http://www.jctjournals.com

ISSN(ONLINE):2278 - 3814

Volume-12, Issue-07, July 2023

JOURNAL OF COMPUTING TECHNOLOGIES (JCT)

International Journal

Page Number: 01-07

# Floating-Point Butterfly Architecture Based On Binary Signed-Digit Representation

[1]Rahul Nath Tiwari, [2]Prof. Nishi Pandey, [3]Prof. Abhishek Agwekar

[1]M. Tech Scholar, [2]Asst.professor, [3]Head of department,

[1,2,3] TIEIT, Bhopal, (M.P.), INDIA

rahulnathtiwari007@gmail.com, nishi.pandey@trubainstitute.ac.in, abhiagwekar@gmail.com

***Abstract— This reveals the motivation to develop a high-speed FP butterfly design to mitigate FP slowness. This brief proposes a fast FP butterfly unit employing a devised FP fused dot product-add (FDPA) unit. During this planned 32 point FFT design. Floating-point arithmetic is enticing for the implementation for a spread of Digital Signal processing (DSP) applications as a result of it permits the designer and user to consider the algorithms and design without concern about numerical problems. In the past, many DSP applications used fixed point arithmetic due to the high cost (in delay, silicon area, and power consumption) of floating-point arithmetic units. The analysis results justify that the planned FP butterfly design is much faster than previous counterparts but at the value of additional area. Within the planned work design area is reduce than previous FFT design. The planned design of this analysis the logic size, area and power consumption using Xilinx 14.2.Fast Fourier transform (FFT) coprocessor, having a significant impact on the performance of communication systems, has been a hot topic of research for many years. The FFT function consists of consecutive multiply add operations over complex numbers, dubbed as butterfly units. Fast Fourier transform (FFT) is one amongst the most necessary tools in digital signal processing in addition as communication system as a result of transforming time domain to S-plane is very convenient using FFT. Applying floating-point(FP) arithmetic to FFT architectures, specifically butterfly units, has become additional common recently. However, the main drawback of FP butterfly is its slowness as compared with its fixed point counterpart.***

***Keywords— butterfly filter , floating-point(FP) arithmetic, API, Xilinx 14.2and FFT function .***

## I. INTRODUCTION

A floating-point number system is enticing for a variety of signal processing applications due to the wide dynamic range that provides freedom from scaling and overflows considerations that arise with fixed-point implementations. Among the varied floating-point number formats, IEEE- 754 single precision normal is used during this paper [1]. The only precision format is 32-bits consisting of a 1-bit sign, an 8-bit exponent, and a 23-bit mantissa. Additionally, there is one arithmetic operations.

Therefore, this paper examines a complex butterfly arithmetic operation within which every element (i.e., real and imaginary) of the data is represented by 32-bit single precision floating point numbers. However, floating-point arithmetic units have additional space, delay and power consumption than fixed point arithmetic units. A floating point number representation can simultaneously provide a large range of numbers and a high degree of precision. As a result, a portion of most microprocessors is often dedicated to hardware for floating point computation. Unlike fixed-point arithmetic, each computer company developed their own standards for the floating-point representation in electronic machines until the IEEE- 754 standard was introduced in 1985. Floating-point operations are widely used for advanced applications such as 3D graphics, signal processing, and scientific computations.

These require a wide dynamic range. Fixed-point arithmetic is not sufficient for this, but floating point arithmetic, such as that which is specified in IEEE-754 Standard for floating-point arithmetic, can represent a wide range of numbers from tiny fractional numbers to nearly infinitely huge numbers so that the overflow and underflow are avoided. This paper presents improved architecture designs and implementation details for a floating-point fused two-term dot product unit. The floating-point fused

 

dot product unit is useful for many digital signal processing (DSP) applications.

***Floating-Point Arithmetic***

Floating-point arithmetic is being used, and preferred over fixed-point, in many applications due to the fact that it provides a large range of numbers and a high degree of precision. It is also common to be used in a variety of Digital Signal Processing (DSP) applications because it relieves the designer of numerical issues e.g., scaling, overflow, and underflow. A floating-point number, as represented in Eqn. 1.1, consists of four components; namely, sign, significant, base and exponent.

