FPGA Implementations of Low Precision Floating Point Multiply-Accumulate

Alexandru Amaricai, Oana Boncalo, and Ovidiu Sicoe

***Abstract*—Floating point (FP) multiply-accumulate (MAC) represents one of the most important operations in a wide range of applications, such as DSP, multimedia or graphic processing. This paper presents a FP MAC half precision (16-bit) FPGA implementation. The main contribution of this work is represented by the utilization of modern FPGA DSP block for performing both mantissa multiplication and mantissa accumulation. In order to use the DSP block for these operations, the alignment right shifts are performed before the multiply-add stage: a right shift on one of the multiplicand, and, a left shift for the other. This results in efficient DSP usage; thus both cost savings and higher performance (high working frequencies and low latencies) are targeted for MAC operations.**

***Index Terms*—Digital Arithmetic, Floating Point Arithmetic, FPGA, Multiply-Accumulate; Dot-Product**

I. Introduction

MULTIPLY-ACCUMULATE or dot-product represents the most important arithmetic operation in application which require convolution or matrix-multiplications, such as: DSP, image processing, graphic processing, scientific computing, etc [2][4][8][10][15]. A dedicated architecture for this critical operation has two major advantages: increased accuracy (due to a single rounding operation with respect for multiple roundings performed for each multiplication and addition) and increased performance.

This paper proposes a new MAC FPGA implementation for half precision format (16 bit) of IEEE 754 FP numbers. The main module of the proposed MAC is represented by the DSP block, which is used for performing both the mantissa multiplication and accumulation sub-operations. This represents a novel approach in optimizing MAC operation in FPGA devices; related architectures use the DSP blocks in order to perform only the multiplication in the mantissa datapath; the accumulation is computed using conventional slices. The proposed approach has been implemented on Xilinx FPGA devices which use DSP48E or DSP48E1 block; these modules contain a 25x18 bit two's complement multiplier and a 48-bit adder/accumulator [16]. In order to use the DSP48E or DSP48E1 block for both multiplication and accumulation, alignment shifting needs to be performed before the multiplication.

This work was supported by the Romanian Research Council PNII-RU-TE-3-2011-0186 FLAG research grant.

A. Amaricai, O. Boncalo and O. Sicoe are with Department of Computer Engineering, University Politehnica of Timisoara, Timisoara, Romania (email: alexandru.amaricai@cs.upt.ro)

This paper is organized as follows: Section II presents related multiply-add fused and multiply-accumulate architectures, modern DSP blocks' structure are depicted in Section III, while Section IV is dedicated to the proposed MAC implementation.

II. Floating Point Multiply-Accumulate

A. Algorithm

The FP multiply-accumulate represents the consecutive addition between multiple products. It is very similar to the FP multiply-add fused. It comprises of the following steps [9][15][17]:

1. **Exponent processing** – the exponent of the product is computed by adding the two exponents and subtracting the bias; the difference between the product exponents' and the accumulator's exponent is computed in order to determine the alignment shifting;
2. **Mantissa multiplication** – the product between mantissa's of the new pair of operands is computed;
3. **Alignment shifting** – shifting is performed to the product for alignment; the shift amount has been determined in the exponent processing phase;
4. **Mantissa accumulation** – the shifted product is accumulated to the previous sum;
5. **Leading zero detection** – this step is necessary in the normalization process;
6. **Normalization** – this step is performed by a mantissa left-shift; the amount of left-shifting is determined in the leading zero detection step; an exponent update has to be performed in parallel;
7. **Rounding** – rounding is performed according to the required rounding algorithm, by a possible addition of 1 ulp to the mantissa;

The major difference between MAC and multiply-add fused is represented by the way in which steps 1-4 are performed. For multiply-add fused, they are performed only once, because only one multiplication and one addition are involved. In MAC, multiple multiplications and accumulations are required. Therefore, steps 1-4 are performed multiple times (depending on the number of the products). Furthermore, compared to multiply-add fused, the MAC does not have inputs for the addend. Another major feature of the MAC operation is the fact that steps 5-7 are performed only once (when all the accumulations between the products are executed) [4].

