ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

An Area Efficient Decomposed Approximate Multiplier for DCT Applications

K.Mohammed Rafi [1], M.P.Venkatesh [2]

P.G. Student, Department of ECE, Shree Institute of Technical Education, Tirupati, India[1]

Assistant Professor, Department of ECE, Shree Institute of Technical Education, Tirupati, India [2]

ABSTRACT: The energy consumption of DSP application devices have been steadily increasing now-a-days. The present multiplication techniques consume a lot of energy. There is a need to reduce this energy consumption by applying new multiplication techniques. Compared with a precise multiplier, the proposed multiplier can consume 58%less energy output with average computational error of nearly 1%.Finally,we demonstrate that such a computational error does not notably impact the quality of DSP and the accuracy of classification applications.

KEYWORDS: approximation ,energy efficient, multiplication.

I. INTRODUCTION

Energy-efficiency has become the paramount concern in design of computing systems. At the same time, as the computing systems become increasingly embedded and mobile, computational tasks include a growing set of applications that involve media processing (audio, video, graphics, and image), recognition, and data mining [1]. A common characteristic of the above class of applications is that often a perfect result is not necessary and an approximate or less-than-optimal result is sufficient. It is a familiar feature of image processing, for example, that a range of image sharpness/resolution is acceptable. In data mining, simply a good output of, say, a search result, is hard to distinguish from the best result. Such applications are imprecision-tolerant [2].

Energy efficient embedded systems consist of a heterogeneous collection of very specific building blocks, connected together by a complex network of many dedicated busses and interconnect options. The trend to merge multiple functions into one device makes the design and integration of these "systems-on-chip" (SOC's) even more problematic [3]. Yet, specifications and applications are never fixed and require the embedded units to be programmable .The topic of this paper is to give the designer architectures and design techniques to find the right balance between energy efficiency and flexibility. The key is to include programmability (or reconfiguration) at the right level of abstraction and tuned to the application domain. The challenge is to provide an exploration and programming environment for this heterogeneous architecture platform [4] -[8].

Paper is organized as follows. Section II describes approximate multiplier exploiting significant segments of operands. In Section III , simulation of existing and proposed systems are shown and outputs are verified on Xilinx Spartan 3E kit. Finally, Section IV presents conclusion.

II. RELATED WORK

In order to motivate and describe our proposed multiplier, 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 $m \times m$ 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. The figure 1 shows the 16x16bit multiplier using the two 8 bit segments.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Here from the first 16 bit operand ,the block where the bits starting with 1's are taken as 8 bit segment .Similarly from the second 16 bit operand rather than taking from the initial bit position starting with 0's.It will reduce the time consumption as well as improve the accuracy of this process.

					0111	**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

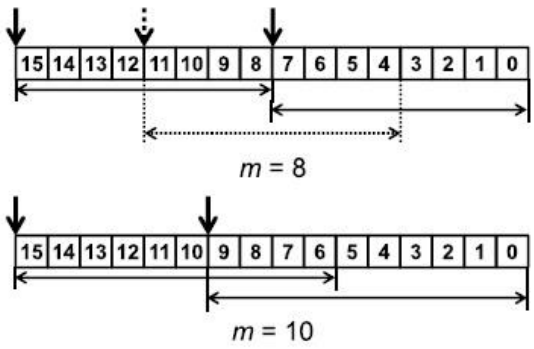

Fig. 2. Possible starting bit positions of 8-b and 10-b segments indicated by arrows; the dotted arrow is the case for supporting three possible starting bit.

In the figure 2, the segments are shown as each block. The segments may be taken as depending on the operands itself. Here the m=8 bits are taken in 3 styles from 0 to 7, 4 to 11 and from 8 to 15. Similarly the m=10 bits are taken in two styles as shown below the first one. 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* . 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. 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*×*m* multiplier; and 3) one 2*n*-bit shifter to expand a 2*m*-bit result to 2*n* bits. 1)–3) incur considerable area and energy penalties completely negating the energy benefit of using the *m* × *m* multiplier.

