ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 3, March 2016

SEGMENTATION BASED ENERGY EFFICIENT APPROXIMATE MULTIPLIER FOR DIGITAL SIGNAL PROCESSING

A.MANSOORALIKHAN[1], P.NIRMALKUMAR[2]

[1]*Student, Dept. of Electronics and Communication, College of Engineering, Guindy*
[2]*Associate Professor, Dept. of Electronics and Communication, College of Engineering, Guindy*

***Abstract*—Now a days the energy consumption of DSP application devices are constantly increasing. The present fixed–point arithmetic and complex multiplication techniques consume a lot of energy while exhibiting tolerance for a large amount of noise and for computational errors too. So, we comparing all the arithmetic computations and multiplication, improving the energy efficiency is critical. To reduce this energy consumption we apply new multiplication techniques in the multipliers. The proposed multiplier compared with a precise multiplier can consume 58% less energy. In this brief, an energy efficient approximate multiplier is proposed which gives a tradeoff between computational accuracy and energy consumption. In the proposed multiplier architecture, the area has reduced compared to other multiplier architectures which process same number of bits. But the approximate multiplier architecture output possess a small amount of computational accuracy which is negligible for DSP applications.**

***Keywords*— digital signal processing, approximation, energy efficiency, multiplication, image compression.**

I. INTRODUCTION

For now a days embedded system and mobiles devices energy consumption is a critical design problem. Embedded and mobile computing devices are required to execute some other way to digital signal processing (DSP) and classification applications. Already more efforts were taken at various levels for improving energy efficiency. Compared to arithmetic operations, multiplication is the most time and power consuming operation. These are more difficulties for large operands and complex multiplication process. To further improve energy efficiency of executing such applications dedicated for specialized processors are often integrated in computing devices. The use of such specialized processors can improve energy efficiency by comparing with general-purpose processors at the constant voltage and technology generation. Algorithmic noise-tolerance (ANT) to compensate for process of degrades in the system output due to errors from soft computation [2]. ANT refers to algorithmic error-control schemes are derived from the system transfer function, input and output signal statistics.

Previous efforts that only focus on one dimension like voltage over scaling but the scalable effort approach focus on synergistic scaling along multiple dimensions that corresponding to different levels of design abstraction [4]. Such computational error tolerance has been derived by trading accuracy with energy consumption. An adaptive PCT (pseudo-carry compensation truncation) scheme is introduced in earlier efforts [6]. On comparing other truncation methods this method yields low error. But the leakage and dynamic power of PCT multipliers are higher than other truncated multipliers.

Later a novel architecture of multiplier with tunable error characteristics is proposed in [7]. The main advantage of this method, the architecture consumes compared to less dynamic power. This multiplier is faster and it needs less gate sizing to meet rising frequency constraints. But the drawback of the architecture is that the error rate is a bit high. Then other multiplier called iterative logarithmic multiplier is introduced which uses logarithmic number system [8]. The iterative logarithmic multiplier many number of correction terms as the error rate reduces with increase in number of correction terms but it increases the power consumption. Mostly error in the image or video is ignored by the human visual system. So that output is numerically approximate rather than accurate. Having an advantage of approximation in multimedia application, the reduced complexity with minimal error and also achieving the minimum power dissipation, while approximating the multiplier by reducing the area and power with minimum error.

Finally, these algorithms are initially designed with floating-point (FP) arithmetic, but they are changed to fixed-point arithmetic due to the area and power cost of supporting floating point units in embedded computing devices. The designed approximate multiplier is used in any digital signal processing like FIR filter and JPEG image compression using DCT. These approximate multipliers are the basic build functionally verified.

II. RELATED WORKS

R. Hegde and N. R. Shan hag in 1999 proposed a framework for energy efficient digital signal processing [2]. The supply voltage is scaled further, which is the critical voltage required to fix the critical path delay to the throughput. This introduction of input-dependent errors leads to degrade in the algorithmic performance, which is compensated through the Algorithmic Noise Tolerance (ANT) schemes. A prediction based error-control scheme is proposed to the filter algorithm performance and the presence of errors computations .The Scaling of CMOS technology has made possible substantial reduction in energy dissipation and hence has led to increase of low cost VLSI systems with increasing high levels of integration. This technology reduction in energy dissipation has made possible due to energy-efficient design techniques at all possible levels of design hierarchy. ANT refers to algorithmic error-control schemes derived from the system transfer function, input and output signal statistics. To avoid the problem in the algorithmic performance, ANT scheme is applied. Because deep submicron (DSM) noise the proposed technology is used to improve the performance of DSP applications.