B. FPGA implementations

Although MAC operations are frequent in a wide range of applications (such as the ones which rely on convolution or matrix multiplications), few such architectures have been proposed. For example, in a wide range of matrix multipliers [2][6][8][10], the FP MAC operation is performed using a separate FP multiplier and FP adder. Several enhancements can be obtained if dedicated FP accumulators are used to add the result of FP multiplier [14]. However, the drawback of this approach is represented by the rounding operation during the multiplication (which decreases the accuracy).

Regarding dedicated MAC architectures, such have been proposed in [4][11][13]. The approach in [13] is straight-forward implementation of an ASIC implementation for an IEEE single precision MAC. The multiplication is performed using conventional slices, similar to the tree multipliers used in ASIC. The alignment shifting of the product is performed after the multiplication step.

A highly versatile multiply-accumulate architecture is the one implemented in FloPoCo [3][4]. FloPoco dot-product operators [4] have a wide range of configuration parameters: FP precision, accumulator precision, frequency target, etc. The product alignment shifting is performed after the multiplication. The multiplication can be implemented using DSP blocks only, slices only, or a combination of both. An important feature of the FloPoCo dot-product is the fact that the MAC part is separated from the FP conversion part (which performs leading zero detection, normalization and rounding). This is motivated by the low number of conversions with respect to accumulation in applications which require large convolutions or matrix multiplications. The FloPoCo tool generates a generic and configurable RTL code. The generated MAC architectures do not use highly dedicated features of the FPGA building blocks (DSP or slices). Thus, the FloPoCo approach targets mostly portability and configurability, rather than performance and cost savings obtained through device based optimizations.

Configurable dot-product architectures which can perform both FP and fixed point operations are presented in [11]. The main feature of these units is represented by the parallel computation of the products. Therefore, these architectures do not require feedback accumulation of products. The main disadvantage is represented by the high cost, due to the large number of multipliers required.

TABLE I
FPGAs FP MAC ARCHITECTURES

Architecture	IEEE Precision	Product Alignment	Multiplier Type	Device
FloPoCo DotProduct+ LongAcc2FP[4]	Generic	After Mult.	Generic	Generic
Single Precision Dot-Product [13]	Single Precision	After Mult.	Slice based	Generic
Hybrid FP/Int Dot-Product [11]	Single Precision Input/Output Variable Internal Accumulation	After Mult.	DSP based	Generic
Proposed	Half Precision	Before Mult.	DSP based	Virtex-4,5,6 7 Series

Table I presents the related approaches for the MAC and dot-product approaches. Our approach is similar to the ones in [4][13]: it relies on only one multiplier, and thus the multiplications in the dot-product are performed in a serial manner. Compared with the [4][13], the main contributions of the proposed approach are:

1. It uses the DSP block for both multiplication and accumulation.
2. The alignment shift operations are performed on the multiplicands and not the product

The present work is an extension of the FP multiply add fused presented in [1]. However, the DSP block and other components of the architecture (exponent processing, alignment shifting) have been tailored in order to accommodate multiple accumulation of products and not the addition of an addend with a product.

III. DSP Block Architecture

FPGA's DSP blocks have been incorporated in order to increase the performance and to lower power consumption in a wide range of applications, such as DSP, multimedia, scientific computing, etc. Early days DSP blocks have included only a two's complement integer multiplier. Modern FPGA devices' DSP blocks have increased functionality, which in many aspects is similar to the datapath processing in a conventional processing core. Another feature of these units is represented by the lack of standardization: different architecture and functionality are found in different FPGA families [3]. The DSP block architecture used in this work is Xilinx DSP48E or DSP48E1, which is part of the Virtex-4, Virtex-5, Virtex-6 and 7 Series FPGA devices [16]. The structure of the DSP48E1 is depicted in Fig. 1. Its main arithmetic components are the 25x18 bits two's complement multiplier and the 48-bit ALU. The ALU has several functionalities, such as carry propagate adder, subtracter or logic functions. Furthermore, selection of the ALU's inputs is possible. One input can be selected between the product and one of the multiplicands (in this case the multiplier is bypassed). The possible cases for the second

