDA adder in this subsection. That is, we implement the different configuration modes of GDA adder and ACA adder[3] as dedicated 32-bit static approximate adders and two approximate adders (Lu's adder [1] and $VLCSA$-1 [13]) for comparison. In addition, we also present results of accurate adders including ripple-carry adder (RCA) and carry-lookahead adder (CLA). The bit-lengths of sub-adders in ACA adder [17] and basic adder units in GDA adder are both set to be 4 bits. Consequently, ACA adder has 4 modes: $M_A = 1$ for only Stage 1 turned on, $M_A = 2$ for Stage 1 and Stage 2 turned on, and so forth. GDA adder has 16 modes: (i) $M_B = 1, 2, 3, 4$ for the maximum sub-adder bit-length to be 4, 8, 12 or 16 bits, respectively; (ii) $M_C = 1, 2, 3, 4$ for the number of carry-in prediction bits to be 4, 8, 12 or 16 bits, respectively. Gate-level simulations are performed with one million randomly-generated input patterns to these adders, and three error metrics are used to evaluate computation accuracy: WCE, ER and AE. Experimental results are shown in Table 1.

[3]Note that, the original ACA adder in [17] is a pipelined adder that requires multiple cycles to achieve higher accuracy. It is implemented as a combinational adder here to exclude the associated timing and area overhead for fair comparison.

	RCA	*CLA*	*VLCSA*-1	*Lu*
Area (μm^2)	1184	1736	2686	2321
Power (W)	2.407E-05	3.106E-05	5.054E-05	3.668E-05
Delay (ns)	7.88	5.40	1.24	0.83
WCE	0	0	269488128	2215641344
ER	0	0	16.6715%	32.2528%
AE	0	0	7906556	45247362

ACA				
M_A	1	2	3	4
Area (μm^2)	2200	2416	2632	2848
Power (W)	4.209E-05	4.692E-05	4.977E-05	5.259E-05
Delay (ns)	2.10	2.78	3.53	4.28
WCE	269488128	269484032	268435456	0
ER	16.7344%	11.6149%	6.0388%	0
AE	8352852	8352848	8351497	0

GDA				
(M_B, M_C)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
Area (μm^2)	1501	1597	1677	1741
Power (W)	3.940E-05	4.383E-05	4.623E-05	4.793E-05
Delay (ns)	1.44	1.78	2.11	2.44
WCE	**269488128**	268439552	268435456	268435456
ER	**16.7344%**	**0.8873%**	**0.0453%**	**0.0027%**
AE	**8352852**	509152	33690	2215
(M_B, M_C)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
Area (μm^2)	1577	1625	1689	1753
Power (W)	4.329E-05	4.567E-05	4.773E-05	4.952E-05
Delay (ns)	2.19	2.48	2.86	3.14
WCE	**16843008**	16777216	16777216	16777216
ER	8.8746%	0.3761%	0.0232%	0.0018%
AE	523985	**31614**	1954	67
(M_B, M_C)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
Area (μm^2)	1596	1628	1660	1708
Power (W)	4.437E-05	4.597E-05	4.703E-05	4.840E-05
Delay (ns)	2.93	3.22	3.51	3.85
WCE	**1048832**	1048576	1048576	1048576
ER	5.9984%	0.1922%	0.0122%	0.0016%
AE	32886	2015	**128**	17
(M_B, M_C)	(4, 1)	(4, 2)	(4, 3)	(4, 4)
Area (μm^2)	1615	1631	1663	1711
Power (W)	4.552E-05	4.631E-05	4.735E-05	4.871E-05
Delay (ns)	3.68	3.97	4.26	4.55
WCE	**65536**	65536	65536	**0**
ER	3.0964%	0.1884%	0.0116%	**0**
AE	2029	123	8	**0**

Table 1: Comparison on static adders.

First of all, let us compare adders that can achieve 100% accuracy, including RCA, CLA, ACA_4 and GDA_{44}[4]. It can be observed that, ACA_4 and GDA_{44} achieve smaller delays when compared to RCA and CLA, but are associated with higher hardware and power cost. In particular, ACA_4 has about 5.93% smaller delay than GDA_{44}, but consumes 66.45% more hardware area and 7.97% more power.