V. K. Chippa et.al.in 2010 proposed [4] scalable effort hardware design as an approach to tap the reservoir of algorithmic recover and translate it into highly efficient hardware implementations. The basic idea of the scalable effort design is to identify mechanisms at each level of design abstraction like circuit, architecture and algorithm that can be used to varying the computational effort. A second major idea of the scalable effort design approach is that fully based on the potential of algorithmic recover requires synergistic cross-layer optimization of scaling mechanisms are identified at different levels of design abstraction[4] .Unlike previous efforts that only focus on one dimension (e.g., voltage over scaling) but the scalable effort approach espouses synergistic scaling along multiple dimensions that correspond to different levels of design abstraction (circuit, micro-architecture, and algorithm). This leads to higher energy savings through better utilization of the reservoir of algorithmic resilience as borne out by the experimental results presented in this paper.

C. H. Chang and R. K. Satzoda in 2010 proposed to ignore the least significant columns in the partial product (pp) bit matrix obtained from an n-bit multiplication [6]. A small amount of additional hardware is added to compensate for the truncation errors. This method achieves lower average and spread of errors by means of adaptive pseudo carry compensation and a constant bias that is independent of supply. By especially the symmetry of the multiplexer-based array multiplier the partial product bits are generated by the multiplexers. The truncated multiplier can be assembling in a carry-save adder format to reduce the area and improve the speed over than other truncated array multipliers. The truncated multiplication scheme is applied to a DCT based image compression algorithm and JPEG.

P. Kulkarni et.al. in 2011 [7] proposed multiplier architecture for 2x2 inaccurate multiplier done by Karnaugh map simplification which was extended up to 16 bit where the error is constant for all architecture. It calculated maximum error and probability of error occurrence. The designed multiplier block can be used to build arbitrarily large power efficient inaccurate multipliers. Finally, the operation of this multiplier for image compression and DSP applications and compare it with voltage scaling and bit width truncation based method and also project power savings from different software configurations and compared with their approach.

Z. Babi´c, A. Avramovi´c, and P. Bulic in 2011 proposed an iterative logarithmic multiplier [8].This iterative logarithmic technique is an achievement of an arbitrary accuracy for multipliers. It is fully based on the Mitchell‟s algorithm number representation but does not use the approximation logarithm. The implemented proposed technique is efficient and simple and the error rate is very low. This method in parallel circuits was given for correction the error and hardware is not much power consuming.

III. BIT SELECTION OF PROPOSED MULTILPIER ARCHITECTURE:

The proposed multiplier consists of *m*-bit segment as *m* contiguous bits starting with the leading one in *n*-bit positive operands. The dynamic segment method (DSM) is in contrast to static segment method (SSM).Taking with two *m*-bit segments from two *n*-bit operands, we can perform multiplication using *m* ×*m* multiplier. The Fig. 1 shows an example of a multiplication after taking 8-b segments from 16-b operands. In this example, we can achieve accuracy 99.4% for a 16×16 multiplication even with an 8×8 multiplier. The figure1 shows the 16x16 bit multiplier using the two 8 bit segments.

×					**0111**	**0111**	**0101**	1010
					0000	0011	**1011**	**1011**
=	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.

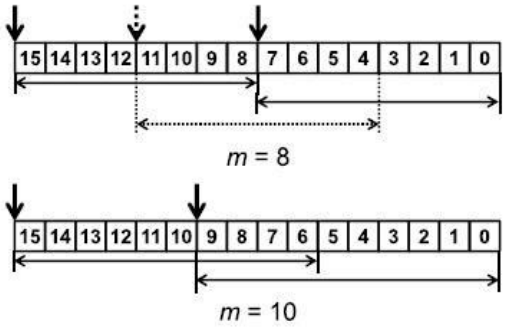

Fig.2. Possible starting bit positions of 8-b and 10-b segments indicated by arrows; the dotted arrow is the case for supporting three possible starting bits.

