IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 3, Issue 3 (Sep. – Oct. 2013), PP 13-18
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Implementation of Floating Point FFT Processor using VHDL

Selvarani[1], Selvaraju[2], Vijayraj[3], Jayaprakash[4], Baskar[5]
[1]*(Department of ECE, Muthayammal College of Engineering /Anna University, India)*
[2]*(Department of ECE, Muthayammal College of Engineering /Anna University, India)*
[3]*(Department of ECE, Muthayammal College of Engineering /Anna University, India)*
[4]*(Department of ECE, Muthayammal College of Engineering /Anna University, India)*
[5]*(Department of ECE, Muthayammal College of Engineering /Anna University, India)*

Abstract: *The Fast Fourier Transform (FFT) is a capable algorithm to compute the Discrete Fourier Transform (DFT) and it's inverse. It has a number of applications in the field of signal processing. The usual butterfly FFT design requires needless computations and data storage which lead to unnecessary power consumption. Use of the IEEE-754 standard 32-bit floating-point format also facilitates using the Fast Fourier Transform (FFT) processors. This paper describes two fused floating-point operations and applies them to the implementation of Fast Fourier Transform (FFT) processors using VHDL. The fused operations are a two-term dot product and add-subtract unit. The FFT processors use "butterfly" operations that consist of multiplications, additions, and subtractions of complex valued data. The statistical results of the fused implementations are slightly more accurate.*

Keywords: *Fast Fourier Transform (FFT), Floating-Point, Radix-2 Butterfly, VHDL.*

I. INTRODUCTION

In computing, floating point describes a method of representing an approximation to real numbers in a way that can support a wide range of values. Numbers are, in general, represented approximately to a fixed number of significant digits and scaled using an exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form

$$\text{Significant digits x base}^{\text{exponent}}$$

The term floating point refers to the fact that the radix point (decimal point, or, more commonly in computers, binary point) can "float" that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation. Over the years, a variety of floating point representations has been used in computers. However, since the 1990s, the most commonly encountered representation is that defined by the IEEE 754 Standard. Use of the IEEE-754 standard 32-bit floating-point format [2] also facilitates using the Fast Fourier Transform (FFT) processors as coprocessors in collaboration with general purpose processors. This paper is concerned with the efficient floating point implementation of the butterfly units that perform the computations in FFT processors. Since many signal processing applications need high throughput more than low latency, the primary focus of the fused elements is to reduce the circuit area. Section 2 describes IEEE-754 Standard. Section 3 describes Fast Fourier Transform. The next two sections describe the Fused DP and Fused AS units. Subsequent sections describe the use of these two operations to implement FFT butterfly units.

II. IEEE-754 STANDARD

The IEEE Standard for Floating Point Arithmetic (IEEE 754) [2] is a technical standard for floating point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). Many hardware floating point units use the IEEE 754 standard. The current version, IEEE 754-2008 published in August 2008, includes nearly all of the original IEEE 754 1985 standard and the IEEE Standard for Radix-Independent Floating-Point Arithmetic (IEEE 854- 1987).

The advantage of floating-point representation over fixed point and integer representation is that it can support a much wider range of values. For example, a fixed point representation that has seven decimal digits with two decimal places can represent the numbers 12345.678, 123.45, 1.23456 and soon, whereas a floating-point representation (such as the IEEE 754 decimal 32 format) with seven decimal digits could in addition represent 1.234567, 0.00001234567, 1234567000, and so on. The floating-point format needs slightly more storage (to encode the position of the radix point), so when stored in the same space, floating-point numbers achieve their greater range at the expense of precision.

IEEE floating point numbers have three basic components: the sign, the exponent, and the mantissa. The mantissa is composed of the fraction and an implicit leading digit. The exponent base (2) is implicit and need not be stored. The following Table 1 shows the layout for single (32-bit) and doubles (64-bit) precision floating-point values. The number of bits for each field are shown (bit ranges are in square brackets).

1. Sign Bit

The sign bit is as simple as it gets. 0 denotes a positive number, 1 denotes a negative number. Flipping the value of this bit flips the sign of the number.