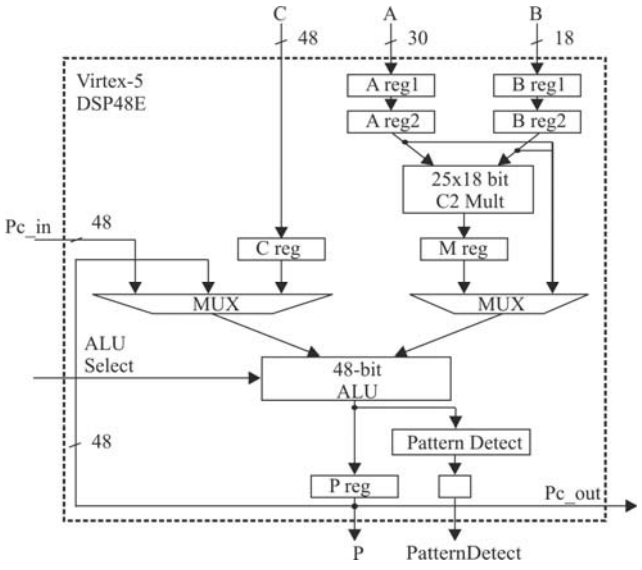

Fig. 1. Xilinx Virtex-5 DSP48E structure [16]

ALU input are: *C* DSP block input, the previous result of the ALU, or the result of a neighboring ALU. The first option is useful for multiply-add fused operations; selecting the previous results of the adder is required in MAC; the result from the adjacent DSP block is needed when performing large precision multiplication.

IV. Proposed Half Precision MAC

A. Proposed architecture

The proposed half precision (1 bit of sign, 5 bits of exponent and 10+1 bits of mantissa) MAC unit is composed of the following modules:

1. **Exponent processing** – it computes the exponent of the product and the alignment shifting; the former is calculated subtracting the first product's exponent from the current exponent.
2. **Alignment shifting** – in the proposed MAC unit the alignment shift is performed before the computation of the product; right-shift alignment is performed to the multiplicand fed to the 18-bits DSP input, while the multiplicand fed to the 25-bits DSP input is left shifted for alignment; the two shifts are mutually exclusive; the alignment shifts are computed with respect to the first product.
3. **Mantissa multiply-accumulation** – the multiplication and the accumulation of mantissas are done as a single operation using the DSP block.
4. **Leading zero detection** – a tree based leading zero detector is used for counting leading zeros [12];
5. **Normalization** – a left shifter is used for mantissa normalization; an exponent update is performed; the amount of left-shifting is determined by the number of leading zeros.
6. **Rounding** – a carry-propagate adder is used in order to implement the rounding operation; an overflow may appear after rounding, which requires a one position mantissa right-shift.

The proposed unit is depicted in Fig. 2. Implementing the mantissas multiplication and accumulation as a single operation using one DSP block introduces two limitations, due to the limited size of the multiplicands (24-bits and 17-bits unsigned) in the DSP's multiplier. For half precision formats, the maximum exponent's difference is 31, which may result in a 31-position alignment shifting.

Regarding the right-shift alignment, the maximum number of positions which guarantee no loss in precision is 6 (the difference between the size of the DSP block multiplicand (17) and the size of the mantissa (11)). For more than 6-bit right-shift, a precision loss will appear. Regarding the left-shift alignment, the maximum number of positions allowed by the proposed design is 13 (the difference between the size of the DSP block multiplicand (24) and the size of the mantissa (11)). For more than 13-bit left-shift, most significant bits of the result will be lost. Therefore, the proposed MAC can handle accumulation of numbers which do not have very high exponent differences. However, in most applications requiring MAC (such as DSP or graphic processing), these cases are rare.

