IMPACT: IMPrecise adders for low-power Approximate CompuTing

Vaibhav Gupta, Debabrata Mohapatra, Sang Phill Park, Anand Raghunathan and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
Email: {gupta64,dmohapat,sppark,raghunathan,kaushik}@purdue.edu

***Abstract*—Low-power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, the final output is interpreted by human senses, which are not perfect. This fact obviates the need to produce exactly correct numerical outputs. Previous research in this context exploits error-resiliency primarily through voltage over-scaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. In this paper, we propose logic complexity reduction 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 (FA) 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. Post-layout simulations indicate power savings of up to 60% and area savings of up to 37% with an insignificant loss in output quality, when compared to existing implementations.**

***Index Terms*—Approximate computing, Low-power, Mirror adder.**

I. INTRODUCTION

Commonly used multimedia applications have Digital Signal Processing (DSP) blocks as their backbone. 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. The limited perception of human vision 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. The freedom can be taken advantage of to come up with low-power designs at different levels of design abstraction, viz. logic, architecture, and algorithm.

Few works which focus on low-power design through approximate computing at the algorithm and architecture levels include Algorithmic Noise Tolerance (ANT) [1], [2], [3], [4], Significance Driven Computation (SDC) [5], [6], [7] and non-uniform Voltage Over-Scaling (VOS) [8]. All these techniques are based on the central concept of VOS, coupled with additional circuitry for correcting or limiting the resulting errors. In [9], a fast but "inaccurate" adder is proposed. It is based on the idea that on an 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 [10] which operates by splitting the input operands into accurate and inaccurate parts. However, both these techniques do not target logic complexity reduction. A power-efficient multiplier architecture is proposed in [11] which uses a 2×2 inaccurate multiplier block resulting from Karnaugh Map simplification. This work does consider logic complexity reduction, but only focuses on a 2×2 multiplier, and does not consider adders as such. Other approaches use complexity reduction at the algorithm level to meet real-time energy constraints [12], [13]. We propose an approach where we apply logic complexity reduction 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 reduced number of transistors, but care is taken to ensure that the internal node capacitances are much 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 non-uniform 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, use them to design image and video processing systems, and highlight the potential benefits. Our contributions in this paper can be summarized as follows :-

- We propose logic complexity reduction as an alternative approach to approximate computing for signal (video and image) processing applications.
- 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 3 different simplified versions of the MA ensuring minimal errors in the Full Adder (FA) truth table.
- We utilize the simplified versions of the FA cell to propose several imprecise or approximate multi-bit adders which 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). The Most Significant Bits (MSBs) are ensured to be correct by using accurate FA cells for them. 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-A). Higher order compressors (using CSA trees) are also used to accumulate partial products in various tree multipliers [14]. 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.*
- We present designs for image and video compression algorithms using the proposed approximate adders and evaluate the approximate architectures in terms of output quality, power dissipation and area.
- Finally, in order to have a fair comparision, we compare our results with truncation, which is a well-known and simple

978-1-61284-660-6/11/$26.00 © 2011 IEEE

technique utilized in DSP systems to trade off output quality for power dissipation and area. Our results show that truncating the LSBs beyond a certain limit has drastic effects on the output quality of multimedia systems. The primary benefit of the proposed approximate adders is to maintain a reasonable output quality, and extract power and area savings almost at par with truncation.

The rest of the paper is organized as follows. In section II, we propose and discuss various approximate FA cells. In section III, we apply them to DSP systems and present the output quality, power and area results. In section IV, we compare the proposed approximations using a metric that considers quality, power and area. Section V concludes the paper.

II. APPROXIMATE FULL ADDER CELLS

In this section, we discuss different methodologies for designing approximate FA cells. Since the MA [14] is one of the widely used economical implementations of the FA, we use it as our basis for proposing different approximations of an FA cell.

A. Approximating the Mirror Adder

In this section, we discuss how we can come up with different approximate versions of the MA with lesser number of transistors. Since series connected transistors contribute to larger delay, removal of some of them 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 $P_{dynamic} = \alpha C V_{DD}^2 f$, where 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. Another benefit is reduced area. Now, let us discuss the conventional MA implementation followed by the proposed approximations.

