

Low-Energy Signal Processing using Circuit-Level Timing-Error Acceptance

Ku He, Andreas Gerstlauer and Michael Orshansky
Department of Electrical and Computer Engineering
University of Texas at Austin

Abstract— In digital signal processing (DSP) applications, large energy gains can be obtained by accepting some degradation in the output signal quality. In this paper, we present static and dynamic techniques for circuit-level timing-error acceptance to significantly improve energy efficiency by shaping the quality-energy tradeoff achievable via aggressive V_{DD} scaling. The proposed techniques specifically target the earliest and worst timing error offenders to allow for larger V_{DD} reduction while maintaining high signal quality. We demonstrate the effectiveness of the proposed techniques on image processing applications, including a DCT/IDCT-based image compression system and an image-sharpening filter. The designs were synthesized using a 45nm standard cell library. Results show that 40-60% energy savings can be achieved at less than 1dB loss in the peak signal-to-noise ratio. The overhead for the needed control logic is less than 6% of the area of the original design.

I. INTRODUCTION

The fast-growing market of portable systems with limited battery life requires continued advances in ultra low-energy design. In this paper, we present novel techniques that exploit special properties of DSP systems to reduce their energy consumption. In conventional DSP designs, as in other digital design flows, timing correctness of all operations is guaranteed by construction. Since in many DSP applications the best signal quality is not necessarily required, it is possible to tolerate some timing errors induced by lower V_{DD} . If aggressive voltage scaling can be made possible with only a small, bounded quality loss, it can lead to significantly reduced energy consumption.

Several efforts in the past have explored the possibility of trading quality in DSP systems for lower energy. In [1], [2], energy is reduced by discarding algorithm steps or iterations that contribute less to the final quality. In [3], adaptive precision of the arithmetic unit output is used to save energy. In [4], [5], energy reduction is enabled by using lower voltage on a main computing block, and by employing a simpler error-correcting block that runs at a higher voltage and is thus, error-free, to improve the results impacted by timing errors of the main block. An important distinction between prior work and our strategy is that in other work, the results produced by blocks subject to timing errors are not directly accepted. By contrast, our strategy allows using the erroneous results directly, provided, of course, that the magnitude of error is carefully controlled.

In the following, we present timing error acceptance techniques at two levels of granularity, at the operation and the architecture level, to allow significantly improved quality-energy (Q-E) tradeoffs. Depending on knowledge about data statistics, these techniques can be either applied at design time or at run time. Techniques are introduced and demonstrated on the design of a Discrete Cosine Transform (DCT) and an Inverse Discrete Cosine Transform (IDCT) as widely used image and video processing kernels, as well as on a Finite Impulse Response (FIR) filter that performs image sharpening. Specifically, the key contributions for Q-E profile shaping are: 1) Controlling large-magnitude timing errors in operations by exploiting the knowledge of statistics of operands. In many cases, we have knowledge of data distributions that can be exploited at design time or at run time. Our technique is based on the realization that functional units with reduced bitwidth can be used to process small-magnitude operands.

2) Controlling the frequency of error-generating additions by statically or dynamically re-arranging the sequence of operations. Specifically, we target a reduction in the cumulative quality loss resulting from multiple consecutive operations. Such multi-operand computations occur, for example, in accumulation, which is a key component of many DSP algorithms.

3) Preventing occurrence of errors which can spread and get amplified throughout the algorithm. An important aspect of a design methodology that allows some timing errors is controlling the impact of these errors on output quality from the perspective of the entire algorithm. Specifically, a result impacted by timing errors early in the algorithm can have a dramatic impact on the overall quality by affecting downstream computations through repeated reuse of incorrect data.

The rest of the paper is organized as follows, Section II discusses the principle of timing error management, followed by an introduction of the techniques to control such errors; Section III shows the experiment results, and finally, Section IV concludes the paper with a summary and outlook.

