Parallel FFT and Parallel Cyclic Convolution Algorithms with Regular Structures and no Processor Intercommunication

Marvi Teixeira, Miguel De Jesus, Yamil Rodriguez
Polytechnic University of Puerto Rico, Hato Rey, PR 00919
mteixeir@caribe.net, eemdejesus@yahoo.com, yrodzii@yahoo.com

Abstract[1]

We have found parallel algorithmic implementations, suitable to compute FFTs and one-dimensional Cyclic Convolutions, which also map well to FPGAs and multiprocessor computational environments. Certain algorithms, such as the Agarwal-Cooley cyclic convolution algorithm transforms the circulant matrix into a block circulant matrix, where each block is itself circulant and therefore can be independently processed as a subconvolution. This method requires that the convolution length, N, be of the form N=RS, where R and S must be mutually prime. There are other techniques in which a one-dimensional convolution can be performed by means of a multidimensional convolution. The problem is that the number of points in each dimension has to be doubled by zero padding. We have found, however, that if the length of the sequences is composite, N = RS (no need for R and S to be mutually prime) the circulant matrix can be factored into a block pseudocirculant matrix. Each block is circulant in itself and amenable to be independently processed. There is a preprocessing stage involving decimation by R, and a final reconstruction stage. Depending on the chosen implementation strategy no processor intercommunication is needed. To compute a parallel DFT the Bluestein Algorithm can be used in order to translate the DFT into a Cyclic Convolution followed by the use of the proposed parallel constructs. The current trend toward FPGA and multiprocessor implementations justifies gaining further insight into these algorithms and exploring their implementation in high performance architectures as well as their application to limited memory processing environments. The target test-bed architecture is a 64 node, beowulf PC cluster. The primary software platform will be pMATLAB developed at MIT. Different FPGA implementations are also been considered.

Introduction

Several researchers have derived in the past algorithms suitable to perform fast cyclic convolutions using the same approach leading to the decimation in time FFT, in particular decimation by R = 2. We have generalized these ideas showing that factorizations of the DFT matrix can be mapped to factorizations of the circulant matrices, which in turn could lead to fast cyclic convolution algorithms. Such algorithms can be considered as "duals" of the FFT algorithms originated by the mentioned factorizations of the DFT matrix, [1], [2]. In particular, decimation (by R) in time FFTs can be related to a class of cyclic convolution algorithms by the following factorization of the circulant matrices [1],

$$H_p = P_{N,R} H_N P_{N,R}^{-1} \tag{1}$$

where $P_{N,R}$ are stride by R permutations, H_N is a circulant matrix and H_P is the resulting block pseudocirculant matrix. Use of Hp, instead of H_N, permits to compute the original convolution using R^2 parallel, subconvolutions. The number of parallel sections can be further reduced by using different well known factorizations. For R = 2 we have

$$P_{N,2}y = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} H_0 & S_{N/2} H_1 \\ H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = (H_P) P_{N,2} x \tag{2}$$

where $H_P = \begin{bmatrix} H_0 & S_{N/2} H_1 \\ H_1 & H_0 \end{bmatrix}$, and $S_{N/2}$ is a cyclic shift operator. A direct implementation uses four (R^2) processors or parallel sections. One possible factorization of the block pseudocirculant matrix renders three or ($R(R-1)+1$) parallel sections.

$$\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} I_{N/2} & 0 & S_{N/2} \\ -I_{N/2} & I_{N/2} & -I_{N/2} \end{bmatrix} \cdot \begin{bmatrix} H_0 & 0 & 0 \\ 0 & H_0 + H_1 & 0 \\ 0 & 0 & H_1 \end{bmatrix} \cdot \begin{bmatrix} I_{N/2} & 0 \\ I_{N/2} & I_{N/2} \\ 0 & I_{N/2} \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} \tag{3}$$

Realization Examples

Case 1: The above factorization is implemented for N = 8. Since this is a case of $N = R^M$ with $M=3$ and $R=2$, the flow graph structure is completely regular. The inner, length-2, convolutions are implemented via frequency domain. Each one of the three parallel sections is a length R^{M-1} sub-convolution that could be realized using any approach.