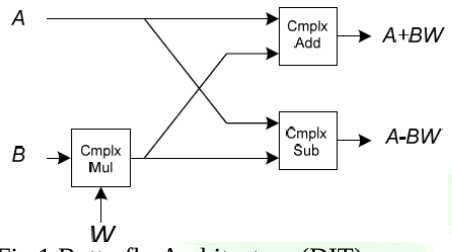


Fig.1 Butterfly Architecture (DIT)

There are two methods to implement a butterfly unit: 1) conventional 2) Golub's approach. Fig. .2 shows the implementation of a DIT butterfly with expanded complex numbers using the conventional approach. Accordingly, it consists of four multipliers and six adders/subtracters. Its hould be noted that, given the constant values of twiddle factors (W), the multipliers are constant and can be implemented via a series of shifters and adders in lieu of the multiplier tree.

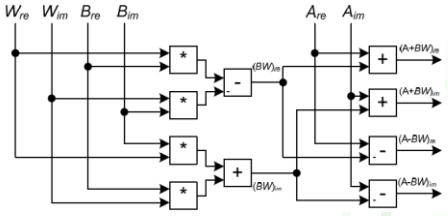


Fig.2 DIT Butterfly architecture using conventional approach

Fig. 3 shows the implementation of a DIT butterfly unit based on the Golub's approach. Accordingly, it consists of three multipliers and nine adders/subtracters.

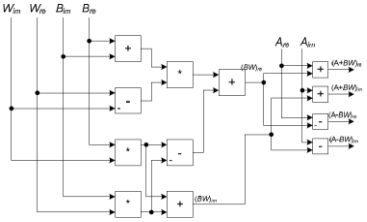

Fig.3 DIT Butterfly architecture using Golub's approach

## II. LITERATURE REVIEW

***Amir Kaivani et.al*** -Point Butterfly Architecture Based on Binary Signed-Digitin this paper planned a high-speed FP butterfly design that is faster than previous works however at the value of higher space. The reason for this speed improvement is twofold: 1) BSD illustration of the significant that eliminates carry-propagation and 2) the new FDPA unit planned during this brief. This unit combines multiplications and additions needed inFP butterfly; therefore higher speed is achieved by eliminating additional LZD, normalization, and rounding units. More analysis is also envisaged on applying dual-path FP design to the threeoperandFP adder and using different redundant FP representations. Moreover, use of improved techniques within the termination phase of the planning (i.e., redundant LZD, standardization, and rounding) would lead to faster architectures, although higher area prices are expected.

***Earl E. Swartzlander*** Jr –Point this paper describes the planning of 2 new fused floating-point arithmetic units andtheir application to the implementation of FFT butterfly operations. Though the fused add subtract unit is particular to FFT applications, the fused dot product is applicable to a wide style of signal process applications. Each the fused real unit and also the fused add-subtract unit are smaller than parallel implementations created with discrete floating-point adders and multipliers. The fused dot product is faster than the standard implementation, since rounding and normalization isn't needed as a part of every multiplication. Due to longer interconnections, the fused add-subtract unit is slightly slower than the discrete implementation. The area of the fused radix-2 butterfly is 35 % smaller and also the latency is 15 % less than the discrete radix-2 FFT butterfly parallel implementation. The area of the fused radix-4 butterfly is 26 % smaller and also the latency is 13 % less than the discrete radix-4 FFT butterfly parallel implementation. Each fused butterflies use fewer rounding operations resulting in a lot of accurate results than the discrete approaches. The errors for a 64K purpose FFT are about 25 % less for the fused implementations.

***Alexandre F. Tenca*** Multi-operand Floating-point Addition in this paper demonstrated the feasibleness of implementing multi- operand floating-point adders to induce additional accurate operations than equivalent networks of FPADD2s. The work was focused on 3-input FP adders however the discussion concerning design problems and various solutions is additionally applicable to adders for additional operands. The experimental results show that the 3-input FP adder design may be synthesized to reach shorter or similar delays than the network of 2-input FP adders, with comparable or better area, and additional accuracy. the capability of this component to get outputs honoring the commutative property for its inputs is very advantageous to eliminate the ordering drawback (non-associative behavior) imposed at the algorithmic rule level on networks of FPADD2s.

