Implementation of 64-Point FFT Processor Based on Radix-2 Using Verilog

T.TIRUMALA KOTESWARA RAO[1], S. SARATH CHANDRA[2]
Student of M. Tech Department of Electronics and Communication Engineering[1], QIS Institute Of Technology, Ongole.
Associate Professor[2], Department of Electronics and Communication Engineering, QIS Institute Of Technology, Ongole.

Abstract

A Fast Fourier transform is an efficient algorithm to compute the discrete Fourier Transform (DFT). The operation has a high computational requirement of large number of operations (N^2 complex multiplications and N (N-1) additions). This makes computational and implementation very difficult. Short length structures are can be obtain higher length FFT. To obtain VLSI structure by using 4-point FFT's to construct N-point FFT rather than 8-point FFT. In this paper the proposed architecture is higher order FFT and it is split into three stages and each stage is radix-2 based 4-point FFT to reduce the number of operations. In this architecture each stage requires 8 complex additions/subtractions to reduce the no of complex multiplications after each 4-point FFT and to keep pipe line way of computation of design. Proposed architecture is implemented using verilog HDL XILINX ISE 12.1. The performance of the proposed architecture is implemented in terms of relative error. The proposed architecture gives best compromise in terms of speed.

Key words: FPGA, 8-point FFT, 4-point FFT, spatial distribution, temporal distribution.

1. Introduction

The Discrete Fourier Transform (DFT) is one of the most important tools used in digital signal processing applications. It has been widely implemented digital communications such as Radars, Ultra wide band receivers (UWB) and many other applications. Computing this operation has high computational requirement and large number of operations (N^2 complex multiplications and N (N-1) additions).This makes computing and implementation very difficult to realize. To reduce the number of operations a fast algorithm has been introduced by Cooley-Tukey [2] called Fast Fourier Transform (FFT). Later FFT reduces the computational complexity from O (N2) to O (NlogN). To reduce the complexity of FFT algorithm other researchers propose numerous techniques like radix-4 [2], split radix [3]. By using these two techniques we can able to avoid the radix-2 structure. These architectures are based on either Decimation in Time Domain (DIT) or Decimation in Frequency (DIF). Much other architecture was proposed on the basis of these architectures. In another way there is growing interest in the Field of Field Programmable Gate Arrays (FPGA). FPGA's have potentially substantially accelerated computational algorithms like FFT's. The Higher Order FFT's are implemented by using High-Cost FPGA's. It is not possible to instantiate 512-point FFT with the XILINX IP core to implement in Spartan-3 family.

To reach this challenge, we present a VLSI structure to allow higher order FFT to be implemented into low cost FPGA's. The remainder this paper is organized as follows. The section I regarding the back ground work for the DFT. Algorithm, in section II is devoted to the proposed low architecture and section III is described about the two kinds of distributions (Temporal and Spatial Distributions). In section II we detail the principle and structure for generalized to higher order FFT's. After that techniques to save area are illustrated. Section IV is presents experimental results and comparisons with IP core and former work quoted in the literature. In Section V finally we conclude the paper.

I Background

For a given sequence x of n samples, the Discrete Fourier transform (DFT) frequency components X (k) may be defined.

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{n.k} \qquad (1)$$

where , $W_N = e^{\frac{-2j\pi}{N}}$ the twiddle factors, n and k are respectively the time and frequency domain indexes. $0 \le k \le N-1,\ 0 \le n \le N-1$ And N is the DFT length. Let us consider that the N = M. T, k = s + T. t and n = l +M. m, where M, T are integers s, l ∈ {0, 1…….M-1} and t, m∈ {0,1…….T-1}. Applying these considerations in (1), we obtain (2)

$$X(s+T.t)=\sum_{l=0}^{M-1}\sum_{m=0}^{T-1}\left[x(l+M.m)W_{M.T}^{((l+M.m)(s+T.t))}\right] \qquad (2)$$

It can be found that (2) is equivalent to

$$X(s+T.t)=\sum_{l=0}^{M-1}\sum_{m=0}^{T-1}\left[x(l+M.m)W_{M.T}^{((l+M.m)(s+T.t))}\right] \qquad (3)$$

