International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

Implementation of Higher Order FFT Processor Using FPGA

Suresh Kumar Dunna[1], B Vijaya Bhaskar[2] , R Suryaprakash[3]

[1]*M.Tech 2nd year, VLSI System Design, Dept of ECE, St.Theressa Inst. of Engg and Tech , Garividi, Vijayanagaram, AP, India.*

[2]*HOD, Dept of ECE, St.Theressa Institute of Engineering & Technology , Garividi, Vijayanagaram, AP. India.*

[3]*Assistant Professor, Dept of ECE, St.Theressa Institute of Engineering & Technology , Garividi, Vijayanagaram AP. India.*

***Abstract* — In this paper, our objective is to detail know-how and techniques that can help the designer of electronic circuits to develop and to optimize their own IP in a reasonable time. For this reason, we propose to optimize existing FFT algorithms for low-cost FPGA implementations. For that, we have used short length structures to obtain higher length transforms. Indeed, we can obtain a VLSI structure by using log4(N) 4-point FFTs to construct N-point FFT rather than (N/8) logs (N) 8-point FFTs. Furthermore, two techniques are used to yield with VLSI architecture. Firstly, the radix-4 FFT is modified to process one sample per clock cycle. Secondly, the memory is shared and divided into 4 parts to reduce the consumed resources and to improve the overall latency.** *(Abstract)*

Keywords-FPGA,8-pointFFT,4-pointFFT,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 in digital communication systems such as Radars, Ultra Wide Band (UWB) receivers and many other applications. Computing this operation has a high computational requirement and needs a large number of operations (N2 complex multiplications and N. (N -1) com-plex 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[1] and called Fast Fourier Transform (FFT). The latter, reduces complexity from O(N2) to O(NlogN). Other researchers, propose numerous tech-niques such as radix-4 [2], split radix [3] to avoid radix-2structure in order to reduce the complexity of FFT algorithm. These architectures are either based on the Decimation-in-Time (DIT) or on the Decimation-in-Frequency(DIP). Several designs based on these architectures were proposed in order to implement these algorithms. On the other hand, there is a growing interest in Field Programmable Gate Arrays (FPGAs) because of their potential to substantially accelerate computational intensive algorithms such as FFTs. Unfortunately, high order FFT are almost implemented into high cost FPGAs. For example, it is not possible to instantiate S12-point FFT with the Xilinx IP core to implement it in Spartan 3 family.

To meet with this challenge, we present in this paper a VLSI architecture to allow the implementation of high order FFT into low cost FPGAs. The remainder of this paper is organized as follows. In section II, definition and two kinds of distributions (spatial and temporal) are introduced. Section III is devoted to the proposed low area architecture. We detail the principle and the structure of 64-point FFT which may be generalized to higher orders. Then, techniques to save area are illustrated. Section IV presents the experimental results and comparisons with IP core and prior works quoted in the literature. Finally, we summarize and conclude this paper in section V.

2. BACKGROUND

2.1 Definition

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

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

Where $W_N = e^{\frac{-2j\pi}{N}}$ is the twiddle factor, n and k are respectively the time and frequency indexes, $0 \le k \le N-1$, $0 \le n \le N-1$ and N is the DFT length.

Let us consider $N = M.T, k = s + T.t$ and $n = l + M.m$, where M, T are integer and $s, l \in \{0,1...M-1\}$ and $t, m \in \{0,1...T-1\}$. Applying these considerations in (1), we obtain (2).

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

It can be found that (2) is equivalent to

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

and finally, (3) can be rewritten

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

Equation (4) means that it is possible to realize N-point FFT by first decomposing into one M-point and one T-point FFT where N = M.T, and then combining them. To illustrate this by example, we take the 64-point as a case study after that we can make generalization to a higher order. To perform 64-point FFT we may choose M = T = 8. Then equation (4) is

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

Equation (5) means that is possible to express the 64-point FFT by two-dimensional structure of 8-point FFT. The processing element of higher order FFT according to equation (5) is the 8-point. Hence, the performance of high length depends in 8-point performance. The choice of 8-point FFT structure becomes crucial. In this work, the 8-point FFT architecture used is the Split Radix DIT because of its lower number of arithmetic operations.

