ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

Verilog Implementation of Floating Point FFT With Reduced Addressing Logic

D. Venkatesh Babu[1], K. Naresh Kumar[2]

M.Tech, Embedded Systems, VLSI Design, G.V.P.C.O.E, Visakhapatnam, Andhra Pradesh, India[1]

Assistant Professor, ECE Dept, G .V.P.C.O.E, Visakhapatnam, Andhra Pradesh, India[2]

Abstract: The Discrete Fourier Transform (DFT) can be implemented very fast using Fast Fourier Transform (FFT). It is one of the finest operations in the area of digital signal and image processing. FFT is a luxurious operation in terms of MAC. To achieve FFT calculation with a many points and with maximum number of samples the MACs requirement could not be matched by efficient hardware's like DSP. So a fine solution is to use dedicated hardware processor to perform efficient FFT working out at high sample rate, while the DSP could perform the less concentrated parts of the processing. Verilog implementation of floating point FFT with reduced generation logic is the proposed architecture, where the two inputs and two outputs of any butterfly can be exchanged hence all data and addresses in FFT dispensation can be reordered.

Keywords: FFT, MAC, butterfly exchanging circuit, FPGA, DSP's.

I. INTRODUCTION

There are different ways to compute the Discrete Fourier Transform (DFT), firstly by solving in simultaneous linear equations or the correlation method. Secondly by using The Fast Fourier Transform (FFT). Where it gives the same result as the other approach, it is extremely more efficient, in reducing the computation time by hundreds. Without FFT, the other techniques which are described would not be practical. The FFT requires only a few lines of code; it is one of the mainly intricate methods in DSP.

J.W. Cooley and J.W. Tukey are given recognition for introducing the FFT to the humankind in their paper: "An algorithm for the machine calculation of complex Fourier Series," Mathematics Computation, Vol. 19, 1965, pp 297-301. [1] The prescribed data are subjected to these transforms i.e., using complex numbers or using real numbers.

The name complex, it doesn't mean that this illustration is difficult or complicated, but that is a particular type of mathematics is used [2]. Complex mathematics often is complicated and intricate, but that isn't the name comes from.

There are various communication standards for wired and wireless communication, a separate FFT length and minimum throughput requires each. FFT operation is frequently implemented as a separate element to congregate computational intensity constraint on a Digital Signal Processor (DSP) [3]. A DSP explanation is relatively simple to execute and usually exhibit high throughput because of elevated clock frequency comparable to FPGAs [4].

To accomplish the minimum throughput requirement of the different standards which are of less power hungry FPGA requires an extremely optimized design. An additional room to this work would be to further improve the proposed explanation to minimize power usage.

The below figure shown termed as simple butterfly diagram because of its faction look. The basic part of the FFT is butterfly.

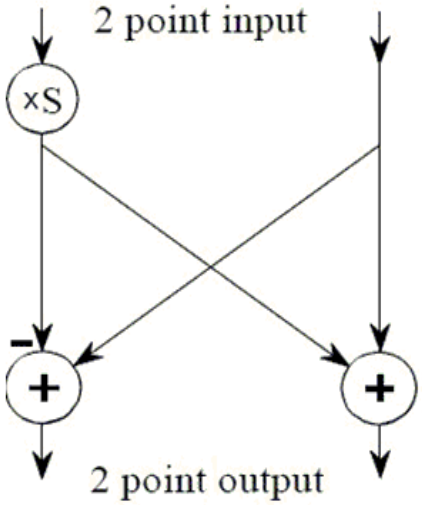

Fig 1: Basic computation part in the FFT

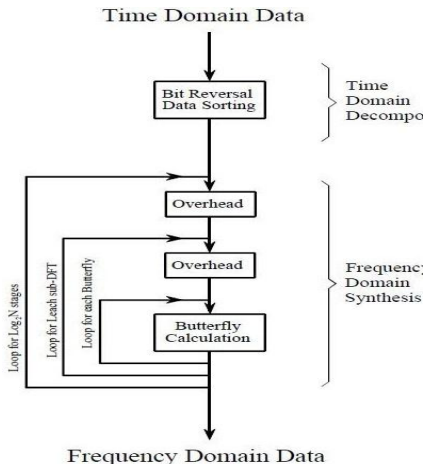

Fig 2: Flow illustration of FFT

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

Above Figure illustrates the arrangement of the complete FFT. The time domain disintegration is obtained with a bit reversal sorting algorithm.