Y And finally, (3) can be rewritten

$$X(s+T.t)=\sum_{l=0}^{M-1}W_M^{l.t}\sum_{m=0}^{T-1}\left[W_{M.T}^{l.s}x(l+M.m)W_T^{m.s}\right] \qquad (4)$$

Equation (4) means that it is possible to realize N-point FFT to first Decomposing into one M-point and one N-point FFT where N =M.T and then combining them. For example take 64-point as a case study after that it is generalized for higher order FFT. To perform the 64-point FFT take N=M=8. Then equation (4) is given as follows:

$$X(s+8.t)=\sum_{l=0}^{7}W_8^{l.t}\sum_{m=0}^{7}\left[W_{64}^{l.s}x(l+8.m)W_8^{m.s}\right] \qquad (5)$$

Equation (5) illustrates that the 64-point is expressed by using two-dimensional structure of 8-point FFT. According to the Equation (5) the processing higher order element is 8-point.

II. LOW AREA ARCHITECTURE

The N-point FFT split in to three stages according to next equation.

$$X(s+Mq+MKp)=\sum_{l=0}^{L-1}\sum_{m=0}^{M-1}\sum_{k=0}^{K-1}x(l,m,k)W_N^{(MKl+Mm+k)(MKp+Mq+s)} \qquad (6)$$

The 64-point FFT may be constructed by either of 8-point, 4-point, 2-point FFTs. The structured is not highly and inhomogeneous. Another solution for constructing the 64-point FFT is to split 64-points into three stages of 4-point FFTs. For L=M=K=4, then the Equation for 64-point FFT is given by

$$X(s+4q+16p)=\sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{k=0}^{3}x(l,m,k)W_{64}^{(16l+4m+k)(16p+4q+s)} \qquad (7)$$

Optimizations

The 64-point FFT according to the signal flow graph of fig 3 Radix-4 is the processing element. The 64-point FFT composed of a control unit and 3 blocks of 4-point FFT units, two blocks of multipliers with two phase generator units, with a complex 64-point memory unit. The control unit managing FFT-4 blocks and multipliers and memory unit. The control unit also generates addresses of inputs and out puts of each block.

1) **Radix-4 modification**: Outputs of such algorithms are presented by the following equations

$$X(0)=\overbrace{x(0)+x(2)}^{A}+\overbrace{x(1)+x(3)}^{C}$$

$$X(1)=\overbrace{x(0)-x(2)}^{B}-\overbrace{j(x(1)-x(3))}^{D}$$
$$X(2)=x(0)+x(2)-x(1)-x(3)$$
$$X(3)=x(0)-x(2)+jx(1)-jx(3) \qquad (8)$$

The signal flow graph of radix-4 structure is given in the fig2. The radix-4 algorithm is composed of 8 complex additions/subtractions. To reduce the number of complex multipliers after each 4-point stage and to perform the pipeline way of computational design we have to modify the 4-point FFT architecture.

The Radix-4 is computed as multi input and multi output. This structure requires 4 multipliers in one clock cycle. This structure presents a high speed design but it requires a S2P block to serialize data. For this reason we have to re design the proposed architecture. The resulting design produces one output for one clock cycle. Inter mediate signals are used to know the parallel processing of the data.

2) Memory unit: Phase generator generates the twiddle factor for each output of 4-point FFT. The multiplier performs the complex multiplication and stores it in the 64-point complex memory. This memory is also known as shared memory. By using sharing memory, the memory is reused and shared between all the blocks.

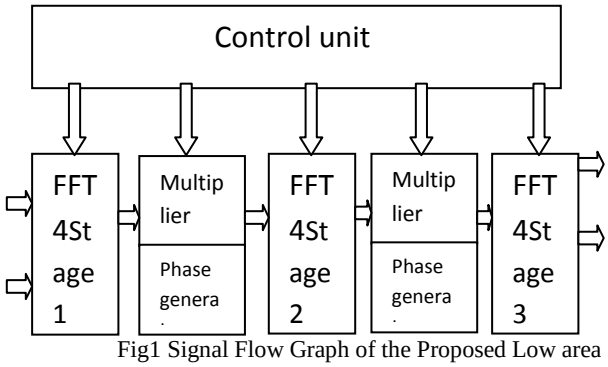

Fig1 Signal Flow Graph of the Proposed Low area architecture

The computation of 64-point FFT based on 4-point FFT needs 3 complex memories. But in the architecture we are using only one memory and that is divided in to the four small parts containing 16-point complex memories in order to improve the latency. By using shared memory we have a problem that it has only one write port. A part of data is saved already and it is not used. If we use the dual port memory and that will be synthesized as BRAM. These are available in the low cost FPGAs.