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*×*m* multiplier. The SSM greatly simplifies the circuit that chooses *m*-

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

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 2$n$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× (n–m) shift when both segments are from the upper ones, as shown in Fig. 3.

×					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.

The above figure 3 shows the 16x16 bit multiplication based on 8x8 bit segments .The unnecessary bits for every operation are made as don't care symbols XXXX. The outputs are placed as MSB and LSB as required depending on the block of which 8 segments which we will take .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. Here for every m bits and n bits the same operation is repeated. The block of B is unnecessary if B is pre-processed as proposed. The main operations are done by the multiplexer placed in the middle and the adder as shown in the figure 4 .It will reduce the time improve the efficiency of the multiplication process.

×					0000	0001	xxxx	xxxx
					0000	0010	xxxx	xxxx

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

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

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 2*m*-bit result to a 2*n*bit results scales linearly with *m*.

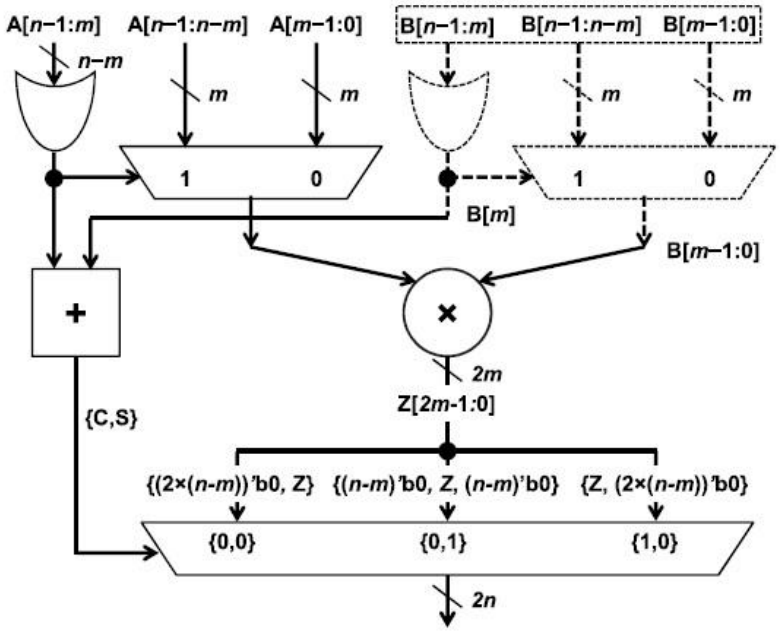

Fig. 5. Proposed approximate multiplier architecture; the logic and wires denoted by the dotted lines are not needed if B is preprocessed as proposed.

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 (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. EXPERIMENTAL RESULTS

Figures shows the simulation results of both existing and proposed systems obtained from the Xilinx ISE 14.1. The Fig .6 shows the output of segmentation process simulation, Fig .7 shows the discrete cosine transformation simulation results of proposed system and Fig .10, shows the multiplier output on FPGA Spartan 3E kit.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

