VLSI Based Low Power FFT Implementation using Floating Point Operations

Pooja Andhale , Manisha Ingle

Abstract—**This paper presents low power floating point FFT implementation based low power multiplier architectures such as FPBZFAD and FPVM. Power is more important and major factor for any system. FFT is widely used in major DSP applications. Fast Fourier Transform processors use butterfly diagrams which requires arithmetic operations like addition, subtraction and multiplication frequently. Floating Point Arithmetic provides a wide dynamic range, freeing special purpose processor designers from the scaling and overflow, underflow concerns that arise with fixed-point arithmetic. Multipliers are responsible for power consumption as they consist of switching activity. In this paper, a low-power architectures like Floating Point based Bypass Zero Feed 'A' Directly (FPBZFAD) and Floating Point based Vedic Multiplier (FPVM) for FFT is proposed and Power consumption is compared with Shift and Add architecture. Thus when these multipliers are used for FFT, the total power consumption by FFT with FPBZFAD and FPVM is reduced up to 50%. The architecture is implemented in Verilog Hardware Description Language and synthesized in Quartus II software.**

Index Terms—**Floating Point Arithmetic, Fast Fourier Transform, Digital Signal Processing, FPBZFAD, FPVM.**

I. INTRODUCTION

Digital Signal Processing Systems are changing from Fixed Point to Floating Point for arithmetic operations. Fixed-point arithmetic has been used for the longest time in computer arithmetic calculations due to its ease of implementation compared to floating-point arithmetic and the limited integration capabilities of available chip design technologies in the past, but now advanced technology applications require a data space that ranges from the infinitesimally small to the infinitely large. Such applications require the design of floating-point hardware. A floating point number representation can simultaneously provide a large range of numbers and a high degree of precision.

Manuscript received July, 2015.

***Pooja Andhale**, E & TC department Maharashtra Institute of Technology, Pune,India,9405404529.*
***Manisha Ingle**, E & TC department, Maharashtra Institute of Technology, Pune,India,7720800505,*

Floating point arithmetic is efficient for the implementation for a variety of Digital Signal Processing (DSP) applications because Designer can directly focus on the architecture without worrying the numerical issues like underflow, overflow and scaling. Up to 1985 each company had their own standards, in 1985 IEEE-754 was introduced which gives the standard format for floating point representation [3]. According to IEEE-754 there are two basic standards, first is Single Precision Number and second is Double Precision Number. Table 1 shows the specifications for both. Single precision and double precision formats are used based on the application requirements. These standards can be used for arithmetic operations like Multiplication, Subtraction, Addition as well as Division. According to the standard of Floating Point Number Standard Table 2 below shows what a number represents for different values of sign, exponent and Significand.

TABLE I. Floating point number format

Type	**Sign**	**Exponent**	**Significand**
Single Precision	1[31]	8[30-22]	23[22-00]
Double Precision	1[63]	11[62-52]	52[51-00]

TABLE 2. Number Representation in Single precision

Single Precision Number		**Number Represented**
Exponent	**Significand**	
0	0	0
0	Nonzero	Denormalized Number
1 to 254	Anything	Floating Point Number
255	0	Infinity
255	Nonzero	Not A Number

Fast Fourier Transform Processors frequently require these operations. Few years ago, power reduction and low power Fast Fourier Transform Processors frequently require these operations. Few years ago, power reduction and low power

devices were not very demanding as the device was enough to meet the requirement as device density was not an issue at that time, but today most of the devices are packed with many of processors in a single chip for increased performance with reduced size. So we have to concentrate on power constraint also that's why in the architectures given lowers the switching activity which takes place in traditional multipliers, So that total power of processor gets reduced.

Section II consists FFT processor algorithm. Floating point based BZFAD multiplier is explained in section III. Section IV is containing floating point based Vedic multiplier. At the end complete comparison is done in results and conclusion.

II. FAST FOURIER TRANSFORM ALGORITHM

Discrete Fourier Transform (DFT) computation can be done efficiently by using Fast Fourier Transform (FFT). Basically there are two types of FFT algorithm, Decimation in Time and Decimation in frequency. The speed, area, power consumption and energy consumption of the butterfly operation have a direct impact on the overall performance of the FFT. All lines carry complex pairs of 32-bit IEEE-754 numbers and all operations are complex. This is the Basic module for FFT. We can find for different Radix-N data. For computation of FFT we always require addition, subtraction and multiplication operation [2] [9].

