Indian Journal of Emerging Electronics in Computer Communications
Vol.4, Issue 2 (2017):Page.757-763
ISSN: 2393-8366

Improved architecture for floating-point four-term dotproduct unit

Treesa Mary Paul[1], and Noble George[2]

[1]*PG Scholar, Department of Electronics and Communication and Engineering, Federal Institute of Science and Technology, Angamaly, Kerala, India. Email: treesamary76@gmail.com*

[2] *Assistant Professor, Department of Electronics and Communication and Engineering, Federal Institute of Science and Technology, Angamaly, Kerala, India. Email: nobleg@fisat.ac.in*

Abstract: *This paper presents an improved architecture for floating-point four-term dot product unit. The proposed design work as a unique single unit for floating-point arithmetic to achieve better performance and accuracy. The fixed point number system is not sufficient to handle some complex computations. In contrast, the floating point operations require complex processing increases the area, power consumption and latency.The dot product unit is widely used in digital signal processing (DSP), multimedia, graphics and statistical applications. The improved architecture have multiplier architectures for four terms separately and it have normalization, rounding and detecting-one logic. The proposed design is implemented for single precision and synthesized in Cadence design suite. In order to evaluate the improvement of the proposed design, the area, latency, and power consumption are investigated. The proposed design reduces the area, power consumption and latency compared to traditional design.*

Keywords—*Digital Signal Processing (DSP), Floating-Point Arithmetic, Dot-Product Unit*

1. Introduction

The dot product is one of the most frequently used operations for a wide variety of graphics, digital signal processing (DSP) and statistical applications. The proposed design computes the four-term dot product in a single unit to achieve better performance and accuracy. The arithmetic units in modern microprocessors execute advanced applications such as 3D graphics, multimedia, signal processing, and a variety of scientific computations that require complex mathematic computations. The fixed-point number system is not sufficient to handle such complex computations. So IEEE-754 Standard for floating-point arithmetic is used here. The floating-point operations require complex processing such as significant alignment, normalization, and rounding, which increases the area, power consumption and latency of the critical path. One approach to reduce the overhead is to execute multiple operations, which are frequently used together in one floating-point unit. In this paper a novel design for a floating-point four-term dot product unit is presented.

The dot product of two vectors X=A.B={ A0, A1,……,An} and Y={ B0, B1,……,Bn} is

$$A \cdot B = \sum_{k=0}^{n} A_k B_k = A_0B_0 + A_1B_1 + \ldots\ldots A_nB_n \quad (1.1)$$

Four-term dot product unit computes the case when n is 4. There are several advanced applications, specifically those with dot products, routinely performed a floating-point multiplication, A x B, immediately followed by a floating-point addition, (A x B) result + C. To increase these applications' performances, design engineers created a new unit that merged a floating-point addition and floating-point multiplication into a single hardware block—the floating-point fused multiplier-adder. The rest of this paper is organized as follows. The architecture of the proposed floating-point four-term dot product is presented in Section II. Section III describes the architecture of the proposed design. The experiment and result is discussed in Section IV. Finally, the conclusion is given in Section V.

2. Proposed Floating-Point Four Term Dot-product Architecture

The floating-point four-term dot product unit takes eight floating-point numbers and computes the sum or difference of the four products as,

$$Z = AB \pm CD \pm EF \pm GH. \tag{2.1}$$

Equation (2) use a pair of fused two-term dot product units in parallel and a single floating-point adder , but has two rounding's after the addition in series . It is called as semi-fused design. This design has a limitation of accuracy due to multiple rounding processes. The fused floating point units reduce area, power consumption and latency compared to a network of traditional floating-point units by executing the common logic for multiple operations in parallel. For that separate multipliers are used for each multiplication to execute operations parallely. The binary floating-point notation, which is specified in IEEE-754 Standard floating-point arithmetic, represents a wide range of numbers from tiny fractional numbers to extremely huge numbers. The floating-point numbers consist of three parts (sign, exponent and significant) so that the operations require complex procedures. For example, the operations frequently require the normalization, which causes an increased logic delay. However, the floating-point operations require complex processing such as significant alignment, normalization, and rounding, which increases the area, power consumption and latency of the critical path. One approach to reduce the overhead is to execute multiple operations, which are frequently used together in one fused floating-point unit.

