Multi Operation Floating Point Architecture using DADDA Multiplier

Girija Alukuru[1], Janardhana Raju M[2], Anilkumar Somasi[3]

[1]*M.TechStudent* , [2]*Assoc.Professor*

[1,2]*E.C.E,JNTUniversity Ananthapur,*

Department of ECE, Siddharth Institute of Engineering & Technology, Puttur-517583, Andhra Pradesh, India

[3]*M.Tech Student, CNST, JNTUniversity Hyderabad,*

Centre for Nano Science & Technology, IST, JNTU Hyderabad, Kukatpally,Hyd-500085, Telangana, India

***Abstract*— Floating-point unit (FPU) is one of the most important custom applications needed in most hardware designs as it adds accuracy and ease of use. Its applications range from multimedia and 3D graphics processing to scientific and engineering applications. In this thesis we designed a ASIC implementation of a novel single-precision floating point processing element (FPPE) using a 24-b variant is presented for multi operations based on selection such as addition, subtraction, multiplication and accumulation operations. This FPPE can be designed by using 24X24 Dadda multiplier. We also present a circuit-level implementation of the Dadda multiplier to explore the various Performance-speed tradeoffs involved.**

The proposed floating point architecture is used in the application development of DSP such as Finite impulse response (FIR) filters, graphics processing, Discrete cosine transforms (DCT), fast Fourier transform (FFTs) division and argument reduction.

Keywords— **Design of Dadda multiplier, floating point 32-b design, floating point reconfiguration and its rounding**

I. INTRODUCTION

Digital signal processing and multimedia applications require large amounts of data, real-time processing ability and very high speed. To represent very large or small values, large range is required as the integer representation is no longer appropriate. These applications and values can be represented using the IEEE-754 standard based floating point representation.

Multipliers are among the fundamental components of many digital systems and, hence, their power dissipation and speed are of primary concern. Multiplication plays an essential role in computer arithmetic operations for both general purpose and digital signal processors. For computational extensive algorithms required by multimedia functions such as finite impulse response (FIR) filters, infinite impulse response (IIR) filters and fast Fourier transform (FFT).In recent trends the column compression multipliers are popular for high speed computations due to their higher speeds.

In Wallace Multipliers partial product of N rows by grouping into sets of three row set and two row set using (3,2) counters and (2,2) counters respectively. But in case of Dadda with the exact placement of the (3,2) counters and (2,2) counters in the maximum critical path delay of the multiplier. The hardware required for Dadda multiplier is lesser than the Wallace multiplier, Wallace multiplier and Dadda multiplier exhibits similar delay but Dadda multiplier is faster than the Wallace multiplier.

II. ACHIEVEMENTS OF THIS WORK

The Architecture of this work was achieved the following things.

- Extendable arithmetic algorithm for future-generation architectures.
- Low area and Data path Elements
- Full processing of cores, with minor architectural modifications.
- Allows Complex Arithmetic Computations.
- High Speed Elements for Media Reconfigurable Processing.

III.DATAPATH ARCHITECTURES

A.DADDA MULTIPLIER:

The Dadda multiplier is a hardware multiplier design invented by computer scientist Luigi Dadda in 1965. It is similar to the Wallace multiplier, but it is slightly faster and requires fewer gates.

Dadda multiplier essentially minimizes the number of adder stages required to perform the summation of partial products. Dadda multiplier consists of three stages. In the first stage, the partial product matrix is formed. In the second stage, this partial product matrix is reduced to a height of two. In the final stage, these two rows are combined using carry propagating adder.

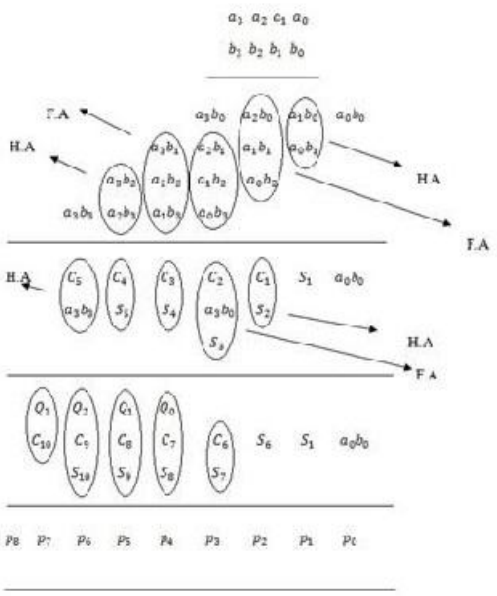

Fig1: Dadda multiplier architecture

Wallace multiplier is similar to Dadda multiplier, Dadda multiplier is slightly faster than Wallace multiplier.

