Design and Implementation of Energy Efficient Approximate Multiplier

B.Anish Fathima
PG Scholar
Department of ECE
Government College of Technology, Coimbatore, Tamil Nadu, India

C.Vasanthanayaki, PhD
Associate Professor
Department of ECE
Government College of Technology, Coimbatore, Tamil Nadu, India

ABSTRACT
Modern Digital signal processing and image processing applications are aiming towards energy efficiency. The prime arithmetic operation performed for these processes is multiplication. Hence energy efficiency of multiplication is critical. Since many digital applications use fixed- point arithmetic, it exhibits computational error tolerance. In this brief, a multiplier is proposed that can trade-off computational accuracy with energy consumption. Segmenting the original operands with significant bits and performing the multiplication only for those segments is the main principle. The proposed method of approximate multiplier consumes lesser power and hence notably lesser energy with average computational error of ~1%, when compared to the existing approximate multipliers with similar principle. Further optimization of the proposed multiplier is also done which improves the average computational accuracy along with a considerable reduction in the area consumed by the proposed multiplier.

Keywords
VLSI, Embedded Systems, Integrated Circuits, DSP

1. INTRODUCTION
The theme for modern VLSI design and Embedded Systems research is Green Technology, which explores the ability of VLSI and embedded circuits and systems to positively impact the environment. Demand for developing energy-efficient VLSI architectures, designing intelligent monitoring and control systems using integrated circuits or embedded systems to leverage novel green technologies.

To improve energy efficiency of such computing devices, many efforts have already been suggested 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. Further improvement energy efficiency for executing such applications can be achieved by, dedicated specialized processors often integrated in computing devices. It has been reported that the use of such specialized processors can improve energy efficiency by 10–100 times 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 only slight degradation in overall DSP quality and classification accuracy instead of a severe failure. Such computational error tolerance has been exploited by trading accuracy with energy consumption. 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. Even though this conversion process leads to some loss of computational accuracy, it does not highly affect the quality of DSP and the accuracy of classification applications due to computational error tolerance.

The multipliers play a significant role in arithmetic operations in DSP applications [10]. Multiplication is a hardware intense operation, and the main areas of interest are higher speed, lower cost, and less VLSI area. Two significant yet often conflicting design criteria are power consumption and propagation delay. Considering these constraints, the development of low power multiplier is of great interest. Many research efforts in the design of multiplier have been introduced to obtain energy efficiency in VLSI circuits [3].

The rest of this thesis is organized as follows. Section II details the emergence of various approximate multiplication techniques. Section III gives a detailed description about the proposed approximate multiplication technique. Section IV analyzes simulation and implementation results. Finally, Section V concludes the thesis.

2. APPROXIMATE COMPUTATION TECHNIQUES
Approximate computing is an emerging design paradigm that enables highly efficient hardware and software implementations by exploiting the inherent resilience of applications to in-exactness in their computations. Several previous efforts have explored approximate computing in software and hardware with promising results. Software techniques typically improve performance by skipping computations or reducing the use of costly operations such as inter-thread synchronization, whereas hardware techniques modify the design at various levels of abstraction to introduce tradeoffs between output quality and efficiency. These efforts have established the significant potential of approximate computing. To evaluate the quality of a particular n-bit approximate arithmetic circuit, several error criteria can be used such as Error magnitude, Relative error, Average error magnitude and Error probability (error rate) [2].

Approximate circuits have been considered for error-tolerant applications that can tolerate some loss of accuracy with improved performance and energy efficiency. Applications including recognition, multimedia and data mining are inherently error-tolerant and do not require a perfect accuracy in computation. For such applications, approximate circuits may play an important role as an effective alternative for reducing power, area and delay in digital systems that can tolerate some loss of accuracy, thereby achieving better performance in energy efficiency. Such exhibit the interesting property of high inherent algorithmic resilience to "errors" from both extrinsic and intrinsic sources. These algorithms [4] are designed to process large amounts of input data that has significant redundancy and may frequently contain significant imperfections or noise. The algorithms typically use iterative, successive refinement techniques, which imparts them with a self-healing nature since subsequent iterations may correct errors introduced in previous iterations.