2.2 Spatial distribution

One possible realization of the 64-point FFT is presented in the Signal Flow Graph (SFG) of Fig. 1. It can be observed that computing 64-point FFT is composed on five levels. The first level is composed of two serial to parallel blocks used to store real and imaginary part of data presented in a serial way. The second floor is composed of 8 blocks of 8-point FFT Split Radix DIT. The third block contains 49 complex multipliers used to compute non trivial complex multiplication. The fourth is similar to the second one. the last level is composed of two parallel to serial blocks gives data in a serial way. At the 64th clock cycle all input data are ready to be proceeded. After 5 clock cycles, the 8-point FFT outputs are available and multiplication can be started. Block multiplier needs 2 clock cycles to perform the 49 complex multiplica-tions. The 64-point FFT outputs are available 5 clock cycles after the last stage of 8-point FFT transformation. Hence, the main advantage of this architecture is the high speed and low-latency. However, the implementation of this architecture on FPGA needs high memory, high number of complex multipliers and complex adders. Therefore, this ar-chitecture is not suitable for low cost FPGA such as Spartan 3 family.

2.3 Temporal distribution

Another possible realization of the 64-point FFT is illustrated in Fig. 2. According to this structure, the first stage is realized by one block of 8-point FFT rather than 8 as in Fig. 1. Similarly, the third stage is performed by only one block of 8-point FFT rather than 8. Consequently, the control unit in Fig. 2 plays an important role to synchronize all the treatments. This architecture performs FFT in a pipeline way.

First, input data comes in a serial manner. To perform the computation input data have to be parallelized. This is realized by S2P blocks which are implemented by means of delay registers. On the other side, the control unit manages the input data addresses. The first 8-point input data has the address in the format 8j, j E {O, 1, ... 7}. On the 56th clock cycle these data have been proceeded to the first stage of 8-point FFT. After 5 clock cycles, the 8-point FFT outputs are available and multiplication can be started. Similarly, on the 57th clock cycle, data indexed 8j + 1 will be transformed by the first 8-point FFT and after 7 clock cycles, results data will be available at the multiplier output. And so one until the last result of multiplier output which will be available at the 71st clock cycle. These results are stored on the fly on 64-complex data memory. Likewise, the second 8-point FFT stage will proceed the stored data to compute 64-point FFT.

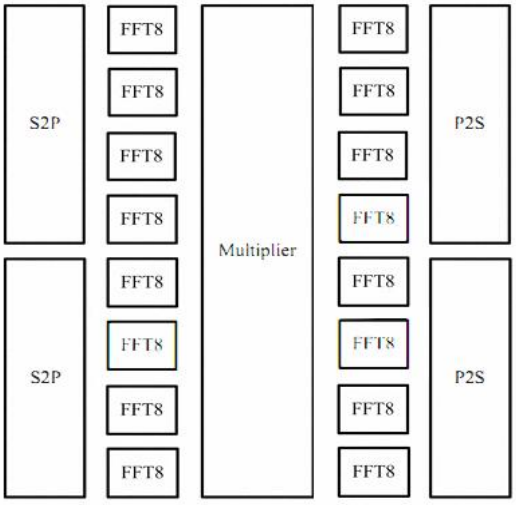

Fig. 1. Signal Flow Graph of the spatial distribution FFT architecture.

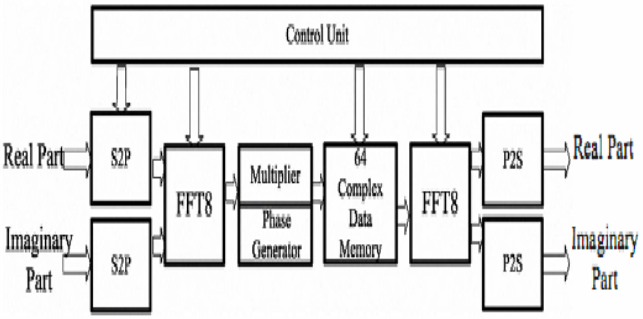

Fig 2. Signal Flow Graph of the temporal distribution FFT architecture

2.4 Compromise analysis

Some concluding remarks related to this section have to be drawn. Firstly, decomposing a high length FFT to 8-point FFTs may be done in a spatial or in temporal distribution. In terms of throughput, the two distributions present one

complex output per clock cycle since data have to be serialized by P2S component. On the other hand, the latency which represents the elapsed time to get the first result is the same. In fact, for a given $N = 8^n$ where n is the number of stages, the latency in both architectures may be expressed as $L(N) = N + 7\log 8(N - 2)$. The main difference between the two distributions is the consumed area. Obviously, the second architecture consumes averagely 7 times less area than the first one. The number of 8-point FFT blocks pass from 16 to 2 and the number of non-trivial multiplier pass from 49 to 7. Furthermore, the complex data memory used in Fig.2 may be avoided by storing the multiplier

Outputs on S2P register. Indeed, since input data at address 8j, 8j+l,.. Are proceeded one can use these addresses to store the multiplier outputs.