***Yao Tao*** Three-Operand Floating-Point Adder in this paper projected an improved design of 3- operand FP adder. The principle to choose the internal width is given and also the Realignment technique is used to avoid quite one sticky bit generated, which may reduce price and avoid

 

ISSN(ONLINE):2278 - 3814

the lost of accuracy. In addition, an OR-logic network rather than the comparer to observe the catastrophic cancellation is used within the improved one. Many sophisticated techniques, like compound adder and LZA, are used to optimize the design. The experiment shows that our design has a competitive area and delays by comparison with each a basic 3- operand FP adder and a network of 2-operand FP adders. The perform verification shows that our design has a similar accuracy as a full precision FP adder.

**A. Problem Formulation**

Fast Fourier transform (FFT) circuitry consists of several consecutive multipliers and adders over complex numbers; hence an appropriate number representation must be chosen wisely. Most of the FFT architectures have been using fixed-point arithmetic, until recently those FFTs based on floating-point (FP) operations grow. The main advantage of FP over fixed-point arithmetic is the wide dynamic range it introduces; but at the expense of higher cost. Moreover, use of IEEE-754-2008 standard for FP arithmetic allows for an FFT coprocessor in collaboration with general-purpose processors.

This offloads compute-intensive tasks from the processors and leads to higher performance. The main drawback of the FP operations is their slowness in comparison with the fixed-point counterparts. A way to speed up the FP arithmetic is to merge several operations in a single FP unit, and hence save delay, area, and power consumption. Using redundant number systems is another well-known way of overcoming FP slowness, where there is no word-wide carry propagation within the intermediate operations. The conversion, from non-redundant, to a redundant format is a carry-free operation; however, the reverse conversion requires carry propagation. This makes redundant representation more useful where many consecutive arithmetic operations are performed prior to the final result.

## III. METHODOLOGY

The FFT could be implemented in hardware based on an efficient algorithm in which the N-input FFT computation is simplified to the computation of two (N/2)-input FFT. We are also implementing proposed butterfly unit in 8 point FFT. The Continuing this decomposition leads to2-input FFT block, also known as butterfly unit. The proposed butterfly unit is actually a complex fused-multiply add with FP operands. Expanding the complex numbers, Fig. 4 shows the required modules.

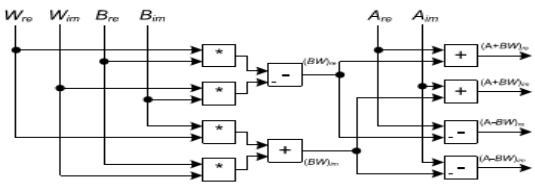


Fig.4 FFT butterfly architecture with expanded complex numbers

According to Fig.4 the constituent operations for butterfly unit are a dot-product (e.g., BreWim + BimWre) followed by an addition/subtraction which leads to the proposed FDPA operation (e.g.,BreWim +Bim Wre +Aim ). Realization information of FDPA, over FP operands, is discussed below. The exponents of all the inputs are assumed and represented in subtracting the bias), while the significant of Are, Aim, Bre, and Bim are represented in BSD. Within this representation every binary position takes values by one negative-weighted bit (negabit) and one positive-weighted bit (posibit). The carry-limited addition circuitry for BSD numbers is shown in Fig. 5, where capital (small) letters symbolizesnega bits (posibits). The critical path delay of this adder consists of three full-adders. The Projected FDPA consist of an unnecessary FP multiplier tag on by an unneeded FP three-operand adder.

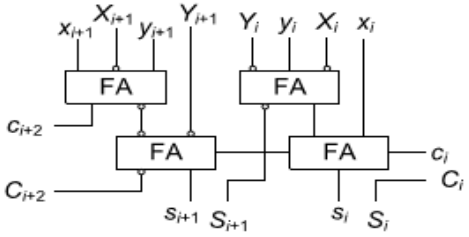


Fig.5 BSD adder (two-digit slice)

**A. Proposed Redundant Floating-Point Multiplier**

The proposed multiplier, likewise other parallel multipliers, consists of two major steps, namely, partial product generation (PPG) and PP reduction (PPR). Though, different to the traditional multipliers, our multiplier feature remains the product in unnecessary format and therefore ether's no need for the final carry-propagating adder. The exponents of the input operands are taken care of within the same means as is completed in the standard FP multipliers; Though, normalization and rounding error are left to be exhausted subsequent block of the butterfly architecture (i.e., three-operand adder).