2. Exponent

The exponent field needs to represent both positive and negative exponents. To do this, a bias is added to the actual exponent in order to get the stored exponent. For IEEE single precision floats, this value is 127. Thus an exponent of zero means that 127 is stored in the exponent field. For double precision, the exponent field is 11 bits, and has a bias of 1023.

3. Mantissa

The mantissa, also known as the significant, represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

TABLE I

Precision	Sign	Exponent	Fraction	Bias
Single Precision	1[32]	8 [30-23]	23 [22-00]	127
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

III. Fast Fourier Transform

The number of complex multiplication and addition operations required by the simple forms both the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) is of order N^2 as there are N data points to calculate, each of which requires N complex arithmetic operations.

For length n input vector x, the DFT is a length n vector X, with n elements:

$$\sum_{n=0}^{N-1} x(n)e^{-j2\frac{\pi}{N}nk} k=0, 1, 2, ..., N-1 \qquad (1)$$

As the name suggests, FFTs are algorithms for quick calculation of Discrete Fourier Transform of a data vector. The FFT is a DFT algorithm which reduces the number of computations needed for N points from O (N^2) to O (N log N) where log is the base-2 logarithm.

The Radix 2 algorithms are useful if N is a regular power of 2 (N=2^p). There are two different Radix 2 algorithms, the so called Decimation in Time (DIT) and Decimation in Frequency (DIF) algorithms. Decimation in Time (DIT) computational elements consist of complex multiplications followed by a sum and difference network. Decimation in Frequency (DIF) computational elements consist of a sum and difference network followed by complex multiplications.

IV. Fused Dot Product Unit

In many DSP algorithms and in other fields calculating the sum of the products of two sets of operands (dot-product) is a frequently used operation. It has several advantages over discrete floating-point adders and multipliers in a floating point unit design. Not only can a fused multiply add unit reduces the latency of an application that executes a multiplication followed by an addition, but the unit may entirely replace a processor's floating-point adder and floating-point multiplier. In traditional floating-point hardware the dot product is performed with two multiplications and an addition. These operations may be performed in a serial fashion which limits the throughput. The multiplications may be performed in parallel with two independent floating-point multipliers followed by a floating-point adder which is expensive (in silicon area and in power consumption). The implementation of a floating point parallel dot-product unit is shown in Fig.1.

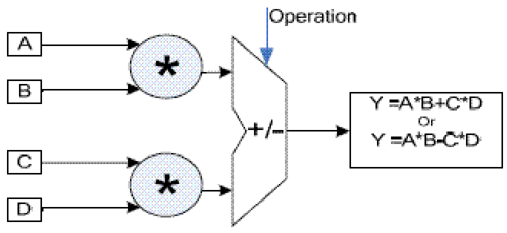


Fig.1. Parallel Dot Product Unit

Alternatively, in fused floating point unit two floating point multipliers, adder and subtraction unit which reduce the silicon area and power consumption. The implementation of a floating point fused dot-product unit shown in Fig.2.

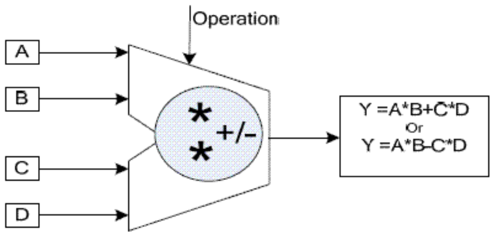


Fig.2. Fused Dot Product Unit

The floating-point two-term fused dot product (Fused DP) unit computes a two term dot product:

$$X= AB \pm CD \qquad (2)$$

Although a conventional dot product adds the two products as shown in (2), the Fused DP unit [4] also allows forming the difference of the two products, which is useful in implementing complex multiplication. There is a significant area reduction compared to a conventional parallel discrete implementation of two multipliers and an adder, since the rounding and normalization logic of both of the multipliers are eliminated.

V. FUSED ADD-SUBTRACT UNIT

The conventional parallel Add-Subtract unit as shown in fig.3, which has separate add and subtract unit in parallel. This may increase the silicon area and consumes more power.