1) Conventional Mirror Adder: Figure 1 shows the transistor level schematic of a conventional MA [14], which is a popular way of implementing an FA. It consists of a total of 24 transistors. Note that this implementation is not based on complementary CMOS logic, and thus provides an opportunity to cleverly design an approximate version with removal of selected transistors.

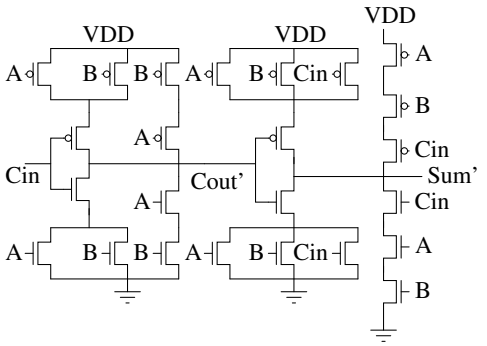


Fig. 1: Conventional MA

2) Approximation 1: In order to get an approximate MA with lesser transistors, we start to remove transistors from the conventional schematic one by one. In doing so, we need to ensure that any input combination of A, B and C_{in} does not result in short circuits or open circuits in the simplified schematic. We also impose another criterion 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 Figure 2. Clearly, this schematic has 8 less transistors compared to the conventional MA schematic.

A close observation of the truth table of an FA shows that $Sum = \overline{C_{out}}$ for 6 cases out of 8, 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 elegant way of simplifying the MA further is to discard the Sum circuit completely. Although one can directly set $Sum = \overline{C_{out}}$ as shown in Figure 1, we introduce a buffer stage after $\overline{C_{out}}$ (see Figure 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 in the conventional MA, the total capacitance at the Sum node would be a combination of 4 source-drain diffusion and 2 gate capacitances. This is an appreciable increase compared to the conventional case. Such a design would lead to a delay penalty in cases where two or more multi-bit approximate adders are connected in a chained fashion, which is a common scenario in DSP applications. Thus we combine the simplified circuit for $\overline{C_{out}}$ in Figure 2 with the idea that $Sum = \overline{C_{out}}$ for 6 cases out of 8. Figure 3 shows the simplified MA obtained using this technique. This introduces 1 error in C_{out} and 3 errors in Sum, as shown in Table I.

3) Approximation 2: Again, a careful observation of the FA truth table shows that $C_{out} = A$ for 6 cases out of 8. Similarly, $C_{out} = B$ for 6 cases out of 8. Since A and B are interchangeable, we consider $C_{out} = A$. Thus we propose a second approximation (approximation 2) where we just use an inverter with input A to calculate $\overline{C_{out}}$ and Sum is calculated similar to the simplified MA in Figure 2. Figure 4 shows the simplified circuit obtained using this technique. This introduces 2 errors in C_{out} and 3 errors in Sum, as shown in Table I.

In both approximations 1 and 2, C_{out} is calculated by using an inverter with $\overline{C_{out}}$ as input.

4) Approximation 3: In approximation 2, we find that there are 3 errors in Sum. We take this approximation a step further by allowing 1 more error, i.e., 4 errors in Sum. We also aim to reduce the dependency of Sum on C_{in} (to save area). This leaves us with 2 choices, $Sum = A$ and $Sum = B$. Also, we use the approximation $C_{out} = A$, as in approximation 2. Thus we have 2 choices for approximation 3, viz. $Sum = A, C_{out} = A$ and $Sum = B, C_{out} = A$. If we observe choice 1, we find that both Sum *and* C_{out} match with accurate outputs in only 2 out of the 8 cases. In choice 2, Sum *and* C_{out} match with accurate outputs in 4 out of the 8 cases. Therefore, to minimize errors both in Sum and C_{out}, we go for choice 2 as approximation 3. **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 bit integers a[19:0] and b[19:0] using a Ripple Carry Adder (RCA)[1]. 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 the approximation 3 $Sum = B, C_{out} = A$ using buffers.

[1]We only mention the ripple carry adder for ease of illustration. The proposed approach can be applied to any arithmetic circuit that is composed of full adder cells, including tree structures, such as carry-save trees considered in this paper.

