AN OVERLAP SAVE ALGORITHM FOR BLOCK CONVOLUTION WITH REDUCED COMPLEXITY

Jung Gap Kuk, Se Yoon Kim and Nam Ik Cho

Seoul National University
School of Electrical Engineering
Seoul 151-744, Korea
Email : jg-kuk@ispl.snu.ac.kr, light4u@ispl.snu.ac.kr and nicho@snu.ac.kr

ABSTRACT

We propose a block convolution algorithm that requires shorter length FFT than the conventional overlap save algorithm (OSA). It is shown that the OSA can be split into two separate processes related to the previous and current data blocks. Hence, only current data block needs to be transformed in the proposed OSA, whereas the concatenated block of previous and current data is transformed in the conventional method. As a result, the number of arithmetic operations for the block convolution is reduced. Also, the reduced transform size gives additional advantage in data manipulation when implemented on DSP and PC.

Index Terms— Block Convolution, Overlap Save Algorithm, Fast Fourier Transform (FFT)

1. INTRODUCTION

It is well known that the block convolution is often implemented by OSA using the FFT [1]. There is another approach using the number theoretic transform, which has advantages when implemented in fixed point arithmetic [2]. There are also variants of block convolution methods considering many aspects of implementation such as input-output delay and hardware/software realizations [3, 4]. For the software implementation of block convolution on modern computers with floating point arithmetics, the conventional OSA is still the most popular method because of its simple data manipulation and the availability of powerful FFT libraries. For example, audio processing that requires large length convolutions employs the conventional OSA as a basic building block [5, 6], and the block adaptive filtering is also implemented with conventional OSA [7]. Hence, we focus on the reduction of computations in the conventional OSA, and propose a new algorithm that requires the half-sized transform. The proposed algorithm separates the computations required for the previous and current block, whereas the conventional OSA computes the transform of the overall block at once. For the efficient implementation of this scheme, a new transform is required, which will be called quarter-DFT (QDFT). The QDFT can be implemented by twiddle factor multiplications to the input followed by a standard DFT. Comparison of computational complexity shows that the OSA based on the proposed QDFT reduces the number of arithmetic operations in proportion to the transform length. In addition to reduced computational complexity, the reduction of transform size gives additional advantage of reduced memory access time, which is demonstrated by comparing the computation times taken by the proposed and conventional methods on PC and DSP.

The rest of this paper is organized as follows. We summarize the notation and review the conventional OSA in Section 2. The proposed algorithm is presented in Section 3, and computational complexity analysis and the results of implementation on DSP and PC are shown in Section 4. Finally, conclusions are given in Section 5 and issue for and efficient implementation of QDFT is discussed in Section 6.

2. REVIEW OF OSA

We first summarize notations and definitions used in the rest of paper, and then review the conventional OSA.

2.1. Summary of Notations and Definitions

L : Filter length
M : Number of input data at each step in OSA, $M \geq L$
N $= 2M$, the size of overall block in OSA
s_n : Input sequence
h_n : Filter coefficients ($n = 0, 1, \cdots, L-1$)
r_n : Output sequence (Result of linear convolution $s_n * h_n$)
$W_N = e^{-j\frac{2\pi}{N}}$
$\mathbf{x}_p$: Previous data vector of size M in OSA
$\mathbf{x}_c$: Current data vector of size M in OSA
$\mathbf{x}$ $= [x_0, x_1, \cdots, x_{N-1}]^T = [\mathbf{x}_p^T : \mathbf{x}_c^T]^T$
$\mathbf{f}$ $= [h_0, h_1, \cdots, h_{L-1}, 0, \cdots, 0]^T$
(Size = M. Zeros are padded when $M > L$)
$\mathbf{0}_M$: zero vector of size M
$\mathbf{g}$ $= [g_0, g_1, \cdots, g_{N-1}]^T = [\mathbf{f}^T : \mathbf{0}_M^T]^T$

