Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications

Srinivasan Narayanamoorthy, Hadi Asghari Moghaddam, Zhenhong Liu, Taejoon Park, and Nam Sung Kim

***Abstract*— The need to support various digital signal processing (DSP) and classification applications on energy-constrained devices has steadily grown. Such applications often extensively perform matrix multiplications using fixed-point arithmetic while exhibiting tolerance for some computational errors. Hence, improving the energy efficiency of multiplications is critical. In this brief, we propose multiplier architectures that can tradeoff computational accuracy with energy consumption at design time. Compared with a precise multiplier, the proposed multiplier can consume 58% less energy/op with average computational error of ∼1%. Finally, we demonstrate that such a small computational error does not notably impact the quality of DSP and the accuracy of classification applications.**

***Index Terms*— Approximation, energy efficiency, multiplication.**

I. Introduction

Achieving high energy efficiency has become a key design objective for embedded and mobile computing devices due to their limited battery capacity and power budget. To improve energy efficiency of such computing devices, significant efforts have already been devoted at various levels, from software to architecture, and all the way down to circuit and technology levels.

Embedded and mobile computing devices are frequently required to execute some key digital signal processing (DSP) and classification applications. To further improve energy efficiency of executing such applications, first, dedicated specialized processors are often integrated in computing devices. It has been reported that the use of such specialized processors can improve energy efficiency by 10–100× compared with general-purpose processors at the same voltage and technology generation [1].

Second, many DSP and classification applications heavily rely on complex probabilistic mathematical models and are designed to process information that typically contains noise. Thus, for some computational error, they exhibit graceful degradation in overall DSP quality and classification accuracy instead of a catastrophic failure. Such computational error tolerance has been exploited by trading accuracy with energy consumption (see [2]).

Finally, these algorithms are initially designed and trained with floating-point (FP) arithmetic, but they are often converted to fixed-point arithmetic due to the area and power cost of supporting FP units in embedded computing devices [3]. Although this conversion process leads to some loss of computational accuracy, it does not notably affect the quality of DSP and the accuracy of classification applications due to computational error tolerance.

Most of such algorithms extensively perform matrix multiplications as their fundamental operation, while a multiplier is typically an inherently energy-hungry component. To improve energy efficiency of multipliers, previous studies have explored various techniques exploiting computational error tolerance. They can be classified into three categories: 1) aggressive voltage scaling [4], [5]; 2) truncation of bit-width [4]–[6]; and 3) use of inaccurate building blocks [7]. Chippa *et al.* [4] proposed scalable effort hardware design and explored algorithm-, architecture-, and circuit-level scaling to minimize energy consumption while offering acceptable classification quality through aggressively scaling voltage and truncating least-significant bits. Kulkarni *et al.* [7] proposed an under-designed 16×16 multiplier using inaccurate 2 × 2 partial product generators (PPG) while guaranteeing the minimum and maximum accuracy fixed at design time. Each PPG has fewer transistors compared with the accurate 2 × 2 one, reducing both dynamic and leakage energy at the cost of some accuracy loss. Babića *et al.* [8] proposed a novel iterative log approximate multiplier using leading one detectors (LODs) to support variable accuracy.

In this brief, we propose an approximate multiplication technique that takes m consecutive bits (i.e., m-bit segment) from each n-bit operand, where m is equal to or greater than $n/2$. An m-bit segment can start only from one of two or three fixed bit positions depending on where the leading one bit is located for a positive number. This approach can provide much higher accuracy than one simply truncating the LSBs, because it can more effectively capture more noteworthy bits. Although we can capture m-bit segments starting from the exact leading one bit position, such an approach requires expensive LODs and shifters to take m-bit segments starting from the leading one position, steer them to an $m \times m$ multiplier, and expand $2m$ bits to $2n$ bits. In contrast, our approach is more scalable than one that captures m-bit segments starting from the leading one bits since it limits the possible starting bit positions of an m-bit segment to two or three regardless of m and n chosen at design time, eliminating LODs, and replacing shifters with multiplexers. Finally, we also observe that one of two operands in each multiplication for DSP and classification algorithms is often stored in memory (e.g., coefficients in filter algorithms and trained weight values in artificial neural networks) and repeatedly used. We exploit it to further improve the energy efficiency of our approximate multiplier.