## IV. IMPLEMENTATION

***A. Computer Arithmetic Overview***

Computer arithmetic is concerned with the hardware realization of mathematical formulas, algorithms, and complex models from a theoretical world. Hardware functions calculate -point and scientific notations (floating-point).

Fixed-Point Representation Overview and Implementation Issues In computing, a fixed-point number representation is a real data type for a number that has a fixed number of digits after (and sometimes before) the radix point. Fixed-point number representations are much less complicated (and less computationally demanding) than floating point number representations [6]. Fixed-point numbers are useful for representing fractional values, usually in base 2, when the executing processor has no floating point unit (FPU) or if fixed-point provides improved performance or accuracy for the application at hand [7]. A fixed-point number may be written as I.F, where I represents the integer part, '.' is

 

ISSN(ONLINE):2278 - 3814

the radix point, and F represents the fractional part. In binary fixed-point numbers, each magnitude bit represents a power of two, while each fractional bit represents an inverse power of two.

**B. Fixed-Point Precision Loss and Overflow**

Information may be lost in fixed point operations when they produce results that have more bits than the operands [8]. For instance, the result of fixed point multiplication could potentially have as many bits as the sum of the number of bits in the two operands. In order to fit the result into the same number of bits as the operands, the answer must be rounded or truncated [9]. If this is the case, the choice of which bits to keep is very important. For instance when multiplying two fixed point numbers with the same format, with I integer bits, and F fractional bits, the answer Could have up to 2*I integer bits, and 2*F fractional bits [9].

Most fixed-point multiplication procedures use the same result format as the operands. This has the effect of keeping the middle bits; the I least significant integer bits, and the F most significant fractional bits. Fractional bits below this value represent a relatively minor precision loss. If any integer bits are lost, however, the value will be radically inaccurate. This is considered to be an overflow, and needs to be avoided in embedded calculations [10].

An Overview of the Floating-Point Fused Multiply-Add (FMA) Operation In 1990, IBM introduced the floating-point fused multiply-add operation on the RISC System 6000 (IBM RS/6000) chip [3], [4]. IBM recognized that several advanced applications, specifically those with dot products, are routinely performed with a floating point multiplication, A x B, immediately followed by a floating-point addition, (A x B) result + C, ad infinitum. To increase the performance of these applications, a new unit was created that merged a discrete floating-point multiplier and floating-point adder into a single hardware block the floating-point fused multiply-add unit. This floating-point arithmetic unit, shown in Figure 4.2, executes the equation (A x B) + C in a single instruction. With the continued demand for 3D graphics, multimedia applications, and new advanced processing algorithms, the IEEE has included the fused multiply-add operation into the 754-2008 standard [2]. Even though the fused multiply-add architecture has troublesome latencies, high power consumption, and performance degradation with single-instruction execution, more and more microprocessor designs implement floating point fused multiply-add units in their silicon.

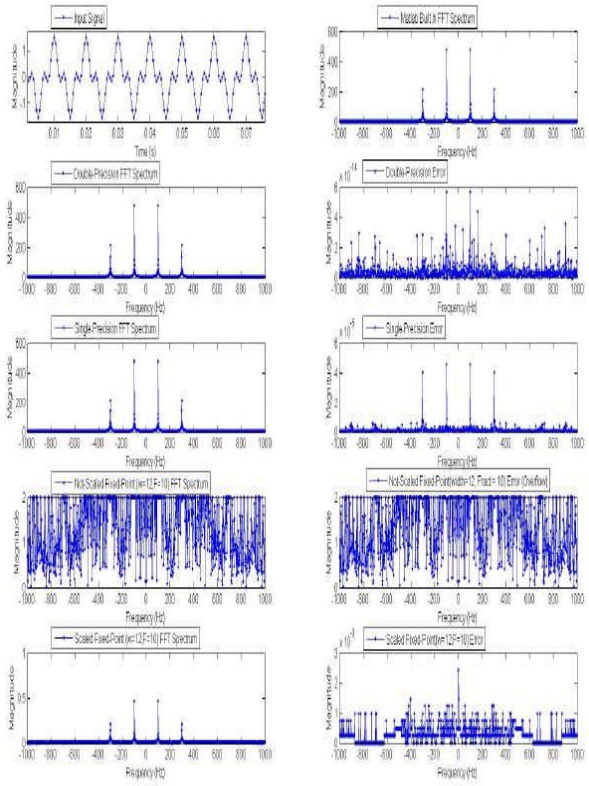

Fig. 6 FFT Spectrum Calculation Using: Double Precision Floating-Point, Single PrecisionFloating-Point and 12-bit Fixed-Point Without and With Scaling

