

Low-Power Digital Signal Processing Using Approximate Adders

Vaibhav Gupta, Debabrata Mohapatra, Anand Raghunathan, *Fellow, IEEE*, and Kaushik Roy, *Fellow, IEEE*

Abstract—Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather useful information from slightly erroneous outputs. Therefore, we do not need to produce exactly correct numerical outputs. Previous research in this context exploits error resiliency primarily through voltage overscaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. In this paper, we propose logic complexity reduction at the transistor level as an alternative approach to take advantage of the relaxation of numerical accuracy. We demonstrate this concept by proposing various imprecise or approximate full adder cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders. In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. We design architectures for video and image compression algorithms using the proposed approximate arithmetic units and evaluate them to demonstrate the efficacy of our approach. We also derive simple mathematical models for error and power consumption of these approximate adders. Furthermore, we demonstrate the utility of these approximate adders in two digital signal processing architectures (discrete cosine transform and finite impulse response filter) with specific quality constraints. Simulation results indicate up to 69% power savings using the proposed approximate adders, when compared to existing implementations using accurate adders.

Index Terms—Approximate computing, low power, mirror adder.

I. INTRODUCTION

DIGITAL SIGNAL processing (DSP) blocks form the backbone of various multimedia applications used in portable devices. Most of these DSP blocks implement image and video processing algorithms, where the ultimate output is either an image or a video for human consumption. Human beings have limited perceptual abilities when interpreting an image or a video. This allows the outputs of these algorithms to be numerically approximate rather than accurate. This relaxation on numerical exactness provides some freedom to carry out imprecise or approximate computation. We can use

Manuscript received March 2, 2012; revised May 10, 2012, July 28, 2012; accepted August 13, 2012. Date of current version December 19, 2012. This paper was recommended by Associate Editor I. Bahar.

V. Gupta, A. Raghunathan, and K. Roy are with the Department of Electrical and Computer Engineering, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: gupta64@purdue.edu; raghunathan@purdue.edu; kaushik@purdue.edu).

D. Mohapatra is with Intel Corporation, Santa Clara, CA 95054 USA (e-mail: debabrata.mohapatra@intel.com).

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/TCAD.2012.2217962

this freedom to come up with low-power designs at different levels of design abstraction, namely, logic, architecture, and algorithm.

The paradigm of approximate computing is specific to select hardware implementations of DSP blocks. It is shown in [1] that an embedded reduced instruction set computing processor consumes 70% of the energy in supplying data and instructions, and 6% of the energy while performing arithmetic only. Therefore, using approximate arithmetic in such a scenario will not provide much energy benefit when considering the complete processor. Programmable processors are designed for general-purpose applications with no application-specific specialization. Therefore, there may not be many applications that will be able to tolerate errors due to approximate computing. This also makes general-purpose processors not suited for using approximate building blocks. This issue has already been discussed in [13]. Therefore, in this paper, we consider application-specific integrated circuit implementations of error-resilient applications like image and video compression. We target the most computationally intensive blocks in these applications and build them using approximate hardware to show substantial improvements in power consumption with little loss in output quality.

Few works that focus on low-power design through approximate computing at the algorithm and architecture levels include algorithmic noise tolerance (ANT) [3]–[6], significance-driven computation (SDC) [7]–[9], and nonuniform voltage overscaling (VOS) [10]. All these techniques are based on the central concept of VOS, coupled with additional circuitry for correcting or limiting the resulting errors. In [11], a fast but “inaccurate” adder is proposed. It is based on the idea that on average, the length of the longest sequence of propagate signals is approximately $\log n$, where n is the bitwidth of the two integers to be added. An error-tolerant adder is proposed in [12] that operates by splitting the input operands into accurate and inaccurate parts. However, neither of these techniques target logic complexity reduction. A power-efficient multiplier architecture is proposed in [13] that uses a 2×2 inaccurate multiplier block resulting from Karnaugh map simplification. This paper considers logic complexity reduction using Karnaugh maps. Shin and Gupta [14] and Phillips *et al.* [15] also proposed logic complexity reduction by Karnaugh map simplification. Other works that focus on logic complexity reduction at the gate level are [16]–[19]. Other approaches use complexity reduction at the algorithm level to meet real-time energy constraints [20], [21].

Previous works on logic complexity reduction have focused on algorithm, logic, and gate levels. We propose logic complexity reduction at the transistor level. We apply this to addition at the bit level by simplifying the mirror adder (MA) circuit. We develop imprecise but simplified arithmetic units, which provide an extra layer of power savings over conventional low-power design techniques. This is attributed to the reduced logic complexity of the proposed approximate arithmetic units. Note that the approximate arithmetic units not only have a reduced number of transistors, but care is taken to ensure that the internal node capacitances are greatly reduced. Complexity reduction leads to power reduction in two different ways. First, an inherent reduction in switched capacitance and leakage results from having smaller hardware. Second, complexity reduction frequently leads to shorter critical paths, facilitating voltage reduction without any timing-induced errors. In summary, our work significantly differs from other works (SDC, ANT, and nonuniform VOS) since we adopt a different approach for exploiting error resiliency. Our aim is to target low-power design using simplified and approximate logic implementations. Since DSP blocks mainly consist of adders and multipliers (which are, in turn, built using adders), we propose several approximate adders, which can be used effectively in such blocks.

A preliminary version of our work appeared in [22]. We extend our paper in [22] by providing two more simplified versions of the MA. Furthermore, we propose a measure of the “quality” of a DSP block that uses approximate adders. We also propose a methodology that can be used to harness maximum power savings using approximate adders, subject to a specific quality constraint. Our contributions in this paper can be summarized as follows.

- 1) We propose logic complexity reduction at the transistor level as an alternative approach to approximate computing for DSP applications.
- 2) We show how to simplify the logic complexity of a conventional MA cell by reducing the number of transistors and internal node capacitances. Keeping this aim in mind, we propose five different simplified versions of the MA, ensuring minimal errors in the full adder (FA) truth table.
- 3) We utilize the simplified versions of the FA cell to propose several imprecise or approximate multi-bit adders that can be used as building blocks of DSP systems. To maintain a reasonable output quality, we use approximate FA cells only in the least significant bits (LSBs). We particularly focus on adder structures that use FA cells as their basic building blocks. We have used approximate carry save adders (CSAs) to design 4:2 and 8:2 compressors (Section III). We derive mathematical expressions for the mean error in the output of an approximate ripple carry adder (RCA) as a function of the number of LSBs that are approximate.
- 4) VOS is a very popular technique to get large improvements in power consumption. However, VOS will lead to delay failures in the most significant bits (MSBs). This might lead to large errors in corresponding outputs and severely mess up the output quality of the application.

We use approximate FA cells only in the LSBs, while the MSBs use accurate FA cells. Therefore, at isofrequency, the errors introduced by VOS will be much higher, when compared to proposed approximate adders. Since truncation is a well-known technique to facilitate voltage scaling, we have compared the performance of proposed approximate adders with truncated adders. Our expressions for mean error demonstrate the superiority of proposed approximations over truncation. Higher order compressors (using carry save trees) are also used to accumulate partial products in various tree multipliers [23]. So our approach is also useful for designing approximate tree multipliers, which are extensively used in DSP systems. In general, our approach may be applied to any arithmetic circuit built with FAs.

- 5) We present designs for image and video compression algorithms using the proposed approximate arithmetic units and evaluate the approximate architectures in terms of output quality and power dissipation.
- 6) We propose a mathematical definition of output quality of a DSP block that uses approximate adders. We also propose simple mathematical models for estimating the degree of voltage scaling and power consumption of these approximate adders. Based on these models, we formulate a design problem that can be solved to get maximum power savings, subject to a given quality constraint.
- 7) Finally, we demonstrate the optimization methodology with the help of two examples, discrete cosine transform (DCT) and finite impulse response (FIR) filter.

The remainder of this paper is organized as follows. In Section II, we propose and discuss various approximate FA cells. In Section III, we demonstrate the benefits of using these cells in image and video compression architectures. We derive mathematical models for mean error and power consumption of an approximate RCA in Section IV. In Section V, we demonstrate the efficient use of approximate RCAs in two DSP architectures, DCT and FIR filter. Finally, conclusions are drawn in Section VI.

II. APPROXIMATE ADDERS

In this section, we discuss different methodologies for designing approximate adders. We use RCAs and CSAs throughout our subsequent discussions in all sections of this paper. Since the MA [23] is one of the widely used economical implementations of an FA [24], we use it as our basis for proposing different approximations of an FA cell.

A. Approximation Strategies for the MA

In this section, we explain step-by-step procedures for coming up with various approximate MA cells with fewer transistors. Removal of some series connected transistors will facilitate faster charging/discharging of node capacitances. Moreover, complexity reduction by removal of transistors also aids in reducing the αC term (switched capacitance) in the dynamic power expression $P_{\text{dynamic}} = \alpha C V_{\text{DD}}^2 f$, where

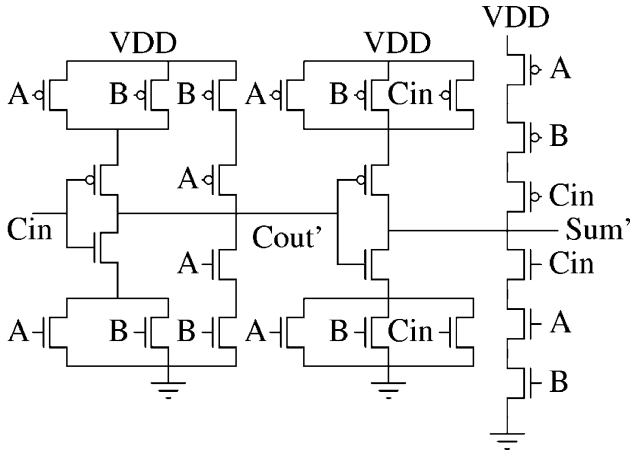


Fig. 1. Conventional MA.

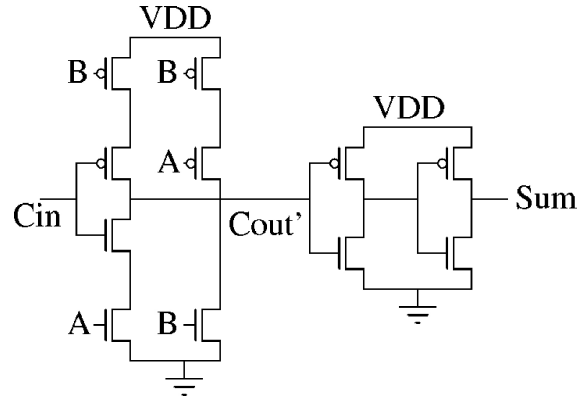


Fig. 4. MA approximation 3.

