Accuracy-Configurable Adder for Approximate Arithmetic Designs

Andrew B. Kahng[†‡] and Seokhyeong Kang[†]
[†]ECE and [‡]CSE Departments, University of California at San Diego
abk@cs.ucsd.edu, shkang@vlsicad.ucsd.edu

ABSTRACT

Approximation can increase performance or reduce power consumption with a simplified or inaccurate circuit in application contexts where strict requirements are relaxed. For applications related to human senses, approximate arithmetic can be used to generate sufficient results rather than absolutely accurate results. Approximate design exploits a tradeoff of accuracy in computation versus performance and power. However, required accuracy varies according to applications, and 100% accurate results are still required in some situations. In this paper, we propose an *accuracy-configurable approximate (ACA) adder* for which the accuracy of results is configurable during runtime. Because of its configurability, the ACA adder can adaptively operate in both approximate (inaccurate) mode and accurate mode. The proposed adder can achieve significant throughput improvement and total power reduction over conventional adder designs. It can be used in accuracy-configurable applications, and improves the achievable tradeoff between performance/power and quality. The ACA adder achieves approximately 30% power reduction versus the conventional pipelined adder at the relaxed accuracy requirement.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS—*Design Aids*; J.6 [**Computer Applications**]: COMPUTER-AIDED ENGINEERING

General Terms

Algorithms, Design, Performance

Keywords

Approximate Arithmetic, Error-Tolerance, Power Minimization, Accuracy-Configurable Adder

1. INTRODUCTION

Guardbands for dynamic variations severely limit performance and energy efficiency of conventional IC designs. To overcome consequences of overdesign, several recent mechanisms for variation-resilient design [4] allow timing errors and manage design reliability dynamically. Relaxing the requirement of correctness for designs may dramatically reduce costs of manufacturing, verification and test [16]. In resilient designs, errors can be corrected with redundancy techniques (*error-tolerance*), or accepted in some applications relating to human senses such as hearing and sight (*error-acceptance*). In the error-acceptance regime, approximation via a simplified or inaccurate circuit can increase performance and/or reduce power consumption.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2012, June 3-7, 2012, San Francisco, California, USA.
Copyright 2012 ACM 978-1-4503-1199-1/12/06 ...$10.00.

Various approximate arithmetic designs have been previously proposed. Lu [7] introduces a faster adder which has shorter carry chains and considers only the previous *k* bits of input in computing a carry bit. Verma et al. [12] provide a variable latency speculative adder ($VLSA$), which is a reliable version of the Lu adder [7] with error detection and correction. Shin et al. [10] also propose a data path redesign technique for various adders which cuts the critical path in the carry chain. Zhu et al. [14] [13] propose three approximate adders – $ETAI$, $ETAII$ and $ETAIIM$. ETAI is divided into an accurate part and an inaccurate part to achieve approximate results. ETAII cuts carry propagation to speed up the adder, and ETAIIM modifies ETAII by connecting carry chains in accurate MSB parts. Kulkarni et al. [5] present a 2x2 underdesigned multiplier, and use it to build large power-efficient approximate multipliers. George et al. [3] define the concept of *probabilistic CMOS (PCMOS)*, and implement efficient arithmetic using $PCMOS$. Shin et al. [11] propose a logic synthesis approach to design an approximate circuit.

The approximate designs produce almost-correct results at the given required accuracy, and obtain power reductions or performance improvements in return. In some applications, however, more accurate or totally accurate results are required *under certain conditions* – e.g., image processing in security cameras would require cleaner images after detecting a motion. In contexts where the required accuracy changes during runtime, the accuracy of results should be configurable to maximize the benefit of approximate operations. Figure 1 illustrates how power benefits can be achieved with an accuracy-configurable design. The accuracy-configurable design can adapt to changing accuracy constraints by using different modes in each situation. To our knowledge, no previous work can configure the output accuracy during runtime, and each is thus restricted (or, best-suited) to particular application contexts. In contexts where the accuracy requirement can change dynamically, the previous methods' benefits from the accuracy tradeoff are reduced since the implementation must be targeted to the maximum accuracy requirement.

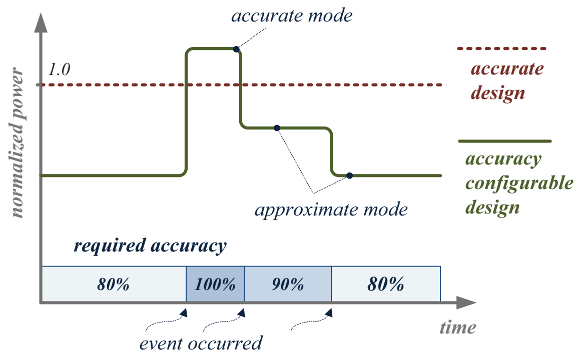

Figure 1: Power benefits from accuracy-configurable design.

In this paper, we propose an *accuracy-configurable approximate* (ACA) adder, which can configure the accuracy of results during runtime. The main contributions of our work are the following.