II. TIMING ERROR MANAGEMENT

A. Error control through knowledge of operand statistics

In digital signal processing applications, input signals usually follow certain distributions. Such properties can be exploited to control the magnitude of timing errors during computations. In DCT/IDCT type applications, the input/output magnitudes exhibit spatial patterns, which make it possible to apply design-time techniques to reduce timing errors. By

contrast, in digital filters, the input magnitudes are random and run-time techniques need to be used.

In [6], a design-time technique is introduced to reduce early and large-magnitude timing errors in an IDCT, where the specific focus is on errors in addition as a fundamental building block of most DSP algorithms. The key observation is that in a DCT/IDCT, computations involving high frequency coefficients usually have operands with a small magnitude and often with opposing sign. In regular addition, it is such operands that trigger the longest carry chains and hence experience the largest timing errors first. The specific technique is based on the realization that an adder with a smaller bitwidth can be used to process these operands. Two objectives are achieved: the magnitude of quality loss is reduced and its onset is delayed. Large-valued operands, of course, require a regular-width adder. Note that in an actual implementation it is possible to utilize a single adder with variable bitwidth.

The design-time application of a reduced-width adder technique involves two questions: 1) how to perform the classification of high versus low frequencies; and 2) how to identify the optimal bitwidth of the reduced-width adder. The solution to these two questions can be represented in terms of a simulation-based model that is used to explore the timing error behavior under varying partition boundaries (x) and small adder widths (W_2) [6]. Based on such an exploration, a Pareto curve of Q-E tradeoffs can be generated to determine the optimal x and W_2 .

Note that in [6], the technique is demonstrated on a 2D-IDCT design. However, it can be applied to a 2D-DCT in a similar manner. The key difference is that instead of partitioning computations according to their location in the input matrix, in the 2D-DCT case, the partition is done on the output matrix. Under such a mapping, the procedure of determining the optimal x and W_2 for the DCT is the same as that in the IDCT.

In digital filtering applications, magnitude information for input/output data is not available at design time. Hence, a run-time technique has to be used to reduce timing errors. The idea is to adjust adder bitwidth dynamically, with the purpose of eliminating early and large timing errors. This requires checking the magnitude of input data and processing operands on an adder with the smallest bitwidth sufficient for the particular inputs. To allow the results to be used in downstream computations, we further perform sign extension to the full bitwidth.

In the implementation, it is assumed that only one physical adder is used. The inputs are first sent to the magnitude-checking logic block, which can be implemented compactly. The checking logic uses *AND* gates to determine whether a specified number of higher-significance input bits are all zeros or all ones. If this condition is true, a width-control logic is activated to perform truncation and sign-extension on the adder output. In essence, each time the magnitude-checking logic initiates a reduced-width addition, a smaller effective configuration of an otherwise full-width adder is used. The technique abstraction is shown in Figure 1.

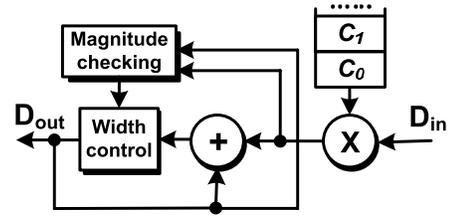


Fig. 1. Dynamic-width adder architecture.

Two questions arise for the implementation of a dynamic technique: 1) how many reduced-width levels are needed, and 2) what their optimal reduced widths are. According to our experiments, two bitwidths are sufficient for an efficient Q-E tradeoff. Increasing the number of bitwidths does not substantially improve results. Compared to a minimum quality loss between 3dB and 15dB in the two-adder case, losses for three or four bitwidths are in the range of 1dB-7dB. Since we use only two adders, the determination of the optimal reduced width can be done by exhaustive bitwidth sweeping: we define the reduced width as W_2 , and for a given timing budget, we generate the quality loss for varying W_2 . The optimal W_2 corresponds to the minimum quality loss for input data with a certain variance.