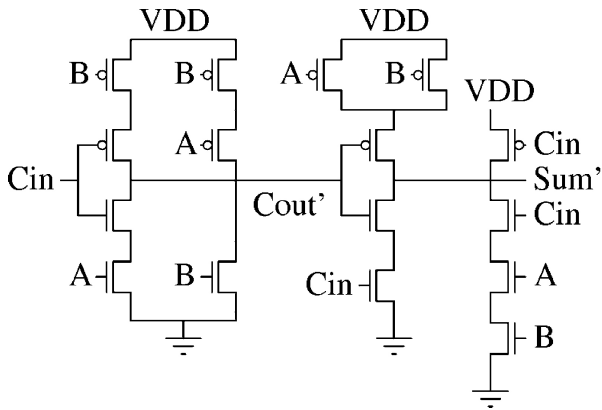


Fig. 2. MA approximation 1.

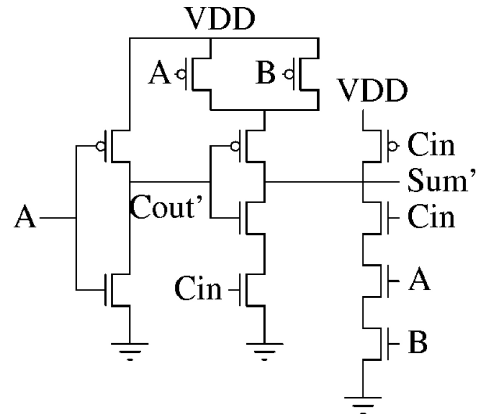


Fig. 5. MA approximation 4.

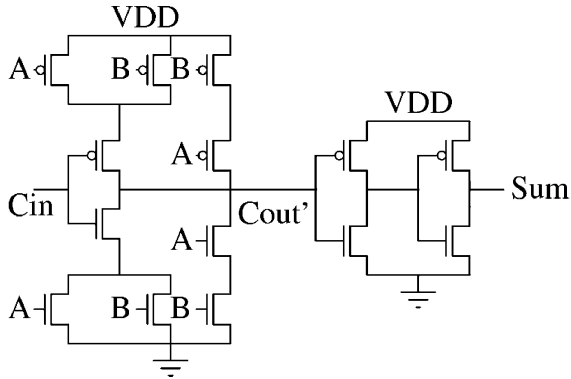


Fig. 3. MA approximation 2.

α is the switching activity or average number of switching transitions per unit time and C is the load capacitance being charged/discharged. This directly results in lower power dissipation. Area reduction is also achieved by this process. Now, let us discuss the conventional MA implementation followed by the proposed approximations.

1) *Conventional MA*: Fig. 1 shows the transistor-level schematic of a conventional MA [23], which is a popular way of implementing an FA. It consists of a total of 24 transistors. Since this implementation is not based on complementary

CMOS logic, it provides a good opportunity to design an approximate version with removal of selected transistors.

2) *Approximation 1*: In order to get an approximate MA with fewer transistors, we start to remove transistors from the conventional schematic one by one. However, we cannot do this in an arbitrary fashion. We need to make sure that any input combination of A, B and C_{in} does not result in short circuits or open circuits in the simplified schematic. Another important criterion is that the resulting simplification should introduce minimal errors in the FA truth table. A judicious selection of transistors to be removed (ensuring no open or short circuits) results in a schematic shown in Fig. 2, which we call approximation 1. Clearly, this schematic has eight fewer transistors compared to the conventional MA schematic. In this case, there is one error in C_{out} and two errors in Sum , as shown in Table I. A tick mark denotes a match with the corresponding accurate output and a cross denotes an error.

3) *Approximation 2*: The truth table of an FA shows that $Sum = \overline{C_{out}}$ for six out of eight cases, except for the input combinations $A = 0, B = 0, C_{in} = 0$ and $A = 1, B = 1, C_{in} = 1$. Now, in the conventional MA, $\overline{C_{out}}$ is computed in the first stage. Thus, an easy way to get a simplified schematic is to set $Sum = \overline{C_{out}}$. However, we introduce a buffer stage after $\overline{C_{out}}$ (see Fig. 3) to implement the same functionality. The reason for this can be explained as follows. If we set $Sum = \overline{C_{out}}$ as it is

¹Henceforth, we denote the complement of a variable V by \overline{V} .

TABLE I
TRUTH TABLE FOR CONVENTIONAL FA AND APPROXIMATIONS 1–4

Inputs			Accurate Outputs		Approximate Outputs							
A	B	C _{in}	Sum	C _{out}	Sum ₁	C _{out1}	Sum ₂	C _{out2}	Sum ₃	C _{out3}	Sum ₄	C _{out4}
0	0	0	0	0	0 ✓	0 ✓	1 ×	0 ✓	1 ×	0 ✓	0 ✓	0 ✓
0	0	1	1	0	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓
0	1	0	1	0	0 ×	1 ×	1 ✓	0 ✓	0 ×	1 ×	0 ×	0 ✓
0	1	1	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	1 ×	0 ×
1	0	0	1	0	0 ×	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	0 ×	1 ×
1	0	1	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓
1	1	0	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓
1	1	1	1	1	1 ✓	1 ✓	0 ×	1 ✓	0 ×	1 ✓	1 ✓	1 ✓

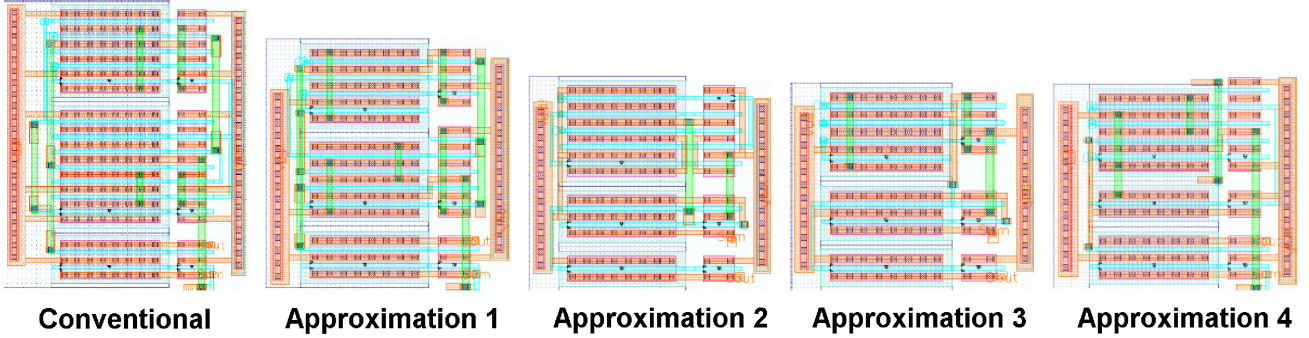


Fig. 6. Layouts of conventional and approximate MA cells.

in the conventional MA, the total capacitance at the Sum node would be a combination of four source–drain diffusion and two gate capacitances. This is a considerable increase compared to the conventional case or approximation 1. Such a design would lead to a delay penalty in cases where two or more multi-bit approximate adders are connected in series, which is very common in DSP applications. Fig. 3 shows the schematic obtained using the above approach. We call this approximation 2. Here, Sum has only two errors, while C_{out} is correct for all cases, as shown in Table I.

4) *Approximation 3*: Further simplification can be obtained by combining approximations 1 and 2. Note that this introduces one error in C_{out} and three errors in Sum, as shown in Table I. The corresponding simplified schematic is shown in Fig. 4.

5) *Approximation 4*: A close observation of the FA truth table shows that $C_{out} = A$ for six out of eight cases. Similarly, $C_{out} = B$ for six out of eight cases. Since A and B are interchangeable, we consider $C_{out} = A$. Thus, we propose a fourth approximation where we just use an inverter with input A to calculate $\overline{C_{out}}$ and Sum is calculated similar to approximation 1. This introduces two errors in C_{out} and three errors in Sum, as shown in Table I. The corresponding simplified schematic is shown in Fig. 5.

In all these approximations, C_{out} is calculated by using an inverter with C_{out} as input.

6) *Approximation 5*: In approximation 4, we find that there are three errors in Sum. We extend this approximation by allowing one more error, i.e., four errors in Sum. We use the approximation $C_{out} = A$, as in approximation 4. If we want to make Sum independent of C_{in} , we have two choices, Sum = A and Sum = B . Thus, we have two alternatives for approxima-

tion 5, namely, Sum = A , $C_{out} = A$ and Sum = B , $C_{out} = A$, which are shown in Table II. The table shows which entries match with and differ from the corresponding accurate outputs (shown by tick marks and crosses). If we observe choice 1, we find that both Sum and C_{out} match with accurate outputs in only two out of eight cases. In choice 2, Sum and C_{out} match with accurate outputs in four out of eight cases. Therefore, to minimize errors both in Sum and C_{out} , we go for choice 2 as approximation 5. Our main thrust here is to ensure that for a particular input combination (A , B and C_{in}), ensuring correctness in Sum also makes C_{out} correct. Now consider the addition of two 20-b integers $a[19 : 0]$ and $b[19 : 0]$ using an RCA. Suppose we use approximate FAs for 7 LSBs. Then, $C_{in}[7] = C_{out}[6]$. Note that $C_{out}[6]$ is approximate. Applying this approximation to our present example, we find that carry propagation from bit 0 to bit 6 is entirely eliminated. In addition, the circuitry needed to calculate $C_{out}[0]$ to $C_{out}[5]$ is also saved. To limit the output capacitance at Sum and C_{out} nodes, we implement approximation 5: Sum = B , $C_{out} = A$ using buffers.

Layouts of conventional MA and different approximations in IBM 90-nm technology are shown in Fig. 6. Layout area for the conventional MA and different approximations are compared in Table III. Approximation 5 uses only buffers. The layout area of a single buffer is $6.77 \mu\text{m}^2$.