- The proposed ACA adder has runtime-configurable accuracy to better enable tradeoff of accuracy in computation versus performance and power.
- We provide quantitative metrics for an approximate arithmetic design. We compare the ACA adder to previous approximate adders based on these metrics.
- We demonstrate the power benefits of the ACA adder over previous approximate and conventional adder designs for accuracy-configurable applications.

The rest of the paper is organized as follows. Section 2 presents the proposed ACA adder design. Section 3 provides experimental results and analysis. Section 4 summarizes and concludes the paper.

2. ACCURACY-CONFIGURABLE ADDER

2.1 Approximate Adder Implementation

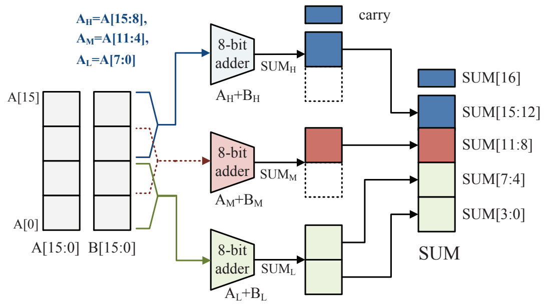

Figure 2: Proposed approximate adder – 16-bit adder case.

Previous approximate adders [7] [10] [14] have difficulty detecting and correcting errors since they are designed for error-acceptable applications with a target accuracy. However, accurate computations are still required at certain times, according to the application. VLSA [12] can provide accurate results, but has large delay and area overhead for the error detection and correction. The central contribution of our present work is to propose an approximate adder which supports both accurate and inaccurate computation with error-correction and accuracy-configuration capability. Figure 2 shows our proposed approximate circuit for the case of a 16-bit adder. In the adder, the carry chain is cut to reduce critical-path delay, and three sub-adders generate results of partial summations. With the reduced critical-path delay, high performance (by increasing the clock frequency) or low power consumption (by decreasing the operating voltage) is obtained. A middle sub-adder (A_M+B_M) is introduced to increase accuracy. Without the middle sub-adder (as in ETAII [13]), error occurs when the eighth carry bit is high, and for random input patterns the error rate is 50.1%. On the other hand, with the introduction of the middle sub-adder, error rate for random input patterns is reduced to 5.5%. (In the real implementation, all redundant parts (four-LSB output of $A_H + B_H$ and $A_M + B_M$ sub-adders) are optimized only for carry-generation.)

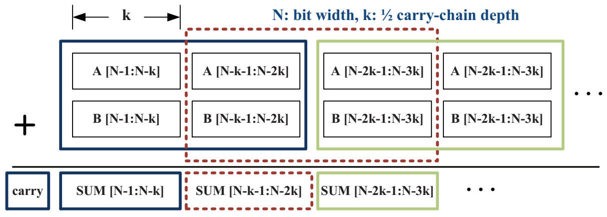

Figure 3: General implementation for the proposed adder.

We can generalize the implementation of the proposed approximate adder. Figure 3 shows the general implementation of an N-bit adder with a parameter k, which is the bit-width of the sub-adder result. In the adder, each divided sub-module produces a k-bit result except for the last sub-module, which produces a $2k$-bit result. The approximate adder thus consists of the $(N/k-1)$ sub-modules as described in Equation (1).

$$\begin{aligned} SUM[N-ik-1:N-(i+1)k] = \\ A[N-ik-1:N-(i+2)k] + \\ B[N-ik-1:N-(i+2)k], \\ where\ i = 0, ..., N/k-2 \end{aligned} \quad (1)$$

In modern adder designs, such as carry-lookahead (CLA), carry-select and Kogge-Stone adders, the path depth and area are asymptotically proportional to log_2N and $Nlog_2N$ respectively, where N is the bit-width of the adder [15]. Based on this, we can express delay, area and power consumption of the proposed adder in terms of the parameters N and k. The proposed ACA adder has $(N/k-1)$ sub-adders, each of which is a $2k$-bit adder. Therefore, delay of the critical path can be expressed with Equation (2) and area can be estimated with Equation (3), where C_{delay} and C_{area} are constants for delay and area, respectively.

$$delay = C_{delay}(log_2k + 1) \quad (2)$$

$$area = C_{area}(N-2k)(log_2k+1) \quad (3)$$

$$Power_{dyn} = C_{power}(N-2k)(log_2k+1)^2 \quad (4)$$

Power consumption of the ACA adder can be roughly estimated as follows. Dynamic power consumption with voltage scaling at a fixed frequency is proportional to $capacitance \cdot V_{dd}{}^2$, where the $capacitance$ is proportional to the area. Cell delay is proportional to $1/(V_{dd}-V_t)^\beta$, and V_{dd}^2 is roughly proportional to $1/(cell\ delay)$ if we assume that β is 2. Since $(cell\ delay) \times (path\ depth)$ is constant at a fixed frequency, V_{dd}^2 is proportional to the path depth, which is log_2k+1. Consequently, dynamic power with voltage scaling can be expressed using Equation (4), where C_{power} is a constant fixed for given V_{dd} for dynamic power consumption. Static power consumption of the adder can be roughly estimated as proportional to the area in Equation (3).