$\mathbf{y}$ = $[y_0, y_1, \cdots, y_{N-1}]^T$: temporary output vector (circular convolution of $\mathbf{g}$ and $\mathbf{x}$)
$\mathbf{D}_K$: DFT matrix of size K ((n,k)-th element is W_K^{nk})
$\mathbf{\Theta}_K$: odd-DFT (ODFT) matrix of size K ((n,k)-th element is $W_K^{n(k+1/2)}$)
$\mathbf{Q}_K$: quarter-DFT (QDFT) matrix of size K ((n,k)-th element is $W_K^{n(k+3/4)}$)
$\mathbf{X}^d = (\mathbf{D}_N)\mathbf{x}$, $\mathbf{X}^\theta = (\mathbf{\Theta}_N)\mathbf{x}$, $\mathbf{X}^q = (\mathbf{Q}_N)\mathbf{x}$
$\mathbf{G}^d = (\mathbf{D}_N)\mathbf{g}$, $\mathbf{G}^\theta = (\mathbf{\Theta}_N)\mathbf{g}$, $\mathbf{G}^q = (\mathbf{Q}_N)\mathbf{g}$
$\mathbf{F}^d = (\mathbf{D}_M)\mathbf{f}$, $\mathbf{F}^\theta = (\mathbf{\Theta}_M)\mathbf{f}$, $\mathbf{F}^q = (\mathbf{Q}_M)\mathbf{f}$
$\mathbf{X}_p^d = (\mathbf{D}_M)\mathbf{x}_p$, $\mathbf{X}_p^\theta = (\mathbf{\Theta}_M)\mathbf{x}_p$, $\mathbf{X}_p^q = (\mathbf{Q}_M)\mathbf{x}_p$
$\mathbf{X}_c^d = (\mathbf{D}_M)\mathbf{x}_c$, $\mathbf{X}_c^\theta = (\mathbf{\Theta}_M)\mathbf{x}_c$ $\mathbf{X}_c^q = (\mathbf{Q}_M)\mathbf{x}_c$
$\mathbf{X}_e, \mathbf{X}_o$: vectors of even and odd indexed elements of $\mathbf{X}^d$
$\mathbf{G}_e, \mathbf{G}_o$: vectors of even and odd indexed elements of $\mathbf{G}^d$

2.2. Conventional OSA

Using above notations, the conventional OSA can be summarized as follows:

Prepare $\mathbf{G}^d$. Copy $\mathbf{x}_c$ into $\mathbf{x}_p$, and get a block of new M data $\{s_n, s_{n+1}, \cdots, s_{n+M-1}\}$ into $\mathbf{x}_c$. Compute $\mathbf{X}^d$, and let $(\mathbf{X}^d \otimes \mathbf{G}^d)$ be $\mathbf{Y}$, where $\otimes$ implies element-wise multiplication. Perform IDFT on $\mathbf{Y}$ to get $\mathbf{y}$, discard the first half of data and save the last half. The saved data is the desired output $\{r_n, \cdots, r_{n+M-1}\}$. Above process is repeated for the next data block (except for the preparation of $\mathbf{G^d}$).

3. PROPOSED ALGORITHM

The main idea of proposed algorithm is to separate the computations for previous and current block in OSA. Then, we exploit the computations that were already performed for the previous block, when computing the convolution for the current block. This is based on the observation of decimation-in-frequency FFT algorithm, where the DFT $\mathbf{X}^d$ is split into shorter transforms as

$$\begin{bmatrix} \mathbf{X}_e \\ \mathbf{X}_o \end{bmatrix} = \begin{bmatrix} \mathbf{D}_M & \mathbf{D}_M \\ \mathbf{\Theta}_M & -\mathbf{\Theta}_M \end{bmatrix} \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_c \end{bmatrix}. \quad (1)$$

From this, the OSA can also be split into the computation of even and odd indexed terms. That is, splitting $\mathbf{G}^d$ into $\mathbf{G}_e$ and $\mathbf{G}_o$, spectral multiplication with $\mathbf{X}_e$ and $\mathbf{X}_o$, followed by inverse transform gives the circular convolution result, which is denoted as

$$\mathbf{y} = \begin{bmatrix} \mathbf{D}_M^{-1} & \mathbf{\Theta}_M^{-1} \\ \mathbf{D}_M^{-1} & -\mathbf{\Theta}_M^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{G}_e \otimes \mathbf{X}_e \\ \mathbf{G}_o \otimes \mathbf{X}_o \end{bmatrix}. \quad (2)$$

But, since first half of $\mathbf{g}$ is $\mathbf{f}$ and the rest is $\mathbf{0}_M$, we have

$$\begin{bmatrix} \mathbf{G}_e \\ \mathbf{G}_o \end{bmatrix} = \begin{bmatrix} \mathbf{D}_M & \mathbf{D}_M \\ \mathbf{\Theta}_M & -\mathbf{\Theta}_M \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{0}_M \end{bmatrix} = \begin{bmatrix} \mathbf{F}^d \\ \mathbf{F}^\theta \end{bmatrix}. \quad (3)$$