B. Voltage Scaling Using Approximate Adders

Let us discuss how the proposed approximations also help in reducing the overall propagation delay in a typical design involving several adder levels. The input capacitance of C_{in} in a conventional MA consists of six gate capacitances (see Fig. 1). In approximation 1, this value is reduced to five gate

TABLE II
CHOOSING APPROXIMATION 5

Choice 1		Choice 2	
Sum= A	C _{out} = A	Sum= B	C _{out} = A
0 ✓	0 ✓	0 ✓	0 ✓
0 ×	0 ✓	0 ×	0 ✓
0 ×	0 ✓	1 ✓	0 ✓
0 ✓	0 ×	1 ×	0 ×
1 ✓	1 ×	0 ×	1 ×
1 ×	1 ✓	0 ✓	1 ✓
1 ×	1 ✓	1 ×	1 ✓
1 ✓	1 ✓	1 ✓	1 ✓

TABLE III
LAYOUT AREA OF APPROXIMATE MA CELLS

MA Cell	Area (μm^2)
Conventional	40.66
Approximation 1	29.31
Approximation 2	25.5
Approximation 3	22.56
Approximation 4	23.91

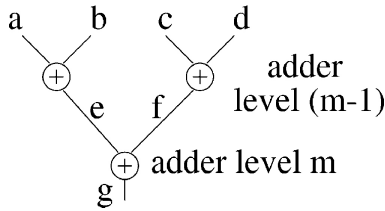


Fig. 7. Adder tree section.

capacitances. Approximation 4 has three such gate capacitances, which further reduces to only two gate capacitances in approximations 2 and 3. This results in faster charging/discharging of C_{out} nodes during carry propagation. Now consider a section of a multilevel adder tree using RCAs as shown in Fig. 7. The Sum bits of outputs e and f become input bits A and B for output g . A reduction in input capacitances at nodes A and B of adder level m results in a faster computation of LSB Sum bits of adder level $m - 1$. This decreases the total critical path of the adder tree. The input capacitance at node A consists of eight gate capacitances in the conventional case. This is reduced to four gate capacitances in approximations 1, 2, and 4, and only two gate capacitances in approximation 3. Similarly, the corresponding values for node B are 8, 5, 4, 3, and 2 gate capacitances for the following cases: conventional, approximations 1–4, respectively. Thus, a reduction in load capacitances is the crux of the proposed approximations, offering an appreciable reduction in propagation delay and providing an opportunity for operating at a lower supply voltage than the conventional case.

Since the proposed approximate FA cells have fewer transistors, this also results in area savings. Approximation 5 provides maximum area savings among all approximations. In Section III, we use the approximate FA cells to design architectures for image and video compression and highlight the potential benefits.

III. IMAGE AND VIDEO COMPRESSION USING APPROXIMATE ARITHMETIC UNITS

In Section II, several approximate FA cells were introduced. Using these approximate FA cells also introduces errors in the truth table of an FA. When approximate FA cells are used to design multi-bit adders, the outputs of these adders will be erroneous. Multimedia DSP algorithms mostly consist of additions and multiplications. Multiplications can be treated as shifts and adds. Therefore, adders can be considered as basic building blocks for these algorithms. Interestingly, most DSP algorithms used in multimedia systems are characterized by inherent error tolerance. Hence, occasional errors in intermediate outputs might not manifest as a substantial reduction in the final output quality. We focus on two algorithms, namely, image and video compression, and present the results of using our approximate FA cells in these algorithms. We use approximate FA cells only in the LSBs, thus ensuring that the final output quality does not degrade too much.

A. Image Compression

The DCT and inverse discrete cosine transform (IDCT) are integral components of a Joint Photographic Experts Group (JPEG) image compression system [25]. One-dimensional integer DCT $y(k)$ for an eight-point sequence $x(i)$ is given by [26]

$$y(k) = \sum_{i=0}^7 a(k, i)x(i), \quad k = 0, 1, \dots, 7. \quad (1)$$

Here, $a(k, i)$ are cosine functions converted into equivalent integers [8]. The integer outputs $y(k)$ can then be right shifted to get the actual DCT outputs. A similar expression can be found for 1-D integer IDCT [9]. We alter the integer coefficients $a(k, i), k = 1, \dots, 7$ so that the multiplication $a(k, i)x(i)$ is converted to two left shifts and an addition (using an RCA). Since $a(0, i)$ corresponds to the dc coefficient, which is most important, we leave it unaltered. The multiplication $a(0, i)x(i)$ then corresponds to an addition of four terms. This is done using a carry-save tree using a 4:2 compressor followed by an RCA. Also, each integer DCT and IDCT output is the sum of eight terms. Thus, these outputs are calculated using a carry-save tree using an 8:2 compressor followed by an RCA. Thus, the whole DCT–IDCT system now consists of RCAs and CSAs. In our design, all RCAs and CSAs are approximate, which use the approximate FA cells proposed earlier.

We consider three cases, where we use approximate FA cells for 7–9 LSBs. FA cells corresponding to other bits in each case are accurate. According to our experiments, using approximate FA cells beyond the ninth LSB results in an appreciable quality loss. So we consider only these three cases. In order to see how truncation fares, we also show results where we truncate 7–9 LSBs instead of computing them approximately. The design using accurate adders everywhere in DCT and IDCT is considered to be the base case.

1) *Output Quality*: We measure the output quality of the decoded image after IDCT using the well-known metric of peak signal-to-noise ratio (PSNR). The output PSNR for the base case is 31.16 dB. Fig. 8 shows the output images for the

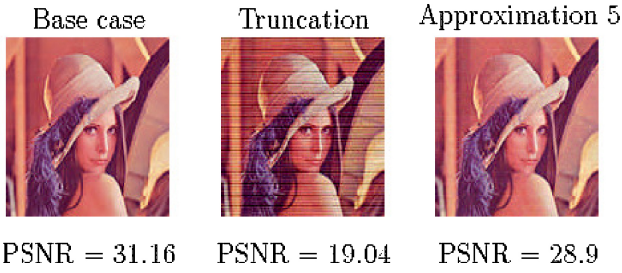


Fig. 8. Output quality when 8 LSBs are approximated.

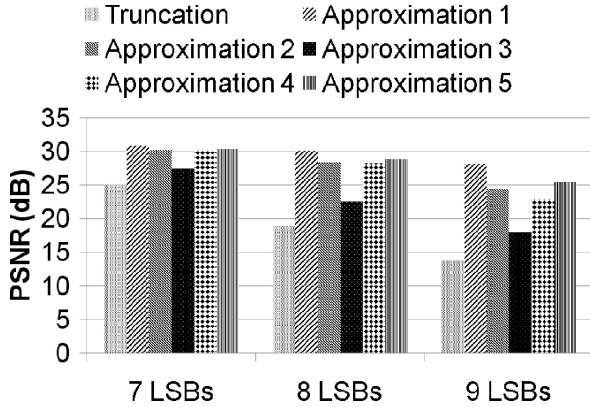


Fig. 9. Output quality for different techniques.

base case, truncation, and approximation 5. We can see severe blockiness in the output images using truncation. This suggests that truncation is a bad idea when more LSBs are approximate. Fig. 9 shows the output quality for truncation and different approximations when 7–9 LSBs are approximated. Truncation leads to an appreciable decrease in PSNR for all cases. On the other hand, using approximate FAs in the LSBs can make up for the lost quality to a large extent, and also provide substantial power savings. Power consumption for different approximations is discussed in Section III-A2.

2) *Power Consumption*: As mentioned in Section II-B, both DCT and IDCT blocks can be operated at a lower supply voltage (compared to the base case) when using approximate adders. Table IV shows the operating supply voltages for different approximations and truncation in IBM 90-nm technology. The power consumption for DCT and IDCT blocks was determined using nanosim [27] run with 12288 vectors from the standard image *Lena* in IBM 90-nm technology. Fig. 10 shows the total power savings for DCT and IDCT blocks over the base case for different approximations and truncation. Approximation 5 provides maximum power savings among all approximations. It is interesting to note that it provides $\approx 60\%$ power savings when 9 LSBs are approximated, with a PSNR of 25.46 dB. In the same scenario, truncation provides $\approx 61\%$ power savings, but the output quality is severely degraded with a PSNR of 13.87 dB.

B. Video Compression

In this section, we illustrate the application of proposed approximate adders in video compression, which is widely

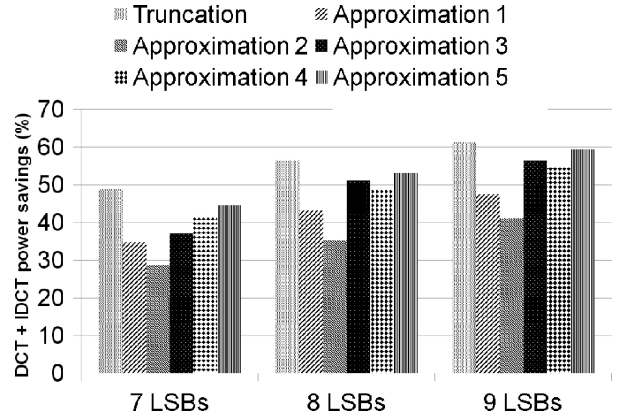


Fig. 10. Power savings for DCT + IDCT over the base case.

TABLE IV
OPERATING VOLTAGES FOR DIFFERENT TECHNIQUES

Technique	V_{DD} (V) for the Three Cases					
	7 LSBs		8 LSBs		9 LSBs	
	DCT	IDCT	DCT	IDCT	DCT	IDCT
Truncation	1.13	1.03	1.10	1.03	1.1	1
Approximation 1	1.18	1.05	1.15	1.03	1.15	1.03
Approximation 2	1.18	1.05	1.15	1.05	1.13	1.05
Approximation 3	1.18	1.05	1.1	1.03	1.1	1.03
Approximation 4	1.15	1.1	1.13	1.1	1.1	1.1
Approximation 5	1.14	1.02	1.11	1.01	1.1	1

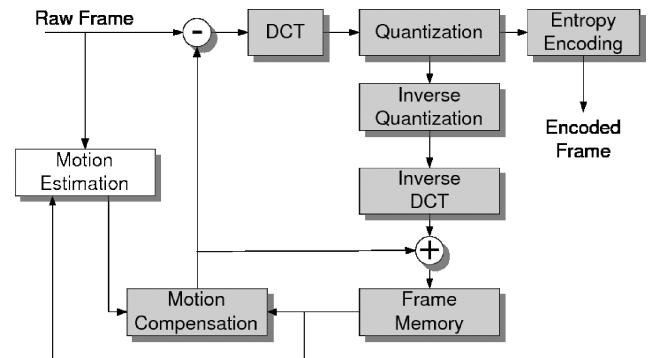


Fig. 11. MPEG encoder architecture.

used in portable mobile devices. Fig. 11 shows the block diagram of MPEG encoder architecture [28].

