Design of low power floating point multiplier with reduced switching activity in deep submicron technology

Sivanantham S
Vellore Institute of Technology
95 PUBLICATIONS **404** CITATIONS

International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 8, Number 6 (2013) pp. 715-722
© Research India Publications
http://www.ripublication.com/ijaer.htm

Design of Floating Point Multiplier for Signal Processing Applications

A. Rakesh Babu, R. Saikiran and Sivanantham S.

School of Electronics Engineering, VIT University
Vellore – 632014, Tamilnadu, India.
{rakesh40622348,saikiran.18692}@gmail.com , ssivanantham@vit.ac.in

Abstract

The performance of the multiplier is of high importance in any Digital Signal Processor (DSPs). Apart from the speed its precision is also plays a vital role. This is rectified by using Floating Point numbers in these multipliers. But it may consume more silicon area and power. In this paper we propose a method for fast floating point multiplication based on Dadda Algorithm. The coding is done for 32-bit single precision floating point multiplication using Verilog and synthesized using Xilinx ISE v13.4. Further the results are compared with the previous work. These results clearly indicate that our method using Dadda method can have a great impacton improving the speed and reduce the area and power consumed by the Digital Signal Processors.

Keywords- 32-bit single precision floating point format; Dadda Algorithm; Floating point.

INTRODUCTION

DSP applications require the multiplication of floating point binary numbers. IEEE standard provides the format for representing the floating point numbers [1]. The floating point numbers are represented in two major formats namely, single precision format and double precision format. The single precision format consists of 32 bits and double precision format consists of 64 bits. These formats are divided into three fields which are Sign, Exponent, and Mantissa. The figure 1 below elaborates the structure of single precision format and double precision formats of IEEE 754 standard. In single precision format mantissa is represented in 23 bits and one implicit exclusive bit ’1’ in MSB for normalization from 0 to 22, Exponent is represented in 8 bits from 23 to 30, Sign bit is represented in 1 bit which is 31st bit.

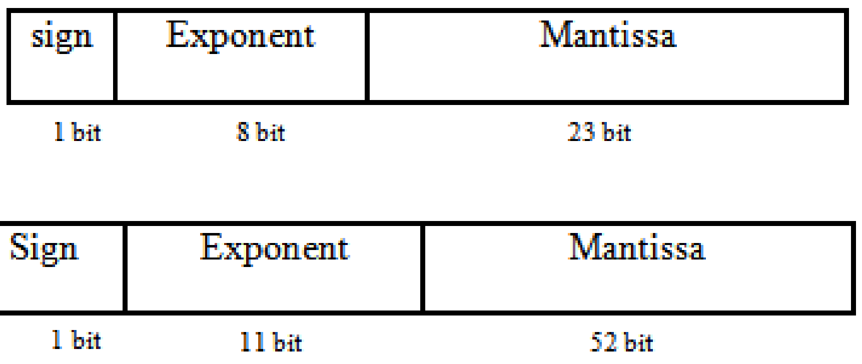

Fig 1. IEEE format for single and double precision formats

Multiplication of floating point binary numbers in IEEE754 format is performed in three steps first multiplication of the 24 bit mantissa, addition of 8 bit exponent and the resultant exponent is converted into excess 127 format and sign bits are XORed to get the resultant sign bit. In this paper we propose the Dadda multiplication algorithm [2] for the multiplication of 24 bit mantissa. The details of the Dadda multiplication algorithm and their advantages over conventional multiplication methods are discussed in section III.

The paper explains the implementation of floating point multiplier using dada algorithm technique. Section II explores the fundamentals of IEEE 754 floating point representation and implementation of floating point multiplier using dada algorithm. Section III elucidates the concept of dada multiplication technique. Section IV pictures the results and conclusions.

FLOATING POINT MULTIPLICATION

The multiplier for single precision floating point numbers can be done mainly in four different units as shown in figure 2.

1. Sign calculation unit.
2. Exponent calculation unit.
3. Mantissa calculation unit.
4. Control unit to check normalization