The rest of this brief is organized as follows. Section II details the proposed multiplier architecture. Section III analyzes energy consumption and computational accuracy of various approximate multipliers and impact of such multipliers on quality of DSP and accuracy of classification algorithms. Finally, Section IV concludes this brief.

II. Approximate Multiplier Exploiting Significant Segments of Operands

In order to motivate and describe our proposed multiplier, we define an m-bit segment as m contiguous bits starting with the

Manuscript received August 5, 2013; revised March 7, 2014 and June 4, 2014; accepted June 5, 2014. Date of publication July 17, 2014; date of current version May 20, 2015. This work was supported in part by the National Science Foundation under Grant CCF-0953603, in part by the Defense Advanced Research Projects Agency under Grant HR0011-12-2-0019, and in part by the MSIP of Korea (CPS Global Center). N. S. Kim has a financial interest in the Advanced Micro Devices, Seoul, Korea and Samsung Electronics Company, Ltd., Seoul. *(Corresponding author: N. S. Kim and T. Park.)*

S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, and N. S. Kim are with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: narayanamoor@wisc.edu; hmoghaddam@wisc.edu; liu238@wisc.edu; nam.sung.kim@gmail.com).

T. Park is with the Department of Information and Communication Engineering, Daegu Gyeongbuk Institute of Science and Technology, Daegu 711-873, Korea (e-mail: tjpark@dgist.ac.kr).

Digital Object Identifier 10.1109/TVLSI.2014.2333366

TABLE I
QoC OF APPLICATIONS USING APPROXIMATE MULTIPLIERS RELATIVE TO THE PRECISE MULTIPLIER

	Audio (PESQ)	Image (SSIM)	Recognition
AM2×2	97.74%	86.6%	97.6%
DSM8×8	99.29%	99.8%	98.3%
DSM6×6	99.14%	98.3%	98.2%
SSM8×8	93.65%	74.2%	94.4%
SSM10×10	99.27%	99.2%	98.2%
ESSM8×8	99.14%	98.4%	98.2%
TRUN8×8	63.00%	1.4%	11.4%
PM	100.00%	99.9%	98.3%

×					0**111**	**0111**	**0**101	1010
					0000	00**11**	**1011**	**10**11
=	0001	0001	0100	1001	0101	0000	0000	0000

Fig. 1. Example of a multiplication with 8-b segments of two 16-b operands; bold-font bits comprise the segments.

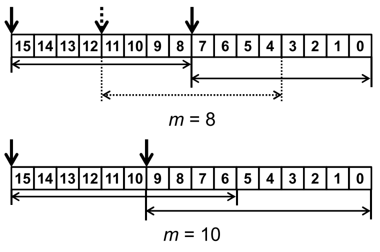

Fig. 2. Possible starting bit positions of 8-b and 10-b segments indicated by arrows; the dotted arrow is the case for supporting three possible starting bit positions.

×					01xx	xxxx	xxxx	xxxx
					0001	xxxx	xxxx	xxxx
=	▨	▨	▨	▨	0000	0000	0000	0000
×					01xx	xxxx	xxxx	xxxx
					0000	0000	01xxx	xxxx
=	0000	0000	▨	▨	▨	▨	0000	0000
×					0000	0000	01xx	xxxx
					01xx	xxxx	xxxx	xxxx
=	0000	0000	▨	▨	▨	▨	0000	0000
×					0000	0000	01xx	xxxx
					0000	0000	01xx	xxxx
=	0000	0000	0000	0000	▨	▨	▨	▨

Fig. 3. Examples of 16×16 multiplications based on 8-b segments with two possible starting bit positions for 8-b segments. The shaded cells represent 8-b segments and the aligned position of 8×8 multiplication results.