III. Techniques for Proposed Low Area

A. Spatial Distribution

The possible realization of 64-point FFT is presented in the signal flow graph of fig1. In this distribution the computation of 64-point FFT is composed on five levels. The First levels have two serial to parallel converters used to store real and imaginary data. The second level composed of 8 blocks of 8-point FFT split radix DIT. The non trivial multiplication is given by the third block and it has 49 complex multipliers. The second level is as same as that of second level. Final level is composed of two P2S blocks to give the data in a serial way. All the input data are ready to processed at 64th clock cycle.

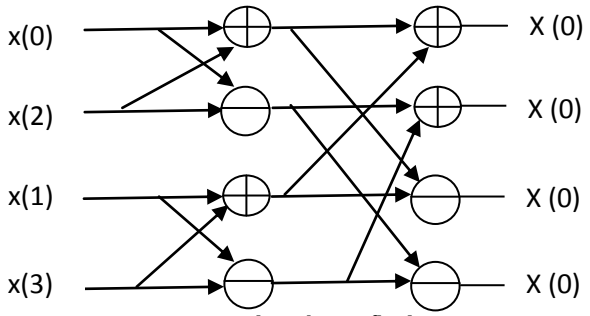

Fig2 Radix-4 butter fly diagram

The 8-point FFT outputs are available and multiplication can be started after 5 clock cycles. The complex multiplications done by Block multiplier needs two clock cycles to perform 49 complex multiplications. 5 clock cycles after the last stage of 8-point FFT blocks the outputs of 64-point FFTs are available. The main advantage of this architecture is high speed and low latency. The implementation of this architecture on FPGA needs high memory and high number of complex multiplications and additions. For this reason the architecture is not suitable for Low cost FPGA such as Spartan 3 family.

B. Temporal distribution

The 64-point FFT may also be realized by temporal distribution. The Temporal distribution has also having 5 levels. In the second and fourth levels it has the only one block of 8-point FFT when compared with spatial distribution. The First level is composed of one S2P block to serialize the input data. The input data may be parallised in order to perform the computation. The S2P blocks are implemented by delay registers. The control unit manages the input data addresses and the first 8-point input data has the address in the form of 8j, j E (0, 1…….. 7).

At the 56th clock cycle the input may processed to the first stage of 8-point FFT. The 8-point FFT outputs are available and multiplication can be started after the 5 clock cycles. At the 57th clock cycle the indexed data 8j+1 may be transformed by the first 8-point FFT and the multiplier outputs are available after 7 clock cycles.The last result of multiplier available at the 71st clock cycle. The results are stored in the complex 64-point memory. Similarly, the second 8-point FFT may be processed and stored in the 64-point complex memory.