Here, we focus on motion estimation (ME) [6] hardware that accounts for nearly 70% of the total power consumption [28]. In addition to the ME block, we have implemented a few other adders/subtractors using approximate adders that are shown in white (ME, subtractor for DCT error, and IDCT reconstruction adder) in Fig. 11. The basic building block for any ME algorithm is the hardware that implements sum of absolute difference (SAD) [28]. We have implemented the basic SAD architecture that consists of an 8-b absolute difference block followed by a 16-b accumulator. We have used various approximate adders discussed earlier to design a low-power approximate SAD and compared its quality and power to the nominal design as well as truncation.

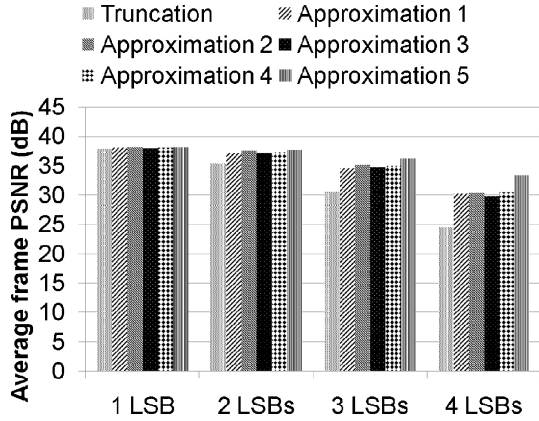
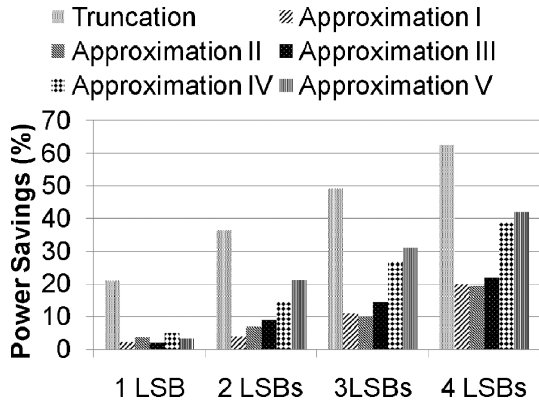
Fig. 12. Output quality for *Akiyo* sequence.

Fig. 13. Power savings over accurate adders for video compression.

1) *Output Quality*: Fig. 12 shows the average frame PSNR for 50 frames of the *Akiyo* benchmark CIF [28] video sequence. We consider the frame quality for truncation as well as the five approximations applied to 1–4 LSBs of the adders in SAD and MPEG hardware shown in white (ME, subtractor for DCT error, and IDCT reconstruction adder) in Fig. 11. With the increase in number of LSBs implemented as imprecise adders, our proposed approximations scale more gracefully in terms of quality when compared to truncation.

2) *Power Consumption*: Fig. 13 shows the power savings for approximate adders. Again, approximation 5 provides maximum power savings ($\approx 42\%$ when 4 LSBs are approximated) among all approximations. As mentioned earlier, truncation results in better power savings compared to all approximations. However, it also results in significant degradation in output frame quality.

IV. POWER AND ERROR MODELS FOR APPROXIMATE ADDERS

In this section, we derive mathematical models for mean error and power consumption of approximate adders.

A. Modeling Error in Approximate Adders

Let us denote the signal probabilities $P(A[x] = 1)$ and $P(B[x] = 1)$ by a_x and b_x , respectively. Here x is the bit

position (starting from 0). Let the accurate signal probabilities $P(\text{Sum}[x] = 1)$ and $P(C_{\text{in}}[x] = 1)$ be denoted by s_x and c_x , respectively. Similarly, let the approximate signal probabilities $P(\text{Sum}'[x] = 1)$ and $P(C'_{\text{in}}[x] = 1)$ be denoted by s'_x and c'_x , respectively. Without loss of generality, let us assume the input bit probabilities $a_x = 0.5$ and $b_x = 0.5$. We calculate the signal probabilities s_x , c_x , s'_x , and c'_x for both conventional FA and approximate FA cells using the ideas mentioned in [29].

1) *Conventional FA*: For the accurate FA, $\text{Sum} = ABC_{\text{in}} + \bar{A}\bar{B}C_{\text{in}} + \bar{A}BC_{\text{in}} + A\bar{B}C_{\text{in}}$, $C_{\text{out}} = AB + \bar{A}BC_{\text{in}} + A\bar{B}C_{\text{in}}$. This gives us the following expressions for s_x and c_{x+1} :

$$\begin{aligned} s_x &= a_x b_x c_x + (1 - a_x)(1 - b_x)c_x \\ &+ (1 - a_x)b_x(1 - c_x) + a_x(1 - b_x)(1 - c_x) \\ &= a_x + b_x + c_x - 2a_x b_x - 2a_x c_x - 2b_x c_x + 4a_x b_x c_x \\ &= 0.5 \end{aligned}$$

$$\begin{aligned} c_{x+1} &= a_x b_x + (1 - a_x)b_x c_x + (1 - b_x)a_x c_x \\ &= a_x b_x + b_x c_x + a_x c_x - 2a_x b_x c_x \\ &= (2c_x + 1)/4. \end{aligned}$$

Clearly, $c_0 = 0$. Thus, using the recursive relationship for c_{x+1} in the above equation, $c_x = 0.5 - 2^{-(x+1)}$, $x = 1, 2, \dots$. Similar to the accurate case, Sum and C_{out} expressions for all approximations can be calculated using Tables I and II. This leads to explicit expressions for s'_x and c'_x for all approximations, which are shown next.

2) *Approximation 1*: $\text{Sum}' = \bar{A}\bar{B}C'_{\text{in}} + ABC'_{\text{in}}$, $C'_{\text{out}} = ABC'_{\text{in}} + \bar{A}BC'_{\text{in}} + A\bar{B}C'_{\text{in}} + \bar{A}BC'_{\text{in}} + \bar{A}BC'_{\text{in}}$

$$s'_0 = 0, \quad c'_x = \frac{2}{3} - \frac{1}{3 \cdot 2^{2x-1}}, \quad s'_x = \frac{1}{3} - \frac{1}{3 \cdot 2^{2x}}, \quad x = 1, 2, \dots$$

3) *Approximation 2*: $C'_{\text{out}} = AB + \bar{A}BC'_{\text{in}} + A\bar{B}C'_{\text{in}}$, $\text{Sum}' = \frac{C'_{\text{out}}}{C'_{\text{out}}}$

$$s'_0 = \frac{3}{4}, \quad c'_x = \frac{1}{2} - \frac{1}{2^{x+1}}, \quad s'_x = \frac{1}{2} + \frac{1}{2^{x+2}}, \quad x = 1, 2, \dots$$

4) *Approximation 3*: $C'_{\text{out}} = ABC'_{\text{in}} + \bar{A}BC'_{\text{in}} + \bar{A}BC'_{\text{in}} + \bar{A}BC'_{\text{in}} + \bar{A}BC'_{\text{in}}$, $\text{Sum}' = \frac{C'_{\text{out}}}{C'_{\text{out}}}$

$$s'_0 = \frac{1}{2}, \quad c'_x = \frac{2}{3} - \frac{1}{3 \cdot 2^{2x-1}}, \quad s'_x = \frac{1}{3} + \frac{1}{3 \cdot 2^{2x+1}}, \quad x = 1, 2, \dots$$

5) *Approximation 4*: $\text{Sum}' = \bar{A}\bar{B}C'_{\text{in}} + \bar{A}BC'_{\text{in}} + \bar{A}BC'_{\text{in}}$, $C'_{\text{out}} = A$

$$s'_0 = 0, \quad c'_x = \frac{1}{2}, \quad s'_x = \frac{3}{8}, \quad x = 1, 2, \dots$$

6) *Approximation 5*: $\text{Sum}' = B$, $C'_{\text{out}} = A$

$$s'_0 = \frac{1}{2}, \quad c'_x = \frac{1}{2}, \quad s'_x = \frac{1}{2}, \quad x = 1, 2, \dots$$

Suppose y LSBs are approximate in an RCA.³ Then the error ϵ in approximate addition is given by

$$\begin{aligned} \epsilon &= (\text{Sum}'[0] - \text{Sum}[0]) + 2(\text{Sum}'[1] - \text{Sum}[1]) \\ &+ \dots + 2^{y-1}(\text{Sum}'[y-1] - \text{Sum}[y-1]) \\ &+ 2^y(C'_{\text{in}}[y] - C_{\text{in}}[y]) \\ &= e[0] + 2e[1] + \dots + 2^{y-1}e[y-1] + 2^y e[y]. \end{aligned} \quad (2)$$

²We denote the approximate version of a variable V by V' .

³RCA is used for ease of illustration. Similar derivations can be done for CSAs.

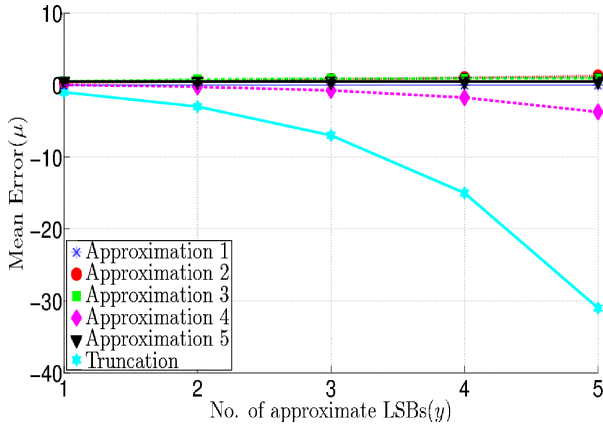


Fig. 14. Mean error in approximate addition.