A. Floating point Add-Subtract unit

The floating point add-subtract unit can be shown as in Fig. 1. The total procedure for addition is given as follows:
1) First exponent of A and B is compared by Exponent compare logic and difference is calculated.
2) The exponent comparison difference is used for the significand swap logic. When exponents are equal there is no need to go for swapping.
3) The smaller number's significand is going to shift by the amount of difference of exponents.
4) These two Significand directly go to the Add-Subtract block. This time it will perform operation like add or subtract. For subtraction LZA block is used. The LZA i.e. Leading Zero Anticipator detects the number of 0's so that the subtraction result is immediately normalized.
5) After getting result we will require it in normalized form so it will go for normalization block and exponent adjust block will adjust the exponent of result accordingly.
6) To get the result in 32 bit it will go to post normalization block and we will get the final Significand of the result.
7) Sign logic block will take care of the sign of the Result [1].

B. Floating point multiplication unit

According to IEEE standard format multiplication of single precision number, it consists of:
1] Sign calculation unit
2] Exponent calculation unit
3] Significand calculation unit

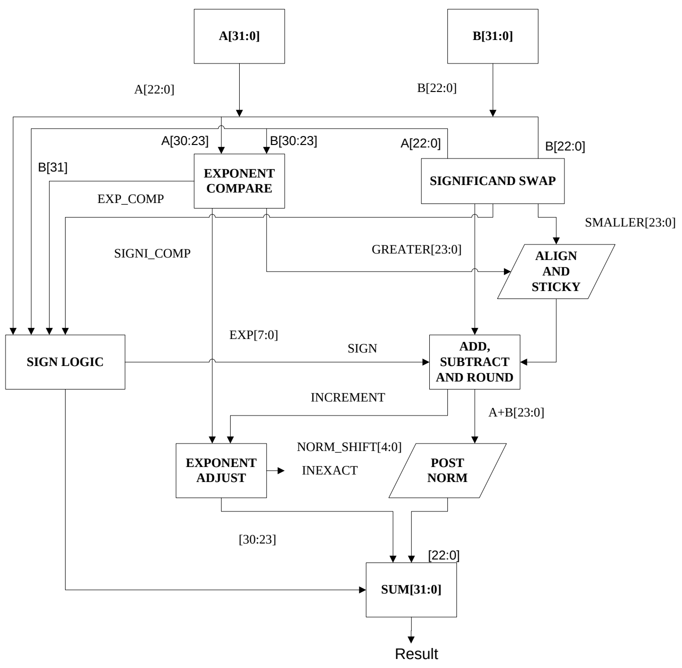

Fig 1. Floating point add-subtract unit

Sign Logic will check for the sign of the result of multiplier. By doing XOR of both sign we will get the exact sign for result. The exponents of both numbers first added with bias individually then exponents with bias are added after that bias is subtracted from the result and this will be the exponent of result. Now the two significands are multiplied by shift and add method. Here we have 24 bit significands for both numbers therefore during this process it takes so many switching activities. In this process there is maximum power consumption and because of that FFT also have maximum power consumption so to get low power FFT we have to replace this multiplier with low power multiplier so this paper has given new multiplier concept for FFT calculation [7].

III. FLOATING POINT BASED BZFAD MULTIPLIER (FPBZFAD)

The traditional floating point multiplication can be done in three sections. i) Multiplication of mantissa ii) xoring of sign bit and iii) addition of exponents. The power in this structure can be reduced by using power efficient mantissa multiplication majorly. Power consumption in the exponent addition can also be significantly reduced by using adders such carry save adder (CSA). Here, we are using low power by pass zero- feed A directly (BZ-FAD) .The major advantage of BZ-FAD structure is we can bypass the direct passage of one of the multiplicand and partial product shift, whenever they are not required. It minimizes switching activity of the circuit drastically, thereby reducing the power. In Shift and Add multiplier, there are many factors like signal activity on counter, shifting of multiplicand and partial products etc. To eliminate all these power dominating factors we have designed BZ-FAD multiplier for significand or mantissa multiplication.