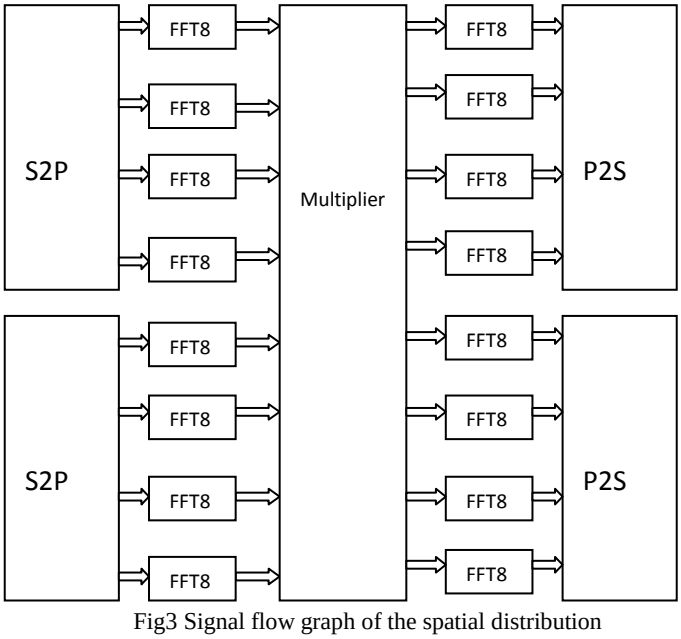

Fig3 Signal flow graph of the spatial distribution

C. Compromise analysis

In both spatial and temporal distributions the 64-point is first decomposed into 8-point FFTs. In terms of throughput both are same and latency in both cases is also same. The latency in both the architectures are given by the formula L (N) = N+7log (N-2). The consumed area is the major difference between the two distributions. The temporal distribution consumes 7 times less area when compared with spatial distribution because it using only one block of 8-point FFT instead of 8 in the spatial distribution. The number of multiplications and number of 8-point blocks is reduced to 7 and 2 respectively. The complex data memory may be removed by using the multiplier to store the result.

The Multiplier outputs are stored in the S2P registers by using the addresses 8j, 8j+1…….. Proceeded one can use the addresses.

The major drawback of decomposition of higher order FFT into 8-point FFT is related to the hardware consumed resources. The percentage of occupied slices in spartan3E XC3S500 while we are using split radix DIT description with 8-point FFT is 30%. The resources are over flowed to design higher order FFT, for this reason we are optimizing our design

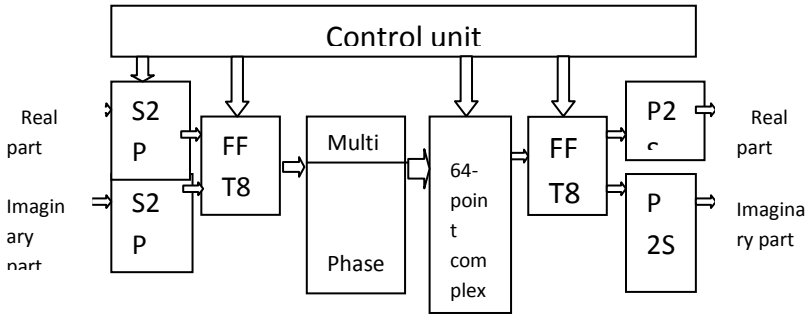

Fig4 Signal flow graph of the temporal distribution

To low cost FPGAs. The other limitation is that the number of inputs by 8-point FFT we have inputs $N=8^n$. To overcome this problem the 8-point may be split into 4-point using radix-4 algorithm. By using this algorithm we can able to reduce 2% of occupied slices in spartan3E XC3S500.

IV. Results

A. Simulation Results

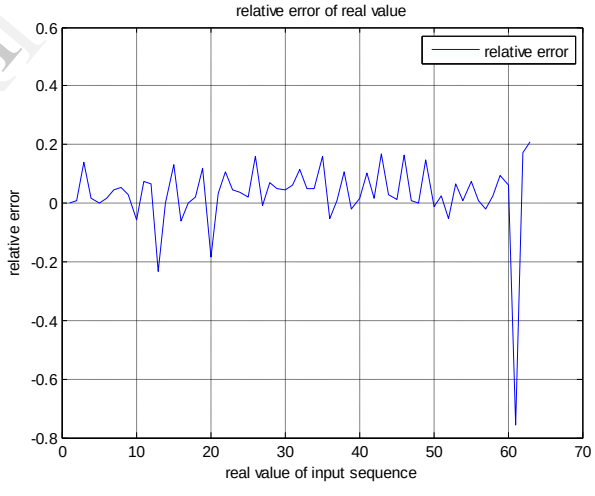

Fig5 Relative error of real value of input sequence

The relative error may be calculated by comparing the MATLAB simulation results with the verilog coding implemented for low cost FPGAs. The phase relative error is as follows. The Relative phase error of phase value gives us the phase angle and the relative error of real value.

