**International Journal of Electrical and Electronics Research** ISSN 2348-6988 (online)
Vol. 3, Issue 4, pp: (70-75), Month: October - December 2015, Available at: www.researchpublish.com

# Power and Area Minimization of Reconfigurable FFT Processor Using Distributed Arithmetic

[1]Anuradha Mokashi, [2]Vishal Raskar

[1,2] Electronics & Telecommunication Engineering Department, Savitribai Phule, Pune University, (India)

***Abstract:*** **Fast Fourier transforms is one of the most important frequency analysis in signal processing. It has different application such as image processing, medical field, communication system, spectral analysis etc. Butterfly is the basic elements of FFT. In this work a Distributed arithmetic technique is used to implement the butterfly module. Distributed arithmetic is Multiplierless technique resulted more efficient butterfly element both in terms of power and area. Butterfly element is the most important building block of Reconfigurable FFT processor. Single precision is used to represent the data. IEEE 754 standard is used to represent the floating point numbers.**

***Keywords:*** **Digital signal processing (DSP), Complex multiplier, Distributed Arithmetic, FFT.**

## I. INTRODUCTION

Fast Fourier transforms is the computation of DFT which reduces the complexity in terms of operations like multiplications & additions. The DFT and FFT results the same the only difference is that it will remove the unnecessary operations. It uses the periodicity property of twiddle terms which is mentioned after.

The FFT computation for x(n) is given by the following formula

X (k) =$\sum_{n=0}^{N-1} x(n)e^{-j2\prod nk/N}$

Where $W_N^k = e^{-j2\prod nk/N}$ which is known as twiddle factor.

$x(n) =$ input sequence

X (k) = output sequence.

Distributed arithmetic is the Multiplierless technique is used which saves the power & area also. It come into existence in 1974.FFT computation using conventional & Vedic method required the high power and area compared to distributed arithmetic technique. In DA it takes only adders and shifter for operations instead of using multiplier block. Basic advantages of DA are it enables the functionality in field programmable gate array and ASICs design. It takes the advantages of look up tables. Reconfigurable is the technique in which we can use switch to select the radix point without disturbing the hardware. It increases the flexibility.

A generic low power reconfigurable architecture features high efficiency in terms of area and power which is applicable to most still picture image compression standards, telecommunication protocol and automatic control.

## II. BACKGOUND STUDY

Out of two algorithms DIF (discrete in time) and DIF (discrete in frequency), DIF is mostly used for implementation of FFT. This radix2 DIF FFT algorithm is starts by dividing the samples in to two groups. It goes till the only computation between two samples. This two point DFT is called as Butterfly block. The butterfly is the basic block of FFT computations. The diagram is given below.

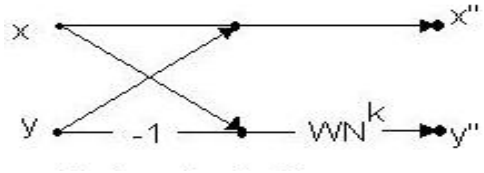



**Fig1.Butterfly Block**

$x^{II}$=x+y.....1

$y^{II} = (x - y)W_N^k$.....2

Where x and y are the input signals. Butterfly implementation is lead saving in terms of silicon area and power. The main objective of this paper is to implement the highly efficient butterfly design which could be used in reconfigurable FFT architecture. Butterfly design using distributed arithmetic technique is implemented. In this architecture distributed arithmetic is used only for the butterfly implementation not the whole FFT architecture. For the design of butterfly it requires two 32 bit inputs (16+16) and one twiddle factor. These numbers are represented in 16 bit floating point number which is represented in standard IEEE 754 format. We cannot be representing the number in binary form if the number is more or very less. This limitation of binary number representation removed in floating point number representation. Hence we prefer the floating point representation. Instead of using base 10 and power of 10 like in scientific notation, IEEE 754 uses binary fraction and exponent which is considered as a power of 2.The floating point reprentaion using IEEE 754 format is given below.

| | | |
|---|---|---|
| Sign 1bit | Exponent<br>5 bits | Fraction/Mantissa<br>11 bits |

**Fig2. Floating point representation using IEEE 754 format**

Given number is in normalised form in which sign bit along with 5 bit exponent and 10 bit mantissa including 1 hidden bit (11 bit mantissa) is distinguished. Hidden bit is always taken an account in calculations. Multiplication in conventional FFT requires more power which in case of DA is completely eliminated. Multiplication with twiddle factor is the main critical task in FFT computation, here in LUT based approach look up tables are prepared based on the inputs and then after proper bit adjustments and shifting these LUTs are accessed based on bit combination.