Transforming the disintegrated data into the frequency domain occupies nil and therefore it won't come into sight in the structure.

II. LITERATURE SURVEY

Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy proposed a method Efficient FPGA implementation of FFT/IFFT Processor.Sneha N.kherde#1 Meghana Hasamnis#2 proposed a method Efficient Design and Implementation of FFT.

Tharanidevi .B, Jayaprakash.r proposed a method Implementation of double precision floating point radix-2 FFT using vhdl Mario Garrido, Member, IEEE, J. Grajal, M. A. Sánchez, and Oscar Gustafsson, Senior Member, IEEE proposed a method Pipelined Radix- Feed forward FFT Architectures.

Xin Xiao proposed An Efficient FFT Engine with reduced addressing Logic in this shared-memory-based method, single radix-2 butterfly calculation unit are used in embedded FFT processor since they necessitate least sum of hardware source, and the "in-place" addressing stratagem is a practical requirement to reduce the total memory required. We present the modified butterfly architecture and the improved address generation logic, which is primarily based on inverter, counter, and multiplexors.

Although a shifter is still needed in this design, it shifts only once for each pass instead of each clock. The goal of this is to reduce both the address generation delay and the hardware complexity.

III. PRELIMINARIES

Here an N point signal (N=16) is divided through four separate stages. The first stage split the 16 point signals into exactly half i.e., each signal consists of 8 points. In the next stage decays the divided 8 points into four signals of 4 points. This pattern persists until N signals of a single point observes.

An intersection is used every time to break a signal in to two i.e., the signal is estranged into its even and odd numbered samples.

Here the binary numbers are the differing of each other, i.e. sample 3 (0011) is switch over with bit reversible number 12 (1100). The FFT time domain disintegration is usually passed out by a bit reversal arrangement algorithm.

The FFT function by rancid an N point time domain signal into N time domain signals which are of single point. To calculate the N frequency spectra equivalent to these N time domain signals is the second step. Formation of a single frequency spectrum from the N spectra is the final step.

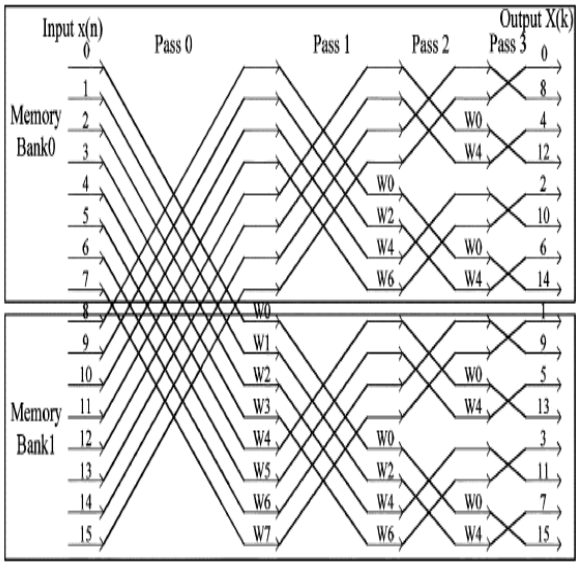

Fig 3: Signal flow graph of a FFT

Sample Numbers in Normal Order	
Decimal	binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Fig 4: The FFT Bit reversal Sorting

The IEEE Standard for Floating Point (IEEE 754), a technical standard for floating point computation. The benefit of floating-point representation more than fixed point and integer representation is that, it can maintain a much broad range of values. 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

IV. PROPOSED METHOD

In the proposed thesis a fractional point radix2 FFT is been generated using 32-bit Single precision IEEE 754 Arithmetic standard with reduced addressing logic is

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

proposed. In the proposed work a 16point FFT is considered and implemented in VERILOG HDL, and synthesized in 40 nm technology of vertex 6. The architecture of address generation circuit is shown below in figure the Heart of the architecture is the BUTTERFLY, The butterfly calculation is discussed below.