Next, let us compare adders with approximated accuracy. From Table 1, $VLCSA$-1 has similar computation accuracy as ACA_1 and GDA_{11}, and it is associated with smaller delay but much larger hardware area. As for Lu's adder, it has the smallest delay, but, unfortunately, much larger errors and relatively larger hardware area. In approximate modes, ACA has almost constant WCE and AE, while GDA has exponentially decreasing WCE with respect to M_B, exponentially decreasing ER with respect to M_C, and exponentially decreasing AE with respect to both M_B and M_C. Such interesting observation is also presented in Fig. 9, justifying that the two mechanisms to reconfigure accuracy of GDA adder are suitable for different types of error metrics and hence complement each other.

None of the above approximate adders outperforms the others in all aspects. Consequently, a static approximate adder design cannot provide the best quality-effort for different workloads, which justifies the necessity of quality-configurable approximate adder designs.

[4] ACA_i means ACA adder with $M_A = i$; GDA_{ij} means GDA adder with $M_B = i$ and $M_C = j$.

ACA				
Area (μm^2)	3119			
M_A	1	2	3	4
Power (W)	4.261E-05	4.882E-05	5.289E-05	5.696E-05
Delay (ns)	2.15	2.90	3.67	4.45
Delay* (ns)	1.64	2.49	3.39	4.42

GDA				
Area (μm^2)	2513			
(M_B, M_C)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
Power (W)	4.679E-05	4.777E-05	5.327E-05	5.733E-05
Delay (ns)	1.58	2.01	2.41	2.82
Delay* (ns)	1.31	1.69	2.24	2.82
(M_B, M_C)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
Power (W)	4.590E-05	4.900E-05	5.065E-05	5.245E-05
Delay (ns)	2.49	2.87	2.96	3.11
Delay* (ns)	2.02	2.47	2.63	2.85
(M_B, M_C)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
Power (W)	4.742E-05	4.953E-05	5.031E-05	5.120E-05
Delay (ns)	3.41	3.78	3.88	4.03
Delay* (ns)	2.85	3.29	3.42	3.61
(M_B, M_C)	(4, 1)	(4, 2)	(4, 3)	(4, 4)
Power (W)	4.879E-05	4.981E-05	5.057E-05	5.149E-05
Delay (ns)	4.33	4.70	4.80	4.95
Delay* (ns)	3.71	4.11	4.25	4.46

Delay/Delay*: the critical path delay of each mode before/after performing voltage scaling to ensure the same power consumption.

Table 2: Accuracy-configurable implementation of ACA and GDA adders.

5.2 Quality-Configurable Adder Comparison

In this subsection, we conduct detailed comparison for reconfigurable approximate adders ACA and GDA, which can be forced into a certain accuracy mode by setting up proper control signals.

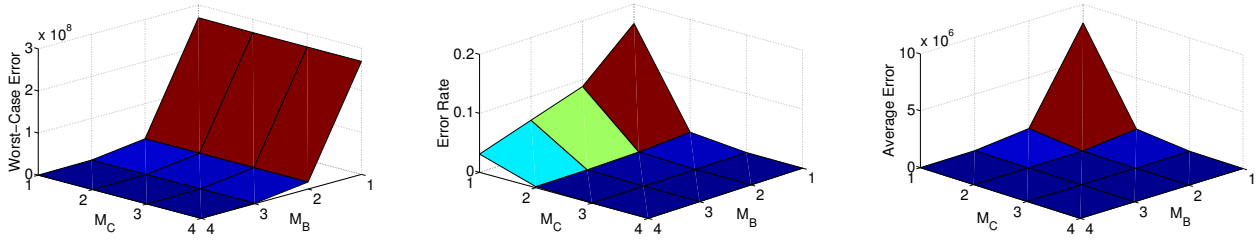
Table 2 reports the hardware areas, power consumptions, and delays of these reconfigurable adders. "Delay*" in the table is a normalized delay value by performing voltage scaling to all the modes of ACA and GDA adders to have the same power consumption as the largest one (i.e., GDA_{14}). Note that, the error metrics of different configuration modes are the same as the static implementation (see Table 1) and hence are not shown in this table. We find that GDA adder costs 19.43% less hardware area than ACA adder and has only 0.90% more delay than ACA adder in accurate mode.

Based on the observations in Fig. 9, we simply choose (1,1), (2,1), (3,1), (4,1) and (4,4) for WCE; (1,1), (1,2), (1,3), (1,4) and (4,4) for ER; and (1,1), (2,2), (3,3) and (4,4) for AE, as marked out as bold in Table 1. After that, we plot out these modes and also ACA adder's modes in Fig. 10. As expected, the error-delay curves of GDA adder present much better characteristics when compared to ACA adder. Note that, the above chosen configuration points do not guarantee serving as optimal candidates for reconfiguration, and a careful selection may provide better performance for a particular application.