Algorithm is given below.

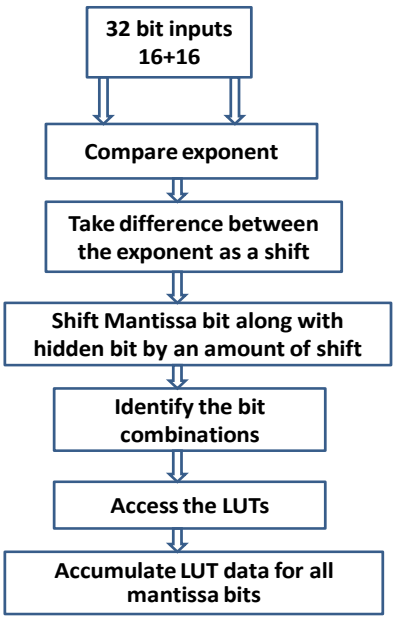



**Fig 3.Flow of operations**

## III. DESIGN AND IMPLEMENTATION

Working with 16-bit floating point numbers (imaginary and real separately) Fig. 3 shows the bit manipulation happens with the 16 bit imaginary and real and imaginary part of the data. Sign bit chooses the set of LUT to be taken into account.

5-bit exponent bits used to equalise the number by proper biasing and then start fetching twiddles from LUTs based on mantissa bit combinations of imaginary and real numbers. These mantissa bits are obtained after shifting the mantissa bits with lesser exponent by an amount of the difference between exponents. This is explained more elaborately in the example given below.

Let DAA is applied to a floating point butterfly module. Let rewrite the butterfly module equation 1 & 2

$x^{||}$=x+y

$$y^{||} = (x - y)W_N^k$$

Where x and y are the input sequence which is complex numbers.

Hence

x = xr + jxi.....3

$y = yr + jyi$.....4

$$xr, yr = \text{real part}$$

$xi, yi$ =Imaginary part

$W_N^k = Wr + jWi$.....5

$x^{||} = x^{||}r + jx^{||}i$.....6

$y^{||} = y^{||}r + jy^{||}i$.....7

Each and every real part and imaginary part of data is 16 bit IEEE 754 single precision floating point number.

$$x^{||} = x^{||}r + jx^{||}i$$

$x^{||} = (xr + yr) + j(xi + yi)$.....from 3 and 4

$$y^{||} = (x - y)W_N^k$$

Let $xr - yr = U1\ and\ xi - yi = U2$

Hence

$$y^{||} = (U1 + jU2)(Wr + jWi)$$

$$y^{||} = (U1Wr - U2Wi) + j(U1Wi + U2Wr)$$

$y^{||}r = U1Wr - U2Wi$.....8

$y^{||}i = U1Wi + U2Wr$.....9

Equation 8 and 9 are realized using DAA .A 16 bit number in single precision floating point is given by

$X = (-1)^S\ [2^E * 1.M]$

S=sign bit 1 bit

M=mantissa 10 bits

E=biased exponent 5 bits and its value is given by

$X = (-1)^S\ [2^E * \sum_{n=0}^{11} Mn2^{-n}]$ .....10

$y^{||}r = U1Wr - U2Wi$.....From 8

$y^{||}r = \{(-1)^{S1}\ [2^{E1} * \sum_{n=0}^{11} M1n2^{-n}]\}Wr - \{(-1)^{S2}\ [2^{E2} * \sum_{n=0}^{11} M2n2^{-n}]\}Wi$

Where M1 & M2 be the 11 bit mantissa of U1 & U2 along with hidden bit which is always 1and E1=E2=E therefore

$y^{||}r=2^E\{\{(-1)^{S1} * \sum_{n=0}^{11} M1n2^{-n}\}Wr - \{(-1)^{S2} * \sum_{n=0}^{11} M2n2^{-n}\}Wi\}$

Let sign bit is zero.S1=S2=0

$y^{||}r=2^E\{\{M102^0 + M112^{-1} + \cdots M1112^{-11}\}Wr - \{M202^{-0} + M212^{-1} + \cdots M2112^{-11}\}Wi\}$

$y^{||}r=2^E\{(M10Wr - M20Wi)2^0 + (M11Wr - M21Wi)2^{-1} + \cdots + (M111Wr - M211Wi)2^{-11}\}$...11

Similarly

$y^{||}i=2^E\{(M10Wi + M20Wr)2^0 + (M11Wi + M21Wr)2^{-1} + \cdots + (M111Wi + M211Wr)2^{-11}\}$.....12

The value inside the braces is calculated from the look up tables which is shown below.