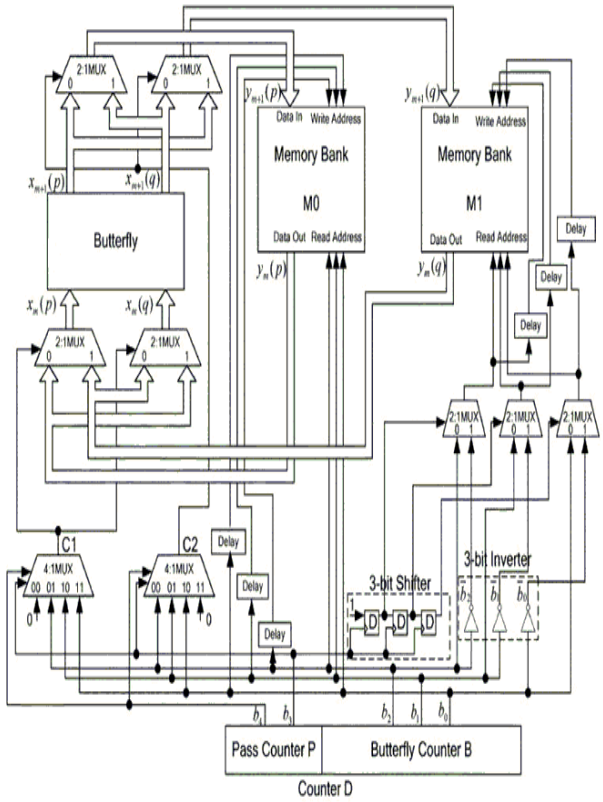

Fig 5: Address generation circuits for a 16-point FFT.

Counter B($b_2b_1b_0$)	Counter $\overline{?}?(\overline{?}_2\ \overline{?}_1\ \overline{?}_0)$	Pass0 (exchange control signal: C1 = 0, C2 = b_2)		Pass 1 (exchange control signal: C1 = b_2, C2 = b_1)		Pass 2 (exchange control signal: C1 = b_1, C2 = b_0)		Pass3 (exchange control signal: == C1 = b_0, C2 = 0)	
		Bank0 address $b_2b_1b_0$	Bank1 address $b_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{?}_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{?}_2\overline{?}_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{?}_2\overline{?}_1\overline{?}_0$
000	111	000	000	000	100	000	110	000	111
001	110	001	001	001	101	001	111	001	110
010	101	010	010	010	110	010	100	010	101
011	100	011	011	011	111	011	101	011	100
100	011	100	100	100	000	100	010	100	011
101	010	101	101	101	001	101	011	101	010
110	001	110	110	110	010	110	000	110	001
111	000	111	111	111	011	111	001	111	000

Table 1: Address generation table of the proposed method for a 16-point FFT

a) BUTTERFLY CALCULATION

The proposed method is for calculation of 16 point fractional FFT; the twiddle factor used in butterfly calculation is given by N/2 where N is variable point FFT. In the proposed method a 16 point FFT is considered, the total number twiddle factors are 8, these 8 twiddle factors are calculated and supplied as inputs to the system. In the verilog design for butterfly calculation, IP CORE GENERATION BLOCKS ADDER, MULTIPLIER, and SUBTRACTOR. RAM'S are used. The inputs and outputs are stored in two RAM'S through multiplexers, the input to the butterfly is considered as a+jb, c+jd the twiddle factor is considered as e+jf and the butterfly operation is performed by adding the both inputs to obtain first output and subtracting the inputs and multiplying with twiddle factor to obtain second output.