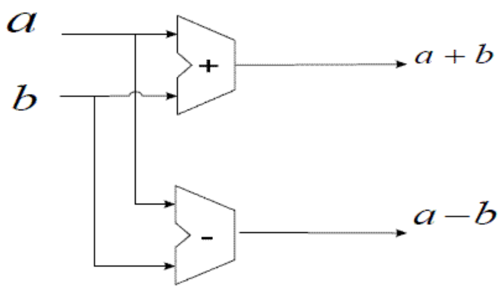


Fig.3. Parallel Add-Subtract unit

Alternatively, in fused floating point Add-Subtract unit [6] is used which reduce the silicon area and power consumption. It combines both addition and subtraction operations as a single unit to perform complex addition and subtraction operation. The implementation of a floating point fused Add-Subtract unit shown in Fig.4.

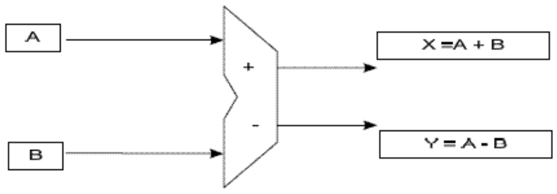


Fig.4. Fused Add-Subtract Unit

The floating-point fused add-subtract unit (Fused AS) performs an addition and a subtraction in parallel on the same pair of data as shown as:

$$X = A \pm B \qquad (3)$$

The both floating point fused dot product unit and fused add-subtract unit are designed based on behavioral modeling using VHDL.

VI. RADIX-2 FFT

To demonstrate the utility of the Fused DP and Fused AS units for FFT implementation, FFT butterfly unit designs using both the discrete and the fused units have been made. First radix-2 decimation in time FFT butterfly [3] was designed. It is shown in Fig. 5. All lines carry complex pairs of 32-bit IEEE-754 numbers and all operations are complex. The complex add, subtract, and multiply operations can be realized with a discrete implementation that uses two real adders to perform the complex add or subtract and four real multipliers and two real adders to perform the complex multiply. The complete butterfly consists of six real adders and four real multipliers. In this all lines are 32-bits wide for the IEEE-754 single-precision data. Alternatively, the complex add and subtract can be performed with two fused add subtract units and the complex multiplication can be realized with two fused dot product units.

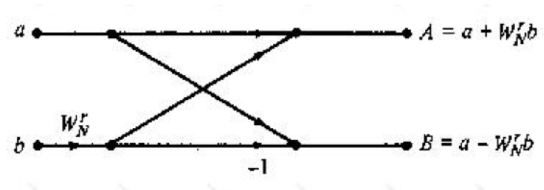


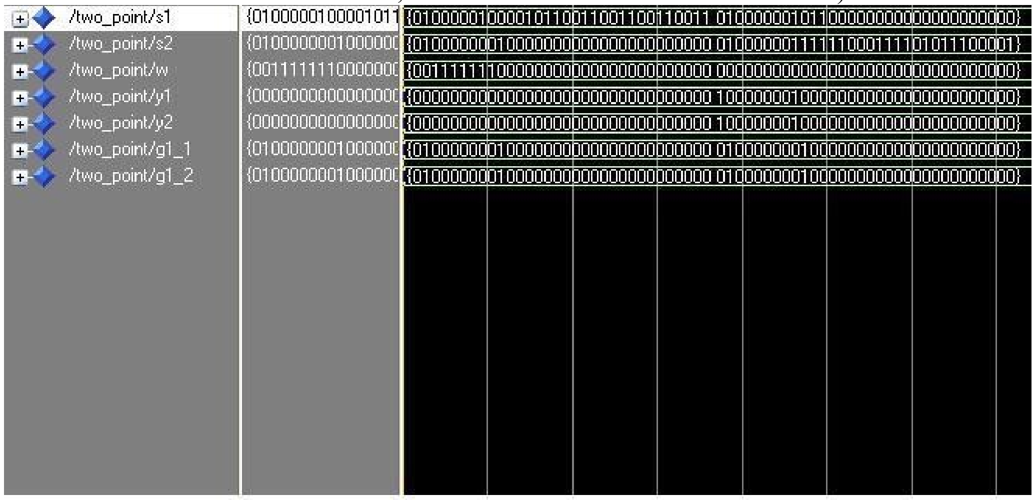
Fig.5. Butterfly for radix-2 DIT FFT

In comparing the discrete and fused radix-2 butterfly units, the fused version requires about one-third less area and is about 15 percent faster than the discrete implementation. The area saving is the most significant, since in most signal processing applications.