Finally, we conduct experiments on DCT computing platform to verify the efficacy of accuracy-configurable adders. We simulate the logic functionality of DCT platform using Matlab. The matrix dimension n for DCT is specified to be 8, since this is a typical value used in most applications (e.g., JPEG). The quantization matrix used in experiments is the same with JPEG. Similar to [17], peak signal to noise ratio (PSNR) is used as the evaluation metric on image quality. Higher PSNR means better image quality.

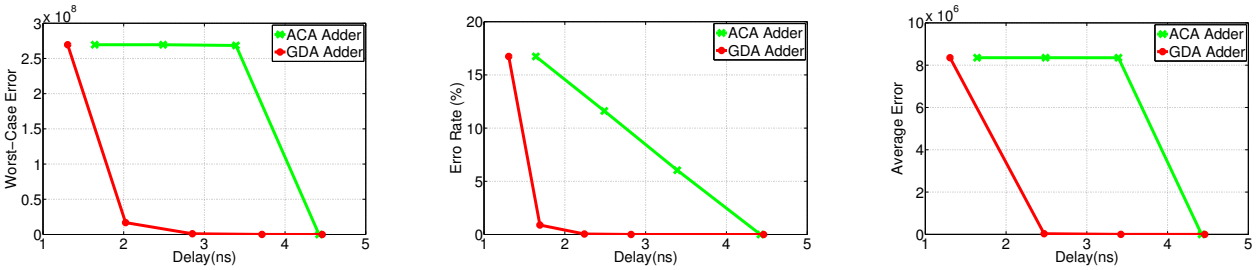
We replace the adders of DCT platform with RCA, CLA, ACA and GDA adders, and compare their performance[5]. As can be seen from Table 3, RCA, CLA and ACA_4 are fully accurate adders with the highest PSNR and among them CLA has the highest throughput. GDA_{41} has higher throughput

[5] The delays of RCA and CLA adders are also normalized to have the same power with others.



(a) Worst-case error versus M_B and M_C. (b) Error rate versus M_B and M_C. (c) Average error versus M_B and M_C.

Figure 9: Error metrics versus M_B and M_C on GDA adder.



(a) Worst-case error versus delay. (b) Error rate versus delay. (c) Average error versus delay.

Figure 10: Error metrics versus delay on ACA and GDA adders.

	RCA	CLA	ACA					GDA				
			ACA_1	ACA_2	ACA_3	ACA_4	ACA*	GDA_{11}	GDA_{21}	GDA_{31}	GDA_{41}	GDA*
Th. (MHz)	249.27	308.00	630.21	434.22	324.41	254.20	334.45	969.10	590.45	432.29	336.76	404.86
PSNR (dB)	26.58	26.58	12.41	12.59	16.87	26.58	22.76	14.32	16.69	22.86	26.57	25.83

Th.: equivalent throughput; PSNR: peak signal to noise ratio; ACA* and GDA*: ACA and GDA in a reconfigurable manner.

Table 3: Results on DCT computing platform.

than CLA with nearly no image quality degradation. When using four different modes of each adder to compute DCT coefficients separately, as expected, we can achieve higher PSNR but lower throughput with high-accuracy modes.

Because image quality is relatively insensitive to high-frequency DCT coefficients with larger index, we use a simple heuristic to utilize this feature and reconfigure the DCT platform as follows: (i) for ACA, the four modes ACA_1, ACA_2, ACA_3 and ACA_4 are used to compute $(\vec{y}_7, \vec{y}_8)$, $(\vec{y}_5, \vec{y}_6)$, $(\vec{y}_3, \vec{y}_4)$, and $(\vec{y}_1, \vec{y}_2)$, respectively; (ii) similarly, for GDA, the four modes GDA_{11}, GDA_{21}, GDA_{31} and GDA_{41} are used to compute $(\vec{y}_7, \vec{y}_8)$, $(\vec{y}_5, \vec{y}_6)$, $(\vec{y}_3, \vec{y}_4)$, and $(\vec{y}_1, \vec{y}_2)$, respectively. These two cases are denoted as ACA* and GDA*. As shown in Table 3, ACA* and GDA* achieve much higher throughput than RCA, CLA, and their high-accuracy modes without much PSNR loss. Such results demonstrate the effectiveness of using quality-configurable adders for DCT computing. When compared to ACA*, GDA* has even higher throughput and PSNR, which proves the efficacy of the proposed GDA adder.

