International Journal of Research in Advent Technology, Vol.2, No.3, March 2014
E-ISSN: 2321-9637

Analysis and study of different multipliers to design floating point MAC units for digital signal processing applications

Ms. Meenu S.Ravi[1], Mr. Ajit Saraf[2]
Department of electronicst[1, 2], MES's PIIT, New Panvel[1, 2]
Email: mravi@mes.ac.in[1],ajitsaraf123@mes.ac.in[2]

Abstract- Digital Signal processing became an application to create high speed data processing systems like 3D rendering, 4G mobile internet, etc., we need best processors with high performance data path units and there is a growing need for research on alternative methods for signal processing hardware implementation. In most systems using digital signal processing Multiply-Accumulate (MAC) is one of the main functions. The performance of the whole system depends on the performance of the MAC units in place. Regardless of that these days' real time signal processing systems require high throughput and high performance MAC units.
Index Terms— MAC Unit, 3D rendering, 4Gmobile internet.

1. INTRODUCTION

Because of the high-accuracy, impressing dynamic range and use of easily operable rules, floating-point operations have found applications in many fields that require the above mentioned qualities. At present in computers, floating- point arithmetic operations are mainly performed by the co- processors. In systems without floating-point hardware, the CPU emulates it with a series of simpler fixed-point arithmetic operations that run on the integer arithmetic and logical unit. This saves the added cost of a floating-point unit (FPU) but is too slower compared to the traditional systems. Coprocessors cannot fetch instructions from the main-memory; perform I/O, manage-memory and so on. These processors require the host main processor to fetch the coprocessor instructions from the memory and handle all operations aside from the coprocessor functions. High processor speeds demand high coprocessor operation speed.

In this paper a review of various multiplier systems used for floating point MAC design have been studied and analyzed, such as new radix-4 booth encoding algorithm, Vedic algorithm and design using configurable devices.

1.1 *Floating point number formats*

The IEEE has standardized the representation for binary floating point numbers in IEEE 754.The standard is followed by almost all new generation machines. Prior to IEEE 754 standard, computers used various forms of floating point. These differed in word sizes, the format of the representations, and the rounding behavior of operations. These differing systems implemented different parts of the arithmetic in hardware and software, with varying accuracy. The IEEE 754 standard was created after word sizes of 32 bits (or 16 or 64) had been generally settled upon. The table 1 shows the number of bits in the exponent significant and range of both single and double precision floating point numbers.

Table 1. IEEE formats of floating point number.

	SIZE (BITS)	*EXPONENT (BITS)*	*SIGNIFICANT (BITS)*	*RANGE*
Single precision	32	8	23	$2*10^{+/-38}$
Double precision	64	11	52	$2*10^{+/-38}$

The single precision floating point number is calculated as $(-1)^{s}$ x 1.F x $2^{(E-127)}$, and the double precision floating point number is calculated as $(-1)^{s}$ x 1.F x $2^{(E-1023)}$.

1.2 Floating point multiplication

We can represent a floating point number as the equation (1),

$$Z= (-1)^{S} \times 2^{(E-Bias)} \times (1.M)\ldots\ldots\ldots\ldots\ldots (1)$$

The multiplication algorithm is as follows
(1) Multiplying the significant i.e., ($1.M_1$ x $1.M_2$).
(2) Placing the decimal point in the result.
(3) Adding the exponents i.e. (E_1+E_2-Bias).
(4) Obtaining the sign i.e.S_1 XOR S_2.
(5) Normalizing the result, i.e. obtaining 1 at the MSB of the result significant.
(6) Rounding the results to fit in the available bits.
(7) Checking for underflow/overflow occurrences.

1. NEW RADIX-4 BOOTH ENCODING ALGORITHM

The floating point multiplier is especially used for a floating point FFT processor; the requirement for operation speed is much more. For the floating point multiplier to be working stably at a frequency of 80MHz, the pipeline technique is adopted in the design, in which the combinational logic circuits are inserted between the levels to save the data temporarily. The clock frequency is increased greatly, because the delay path length of the signal through the combinational logic circuit is shortened in one clock cycle. The running frequency is equal to that of the clock to the synchronous sequential logic circuit. The faster the synchronous clock, the shorter the interval between the output, and the data quantity will be greater, therefore the greater the throughput. Consequently, the running velocity of circuit is speeded up greatly by the pipeline technique. The FPGA has been well suited for adoption of the pipeline design technique because of their characteristics and structure. That is, arithmetic operations are decomposed into several small basic Operations and configured into the LE (Logic Elements), and then the middle values produced is saved to the temporary registers, and the operations are performed in the next clock cycle. Therefore, the unnecessary extra cost is needed if the pipeline technology is adopted in FPGA. As we divide the hardware system, five levels of pipeline scheme and Booth's encoder and partial product compressor improved in their structure, along with a carry look-ahead adder are adopted to satisfy the requirement of the high speed in the design. But in this review paper only the multiplier section is under consideration. The hardware structure of the floating-point multiplier is given in Fig (1).