3. Architecture of Proposed Floating-Point Four Term Dot-Product

The four-term dot product unit can be implemented by concatenating four multipliers and three adders, which is referred to as a discrete four-term dot product unit. Although the floating point multipliers and adders may be well-optimized the discrete design of the four-term dot product requires large area, latency, and power consumption. Moreover, the discrete design computes a multiplication and two additions in series, so the rounding errors are accumulated. Use of a pair of fused two term dot product units in parallel and a floating-point adder eliminates the rounding after the multiplications, but still has two rounding after the additions in series.The four-term dot product unit can be implemented by concatenating four multipliers and three adders, which is referred to as a discrete four-term dot product unit. The architecture of floating-point four-term dot-product unit represent in IEEE754 format. It is a format used to represent floating-point numbers or binary floting-point arithmetic. The standard defines five basic formats, see Table (3.1), the first three formats named single, double and quadruple precision basic formats are used to encode binary floating point numbers and use 32, 64 and 128 bits respectively.

Table no.3.1 Basic Floating-point Formats

Name	Common name	Base	Digits	E_{min}	E_{max}
binary 32	Single precision	2	23+1	-126	+127
binary 64	Double precision	2	52+1	-1022	+1023
binary 128	Quadruple precision	2	112+1	-16382	+16383
decimal 64		10	16	-383	+384
decimal 128		10	34	-6143	+6144

A floating-point number is said to be normalized if the exponent field contains the real biased exponent other than all 0's and all 1's. For all the normalized numbers, the first bit just left to the decimal point is considered to be 1 and not encoded in the floating-point representation and thus also called the implicit or the hidden bit. A floating-point number is considered to be denormalized if the biased exponent field contains all 0's and the fraction field doesn't contain all 0's. The implicit or the hidden bit is always set to 0. Denormalized numbers fill in the gap between zero and the lowest normalized number

The architecture of proposed fused four-term dot product unit consist of four different stages. It include,

1. First Stage: Multipliers And Exponent Logic
2. Second Stage: Alignment And Dual-Reduction
3. Third Stage: Detecting-One Logic And Final Addition
4. Fourth Stage: Normalization and Rounding

There are four different stages for this architecture. First stage is multipliers and exponent logic and sign logic. In this session four different multipliers are used to calculate the multiplied results of four terms in dot-product unit separately. In order to reduce latency, area and power it is necessary to take care of implementing an efficient multiplier architecture. It is proved that Vedic multiplier has an efficient architecture. So the Vedic multiplier is used here to multiply the mantissas of each number taken. Four different Vedic multipliers are used here for each term. The multiplied results are obtained from this. So mantissas are ready. But a floating-point number have sign bit and exponent bits. The sign bit is obtained just by xoring the sign bits of input numbers. Exponent value in this session is obtained by adding the exponents of two inputs. And the result from that is adjusted with bias value of corresponding precision.