In our proposed adder design, the output of each sub-adder (except the last sub-adder) is incorrect when a carry input should be propagated to the results. In Figure 2, when the $carry[4]$ (carry bit from A_L+B_L) is '1' and $SUM_M[3:0]$ is $1111_{(2)}$, the output result has an error in $SUM[11:8]$. In the general implementation, the output result will be correct when there are no errors in all $(N/k-1)$ sub-adders. In the i^{th} sub-adder, errors occur when (1) the LSB part of the result $(SUM_i[k-1:0])$ has all '1' values (probability $P = \frac{1}{2^k}$) and (2) the LSB part $([k-1:0])$ of the $(i+1)^{th}$ sub-adder produces a carry bit (probability $P = \frac{1}{4} + \frac{1}{2}\cdot\frac{1}{4} + \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{4} + ...$). Therefore, with a random input vector, the probability of having a correct result in the proposed adder is

$$P(N,k) = (1 - \frac{1}{2^k}\cdot\frac{2^k-1}{2^{k+1}})^{\frac{N}{k}-2} \quad (5)$$

Table 1 shows the estimated results of 16-bit ACA adders with different parameter values k. With smaller k value, the minimum clock period and dynamic power can be reduced, but the pass rate (probability of having a correct result) will be decreased. The estimations come from Equations (2), (3), (4) and (5). In Section 3.3 below, we validate the above estimation with real implementations.

Table 1: Estimated minimum clock cycle, area, dynamic power and pass rate for each k value when N = 16 (normalized to the conventional CLA 16-bit adder).

	k=2	k=3	k=4	k=5	k=6
min. clock period	0.5	0.65	0.75	0.83	0.89
area	0.87	1.05	1.12	1.15	1.12
dynamic power	0.44	0.68	0.84	0.95	1.00
pass rate	0.554	0.829	0.942	0.982	0.995

2.2 Error Detection and Correction for Accurate Computation

As described in Section 2.1, our proposed adder is incorrect when a carry bit is propagated between sub-adders. However, the error can be detected and corrected with a small overhead. We detect an error for each sub-adder by checking the output of the sub-adder and the carry-in signal that comes from the previous sub-adder. Error detection can be implemented with several 'and' gates. To correct the error, '1' should be added to the approximate (inaccurate) output, and the error correction can be implemented with an incrementor circuit.

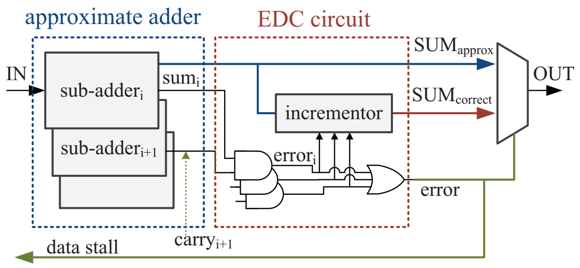

Figure 4: Error detection and correction with the approximate adder.

With these simple error detection and correction circuits, our proposed adder can be implemented to have variable latency like the previous VLSA adder [12], with a small overhead for an error detection and correction (EDC) system. Figure 4 shows an EDC system with our proposed adder. The error detection circuit ('and' gates) checks the carry propagation and generates an error signal. The error correction (incrementor) circuit produces an error-free output by adding compensation data, and requires an additional clock cycle. When errors are detected from input patterns, the $error$ signal is activated. The error signal holds the input pattern during the error correction and chooses the error-corrected value ($SUM_{correct}$) as an output. With this approach, our approximate adder can provide accurate results at a higher clock frequency than that of conventional adders (e.g., CLA). According to the estimated results in Table 1, clock period can be reduced by 25% with 6% (= error rate) recovery-cycle overhead (16-bit ACA, $k = 4$).

2.3 Accuracy Configuration with Pipelined Architecture

When our proposed adder is combined with a pipelined architecture, we can obtain accurate results with the same throughput as a conventional adder. In the pipelined architecture, approximate additions are computed at the first pipeline stage, and error correction can be completed at the second stage. Figure 5 shows the conventional pipelined adder (above) and the approximate adder (below). The pipelined implementation of approximate adder has a structural analogy with the pipelined adder of the 2006 U.S. patent [8] in which partial summations are performed at the first stage and carry bits are added at the later stages. However, the patent is clearly directed to accurate operations, not approximate computations. In addition, we use our approximate adder (Figure 3) in the first stage. In the pipelined approach, there is no improvement of the clock frequency since the achievable clock period is the same as that of the conventional adder. However, power benefits are obtained through configuration of accuracy: in the approximate mode, the error correction stage is power-gated with foot (or, head) switches in Figure 5, and power reduction over the conventional adder design can be achieved. We compare the conventional and approximate pipelined adders in Section 3.