**Table I.LUT for real number calculation (S1&S2 are negative).**

| M1n | M2n | LUT value |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | Wi |
| 1 | 0 | -Wr |
| 1 | 1 | -Wr+Wi |

The value o the LUT is changes according to changes in the sign bits. The real part is calculated by using the scaled accumulator which accumulates the values from the LUT for each bit of the mantissa. Similar method is used for the imaginary part calculations.

## IV. DA BASED BUTTERFLY ARCHITECTURE

The software implementation of Equation 11 & 12 is given below. The implementation is done by using Xilinx software and the coding is done in Verilog.

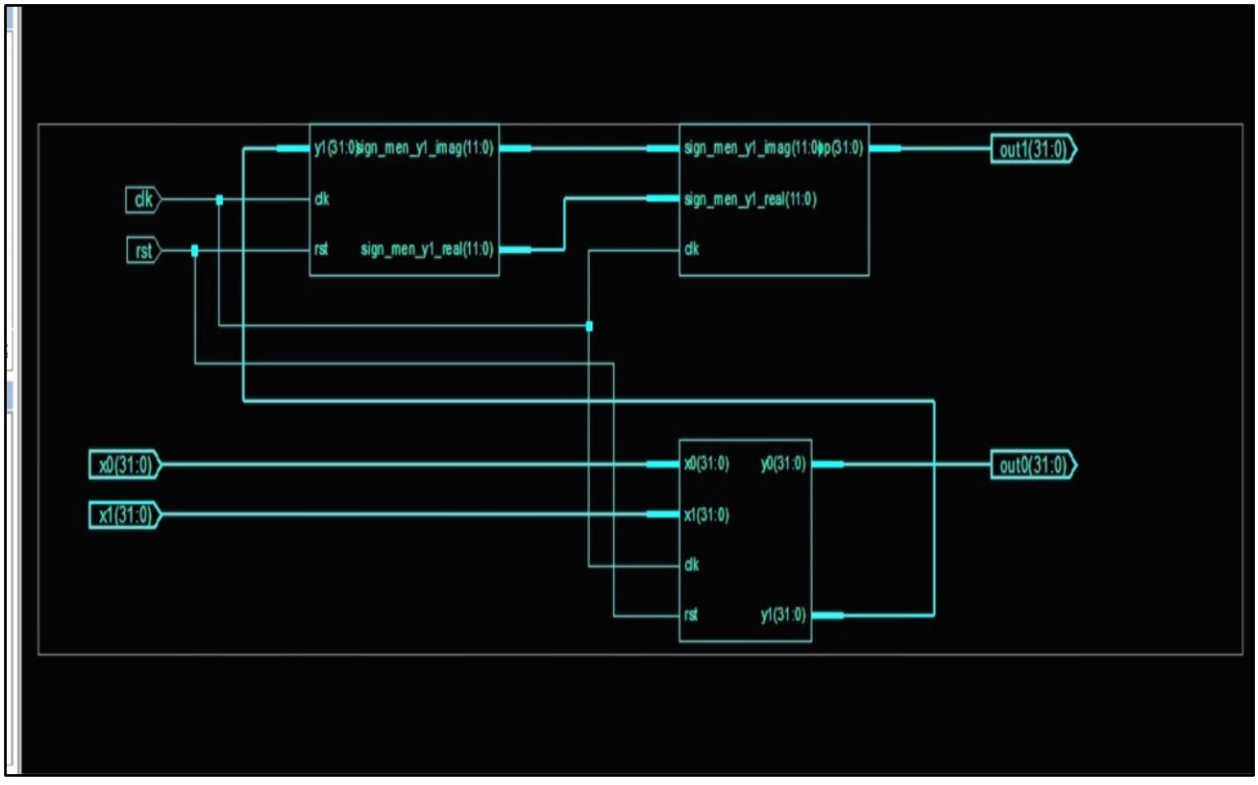
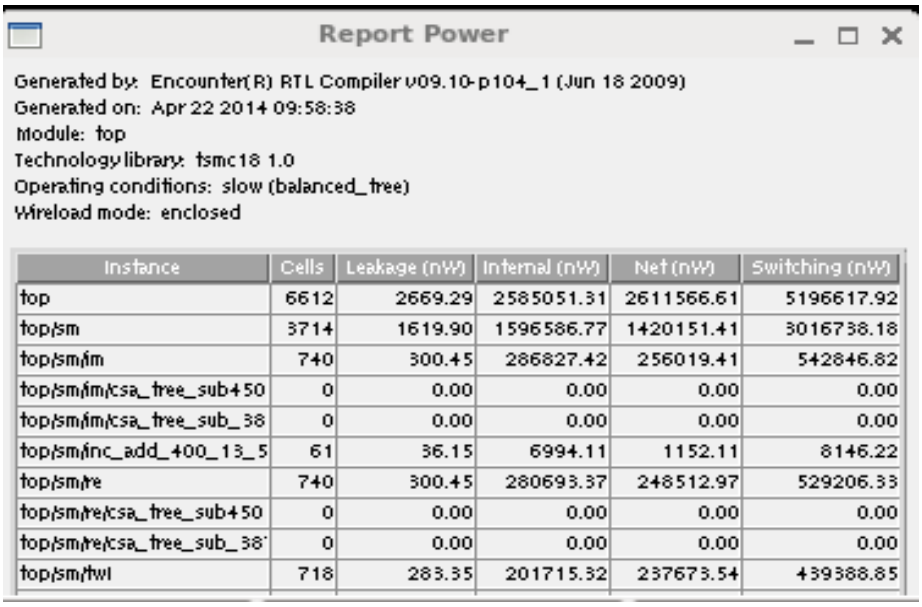



