High-Speed Parametric FPGA Implementation of FFT/iFFT Blocks for OFDM Transceivers

Behzad Eghbalkhah
University of Tehran
B.Eghbalkhah@ece.ut.ac.ir

Zhila Amini Sheshdeh
University of Tarbiat Modares
Zh.Amini@gmail.com

Mehdi Ehsani-Nick
University of Tehran
Ehsani@gmail.com

Abstract: *FFT and iFFT are ones of the most useful blocks in DSP systems. Since the speed of calculation of the result is one of the important factors of this basic block, in this paper we develop a new method for FFT/iFFT implementation which leads to better computation speed in comparison with other common implementation methods.*

Keywords: FFT, iFFT, implement, FPGA.

1 INTRODUCTION

High performance fast Fourier transform (FFT) are widely used in different areas of applications such as communications, radars, imaging, etc. One of the major concerns for researchers is the enhancement of processing speed. However according to use of portable systems working with limited power supplies, low-power techniques are of great interest in implementation of this block.

FFT and iFFT blocks are used in OFDM links such as Terrestrial Digital Video Broadcasting (DVB-T) systems, Digital Audio Broadcasting (DAB) systems and microwave portable links.

Of course there are many ways to measure the complexity and efficiency of an algorithm, and the final assessment depends on both the available technology and the intended application. Normally the number of arithmetic multiplications and additions are used as a measure of computational complexity.

Several methods of for computing FFT (and iFFT) are discussed in [1]. These are basic algorithms for implementation of FFT and iFFT blocks. A memory-based power-efficient FFT processor is also proposed in [2]. The proposed scheme is based on the minimization of the coefficient access and reduction of switching activity by modifying the butterfly sequence. A reconfigurable FFT processor architecture is proposed in [3]. Another structure using novel order-based processing scheme is proposed in [4].

In this paper we will introduce a high-speed memory-based implementation method for 2K and 8K FFT and iFFT (used in DVB-T systems) exploiting a common structure which is able to be configured with different number of butterflies according to speed-area trade off. This core is developed using VHDL language.

The following of this paper is as follows: in section 2 we will discuss the theory of the implementation method. Section 3 will explain the implementation algorithm and synthesis results and comparison will be declared in section 4 and Conclusions will be followed in section 5.

2 THEORY

Design of this core is based on a famous SFG shown in Fig.1. This SFG is achieved by decimation in frequency (DIF) algorithm and its main specification is sequential and simple method for addressing. Fig. 1 illustrates this SFG for 16 points FFT as an example. The structure would be the same for any other number of FFT points. In this figure the coefficients of the arrow lines are -1 and for the other simple lines the coefficients are 1. In other situations the coefficients are shown on corresponding lines of SFG.

If we use this SFG for implementation of FFT, the W_N-coefficients equal to $W_N = e^{-j\frac{2\pi}{N}}$. Note that the difference between FFT and iFFT is that for iFFT the coefficients will equal to $W_N = e^{j\frac{2\pi}{N}}$. In addition for iFFT a division by N should be done at the end but since $N = 2^{No.\,of\ Stages}$, it is possible to divide the result of each stage by 2 instead of dividing the main result by N.

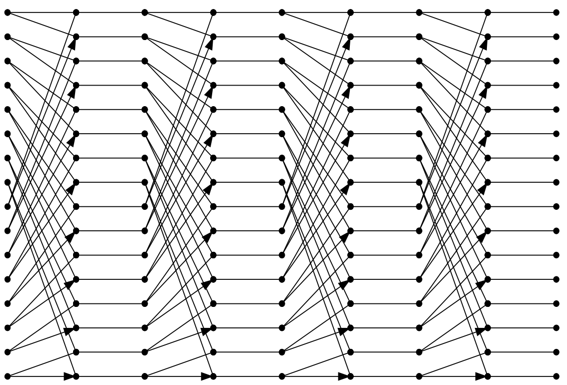

Fig 1. The SFG used for implementation

As illustrated in figure 1, according to regular structure of the SFG the hardware used for implementation of one stage can be used for implementation of other stages as well. In this SFG, applying the points with indices k and k+N/2 to a butterfly as inputs will result two points with indices 2k and 2k+1 in the output. N is the total number of FFT points and $0 \le \mathrm{k} < \mathrm{N}/2$.

