IMPLEMENTATION OF FPGA-BASED FFT CONVOLUTION

Ömer Özdil, Mehmet İspir, Emrah Onat and Alper Yıldırım

TUBITAK BILGEM UEKAE ILTAREN
{omero,mehmeti,emraho,alpery}@iltaren.tubitak.gov.tr

Keywords: Overlap-Add, FFT Convolution, Signal Processing

Abstract

In this research, an efficient pipelined overlap-add filter is designed for the linear convolution of the streaming data. The designed filter uses the fft convolution method and modified radix-4 architecture for the zero-padded input data. Several techniques are used in order to decrease the hardware resources for the proposed design. Firstly, FFT (Fast Fourier Transform) resources are shared between the two FFT units used for the FFT convolution method. Second, bit-reversed inputs are used for inverse FFT operation. The target device for the implementation is a Xilinx Virtex-5 FPGA. The architecture yields similar accuracy and throughput with the cost of less hardware resources than the designs using the commercial FFT IP cores.

1 Introduction

Convolution of streaming data with the long impulse responses has vast uses in area of digital signal processing such as the matched filters in radar receivers, clutter filtering, modelling correlated clutter, noise or band-pass filtering the received signal. FFT convolution method used with an efficient FFT processor addresses this issue and takes the linear convolution to the frequency domain [9]. FFT convolution method is an efficient algorithm for the block data or the streaming data with less throughput rate than the processing rate. In order to use the FFT convolution method on a streaming data, more than one serial FFT blocks have to be used with a multiplexer so as to keep up with the streaming data. Therefore, we propose a new FFT structure for the zero-padded input data in order to decrease the hardware resources required.

Cooley-Tukey [3] proposes a radix-2 algorithm for the computation of the DFT. Several techniques are proposed in order to reduce the complexity of the radix-2 structure such as the radix-4 [7] and split-radix [10] structures. Also, these algorithms can be classified in each other by their design motivations. In [5], radix-4 algorithms are classified as, power-efficient architectures [11], [2], [6], pipelined architectures, area-efficient architectures [12], [1], and lastly, the parallel architectures where [5] is in this group, too. Our design goal is a parallel area-efficient FFT architecture for the specific purpose of implementing overlap-add or overlap-save methods.

The roadmap of this paper is as follows. Summary of the FFT convolution method and the derivation of FFT operation is presented in the next chapter. Then, the need for a modified architecture and the details of the modified FFT convolution and FFT architecture is presented. Lastly, comparisons with a commercial IP core is given.

2 Background

2.1 FFT Convolution

As seen from the Fig. 1, overlap-add method divides the input into smaller segments of length L, then appends $M-1$ zeros to these segments in order to avoid aliasing where M is the length of the filter impulse response. After the N-point DFT (Discrete Fourier Transform) of each segment where $N \geq L + M - 1$, the signal is filtered according to the Eq. (1),

$$Y(k) = H(k)X(k), k = 0, ..., N - 1 \tag{1}$$

where $H(k)$ is the frequency response of the filter, $X(k)$ and $Y(k)$ are the frequency responses of the input and the output. After multiplication with the DFT of the zero-padded impulse response, the inverse DFT of each segment is taken and last $M - 1$ points of all the segments are added together with the first $M - 1$ points of the segments coming next. If we assume that the filter coefficients are stored in the frequency domain, FFT convolution method needs one DFT, one inverse DFT and N multiplications in order perform a convolution. With the help of FFT which is a more computationally-efficient version of DFT, FFT convolutions can be done with the overlap-add or overlap-save method by using less hardware resources than the convolution operation done in time domain.