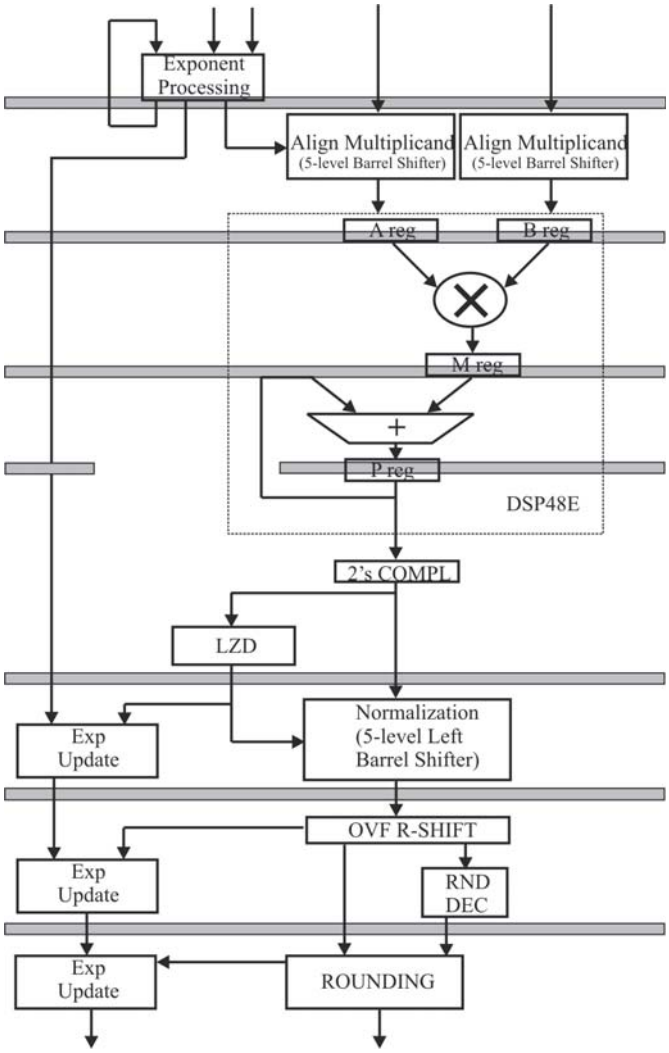

Fig. 2. Proposed 8-pipeline MAC unit (grey box indicate the pipeline registers)

Eight pipeline stages have been implemented (see Fig. 2). In order to increase both the working frequency and the cost efficiency, the following built-in pipeline registers within the DSP blocks are used: the registers latching the multiplicands, the register corresponding to the product and the register latching the accumulation result. The pipeline stages in the proposed architecture are:

- Exponents processing stage;
- Alignment shifting stage;
- DSP stages (1 stage for multiplication and 1 for accumulation)
- Leading zero detection stage
- Normalization stage
- Rounding stages (2 stages)

The first four stages are part of the accumulation pipeline, and are executed for each multiplication in the MAC. The last four stages are computed once, at the end of operation.

B. Results

Table II depicts the synthesis results for Xilinx Virtex-5 (which incorporates DSP48E block) and Xilinx Virtex-7 (which incorporates DSP48E1 block). With respect to the

FloPoCo approach, the proposed MAC presents a similar cost, and a lower working frequency. However, the performance of the proposed architecture is higher, as the presented MAC has 8 pipeline stages at around 300 MHz, while the FloPoCo has 12 pipeline stages at around 360 MHz. Regarding the dot-product approach in [11], the results are given for single precision (32-bit format). Although straightforward comparison is impossible to perform, the architecture presented in [11] has some disadvantages compared to our approach. The [11] dot-product relies on parallel multiplication, which requires multiple DSP blocks. Therefore, even a half-precision implementation would require multiple parallel DSP blocks. Thus, the dot-product unit would have an increased cost compared to our architecture (especially in terms of DSP block count). Furthermore, it lacks flexibility, because it cannot perform MAC operations with more multiplications than the parallel multiplier modules incorporated.