3 IMPLEMENTATION

Since the intended application for this core is being used in base-band section of a DVB-T modulator the FFT and iFFT blocks are designed for two 2048 (2K) and 8192 (8K) points and because the second one needs more hardware for implementation, it is assumed as a base of our design. Since the implementation of FFT and iFFT are very similar, only the implementation of iFFT core will be explained.

For an iFFT core with 8192 points, the first stage of the mentioned SFG is shown as in Fig. 2. Note that because of similarity between the stages of the SFG, it is possible to do the whole calculations by repeating the first stages for 13 times.

In this structure only coefficient from W_N^0 to $W_N^{\frac{N}{2}-1}$ are needed and it is also possible to write:

$$W_N^k = e^{j\frac{2\pi}{N}k} = \cos(2\pi k/N) + j\sin(2\pi k/N) \qquad 0 \le k < N/2$$

In this equation the Sin values are all positive for any value of k, but values of Cosinus are positive for $0 \le k < N/4$ and negative for $N/4 \le k < N/2$. For this reason Fig. 2 is divided into 2 parts that in the above part both Sin and Cos coefficients are positive and in the below one the Sin coefficients are positive but Cos coefficients are negative (except for last stage).

Therefore in the calculations the sign of Sinus and Cosinus coefficients is always considered positive and instead the kind of operation is chosen according to the signs. For example if the sign is negative the operation will be changed to subtraction from addition.

Four dual port ROM (DPROM) modules have been used for saving the values of Sin and Cos. In these ROMs only the Sin values for the first quarter of triangular circle are stored. Total number of these values for 8192 points iFFT is 2048. One out of four of these values is used for iFFT with 2048 points.

Since the ROM blocks are dual port, eight Sin values are accessible at the same time, which makes working of 8 butterflies possible simultaneously.

Assuming 16 bit data width, the saved values in ROMs are calculated from $\sin(2\pi k/N) \times ۲^{16}$. With this assumption when $k = 0\ and\ k = N/4$ the Sin or Cos coefficient equals to one, and consequently 17 bits are needed for storage of this coefficients. In order to efficient usage of ROM in these two situations, instead of saving the value of 1×2^{16}, auxiliary circuit for detection of these special situation and generation of 1×2^{16} is used. Thus ROM blocks have 16-bit data width but multipliers have 17-bit input ports.

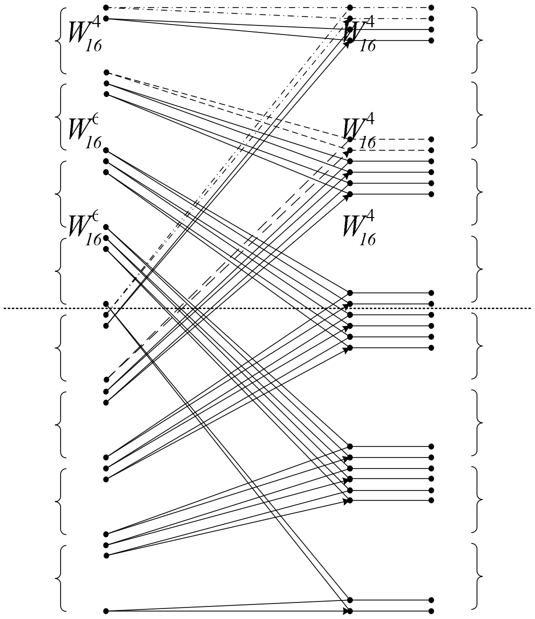

Fig. 2. The First Stage of the implemented SFG

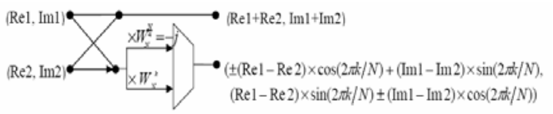
(Re1, Im1)
(Re2, Im2)
(Re1+Re2, Im1+Im2)
$\times W_N^{N/4} = -j$
$\times W_N^k$
$(\pm(\mathrm{Re}1-\mathrm{Re}2)\times\cos(2\pi k/N)+(\mathrm{Im}1-\mathrm{Im}2)\times\sin(2\pi k/N),$
$(\mathrm{Re}1-\mathrm{Re}2)\times\sin(2\pi k/N)\pm(\mathrm{Im}1-\mathrm{Im}2)\times\cos(2\pi k/N))$

Fig. 3. Structure of the used butterflies

Since Flat implementation of this SFG consumes significant hardware, divide Figure 2 is divided into 8 part and two points from each part are accessible simultaneously. So in each step 8 butterflies are needed instead of 4096 butterfly in flat implementation but on the other hand each step will take 512 clock cycles. The structure of these butterflies is illustrated in figure 3.