×					0000	0001	xxxx	xxxx
					0000	0010	xxxx	xxxx

Fig. 4. Example of low accuracy for SSM16×16.

leading one in an n-bit positive operand. We dub this method dynamic segment method (DSM) in contrast to static segment method (SSM) that will be discussed later in this section. With two m-bit segments from two n-bit operands, we can perform a multiplication using an $m \times m$ multiplier. Fig. 1 shows an example of a multiplication after taking 8-b segments from 16-b operands. In this example, we can achieve 99.4% accuracy for a 16×16 multiplication even with an 8×8 multiplier.

Such a multiplication approach has little negative impact on computational accuracy because it can eliminates redundant bits (i.e., sign-extension bits) while feeding the most useful m significant bits to the multiplier; we will provide detailed evaluations of computational accuracy for various m in Section III. Furthermore, an $m \times m$ multiplier consumes much less energy than an $n \times n$ multiplier, because the complexity (and thus energy consumption) of multipliers quadratically increases with n. For example, the 4×4 and 8×8 multipliers consume almost 20 × and 5 × less energy than a 16×16 multiplier per operation on average. However, a DSM requires: 1) two LODs; 2) two n-bit shifters to align the leading one position of each n-bit operand to the MSB position of each m-bit segment to apply their m-bit segments to the $m \times m$ multiplier; and 3) one $2n$-bit shifter to expand a $2m$-bit result to $2n$ bits. 1)–3) incur considerable area and energy penalties completely negating the energy benefit of using the $m \times m$ multiplier; we provide detailed evaluations for two m values in Section I.

The area and energy penalties associated with 1)–3) in DSM are to capture an m-bit segment starting from an arbitrary bit position in an n-bit operand because the leading one bit can be anywhere. Thus, we proposed to limit possible starting bit positions to extract an m-bit segment from an n-bit operand to two or three at most in SSM, where Fig. 2 shows examples of extracting 8-b and 10-b segments from a 16-b operand. Regardless of m and n, we have four possible combinations of taking two m-bit segments from two n-bit operands for a multiplication using the m-bit SSM.

For a multiplication, we choose the m-bit segment that contains the leading one bit of each operand and apply the chosen segments from both operands to the $m \times m$ multiplier. The SSM greatly simplifies the circuit that chooses m-bit segments and steers them to the $m \times m$ multiplier by replacing two n-bit LODs and shifters for the DSM with two $(n–m)$-input OR gates and m-bit 2-to-1 multiplexers; if the first $(n–m)$ bits starting from the MSB are all zeros, the lower m-bit segment must contain the leading one. Furthermore, the SSM also allows us to replace the $2n$-bit shifter used for the DSM with a $2n$-bit 3-to-1 multiplexer. Since the segment for each operand is taken from one of two possible segments in an n-bit operand, a $2m$-bit result can be expanded to a $2n$bit result by left-shifting the $2m$-bit result by one of three possible shift amounts: 1) no shift when both segments are from the lower m-bit segments; 2) $(n–m)$ shift when two segments are from the upper and lower ones, respectively; and 3) $2\times$ $(n–m)$ shift when both segments are from the upper ones, as shown in Fig. 3.

Note that the accuracy of an SSM with $m = n/2$ can be significantly low for operands shown in Fig. 4, where many MSBs of m-bit segments containing the leading one bit are filled with zeros. On the other hand, such a problem becomes less severe as m is larger than $n/2$; there is an overlap in a range of bits covered by both possible m-bit segments as shown for $m = 10$ in Fig. 2. Thus, for an SSM with $m = n/2$, we propose to support one more bit position that allows us to extract an m-bit segment indicated by the dotted arrow in Fig. 2. This will be able to effectively capture operand pairs similar to the one shown in Fig. 4.