To improve energy efficiency of multipliers, previous studies have explored various techniques exploiting computational error tolerance. They can be classified into three categories: Aggressive voltage scaling; Truncation of bit-width; Use of inaccurate building blocks. In [9], a new systematic approach based on the concept of scalable effort hardware, for the design of efficient hardware implementations for algorithms that demonstrate inherent error resilience was presented. The efficiency of digital multiplication can be improved tremendously by truncation methods provided precise outputs are not required for the operation. In the paper [8], a new multiplexer based truncation scheme with lower average and mean square errors than existing truncation methods was proposed. In [7], with a mean error of 1.39% − 3.35% and power savings between 30%−50%, the under designed multiplier architecture presented allows for trading of accuracy for power. In the paper [6], we have proposed a framework for designing a low power multiplier using energy efficient full adder. In the paper [5], recent progress on approximate computing is reviewed, with a focus on approximate circuit design, pertinent error metrics, and algorithm-level techniques.

3. PROPOSED APPROXIMATE MULTIPLICATION TECHNIQUE

3.1 Segment Based Multiplication

Energy efficiency of multiplication is a critical objective. Since many digital applications use fixed- point arithmetic, it exhibits tolerance for computational errors. In this project, a multiplier is proposed that can tradeoff computational accuracy with energy consumption.

"Segmenting the original operands with significant bits and performing the multiplication only for those segments is the main principle". The proposed technique performs approximate multiplication exploiting significant segments of operands unlike the existing techniques such as aggressive voltage scaling; truncation of bit-width and use of inaccurate building blocks for approximate multiplication.

The general procedure for the segmentation based multiplication is given below: If two n- bit operands are to be multiplied then the general procedure for multiplication through the method of segmentation is given below:

- Given two n-bit (say 16-bit) operands
- Select m-bit (say 8-bit)segment from each n-bit operand
- This Segment must contain the leading one bit
- Multiply these two m-bit segments
- Expand the 2m-bit product to 2n-bit product

3.2 Methods

Based on the way in which a segment is selected from the operand, three methods are followed:

- Dynamic Segment Method (DSM)
- Static Segment Method (SSM)
- Enhanced Static Segment Method (ESSM)

DSM is an earlier existing segmentation method. The proposed technique of approximate multiplication includes SSM and ESSM. The complete description about each method is given below in detail.

3.2.1 Dynamic Segment Method (DSM)

In order to motivate and describe the proposed multiplier [1], we define an m-bit segment as m contiguous bits starting with the 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 ×m multiplier. In this method, we can achieve 99.4% accuracy for a 16 × 16 multiplication even with an 8 × 8 multiplier. This method can capture m-bit segments starting from the exact leading one bit position as shown in Figure 1.

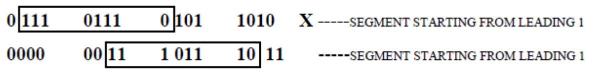

0 111 0111 0 101 1010 X -----SEGMENT STARTING FROM LEADING 1
0000 00 11 1 011 10 11 -----SEGMENT STARTING FROM LEADING 1

Figure 1: An example of Dynamic Segment Method

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. Furthermore, an m×m multiplier consumes much less energy than an n×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.

A DSM requires: Two LODs; 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×m multiplier; and One 2n-bit shifter to expand a 2m-bit result to 2n bits. Hardware requirements incur considerable area and energy penalties completely negating the energy benefit of using the m × m multiplier; The area and energy penalties associated with three requirements 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, 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 is proposed.

3.2.2 Static Segment Method (SSM)

The proposed method for approximate multiplication is the Static Segment Method (SSM).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×m multiplier. Two design architectures using two different numbers of bits for the segment is given as shown in figure 2 and 3 below:

- SSM_8X8 – where the segment size (m) is 8 bit
- SSM_10X10 – where the segment size (m) is 10 bit