TABLE II
SYNTHESIS RESULTS & COMPARISSONS

Architecture	Performance	Cost	Device
Proposed	8 stages 298 MHz	1 DSP 455 LUT-FF pairs	Xilinx Virtex-5 -3 speed grade
Proposed	8 stages 378 MHz	1 DSP 408 LUT-FF pairs	Xilinx Virtex-7 -2 speed grade
FloPoCo [4] 16-bit IEEE 34 bit accumulate 400 MHz target	12 stages 369 MHz	1 DSP 434 LUT-FF pairs	Xilinx Virtex-5 -3 speed grade
Hybrid FP/Int Dot Product [11] 32 bit FP	275 MHz for 65 bit acc 400 MHz for 10 bit acc	n DSPs (n – number of multiplications) 1000 LUT- FF pairs	Xilinx Virtex-6

V. CONCLUSIONS

This paper proposes FPGA based architectures for half precision FP MAC which exploits modern FPGA features for increased performance and cost efficiency. The main contributions of this paper are:

1. Merging the multiplication and the accumulation into a single step – we have exploited the built-in multiplier and accumulator within the modern DSP blocks for in order to achieve this goal
2. Performing the alignment shifting on the multiplicands – in order to implement the first contribution, alignment shifting has to be performed before the multiplication; thus, compared to other approaches, we perform shifting on multiplicands, rather than the product.

We have performed device based optimization for the MAC operation, thus resulting in a performance increase of up to 20% compared with at a similar cost. This represents a similar approach, which has been used for FP addition and multiplication [5][7].

The proposed unit has two disadvantages:

1. Low flexibility – our architecture has been designed in order to take advantage of the Xilinx DSP48E and DSP48E1 blocks; "porting" to other devices needs rework
2. Precision loss for great alignment shifts – this limitation is due to the limited size of DSP block's multiplicands size.

Therefore, regarding FPGA FP design, the main tradeoff is represented by performance and cost improvements using device based optimizations versus flexibility/portability of FP units (i.e. reduced design time).

REFERENCES

[1] A. Amaricai, O. Boncalo, C.E. Gavriliu, "Low Precision DSP Based Floating Point Multiply-Add Fused for FPGAs", submitted to IET Computing & Digital Techniques, 2013

[2] F. Bensaali, A. Amira, R. Sotudeh, "Floating-point matrix product on FPGA", Proc. IEEE/ACS Int. Conf. on Computer Systems and Applications, pp. 466-473, 2007

[3] F. de Dinechin, B. Pasca, "Designing custom arithmetic data paths with FloPoCo" IEEE Design and Test of Computers, Vol. 28, Issue 4, pp. 18-27, 2011

[4] F. de Dinechin, B. Pasca, O. Cret, R. Tudoran, "An FPGA-specific approach to floating-point accumulation and sum-of-products" Proc. 2008 Int. Conf. on Field Programmable Technology (FPT), pp. 33-40, 2008

[5] K.S. Hemmert, K.D. Underwood, "Fast, Efficient Floating-Point Adders and Multipliers for FPGAs", ACM. Tran.s on Reconfigurable Technology and Systems (TRETS), Vol. 3, Issue 3, Art. No. 11, 2010

[6] B. Holanda, R. Pimentel, J. Barbosa, R. Camarotti, A. Silva-Filho, L. João, V. Souza, J. Ferraz, M. Lima, "An FPGA-Based Accelerator to Speed-Up Matrix Multiplication of Floating Point Operations", Proc. 2011 IEEE Int. Symp. on Parallel and Distributed Processing Workshops and PhD Forum, pp. 306-309, 2011