2.2 FFT

DFT of a sequence x with length N is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \tag{2}$$

where $W_N^{nk} = e^{-2\pi jnk/N}$ is twiddle factor, n and k are respectively time and frequency indexes, $0 \leq k \leq N - 1$, $0 \leq n \leq N - 1$ and N is the DFT length. Let $N = MT$, $k = s + Tm$ and $n = l + Mt$ where M and T are integer and $m, l \in \{0, 1, ...M-1\}$ and $s, t \in \{0, 1, ...T-1\}$. After making the change of variables in Eq. (2),

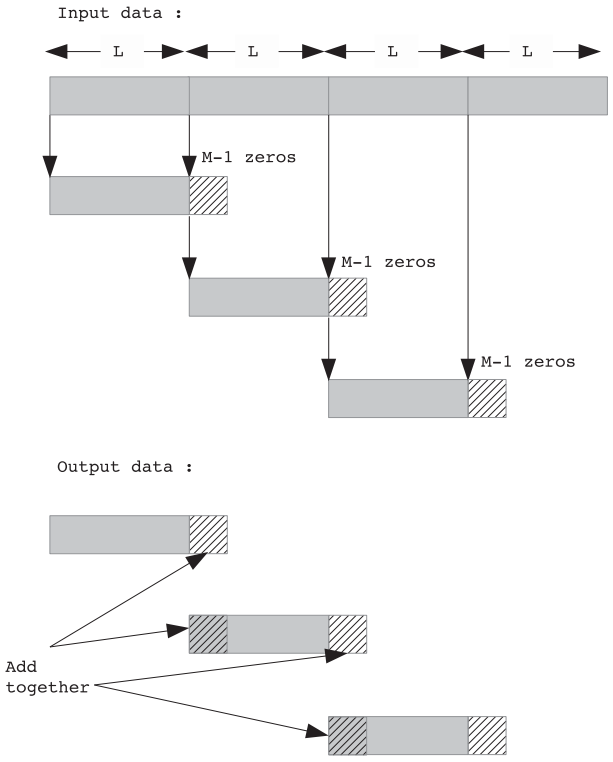

Figure 1: Overlap-add Method

$$X(s+Tm) = \sum_{l=0}^{M-1} W_M^{lm}[W_{MT}^{ls} \sum_{t=0}^{T-1} x(l+Mt)W_T^{ts}] \quad (3)$$

Eq. (3) means that N-point FFT can be accomplished by one M-point FFT and one T-point FFT[8].

Eq. (3) can be extended to four stages for $N = 256$.

$$X(k) = \sum_{l=0}^{3}\sum_{x=0}^{3}\sum_{q=0}^{3}\sum_{z=0}^{3} x(l,x,q,z)W_{256}^{nk} \quad (4)$$

where $n = l+4x+16q+64z$ and $k = e+4f+16d+64m$.

3 Implementation

3.1 Radix4 Unit

For a 64-point parallel FFT, the gain from using radix-4 instead of radix-2 is nearly 60 multipliers and the gain from using split-radix instead of radix-4 is nearly 10 multipliers [4]. Because of the ease of the implementation and the size of the FFT designed for the FFT convolution method being 256, radix-4 architecture is found suitable for the FFTs. Radix-4 processing element can described with the following equations.

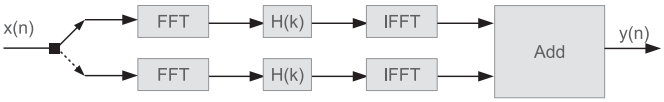

Figure 2: Signal Flow Graph of the FFT Convolution Method Designed with Commercial FFT Cores(Standard Implementation)

$$X(0) = \overbrace{x(0)+x(2)}^{A} + \overbrace{x(1)+x(3)}^{C} \quad (5)$$
$$X(1) = \overbrace{x(0)-x(2)}^{B} - j\overbrace{(x(1)-x(3))}^{D}$$
$$X(2) = x(0)+x(2)-x(1)-x(3)$$
$$X(3) = x(0)-x(2)+j(x(1)-x(3))$$

For the fully parallel systems, the radix-4 processing unit takes four inputs and generates four outputs which requires multiple multiplication units appending the radix-4 unit. For the serialized, pipelined radix-4 unit one input is taken and one output is generated in order to use one multiplication block appending the radix-4 unit. Fully parallel unit uses too many multipliers and the fully serial unit can not keep up with the input data rate without using two FFT blocks for the zero-padded FFT. The signal flow graph of an implementation of the FFT convolution method with a commercial fully-serial FFT core is presented at Fig. 2. The input data has to be switched between two FFT blocks in order to keep up with the input data rate without changing the operation clock or the input rate. As it is seen at Fig. 1, after L samples, $M-1$ zeros are added to the data and the FFT is taken, but if this operation is done with a serial FFT, while zeros are being appended to the input, more data is incoming at the same time, so more than one FFT blocks are needed.