VII. SIMULATION RESULT

1. 2-Point Fused FFT (Single Precision)

X_0= (01000001000010110011001100110011, 01000001011000000000000000000000)
X_1= (01000000010000000000000000000000, 01000000111111000111101011100001)
Y_0= (00000000000000000000000000000000, 10000000100000000000000000000000)
Y_1= (00000000000000000000000000000000, 10000000100000000000000000000000)



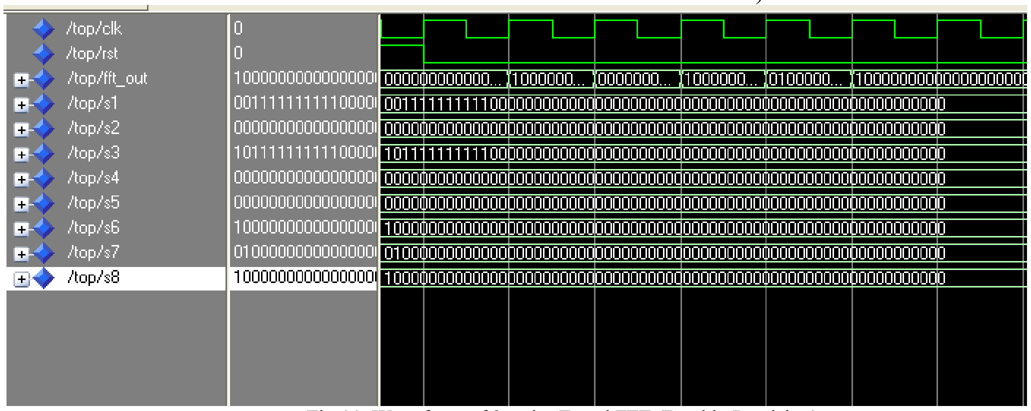
```
/two_point/s1     {010000010000101 1  {01000001000010110011001100110011 01000001011000000000000000000000}
/two_point/s2     {010000000100000C   {01000000010000000000000000000000 01000000111111000111101011100001}
/two_point/w      {001111111000000C   {00111111100000000000000000000000 00000000000000000000000000000000}
/two_point/y1     {000000000000000C   {00000000000000000000000000000000 10000000100000000000000000000000}
/two_point/y2     {000000000000000C   {00000000000000000000000000000000 10000000100000000000000000000000}
/two_point/g1_1   {010000000100000C   {01000000010000000000000000000000 01000000010000000000000000000000}
/two_point/g1_2   {010000000100000C   {01000000010000000000000000000000 01000000010000000000000000000000}
```

Fig.10. Waveform of 2-point Fused FFT (Single Precision)

2. 2-Point Fused FFT (Double Precision)

X0= (0011111111000000000000000000000000000000000000000000000000000000, 0000000000000000000000000000000000000000000000000000000000000000)

X1= (1011111111000000000000000000000000000000000000000000000000000000,
0000000000000000000000000000000000000000000000000000000000000000)
Y0= (0000000000000000000000000000000000000000000000000000000000000000,
1000000000000000000000000000000000000000000000000000000000000000)
Y1= (0100000000000000000000000000000000000000000000000000000000000000,
1000000000000000000000000000000000000000000000000000000000000000)



```
/top/clk      0
/top/rst      0
/top/fft_out  1000000000000000   000000000000... 1000000... 0000000... 1000000... 0100000... 1000000000000000000
/top/s1       0011111111110000
/top/s2       0000000000000000
/top/s3       1011111111110000
/top/s4       0000000000000000
/top/s5       0000000000000000
/top/s6       1000000000000000
/top/s7       0100000000000000
/top/s8       1000000000000000
```