As per the floating point multiplication rules, the multiplication by their mantissa each other is only is made, and their exponents are added up in parallel and the overflow is checked for the multiplication of a floating-point number(A)by another one(B). For the 32-bits floating point numbers, their exponents have 8 bits, therefore, the operations of addition of two exponents and remove of the offset is implemented by an adder having carry output with a look-ahead carry.

The encoding algorithm is shown in Table 3.1, where "Zero" expresses "the operation of the mantissa in the multiplicand is 'multiplied by 0', "; "NEG" expresses "the operation sign of the mantissa in the multiplicand", that is , "0" expresses "positive", and "1" expresses "negative" ; The combination of "A_S1" and "NEG" expresses "the operation of the mantissa in the multiplicand is 'multiplied by 1 or -1'"; The combination of "A_S2" and "NEG" expresses "the operation of the mantissa in the multiplicand is 'multiplied by 2 or -2' ".

3 bit encoding[2:0]	Zero	A_S1	A_S2	NEG
000	1	0	0	0
001	0	1	0	0
010	0	1	0	0
011	0	0	1	0
100	0	0	1	1
101	0	1	0	1
110	0	1	0	1
111	1	0	0	0

Table. 1. Truth table of booth encoding

2. VEDIC ALGORITHM

It is one of the fastest growing technologies in this century. Faster additions and multiplications are having extreme importance in DSP applications such as convolution, discrete Fourier transforms digital filters, etc. The main computing process is always a multiplication routine. Vedic mathematics is the name given to the ancient system of mathematics, which was rediscovered, from the Vedas between 1911 and 1918 by Sri Bharati Krishna Tirthaji. The whole of Vedic mathematics is based on 16 sutras (word formulae) and manifests a unified structure of mathematics. As such, the methods are complementary, direct and easy.

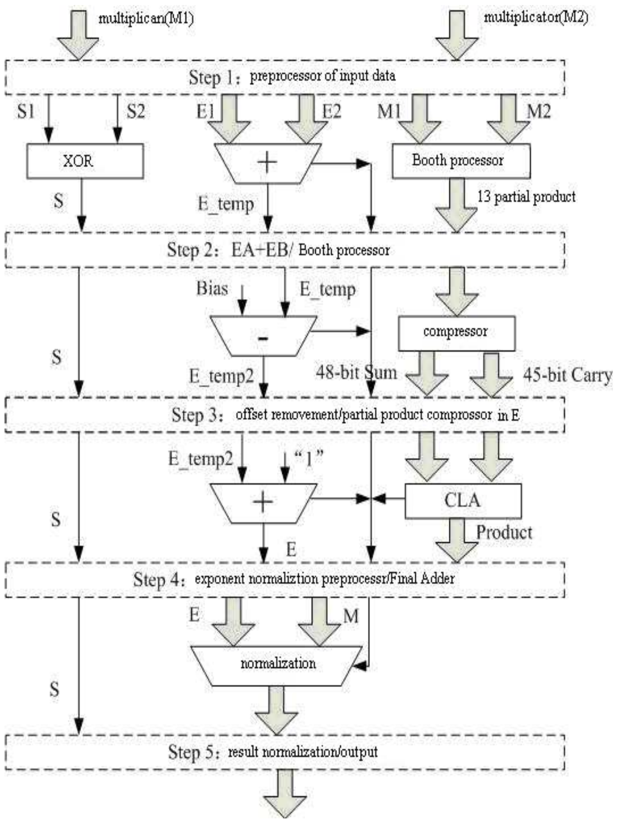
multiplican(M1)
multiplicator(M2)
Step 1: preprocessor of input data
S1
S2
E1
E2
M1
M2
XOR
+
Booth processor
S
E_temp
13 partial product
Step 2: EA+EB/ Booth processor
Bias
E_temp
-
compressor
S
E_temp2
48-bit Sum
45-bit Carry
Step 3: offset removement/partial product compressor in E
E_temp2
"1"
+
CLA
S
Product
E
Step 4: exponent normaliztion preprocessr/Final Adder
E
M
normalization
S
Step 5: result normalization/output