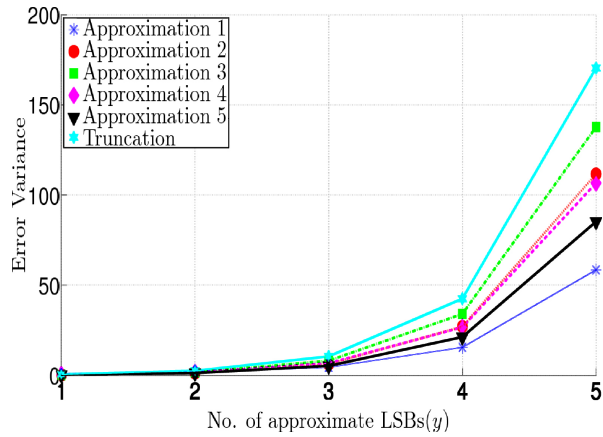


Fig. 15. Error variance in approximate addition.

Thus, the mean error μ can be written as

$$\begin{aligned} \mu(y) &= E(\epsilon) \\ &= \sum_{x=0}^y 2^x E(e[x]) \end{aligned} \quad (3)$$

where $E(\cdot)$ denotes the expectation.

Thus, $\mu(y)$ denotes the mean of the probability distribution of error in approximate addition (over an exhaustive set of inputs). The mean error for different approximations as a function of y is plotted in Fig. 14. A detailed derivation for all approximations is provided in the Appendix. We also compare the mean error for the proposed approximations with truncation, another well-known approximation technique. From the plot, it is clear that the proposed approximations are better than truncation in terms of mean error. Approximation 1 is the best (having an mean error of zero), while Approximation 4 is the worst. We also plot the variance of error for different approximations and truncation in Fig. 15. Again, Approximation 1 has the least variance. Truncation has larger variance than all approximations. This completely characterizes the probability distribution of error for the proposed approximations.

B. Modeling Power Consumption of Approximate Adders

Now we derive simple mathematical models for estimating the power consumption of an approximate RCA. Let C_{gn} and

TABLE V
CAPACITANCES FOR DIFFERENT APPROXIMATIONS

Approximation Technique	Node A	Node B	Node C_{in}
Approximation 1	12	15	13
Approximation 2	12	12	8
Approximation 3	8	11	8
Approximation 4	12	8	9
Approximation 5	4	8	0

C_{gp} be the gate capacitance of a minimum size nMOS and pMOS transistor, respectively. Similarly, let C_{dn} and C_{dp} be the drain diffusion capacitances respectively. If the pMOS transistor has three times the width of the nMOS transistor, then $C_{gp} \approx 3C_{gn}$ and $C_{dp} \approx 3C_{dn}$. Let us also assume that $C_{dn} \approx C_{gn}$. In a multilevel adder tree, the Sum bits of intermediate outputs become the input bits A and B for the subsequent adder level. The output capacitance at each Sum node is $C_{dn} + C_{dp}$. The schematic of the conventional MA in Fig. 1 can be used to calculate the input capacitances at nodes A , B and C_{in} . Thus, the total capacitance at node A can be written as $(C_{dn} + C_{dp}) + 4(C_{gn} + C_{gp}) \approx 20C_{gn}$. Similarly, the total capacitance at node B is $(C_{dn} + C_{dp}) + 4(C_{gn} + C_{gp}) \approx 20C_{gn}$, while the capacitance at node C_{in} is $(C_{dn} + C_{dp}) + 3(C_{gn} + C_{gp}) \approx 16C_{gn}$. Continuing this way, the total capacitances at nodes A , B and C_{in} for all approximations can be calculated using their transistor-level schematics. Table V shows these values (normalized with respect to C_{gn}). Note that $C_{in}[1], C_{in}[2], \dots, C_{in}[y-1]$ are not calculated in approximation 5 (if y bits are approximate). Hence, the capacitance at node C_{in} for approximation 5 is zero. Henceforth, we will use normalized capacitances for all our subsequent discussions.

In the adder tree shown in Fig. 7, the inputs a , c , and e correspond to the input “A,” while the inputs b , d , and f correspond to the input “B” in an RCA. Let C_{app} and C_{acc} be the capacitances at node C_{in} for an approximate FA and accurate FA, respectively. Similarly, we define S_{app} and S_{acc} to be the capacitances at Sum node. S_{app} and S_{acc} are defined as mean of the capacitances at corresponding nodes A and B . For example, $S_{acc} = 20$, while $S_{app} = 13.5$ for approximation 1. Consider y bits to be approximate in an RCA of bit width N . Then the total capacitance C_{sw} that is switched due to bits Sum and C_{in} is given by

$$C_{sw} = (y-1)C_{app} + yS_{app} + (N-y)(C_{acc} + S_{acc}). \quad (4)$$

Next, we need to determine the scaled voltage that can be applied due to the approximation. Let the slack available due to approximation be p per bit. Since $V_{DD} \propto 1/\text{delay}$, the scaled voltage V_{DDapp} is given by

$$V_{DDapp} = V_{DD} \left(1 - \frac{yp}{T_c} \right). \quad (5)$$

Here, V_{DD} is the nominal voltage (accurate case), $T_c = 1/f_c$ is the clock period, and f_c is the operating frequency. The parameter p can be estimated by simulating an N -bit approximate RCA for different values of y , and using the above equation for fitting. The critical path in an adder tree is primarily dominated

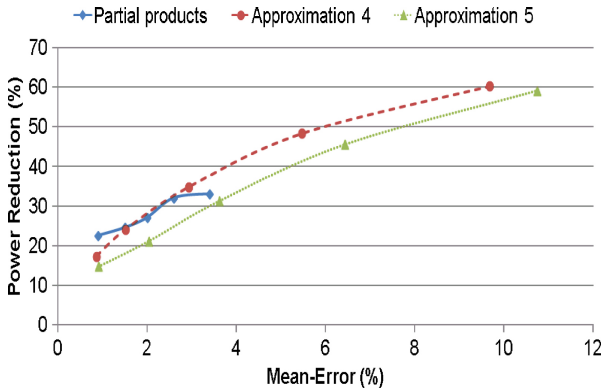


Fig. 16. Comparison with partial products approach.

by the carry chain of the last adder. Hence, the same value of p can be used for determining the scaled voltage as a function of y in an adder tree. Using the above equations, a first order estimate P_{app} for the power consumption of an approximate RCA can be written as $P_{\text{app}} = (1/2)C_{\text{sw}}V_{\text{DDapp}}^2f_c$, which is a function of y .

C. Comparison With Approximate Multipliers

Approximate multipliers based on Karnaugh map simplification have been proposed in [13]. There the errors are introduced via partial products. We constructed an 8×8 multiplier using our approximate FA cells. The accumulation of partial products (accurate) was done using a carry-save tree using an 8:2 compressor (approximate) followed by an RCA (approximate). We considered five cases, where 5–9 LSBs were kept approximate. The power consumption was calculated by running exhaustive nanosim simulations for both accurate and approximate 8×8 multipliers. The mean errors for the respective cases were obtained using MATLAB. The error-power tradeoff for the proposed approximate multiplier was compared with the data in [13]. Fig. 16 shows the results, where we plot percentage power reduction versus mean error. We find that approximation 4 performs better than the partial product approach for larger mean errors. Also, the partial product approach reaches the saturation point faster than approximations 4 and 5.

V. QUALITY CONSTRAINED SYNTHESIS OF DSP SYSTEMS

In this section, we consider synthesis of common DSP blocks using the proposed approximate adders. The outputs of most DSP blocks can be expressed as linear chains of adders, i.e., consisting of shifts and additions only. Since the final output in most DSP applications is resilient to errors in intermediate blocks, such blocks can be synthesized using approximate adders. The DCT block used in image compression is a good example. In DCT, all outputs do not contribute equally to the quality of the image obtained after decoding. Since the human eye is more sensitive to low frequencies, low-frequency terms in DCT outputs are more significant compared to high-frequency terms. Therefore, high-frequency terms can tolerate errors of greater magnitude compared to low-frequency terms.

There is a notion of “significance” in most DSP blocks, where some outputs are more significant compared to others [7], [8], [30]. This implies that the outputs that are less significant can use adders with more approximate LSBs, as compared to those which are more significant. This leads us to a design problem where it is required to find out how many bits to approximate for the calculation of each output. The goal of the design problem is to minimize the total power dissipation, while meeting an error/quality constraint. In the next section, we define a quality metric that takes into account the significance of different outputs.

A. Quantifying Error in Approximate Outputs of a DSP Block

Based on the discussion in previous paragraphs, we formulate a quality metric for the outputs of a DSP block using approximate adders. Suppose a DSP block has n outputs. Let the errors in the outputs be $\epsilon_1(y_0)$, $\epsilon_2(y_1)$, $\epsilon_3(y_2)$, \dots , $\epsilon_n(y_{n-1})$. Here, y_0, y_1, \dots, y_{n-1} are the number of approximate LSBs in the calculation of the respective outputs. We assign a significance factor λ_i , $i = 0, 1, \dots, (n-1)$ for each output, depending on its contribution to the final output quality of the DSP application. We define our quality metric Q as follows:

$$Q = \sum_{i=0}^{n-1} \lambda_i E[\epsilon_i(y_i)]. \quad (6)$$

Thus, Q is defined as the sum of mean errors weighted by their significance factor. Here we assume that each output is expressed only using shifts and additions, which can be done in most cases (multiplication can be converted to shifts and additions). Therefore, if a_i is the number of approximate adders used to calculate the i th output, then $E[\epsilon_i(y_i)] = \sum_{j=1}^{a_i} \beta_j \mu(y_i)$. Here, $\mu(y_i)$ is the mean error in each approximate addition (discussed in Section IV-A), and the coefficients β_j depend on the values of intermediate shifts. As derived in Section IV-B, the power consumption P_i for each output is also a function of y_i , and can be approximately written as $P_i \approx a_i P_{\text{app}}$, where P_{app} is the power consumption of a single approximate RCA. The design problem can then be stated as follows:

$$\text{minimize } \sum_{i=0}^{n-1} P_i(y_i)$$

$$\text{subject to } Q_{\min} \leq Q(y_0, y_1, \dots, y_{n-1}) \leq Q_{\max}.$$

We illustrate the solution to this design problem with the help of two examples, DCT and low-pass FIR filter, which are described in the next section. We target these blocks owing to their extensive use in several applications in the DSP domain. For example, DCT is widely used in image compression, video compression, face recognition, and digital watermarking. FIR filters are used in hearing aids, digital video broadcast, digital video effects, and digital wireless communications.