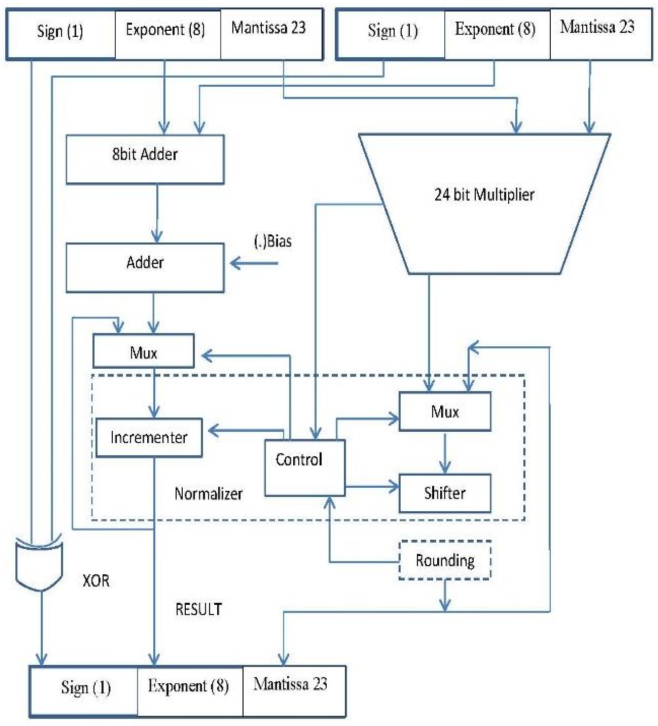

Fig 2: Proposed architecture of single precision floating point multiplier

A typical number can be represented exactly as **Mantissa*base$^{\text{exponent}}$.** The standard representation of floating point binary number in single precision IEEE 754 format is given by$\mathbf{(-1)^{sign}\ (1.b_{-1}b_{-2}b_{-3}\ldots\ldots b_{-23})\ *\ 2^{(e+127)}}$**.** The mantissa calculation unit requires 24 bit multiplier as we are dealing with single precision format. In this paper we propose the efficient use of dadda multiplication technique for this mantissa calculation unit. The exponent calculation unit in this paper is designed using 8 bit carry propagate adder. The control unit raises the flag when exceptional case occurs such as:

If exponent = 255 and mantissa ≠ 0, then it is NaN.
If exponent = 255 and mantissa = 0, then it is infinite.
If exponent = 0 and mantissa = 0, then it is zero.
If exponent > 255 then overflow occurs.

Figure 2 clearly pictures the proposed architecture for floating point multiplication.Consider a floating point representation of two numbers similar to the IEEE 754 single precision floating point format, while still maintaining the hidden bit for normalized numbers which is 1.

A = -18.0 = 1 10000011 00100000000000000000000.
B = 9.5 = 0 10000010 00110000000000000000000.

To multiply A and B’s mantissas we should extract the mantissas and adding the MSB 1.Multiplying both extracted mantissas:

100100000000000000000000
100110000000000000000000

We get the 48-bit result of the multiplication as: 0x558000000000.

The most significant digits are only useful in this 48-bit product and removing the most significant 1 and truncating the remaining bits we get the mantissa to be 01010110000000000000000. Here we used round to zero rounding technique.

Adding the exponents: The exponent representing the two numbers is biased by the bias value 127 and is not the true exponent. So, bias is subtracted from the resultant exponent.

```
                        10000011
                        10000010
               (-127 )10000001
   10000110  ____________________________
```

Obtain the sign bit by XORing the sign bits of multiplicand and multiplier and put the result together:

1 10000110 01010110000000000000000

The mantissa bits are 23 bits. Therefore rounding is not needed as we performed it earlier.

DADDA ALGORITHM

The performance of Mantissa calculation Unit determines overall performance of the Floating Point Multiplier. The picture shown in figure 3 elucidates the Dadda Algorithm. This algorithm goes on as follows:

- Multiply (logical AND) each bit of one of the inputs, by each bit of the other yielding partial product matrix.
- Reduce the partial product matrix to two vectors using series of full and half adders.
- This reduction can be done with the help of height of each level.
- The height of the preceding level is the height of the succeeding level * (1.5).
- Till the height is less than the number of bits in the operands we are operating on.
- Make the vectors into two numbers, and add them with a conventional multi bit adder.

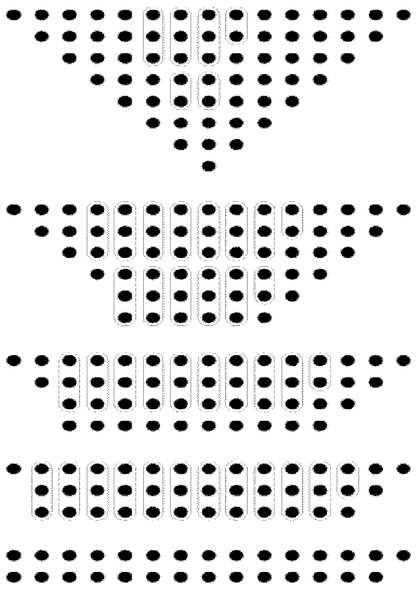

Figure.no: 3 Dadda Algorithm

MAC UNIT

In many of the DSP applications the operations performed generally involve many multiplications and accumulations. In real-time signal processing, a high speed and high throughput MAC is always a key to achieve high performance in digital signal processing processors. The main consideration of MAC design is to enhance its speed. Because speed and throughput rate is always the main objective of digital signal processing system. But in case personal communication, low power design also becomes another main theme because, energy in the battery available for these products hinders the power consumption of the personal communication system. A conventional MAC unit consists of multiplier, adder and an accumulator unit that contains the sum of the previous consecutive products. The function of the MAC unit: $F = \Sigma A_i*B_{i.}$