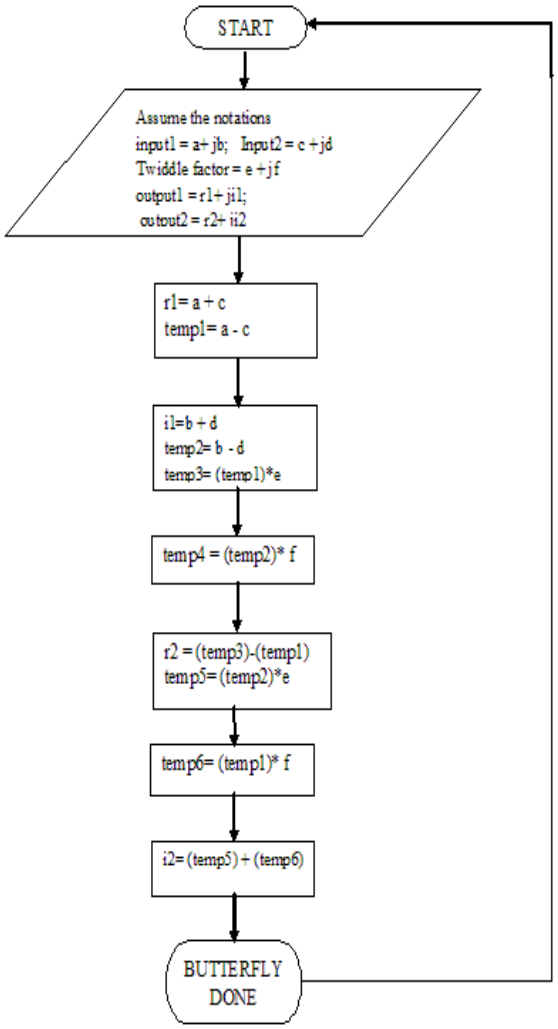

Fig 6: Butterfly Flow chart

From the shown flowchart it clearly explains about butterfly operation the main heart of the FFT. Firstly it assumes the all input data and then operates to get output data 1 and the other results stores in temporary files and evaluates with the twiddle factor to get the final output to store in the output 2.

After all the operation completed it checks and it goes to the butterfly done. And again resets to the first stage. It repeats up to 8 times of each stage and 32 times for 4 stages in the whole FFT likewise 32 times. Hence we can come to know that FFT has done.

For floating calculation we will use floating adder, floating Multiplier, and floating subtractor. Hence the floating point FFT is possible in Verilog with a high clock frequency up to 463MHz.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

V. RESULTS AND DISCUSSION

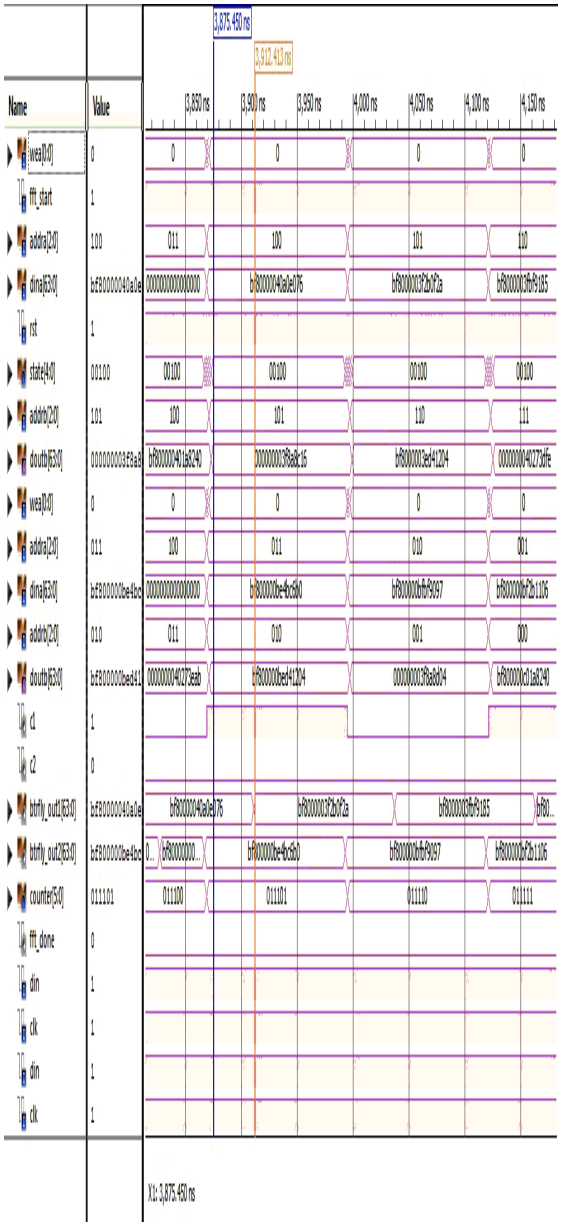

Fig 7: Simulation result shows the output data in Hexadecimal form in RAM1 and RAM 2.

Device utilization summary:

Selected Device: 6vlx75tlff484-1l

Slice Logic Utilization:

Number of Slice Registers:	2570 out of 93120		2%
Number of Slice LUTs	: 2305 out of 46560		4%
Number used as Logic	: 2172 out of 46560		4%
Number used as Memory	: 133 out of 16720		0%
Number used as SRL	: 133		

Timing Summary:

Speed Grade : -1

Minimum period: 2.159ns (Maximum Frequency : 463.237MHz)

Minimum input arrival time before clock : 1.051ns

Maximum output required time after clock : 0.280ns

Table 2: Obtained result compared to existing method

Clock Frequency of Existing structure	Clock Frequency of Proposed structure
280 MHz	463.6 MHz

The Proposed method is simulated and synthesized for test input:

{1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2}.