TABLE I: Truth table for conventional full adder and approximations 1, 2 and 3

Inputs			Accurate outputs		Approximate outputs					
A	B	C_{in}	Sum	C_{out}	Sum_1	C_{out1}	Sum_2	C_{out2}	Sum_3	C_{out3}
0	0	0	0	0	1 ✗	0 ✓	0 ✓	0 ✓	0 ✓	0 ✓
0	0	1	1	0	1 ✓	0 ✓	1 ✓	0 ✓	0 ✗	0 ✓
0	1	0	1	0	0 ✗	1 ✗	0 ✗	0 ✓	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓	1 ✗	0 ✗	1 ✗	0 ✗
1	0	0	1	0	1 ✓	0 ✓	0 ✗	1 ✗	0 ✗	1 ✗
1	0	1	0	1	0 ✓	1 ✓	0 ✓	1 ✓	0 ✓	1 ✓
1	1	0	0	1	0 ✓	1 ✓	0 ✓	1 ✓	1 ✗	1 ✓
1	1	1	1	1	0 ✗	1 ✓	1 ✓	1 ✓	1 ✓	1 ✓

Layouts of conventional MA and different approximations in IBM 90nm technology are shown in Figures 5a, 5b, 5c and 5d. Table II compares their areas.

TABLE II: Layout area of approximate MA cells

MA cell	Area (μm^2)
Conventional	40.66
Approximation 1	22.56
Approximation 2	23.91
Approximation 3	13.54

B. Discussion on proposed approximations

Now we 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 6 gate capacitances (see Figure 1). Approximation 2 has only 3 such gate capacitances, which further reduces to only 2 gate capacitances in approximation 1. Now consider a section of a multi-level adder tree as shown in Figure 6. 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 the Sum bits of adder level $m-1$. The input capacitance at node A consists of 8 gate capacitances in the conventional case. This is reduced to 4 gate capacitances in approximation 2, and only 2 gate capacitances in approximation 1. Similarly, the corresponding values for node B are 8, 3 and 2 gate capacitances for the conventional case, approximations 1 and 2 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.*

Next, we apply these approximations to two DSP systems and present the analyses and results in the following sections.

III. Application to DSP Systems

In the previous section the impact of using approximate FA cells on the truth table of an FA was discussed. However, since most DSP algorithms used in multimedia systems have inherent error-resiliency, these occasional errors might not manifest as an