For $L = 128$ and $M = 129$, for FFT convolution method the input rate of the FFT is twice the input rate. In order to exploit this, the FFTs are made to process two inputs at a time. The proposed design takes two inputs and generates two outputs and needs two multipliers appending the radix-4 units. There is not any benefit in terms of multipliers, but all the other control, addition and storage units can be shared. Although, the number of multipliers are same in this design, this design enables the sharing of multiplier units for further research by exploiting the fact that often the twiddle factors are equal to 1. The timing diagram of the proposed radix-4 unit is given at Fig. 3.

3.2 Complex Multiplication

After every radix-4 stages, complex data is multiplied with the complex twiddle factors. Complex multiplication can be defined with the following equations.

$$y_r + jy_i = (x_r + jx_i)(w_r + jw_i) \quad (6)$$

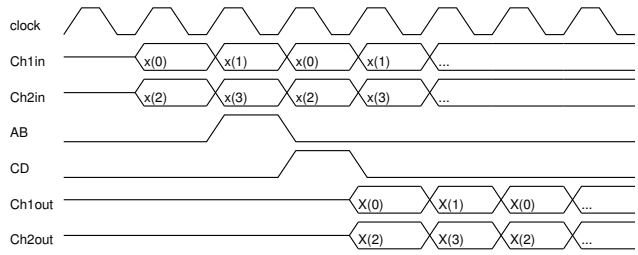

Figure 3: Timing Diagram of the Proposed Radix-4 unit.

$$y_r = x_r w_r - x_i w_i \tag{7}$$
$$y_i = x_r w_i + x_i w_r$$

Four real multiplications are needed in order to calculate a complex multiplication according to Eq. (7). Complex multiplication can be modified as:

$$y_r = x_i(w_r - w_i) + (x_r - x_i)w_r \tag{8}$$
$$y_i = x_i(w_r + w_i) - (x_r - x_i)w_r$$

The modified complex multiplication algorithm at Eq. (8) needs three real multiplications and 3 additional addition operations in order to calculate a complex multiplication. In this research and in commercial IP cores, generally complex multiplication with 3 multipliers is used, because the multiplication is a resource-heavier operation than addition.

3.3 FFT Convolution Method

For the FFT Convolution method the input data is divided into segments. These segments are padded by zeros before the FFT operation. In the proposed architecture the data is reordered and fed to the first input data channel shown in Fig. 4. The other input channel is fed with zeros. The data is propagated through four stages of radix-4 units for the FFT of length 256. Between each stages of radix-4 units, the data is multiplied with the complex twiddle factors. For the two data channels, 6 real multiplications are needed, because 3 multiplications are used for one complex multiplication. After the last stage of the radix-4 unit, the output of the FFT is in bit-reversed order. The inverse FFT operation needs data to be in bit-reversed order so there is not any need for data reordering. The output of the FFT is multiplied with the coefficients which are also in bit-reversed order too and fed to the inverse FFT operation as seen at Fig. 6. The IFFT operation is seen at Fig. 5 which is same with the FFT operation except the twiddle factors. As opposed to the design in Fig. 2, this modified architecture uses one FFT and one IFFT block for the FFT convolution method.

4 Results

The implementation results of the proposed architecture and the design using commercial IP cores are presented at Table 1 for $L = 128$ and $M = 129$. The proposed