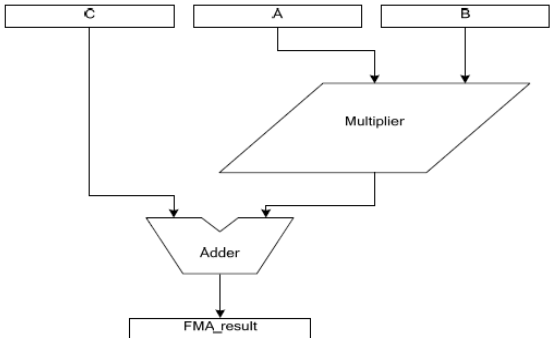


Fig.7Block Diagram of a Floating-point Fused Multiply-add Unit, reduced from

## V. SIMULATION RESULTS

***A. Simulation Results***

The Design the 8 point FFT using the proposed FP-Butterfly unit implemented in Modelsim andXilinx ISE simulation result introduced in this chapter. In order to evaluate the designs the area,delay and power consumption are estimated using the simulation.

Fig.7 shows start the Xilinx ISE Project Navigator. Choose File New Project. A popupdialog box will appear. Enter tutor1 for Project Name. For the Project Location, select thedirectory where the project will be stored for your project. Click Next to move to the deviceproperties page. Fill in the properties. Click Next to proceed to the Create New Source windowin the New Project Wizard.

 

ISSN(ONLINE):2278 - 3814

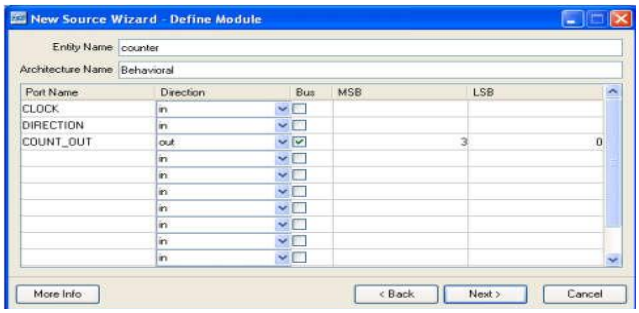


Fig.8 Define Module

Fig.8 Click the New Source button in the New Project Wizard. Select VHDL Module as the source type. Type in the file name counter. Verify that the Add to project checkbox is selected. Click Next. Declare the ports for the counter design by filling in the port information as shown infig.5.2. Click next, and then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.

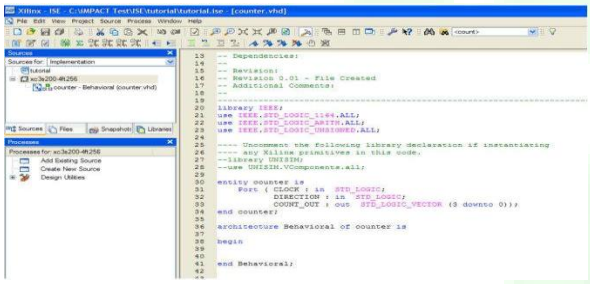

Fig.9 New Project in ISE

Fig.9 The Xilinx Project Navigator is the heart of the Xilinx ISE software. Editing, compiling, and programming can all be accomplished through the Project Navigator. This screen shot shows the default layout which includes the Module View, Process View, Console, and Editing/Viewing windows. The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown in fig.9.

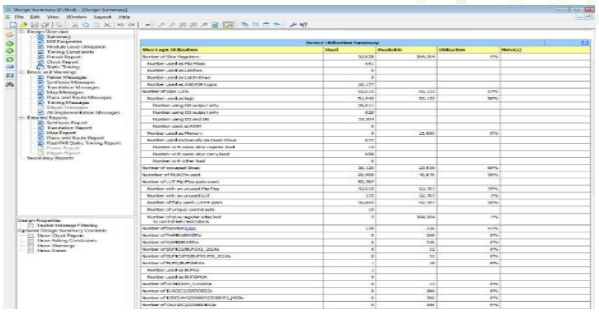

Fig.10 Synthesis report of proposed system

Fig.10 depicts the synthesis report of proposed system. In this report device utilization summary are given. In this discuss how many devices are used and available and how many percentages utilization are given. In this proposed system used are number of slice register, Look-up-tables, number of AND/OR logic gates, number of filp-flop.