Fig. 5 shows an SSM allowing to take an m-bit segment from two possible bit positions of an n-bit operand. The key advantage is its scalability for various m and n, because the complexity (i.e., area and energy consumption) of auxiliary circuits for choosing/steering m-bit segments and expanding a $2m$-bit result to a $2n$bit results scales linearly with m.

For applications where one of operands of each multiplication is often a fixed coefficient, we propose to precompute the bit-wise OR value of B$[n–1:m]$ and preselect between two possible m-bit segments

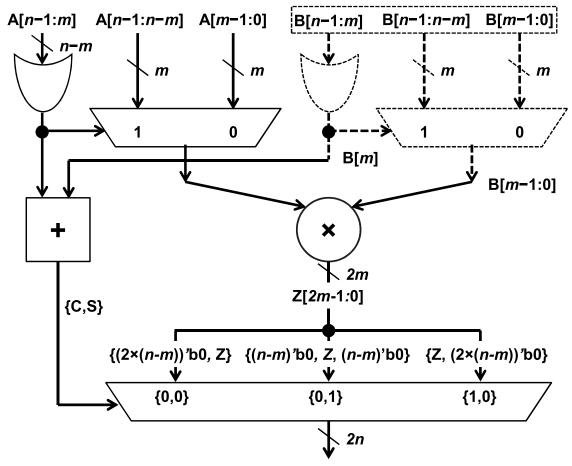

Fig. 5. Proposed approximate multiplier architecture; the logic and wires denoted by the dotted lines are not needed if B is preprocessed as proposed.

(i.e., B[n–1:n–m] and B[m–1:0]) in Fig. 5, and store them instead of the native B value in memory. This allows us to remove the n–m input OR gate and the m-bit 2-to-1 multiplexer denoted by the dotted lines in Fig. 5.

Finally, to support three possible starting bit positions for picking an m-bit segment where $m = n/2$, the two 2-to-1 multiplexers at the input stage and one 3-to-1 multiplier at the output stage are replaced with 3-to-1 and 5-to-1 multiplexers, respectively, along with some minor changes in logic functions generating multiplexer control signals; we will show this enhanced SSM design for $m = 8$ and $n = 16$ (denoted by ESSM8×8) can provide as good accuracy as SSM10×10 at notably lower energy consumption later.

III. Evaluation

A. Evaluation Methodology

In this section, we describe the methodology for evaluating computational accuracy and energy consumption of precise and various approximate multipliers. All the multipliers are described to support two 16-b inputs and 32-b output with Verilog HDL and synthesized using Synopsys Design Compiler and a TSMC 45-nm standard cell library at the typical process corner. We repeatedly synthesize each multiplier until it achieves the highest operating frequency. Then, we choose the frequency of the slowest one (i.e., 2 GHz) to resynthesize all other multipliers.

To evaluate computational accuracy, we take four sets of 16-b operand pairs from: 1) all possible pairs of 16-b values (denoted by random); 2) noise canceling algorithm [9] (denoted by audio); 3) 2-D optical coherence tomography [10] (denoted by image); and 4) isolated spoken digit recognition [11] (denoted by recognition), where each set is comprised of billions of operand pairs. To evaluate energy consumption, we use Synopsys PrimeTime-PX, which can estimate energy consumption of a synthesized design based on annotated switching activities from gate-level simulation. The input vectors for energy estimation are directly taken from the execution of multiplication intensive kernels in each application. We observe that the extracted input vectors exhibit inherent periodicity in the operand values applied to the multiplier. Thus, we take many such periods such that the number of vectors is 10 000 at least.