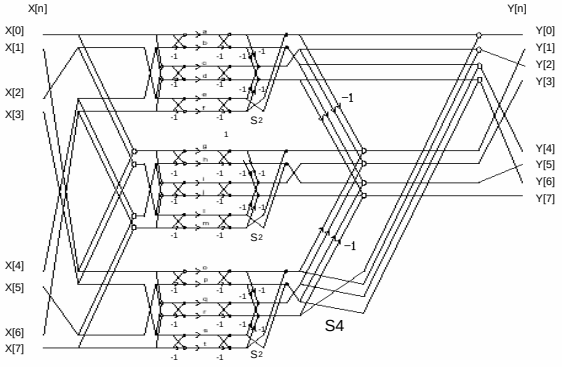

Figure 1: Cyclic Convolution using 3 parallel sections. Note that the smaller sections have a recurrent structure.

1 This work was supported in part by a grant from the Puerto Rico Industrial Development Company (PRIDCO), Hato Rey, Puerto Rico, USA.

Case 2: Decimation by *R=3*. In this case the direct implementation renders $R^2 = 9$ parallel sections.

$$Yp = P_{N,3}y = \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} H_0 & S_{N/3}H_2 & S_{N/3}H_1 \\ H_1 & H_0 & S_{N/3}H_2 \\ H_2 & H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = (H_P)P_{N,3}x \quad (4)$$

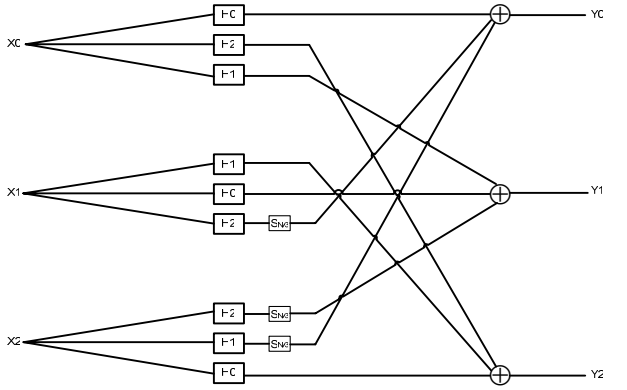

Figure 2: Cyclic Convolution using 9 (processors) parallel sections. Stages marked with $S_{N/3}$ are cyclic shifts.

One, out of various, possible factorization reduces the number of parallel sections to *R(R-1)+1 =7*. The tradeoff is an increase in structural complexity.

$$Y_P = \begin{bmatrix} I_3 & S_3 & S_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_3 & I_3 & I_3 & S_3 \\ -I_3 & -I_3 & -I_3 & I_3 & -I_3 & -I_3 & -I_3 \end{bmatrix} \begin{bmatrix} H_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & H_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & H_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & H_0+H_1+H_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & H_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & H_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & H_2 \end{bmatrix} \begin{bmatrix} I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & I_3 \\ I_3 & I_3 & I_3 \\ I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}$$

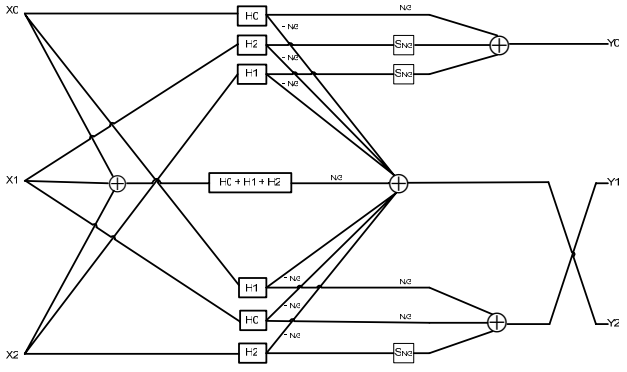

Figure 3: Cyclic Convolution using 7 parallel sections.