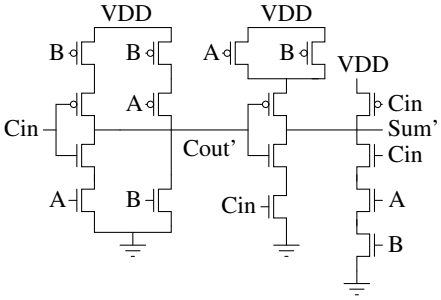


Fig. 2: Simplified MA

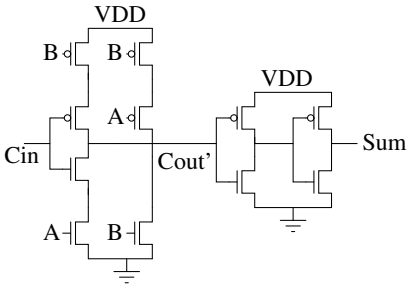


Fig. 3: MA approximation 1

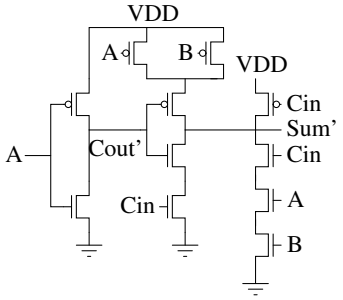
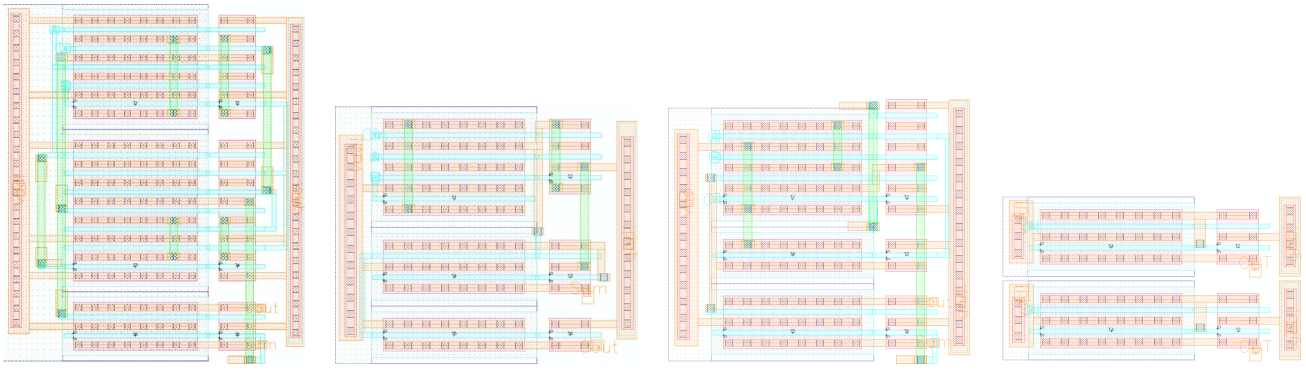


Fig. 4: MA approximation 2



(a) Conventional (b) Approximation 1 (c) Approximation 2 (d) Approximation 3

Fig. 5: Layouts of conventional and approximate MA cells

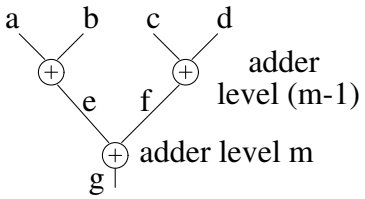


Fig. 6: Adder tree section

appreciable reduction in the final output quality. Multimedia DSP algorithms mostly consist of additions and multiplications, which use adders as basic building blocks. We focus on two algorithms, viz. image and video compression, and present the results of using our approximate FA cells in these algorithms.

A. Image Compression

The Discrete Cosine Transform (DCT) and Inverse Discrete Cosine Transform (IDCT) are integral components of a JPEG image compression system [15]. One-dimensional integer DCT $y(k)$ for an 8-point sequence $x(i)$ is given by [16]

$$y(k) \quad = \quad \sum_{i=0}^{7} a(k,i)x(i), \;\; k = 0, 1, \ldots, 7 \qquad (1)$$

Here $a(k,i)$ are cosine functions converted into equivalent integers [6]. The integer outputs $y(k)$ can then be right-shifted to get the actual DCT outputs. A similar expression can be found for one-dimensional integer IDCT [7]. We alter the integer coefficients $a(k,i), k = 1, \ldots, 7$ so that the multiplication $a(k,i)x(i)$ is converted to 2 left-shifts and an addition. 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 4 terms. This is done using a carry-save tree using 4:2 compressors followed by a merge adder. Also, each integer DCT and IDCT output is the sum of 8 terms. Thus these outputs are calculated using a carry-save tree using 8:2 compressors followed by a merge adder. Thus the whole DCT-IDCT system now consists of RCAs and CSA trees. In our design, all adders are replaced by approximate adders, which use the approximate FA cells proposed previously. In one-dimensional integer DCT, the maximum value of the DC coefficient is given by $45 \times 8 \times 255 = 91800$ [6]. This is represented in two's complement using 18 bits. Therefore, we use a bit-precision of 20 bits (a multiple of 4, for convenience) in all our computations.

According to our experiments, using approximate FA cells beyond the 9^{th} LSB results in an appreciable quality loss, so we consider 3 cases, where we use approximate FA cells for 7, 8 and 9 LSBs. FA cells corresponding to other bits in each case are accurate. We also show corresponding results for truncation. The design using accurate adders everywhere in DCT and IDCT is considered to be the base case.

1) Output quality: The output Peak Signal to Noise Ratio (PSNR) for the base case is 31.16 dB. Figure 7 shows the output images for the base case, truncation, and approximation 3. Clearly, since the error introduced by truncation has an accumulative nature, the output image has severe blockiness. Hence using truncation may not yield fruitful results. Figure 8 shows the output quality for truncation and different approximations when 7, 8 and 9 LSBs are approximated. Clearly, using approximate FAs in the LSBs can make up for the lost quality (due to truncation) to a large extent, and also provide substantial power savings. Power consumption for different approximations is discussed in the next section.