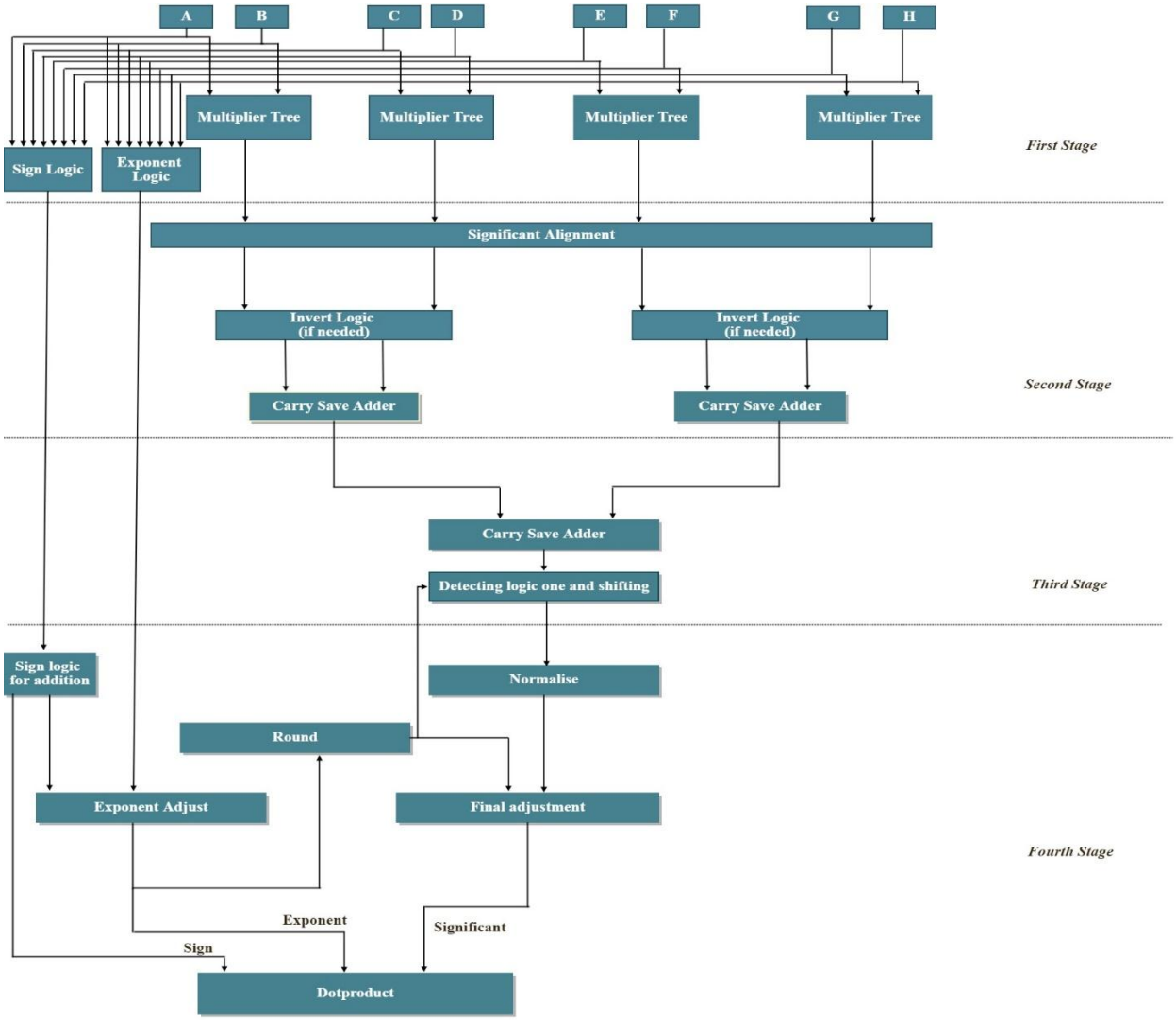

Figure no. 2.1 Architecture of proposed floating-point four term dot-product

Next stage includes addition or subtraction operation in the dot-product unit. The operation select is determined by sign value of the multiplied values. Because the addition or subtraction operation is performed with the multiplied results. If the sign value of multiplier is logic zero, then add two numbers. And if it is a logic one, then subtract two numbers. The subtraction operation is performed by taking the two's compliment form. So an invert block is needed if there is a subtraction operation. But before performing addition or subtraction operation an alignment of multiplier result should be done. For that alignment, the exponents of inputs to addition block should be evaluated. And according to that alignment of mantissas are done. Once it is done addition or subtraction operation is done. The aligned significant pairs are passed to the reduction trees. Since the significant pairs are not sorted based on the comparison, the significant sum must be complemented, if it is negative. Dual-reduction is used to