B. Design Examples

As discussed in Section IV-B, the scaled supply voltage depends on the number of approximate LSBs. Since using a large number of different supply voltages is not feasible in a design, we allow a maximum of three different supply voltages.

TABLE VI
 COSINE COEFFICIENTS FOR DCT

Coefficient	Value	Implementation
a	64	$1 \lll 6$
b	60	$(1 \lll 6) - (1 \lll 2)$
c	56	$(1 \lll 6) - (1 \lll 3)$
d	45	$1 + (1 \lll 2) + (1 \lll 3) + (1 \lll 5)$
e	36	$(1 \lll 2) + (1 \lll 5)$
f	24	$(1 \lll 3) + (1 \lll 4)$
g	12	$(1 \lll 2) + (1 \lll 3)$

1) *DCT*: Here we consider DCT in the context of JPEG image compression. Two-dimensional DCT of an 8×8 input block X is given by $Z = C(CX)^T$, where C is the coefficient matrix consisting of cosine terms [31]. As it is clear from the expression for Z , 2-D DCT can be broken down into two 1-D DCT operations. One-dimensional DCT outputs w_i , $i = 0, 1, \dots, 7$ are given by

$$\begin{aligned}
 w_0 &= dx_0 + dx_1 + dx_2 + dx_3 + dx_4 + dx_5 + dx_6 + dx_7 \\
 w_1 &= ax_0 + cx_1 + ex_2 + gx_3 - gx_4 - ex_5 - cx_6 - ax_7 \\
 w_2 &= bx_0 + fx_1 - fx_2 - bx_3 - bx_4 - fx_5 + fx_6 + bx_7 \\
 w_3 &= cx_0 - gx_1 - ax_2 - ex_3 + ex_4 + ax_5 + gx_6 - cx_7 \\
 w_4 &= dx_0 - dx_1 - dx_2 + dx_3 + dx_4 - dx_5 - dx_6 + dx_7 \\
 w_5 &= ex_0 - ax_1 + gx_2 + cx_3 - cx_4 - gx_5 + ax_6 - ex_7 \\
 w_6 &= fx_0 - bx_1 + bx_2 - fx_3 - fx_4 + bx_5 - bx_6 + fx_7 \\
 w_7 &= gx_0 - ex_1 + cx_2 - ax_3 + ax_4 - cx_5 + ex_6 - gx_7.
 \end{aligned}$$

The 8-b cosine coefficients are shown in Table VI.

In JPEG compression, quantization of 2-D DCT outputs is done before encoding them. This process accounts for the fact that the human visual system is more sensitive to low frequencies. The quantization process for a DCT coefficient involves division by an integer followed by rounding. These integer divisors are specified in a standard JPEG quantization table [32]. Each 1-D DCT output has a mean error $E(\epsilon_i)$, $i = 0, 1, \dots, 7$. These errors are propagated to the 2-D DCT outputs. The significance factor λ_i for each 1-D DCT output is determined as follows. The mean errors $E(\epsilon_i)$ are propagated to 2-D DCT outputs and an error matrix is generated. The components of this error matrix are weighted by the reciprocal of integer divisors in the quantization table and the mean error for the whole DCT output matrix is calculated. The mean error expression is a linear sum of the mean errors $E(\epsilon_i)$ multiplied by certain coefficients, which are used as significance factors in the expression for Q .

From the implementation given in Table VI, the outputs w_0 and w_4 require 31 adders in total; w_2 and w_6 require 15 adders, while outputs w_1 , w_3 , w_5 , and w_7 require 13 adders. Since we allow three different supply voltages, w_0 and w_4 have l_1 approximate LSBs, w_1 , w_3 , w_5 , and w_7 have l_2 approximate LSBs, while w_2 and w_6 have l_3 approximate LSBs. The allowable range for l_1 , l_2 , and l_3 is fixed to be $2 \leq l_i \leq 10$, $i = 1, 2, 3$. These ranges can be decided based on their effects on the final output image quality obtained after decoding. Thus, Q is a function of l_1 , l_2 , and l_3 . Similarly, the

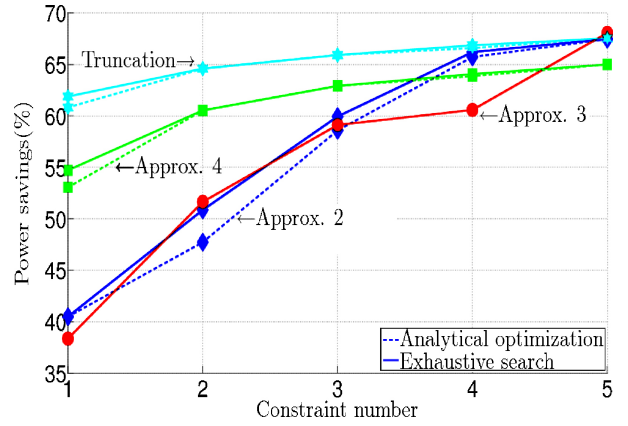
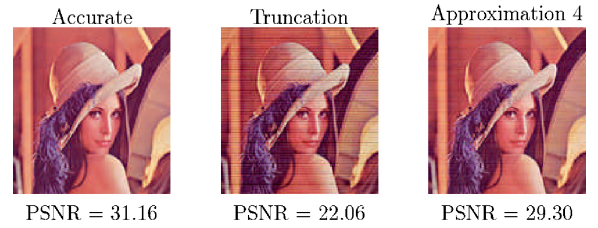


Fig. 17. Comparison between analytical optimization and exhaustive search.


 Fig. 18. Output quality for $-6 \leq Q \leq 0$.

total power consumption is also a function of l_1 , l_2 , and l_3 (Section V-A). Thus, the design problem mentioned in Section V-A can be solved to yield the optimum values of l_1 , l_2 , and l_3 that yield minimum power dissipation subject to a constraint on Q . This problem has been solved using MATLAB. The results for different approximations are shown next.

The output quality of the decoded image after IDCT is measured in terms of PSNR. The PSNR for the nominal case using accurate RCAs is 31.16 dB. The percentage power savings is calculated between the case using approximate RCAs and the nominal case using accurate RCAs. In order to test the efficacy of the power model introduced in Section IV-B, we obtain the simulated power dissipation for all triplets l_1 , l_2 , and l_3 satisfying a particular constraint on Q . This is done by running nanosim on 12 288 vectors from the standard image *Lena* in IBM 90-nm technology. The operating frequency is set to 333.33 MHz. The triplet that gives the minimum power is then determined using exhaustive search. Another optimum triplet $[l_1, l_2, l_3]$ (see Table VII) is obtained by solving the design problem using the power model. Power savings is calculated using both approaches and compared. The results are shown in Table VII. Clearly, the power savings obtained from analytical optimization using the power model agree closely with that obtained by exhaustive search for most cases. These trends are also shown in Fig. 17 for the five constraints on Q .

Fig. 18 shows the output images for the constraint $-6 \leq Q \leq 0$. Truncation provides more power savings (65.9%) compared to approximation 4 (62.9%) for the same constraint. However, truncation results in blocky output images with PSNR loss of 7.24 dB compared to approximation 4, as is clear from Fig. 18.

The mean error due to approximation 1 is zero, while that due to approximation 5 is 0.5 (see the Appendix). Thus,

TABLE VII
QUALITY AND POWER RESULTS FOR DCT

Approximation Technique	Range of Q	Optimum $[l_1, l_2, l_3]$	PSNR (dB)	Power Savings (%) – Power Model	Power Savings (%) – Exhaustive Search
Approximation 1	0	[10, 10, 10]	25.94	57.08	57.08
Approximation 2	[0.09, 0.1]	[2, 3, 2]	31.34	40.48	40.48
	[0.09, 0.2]	[5, 5, 4]	31.30	47.72	50.85
	[0.09, 0.3]	[6, 10, 5]	29.13	58.58	59.93
	[0.09, 0.4]	[10, 10, 8]	20.89	65.71	66.18
	[0.09, 0.5]	[5, 5, 4]	20.73	67.45	67.45
Approximation 3	[0.126, 0.13]	[2, 2, 2]	31.33	38.35	38.35
	[0.126, 0.14]	[2, 10, 2]	28.09	51.64	51.64
	[0.126, 0.15]	[2, 10, 10]	27.20	59.11	59.11
	[0.126, 0.16]	[10, 10, 2]	19.56	60.57	60.57
	[0.126, 0.17]	[10, 10, 10]	19.2	68.04	68.04
Approximation 4	[-2, 0]	[7, 6, 6]	30.78	53.05	54.71
	[-4, 0]	[8, 7, 7]	29.74	60.52	60.52
	[-6, 0]	[8, 8, 8]	29.30	62.9	62.9
	[-8, 0]	[9, 8, 8]	26.94	63.85	64.05
	[-10, 0]	[9, 9, 8]	26.12	64.99	64.99
Approximation 5	0.021	[10, 10, 10]	25.30	69.32	69.32
Truncation	[-2, 0]	[7, 6, 6]	27.29	60.81	61.87
	[-4, 0]	[8, 7, 7]	22.21	64.58	64.58
	[-6, 0]	[8, 8, 8]	22.06	65.9	65.9
	[-8, 0]	[9, 8, 8]	16.61	66.58	66.84
	[-10, 0]	[9, 9, 8]	16.36	67.53	67.53

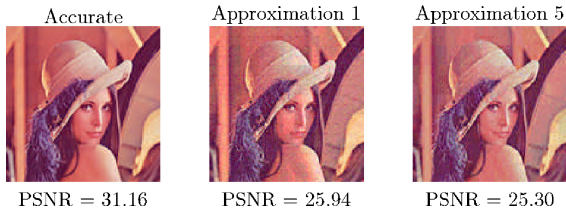


Fig. 19. Output quality for $l_1 = 10$, $l_2 = 10$, and $l_3 = 10$.

$Q = 0$ for approximation 1 and $Q = 0.021$ for approximation 5. Since Q is not a function of l_1 , l_2 , and l_3 for these approximations, we set them to their maximum allowed values, i.e., $l_1 = 10$, $l_2 = 10$, $l_3 = 10$. This provides maximum power savings for approximations 1 and 5. The output images for these approximations are shown in Fig. 19. Approximation 5 provides maximum power savings of 69.32% with satisfactory output image quality. In general, approximation 4 is the right choice when Q is negative. For positive values of Q , the suitable choices are approximation 1, approximation 5, approximation 3, and approximation 2, in that order. This is in accordance with the trends of mean error $\mu(y)$ for large values of y (see the Appendix).