The output thus acquired by performing FFT algorithm manually, the obtained theoretical results are:

{24, (-1+5.023i), 0, (-1+1.4966i), 0, (-1+0.6682i), 0, (-1+0.1989i), 0, (-1-0.1989i), 0,

(-1-0.6682i), 0, (-1-1.4966i), 0, (-1-5.0273i)}

In the above figures the outputs are displayed in 64bit hexadecimal format where in that 64 bit, 32 bits represents real part and the other 32bits represents imaginary part.

Wea- it denotes write enable bit it writes the data in to ram after the every butterfly operation done it only sets for 3 bit after that it resets until the butterfly operation done.

FFT_start: here it sets when ever only the all data received to do the butterfly operation unless it won't sets, when it sets it starts the butterfly operation.

Addra [2:0] - here address denotes the address of the data where it stores in the RAM.

Dina [63:0] - here it denotes the 64 bit real and imaginary input data.

Rst- resets after the fft done bit sets and again after this whole loop starts.

State- hare state denotes where the present butterfly operates i.e. from the address generation circuit it consists 5 bit counter it denotes the stage and 3 bit counter for 8 butterfly operations for each stage, the total denotes the state.

Addrb [2:0] - here this is the address where the data would be read from the ram or memory location for the butterfly operation.

Doutb[63:0]- The data which is read from the ram will apply as an inputs to the multiplexers the data will be in hexadecimal form i.e. 64bit 32 bit represents real part and other 32bit

VI. CONCLUSION

As the Fast Fourier Transform (FFT) is simply a professional technique to compute the Discrete Fourier Transform (DFT). Whilst memory based FFT processors need less hardware resource but require operating at higher clock frequency to meet the throughput. In this, A Verilog implementation of floating point FFT has been generated with reduced addressing logic using single precision floating point number IEEE 754 standard and improved the throughput of the system with respect to the speed in terms of high clock frequency. The proposed FFT algorithm is synthesized using vertex 6 as an target device. Synthesis is performed with Xilinx version 13.The synthesis results for a 16-point FFT with 64-bit complex number inputs show a maximum clock frequency of 463.6MHz compared to existing method.

REFERRENCES

[1] J.W. Cooley and J.W. Tukey "An algorithm for the machine calculation of complex Fourier series," Mathematics Computation, Vol. 19, 1965, pp 297-301.

[2] Xin Xiao, An Efficient FFT Engine With Reduced Addressing Logic Student Member, IEEE, Erdal Oruklu, Member, IEEE, and Jafar Saniie, Senior Member, IEEE transactions on circuits and systems—ii: express briefs, vol. 55, no. 11, November 2008

[3] Y. Ma, "An effective memory addressing scheme for FFT processors,"IEEE Trans. Signal Process. vol. 47, no. 3, pp. 907–911, Mar. 1999.

[4] D. Cohen, "simplified control of FFT hardware," IEEE trans. Acoustic, Speech, signal process. Vol. ASSP-24, no. 6, pp. 577–579, December 1976.

[5] Mario Garrido , J. Grajal, M. A. Sánchez, and Oscar Gustafsson, " Pipelined Radix- Feed forward FFT Architectures" , IEEE transactions on very large scale integration (vlsi) systems, vol. 21, no. 1, January 2013.

[6] Ahmed Saeed, M. Elbably, G. Abdelfadeel, "Efficient FPGA implementation of FFT/IFFT Processor", INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING Issue 3, Volume 3, 2009

[7] Sneha N.kherde , Meghana Hasamnis, "Efficient Design and Implementation of FFT",

[8] Vinay Gautam, Kailash Chandra Ray, Pauline Haddow Hardware efficient design of Variable Length FFT Processor IEEE, Issue-11, 2011

[9] R. M. Jiang, "An area-efficient FFT architecture for OFDM digital video broadcasting," IEEE Trans. Consum. Electron., vol. 53, no. 4, pp. 1322–1326, Nov. 2007. [2] W. D. Li and L. Wanhammar, "A pipeline FFT processor," in Proc. IEEE Workshop Signal Process. Syst., Oct. 1999, pp. 654–662.

[10] S. S. He and M. Torkelson, "A new approach to pipeline FFT processor," in Proc. 10th Int. Parallel Process. Symp., Apr. 1996, pp. 766–770.

[11] T. M. Hopkinson and G. M. Butler, "A pipelined, high-precision FFT architecture," in Proc. 35th Midwest Symp. Circuits Syst., Aug. 1992, vol. 2, pp. 835–838.