B. Error control by reordering of accumulations

In the previous section, we demonstrated techniques for controlling timing errors in individual operations. In this section, we present static and dynamic techniques for reduction of the cumulative quality loss resulting from multiple additions, such as in accumulations.

At design time, the philosophy is to manipulate the input data distributions of intermediate MAC operations by reordering operations (e.g. filter taps), such that further reduced widths can be applied. In a traditional single MAC unit design, the width of the MAC unit is determined by the maximal bitwidth over all operations, which is generally independent of any intermediate reordering. However, under a timing error acceptance philosophy, reordering will allow us to statically apply adders of different width to different steps in order to reduce timing errors. Furthermore, in combination with dynamic bitwidth adjustment (Section II-A), reordering can, on average, reduce the magnitude of data in intermediate operations and hence increase the effectiveness of this technique. For example, in a digital filter, the computation process can be formulated as: $y(n) = \sum_{i=0}^N b_i \cdot x(n-i) + \sum_{i=1}^N a_i \cdot y(n-i)$. In computing the final output, a filter needs to generate a set of intermediate results, which correspond to a set of intermediate transfer functions H_0-H_{2N} . These transfer functions determine the maximum possible gain over all frequencies at internal nodes and, hence, the minimum required bitwidth for each intermediate result in the datapath. Without affecting the output of the filter, intermediate transfer functions vary when the order of filter taps is changed. As such, we can reduce gains and hence the bitwidth of intermediate operations by optimally reordering the taps. Such a reordering can be done at design time. It allows us to apply an adder of smaller width

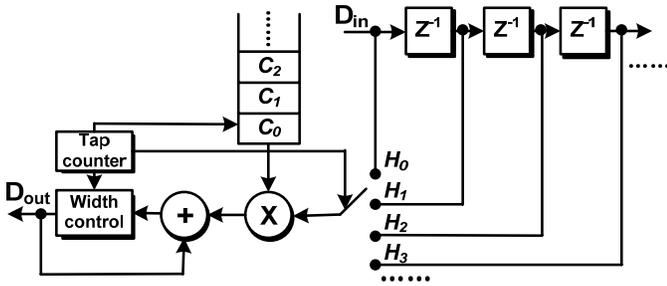


Fig. 2. Filter reordering technique.

to each tap in order to reduce the timing errors under voltage scaling. The technique diagram is shown in Figure 2.

A second approach is to perform reordering at run-time. The key observation is that if positive and negative operands are accumulated separately, the number of error-producing operations is reduced to one last addition that involves operands with opposite sign. At the same time, the operands involved in this last addition are guaranteed to be larger in absolute value than any individual opposite-sign operands involved in the original sequence. This guarantees that the reordered accumulation will result in a smaller quality loss under scaled timing.

The difference between optimized and un-optimized sequences is significant. As an example, consider four numbers (-1, 1, -1, 1) being accumulated. There are three possible sequences of accumulation:

- Case 1: 11111111+00000001+11111111+00000001
- Case 2: 11111111+11111111+00000001+00000001
- Case 3: (11111111+11111111)+(00000001+00000001)

For Case 1, the 1st and the 3rd additions have large delay, each with a carry chain length of 8. For Case 2, the 3rd addition has large delay with a carry chain of 8. For Case 3, only the addition outside the brackets has large delay with a carry length of 7. The total timing budget in Case 3 is roughly half of that of Case 1. Thus, we observe that the order of accumulation can significantly affect the frequency of worst-case delay as well as the length of the longest carry chain.

The proposed implementation uses the sign bits in the MSB to separate the positive and negative operands when loading data. Details are presented in [6]. Compared to the original implementation, the reordered accumulation carries extra overhead for the reordering logic and duplicate accumulation registers. Nevertheless, experiments show that the technique can significantly improve the quality-energy profile under scaled timing.