Case 3: Decimation by *R = 4*. In this instance the direct implementation renders $R^2 = 16$ parallel sections.

$$Yp = P_{N,4}y_n = Y_P = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} H_0 & S_{N/4}H_3 & S_{N/4}H_2 & S_{N/4}H_1 \\ H_1 & H_0 & S_{N/4}H_3 & S_{N/4}H_2 \\ H_2 & H_1 & H_0 & S_{N/4}H_3 \\ H_3 & H_2 & H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = (H_P)P_{N,4}x \quad (5)$$

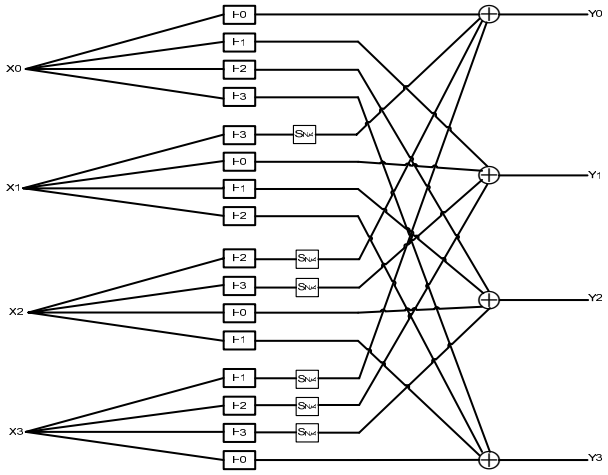

Figure 4: Cyclic Convolution using 16 parallel sections.

If the length of each parallel sections is composite then a parallel construct can be considered in order to realize each subconvolution. In such case, the complexity of pre and post processing stages increases.

Computational Complexity

Assuming that each parallel subconvolution is performed via frequency domain using the FFT, we depict the multiplicative cost for cc. (see our poster for other options):

Table 1: Parallel Cyclic Convolution Complexity

Method	N = 2^M = R·S	# of Mults per Method	# of Processors	# of Mults per Processor
Direct	N^2 = 2^(2·M)	N^2	1	2^(2·M)
Direct FFT	N = 1·2^M	3/2·N·log2(N) + N	1	2^M·[3/2·M + 4]
Parallelized	N = 2·2^(M-1)		3	2^(M-1)·[3/2·(M-1) + 4]
Parallelized	N = 4·2^(M-2)	{R·(R-1)+1]·[3/2·S·log2(S) + 4·S]	13	2^(M-2)·[3/2·(M-2) + 4]
Parallelized	N = 8·2^(M-3)		57	2^(M-3)·[3/2·(M-3) + 4]

		# of Mults per Processor					
Method	# of Processors	M = 4	M = 8	M = 12	M = 16	M = 20	Processing Time
Direct	1	256	65536	16,777,216	4.30E+09	1.10E+12	~T^2
Direct FFT	1	160	4096	90112	1,835,008	35,651,584	T
Parallelized	3	68	1856	41984	868352	17039360	~T/2
Parallelized	13	28	832	19456	409600	8126464	~T/4
Parallelized	57	11	368	8960	192512	3866624	~T/8

Conclusions

A class of algorithms suitable to map parallel FFTs and parallel Cyclic Convolutions to FPGAs and multiprocessor environments has been presented. Different combinations of parallel sections and decimation rates may yield different performance rates.

References

[1] M. Teixeira and D. Rodriguez, "A class of fast cyclic convolution algorithms based on block pseudocirculants," *IEEE Signal Processing Letters* Vol. 2, No. 5, May 1995.

[2] M. Teixeira and D. Rodriguez, "A novel derivation of the Agarwal-Cooley fast cyclic convolution algorithm based on the Good Thomas prime factor algorithm," Proceedings of the IEEE 37th Midwest Symposium on Circuits and Systems, Lafayette, Lousiana, Aug. 1994.