Fig.11. Waveform of 2-point Fused FFT (Double Precision)

3. FPGA Implementation

3.1 Spartan-6 FPGA

Unified Learning Kit is based on Texas Instruments OMAP3530 application processor & Spartan-6 FPGA. The Spartan-6 family provides leading system integration capabilities with the lowest total cost for high-volume applications. The thirteen-member family delivers expanded densities ranging from 3,840 to 147,443 logic cells, with half the power consumption of previous Spartan families, and faster, more comprehensive connectivity. Spartan-6 FPGAs are the programmable silicon foundation for Targeted Design Platforms that deliver integrated software and hardware components that enable designers to focus on innovation as soon as their development cycle begins.
Supporting Appliances : The Spartan-6 FPGA supports interfaces such as Ethernet, FPGA HDR2 20-pin Header, Keypad connector(4X4), FPGA Expansion connector, Mictor connector, ADC, DAC, LED, UART Transceiver, 7 segment LED, 16x2 Char LCD, Oscillators (10 & 100Mhz), DDR2 SDRAM, PROM, Dip Switches.

Fig.11. Unified Learning Kit

3.2 70-Pin Expansion connector

Most of FPGA signals are directly driven by FPGA which are 3.3V voltage level. For some signals, level translators are used to convert the 1.8V FPGA signals to 3.3V compatible signals.
Using 70-Pin Expansion connector leds are interfaced with FPGA to display the outputs of 32 bit FFT Processor is shown in figure 12.

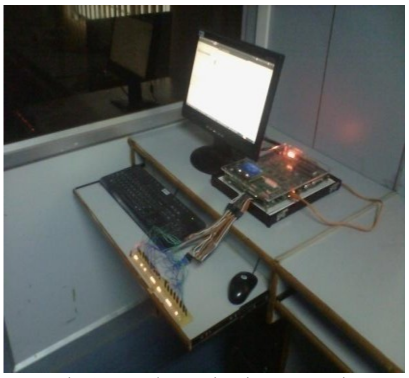


Fig.12. Implementing in UTLP Kit

4. Comparison

4.1 32-bit FFT

Architecture / Parameter	Parallel	Fused
Area	81,029 Gates	60,094 Gates
Power	42mW	37mW
Delay	8.442ns	7.210ns

Fig.13. 32-bit FFT Comparison

4.2 64-bit FFT

Architecture / Parameter	Parallel	Fused
Area	139,123 Gates	93,624 Gates
Power	43mW	34mW
Delay	8.833ns	7.713ns

Fig.14. 64-bit FFT Comparison

VIII. CONCLUSION AND FUTURE WORKS

This paper describes the design of two new fused floating-point arithmetic units and their application to the implementation of FFT butterfly operations. Although the fused Add-Subtract unit is specific to FFT applications, the fused dot product is applicable to a wide variety of signal processing applications. Both the fused dot product unit and the fused add subtract unit are smaller than parallel implementations constructed with discrete floating-point adders and multipliers. Future work will be design and implementation of Double Precision Floating point in FFT application.

REFERENCES

[1] Earl E. Swartzlander Jr., and Hani H.M. Saleh, "FFT Implementation with Fused Floating-Point Operations," in IEEE Transactions on Computers, 2012.

[2] IEEE Standard for Floating-Point Arithmetic, ANSI/IEEE Standard 754-2008, Aug. 2008.

[3] Sneha N.kherde and Meghana Hasamnis, "Efficient Design and Implementation of FFT," in International Journal of Engineering Science and Technology, 2011.

[4] H.H. Saleh and E.E. Swartzlander, Jr., "A Floating-Point Fused Dot-Product Unit," Proc. IEEE Int'l Conf. Computer Design (ICCD), 2008.

[5] H.H. Saleh, "Fused Floating-Point Arithmetic for DSP," PhD dissertation, Univ. of Texas, 2008.

[6] H. Saleh and E.E. Swartzlander, Jr., "A Floating-Point Fused Add-Subtract Unit," Proc. IEEE Midwest Symp. Circuits and Systems (MWSCAS), pp. 519- 522, 2008.