32 DPRAM blocks and 4 DPROM blocks are used for implementation of iFFT core. 16 DPRAM blocks are used for storing real part and the other 16 blocks are used parallel for saving of imaginary part of data. All of DPROM blocks are used for storing the coefficients.

Each group of 16 DPRAM blocks which are used for real or imaginary part of data are divided into two groups and each group can save 8192 symbols. In the first step one of these groups acts as the source and the other one acts as destination and in the next step their position will be reversed. This will continue up to the last stage. Figure 4 shows the arrangement of DPRAM blocks.

Since these RAM blocks are dual port, there is an opportunity to access to two points in one moment. Therefore for calculation of iFFT with 8192 points each RAM is divided into two parts, the first 512 symbols are stored in the first part and the last 512 in the second part. Port A is used to access to the upper part and the lower part is accessible via port B.

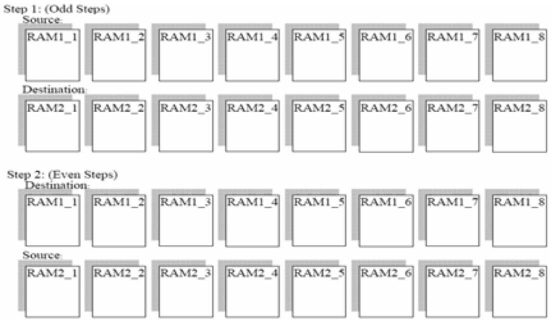

Fig. 4. Arrangement of DPRAM blocks

In each step data from (k+1024×(i-1))th and (k+1024×(i-1)+N/2)th rows through port A and data from (k+512+1024×(i-1))th and (k+512+1024×(i-1)+N/2)th rows through port B are read from the source RAM and fed into two butterflies and the results will be written in (2k+2048×(i-1)), (2k+2048×(i-1)+1), (2k+1024+2048×(i-1)), (2k+1025+2048×(i-1)) th rows of the destination RAMs where 'i' is the RAM index, N is number of iFFT points and k is the index of proper cell. It is important to note that only ¼ of the capacity of the DPRAM blocks is used for implementation of iFFT with 2048 points.

Since 8 DPRAM blocks are used in this architecture in parallel way, 16 points are accessible in each clock. So in each step two blocks of source RAMs contribute to generate the data for two blocks of destination RAMs. Table 1 indicates the access pattern to RAM blocks.

Table 1. RAM access sequence in even and odd stages

Source	Destination
RAM1_1_A, RAM1_5_A	RAM2_1_A, RAM2_1_B
RAM1_1_B, RAM1_5_B	RAM2_2_A, RAM2_2_B
RAM1_2_A, RAM1_6_A	RAM2_3_A, RAM2_3_B
RAM1_2_B, RAM1_6_B	RAM2_4_A, RAM2_4_B
RAM1_3_A, RAM1_7_A	RAM2_5_A, RAM2_5_B
RAM1_3_B, RAM1_7_B	RAM2_6_A, RAM2_6_B
RAM1_4_A, RAM1_8_A	RAM2_7_A, RAM2_7_B
RAM1_4_B, RAM1_8_B	RAM2_8_A, RAM2_8_B

RAM access sequence in odd stages

Source	Destination
RAM2_1_A, RAM2_5_A	RAM1_1_A, RAM1_1_B
RAM2_1_B, RAM2_5_B	RAM1_2_A, RAM1_2_B
RAM2_2_A, RAM2_6_A	RAM1_3_A, RAM1_3_B
RAM2_2_B, RAM2_6_B	RAM1_4_A, RAM1_4_B
RAM2_3_A, RAM2_7_A	RAM1_5_A, RAM1_5_B
RAM2_3_B, RAM2_7_B	RAM1_6_A, RAM1_6_B
RAM2_4_A, RAM2_8_A	RAM1_7_A, RAM1_7_B
RAM2_4_B, RAM2_8_B	RAM1_8_A, RAM1_8_B

RAM access sequence in even stages

As mentioned above for calculation of iFFT with 8192 points there are 512 groups with 16 members that operation for each group takes one clock. So for calculation of 13 stages, total operation takes 512×13 clock cycles. This will take 128×11 clock cycles to calculate an FFT of iFFT with 2048 points.