Fig..1. The architecture of the 32-bits floating point multiplier

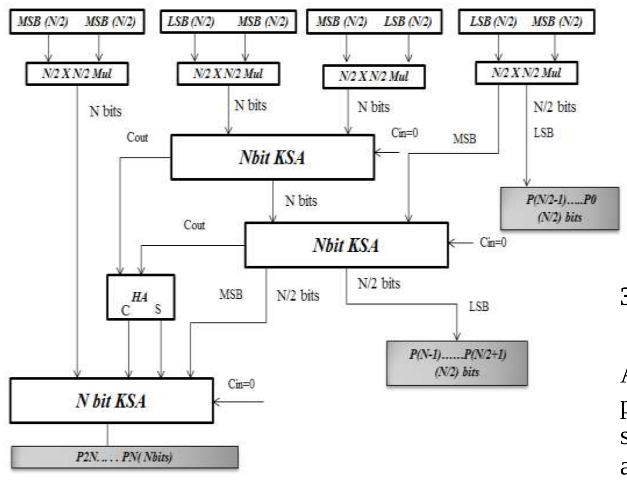

Fig.2. the Vedic multiplier

For 3 digit numbers, the line diagram is represented as follows:

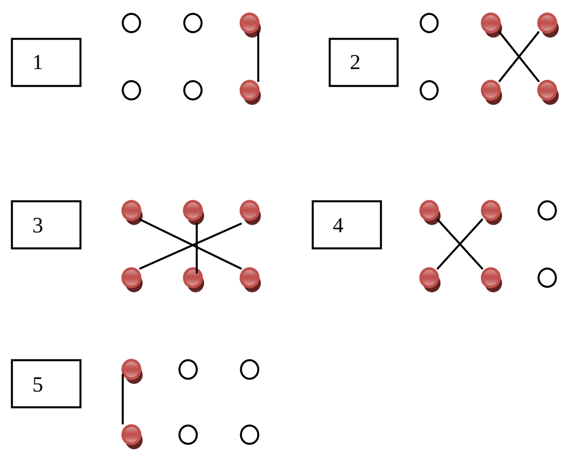

Fig.3. 3digit numbers multiplication

For 4 digit numbers, the line diagram is represented as follows:

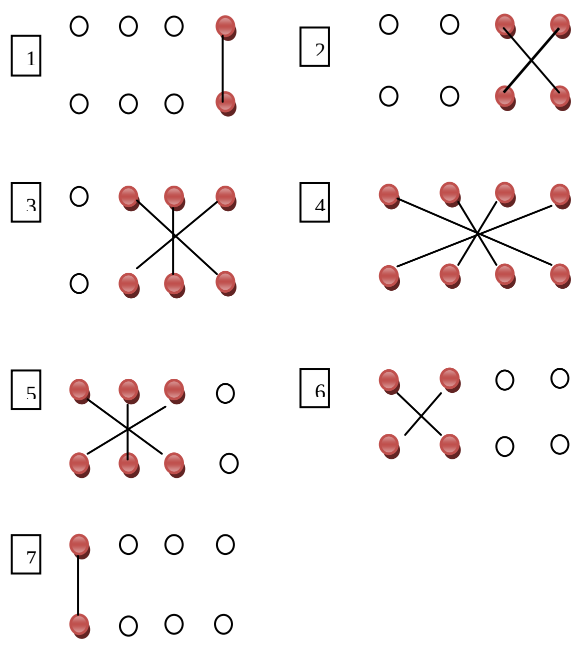

Fig.4. 4digit numbers multiplication

3. USING CONFIGURABLE DEVICES

As explained in previous sections, a binary floating-point number is represented as a sign bit, the significant and the exponent. Given two numbers A and B, the flowchart in figure 2 can be used to find their product, given that fracA, fracB and eA, eB are the significants and exponents of the numbers, respectively. The detailed description of the algorithm is as follows:

1. The hidden bit (24th bit) is made explicit. If ea or eb = 0, it is made '0', otherwise a '1'. At this point 33 bits are needed to store the number, 8 for the exponent, 24 for the significant and 1 for the sign.
2. The result of the multiplication is given by the formula:
Sign =sign A xor sign B,
e = eA + eB,
Frac = fracA x fracB

The addition of the exponents is a trivial operation. This means that in order to get the right result, we have to subtract **127** (bias) from their sum. The sign of the result will be the XOR of the two sign bits. The multiplication of the significants is an unsigned, integer multiplication.
3. The product of two 24-bit numbers will be a 48-bit data. But only, 24 bits can be accommodated in the significant position. Therefore, 48-bit result rounds up to 24 bits. There are four different rounding methods: Round-up, Round-down, Round-to-nearest-even, and round-to-zero; in which round-to-nearest-even is the most used. Since the precision of result is not infinite, sometimes rounding is necessary.