In the proposed adder implementation, to achieve higher performance or lower power consumption, we can reduce the carry chain depth (k) of sub-adders (see Table 1). However, when k is less than $N/4$, it is impossible to correct all errors and achieve 100% correct results within one clock cycle since the error-correction paths become critical. To achieve correct results in the pipelined implementation, the error-correction stage should be extended to multiple stages. Figure 6 shows the pipelined adder implementation ($k = N/8$ case), in which four pipeline stages are required to achieve a 100% accurate result. In the pipelined adder, each stage generates a result with different accuracy; the output accuracy increases as the number of pipeline stages increases. According to the accuracy requirement, we can turn off the later stages with a power gating technique, and we can reduce the power consumption further with the accuracy tradeoff.

Since the proposed adder supports both approximate and accurate results, it can be used in applications that require accurate results only under certain conditions. Conventional accurate designs are energy-inefficient in the error-acceptable application context, because they always compute the exact function. Previous approximate designs cannot handle a varying accuracy requirement, and this limits the benefit of the accuracy tradeoff: as noted above, the approximate function must meet the maximum accuracy threshold across all applications. Moreover, if the application requests an exact computation, additional accurate circuits must be added to the previous approximate designs. By contrast, the ACA design efficiently exploits a tradeoff between accuracy and power/performance with its runtime accuracy configurability.

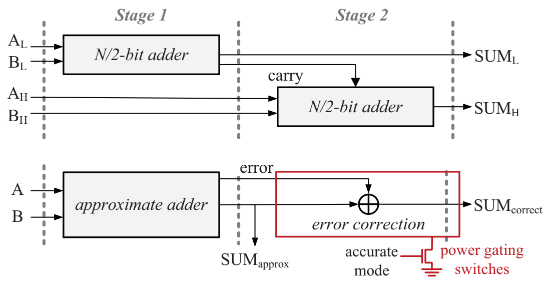

Figure 5: Pipelined adder implementation – conventional adder (above) and approximate adder (below). In approximate operation, the error correction stage is power-gated.

3. EXPERIMENTAL SETUP AND RESULTS

3.1 Experimental Setup

To test approximate designs, we have written each design in Verilog and synthesized it to a TSMC 65GP cell library with *Synopsys DesignCompiler* [17]. We then perform gate-level simulations using *Cadence NC-Sim* [18]. In the simulation, gate delay is taken from an SDF (standard delay format) file. For voltage scaling experiments, we prepare *Synopsys Liberty* (.lib) files for each voltage from 1.00V to 0.60V in 0.01V increments, using *Cadence Library Characterizer v9.1* [19]. The prepared libraries are used for SDF file generation and power estimation at each voltage. Each simulation is performed with input patterns for one million cycles. During the simulation, each output value is compared with a reference (correct) value to produce the accuracy metrics. For the input patterns, we use random data, as well as actual data from *SPEC 2006* [20] benchmarks. We extract operand data from ADD instructions in the SPEC benchmarks.

3.2 Metric for Approximate Design

To quantify errors in approximate designs, two metrics have been previously proposed [1]. *Error rate* (ER) is the percentage of cycles in which output value is different from the correct value. *Error significance* (ES) is the numerical difference between correct and output results; this quantifies the amount of error. In image/video applications, [2] uses the product of ES and ER as a metric of error tolerance. [10] introduces a criterion for acceptability: $ES \times ER \leq$ *acceptance threshold*, where the acceptance threshold is specified according to the application. For the error significance (ES) metric, [14] considers only amplitude of error. This is useful for many digital signal processing (DSP) systems that process, e.g., sound and image data. However, in communication systems that mainly handle information data, the number of incorrect bits

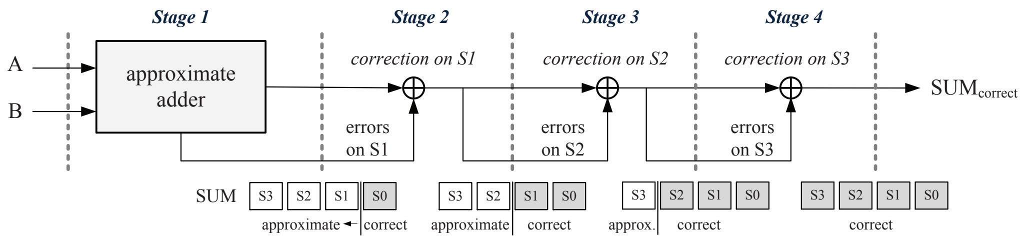

Figure 6: Accuracy-configurable implementation for pipelined adder.

(Hamming distance) is a more meaningful metric for accuracy – e.g. a *(32,28) Reed-Solomon code* can correct up to 2-byte errors. This consideration for the ES metric is required when approximate arithmetic is applied to error-tolerant systems with a redundancy technique.

Table 2 shows two accuracy metrics for amplitude data and information data. ACC_{amp} used in [14] quantifies the amplitude of errors, where R_c and R_e are the correct and obtained results, respectively. We propose another accuracy metric, ACC_{inf}, which measures error significance as Hamming distance, where B_e is the number of error bits and B_w is the bit-width of the data. For example, when the correct (reference) data is $1000\_0000_{(2)}$ and the result data is $1100\_0000_{(2)}$, accuracy with ACC_{amp} and ACC_{inf} will be $\frac{1}{2}$ and $\frac{7}{8}$, respectively. To evaluate the approximate circuits, we obtain average values of accuracy metrics ACC_{amp} and ACC_{inf} over the entire simulation to consider both ER and ES.