B. Computational Accuracy

Fig. 6 plots the probability distribution of computational accuracy of AM, DSM8×8, DSM6×6, SSM8×8, SSM10×10, ESSM8×8, and TRUN8×8 for four sets of operand pairs. We observe that the average computational accuracy of all these approximate multipliers is very high. For random, AM, DSM8×8, DSM6×6, SSM8×8, SSM10×10, ESSM8×8, and TRUN8×8 exhibit average compute accuracy of 96.7%, 99.7%, 97.8%, 98.0%, 99.6%, 99.0%, and 97.1%, respectively. However, for three classes of applications, AM, SSM8×8, and TRUN8×8 show notably deteriorating accuracy compared to SSM10×10 and ESSM8×8. For example, AM, SSM8×8, and TRUN8×8 can achieve computational accuracy higher than 95% only for 45%, 64%, and 61 % of operand pairs from image. In contrast, other approximate multipliers such as DSM8×8, SSM10×10, and ESSM8×8 can offer computational accuracy higher than 95% for 100%, 98%, and 98% of operand pairs, respectively. We expect that the high computational accuracy of SSM10×10 and ESSM8×8 for such a high fraction of operand pairs will barely impact quality of computing (QoC). The computational accuracy trend of audio and recognition is similar to that of image as shown in Fig. 6.

C. Quality of Computing

Table I tabulates the QoC obtained using different approximate multipliers relative to PM. To measure QoC, we use perceptual evaluation of speech quality (PESQ) [12] and structural similarity (SSIM) [13] for audio and image, respectively. In audio and image, PESQ and SSIM that are higher than 99% do not incur notable perceptual difference. Thus, our SSM10×10 and ESSM8×8 are sufficient for QoC while TRUN8×8 shows considerable QoC degradation for all three applications.

D. Energy and Area Analysis

Fig. 7 shows the average energy/op of AM, DSM8×8, DSM6×6, SSM8×8, SSM10×10, ESSM8×8, and TRUN8×8 for each of four sets of operand pairs, normalized to that of PM, respectively. The average energy/op of AM, DSM8×8, DSM6×6 across all four sets of operand pairs is 13%, 3%, and 28% lower than that of PM, respectively. In contrast, the average energy/op of SSM10×10 and ESSM8×8, which can offer sufficient computational accuracy and QoC, is 35% and 58% lower than that of PM. ESSM8×8 that is simplified to accept preprocessed fixed coefficients consumes 6% lower than the original ESSM8×8. We do not provide detailed energy/op analysis for TRUN8×8, SSM8×8 and SSM12×12, because: 1) TRUN8×8 and SSM8×8 may not provide sufficient computational accuracy and QoC regardless of very low energy/op; and 2) SSM12×12 does not exhibit notably higher computational accuracy and QoC than SSM10×10 while consuming much higher energy/op.

Analyzing the energy/op reduction of the evaluated multipliers, we first note that energy/op of AM can consume even lower than that of PM with higher target synthesis frequency [7]. However, the target synthesis frequency is limited by DSM8×8 while the energy/op of the multipliers should be compared at the same target frequency. Even though we remove DSM8×8 in the comparison, which allows us to increase the target synthesis frequency for AM, SSM10×10, ESSM8×8, and PM, we see that the relative energy/op difference between AM and SSM10×10 (or ESSM8×8) does not notably change. In other words, SSM10×10 and ESSM8×8 also benefits from higher target synthesis frequency, exhibiting lower energy/op.

Second, we observe that the power overhead of extra logic such as LODs and shifters is almost considerably larger than the 8×8 and 6×6 multipliers themselves for DSM8×8 and DSM6×6, although the bit width of the multiplier is a half of PM. This significantly reduces the overall benefit of DSM8×8 and DSM6×6. Furthermore, the fraction of logic gates switching in the 8×8 multiplier for DSM8×8