Also from (1) and the definitions, it can be seen that $\mathbf{X}_e = \mathbf{X}_p^d + \mathbf{X}_c^d$ and $\mathbf{X}_o = \mathbf{X}_p^\theta - \mathbf{X}_c^\theta$. Hence, the circular convolution in (2) can be rewritten as

$$\mathbf{y} = \begin{bmatrix} \mathbf{D}_M^{-1} & \mathbf{\Theta}_M^{-1} \\ \mathbf{D}_M^{-1} & -\mathbf{\Theta}_M^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{F}^d \otimes (\mathbf{X}_p^d + \mathbf{X}_c^d) \\ \mathbf{F}^\theta \otimes (\mathbf{X}_p^\theta - \mathbf{X}_c^\theta) \end{bmatrix} \quad (4)$$

which can be separated into the computations of previous and current blocks as

$$\begin{aligned} \mathbf{y} = &\begin{bmatrix} \mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_p^d) + \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_p^\theta) \\ \mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_p^d) - \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_p^\theta) \end{bmatrix} + \\ &\begin{bmatrix} \mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_c^d) - \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_c^\theta) \\ \mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_c^d) + \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_c^\theta) \end{bmatrix}. \end{aligned} \quad (5)$$

In the computation for the current block (the second term in the above equation), it seems that two kinds of convolution are needed, namely $\mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_c^d)$ and $\mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_c^\theta)$. For the convenience in notations and analysis, let us consider $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d)$ and $\mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$ instead of the above M-point transforms, without loss of generality. In the time domain, $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d)$ corresponds to the standard circular convolution as

$$y_n = \sum_{k=0}^{n} x_k g_{n-k} + \sum_{k=n+1}^{N-1} x_k g_{n-k+N}. \quad (6)$$

and $\mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$ corresponds to the skew circular convolution as

$$y_n = \sum_{k=0}^{n} x_k g_{n-k} - \sum_{k=n+1}^{N-1} x_k g_{n-k+N}. \quad (7)$$

Hence, the addition of the two terms $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d) + \mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$ corresponds to time domain computation

$$y_n = \sum_{k=0}^{n} x_k g_{n-k}, \quad (8)$$

and the subtraction $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d) - \mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$ to

$$y_n = \sum_{k=n+1}^{N-1} x_k g_{n-k+N}. \quad (9)$$

It is found that these two terms can actually be obtained from a single transform domain processing "QDFT." As defined in section 2, the input-output relationship of N-point QDFT is

$$X_k^q = \sum_{n=0}^{N-1} x_n W_N^{n(k+\frac{3}{4})}. \quad (10)$$

It can be easily shown that the spectral multiplication and inverse transform in the QDFT domain, *i.e.,* $\mathbf{Q}_N^{-1}(\mathbf{G}^q \otimes \mathbf{X}^q)$ corresponds to the time domain computation given by

$$y_n = \sum_{k=0}^{n} x_k g_{n-k} + j \sum_{k=n+1}^{N-1} x_k g_{n-k+N}. \qquad (11)$$

Comparing (8) and (9) with (11), we can see that the real part of $\mathbf{Q}_N^{-1}(\mathbf{G}^q \otimes \mathbf{X}^q)$ is equivalent to $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d) + \mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$ and the imaginary part to $\mathbf{D}_N^{-1}(\mathbf{G}^d \otimes \mathbf{X}^d) - \mathbf{\Theta}_N^{-1}(\mathbf{G}^\theta \otimes \mathbf{X}^\theta)$. In the same manner, it can be shown that $\mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_c^d) + \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_c^\theta)$ in (5) can be replaced by the real part of $\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_c^q)$ and $\mathbf{D}_M^{-1}(\mathbf{F}^d \otimes \mathbf{X}_c^d) - \mathbf{\Theta}_M^{-1}(\mathbf{F}^\theta \otimes \mathbf{X}_c^\theta)$ by the imaginary part. In summary, (5) is equivalent to

$$\begin{aligned} \mathbf{y} = & \begin{bmatrix} \Re\{\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_p^q)\} \\ \Im\{\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_p^q)\} \end{bmatrix} \\ & + \begin{bmatrix} \Im\{\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_c^q)\} \\ \Re\{\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_c^q)\} \end{bmatrix}. \end{aligned} \qquad (12)$$

Hence, we need to compute only the second term (QDFT, spectral multiplication, and IQDFT of current data block $\mathbf{x}_c$) at each step. For describing the proposed algorithm with the above notations, let us define new vectors of size M : $\mathbf{v}$ that stores $\Im\{\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_p^q)\}$, and $\mathbf{z}_r$ and $\mathbf{z}_i$ that store the real and imaginary part of $\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_c^q)$ respectively. Then the proposed algorithm can be summarized as follows :
Prepare $\mathbf{F}^q$. Store $\mathbf{z}_i$ into $\mathbf{v}$. Get a new block of input data into $\mathbf{x}_c$. Compute $\mathbf{X}_c^q$. Compute $\mathbf{Q}_M^{-1}(\mathbf{F}^q \otimes \mathbf{X}_c^q)$ and store the real part of this result into $\mathbf{z}_r$ and the imaginary part to $\mathbf{z}_i$. Then $\mathbf{v} + \mathbf{z}_r$ is the linear convolution result for the current input data. Repeat above process for the next input block.