According to speed-area trade-off, addition of more butterflies to the hardware will result in less clock cycles and consequently smaller calculation time as well as more needed hardware specially multipliers.

4 SYNTHESIS RESULTS

Employing the parametric nature of this core, the iFFT block is implemented on one of Xilinx's Virtex-II Pro™ FPGAs with different configurations. The main difference between these configurations is in the number of their parallel butterflies. Table 2 indicates the number of used FPGA slices, number of used internal 18×18 multipliers and required number of clock cycles.

Table 2. Synthesis results

	Number of Butterflies	Number of Slices	Number of 18×18 Multipliers	Number of Clock Cycles
2048 Point iFFT	2	443	8	128×11
	4	768	16	64×11
	8	1430	32	32×11
	16	2612	64	16×11
8192 Point iFFT	2	504	8	512×13
	4	835	16	256×13
	8	1489	32	128×13
	16		64	64×13

The maximum allowed clock frequency of the core is about 60 MHz. This means that this core can calculate more than 320,000 iFFT operations with 2048 points in one second.

The same algorithm also simulated in Matlab where the input data for this simulation was generated randomly and written in two .dat files. One of these files contains the real part of data and another one is used to store imaginary part of data. Therefore we can easily compare the intermediate data that are calculated in Matlab with the signal values in FPGA implemented design and compare the result in two ways. Also with the use of Matlab iFFT function we can get sure about the final result.

Figure 5 shows the trade off between area and operating speed of iFFT/FFT core for both 2048 and 8192 points.

5 CONCLUSION

A parametric design for iFFT/FFT blocks for implementation on FPGA introduced and synthesis results reported. Since the design was intended to be implemented on a FPGA from Xilinx's Vertex-II Pro™ family, the used blocks were designed according to the specification of this family of FPGA. On the other side because of the need for high data rates, the main goal was to increase the operating frequency and less attention was paid to reduction of consumed area.

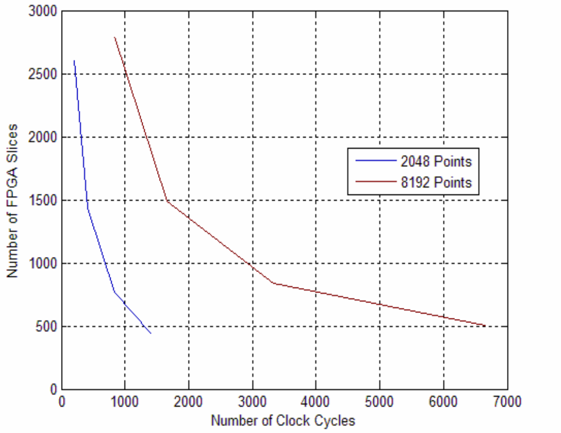

Fig. 5. Number of FPGA Slices vs. Number of needed clock cycles

REFERENCES

[1] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, Second edition, pp. 646-652, 1999.

[2] Seungbeom Lee, Duk-bai Kim, Sin-Chong Park, "Power-efficient design of memory-based FFT processor with new addressing scheme," *Communications and Information Technology, 2004 (ISCIT 2004). IEEE International Symposium on,* Volume 2, 26-29 Oct. 2004 Page(s):678 - 681.

[3] M. Hasan, T. Arslan, J. S. Thompson, "A delay spread based low power reconfigurable FFT processor architecture for wireless receiver," *System-on-Chip, 2003. Proceedings. International Symposium on,* 19-21 Nov. 2003 Page(s):135 – 138.

[4] M. Hasan, T. Arslan, "Implementation of low-power FFT processor cores using a novel order-based processing scheme," *Circuits, Devices and Systems, IEE Proceedings [see also IEE Proceedings G- Circuits, Devices and Systems]*, Volume 150, Issue 3, 6 June 2003 Page(s):149 – 154.