C. Preventing error spread and amplification

In previous sections, we presented techniques for targeting individual error sources at the operation and architecture levels. With knowledge of the application, we now focus on control of sources of errors that have the potential to be spread and amplified at the algorithm level. More specifically, we propose a technique using algorithm-level retiming to explicitly prevent errors in critical steps that may have a large impact on downstream results and hence overall quality.

For the 2D-DCT/IDCT algorithm, analysis of control and data flow is relatively simple because it consists of two nearly-identical steps: (1) $T = C^T \cdot A$ and (2) $I = T \cdot C$. In [6], we address the problem of a timing error in Step 1. Such an error can generate multiple output errors in I because each element of T is used in multiple computations of Step 2. As a result, the noise in the decoded image of an unmodified DCT/IDCT has a stripe pattern (see Figure 4 in Section III).

Thus, to avoid such wide-spread quality loss, we need to ensure that no errors occur in Step 1. We assume an architecture in which supply voltage can only be scaled uniformly. If timing budgets are allocated to steps based on worst-case analysis, any reduction in V_{DD} would lead to a reduced timing slack in Step 1 and hence un-allowable levels of errors being generated there. We therefore propose a strategy to allocate extra timing margins to critical steps, such as Step 1. Importantly, given overall latency constraints for the design, as is the case for many real-time image or video coding applications, end-to-end algorithm timing must remain constant and performance must not be degraded. Thus, an important element of protecting the early algorithm steps is a re-allocation strategy that shifts timing budgets between steps. Maintaining a constant total time, the approach is to borrow computing time from non-critical algorithm steps in order to increase timing budgets in critical ones, and reduce overall quality loss. This is achieved by multi-cycling operations in Step 1 while increasing the overall clock frequency such that total latency remains constant [6].

III. EXPERIMENTAL RESULTS

We demonstrate the effectiveness of our techniques on the design of a 2D-DCT/IDCT and an image sharpening FIR filter. In the DCT/IDCT case, we apply static bitwidth reduction, dynamic accumulation reordering and algorithm-level retiming optimizations. For the image sharpening filter, we utilize dynamic bitwidth adjustment and static filter tap reordering.

All designs are implemented in Verilog-HDL and synthesized using Design Compiler with the OSU 45nm PDK. We use Synopsys Hercules to translate the RTL code into a SPICE netlist. Then, we build a NanoSim + VCS testbench to enable both RTL-level and SPICE-level simulations to obtain final output images and energy-delay results, respectively.

The architecture of our final 2D-DCT/IDCT implementation is a folded one [7], where each 1D-DCT/IDCT shares the same pipelined arithmetic unit containing an adder and a multiplier. The DCT data and coefficient matrices have 8-bit and 8-bit resolution, and the final output has 16-bit resolution. The IDCT data and coefficient matrices have 16-bit and 8-bit resolution, respectively. The multiplier is pipelined and has a width of 8×16 bits. The adder is a ripple-carry adder with a width of 24 bits. Such a design restricts timing errors entirely to the adder for acceptable quality loss. Individual timing error control techniques can be combined to achieve maximum energy savings [6]. Different from [6], in this paper, we use quantized data for our experiments. Only the Y signal of a Y:Cb:Cr format image is used.

TABLE I
ENERGY SAVINGS AND AREA OVERHEADS IN 2D-DCT/IDCT.

	V_{DD}	Energy Saving	Area Ov.
Original	1.1V	0%	0%
Reduced-width	1.00V / 0.95V	45.0% / 49.1%	0.2% / 0.4%
Reordering	0.95V / 0.95V	62.4% / 32.1%	3.8% / 3.1%
Rescheduling	1.00V / 0.95V	2.0% / 42.1%	0.1% / 0.1%
Combined	0.90V / 0.90V	73.1% / 63.1%	3.9% / 3.5%



(a) Original: Energy=570 μ J
PSNR=31.6dB



(b) Original: Energy=137 μ J
PSNR=16.5dB



(c) Proposed: Energy=143 μ J
PSNR=31.2dB



(d) Proposed: Energy=99 μ J
PSNR=28.3dB

Fig. 3. Images from IDCT under different energy budgets.