Table 2: Accuracy metrics for error significance (ES).

metric	definition	data type
ACC_{amp}	$1 - \lvert R_c - R_e \rvert / R_c$	amplitude data
ACC_{inf}	$1 - B_e/B_w$	information data

Table 3: ACA adder results with different k values.

k	2	3	4	5
min. clock period (ps)	180	190	220	230
area (um^2)	550	990	920	840
pass rate (%)	55.3	82.8	94.0	98.1
throughput improvement (%)	11.3	24.6	22.3	21.4

Table 4: Design comparison for each adder design.

	CLA	LU	ACA	ETAI	ETAIIM
area (um^2)	910	1356	923	576	678
min. clock period (ps)	280	210	200	200	260
pass rate (%)	100	99.2	94.1	10.0	97.0
ACC_{amp} (maximum)	1.000	0.998	0.997	0.999	0.999
ACC_{inf} (maximum)	1.000	0.999	0.993	0.694	0.996
area overhead for EDC	N/A	75%	28%	N/A	15%

3.3 Approximate Adder with Different Parameters

We explore the proposed adder with different parameters (k: half of carry-chain depth). Table 3 summarizes results – minimum clock period, area, error rate and throughput improvements – for each implementation of the 16-bit adder with different k values. According to the results, with smaller k, the maximum operating frequency increases, but the error rate increases as well. With higher k, the error rate is reduced significantly, but the benefit of the approximate circuit, i.e., clock period reduction, is small. In the table, throughput improvement over conventional design is calculated including error recovery overhead. From the implementations, a maximum throughput improvement is achieved when $k = 3$. If we correct erroneous results with EDC as in Figure 4, then 17.2% additional clock cycles are required for error correction. With this overhead, ACA adder can improve data throughput by 24.6% over the conventional CLA adder.

3.4 Approximate Adder Comparison

We evaluate each approximate adder with respect to the pass rate and the accuracy metrics which we have proposed. We use gate-level simulation at each possible clock period to compare five adders: CLA, Lu's adder [7], ETAI, ETAIIM [14] and the proposed ACA adder (without error correction). In the experiment, the same carry-chain width (8-bit) is selected for the four approximate adders. In the implementation, a register (flip-flop) is inserted in each output port to detect timing errors.

Table 4 shows area, pass rate, accuracy, minimum clock period and EDC overhead for each adder design. According to the results, the ETAI adder has the smallest design area, but has a low pass rate and limited accuracy with respect to the ACC_{inf} metric. Therefore, the ETAI adder is preferred for applications which allow low accuracy in results. The ETAIIM adder shows fairly high accuracy, but does not have speed (clock period) benefit. Lu's adder shows a smaller error rate and high accuracy with respect to both ACC_{amp} and ACC_{inf} metrics. However, it requires larger area than the other designs. The proposed adder shows similar results for both metrics as Lu's adder. However, the area of the ACA adder is smaller than that of Lu's adder, and EDC is possible with small area overhead (28%). With the ACA adder, the minimum clock period can be reduced by 26% compared to the accurate CLA.

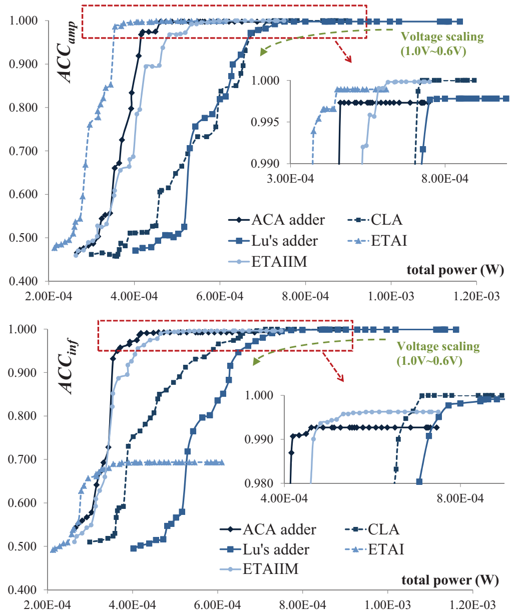

Figure 7: Accuracy (y-axis) vs. power consumption (x-axis) under fixed clock period (0.25ns) and scaled voltage (from 1.0V to 0.6V).

Figure 7 shows a power vs. accuracy tradeoff in a voltage scaling scenario: the x-axis shows total power consumption, and the y-axis shows the accuracy (ACC_{amp}, ACC_{inf}). The power consumption and the accuracy are measured with different voltage libraries characterized using *Cadence Library Characterizer* [19]. The clock period is fixed at 0.30ns during the simulations. In the results, Lu's adder does not show power benefits due to its design size. ETAI shows low power consumption and high ACC_{amp} accuracy, but has low ACC_{inf} accuracy, and cannot detect and correct errors. ETAIIM shows similar characteristics to ACA in the voltage scaling case, but the adder cannot be used for a high-performance (high-frequency) design, as shown in Table 4. The results in Figure 7 imply that our proposed adder can provide a significant power reduction with small accuracy penalty. When the required accuracy is 0.970 (ACC_{amp}), the ACA adder shows 37.0%, 36.4% and 15.9% total power reduction over CLA, Lu's adder and ETAIIM, respectively.