Definitively, the major drawback of the decomposition of high length FFT on 8-point FFTs is related to the hardware consumed resources of the 8-point FFT. Synthesis results of the split radix DIT description of 8-point FFT show that the percentage of occupied slices in Spartan3E XC3S500 is about 30%. Therefore, to design a higher order FFT, the FPGA resources will be overflowed. Another drawback is about the limitation of the number of input with exclusively 8-point FFT elements since $N = 8^n$. To overcome this problem we replace the 8-point FFT by a 4-point FFT using radix-4 algorithm. This choice is reinforced by the synthesis results of radix 4 in terms of slice occupation which is about 2%.

3. LOW AREA ARCHITECURE

3.1 Definition

The N-point FFT equation can be split into three stages according to next equation.

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

For N = 64, one possible solution consists on constructing the 64-point FFT according to the temporal distribution by using 8-point, 4-point FFT and 2-point FFTs. The obtained design is not highly structured and inhomogeneous. The second solution consists in constructing the 64-point FFT by three stages of 4-point FFT. For L = M = K = 4, 64-point FFT equation can be written as

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

3.2 Optimizations

Using the radix-4 processing element, we can represent the64-point FFT according to SFG in Fig. 3. The 64-point FFT is composed of a control unit, three blocks 4-point FFT units, two blocks multipliers units with two phase generator units and a complex 64-point memory unit. The control unit, indeed of managing the FFf4, multipliers and memorizing unit, it is used also to generate addresses of the inputs and the outputs of each block.

3.2.1 *Radix-4 modification:* Outputs of such algorithm are presented in next equations

A C

$$X(0)=x(0)+x(2)+x(1)+x(3)$$

B D

$$X(1)=x(0)-x(2)-j(x(1)-x(3))$$

$$X(2)=x(0)+x(2)-x(1)-x(3)$$

$$X(3)=x(0)-x(2)+jx(1)-jx(3) \quad (8)$$

The SFG of the radix-4 structure is illustrated in Fig. 4. It is shown that radix-4 algorithm is composed of 8 complex additions/subtractions.

In order to reduce the number of complex multipliers, after each 4-point FFT and to keep the pipeline way in computation of the design we modify the 4-point FFT architecture.

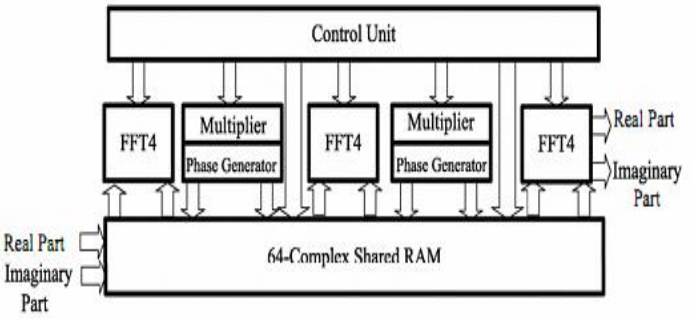

Fig 3. Signal Flow Graph of the proposed low area 64-point FFT architecture.

Usually, the radix-4 is computed as multi-inputs multi-outputs system. This structure requires 4 multipliers in one clock cycle. It is true that this structure presents a high speed design, but almost a P2S block is used to serialize data. For these reasons, we rectify the architecture in order to have one multiplier per clock cycle. So, the resulting design have one complex input and give one complex output per clock cycle as represented in Fig. 4. Intermediate signals A, B, C

and D used in the diagram are indicated to understand the parallel computing.

3.2.2 *Sharing memory:* For each output of the 4-point sFFT block the phase generator generates the correspondent twiddle factor and the multiplier unit performs the complex multiplication and stores the result on 64 complex data memory. This last will be reused and shared between all the blocks as it is shown on Fig. 3.