Fig. 7: Output quality when 8 LSBs are approximated

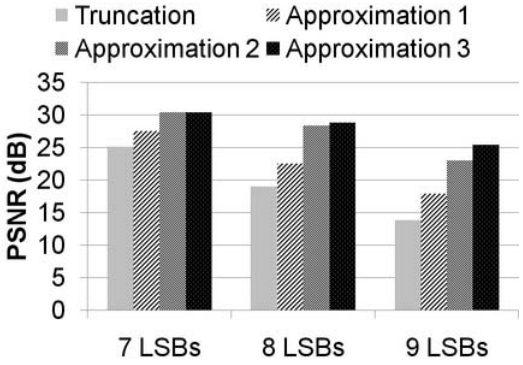


Fig. 8: Output quality for different techniques

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. DCT operates at 1.28 V and IDCT operates at 1.13 V for the accurate case. Table III shows the operating supply voltages for different approximations and truncation in IBM 90nm technology. These voltages are chosen such that the errors are due to *functional/truth table approximation* only, and not due to *VOS*. The power consumption for DCT and IDCT blocks was determined using nanosim run with 12288 vectors from the standard image Lena in IBM 90nm technology. Spice netlists for conventional as well as approximate FA cells were extracted from the respective layouts (Figures 5a - 5d) and used in the simulation. Figure 9 shows the total power savings for DCT and IDCT blocks over the base case for different approximations and truncation. Approximation 3 provides maximum power savings among all approximations. The use of approximate adders also results in area savings, which is discussed in the next section.

TABLE III: Operating voltages for different techniques

Technique	V_{DD}(V) for the 3 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
Approx. 1	1.18	1.05	1.1	1.03	1.1	1.03
Approx. 2	1.15	1.1	1.13	1.1	1.1	1.1
Approx. 3	1.14	1.02	1.11	1.01	1.1	1

3) Area savings: As shown in Table II, the layout area of approximate FA cells is lesser compared to the conventional case. Furthermore, approximation 3 also helps in avoiding calculation of C_{out} for the LSBs. Thus the area of the whole approximate DCT-IDCT system is less compared to the base (or accurate) case. Figure 10 shows the area savings for different approximations over the base case. Approximation 3 has the maximum area savings irrespective of the number of LSBs approximated.

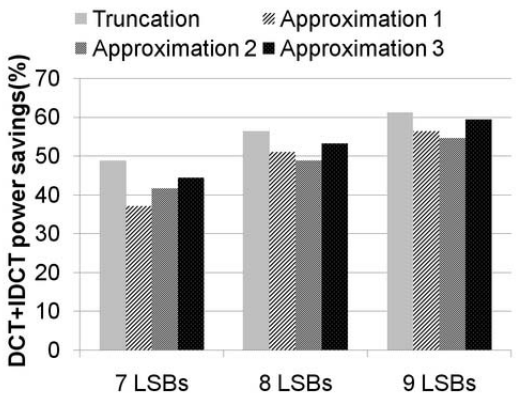


Fig. 9: Power savings for DCT+IDCT over the base case

Truncating 4 LSBs provides a PSNR of 31.08 dB, 24.22% power savings, and 20% area savings. On the other hand, using approximation 3 for 8 LSBs provides a PSNR of 28.9 dB, 53.29% power savings, and 32.67% area savings. This beats truncation in both power and area, with very minimal loss in quality. In the next section, we present the results of using our approximate adders in video compression.

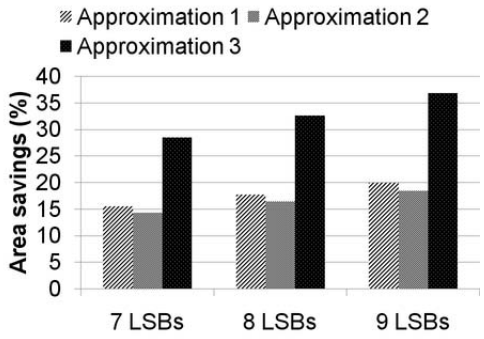


Fig. 10: Area savings for DCT+IDCT over the base case

B. Video Compression

Another popular multimedia application to which we have applied imprecise adders is video compression. Figure 11 shows the block diagram of MPEG encoder architecture [17].