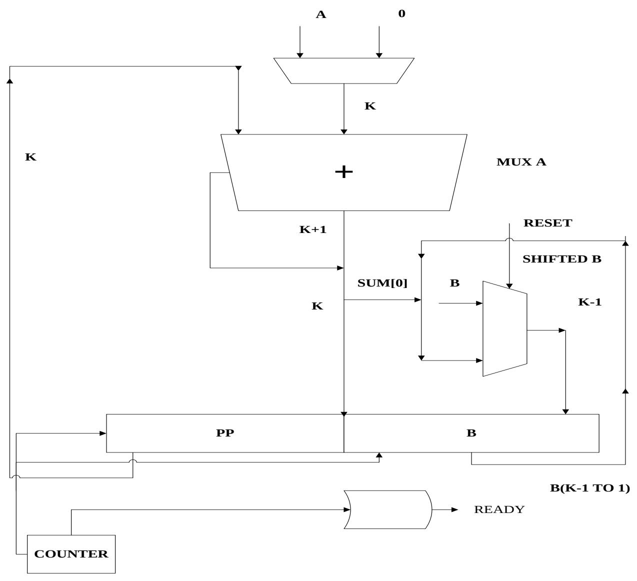

Fig. 2. Traditional multiplier architecture

A.Shift of the multiplicand

In the traditional multiplier architecture Figure 2, the one multiplicand, say B. depending upon the LSB bit of B,B(0) it has to decide whether another multiplicand , say A, need to be added to partial product or not. Hence B has to shift at each cycle, so as to generate partial product. This increases the signal activity and hence power [4].

B .Power reduction in Adder circuit

In traditional multiplier, depending upon the bit B(0), each time, at each clock cycle, A has to be added to partial products. Hence when B(0) is 0, partial product would be same as that of previous. But still, some power will be waste due to unwanted transition while adding it to partial product. This can be avoided in FPBZFAD (Figure 3)structure by the provision of two registers i) feeder register and ii) bypass register. In this structure, at each current clock cycle, the LSB bit of B register B(0) for next clock cycle is checked. If it is zero, then there is no need of addition operation of A and partial products, at this time bypass register is used to store the partial product generated at that particular clock cycle. If for the next clock cycle, addition operation is required, then feeder register is used to stored current partial product generated. This partial product is then fed in the next clock cycle. To select bypass or feeder register, we have used a NAND and NOR combinational logic. In this way, this entire circuitry will determine, whether partial products will generate from feeder register or bypass register [6]. Hence whenever the LSB of B is zero, no more transitions are present in the adder circuit. Previous partial products are preserved in the bypass register and feeder register maintain its same value. Another input is A, which remains same during multiplication operation. This detaches multiplexer and A is directly provided to adder, yielding reduction in the power.

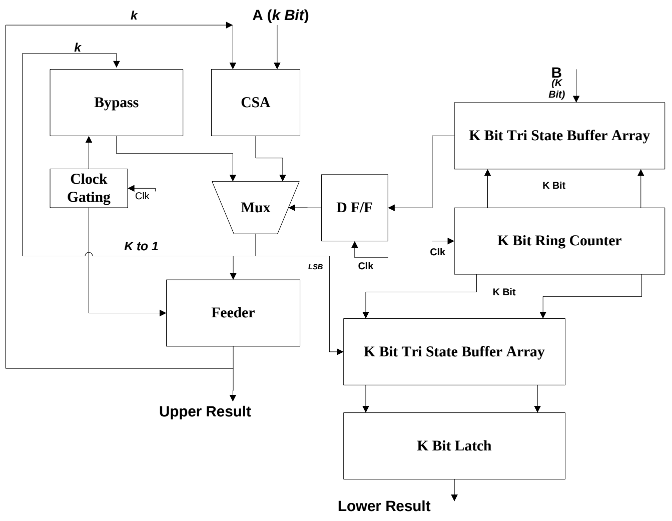

Figure 3. Low power multiplier architecture (FPBZFAD)

C.Partial product generation

In the design of traditional multiplication (shown in Fig. 2), partial product is moved or shifted in each clock cycle, giving rise to the transitions and responsible for the power consumption. Multiplication is finished when most significant bits are processed. In conventional structure, the partial products are shifted until all bit multiplication is performed [6]. We can cease the power consumption due to this, by not shifting the least significant bits of generated partial product. As shown in the figure 3, K numbers of latches are used to save lower half of the partial products in the ring counter. The ring counter is provided with sample and hold circuit, which ensures that nth bit of counter, is one by nth clock cycle. In this way generated partial product bits are stored properly into the latch. In the first clock cycle, when pp (0) bit is determined, it is stored into the rightmost latch [8]. By this way, there is no need to shift the partial product. Only higher part of the partial product is shifted while lower part remains same.

D. Low power ring counter and clock gating

We have made use of ring counter which shifts the 1 from right to left, at each clock cycle. Hence only 1 is shifted at a time, hence just two flip flops are triggered. So this ring counter is called as low power ring counter. The counter is subdivided into number of groups or blocks and which are clock gated so as to minimize transition and power.