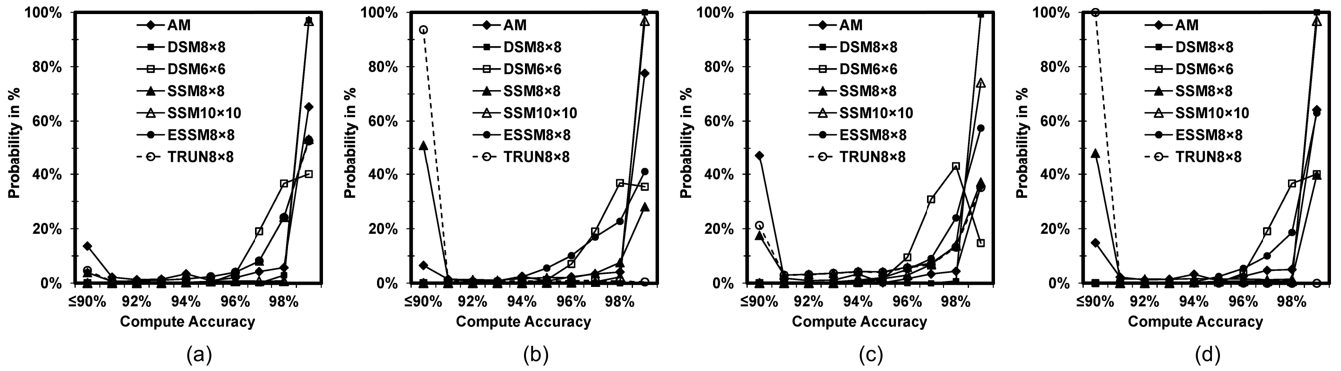

Fig. 6. Probability distribution of compute accuracy of AM2×2, DSM8×8, DSM6×6, SSM8×8, ESSM8×8, and 8×8 (truncated) for random vectors, audio/image processing, and recognition applications. (a) Random. (b) Audio. (c) Image. (d) Recognition.

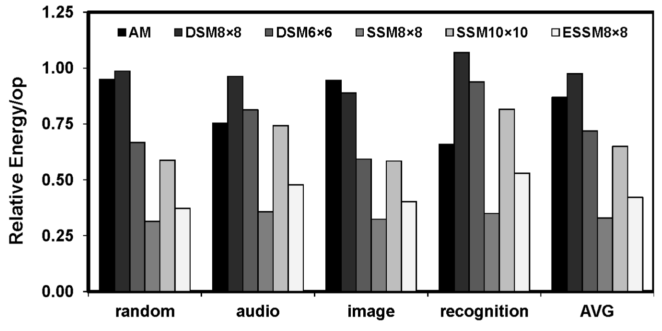

Fig. 7. Energy/op of AM, DSM8×8, DSM6×6, SSM8×8, SSM10×10, and ESSM10×10, relative to that of PM for four sets of operand pairs.

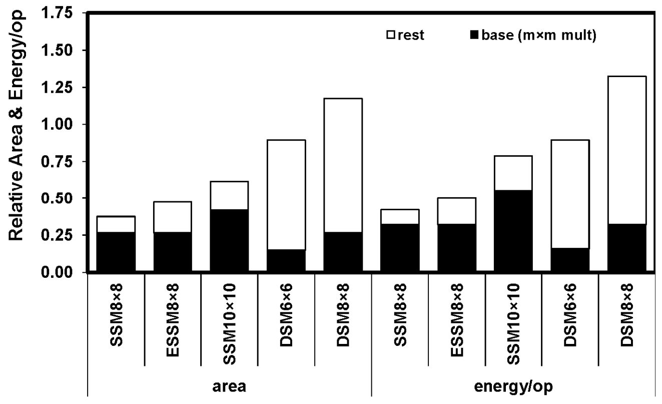

Fig. 8. Breakdown of area and energy/op of 16×16.

and DSM6×6 can be much higher than the 16×16 multiplier for PM. This is because DSM8×8 and DSM6×6 remove redundant sign-extension bits, which may incur no switching in many logic gates corresponding to MSB portion of PM before each multiplication.