Here we focus on Motion Estimation (ME) [4] hardware which accounts for nearly 70% of the total power consumption [17]. In addition to the motion estimation block we have implemented few other adders/subtractors using approximate adders which are shown in white (Motion Estimation, subtractor for DCT error, and IDCT reconstruction adder) in Figure 11. The basic building block for any ME algorithm is the hardware that implements Sum of Absolute Difference (SAD) [17], which consists of an 8-bit absolute difference block followed by a 16-bit 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.

1) Output quality: Figure 12 shows the average frame PSNR for 50 frames of the AKIYO benchmark CIF [17] video sequence. We consider the frame quality for truncation as well as the three approximations applied to 1, 2, 3 and 4 LSBs of the adders in SAD

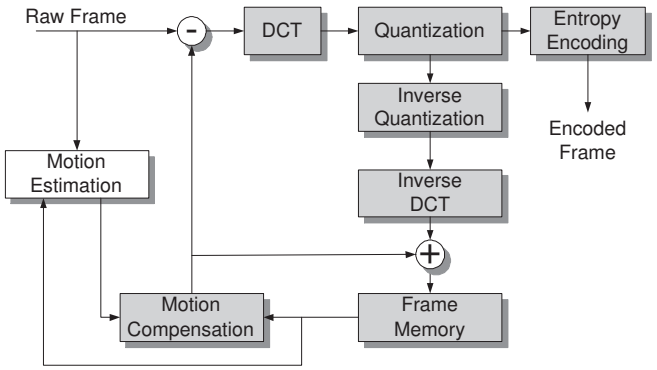


Fig. 11: MPEG encoder architecture

and MPEG hardware shown in white (Motion Estimation, subtractor for DCT error, and IDCT reconstruction adder) in Figure 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: Figure 13 shows the power savings for approximate adders. Again, approximation 3 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. Overall Analysis of Different Approximations

We have compared the different approximations proposed for image compression on the basis of PSNR, power dissipation, and area. In order to have a unified comparison, we define a metric named QUAP as QUAP = (QUality X Area savings (%) X Power savings (%)). In this formula, quality is computed as $PSNR^2$. Here we place a greater emphasis on output quality, since it is of prime importance to the end user. Figure 14 shows the metric calculated for different approximations. As expected, approximation 3 outperforms all other approximations by a huge margin. Approximation 2 is the next best approximation. Table IV shows the Quality Loss (QL) and Power Savings (PS) when approximation 3 is used for some standard benchmarks, out of which 3 are from [18]. The corresponding area savings for 7, 8 and 9 LSBs are 28.5%, 32.67% and 36.85% respectively.

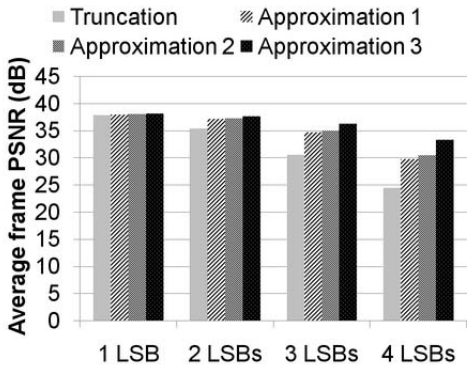


Fig. 12: Output quality for AKIYO sequence

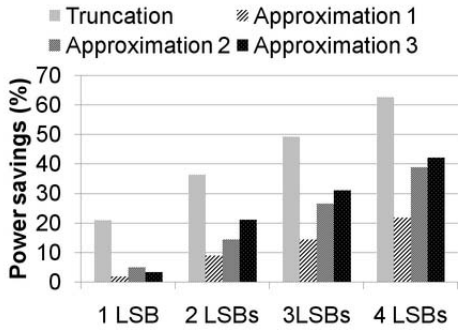


Fig. 13: Power savings over accurate adders for video compression

In particular, approximation 3 can provide power and area savings with almost no loss in output quality when 7 or less than 7 LSBs are approximated.