IV. FLOATING POINT BASED VEDIC MULTIPLIER (FPVM)

The technique used is ‘Urdhva-Tiryakbhyam’ sutra. It does the multiplication by generating partial product parallel and adding it simultaneously. So the switching activity gets reduced which is helpful for the Power reduction of FFT [5]. The architectures for 3×3, 6×6, 12×12, 24×24…N×N bit modules are discussed. In this project we are going for 24 bit Multiplication. To get output for 24 bit first we have designed

A. 3×3 Vedic multiplier block

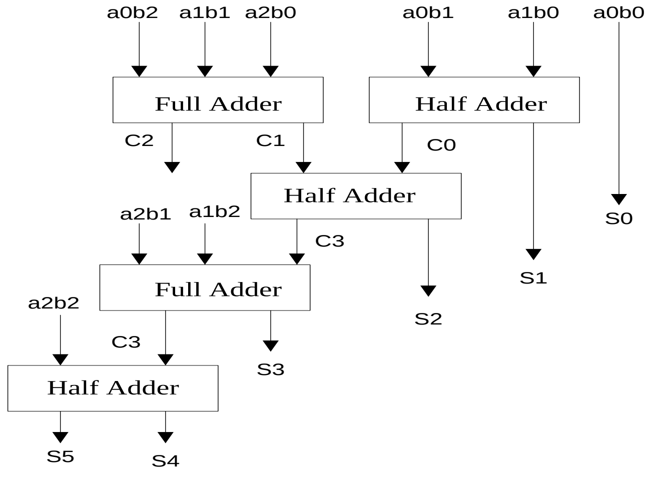

Fig. 4. 3 × 3 Vedic multiplier block

3×3 multiplier which is used for 6×6 and so on for 24×24 [5].

To explain this method let us consider 2 numbers with 3 bits each and the numbers are A and B where A=a2a1a0 and B=b2b1b0 as shown in the below line diagram. First the least significant bit (LSB) bit of final product (vertical) is obtained by taking the product of two least significant bit (LSB) bits of A and B is a0b0. Second step is to take the products in a crosswise manner such as the least significant bit (LSB) of the first number A (multiplicand) is multiplied with the second bit of the multiplicand B in a crosswise manner. The output generated is 1-Carry bit and 1bit used in the result as shown below. Next steps can be shown as given below.

$$S0 = a0b0 \qquad (1)$$
$$S1 = a0b1 + a1b0 \qquad (2)$$
$$S2 = a0b2 + a2b0 + a1b1 + c1 \qquad (3)$$
$$S3 = a1b2 + a2b1 + c2 \qquad (4)$$
$$S4 = a2b2 + c3 \qquad (5)$$
$$S5 = c4 \qquad (6)$$

B. 24×24 Vedic multiplier block

We have seen the basic 3×3 multiplier block. Now for designing of the 24 bit multiplier we require 3×3, 6×6, 12×12 multiplier blocks. As shown in the above figure for 6 bit multiplier we have used 3 bit multiplier. For 12 bit multiplier we have used 6 bit multiplier and for 24 bit we have used 12 bit multiplier. Here we have one additional block RCA i.e. Ripple Carry Adder. Ripple carry adder is designed using multiple full adders to add 24-bit numbers. Each full adder inputs a C_{in} which is the C_{out} of the previous adder. In the Ripple Carry Adder every carry bit gets rippled to full adder which is next to him. It has very fast design time because of its simple layout. Ripple adder is a combination of 4 full adders in which output carry is used as input carry to the next full adder. RCA uses large number of AND, OR, NOT gates. It has the advantages of high speed and less delay. Thus it is a simple technique for multiplication with lesser number of steps and also in very less computational time [5].

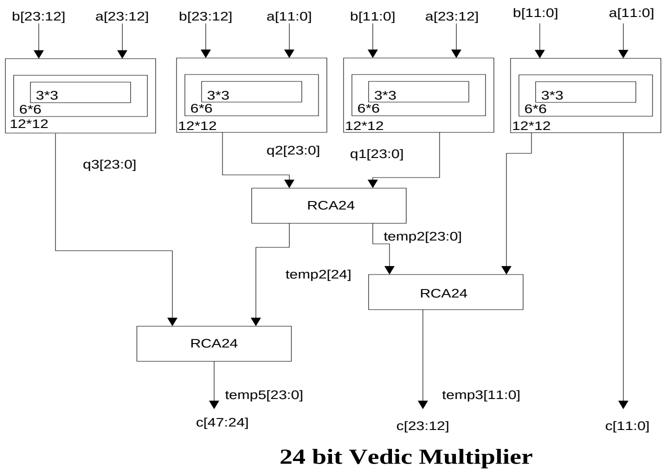

Fig 5. 24×24 bit multiplier

V. RESULTS AND DISCUSSIONS

To get the improved architecture for FFT, we implemented three different Radix-4 FFT with different multipliers. Traditional, FPBZFAD and FPVM architecture. The power consumed by FFT with FPBZFAD, FPVM and Traditional Shift and Add Multiplier is shown in Fig below. The power consumed is given in milliWatt. As Vedic multiplier has recursive architecture, it contains more area as compared to other structures. Hence, proposed structures are used as per the area and power requirements.