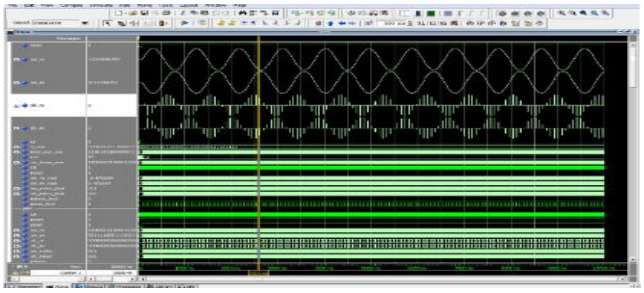

Fig.11 Simulation output of proposed system

Fig.11 shows the simulation output of entire proposed system. This is a simulation output of FFT butterfly architecture with expanded complex numbers.

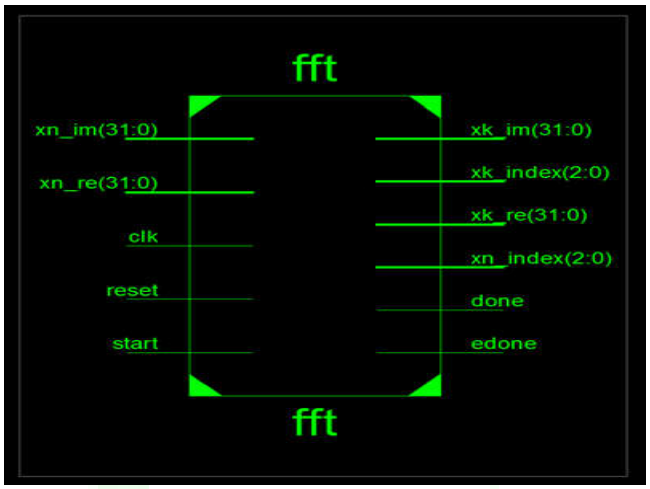


Fig.12 RTL view of top module

Fig.12 demonstrates the RTL view of top module that is FFT. After the HDL synthesis phase of the synthesis process, use the RTL Viewer to view a schematic representation of the preoptimized design in terms of generic symbols that are independent of the targeted Xilinx device, for example, in terms of adders, multipliers, counters, AND gates, and OR gates.

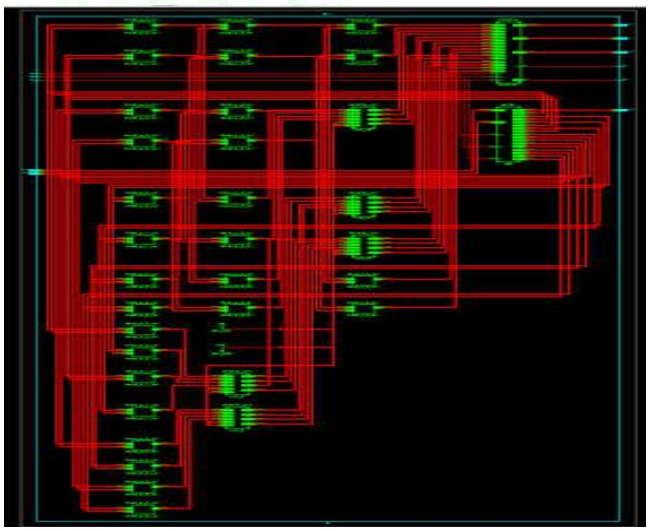

Fig.13 overall design

ISSN(ONLINE):2278 - 3814

Fig.13 shows the overall design of proposed system. The overall design of proposed system shows combination of all small entity used in formation of modified work, which gives optimized and efficient output in comparison of previous work. The area and power consumption of proposed work gives efficient output in comparison of work done before present time.