Usually, computing 64-point FFT based on 4-point FFT needs 3 complex memories. In our architecture we use only one complex 64-point. Moreover, this memory is divided into four small 16-point complex memories in order to improve the latency. Indeed, the problem behind this consists in using one shared memory with only one writer port. This is impossible since a part of data already saved in the memory are not used. Furthermore, if we use a dual port memory, this will be synthesized as BRAM blocks which are oversize and available in limited number in low cost FPGAs.

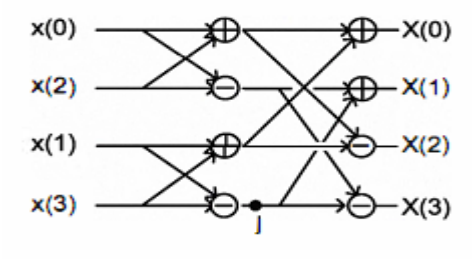

Fig.4 Radix-4 butterfly diagram

4 Experimental Results

4.1 Synthesis results

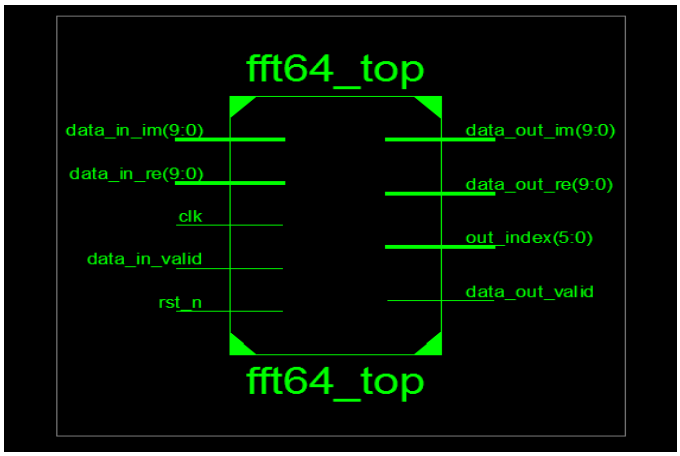

Fig 5.TOP LEVEL

4.2 Implementation results

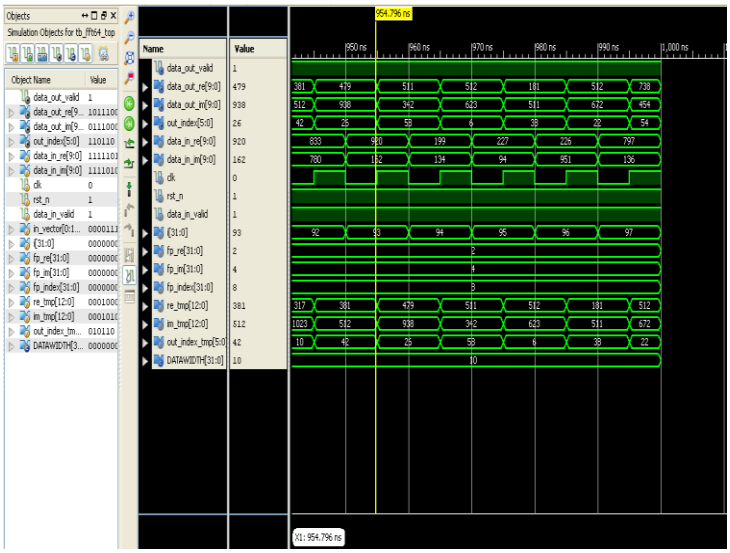

Fig 6.implementation results of higher order FFT

5 Conclusion

Techniques to implement high order FFT into low cost FPGAs were presented and validated. After a comprehensive and a comparative study of existing high order FFTs, an optimized architecture of 64-point FFT was proposed. The transition between 64-point and 256-point was exploited. Higher order FFTs could be obtained with the same manner. Our future work for the FPGA implementation will be devoted to the optimization of the block multiplier and the use of the method proposed in [7]to replace embedded multipliers.

References

[1] Yousri Ouerhani, Maher Jridi and A. Alfalou,implementation techniques of higher order FFT into low cost FPGA Equipe Vision, Laboratoire L@bISEN, CS 42807, 29228 Brest Cedex 2, Francee-mail: {yousri.ouerhani.maher.jridi and ayman.al-falou}@isen.fr

[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, 2009.