We have tested our approximate adder on a real application – a Gaussian smoothing filter used in [6]. Gaussian smoothing is performed on the input image by convolving with a matrix in the spatial domain. In the convolution, the addition operation is done with approximate 16-bit adders. Other operations, such as multiplication and division, are accurate computations. Figure 8 shows results for various approximate adders when they consume 50% of the power of accurate CLA. From the results, the ACA adder has PSNR of 24.5dB, and this suggests that image processing/filtering applications could employ our proposed adder with significant power savings and only small loss in image quality.

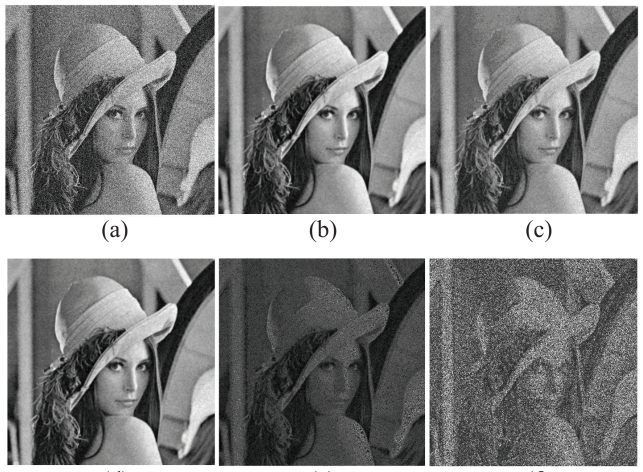

Figure 8: Image smoothing: (a) original image with noise; (b) accurate adder; (c) ACA, PSNR: 24.5 dB; (d) ETAI, PSNR: 25.3 dB; (e) ETAIIM, PSNR: 16.2 dB; (f) Lu's adder, PSNR: 11.1dB.

Table 5: Comparison between conventional and approximate (2-stage) pipelined adders at the accurate mode.

	conventional pipelined				approximate pipelined		
adder width (N)	area (um^2)	clock period (ns)	total power (mW)	k	area (um^2)	clock period (ns)	total power (mW)
8	459	0.313	0.557	2	576	0.312	0.564
16	1082	0.357	1.558	4	1171	0.358	1.669
32	2252	0.404	2.860	8	2420	0.414	2.914

Table 6: Implementation results of 32-bit ACA adder with 4-stage pipeline (power consumption of each mode and power reduction over conventional pipelined adder).

config.	power-gating	ACC_{amp} (max.)	ACC_{inf} (max.)	total power (mW)	reduction (%)
mode-1	none	1.000	1.000	5.962	-11.5%
mode-2	stage-4	0.998	0.960	4.683	12.4%
mode-3	stage-3, 4	0.991	0.925	3.691	31.0%
mode-4	stage-2, 3, 4	0.983	0.900	2.588	51.6%

3.5 Accuracy Configuration and Power Savings

When the architecture allows pipelining for addition, our proposed adder can be implemented as shown in Figure 5. We implement both the conventional pipelined adder and the approximate pipelined adder to compare the designs in terms of area, timing and power. In the implementation, registers (flip-flops) are included at each pipeline stage (before *stage-1*, between *stage-1* and *stage-2*, and after *stage-2*).

Table 5 shows the implementation results for the conventional and approximate pipelined adders. The parameter k has been selected as $N/4$ for a two-stage pipelined implementation. In the table, minimum clock period is measured at a fixed voltage ($1.0V$), and total power is measured at a fixed frequency ($2.5GHz$) with voltage scaling. In the ACA adder case, timing and power overheads from power gating cells, output MUXes, and IR drop are included. We can see that area, timing and power of both designs are similar when the ACA adder operates in the accurate mode. Total power of the approximate adder is comparable to that of the conventional adder, even though ACA has additional EDC circuits. This is because ACA has fewer registers between *stage-1* and *stage-2* than the conventional pipelined adder. (In Figure 5, the conventional adder requires registers for A_H, B_H, SUM_L and carry at the first stage. For a 16-bit adder, 25 registers (8 + 8 + 8 + 1) are required. On the other hand, ACA requires 18 registers (16 for SUM_{approx} and 2 for error indication).)