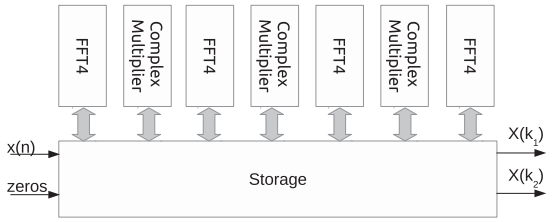

Figure 4: Signal Flow Graph of the Modified Radix-4 256-point FFT Block

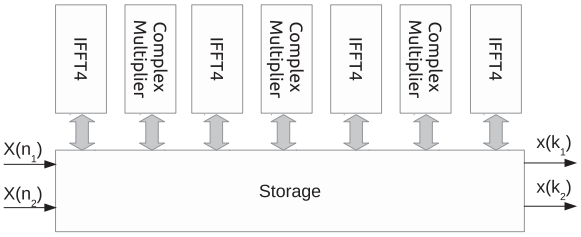

Figure 5: Signal Flow Graph of the Modified Radix-4 256-point IFFT Block

FFT Convolution architecture and FFT architecture uses the same number of multiplications with the design using commercial IP cores. However, these design uses less slice LUTs and slice registers, because all the other storage, control and addition operations are shared between the FFT operations. Another benefit of this design is the capability of the inverse FFT operation to take bit-reversed inputs, so the data can be propagated from FFT operation to IFFT without data reordering.
The latency of the FFT convolution operation for $L = 128$ and $M = 129$ is 1.16μs while the latency of the design using commercial IP cores is 5.3 μs.
The accuracy of the proposed design is demonstrated at Table 2. The proposed design is scaled at every stage so in order to compare the accuracy of the design with the matlab fft, the peak lobe level and sidelobel level ratio is used as a metric. f_s is the sampling frequency. The difference between the MATLAB FFT function and the proposed design results from the quantization error.

5 Conclusion

In this research, a new modified design for the radix-4 architecture is proposed for the FFTs used in FFT Convolution methods. The proposed architecture is implemented for the filter length of 129 and can be extended for the other filter lengths. The proposed design uses less hardware resources and less latency, and yields similar throughput and accuracy with the commercial IP cores. The current design

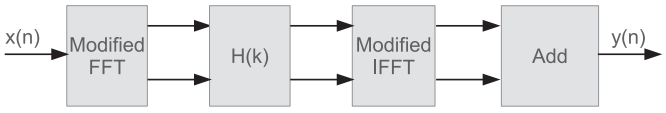

Figure 6: Signal Flow Graph of the Modified FFT Convolution Method

Table 1: FPGA Implementation Comparison

Resource Type	Proposed	Xilinx IP
Slice Registers	3021	10152
Slice LUTs	3712	9082
Block Rams	3	3
DSP48E	42	42

Table 2: Comparison with MATLAB FFT function.

Frequency	Proposed	Matlab	Error
$f_s/4$	9.542425	9.535447	0.007
$f_s/8$	9.383047	9.316743	0.063
$f_s/16$	9.028317	8.987808	0.0405
$f_s/32$	8.299467	8.309956	0.0105

uses the same number of multipliers with the commercial IP core but it enables future work for the sharing of multipliers in FFT stages. By adding memory between the radix-4 stages and multipliers, and a controller, the multiplier usage can be optimized in the design.

References

[1] Xilinx product specification, high performance 64-point complex fft/ifft v.7.0, 2009.

[2] C. Wang C. Chang. Efficient vlsi architectures for fast computation of the discrete fourier transform and its inverse. *IEEE Transactions on Signal Processing*, 48(11), Nov 2000.

[3] J.W. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput.*, 19:297–301, Apr 1965.

[4] Chi hau Chen. *Signal Processing Handbook*. CRC Press, 1988.

[5] M. Jamali, J. Downey, N. Wilikins, C.R. Rehm, and J. Tipping. Development of a fpga-based high speed fft processor for wideband direction of arrival applications. In *Radar Conference, 2009 IEEE*, pages 1 –4, may 2009.

[6] T. Arslan M. Hasan and J.S. Thompson. A novel coefficient ordering based low power pipelined radix-4 fft processor for wireless lan applications. *IEEE Transactions on Consumer Electronics*, 49(1), Feb 2003.

[7] A.V Oppenheim, R.W. Schaefer, and J.R. Buck. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1998.

[8] Y. Ouerhani, M. Jridi, and A. Alfalou. Implementation techniques of high-order fft into low-cost fpga. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1 –4, aug. 2011.

[9] J.G. Proakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[10] H. Sorensen, M. Heideman, and C. Burrus. On computing the split-radix fft. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(1):152 – 156, feb 1986.

[11] J. Takala. T. Pitkanen, T. Partanen. Low-power twiddle factor unit for fft computation. In *SAMOS*, 2007.

[12] C. Wang and Y. Lin. An efficient fft processor for dab receiver using circuit-sharing pipeline design. *IEEE Transactions on Broadcasting*, 53(3), Sep 2007.