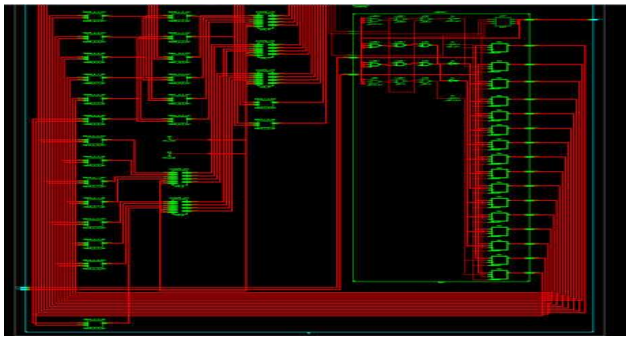

Fig.14 splitter

Fig.14 demonstrates the splitter. There are many devices used for design splitter. Splitter should be used for the divide/split 8 point, 16 point and 32 point fast Fourier transform circuit into two or multiple small same bit like if we split 8 bit FFT into two different form splitter should be implemented in two 4 bit FFT. Similarly if 16 point FFT split into four different small size bit it should be taken as four 4 bit FFT.

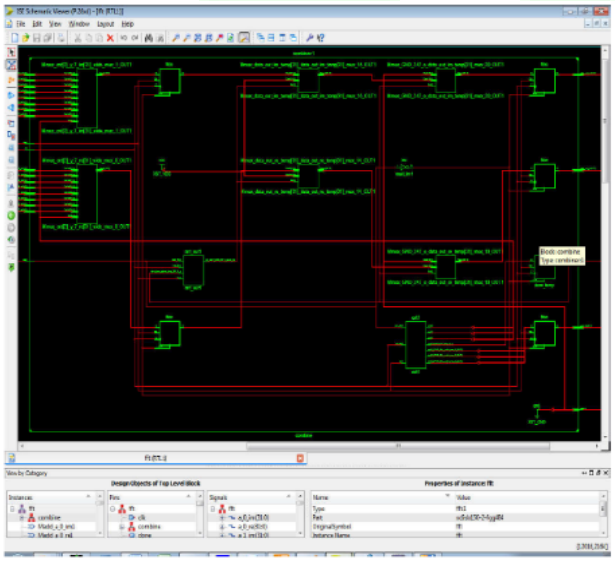

Fig.15 Combiner

Fig.15 elaborates the combiner. Combiner should be used for the combination of two small size bit architecture into a big architecture for example if we combined to 8 bit architecture then output of combiner should be 16 bit architecture. The application of combiner is mostly in streaming videos in which number of frame architecture should be combined to give better and efficient result.

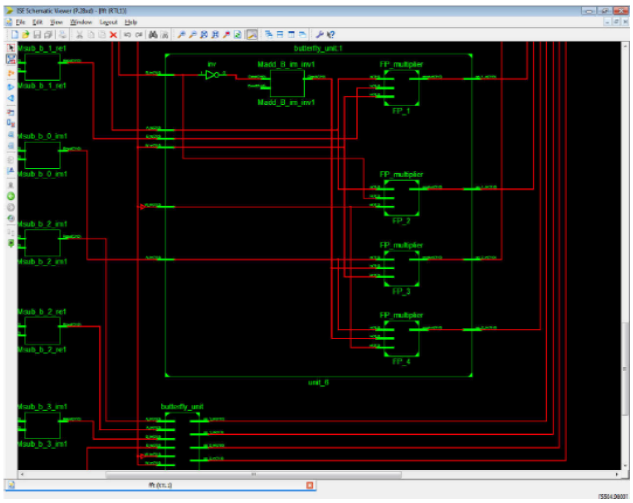

Fig.16 butterfly unit

Fig.16 shows butterfly unit. In this there are many adder, subtracted and floating point multiplier are used. The fast Fourier transform algorithms, a butterfly is a portion of the Computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa (breaking a larger DFT up into sub transforms). The name "butterfly" comes from the shape of the data-flow diagram in the radix-2 case.