The segments are taken depending on the operand itself. Here the m=8 bits are taken in 3 styles from 0 to 7, 4 to

11 and 8 to 15. Similarly the m=10 bits are taken in two styles as shown below the first one. Such a multiplication approach have little negative impact on computational accuracy because it can eliminates redundant bits (i.e. sign-extension bits) while feeding the *m* significant bits to the multiplier we will provide detailed evaluations of computational accuracy for various *m*, and furthermore, an *m*×*m* multiplier consumes much less energy than an *n*×*n* multiplier.Because the complexity an energy consumption of multipliers slightly increases with *n*. The 4 × 4 and 8 × 8 multipliers consume almost less energy than a 16×16 multiplier. However, a DSM requires: 1) two leading one detector; 2) two *n*-bit shifters to fix the leading one position of each *n*-bit operand to the MSB position of each *m*-bit segment to apply the *m*×*m* multiplier; and 3) one 2*n*-bit shifter to expand a 2*m*-bit result to 2*n* bits. The area and energy penalties associated with in DSM are to capture an *m*-bit segment starting from an arbitrary bit position in an *n*-bit operand for the leading one bit can be anywhere. Thus, we proposed a multiplier to limit possible starting bit positions to taken an *m*-bit segment from in *n*-bit operand to two or three at most in SSM. For taking two *m*-bit segments from two *n*-bit operands for a multiplication using an *m*-bit SSM, we have four possible combinations. For a multiplication choose the *m*-bit segment that contains the leading one bit of the each operand and apply the chosen segments from both operands to the *m*×*m* multiplier. The SSM simplifies the circuit that chooses *m* bit segments and towards to the *m*×*m* multiplier by replacing two *n*-bit LODs and shifters for DSM with two (*n*–*m*)-input OR gates and *m*-bit 2-to-1 multiplexers. The first (*n*–*m*) bits starting from the MSB are all zeros and the lower *m*-bit segment must contain the leading one. Furthermore, the SSM method also allows replacing the 2*n*-bit shifter used for the DSM method with a 2*n*-bit 3-to-1 multiplexer. Because the segment for each operand are taken from one of two possible segments in an *n*-bit operand and 2*m*-bit result can be expanded to 2*n*bit result by left-shifting and that we get 2*m*-bit result by one of three possible shift amounts:

1) There is no shift when both segments are taken from the lower *m*-bit segments;

2) (*n*–*m*) shift when two segments are taken from the upper and lower ones;

3) 2×(*n*–*m*) shift when both segments are taken from the upper ones.

x					01xx	xxxx	xxxx	xxxx
					0001	xxxx	xxxx	xxxx
=					0000	0000	0000	0000
x					01xx	xxxx	xxxx	xxxx
					0000	0000	01xxx	xxxx
=	0000	0000					0000	0000
x					0000	0000	01xx	xxxx
					01xx	xxxx	xxxx	xxxx
=	0000	0000					0000	0000
x					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.

x					0000	0001	xxxx	xxxx
					0000	0010	xxxx	xxxx

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

IV. PROPOSED MULTIPLIER ARCHITECTURE

SSM allowing to taking an *m*-bit segment from two possible bit positions of an *n*-bit operand. The main advantage is scalability for different m and *n*, because the complexity area and energy consumption of auxiliary circuits for choosing/steering *m*-bit segments and expanding a 2*m*-bit result to a 2*n* bit results scales linearly. For applications where one of operands of each multiplication given a fixed coefficient, we propose to pre compute the bit-wise OR value of B[*n*–1:*m*] and preselect between two *m*-bit segments (i.e., B[*n*–1:*n*–*m*] and B[*m*–1:0]) in Fig. 5, and store them equivalent of the native B value in memory. This architecture 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. Three possible starting bit positions for taking an *m*-bit segment where *m* = *n*/2, the two 2-to-1 multiplexers at the input stage and the output stage one 3-to-1 multiplier are replaced with 3-to-1 and 5-to-1 multiplexers. Small minor changes in logic functions producing multiplexer control signals. We will show this enhanced SSM design for *m*= 8 and n = 16 by ESSM8×8(Extended Static Segment Method) can provide as good accuracy as SSM10×10 at notably lower energy consumption later.