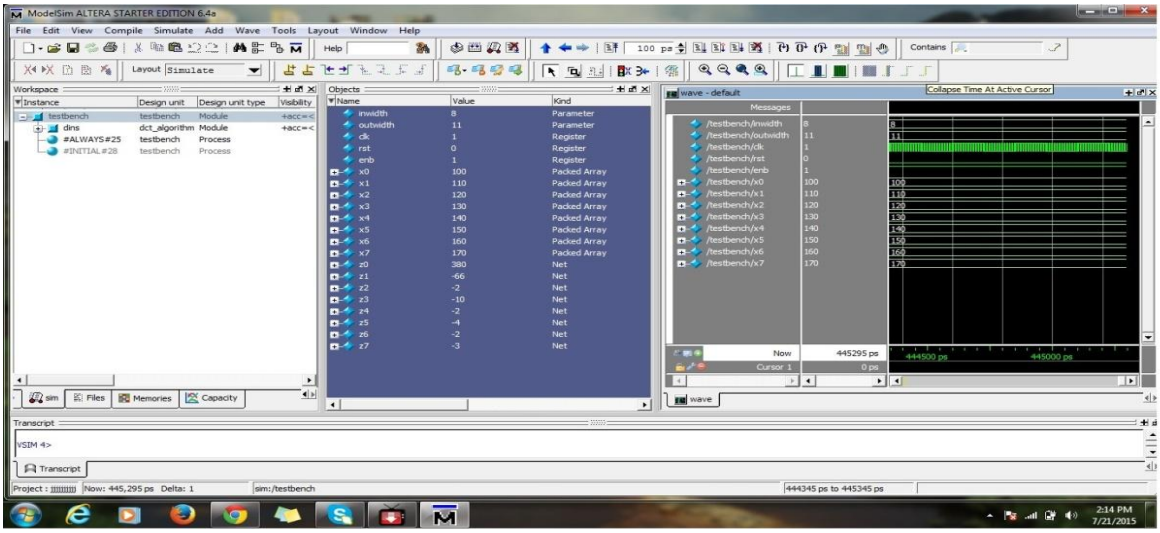

Fig. 6. Segmentation process simulation output.

From the figures 5 and 6 the way of approach is different but the output is identical for both DCT and segmentation process. In both the existing and proposed systems, the outputs are same i.e.,17c,fbe,ffe,ff6,ffe,ffc,ffe,ffd for the given inputs.

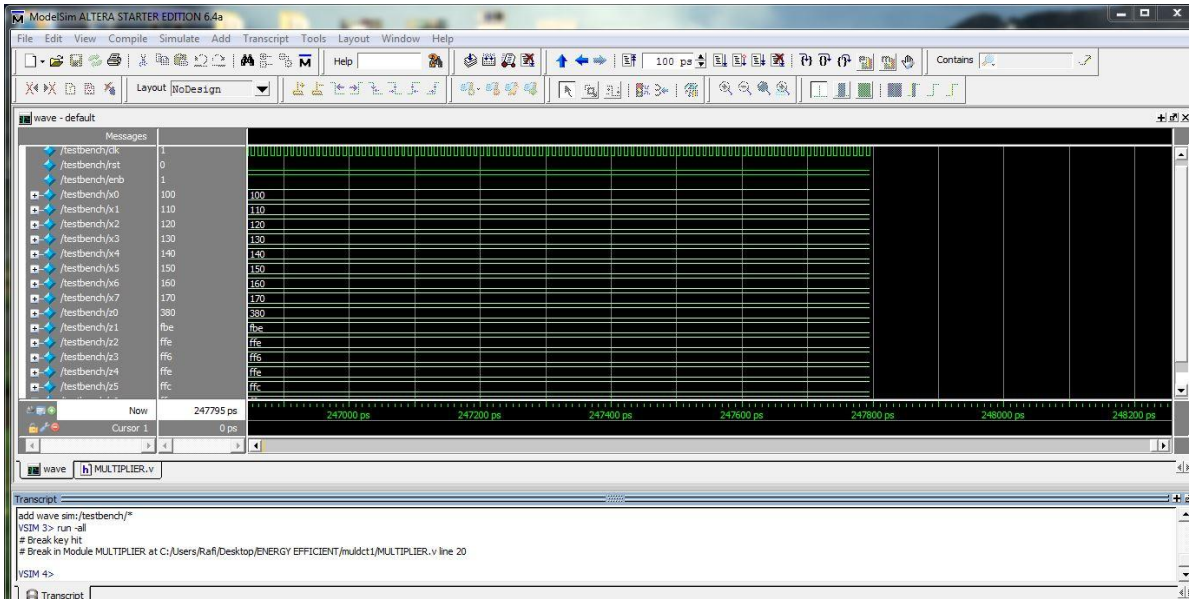

Fig. 7. Proposed Discrete Cosine Transform (DCT) .

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

Fig. 10. Expected output on Spartan 3E FPGA kit.

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*×*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 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.

Z. Babi´c, A. Avramovi´c, and P. Buli´c, "An iterative logarithmic multiplier," *Microprocessors Microsyst.*, vol. 35, no. 1, pp. 23–33, 2011.