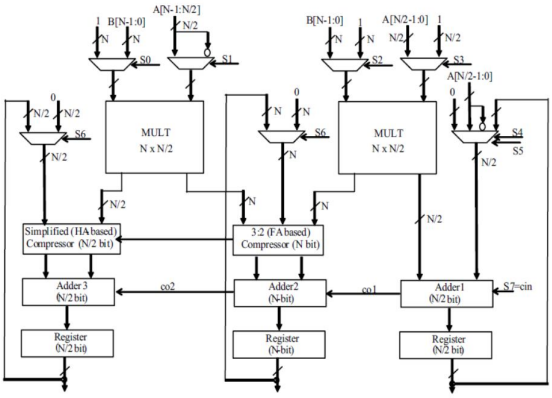

Fig2: Wallace N-bit multiplier architecture

The second proposed data path structure is shown in Fig. 4.It can be observed from the figure, that the data path also relies on a divide and conquers approach for multiplication, by following the same operand splitting technique described earlier. However, an advantage over the previously proposed Design is that this architecture eliminates the intermediate Compressor stage by transmitting the partial products directly to the 2*N*-bit carry-linked adders. Multiplexers placed after the Multipliers impart additional flexibility and increase the range of operations performed by the data path. These multiplexers are controlled by one-hot select signals ADD, MUL, and ACC, and send the appropriate signals to the inputs of the adders. For a multiplication operation, the multiplexers send the outputs of the two multipliers to the adders. For an addition/subtraction operation, the two operands are selected to be sent to the adders, while for an accumulation operation, the multiplexers send the accumulated result along with a string of zeroes to the adders.

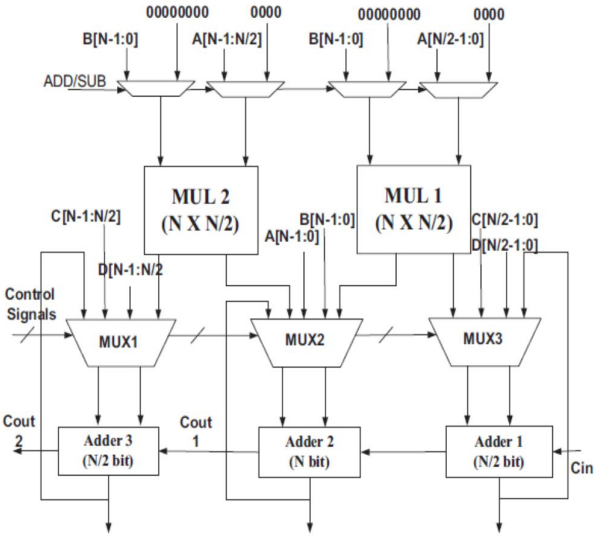

Fig3: Dadda N-bit multiplier architecture

IV. Implementation of Floating Point Architecture and It's Rounding Configuration

In this section, we present the organization of the proposedFPPE based on the generalized data path architectures. The proposed FPPE accepts 32-b single-precision floating point operands *A* and *B* at the input stage. The operands go through a data conditioning stage which involves aligning the two sign bits S_A and S_B mantissas M_A and M_B and adjusting the exponents E_A and EB. These adjusted operands then go through the arithmetic unit which performs the addition, subtraction, and multiplication operations. The result is then normalized and rounded before the output stage.

The operand A and B are compared for exponent values.Thecomparision operation involves an 8-b subtraction depending on $E_A$and E_B, produces difference and borrow bits. This borrows (SS) bit is used to control the multiplexers. If $EA>E_B$, shift select is zero then the normal mantissa bits goes to the pipeline-register, If $E_A<E_B$, shift select is one then the shifted mantissa bits goes to the pipeline-register through Barrel shifter. A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in one clock cycle. After alignment, the mantissas of the two numbers are sent to a 24-b integer PE. This PE is a 24-b extension of the two data path structures proposed in Section II. A bulk of the area of this data path is occupied by the two 24 × 12multipliers. Pipelining stages are often required in large dataPath or multiplier structures, to ensure a high throughput and High speed of operation. The exponent result is controlled by using control signal. When control signal is set to one adder result is directly sent to the pipeline register as exponent result

otherwise E_A or E_B sent to the pipeline register based on shift select value.