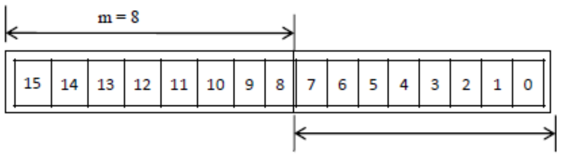


Figure 2: Segmenting in SSM where m=8

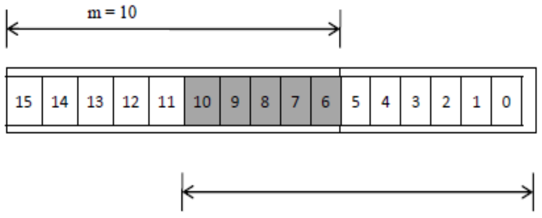


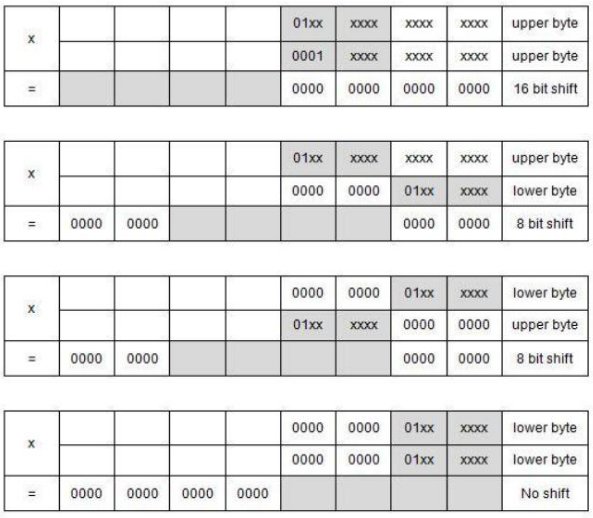
Figure 3: Segmenting in SSM where m=10

The SSM greatly simplifies the circuit that chooses m-bit segments and steers them to the m×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 2nbit result by left-shifting the 2m-bit result by one of three possible shift amounts as shown in Figure 4:

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;

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



x					01xx	xxxx	xxxx	xxxx	upper byte
					0001	xxxx	xxxx	xxxx	upper byte
=					0000	0000	0000	0000	16 bit shift

x					01xx	xxxx	xxxx	xxxx	upper byte
					0000	0000	01xx	xxxx	lower byte
=	0000	0000					0000	0000	8 bit shift

x					0000	0000	01xx	xxxx	lower byte
					01xx	xxxx	0000	0000	upper byte
=	0000	0000					0000	0000	8 bit shift

x					0000	0000	01xx	xxxx	lower byte
					0000	0000	01xx	xxxx	lower byte
=	0000	0000	0000	0000					No shift

Figure 4: Three possible shifting processes in SSM

Figure 5 shows SSM 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 2nbit results scales linearly with m.

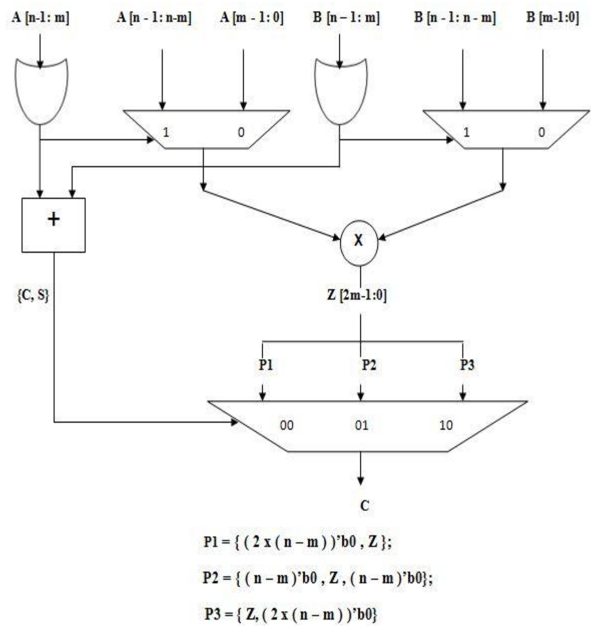
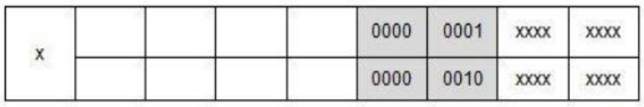


Figure 5: SSM architecture

For applications where one of operands of each multiplication is often a fixed coefficient, we propose to pre compute the bit-wise OR value of B[n–1:m] and preselect between two possible m-bit segments (i.e., B[n–1:n–m] and B[m–1:0]) in Figure 5, and store them instead of the native B value in memory.

The accuracy of an SSM with m = n/2 can be significantly low for operands shown in figure 6:



x					0000	0001	xxxx	xxxx
					0000	0010	xxxx	xxxx

Figure 6: An example for low accurate operands

Here 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.

3.2.3 Enhanced Static Segment Method (ESSM)

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 multiplexer 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. These changes lead to an enhanced architecture method called Enhanced Static Segment Method (ESSM) which is shown in figure 7.

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.

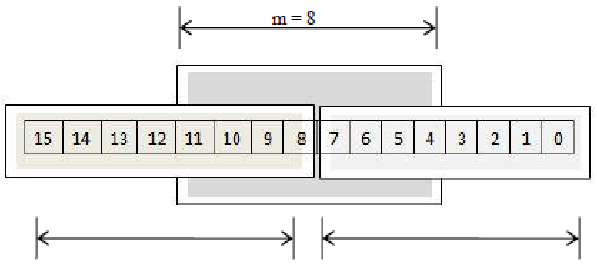


Figure 7: Segmenting in ESSM where m=8

3.2.4 Optimization of Proposed Architecture

The proposed ESSM method of multiplication though consumes lesser energy than SSM method of multiplication, it has large area overhead. Thus further optimization is done with a modification to the SSM algorithm. As a result of this modification, SSM method can achieve a better computational accuracy with considerable reduction in area overhead. This optimized SSM is named Improved Static Segment Method (ISSM).