2) *FIR Filter*: We consider a 25 tap low-pass equiripple FIR filter with the following specifications: $F_s = 48\,000\text{ Hz}$, $F_{\text{pass}} = 10\,000\text{ Hz}$, $F_{\text{stop}} = 12\,000\text{ Hz}$. The coefficients of this filter c_0, \dots, c_{24} were obtained using the MATLAB FDA tool. We consider the implementation of the filter in transposed form [30], where the coefficients are synthesized in the multiplier block. The 25 coefficients are converted to their equivalent 16-b fixed point representations. The FIR filter input x is also assumed to be 16b. According to the terminology mentioned in Section V-A, $n = 25$ in this case.

TABLE VIII
SORTED COEFFICIENTS FOR FIR FILTER

Coefficient	Absolute Value	Fixed Point
c_1	0.0017219515241194	0000000000111000
c_3	0.0116238118940803	0000000101111100
c_5	0.0231495073798233	0000001011110110
c_2	0.0238666379355453	0000001100001110
c_4	0.0277034324942943	0000001110001011
c_6	0.0347323795277931	0000010001110010
c_0	0.0369760641356146	0000010010111011
c_8	0.0371467378315251	0000010011000001
c_{10}	0.0415921203506734	0000010101010010
c_7	0.0479147131729558	0000011000100010
c_9	0.0946865327578015	0000110000011110
c_{11}	0.3155493020793390	0010100001100011
c_{12}	0.4591823044074993	0011101011000110

The significance factors $\lambda_i, i = 0, 1, \dots, 24$ are determined by the process mentioned in [30]. Each filter coefficient is set to zero one at a time, and the filter response (maximum pass-band and stop-band ripple) is calculated using the new set of coefficients. It is observed that the degradation in filter response follows the magnitude of the coefficients. The greater the magnitude, the more the degradation in maximum pass-band and stop-band ripple when that coefficient is set to zero. Based on this sensitivity analysis, the coefficients arranged in ascending order of magnitude are shown in Table VIII.

This is also the order of contribution of the coefficients to the filter output quality. Therefore, the significance factors $\lambda_i, i = 0, 1, \dots, 24$ are set to the absolute value of the coefficients. Since the filter is symmetric, only coefficients c_0 to c_{12} need to be synthesized. Since we allow three supply voltages, we divide the 13 coefficients into three sets. The first five coefficients in Table VIII form the first set, the next four the second set, while the last four form the third set. The

TABLE IX
QUALITY AND POWER RESULTS FOR FIR FILTER

Approximation Technique	Range of Q	Optimum $[l_1, l_2, l_3]$	Δ MPBR (%)	Δ MSBR (%)	Power Savings (%) – Power Model	Power Savings (%) – Exhaustive Search
Approximation 1	0	[10, 10, 10]	2.08	1.48	51.97	51.97
Approximation 2	[3.3, 6.6]	[10, 9, 3]	6.01	9.31	37.22	38.34
	[3.3, 9.2]	[10, 10, 4]	6.08	12.8	40.13	40.13
	[3.3, 11.8]	[10, 10, 6]	5.93	12.7	44.76	44.76
	[3.3, 14.4]	[10, 10, 8]	7.28	12.53	47.88	47.88
	[3.3, 17]	[10, 10, 10]	13.43	12.2	50.85	50.86
Approximation 3	[4.9, 5.3]	[10, 10, 2]	9.69	18.27	37.5	37.5
	[4.9, 5.6]	[10, 10, 2]	9.69	18.27	37.5	37.5
	[4.9, 5.9]	[10, 10, 3]	9.66	18.27	42.95	42.95
	[4.9, 6.2]	[10, 10, 4]	9.57	18.26	41.65	42.95
	[4.9, 6.5]	[10, 10, 10]	17.93	18.18	55.14	55.14
Approximation 4	[-200, 0]	[10, 9, 7]	1.35	1.77	41.00	41.90
	[-350, 0]	[10, 10, 8]	2.23	3.19	44.55	44.55
	[-500, 0]	[10, 10, 9]	2.73	3.11	47.73	47.73
	[-650, 0]	[10, 10, 9]	2.73	3.11	47.73	47.73
	[-800, 0]	[10, 10, 10]	1.99	1.82	47.07	47.73
Approximation 5	3.24	[10, 10, 10]	0.27	1.92	61.77	61.77
Truncation	[-200, 0]	[7, 6, 4]	1.03	1.5	32.31	35.66
	[-350, 0]	[8, 6, 5]	2.69	2.08	38.08	42.08
	[-500, 0]	[9, 7, 5]	6.00	4.55	44.49	46.06
	[-650, 0]	[9, 7, 6]	6.21	4.53	46.21	48.26
	[-800, 0]	[9, 8, 6]	6.38	6.28	49.09	50.67

coefficients are synthesized using the shift-and-add approach based on their fixed-point representations.

The number of approximate LSBs for the three sets are assumed to be l_1 , l_2 , and l_3 , respectively. The allowable range of l_1 , l_2 , and l_3 is kept same as in the DCT example, i.e., $2 \leq l_i \leq 10, i = 1, 2, 3$. The design problem mentioned in Section V-A is solved to obtain the optimum values of l_1 , l_2 , and l_3 . In the FIR filter, the outputs $c_0x, c_1x, \dots, c_{12}x$ depend on the input x . When we use approximate adders in synthesis, we calculate the equivalent approximate coefficients $c'_0, c'_1, \dots, c'_{12}$. We calculate the respective approximate outputs for a particular input x and then divide by x to calculate one instance of approximate coefficients. This process is repeated for 1000 random inputs and the mean is taken to calculate the equivalent approximate coefficients. The pass-band and stop-band ripples are calculated for the FIR filter using the equivalent approximate coefficients. The output quality is thus measured as the percentage change in pass-band and stop-band ripples compared to the case using original coefficients.

The results are shown in Table IX. The operating frequency is set to 300 MHz. Δ MPBR and Δ MSBR denote the percentage changes in maximum pass-band and stop-band ripples. As the range of Q increases on the negative side, the percentage change in maximum pass-band and stop-band ripples increases for truncation. Thus, approximation 4 is better than truncation for such cases. Approximation 5 provides maximum power savings of 61.77% among all approximations with minimum percentage change in maximum pass-band ripple. Approximation 1 has the least percentage change in maximum stop-band ripple. As discussed in the DCT case, approximation 1 is best for $Q = 0$. For increasing positive values of Q , the suitable choices are approximation 5, approximation 3, and approximation 2, in that order. This trend is same as in the DCT example.

VI. CONCLUSION

In this paper, we proposed several imprecise or approximate adders that can be effectively utilized to trade off power and quality for error-resilient DSP systems. Our approach aimed to simplify the complexity of a conventional MA cell by reducing the number of transistors and also the load capacitances. When the errors introduced by these approximations were reflected at a high level in a typical DSP algorithm, the impact on output quality was very little. Note that our approach differed from previous approaches where errors were introduced due to VOS [3]–[10]. A decrease in the number of series connected transistors helped in reducing the effective switched capacitance and achieving voltage scaling. We also derived simplified mathematical models for error and power consumption of an approximate RCA using the approximate FA cells. Using these models, we discussed how to apply these approximations to achieve maximum power savings subject to a given quality constraint. This procedure has been illustrated for two examples, DCT and FIR filter. We believe that the proposed approximate adders can be used on top of already existing low-power techniques like SDC and ANT to extract multifold benefits with a very minimal loss in output quality.

APPENDIX

DERIVATION OF MEAN ERROR FOR AN APPROXIMATE RCA

As mentioned in Section IV-A, the expression for mean error is given by

$$\begin{aligned} \mu(y) &= E(\epsilon) \\ &= \sum_{x=0}^y 2^x E(e[x]). \end{aligned}$$

Clearly, $e[x] \in \{-1, 0, 1\}$. Thus, $E(e[x]) \forall x \in \{0, 1, \dots, y\}$ can be calculated as follows:

$$\begin{aligned}
E(e[x]) &= P(e[x] = 1) - P(e[x] = -1) \\
&= P(\{\text{Sum}'[x] = 1\} \cap \{\text{Sum}[x] = 0\}) \\
&\quad - P(\{\text{Sum}'[x] = 0\} \cap P\{\text{Sum}[x] = 1\}) \\
&= s'_x(1 - s_x) - s_x(1 - s'_x) \\
&= s'_x - s_x \\
&= s'_x - \frac{1}{2}, \quad x \in \{0, 1, \dots, y-1\} \\
E(e[y]) &= P(\{C'_{in}[y] = 1\} \cap \{C_{in}[y] = 0\}) \\
&\quad - P(\{C'_{in}[y] = 0\} \cap \{C_{in}[y] = 1\}) \\
&= c'_y - c_y \\
&= c'_y - \left(\frac{1}{2} - \frac{1}{2^{y+1}}\right).
\end{aligned}$$

Expressions for s'_x and c'_y were derived in Section IV-A. Thus, the mean error for the proposed approximations and truncation can be calculated as follows.