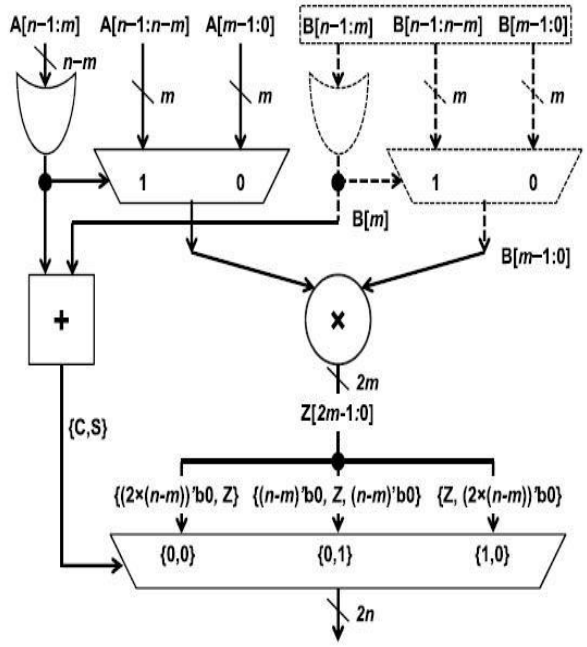

Fig.5. Proposed approximate multiplier architecture; the logic and wires denoted by the dotted lines

V. ACCURACY

Computational accuracy of PM,DSM8×8,SSM8×8, SSM10×10 and ESSM8×8 with four sets of operand pairs. We observed that the average computational accuracy of all these approximate multipliers is very high. For random DSM8×8, SSM8×8, SSM10×10, ESSM8×8 and exhibit average compute accuracy of 99.4%,26.8%,76.6%,94.8% respectively. As expected DSM8×8 give accuracy of 99.4% near to original precise multiplier value.Already existing SSM technique SSM10×10 output performs SSM8×8.Since the MSB containing lots of zero, SSM8×8 gives lowest possible accuracy and the proposed ESSM8×8 technique eliminates the problem of low accuracy in SSM at this same time its output performs SSM10×10.So in general ESSM improved version of SSM8×8 that give better performance than SSM10×10. This accuracy is calculated from the random inputs that are taken.

TABLE I
ACCURACY OF EACH TECHNIQUE

METHODS	ACCURACY
DSM8×8	99.4%
SSM8×8	26.8%
SSM10×10	76.6%
ESSM8×8	94.8%

VI. CONSTRUCTION OF 2D DCT STRUCTURE

DCT is generally a frequency domain transformation of a signal or image which is extensively used in image and video compressions. An 'N' point DCT is obtained from the following:

$$X(k) = \frac{c(k)}{2}\sum_{i=0}^{N-1} x(i)\cos\left(\frac{(2i+1)k\pi}{2N}\right) \qquad (1)$$

DCT is an orthogonal transform, thus it can be transformed back to original samples using Inverse Discrete Cosine Transform (IDCT). The 'N' point IDCT is given by the following:

$$x(i) = \frac{1}{N}\sum_{k=0}^{N-1} c(k)X(k)\cos\left(\frac{(2i+1)k\pi}{2N}\right) \qquad (2)$$

Here,

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & otherwise \end{cases} \qquad (3)$$

The one dimensional transformation is to be transposed followed by another one dimensional transformation and quantisation to complete the two dimensional DCT operation on images as shown in Fig.6.

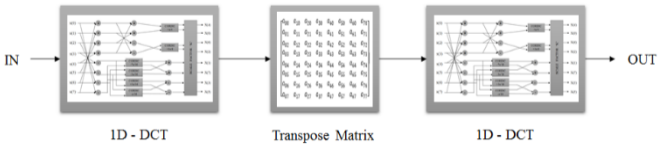

Fig.6. Entire block diagram of an 8-point 2D DCT architecture

A. Simulation result:

The proposed multiplier architecture ESSM technique was designed using Verilog HDL and Isim simulator. The obtained outputs are from the input taken at random.