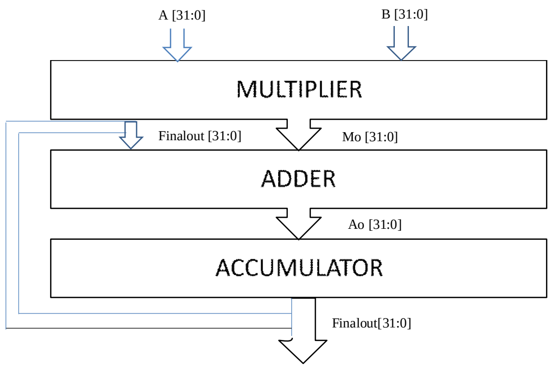

Figure 4: Structure of Mac unit

MAC unit consists of multiplier adder and an accumulator as shown in figure 4. Generally, adders implemented in MAC unit are Carry- Select adders or Carry-Save adders because speed is of high priority in DSP processors. But as we are designing floating point MAC we can design a floating point adder. The inputs for the MAC are to be given to the multiplier block of the MAC, which will multiply them and give the product to adder which will add the result and the previous accumulated results and the final result and will store the result into a memory location if required.

IMPLEMENTATION RESULTS

Table.no: 1: The above table 1 describes various parameters of the proposed multiplier when synthesized.

s.no	IC process (nm)	clock	cells	Leakage power (nw)	Dynamic power (mw)	Total power (mw)	Cell area	Net area
1	180	16730	149	31.003	0.207	0.20705	4494	1944
2	90	10000	159	8187.145	0.057	0.06604	1488	801
3	45	12450	154	16.17	0.027	0.0274	406	257

Table.no:2: The above table 2 describes various parameters of the proposed mac unit when synthesized.

s.no	IC process (nm)	clock	cells	Leakage power (nw)	Dynamic power (mw)	Total power (mw)	Cell area	Net area
1	180	16730	603	110.193	1.3719	1.372	16895	9481
2	90	10000	957	45011.338	0.5606	0.6056	7524	4947
3	45	12450	993	136.239	0.3067	0.3068	2573	1627

Table.no:3: The above table 3 compares the multiplier results with that of the previous work's results.

Parameters	This work	[11]
Device	Virtex 5	Virtex 5
Power consumption(Watt)	0.207m	27.29m
Number of LUTs	1008	966
Number of IOs	96	99

Table.no:4: The above table 4 compares the mac unit's results with that of the previous work's results.

Parameters	This work	[12]
Device	VERTEX 5	VERTEX 2P
No. of Slices	33 out of 12480	381 out of 1408
No. of input LUTs	1473 out of 12480	680 out of 2816
No. of bonded inputs	166 out of 172	67 out of 140

CONCLUSION

A floating point multiplier is designed for the calculation of binary numbers represented in single precision IEEE format is designed in this thesis. This multiplier is designed and simulated using Xilinx ISE v13.4. In this implementation exceptions like infinity, zero, NaN,overflow are considered. In this implementation rounding methods like round to zero, round to positive infinity, round to negative infinity, round to even are considered. To analyse the working of our designed multiplier we designed a MAC unit and test its performance. The design is verified using finalmac_tb test bench. Simulation results are provided within thethesis. This design is synthesized using Cadence and the corresponding results are provided.These results are compared with the previous work done by various authors. Maximum frequency of the design is 125.556 MHz and the Maximum period is 7.965ns.

REFERENCES

[1] IEEE Standard for Binary Floating-point Arithmetic, ANSI/IEEE 754, 1985.
[2] 8 x 8 bit pipelined Dadda multiplier in CMOS, IEEE proceedings, vol. 135, pt. G, no. 6, December 1988.
[3] Shanthala S, Cyril Prasanna Raj, Dr.S.Y.Kulkarni. Design and VLSI Implementation of Pipelined Multiply Accumulate Unit. Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09.
[4] Pascal C.H. Meier, Rob A. Rutenbar and Richard carley, "Exploring multiplier architecture and Layout for low power", IEEE Custom Integrated circuits Conference, 1996.
[5] Some schemes for Parallel Multipliers. L.DADDA. IEEE April 1965.
[6] M.V.Praveen Kumar, S.Sivanantham, S.Balamurugan, P.S.Mallick. Low power reconfigurable multiplier with reordering of partial products. Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011).
[7] NabeelShirazi, Al Walters, and Peter Athanas. Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing MachinesPresented at theIEEESymposium on FPGAs for Custom Computing Machines Napa Valley, California, April 1995.

[8] Kelly Liew Suet Swee, Lo HaiHiung. Performance Comparison Review of 32-Bit Multiplier Designs. 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012).

[9] Ashish B. Kharate, Prof. P.R. Gumble. VLSI Design and Implementation of Low Power MAC for Digital FIR FilterIJECCEVolume 4, Issue (2) REACT-2013.

[10] VaijyanathKunchigi, Linganagouda Kulkarni, 32-bit mac unit design using vedic multiplierInternational Journal of Scientific and Research Publications, Volume 3, Issue 2, February 2013.

[11] AniruddhaKanhe, Shishir Kumar Das, Shishir Kumar Das. Design and Implementation of Floating Point Multiplier based on Vedic Multiplication Technique2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India.

[12] Aniruddha Ghosh, SatrughnaSinghaand Amitabha Sinha. Floating Point RNS"-A New Concept for Designing the MAC unit of Digital Signal ProcessorACM SIGARCH Computer Architecture News. Vol. 40, No. 2, May 2012.