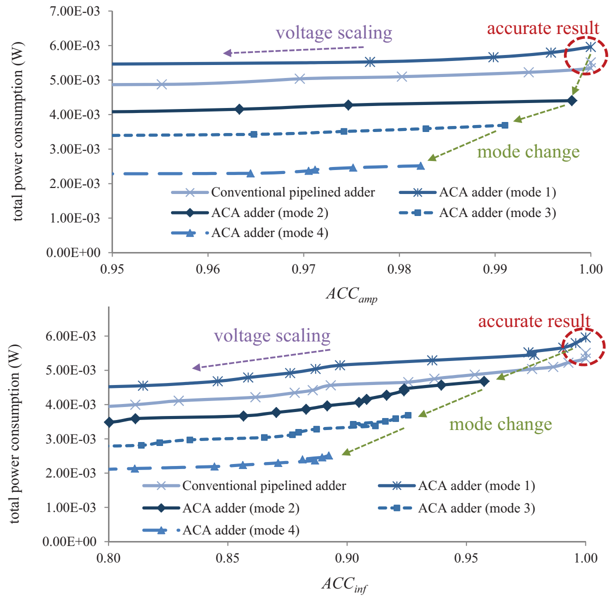

Figure 9: Accuracy metric ACC_{amp} (above) and ACC_{inf} (below) vs. power consumption for conventional pipelined adder, ACA adder in accurate mode, and ACA adder in approximate mode (4-stage, 32-bit adder).

In the pipelined architecture, the ACA adder can provide various configurable modes according to the pipeline depth. To improve the design performance, we increase the pipeline depth; the deeper pipeline reduces the path depth of the design. In the conventional pipelined adder, bit-width of the adder in each stage can be reduced to $N/\#stage$, where N is the entire bit-width and $\#stage$ is the depth (number) of the pipeline stages. In the ACA adder, we can reduce the value of parameter k with deeper pipeline depth as shown in Figure 6. To show the benefit of accuracy configuration, we have implemented a 32-bit ACA adder ($N = 32$, $k = 4$) with 4-stage pipeline, and compared it with a conventional pipelined adder with an 8-bit CLA in each stage. Table 6 shows the implemented results for the 32-bit ACA adder. For the accuracy estimation, one million cycles of random patterns are used. The ACA adder can operate in four different modes, based on the power gating of each stage. We can see that the modes show different power consumptions and different achievable accuracies. The ACA adder consumes 11.5% more power than the conventional adder in accurate mode (mode-1) due to the presence of recovery circuits. At the same time, it shows a significant power reduction in the approximate modes: 12.4%, 31.0% and 51.6% in mode-2, mode-3 and mode-4, respectively. Figure 9 shows detailed results for power consumption versus accuracy metrics in each configuration. From the results, we can see that accuracy configuration with the mode change is much more effective than with voltage scaling, in terms of the tradeoff between accuracy and power.

Table 7: Accuracy (ACC_{amp}, ACC_{inf}) results of 32-bit ACA adder for real benchmarks (SPEC 2006).

accuracy metric	benchmark	astar	bzip2	calculix	gcc	h264ref	mcf	sjeng	soplex
ACC_{amp}	mode-1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	mode-2	0.9999	1.0000	0.9999	0.9992	0.9999	0.9997	0.9998	0.9999
	mode-3	0.9993	0.9998	0.9972	0.9990	0.9990	0.9997	0.9995	0.9998
	mode-4	0.9979	0.9970	0.9958	0.9951	0.9978	0.9991	0.9981	0.9953
ACC_{inf}	mode-1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	mode-2	0.9979	1.0000	0.9978	0.9881	0.9953	0.9819	0.9897	0.9985
	mode-3	0.9949	0.9984	0.9967	0.9849	0.9897	0.9809	0.9876	0.9965
	mode-4	0.9940	0.9931	0.9910	0.9617	0.9851	0.9596	0.9787	0.9925

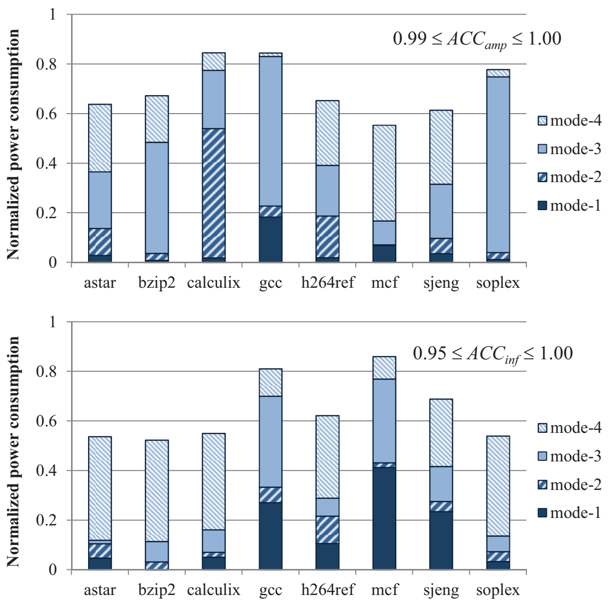

Figure 10: Normalized power consumption versus conventional pipelined design when the accuracy requirement is varied uniformly over the interval 0.99 $\leq ACC_{amp} \leq$ 1.00 and 0.95 $\leq ACC_{inf} \leq$ 1.00.