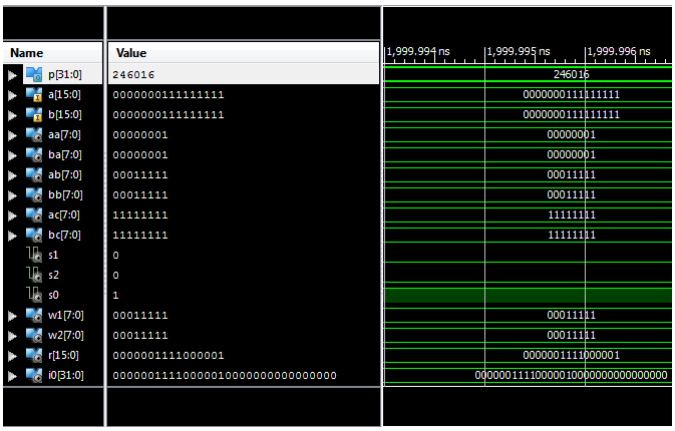

Fig.6.Simulated ESSM output

A trial image of 8x8 was set to provide as input to the 1D DCT row-wise and column-wise. The coding for 1D DCT was written using Verilog HDL and simulated using Isim. For further advancement, the top module was written using

System Verilog that provides the test inputs for the DCT module in parallel. The collected information is stored in an array. The inverse transform of the image is taken using Matlab and stored as image format. Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) is then calculated for the obtained image.

1) *Output from Xilinx ISE*: A DCT of 8-bins that uses the ESSM architecture was constructed in Verilog HDL. The inputs are drawn from the image row-wise and then column-wise. The outputs are obtained as expected with few errors due to quantisation, which is claimed to be negligible.

2) Using ModelSim: As there is a limitation in Xilinx ISE that it cannot offer looping over modules or 2D array like structures, the top module is tested using the 8x8 test image as inputs. It performs 1D DCT row-wise, transposed and then another 1D DCT column-wise. The final output is then stored.

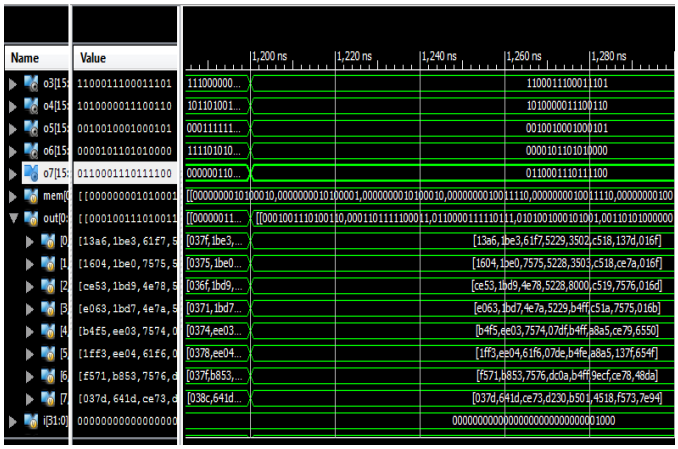

Fig.7.simulated DCT using ESSM method

3) *2D DCT realisation in Matlab:* The file stored in the previous step is opened using Matlab and then inverse transformed to obtain the original image. In the Fig. 8 it appears to be damaged because the input is a simple 8x8 array and the pixel changes are noticeable. If a complete 64x64 array is produced, the errors become unnoticed without damaging the quality of the image.

Fig.8.Output of 2D DCT of an image using ESSM8x8 method

B. Results Comparison

The simulation process is performed for multiplier architecture. The comparison performed with fair norms using Cadence RTL compiler in Linux platform.

1) *Area, Power and Delay*: The comparison is shown clearly in Table II. Comparing ESSM with PM the Power Delay Product (PDP) is reduced by 89.4% and Area Delay Product (ADP) by 78.7%.
2) *Noise and Compression Ratio*: The increase in MSE and PSNR does not affect much and it can be clearly observed from the Fig.8. The Compression ratio is calculated as 1.333 with 25% compression.