A. Approximation 1

$$\begin{aligned}
\mu(y) &= -\frac{1}{6} \sum_{x=0}^{y-1} 2^x - \frac{1}{3} \sum_{x=0}^{y-1} \frac{1}{2^x} + \left(\frac{1}{6} - \frac{1}{3 \cdot 2^{2y-1}} + \frac{1}{2^{y+1}}\right) 2^y \\
&= -\frac{1}{6}(2^y - 1) - \frac{2}{3}(1 - 2^{-y}) + \frac{2^y}{6} - \frac{1}{3 \cdot 2^{y-1}} + \frac{1}{2} \\
&= 0.
\end{aligned}$$

B. Approximation 2

$$\begin{aligned}
\mu(y) &= \sum_{x=0}^{y-1} \frac{1}{2^{x+2}} \times 2^x + 0 \\
&= \frac{y}{4}.
\end{aligned}$$

C. Approximation 3

$$\begin{aligned}
\mu(y) &= -\frac{1}{6} \sum_{x=0}^{y-1} 2^x + \frac{1}{3} \sum_{x=0}^{y-1} \frac{1}{2^{x+1}} \\
&\quad + \left(\frac{1}{6} - \frac{1}{3 \cdot 2^{2y-1}} + \frac{1}{2^{y+1}}\right) 2^y \\
&= -\frac{1}{6}(2^y - 1) + \frac{1}{3}(1 - 2^{-y}) + \frac{2^y}{6} - \frac{1}{3 \cdot 2^{y-1}} + \frac{1}{2} \\
&= 1 - 2^{-y}.
\end{aligned}$$

D. Approximation 4

$$\begin{aligned}
\mu(y) &= -\frac{1}{2} - \frac{1}{8} \sum_{x=1}^{y-1} 2^x + \frac{1}{2} \times \frac{2}{2^{y+1}} \times 2^y \\
&= \frac{1 - 2^{y-1}}{4}.
\end{aligned}$$

E. Approximation 5

$$\begin{aligned}
\mu(y) &= 0 + \frac{1}{2^{y+1}} \times 2^y \\
&= \frac{1}{2}.
\end{aligned}$$

F. Truncation

$$\begin{aligned}
\mu(y) &= -\frac{1}{2} \sum_{x=0}^{y-1} 2^x + \left(\frac{1}{2} - \frac{1}{2^{y+1}}\right) 2^y \\
&= 1 - 2^y.
\end{aligned}$$

REFERENCES

- [1] W. Dally, J. Balfour, D. Black-Shaffer, J. Chen, R. Harting, V. Parikh, J. Park, and D. Sheffield, "Efficient embedded computing," *Computer*, vol. 41, no. 7, pp. 27–32, Jul. 2008.
- [2] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th IEEE Int. Conf. VLSI Design*, Jan. 2011, pp. 346–351.
- [3] R. Hegde and N. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Aug. 1999, pp. 30–35.
- [4] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 6, pp. 813–823, Jun. 2001.
- [5] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 5, pp. 497–510, May 2004.
- [6] G. Varatkar and N. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Oct. 2006, pp. 113–118.
- [7] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Aug. 2009, pp. 195–200.
- [8] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in *Proc. Design, Automat. Test Eur.*, 2007, pp. 1–6.
- [9] G. Karakonstantis, D. Mohapatra, and K. Roy, "System level DSP synthesis using voltage overscaling, unequal error protection and adaptive quality tuning," in *Proc. IEEE Workshop Signal Processing Systems*, Oct. 2009, pp. 133–138.
- [10] L. N. Chakrapani, K. K. Muntimadugu, L. Avinash, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation," in *Proc. CASES*, 2008, pp. 187–196.
- [11] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Automat. Test Eur.*, 2008, pp. 1250–1255.
- [12] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. IEEE Int. Symp. Integr. Circuits*, Dec. 2009, pp. 69–72.
- [13] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electron.*, vol. 7, no. 4, pp. 490–501, 2011.
- [14] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Proc. Design, Automat. Test Eur.*, 2010, pp. 957–960.
- [15] B. J. Phillips, D. R. Kelly, and B. W. Ng, "Estimating adders for a low density parity check decoder," *Proc. SPIE*, vol. 6313, p. 631302, Aug. 2006.
- [16] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. Part I*, vol. 57, no. 4, pp. 850–862, Apr. 2010.
- [17] D. Shin and S. K. Gupta, "A re-design technique for datapath modules in error tolerant applications," in *Proc. 17th Asian Test Symp.*, 2008, pp. 431–437.
- [18] D. Kelly and B. Phillips, "Arithmetic data value speculation," in *Proc. Asia-Pacific Comput. Syst. Architect. Conf.*, 2005, pp. 353–366.
- [19] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, Mar. 2004.

- [20] Y. V. Ivanov and C. J. Bleakley, "Real-time h.264 video encoding in software with fast mode decision and dynamic complexity control," *ACM Trans. Multimedia Comput. Commun. Applicat.*, vol. 6, pp. 5:1–5:21, Feb. 2010.
- [21] M. Shafique, L. Bauer, and J. Henkel, "enBudget: A run-time adaptive predictive energy-budgeting scheme for energy-aware motion estimation in H.264/MPEG-4 AVC video encoder," in *Proc. Design, Automat. Test Eur.*, Mar. 2010, pp. 1725–1730.
- [22] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: Imprecise adders for low-power approximate computing," in *Proc. IEEE/ACM Int. Symp. Low-Power Electron. Design*, Aug. 2011, pp. 409–414.
- [23] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ: Prentice-Hall, 1996.
- [24] E. Lyons, V. Ganti, R. Goldman, V. Melikyan, and H. Mahmoodi, "Full-custom design project for digital VLSI and IC design courses using synopsys generic 90nm CMOS library," in *Proc. IEEE Int. Conf. Microelectron. Syst. Edu.*, Jul. 2009, pp. 45–48.
- [25] G. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.
- [26] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.
- [27] G. Ying. (2012). *Nanosim: A Next-Generation Solution for SoC Integration Verification* [Online]. Available: <http://www.synopsys.com/Tools/Verification/Pages/socintegration.aspx>
- [28] P. M. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, 1st ed. Norwell, MA: Kluwer, 1999.
- [29] K. Parker and E. McCluskey, "Probabilistic treatment of general combinatorial networks," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 668–670, Jun. 1975.
- [30] J. Choi, N. Banerjee, and K. Roy, "Variation-aware low-power synthesis methodology for fixed-point FIR filters," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 87–97, Jan. 2009.
- [31] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Norwell, MA: Kluwer, 1997.
- [32] International Telecommunication Union. (1992). *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines* [Online]. Available: <http://www.w3.org/Graphics/JPEG/itu-t81.pdf>



Vaibhav Gupta received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2008, and the M.S. degree from Purdue University, West Lafayette, IN, in 2012, both in electrical engineering.

His current research interests include low-power approximate computing for communication and signal processing applications.

Mr. Gupta was a recipient of the Third Prize in the Altera's Innovate North America Design Contest 2010.



Debabrata Mohapatra received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2005, and the Ph.D. degree from Purdue University, West Lafayette, IN, in April 2011.

Since 2011, he has been a Research Scientist with the Microarchitecture Research Laboratory, Intel Corporation, Santa Clara, CA. His current research interests include design of low-power and process-variation-aware hardware for error-resilient applications.



Anand Raghunathan (F'12) received the B.Tech. degree in electrical and electronics engineering from the Indian Institute of Technology, Madras, India, in 1992, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1994 and 1997, respectively.

He is currently a Professor with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN. He was a Senior Research Staff Member with NEC Laboratories America, Princeton, where he led research projects related

to system-on-chip architectures, design methodologies, and design tools. He was the co-author of a book entitled *High-Level Power Analysis and Optimization* (A. Raghunathan, N. K. Jha, and S. Dey, Kluwer Academic, Norwell, MA, 1998) and eight book chapters. He holds 21 U.S. patents. He has presented several full-day and embedded conference tutorials.

Dr. Raghunathan was a recipient of the IEEE Meritorious Service Award in 2001 and the Outstanding Service Award in 2004. He was a recipient of eight Best Paper Awards and four Best Paper Nominations at leading conferences. He received the Patent of the Year Award (an award recognizing the invention that has achieved the highest impact) and two Technology Commercialization Awards from NEC. He was chosen by MIT's Technology Review among the TR35 (top 35 innovators under 35 years, across various disciplines of science and technology) in 2006, for his work on "making mobile secure." He has been a member of the technical program and organizing committees of several leading conferences and workshops. He was the Program and General Co-Chair of the ACM/IEEE International Symposium on Low Power Electronics and Design, the IEEE VLSI Test Symposium, and the IEEE International Conference on VLSI Design. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the *ACM Transactions on Design Automation of Electronic Systems*, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the *ACM Transactions on Embedded Computing Systems*, the *IEEE Design and Test of Computers*, and the *Journal of Low Power Electronics*. He is a Golden Core Member of the IEEE Computer Society.



Kaushik Roy (F'12) received the B.Tech. degree in electronics and electrical communications engineering from the Indian Institute of Technology (IIT) Kharagpur, Kharagpur, India, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, in 1990.

He was with the Semiconductor Process and Design Center of Texas Instruments, Dallas, TX, where he worked on field-programmable gate array architecture development and low-power circuit design.

He joined the Electrical and Computer Engineering Faculty at Purdue University, West Lafayette, IN, in 1993, where he is currently a Professor and holds the Roscoe H. George Chair of the Electrical and Computer Engineering Department. He has published more than 600 papers in refereed journals and conference proceedings. He holds 15 patents. He guided 56 Ph.D. students. He is the co-author of two books on low power CMOS VLSI design. His current research interests include spintronics, device-circuit codesign for nanoscale silicon and non-silicon technologies, low-power electronics for portable computing and wireless communications, and new computing models enabled by emerging technologies.

Dr. Roy was a recipient of the National Science Foundation Career Development Award in 1995, the IBM Faculty Partnership Award, the ATT/Lucent Foundation Award, the 2005 SRC Technical Excellence Award, the SRC Inventors Award, the Purdue College of Engineering Research Excellence Award, the Humboldt Research Award in 2010, the IEEE Circuits and Systems Society Technical Achievement Award in 2010, the Distinguished Alumnus Award from IIT Kharagpur, and the Best Paper Award at the 1997 International Test Conference, the IEEE 2000 International Symposium on Quality of IC Design, the 2003 IEEE Latin American Test Workshop, the 2003 IEEE Nano, the 2004 IEEE International Conference on Computer Design, the 2006 IEEE/ACM International Symposium on Low Power Electronics and Design, and the 2005 IEEE Circuits and System Society Outstanding Young Author Award (with C. Kim). He also was the recipient of the 2006 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS Best Paper Award and the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design Best Paper Award. From 1998 to 2003, he was a Purdue University Faculty Scholar. He was a Research Visionary Board Member of the Motorola Laboratories in 2002. He was also the M.K. Gandhi Distinguished Visiting Faculty at IIT Bombay, Mumbai, India. He was an Editorial Board Member of the *IEEE Design and Test of Computers*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, and the IEEE TRANSACTIONS ON ELECTRON DEVICES. He was the Guest Editor of the special issue on low-power VLSI in the *IEEE Design and Test of Computers* in 1994, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS in June 2000, the *IEEE Proceedings: Computers and Digital Techniques* in July 2002, and the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS in 2011.