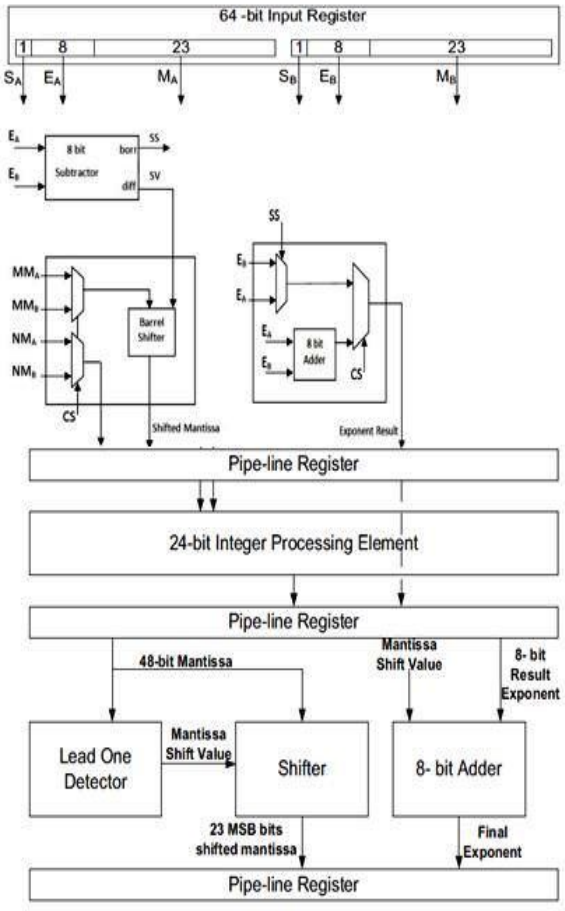

Fig2: 32-bFloating point multiplier architecture

The mantissa and exponent of the result obtained from the integer PE now need to be normalized and rounded so as to be represented back in the IEEE 754 floating point format. For this purpose, a copy of the mantissa of the result is fed to a modified leading one detector (LOD). This LOD works as a priority encoder. This same value is used to adjust the exponent accordingly. Once, the mantissa and exponent have been adjusted to the IEEE 754 single-precision format, the 24 LSBs of the rotated mantissa are dropped. That is the 48-b mantissa is truncated. This approach compromises on the speed of the result.

A.ROUNDING:

First we take care of the sign and the exponent. The sign is stored as is (usually in 1 bit). The exponent is stored as is, if it is within the given range, otherwise we have underflow if the exponent is too small, or overflow if it is too big, these and other exceptions are dealt with differently on different machines, sometimes underflow is set to 0.For the mantissa we apply rounding. Rounding produces the computer number closest to the real number. Notation: if x is a real number, fl(x) is the computer representation of that number. Assume that the computer can store k digits (in base _) for the mantissa. Thus if x = ±0.d1d2d3d4 . . . × _n

Then with rounding fl(x) = ±0.e1e2e3e4 . . . ek × _n

Where ek = dk if dk+1 < _/2 or ek = dk + 1 if dk+1 _ _/2, and the rest of the digits e1, . . . , ek−1 are the di appropriately adjusted, i.e. if dk = _ −1 and dk+1 > _/2, then ek = 0 and ek−1 = dk−1 +1, etc.

In some cases the exponent could also change (and cause overflow).

V.CONCLUSION

In this paper, we presented our recent efforts in the design of high-speed and low-area, data path elements for reconfigurable media processing architectures. It was observed that Dadda multiplier was around 14% faster and consumed 27%–45% lower power, hence it was selected to build the FPPE.The data paths are scalable and parameterizable. This was demonstrated through the implementation of a new FPPE.The generalized structure of the data paths makes them ideal implementation platforms for soft-processing-based systems.Also the power-delay product of the proposed design is significantly lower than that of the regular Wallace multiplier.Our future efforts in this area will involve integrating these data path structures into a hybrid, multigranular FPGA as well as soft-processing reconfigurable array for low-cost, high-speed multimedia processing.

VI.REFERENCES

[1] DAPDNA-2 Product Brochure. (2010) [Online]. Available:http://www.ipflex.com

[2]L.DADDA,"some schemes for parallel multipliers", Alta Frequenza,vol.34, pp.349-356,1965.

[3] S. Xydis, G. Economics, and K. Pekmestzi, "Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths," Integration, VLSI J., vol. 42, pp. 486–503, Mar. 2009.6.

[3] S. Chalamalasetti, W. Vanderbauwhede, S. Purohit, and M. Margala, "A low cost reconfigurable soft processor for multimedia applications:Design synthesis and programming model," in Proc. Int. Conf. Field Program. Logic Devices, 2009, pp. 534–538.

[4] C. Baugh and B. Wooley, "A 2s complement parallel array multiplication algorithm," IEEE Trans. Comput., vol. 22, no. 2, pp. 1045–1047, Dec.1973.

[5] H. Oh, S. Mueller, C. Jacobi, K. Tran, S. Cottier, B. Micheal, H.Nishikawa, Y. Totsuka, T. Namatame, N. Yano, T. Machida, and S.Dhong, "A fully pipelined single-precision floating point unit in thesynergistic processor element of a CELL processor," IEEE J. Solid-StateCircuits, vol. 41, no. 4, pp. 759–771, Apr. 2006.

[6] N. Hockert and K. Compton, "FPPU: Fractured floating point unit for FPGA soft processors," in Proc. Int. Conf. Field-Program.Technol., Dec.2009, pp. 143–150.

[7] S. Liang, R. Tessier, and O. Mencer, "Floating point unit generation and evaluation for FPGAs," in Proc. 11th Annu. IEEE Symp. Field-Program. Custom Comput. Mach., Apr. 2003, pp. 185–194.