The energy saving and area for each technique and their combination are shown in Table I. Note that for DCT rescheduling is not as effective as for IDCT because of nearly-equal importance of both matrix multiplication steps for final error. A significantly improved trade-off curve is generated by a non-trivial combination of individual techniques. Finally, a set of sample images under scaled V_{DD} is shown in Figure 3. Note that achieving a similar energy reduction by conventional V_{DD} scaling would result in unacceptable degradation of image quality (Figure 3(b)).

For the implementation of the sharpening filter, we generated a coefficient kernel using MATLAB's *fspecial* function with the filter option *unsharp*. The filter is realized as a 2-D convolution of each pixel with this kernel. We implement a 1-D version on our architecture using the algorithm as a 9th-order FIR filter. The core multiply-accumulate (MAC) operations are realized using a multiplier and adder that are chained to operate in one clock cycle. Hence, under voltage scaling, timing errors will affect the addition at the end of the chain first. The full and reduced width adders in this case have 24 and 20 bits, respectively.

TABLE II
ENERGY SAVINGS AND AREA OVERHEADS IN SHARPENING FILTER.

	V_{DD}	SSNR/PSNR	Eng. Saving	Area Ov.
Reorder	0.80V	22.8dB	58.1%	1.0%
Dyn. width	0.75V	23.0dB	64.2%	2.0%
Combined	0.70V	23.3dB	69.7%	2.1%



(a) Sharpened:
Energy=2.19 μ J
PSNR=23.9dB



(b) Unoptimized:
Energy=1.27 μ J
PSNR=19.6dB



(c) Dynamic width adder:
Energy=1.27 μ J
PSNR=23.0dB



(d) Combined:
Energy=1.27 μ J
PSNR=23.2dB

Fig. 4. Modified image sharpening filter output.

Energy savings and area overhead are listed in Table II. Sample images after applying the sharpening filter with and without our error control are shown in Figure 4. From Figure 4(b) we can see that, compared to a sharpened image at nominal voltage (Figure 4(a)), voltage scaling without error control causes a lot of visually noticeable salt-and-pepper artifacts. By contrast, using our techniques (Figure 4(d)), such noise is significantly reduced and resulting images exhibit good perceived quality at the same reduced energy.

IV. CONCLUSIONS

This paper presented techniques that enable shaping of the quality-energy tradeoff under aggressively scaled V_{DD} through controlled timing error acceptance. We demonstrated the implementation of these techniques on 2D-DCT/IDCT designs, as well as an image sharpening filter. Results show that significant energy savings can be achieved while maintaining a constant performance and good image PSNR.

REFERENCES

- [1] S. H. Nawab, A. V. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig, "Approximate signal processing," *VLSI Signal Processing*, vol. 15, pp. 177–200, 1997.
- [2] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *JSSC*, pp. 395–400, 1996.
- [3] A. Sinha and A. P. Chandrakasan, "Energy efficient filtering using adaptive precision and variable voltage," *ASIC SOC Conference*, pp. 327–331, 1999.
- [4] R. Hedge and N. R. Shanbhag, "Soft digital signal processing," *TVLSIS*, pp. 379–391, 2000.
- [5] L. Wang and N. R. Shanbhag, "Low-power filtering via adaptive error-cancellation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 51, no. 2, pp. 575–583, 2003.
- [6] K. He, A. Gerstlauer, and M. Orshansky, "Controlled timing-error acceptance for low energy idct design," *DATE*, pp. 492–499, 2011.
- [7] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, and Y. Yamashita, "A 100-mhz 2-d discrete cosine transform core processor," *JSSC*, vol. 27, pp. 492–499, 1992.