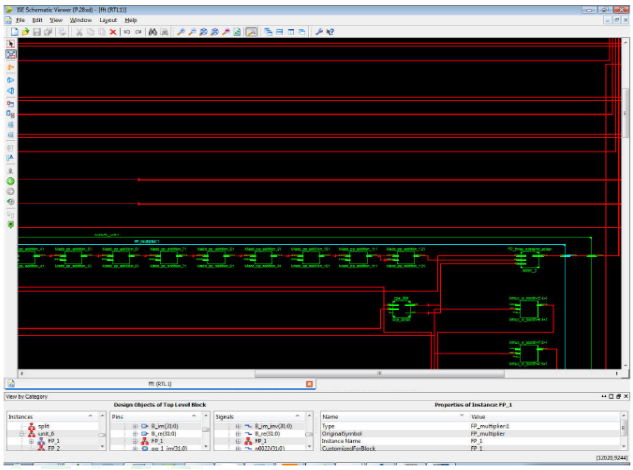

Fig.16 FP multiplier

Fig.16 elaborates the FP multiplier. In this there are many adders; FP three operand adders and booth multiplier are used. This multiplier is mainly used to multiply two floating point numbers. Separate algorithm is essential for multiplication of these numbers. Here multiplication operations simple than addition this is especially true if it are using a 32-bit format. Table 5.1 shows the result summary of proposed research work. The proposed work is design 8point FFT. In this Power consumption (W) is 1.068; Delay (ns) is 107.053, Area is LUT 67255,Slice register 651, IOB 139.

## VI. CONCLUSION AND FUTURE WORK

In this research work proposed an improved architecture of a high-speed FP butterfly, which is faster

than previous works but at the cost of higher area. In this proposed work proposes a fast FP butterfly unit using a devised FP fused-dot product-add (FDPA) unit. In this research is applying dual-path FP architecture to the three-operand FP adder and using other redundant FP representations. Moreover, use of improved techniques in the termination phase of the design (i.e., redundant LZD, normalization, and rounding) would lead to faster architectures, though higher area costs are expected. In this proposed work Area is reduced. In the proposed work the design of two new fused floating-point arithmetic units and their application to the implementation of FFT butterfly operations. In this research work 32 point FFT architecture is used. In this Combiner and splitter are used. In this proposed work floating point arithmetic is used. In this proposed work area is reduced than previous research work and power consumption is less. This proposed work describes two fused floating-point operations and applies them to the implementation of fast Fourier transform (FFT) processors.

In the future work the design stage of the abolition of the use of improved techniques (ie, repeating LZD, normalize, and rounding) of the estimated costs, however, led to faster architectures. The fusing concept could be extended to other types of computation extensive applications and might result in delay, area and power consumption reduction. As a future work, one could modify the proposed design to include a dual-path FP architecture which would be expected to have lower latency but at the cost of more area. Further improvement is achieved by the dual-path algorithm. The dual-path floating-point fused dot product unit consists of a far path and a close path and one path is selected based on the exponent difference.

**References**

[1] Kaivani, Amir, and SeokbumKo. "Floating-point butterfly architecture based on binary signed-digit representation." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24.3 (2016): 1208-1211.

[2] Swartzlander, Earl E., and Hani HM Saleh. "FFT implementation with fused floating- point operations." IEEE transactions on computers 61.2 (2012): 284-288.

[3] Tenca, Alexandre F. "Multi-operand floating-point addition." Computer Arithmetic, 2009. ARITH 2009. 19th IEEE Symposium on. IEEE, 2009.

[4] Tao, Yao, et al. "Three-operand floating-point adder." Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on. IEEE, 2012.

[5] Nielsen, AsgerMunk, et al. "An IEEE compliant floating-point adder that conforms with the pipeline packet-forwarding paradigm." IEEE Transactions on Computers 49.1 (2000): 33-47.

[6] Min, Jae Hong, Seong-Wan Kim, and Earl E. Swartzlander. "A floating-point fused FFT butterfly arithmetic unit with merged multiple-constant multipliers." Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on. IEEE, 2011.

[7] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.

[8] Kaivani, Amir, and Seok-Bum Ko. "Area efficient floating-point FFT butterfly architectures based on multi-operand adders." Electronics Letters 51, no. 12 (2015): 895- 897.

[9] BehroozPrahami, Computer Arithmetic Algorithms and Hardware Designs, New York: Oxford University Press, 1999.

[10] A. V. Oppenheim, R.W. Schafer, and J. R. Buck, Discrete-Time Signal Processing, Second Edition, Upper Saddle River: Prentice Hall, 1999.

 