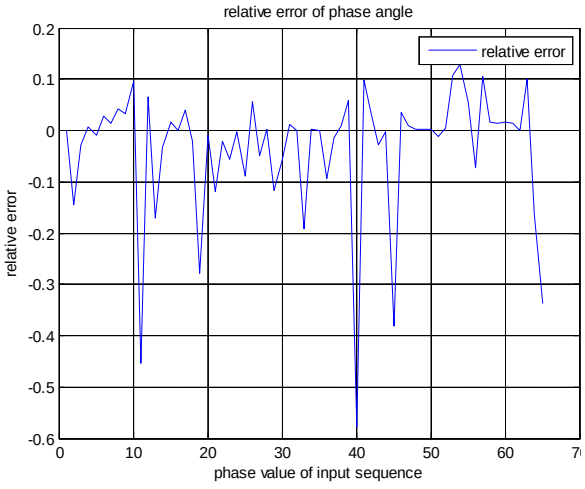

Fig6 Relative error of phase value to the input sequence

B. Synthesis results

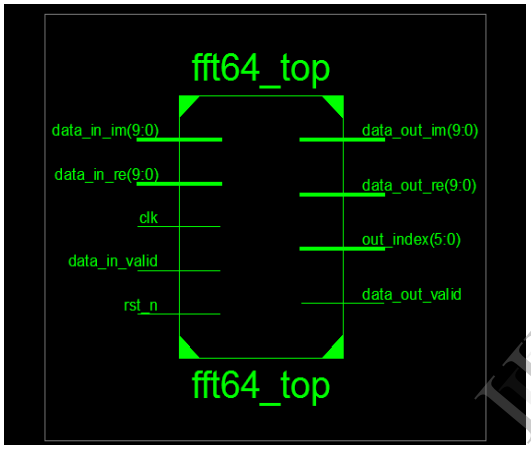

Fig7 TOP LEVEL

C. Implementation results

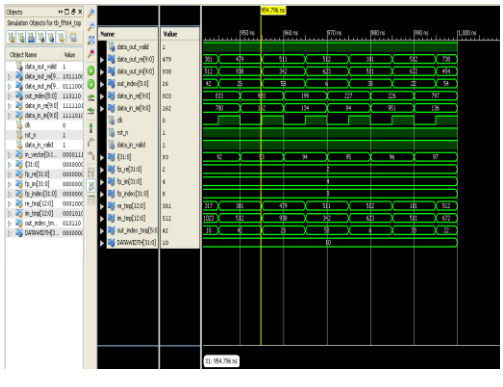

Fig 8 Implementation Results of higer order FFT

V. Conclusion

The techniques to implement higher order FFT into Low Cost FPGA are proposed and implemented. An optimized architecture proposed for higher orders after a detailed study and it is implemented. Our Future work is devoted to the FPGA implementation by the optimization of block multipliers and algorithm proposed in [6] to replace embedded multipliers.

REFERENCES

[1] Yousri Ouerhani, Maher Jridi and A. Alfalou "Implementation techniques of higher order FFT in to low cost FPGA".

[2] W. Cooley and 1. Tukey, An algorithm for the machine calculation of Complex Fourier series, Math. Comput., vol. 19, pp. 297-301, April 1965.

[3] A. Y. Oppenheim, R. W. Schafer, and J. R. Buck,Discrete-Time Signal Processing, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[4] H. Sorensen, M. Heindeman, and C. Burrus, On computing the split-radix FFT, IEEE Trans. Acoustics, Speech, Signal Process, vo1.34, pp. 152-156, 1986.

[5] K. Maharatna, E. Grass, and Ulrich Jagldhold, A 64-Point Fourier Transform Chip for High-Speed Wireless LAN Application Using OFDM, IEEE 1. Solid-State Circuits, vol. 39, pp. 484-493, March 2004.

[6] Xilinx Product Specification, High perfomance 64-point Complex FFTIIF Y.7.0 June 2009 [online]. Available on: http://www.xilinx.com/ipcenter.

[7] M. Jridi and A. Alfalou, A Low-Power. High-Speed DCT architecture for image compression: principle and implementation, in Proc. VLSI Syst. in Chip Conf (VLSI-SoC), pp. 304-309, Sept 2010.

[8] M. Jridi and A. Alfalou, Direct Digital Frequency Synthetizer with CORDIC Algorithm and Taylor Series Approximation for Digital Re-ceivers, Euro Journal of Scientific Research, vol. 30, No. 4, pp. 542-553.