TABLE II
AREA, POWER, DELAY COMPARISON

ARCHITECTURE	Area (μm^2)	Power (μW)	Delay (ns)
PM	20491	2260	15.340
DSM8×8	15225	1075	10.009
SSM8×8	6310	349	7.665
SSM10×10	9587	700	10.991
ESSM8×8	7681	420	8.716

TABLE III
DATA COMPRESSION AND NOISE ANALYSIS

ARCHITECTURE	PDP (10^{-12})	ADP (10^{-12})	MSE	PSNR (dB)
PM	34.6684	314.3319	0.0315	28.033
DSM8×8	10.7596	152.3870	0.0925	29.235
SSM8×8	2.6750	48.3661	0.0224	32.314
SSM10×10	7.6937	105.3707	0.0811	29.521
ESSM8×8	3.6607	66.9475	0.0327	28.492

C. Area Analysis

Area of approximate multiplier are analysed by normalized

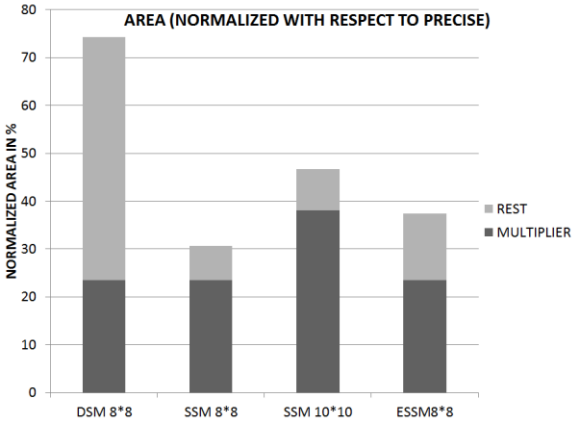

Fig.9. Area analysis with precise multiplier

percentage saving with respect to precise multiplier. As shown in figure.9.DSM8×8,SSM8×8,SSM10×10,ESSM8×techniques gives for 74.1%,30.7%,46.7%,37.4 % respectively. DSM occupies largest possible area of 74.1% out of this multiplier occupies 23.5% and rest of other 50.8%,thus we can see that the shifters, LODs consumes two times more than the multiplier itself. This output performs the saving of approximate multiplication. Among SSM techniques 10x10 occupies largest area followed by ESSM and SSM8x8 respectively. The large area of SSM10x10 is due to 38.14% area occupied by 10 bit multiplier .SSM8x8 occupy 8% less area than ESSM due to savings from simple multiplexer and shifter units .Thus EESM occupies slightly improved area than SSM8x8 ,but less than the rest of other technique.

D. Power Analysis

As shown in figure.10.DSM8×8,SSM8×8,SSM10×10, ESSM8×8 techniques power consumption gives for 47.5%,15.5%,31%,18.5 percentage respectively.DSM8x8 consume largest power this is already expected due to large LODs and shifters.SSM10x10 occupies 15% more power than SSM 8x8 because of large size of multipliers. ESSM only consumes 3% more power than SSM8x8.Thus power consumption of proposed ESSM almost equal to that SSM.

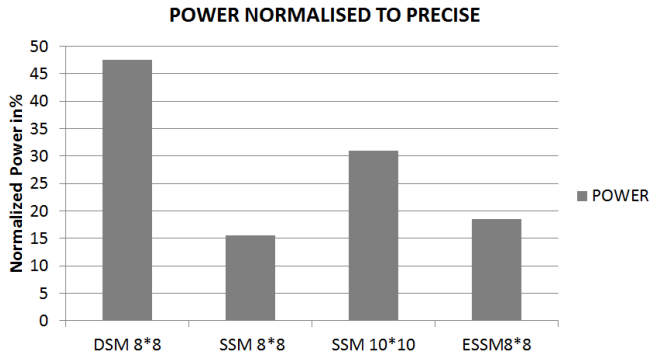

Fig.10.Power analysis with precise multiplier

VII. CONCLUSION

In this brief, we proposed an approximate multiplier that can trade off accuracy and energy output at design time for DSP applications. The proposed multiplier when compared to other multipliers approach that identifies the exact leading one positions of two operands and applies two m-bit segments starting from the leading one positions. This multiplier consumes much less energy and area compared to DSM and SSM. This improved energy and area efficiency comes at the cost of slightly lower compute accuracy than SSM and DSM. However, we evaluate that the loss of small computational accuracy using SSM does not notably impact of image, audio, and recognition applications. The proposed multiplier architecture ESSM8×8 can achieve 94.8% computational accuracy and with negligible degradation in for audio, image, and recognition applications. This can be further extended in FIR filter applications.

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 floatingpoint 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. Babic´, A. Avramovi´c, and P. Bulic´, "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.