**Fig 4.Simulation of DA butterfly using Xilinx Software**

## V. RESULTS & DISCUSSIONS

DA based butterfly unit which as the only luring feature of tremendous power saving shows area power in Cadence RTL compiler and timing responses are observed in Modelsim.



Report Power

Generated by: Encounter(R) RTL Compiler v09.10-p104_1 (Jun 18 2009)
Generated on: Apr 22 2014 09:58:38
Module: top
Technology library: tsmc18 1.0
Operating conditions: slow (balanced_tree)
Wireload mode: enclosed

| Instance | Cells | Leakage (nW) | Internal (nW) | Net (nW) | Switching (nW) |
|---|---|---|---|---|---|
| top | 6612 | 2669.29 | 2585051.31 | 2611566.61 | 5196617.92 |
| top/sm | 3714 | 1619.90 | 1596586.77 | 1420151.41 | 3016738.18 |
| top/sm/m | 740 | 300.45 | 286827.42 | 256019.41 | 542846.82 |
| top/sm/m/csa_tree_sub450 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| top/sm/m/csa_tree_sub_38 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| top/sm/inc_add_400_13_5 | 61 | 36.15 | 6994.11 | 1152.11 | 8146.22 |
| top/sm/re | 740 | 300.45 | 280693.37 | 248512.97 | 529206.33 |
| top/sm/re/csa_tree_sub450 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| top/sm/re/csa_tree_sub_38 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| top/sm/twi | 718 | 283.35 | 201715.32 | 237673.54 | 439388.85 |

**Fig 5. Area and power reports from Cadence RTL compiler**

**Table II. Comparison Area and power reports from Cadence RTL compiler**

| | CONVENTIONAL | VEDIC | DA |
|---|---|---|---|
| POWER | 14.2 mW | 7.2 mW | 4.8 mW |
| CELLS | 16353 | 12612 | 6612 |
| AREA | 207.6 $mm^2$ | 223.92$mm^2$ | 102.42$mm^2$ |
| FREQUENCY | 88.33 MHz | 88.33 MHz | 88.33 MHz |

DA based butterfly unit which as the only luring feature of tremendous power saving shows area power and timing responses are observed. Referring to the contents of Table II, it is quite apparent that for same reference frequency of operation DA is proved to be more power efficient compared to Vedic based approach and conventionally used multipliers. Butterfly unit is operated at 88.33MHz.