As shown in Fig. 11, the image generated by GDA* (see Fig. 11(d)) has notablely better quality than the image generated by ACA* (see Fig. 11(c)), and its quality is very close to the image quality achieved with accurate adder (see Fig. 11(b)).

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a reconfiguration-oriented approximate adder design. By explicitly optimizing the design towards a better "quality-effort" curve, our adder is shown to be much more effective than existing designs. In particular, experimental results demonstrate that we can simultaneously achieve much better throughput and image quality when applying our adder to a DCT application.

This work highlights the importance of "quality-effort" curve in reconfiguration-oriented approximate circuit designs. Simple strategies to perform system reconfiguration are used in our case studies, since they are not the focus of this work. How to further improve QCS designs with more elegant reconfiguration policy is an interesting avenue for future exploration.

7. ACKNOWLEDGEMENT

This work was supported in part by the Hong Kong SAR Research Grants Council (RGC) under General Research Fund No. CUHK418111 and No. CUHK418812, in part by NSFC/RGC Joint Research Scheme No. N_CUHK444/12, and in part by SRC project 2010-HJ-2081.

8. REFERENCES

[1] S. L. Lu, Speeding up processing with approximation circuits. In *IEEE Computer*, vol.37, no.3, pp.67-73, Mar 2004.

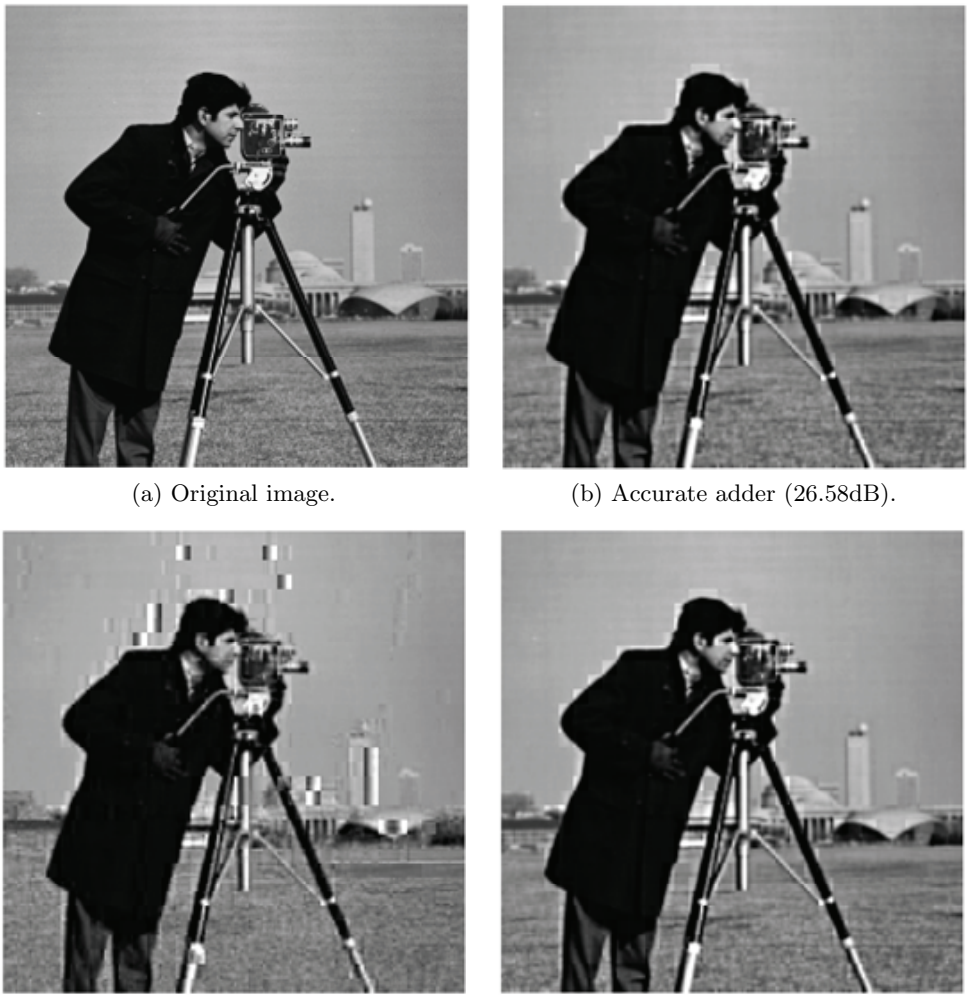
[2] M. Breuer, Hardware that produces bounded rather than exact results. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp.871-876, 2010.