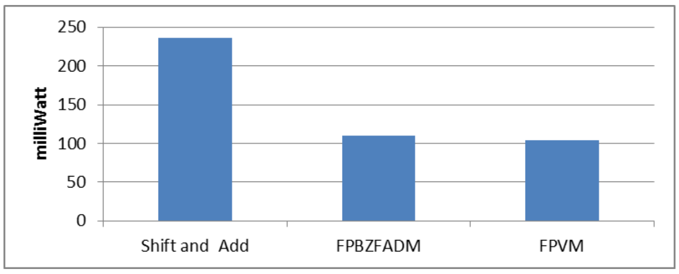

Fig 6. Power comparison of various architectures

According to the number of registers required for the total design the Area is compared and can be shown as given in the Fig. 7

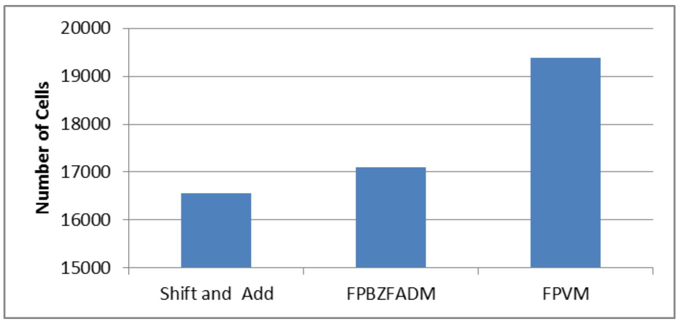

Fig 7. Area comparison of various architectures

VI. CONCLUSION

Thus proposed work shows various approaches to reduce the power of traditional FFT up to some extent. The Power consumed by FFT with FPBZFAD and FPVM is almost reduced up to 50% as compared to FFT with Shift and Add Architecture. The area required for Normal FFT is greater than the remaining two.

VII. REFERENCES

[1] Jongwook Sohn, and Earl E. Swartzlander, "Improved Architectures for a Fused Floating-Point Add-Subtract Unit" IEEE transactions on circuits and systems—i: regular papers, vol. 59, no. 10, october 2012

[2] Earl E. Swartzlander and Hani H.M.Saleh "FFT Implementation with Fused Floating-Point Operations" IEEE Transactions on Computers,vol.61,No. 2,February 2012

[3] *IEEE Standard for Floating-Point Arithmetic*, ANSI/IEEE Standard 754-2008, New York: IEEE, Inc., Aug. 29, 2008.
[4] M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram "BZ-FAD: A Low-Power Low-Area Multiplier based on Shift-and-Add Architecture" *IEEE Trans. on VLSI Systems, 2008*

[5] Bhavani Prasad.Y, Ganesh Chokkakula, Srikanth Reddy.P and Samhitha.N.R "Design of Low Power and High Speed Modified Carry Select Adder for 16 bit Vedic Multiplier" ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

[6] Premananda B.S., Samarth S. Pai, Shashank B, Shashank S. Bhat "Design and Implementation of 8 Bit Vedic Multiplier" IJAREEIE Vol. 2, Issue 12, December 2013

[7] Akash Kumar Gupta , Birendra Biswal "A High Speed Floating Point Dot Product Unit" International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT),2014

[8] P. Sheela M. Ramesh Kumar "Low Dense Packed Fused Floating Point FFT with Optimized Complex Structures" International Journal of Applied Research & Studies ISSN 2278 – 9480

[9] Takahashi, "A Radix-16 FFT Algorithm Suitable for Multiply-Add Instruction Based on Goedecker Method," Proc. Int'l Conf. Multimedia and Expo, vol. 2, pp. II-845-II-848, July 2003.

[10] Manuel de la Guia Solaz, WeiHan, and Richard Conway "A Flexible Low Power DSP With a Programmable Truncated Multiplier" IEEE transactions on circuits and systems—i: regular papers, vol. 59, no. 11, november 2012

ME (Digital Systems)
Maharashtra Institute of Technology.
Pune, Maharashtra ,India