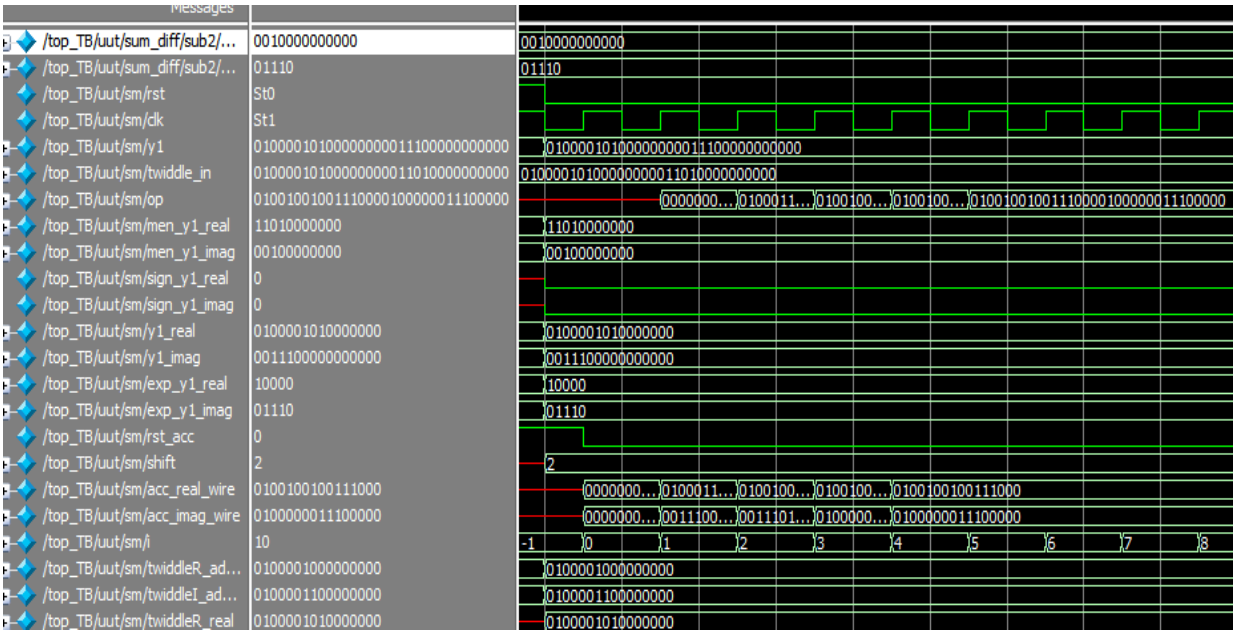



**Fig 6.Butterfly output and calculation based on bit combination of mantissa**

## V. CONCLUSION

Concluding the purpose of the thesis, Distributed arithmetic based multiplication is 28-30% power efficient than Vedic and compared to conventional multiplication it is beyond comparison level. Adders and shifter are the major part of the hardware and numbers of different cells used are comparatively less. Future scope of the work is to implement the whole architecture of FFT.

## REFERENCES

[1] Sang Yoon Park, "Efficient FPGA and ASIC realizations of DA –based Reconfigurable FIR Digital filter," IEEE 2014

[2] M.J. Wirthlin, "Constant Coefficient Multiplication Using Look-Up Tables," Journal of VLSI Signal Processing IEEE, vol. 36, pp. 7-15, 2004.

[3] "Distributed Arithmetic FIR Filter v9.0," Xilinx Product Specification IEEE 2004.

[4] S.J. Melnik off, S.F. Quigley, and M.J. Russell, "Implementing a Simple Continuous Speech Recognition System on an FPGA,"presented at International Symposium on Field-Programmable Custom Computing Machines (FCCM), IEEE 2002.

[5] A. Berkeman, V. Öwall and M. Torkelson, "A Low Logic Depth Complex Multiplier Using Distributed Arithmetic," IEEE Journal of Solid-state Circuits, Vol. 35, No.4, April 2000

[6] I. R. Mactaggart and M. A. Jack, "Radix-2 FFT butterfly processor using distributed arithmetic," Electronics letters 20th January 1983 IEEE vol. 19, December 1982

[7] .A. Das, A. Mankar, N. Prasad and K. Mahapatra "Efficient VLSI Architectures of Split-Radix FFT

[8] Using New Distributed Arithmetic," International Journal of Soft Computing and Engineering ISSN: 2231-2307, Volume-3, Issue-1, March 2013

[9] M.A. Sanchez, M. Garrido, M.L.Vallejo and C.L. Barrio, Automated Design Space Exploration of FPGA –based FFT architectures based on Area and Power Estimation, IEEE 2006

[10] C. R. Rupp, M. Landguth, T. Garverick, E. A. G. E. Gomersall, H. A. H. H. Holt, J. M. A. A. J. M. Arnold, and M. A. G. M. Gokhale, "The NAPA adaptive processing architecture," in FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on, 1998, pp. 28-37.

[11] S. A. White, "Applications of distributed arithmetic to digital signal processing: atutorial review," ASSP Magazine, IEEE [see also IEEE Signal Processing Magazine], vol. 6, pp. 4-19, 1989.