Fig. 8 shows the breakdown of area and energy/op of 16×16 multipliers using DSM, SSM, and ESSM for various m, normalized to those of the 16×16 PMs. In the plot, each bar is comprised of area (or energy/op) of the base $m \times m$ multiplier (denoted by base $m \times m$ mult) and the remaining components (denoted by rest) such as the segment selection logic and multiplexers in SSM$m \times m$, and LODs and shifters in DSMm×m, respectively. The average energy/op in the plot is based on random.

First, the area of SSMm×m and DSMm×m are very closely correlated with the energy/op. For example, SSM10×10 consumes only 62% and 58% of PMs area and energy/op, respectively. Second, the base $m \times m$ multiplier contributes to consider-able area and energy/op for SSMm×m, while the remaining components dominate those for DSMm×m. For example, the 10×10 multiplier is responsible for 67% and 71% of total area and energy/op of SSM10×10, respectively. In other words, SSM is much more efficient than DSM because the overhead of the extra circuits to support SSM is small. Third, we observe that increasing the number of possible starting bit positions to take an m-bit segment from two to three does not notably increase the area and energy/op because both the area and energy/op are dominated by the base $m \times m$ multiplier. Finally, DSM6×6 does not significantly reduce area and energy/op compared with DSM8×8 because its area and energy/op are dominated by the peripheral gates as discussed earlier.

IV. Conclusion

In this brief, we propose an approximate multiplier that can trade-off accuracy and energy/op at design time for DSP and recognition applications. Our proposed approximate multiplier takes m consecutive bits (i.e., an m-bit segment) of an n-bit operand either starting from the MSB or ending at the LSB and apply two segments that includes the leading ones from two operands (i.e., SSM) to an $m \times m$ multiplier. Compared with an approach that identifies the exact leading one positions of two operands and applies two m-bit segments starting from the leading one positions (i.e., DSM), ours consumes much less energy and area than PM and DSM. This improved energy and area efficiency comes at the cost of slightly lower compute accuracy than PM and DSM. However, we demonstrate that the loss of small compute accuracy using SSM does not notably impact QoC of image, audio, and recognition applications we evaluated. On average, 16×16 ESSM8×8 can achieve 99% computational accuracy, respectively, with negligible degradation in QoC for audio, image, and recognition applications. On the other hand, 16×16 ESSM8×8 consumes only 42% energy/op of PM.

References

[1] R. K. Krishnamurthy and H. Kaul, "Ultra-low voltage technologies for energy-efficient special-purpose hardware accelerators," *Intel Technol. J.*, vol. 13, no. 4, pp. 102–117, 2009.

[2] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 1999, pp. 30–35.

[3] D. Menard, D. Chillet, C. Charot, and O. Sentieys, "Automatic floating-point to fixed-point conversion for DSP code generation," in *Proc. ACM Int. Conf. Compilers, Archit., Syn. Embedded Syst. (CASES)*, 2002, pp. 270–276.

[4] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *Proc. 47th IEEE/ACM Design Autom. Conf.*, Jun. 2010, pp. 555–560.

[5] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. 14th IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2009, pp. 195–200.

[6] C. H. Chang and R. K. Satzoda, "A low error and high performance multiplexer-based truncated multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1767–1771, Dec. 2010.

[7] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th IEEE Int. Conf. VLSI Design (VLSID)*, Jan. 2011, pp. 346–351.

[8] Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocessors Microsyst.*, vol. 35, no. 1, pp. 23–33, 2011.

[9] B. Widrow *et al.*, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.

[10] K. Zhang and J. U. Kang, "Graphics processing unit-based ultrahigh speed real-time Fourier domain optical coherence tomography," *IEEE J. Sel. Topics Quantum Electron.*, vol. 18, no. 4, pp. 1270–1279, Jul./Aug. 2012.

[11] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the liquid state machine: A case study," *Inform. Process. Lett.*, vol. 95, no. 6, pp. 521–528, Sep. 2005.

[12] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 1, pp. 229–238, Jan. 2008.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.