eliminate the complementation after the significant addition. The four significant pairs are duplicated and they are inverted based on the effective operations. One set of the significant pairs are inverted if the operations are subtraction. In order to handle the four inverted significant pairs, two levels of 4:2 CSA trees are used. The first level CSA trees are included in the second stage and the second level CSA trees are included in the third stage to balance timing between the stages. Third stage include the addition or subtraction of final addition. After the final addition there is a logic one detection block. This is done to adjust the final result to representation of floating-point in IEEE754 format. In this logic. The first logic one in the mantissa sequence from least significant bit (LSB) need to be detected. This is done to exclude the 'hidden-one' from the representation. Fourth stage have sign logic for final result exponent for final result and normalization of final result.Sign logic for final addition include comparison of sign inputs and obtained mantissas and their corresponding operation. Exponent logic for final result is calculated from the round-off values obtained from multiplier. And a normalisation is needed for result. Thus the exponent,sign and mantissa bits are completely obtained.

4. Experiment and Results

Previous sections have introduced a novel floating-point four-term dot product unit design. The proposed design has been implemented for single precision in Verilog HDL and synthesized with Xilinx Tool. To estimate the area, latency and power consumption of the designs, Cadence Design Suite is used. The estimated results have been compared to evaluate the improvement of the proposed design over the traditional designs [6]. Table 4.1 compares the area, latency, power of the proposed and traditional designs for single precision.

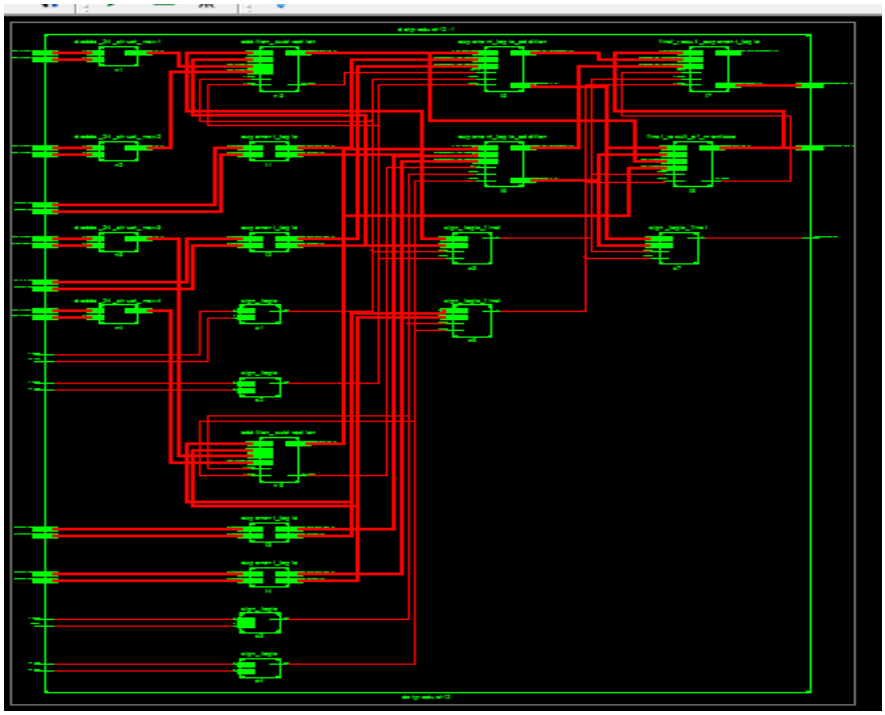

Figure no.4.1 Schematic of proposed architecture

Table no.4.1 Compare values between proposed and traditional designs

PARAMETERS	TRADITIONAL DESIGN (referred to [6])	PROPOSED DESIGN (referred to [1])
AREA (um^2)	94,400	355.677
TIMING (ns)	2.96	0.02063
POWER (mW)	49.67	15.544

As an experimental discussion five-term dot product unit is obtained. And synthesized using Xilinx tool.