We also obtain the accuracy results in each accuracy mode with real input patterns extracted from SPEC 2006 benchmarks. Table 7 shows accuracy results of a 32-bit ACA adder with such real input patterns. The accuracy results are different for each benchmark, e.g, the measured accuracy for $bzip2$ is higher than for gcc. Furthermore, the accuracy with real patterns is greater than with random input patterns (Table 6), most likely because addition inputs for MPU have infrequently and/or systematically changing patterns in the applications. We evaluate power reductions across accuracy requirements with the patterns from SPEC 2006 benchmarks. Figure 10 shows power reduction achieved by the ACA adder versus the conventional pipelined adder under the accuracy requirements. We assume that required accuracy is from 0.99 (0.95) to 1.0 for ACC_{amp} (ACC_{inf}), and that it varies uniformly over this range during the entire runtime. From the results, dynamic accuracy configuration achieves up to 44.5% (30.0% on average) and 47.1% (35.8% on average) power reduction over the conventional pipelined design for ACC_{amp} and ACC_{inf} metrics, respectively.

4. CONCLUSIONS

In this paper, we propose an accuracy-configurable approximate (ACA) adder for which the accuracy of results is configurable during runtime. Due to its configurability, the ACA adder can operate adaptively in both approximate (inaccurate) mode and accurate mode. To quantify the accuracy in approximate computation, we provide two metrics for amplitude data and information data. We compare the ACA adder against previous approximate adders based on the proposed metrics. The ACA adder shows high accuracy with respect to the metrics, and can provide up to 24.6% throughput improvement and 37.0% power reduction over the conventional CLA adder. The ACA adder can also be used in accuracy-configurable applications with pipelining. We demonstrate that the ACA adder can provide approximately 30% power reduction under a relaxed accuracy requirement versus the conventional pipelined adder. Finally, we show that our ACA adder can improve the achievable tradeoff between performance, power and quality for given accuracy requirements.

Our ongoing work seeks to implement accuracy-configurable designs for other arithmetic components such as multipliers, multi-input adders, etc. More broadly, our research addresses additional aspects of (runtime) accuracy-configurable systems and applications.

5. REFERENCES

[1] M. A. Breuer, "Intelligible Test Techniques to Support Error-Tolerance", *Proc. Asian Test Symp.*, 2004, pp. 386–393.

[2] I. Chong, H. Y. Cheong and A. Ortega, "New Quality Metric for Multimedia Compression Using Faulty Hardware", *Proc. International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006, pp. 267–272.

[3] J. George, B. Marr, B. E. S. Akgul and K. V. Palem, "Probabilistic Arithmetic and Energy Efficient Embedded Signal Processing", *Proc. CASES*, 2006, pp. 158–168.

[4] S. Ghosh and K. Roy, "Parameter Variation Tolerance and Error Resiliency: New Design Paradigm for the Nanoscale Era", *Proceedings of the IEEE* 98(10) (2010), pp. 1718–1751.

[5] P. Kulkarni, P. Gupta and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture", *Proc. IEEE/ACM International Conference on VLSI Design*, 2011, pp. 346–351.

[6] M. S. Lau, K.-V. Ling and Y.-C. Chu, "Energy-Aware Probabilistic Multiplier: Design and Analysis", *Proc. CASES*, 2009, pp. 281–290.

[7] S.-L. Lu, "Speeding Up Processing with Approximation Circuits", *IEEE Computer* 37(3) (2004) pp. 67-73.

[8] H. D. Mohammed and L. Hemmert, "Fast Pipelined Adder/Subtractor using Increment/Decrement Function with Reduced Register Utilization", *U.S. Patent* No. 7,007,059, 2006.

[9] B. J. Phillips, D. R. Kelly and B. W. Ng, "Estimating Adders for a Low Density Parity Check Decoder", *Proc. SPIE*, vol. 6313, 2006, pp. 1–9.

[10] D. Shin and S. K. Gupta, "A Re-Design Technique for Datapath Modules in Error Tolerant Applications", *Proc. Asian Test Symp.*, 2008, pp. 431–437.

[11] D. Shin and S. K. Gupta, "Approximate Logic Synthesis for Error Tolerant Applications", *Proc. DATE*, 2010, pp. 957–960.

[12] A. K. Verma, P. Brisk and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design", *Proc. DATE*, 2008, pp. 1250–1255.

[13] N. Zhu, W. Goh and K. Yeo, "An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application" *Proc. Intl. Symp. on Integrated Circuits*, 2009, pp. 69–72.

[14] N. Zhu, W. Goh, W. Zhang, K. Yeo and Z. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing", *IEEE Trans. on VLSI Systems* 18(8) (2010), pp. 1225–1229.

[15] M. Ziegler and M. Stan, "Optimal Logarithmic Adder Structures with a Fanout of Two for Minimizing the Area-Delay Product", *Proc. ISCAS*, 2001, pp. 657–660.

[16] *International Technology Roadmap for Semiconductors*, 2009, http://www.itrs.net .

[17] *Synopsys Design Compiler User's Manual.* http://www.synopsys.com .

[18] *NC-Sim User's Manual.* http://www.cadence.com .

[19] *Cadence LC User's Manual.* http://www.cadence.com .

[20] *Standard Performance Evaluation Corporation (SPEC) CPU2006.* http://www.spec.org/cpu2006 .