This method attempts to design hardware architecture for single precision floating-point multiplication that is easily implementable with high efficiency, and the hardware used to implement is FPGA.

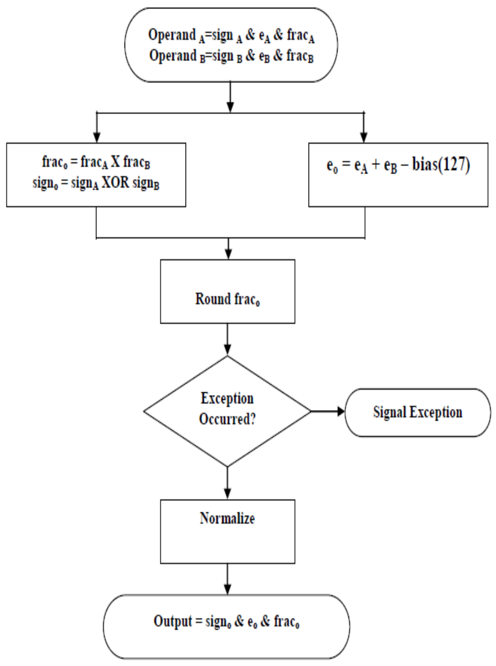
Operand $_A$=sign $_A$ & e_A & frac$_A$
Operand $_B$=sign $_B$ & e_B & frac$_B$
frac$_o$ = frac$_A$ X frac$_B$
sign$_o$ = sign$_A$ XOR sign$_B$
$e_o = e_A + e_B$ – bias(127)
Round frac$_o$
Exception Occurred?
Signal Exception
Normalize
Output = sign$_o$ & e_o & frac$_o$

Fig.5 the Vedic multiplier

3. CONCLUSION

Vedic mathematical methods are extracted from ancient systems of computations, now made globally available to everyone through the great work done by Swami Sri Bharati Krisna Tirthaji Maharaja, who has published a book on Vedic mathematics in 1965. Compared to other conventional mathematical methods, these are faster and easy to compute. It has been one of the fastest algorithms in computations due to the reduction in number of intermediate products (cross products). Therefore, such approaches have been extremely beneficial in digital signal processing applications. The engineers can develop such fast and low cost devices if it is included in engineering education. The floating point multiplier can be stably run at the frequency 80 MHz if it is implemented on a configurable device such as FPGA. Then the multiplier can be successfully adopted in the FFT Processor.

Acknowledgments

I am thankful to my guide Assistant Professor Ajit Saraf (Pillai Institute of Information Technology, India) for the helpful discussions and valuable guidance.

REFERENCES

[1] Dhananjaya A; Dr. Deepali Koppad(2013): “Design of high speed floating point MAC using Vedic multiplier and parallel prefix adder” : International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 6, june-2012 ,ISSN: 2278-0181, pp 3359-3363.
[2] Dinesh Kumar1;Girish Chander Lall2 (2013) :
“Simulation And Synthesis Of 32-Bit Multiplier Using Configurable Devices”: International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 6, June – 2013, ISSN: 2278-0181, pp. 216-223.
[3] Anitha R; Alekhya Nelapati; Lincy Jesima W; V. Bagyaveereswaran(2012):”
Comparative Study of High performance Braun’s Multiplier using FPGAs”: IOSR Journal of Electronics and Communication Engineering (IOSRJECE) ISSN: 2278-2834 Volume 1, Issue 4 (May-June 2012), pp 33-37
[4] Purushottam D. Chidgupkar; Mangesh T. Karad(2004):“The Implementation of Vedic Algorithms in Digital Signal Processing”: Global J. of Engng. Educ., Vol.8, No.2 © 2004 UICEE
Published in Australia, pp 153-158.
[5] Gong Renxi; Zhang Shangjun;Zhang Hainan; Meng Xiaobi;Gong Wenying;Xie Lingling; Huang Yang;(2009),”Hardware Implementation of a High Speed Floating Point Multiplier Based on FPGA”:Proceedings of 2009 4th International Conference on Computer Science & Education, pp 1902-1906.