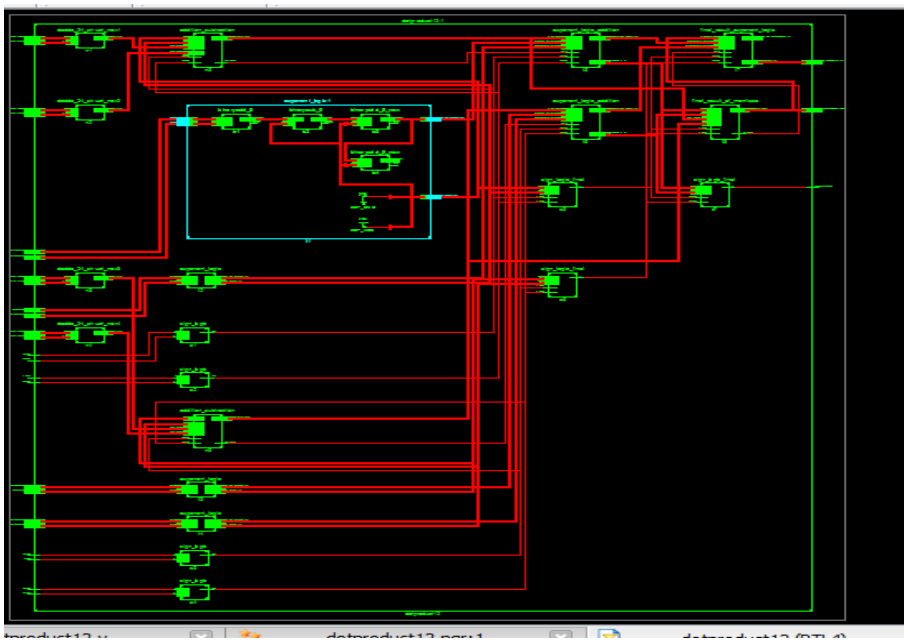

Figure no.4.2 Schematic of five-term dot product unit.

6. Conclusion

A novel design for a fused floating-point four-term dot product unit has been presented. Since the dot product is a key operation for the 3D graphics and DSP applications, the proposed floating-point four-term dot product unit will improve the performance of such applications. The proposed fused floating-point four-term dot product unit applies a new exponent compare and significant alignment, dual-reduction, early normalization and compound addition and rounding. As a result, the area, latency, and power consumption is reduced.

REFERENCE

[1] Jongwook Sohn, Member, IEEE, and Earl E. Swartzlander, Jr., Life Fellow, IEEE ”A Fused Floating-Point Four-Term Dot Product Unit”, IEEE Transactions On Circuits And Systems—i: Regular Papers, pp. 1549-8328,2016 IEEE.
[2] Y. Tao, G. Deyaun, F. Xiaoya, and J. Nurmi, “Correctly rounded architectures for floating-point multi-operand addition and dot-product computation,” in Proc. IEEE 24th Int. Conf. Appl.-Specific Syst., Archit., Processors, 2013, pp. 346–355.
[3] J. Sohn and E. E. Swartzlander, Jr., “A fused floating-point three-term adder,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, pp. 2842–2850,2014.
[4] J. Sohn and E. E. Swartzlander, Jr., “Improved architectures for a floating point fused dot product unit,” in Proc. 21th IEEE Symp. Comput. Arith., 2013, pp. 41–48.
[5] J. Sohn and E. E. Swartzlander, Jr., “Improved architectures for a fused floating-point add-subtract unit,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, pp. 2285–2291, 2012.
[6] Y. Tao, G. Deyaun, F. Xiaoya, and J. Nurmi, “Correctly rounded architectures for floating-point multi-operand addition and dot-product computation,” in Proc. IEEE 24th Int. Conf. Appl.-Specific Syst., Archit., Processors, 2013, pp. 346–355.
[7]D. Kim and L. Kim, “A floating-point unit for 4D vector inner product with reduced latency,” IEEE Trans. Comput., vol. 58, pp. 890–901, 2009.