[3] S. T. Chakradhar, A. Raghunathan, Best-effort computing: re-thinking parallel software and hardware. In *Proc. of IEEE/ACM Design Automation Conference (DAC)*, pp.865-870, 2010.

[4] S. Narayanan, J. Sartori, R. Kumar, D. Jones, Scalable Stochastic Processors. In *Proc. IEEE/ACM Design, Automation and Test in Europe (DATE)*, pp.335-338, 2010.

[5] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, M. Rinard, Dynamic knobs for responsive power-aware computing. In *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp.199-212, 2011.

[6] L. Leem, C. Hyungmin, J. Bau, Q. A. Jacobson, S. Mitra, ERSA: Error Resilient System Architecture for probabilistic applications. In *Proc. IEEE/ACM Design, Automation & Test in Europe (DATE)*, pp.546-558, 2010.



(a) Original image. (b) Accurate adder (26.58dB).

(c) ACA^* adder (22.76dB). (d) GDA^* adder (25.83dB).

Figure 11: Image quality comparison.

[7] R. Venkatesan, A. Agarwal, K. Roy, A. Raghunathan, MACACO: modeling and analysis of circuits for approximate computing. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.667-673, 2011.

[8] M. Dehbashi, G. Fey, K. Roy, A. Raghunathan, On Modeling and Evaluation of Logic Circuits Under Timing Variations. In *Proc. Euromicro Conference on Digital System Design*, pp.431-436, 2012.

[9] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, A. Raghunathan, SALSA: systematic logic synthesis of approximate circuits. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp.796-801, 2012.

[10] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, Z. H. Kong, Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.18, no.8, pp.1225-1229, Aug. 2010.

[11] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, K. Roy, IMPACT: imprecise adders for low-power approximate computing. In *Proc. IEEE/ACM International Symposium on Low-Power Electronics and Design (ISLPED)*, pp.409-414, 2011.

[12] J. Miao, K. He, A. Gerstlauer, M. Orshansky, Modeling and Synthesis of Quality-Energy Optimal for Approximate Adder. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp.728-735, 2012.

[13] K. Du, P. Varman, K. Mohanram, High performance reliable variable latency carry select addition. In *Proc. IEEE/ACM Design, Automation Test in Europe (DATE)*, pp.1257-1262, 2012.

[14] P. Kulkarni, P. Gupta, M. Ercegovac, Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In *Proc. IEEE International Conference on VLSI Design*, pp.346-351, Jan. 2011.

[15] V. Chippa, A. Raghunathan, K. Roy, S. Chakradhar, Dynamic effort scaling: Managing the quality-efficiency tradeoff. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp.603-608, June 2011.

[16] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, S. T. Chakradhar, Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp.555-560, 2010.

[17] A. B. Kahng, S. Kang, Accuracy-configurable adder for approximate arithmetic designs. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp.820-825, 2012.

[18] S. Yu; E. E. Swartziander Jr., DCT implementation with distributed arithmetic. In *IEEE Transactions on Computers*, vol.50, no.9, pp.985-991, Sep 2001.

[19] M. T. Sun, T. C. Chen, A. M. Gottlieb, VLSI implementation of a 16×16 discrete cosine transform. In *IEEE Transactions on Circuits and Systems*, vol.36, no.4, pp.610-617, Apr 1989.

[20] D. Gong, Y. He, Z. Gao, New cost-effective VLSI implementation of a 2-D Discrete Cosine Transform and Its Inverse. In *IEEE Transactions on Circuits and Systems for Video Technology*, vol.14, pp.405-415, 2004.

[21] A. B. Watson, DCT quantization matrices visually optimized for individual images. In *Proc. SPIE Conf. Human Vision, Visual Processing and Digital Display IV*, pp.202-216, 1993.

[22] A. B. Kahng, S. Kang, R. Kumar, J. Sartori, Designing a processor from the ground up to allow voltage/reliability tradeoffs. In *Proc. IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp.1-11, Jan. 2010.