4. SIMULATION AND IMPLEMENTATION RESULTS

In this project, design and implementation of energy efficient multiplier architectures were simulated using Verilog Hardware Description Language. The Xilinx Design ISE (Integrated Software Environment) 13.2 was used to analyze the performance parameters of these architectures. The performance parameters include power, delay and area. Spartan 6 was considered as the operating device.

4.1 Comparative Analysis of Approximate Multipliers

The comparison here involves around the performance parameters of approximate multiplier methods discussed so far. Table 1 represents the average computational error (in %) of the product obtained from different methods discussed. On an average, 30 different pairs of operands are considered for this tabulation. ESSM proves to be better in accuracy.

Table 1: Analysis of Average Computational Error

Segment Method	Avg. computational Error (%)
SSM_8X8	~6
SSM_10X10	~1
Proposed Methods	
ESSM_8X8	~1
ISSM_8X8	~3

Table 2 represents the analysis of performance parameters such as delay, power, energy and area consumed for all the four methods. ESSM proves to be better in energy whereas ISSM reduces the area overhead.

Table 2: Analysis of Delay, Power and Area

Segment Methods	Delay (ns)	Power (mW)	Energy (pJ)	Area (No. of LUTs)
SSM_8X8	13.073	29	379.12	28
SSM_10X10	12.451	27	336.18	28
Proposed Methods				
ESSM_8X8	13.691	27	**369.66**	59
ISSM_8X8	14.380	30	431.40	**38**

5. CONCLUSION

In this brief, an approximate multiplier that trades off accuracy and energy has been proposed. The proposed method of 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 applies these two segments that include the leading ones from two operands (i.e., SSM) to an m×m multiplier. The proposed Enhanced Static Segment Method consumes lesser power and hence notably lesser energy with average computational error of ~1%, when compared to a true multiplier. Further optimization of the proposed multiplier is also done which improves the average computational accuracy along with a considerable reduction in the area consumed by the proposed multiplier.

Thus the proposed approximate multiplier can lead to better quality in signal processing and image processing computations when compared to other forms of approximate multiplier such as truncated multiplier. The most important aspect of this proposed method is that it is maintaining the quality even in energy efficient devices. It can also lead to better performances when one of the operands of the multiplication is kept fixed.

6. REFERENCES

[1] Srinivasan Narayanamoorthy, Hadi Asghari Moghaddam, Zhenhong Liu, Taejoon Park, and Nam Sung Kim, July 2014, “Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Early Access Articles, pp. 1 – 5.

[2] Zdenek Vasicek and Lukas Sekanina, April 2014, “Evolutionary Design of Approximate Multipliers under Different Error Metrics”, Design and Diagnostics of Electronic Circuits & Systems, 17th International IEEE Symposium, pp. 135 – 140.

[3] K. S. Ganesh Kumar, J. Deva Prasannam, M. Anitha Christy, March 2014, “Analysis of Low Power, Area and High Performance Multipliers for DSP applications”, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 3,pp.278-382.

[4] Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy and Anand Raghunathan, May- June 2013, “Analysis and Characterization of Inherent Application Resilience for Approximate Computing”, IEEE Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE, pp. 1- 9.

[5] Jie Han, Michael Orshansky, May 2013, “Approximate Computing: An emerging Paradigm for Energy-Efficient Design”, Test Symposium (ETS), 2013 18th IEEE European, pp. 1-6.

[6] A.Kishore Kumar, D. Somasundareswari, V. Duraisamy and T. Shunbaga Pradeepa, February 2013 , “Design of Low Power Multiplier with Energy Efficient Full Adder Using DPTAAL” , Hindawi Publishing Corporation,VLSI Design, Volume 2013, pp. 1-9.

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

[8] C.H. Chang and R. K. Satzoda, December 2010, “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.

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

[10] Sjalander.M, Larsson-Edefors,P., Aug. 31 2008-Sept. 3 2008, “High-speed and low-power multipliers using the Baugh-Wooley algorithm and HPM reduction tree”, Electronics, Circuits and Systems, 2008, ICECS-2008. 15th IEEE International Conference, pp. 33 – 36.