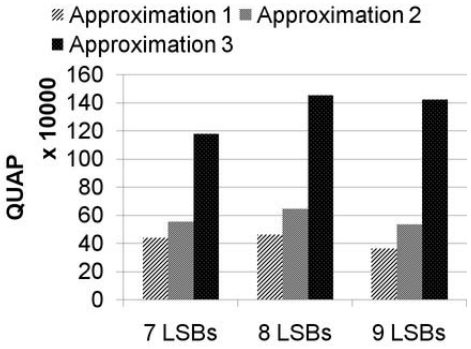


Fig. 14: Comparision of different approximations

TABLE IV: Approximation 3 tested across standard benchmarks

Image	7 LSBs		8 LSBs		9 LSBs	
	QL(dB)	PS(%)	QL(dB)	PS(%)	QL(dB)	PS(%)
lena	0.69	44.57	2.26	53.29	5.7	59.57
mandril	0.17	40.48	0.58	49.67	2.05	54.35
kodim09	0.31	43.24	1.14	52.07	3.65	57.07
kodim19	0.42	43.2	1.34	52.54	4.04	56.94
kodim23	0.46	43.2	1.35	52.87	4.39	58.39

V. CONCLUSION

In this paper, we have proposed several imprecise or approximate FA cells that can be effectively utilized to build multi-bit arithmetic units and trade off power, area and quality for error-resilient DSP systems. Our approach aims to simplify the complexity of a conventional mirror adder cell by reducing the number of transistors and also the load capacitances. When the errors introduced by these approximations are reflected at a high level in a typical DSP algorithm, the impact on output quality is almost negligible. Thus our approach differs from previous approaches where errors are introduced due to VOS [1], [2], [3], [4], [5], [6], [7], [8]. A decrease in the effective switched capacitance results in a lower power dissipation. Moreover, the proposed approximate FA cells also allow the system to operate at a lower supply voltage than the conventional case. This further contributes to a quadratic reduction in power dissipation. An added benefit due to lower number of transistors is a reduction in total area. While techniques like SDC and ANT have an area overhead, on the contrary, our technique provides substantial area savings too. We have demonstrated the utility of the proposed approximate FA cells in two DSP systems, viz. image and video compression. We believe that they can be used on the top of already existing low-power techniques like SDC and ANT to extract multi-fold benefits with a very minimal loss in output quality.

REFERENCES

[1] R. Hegde and N. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. IEEE/ACM International Symposium on Low Power Electronics and Design*, 1999, pp. 30 – 35.

[2] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. VLSI Syst.*, vol. 9, no. 6, pp. 813–823, 2001.

[3] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. VLSI Syst.*, vol. 12, no. 5, pp. 497 – 510, 2004.

[4] G. Varatkar and N. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *Proc. IEEE/ACM International Symposium on Low Power Electronics and Design*, 2006, pp. 113 – 118.

[5] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. IEEE/ACM International Symposium on Low Power Electronics and Design*, 2009, pp. 195–200.

[6] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power dct architecture," in *Proc. Design, Automation, and Test in Europe*, 2007, pp. 1 – 6.

[7] 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 on Signal Processing Systems*, 2009, pp. 133 – 138.

[8] 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 *CASES*, 2008, pp. 187–196.

[9] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Automation, and Test in Europe*, 2008, pp. 1250–1255.

[10] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. International Symposium on Integrated Circuits*, December 2009, pp. 69 –72.

[11] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th International Conference on VLSI Design*, January 2011, pp. 346 –351.

[12] 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. Appl.*, vol. 6, pp. 5:1–5:21, February 2010.

[13] 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, Automation, and Test in Europe*, March 2010, pp. 1725 –1730.

[14] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[15] G. Wallace, "The jpeg still picture compression standard," *IEEE Trans. Consumer Electronics*, vol. 38, no. 1, pp. xviii –xxxiv, 1992.

[16] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons, 1999.

[17] P. M. Kuhn and K. P. M., *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[18] http://r0k.us/graphics/kodak/.