[7] M. K. Jaiswal, R.C.C. Cheung, "Area-efficient architectures for double precision multiplier on FPGA, with run-time-reconfigurable dual single precision support" Microelectronics Journal, Vol. 44, Issue 5, pp. 421-430, 2013

[8] Z. Jovanovic, V. Milutinovic, "FPGA accelerator for floating-point matrix multiplication" IET Computer and Digital Techniques, Vol. 6, Issue 4, 249-256, 2012

[9] T. Lang, J.D. Bruguera, "Floating-Point Fused Multiply-Add with Reduced Latency" Proc. 2002 IEEE Int. Conf. on Computer Design (ICCD) ,pp.145-150, 2002

[10] M. deLorimier, A. DeHon, "Floating-Point Sparse Matrix-Vector Multiply for FPGAs" Proc. ACM/SIGDA 13th Int. Symp. On Field Programmable Gate Arrays (FPGA), pp. 75-85, 2005

[11] A.R. Lopes, G. Constantinides, "A fused hybrid floating-point and fixed-point dot-product for FPGAs" Proc. 6th Int. Conf. on Reconfigurable Computing: Architectures, Tools and Applications (ARC'10), pp. 157-168, 2010

[12] V.G. Oklobdzija, "An algorithmic and novel design of a leading zero detector circuit: Comparison with logic synthesis", IEEE Trans. On VLSI Systems, Vol.2, Issue 1, pp. 124-128, 1994

[13] A. Paidimari, A. Cevrero, P. Brisk, P. Ienne, "FPGA Implementation of a Single-Precision Floating-Point Multiply-Accumulator with Single-Cycle Accumulation" Proc. 17th IEEE Symp. On Field Programmable Custom Computing Machines (FCCM), pp. 267-270, 2009

[14] S. Sun, J. Zambreno, "A Floating-point Accumulator for FPGA-based High Performance Computing Applications", Proc. 2009 Int. Conf. on Field Programmable Technology (FPT), pp. 493-499, 2009

[15] Y. Tao, G. Deyuan, D. Xiaoya, J. Nurmi, "Correctly Rounded Architectures for Floating-Point Multi-Operand Addition and Dot-Product Computation", Proc. 24th IEEE Conf. on Application-Specific Systems, Architectures and Processors (ASAP), pp. 346-355, 2013

[16] Xilinx, "Virtex-5 FPGA XtremeDSP Design Considerations" – User Guide, 2012

[17] A.M. Zaki, M.H. El-Shafey, A.M.B. Eldin, G.M. Ali, "A New Architecture for Accurate Dot Product of Floating Point Numbers" Proc. 2010 Int. Conf. on Computer Engineering and Systems, pp. 139-145, 2010

Alexandru Amaricai has received his Dipl. Eng. from University Politehnica of Timisoara in 2006 and his PhD Degree in Computer Engineering from the same university in 2009. Since 2011, he is Assistant Professor in the Mobile and Embedded Systems Research Group within the Computer Engineering Department from University Politehnica of Timisoara. His main research areas include floating point arithmetic, FPGA design and digital systems reliability.

Oana Boncalo has received his Dipl. Eng. from University Politehnica of Timisoara in 2006 and his PhD Degree in Computer Engineering from the same university in 2009. Since 2011, she is Assistant Professor in the Mobile and Embedded Systems Research Group within the Computer Engineering Department from University Politehnica of Timisoara. Her main research interests include reliability evaluation and fault modeling, FPGA design and FPGA implementation of forward error correction schemes.

Ovidiu Sicoe has received his BSc. in Computer Engineering from University Politehnica of Timisoara in 2011 and his MSc. degree in 2013. He is currently pursuing the PhD. Degree in Computer Engineering from the same university. His main research interests include floating point arithmetic and graphic accelerators design.