4. COMPLEXITY ANALYSIS

All the transforms in this paper are implemented based on the radix-4 FFT which is widely used in DSP and PC library[8, 9]. And M is assumed to be power of 4 for the convenience of complexity analysis. The computational complexity of radix-4 FFT can be summarized as

$\mu_M^c = \frac{3}{8} M \log_2 M$
$\alpha_M^c = M \log_2 M$
$\mu_{2M}^r = \frac{3}{8} M \log_2 M + 2M$
$\alpha_{2M}^r = M \log_2 M + M$

where μ_n and α_n mean the number of multiplications and additions for the n-point transform respectively, and the superscript c and r mean the complex and real data respectively. It is noted that $2M$ point real FFT can be calculated efficiently from the computation of one M point complex FFT with additional $2M$ additions by simple data manipulation [8]. In the case of conventional OSA, the $2M$-point FFT is performed on the real data. Then complex spectral multiplication is performed, and IDFT on the complex data is performed. Therefore, the total number of multiplications for the conventional OSA is $\mu_{2M}^r + M + 1 + \mu_{2M}^r$, and the number of addition is $\alpha_{2M}^r + \alpha_{2M}^r$.

Table 1. Computational complexity of block convolution with DFT and QDFT

	MULT	ADD
DFT	$\frac{3}{4} M \log_2 M + 5M + 1$	$2M \log_2 M + 2M$
QDFT	$\frac{3}{4} M \log_2 M + 3M$	$2M \log_2 M$

Table 2. Number of arithmetic operations (sum of mults + adds)

M	DFT	QDFT	savings(%)
512	16257	14208	12.6
1024	35329	31232	11.6
2048	76289	68096	10.7
4096	163841	147456	10.0

From the input-output relationship of QDFT in eq. (10), it can be seen that the QDFT can be implemented by twiddle factor multiplication to the input (multiplication of $W_N^{\frac{3}{4}n}$ to the n-th input), followed by standard DFT. Hence, the computational complexity for the block convolution with the QDFT is : mult = $M + \mu_M^c + M + \mu_M^c + M$, and add = $\alpha_M^c + \alpha_M^c + M$, where the last M of add is for the real addition with the result of previous block. These results are summarized in Table 1 with respect to the number of input data M. Also for several lengths, specific numbers of arithmetic operations are compared in Table 2.

It is evident that the advantage of the proposed algorithm diminishes as the convolution length increases, because both algorithms require $O(M \log_2 M)$ complexity and the difference is just proportional to M. However, the long convolution is usually split into smaller ones in order to reduce the output delay which is inevitable in block processing[5, 6]. In other words, the long convolution is usually implemented by many of smaller length convolutions. In these practical implementations, replacing the conventional OSA by the proposed one would greatly reduce the computation times.

The conventional and proposed algorithms are implemented on two platforms - general PC equipped with Intel Pentium 4 processor 3 GHz and Analog Devices BLACKFIN 533 DSP board. For PC implementation of Radix-4 FFT, the state of art Intel IPP (Integrated Performance Primitives) library[9] is exploited. FFT in technical library provided by analog devices is used for DSP simulation. Convolutions are performed for 100 different input data sequences, and the averaged elapsed times are compared. Table 3 and Table 4 show the average elapsed times on PC and DSP respectively.

The savings of actual elapsed time in Table 3 are similar in tendency to those in Table 2 but larger than expected. This shows that the savings come not only from the reduced num-

ber of arithmetic (Table 2) but also from the reduced transform length ($N(=2M)$ vs. M) which results in reduced data access time when manipulating data. On the other hands, Table 4 does not have similar tendency to the expected result (Table 2). Generally DSP does not support sufficient hardware resources such as cache and memory compared with PC, and thus the time taken by memory access in DSP is more influential to the overall performance than in the PC. Hence the savings are getting larger as data size M increases. However, we can see that there are abrupt increase in savings when $M = 1024$ and abrupt decrease when $M = 2048$. In the case of abrupt increase when $M = 1024$, severe increase of elapsed time in the conventional method occurs due to the limit of cache size and thus it causes the increase in savings. On the other hand there is severe increase in the proposed method when $M = 2048$ because proposed method can perform 2 times larger OSA under the same cache size than the conventional method. Thus abrupt increase of elapsed time in the proposed method results in decrease in savings when $M = 2048$.

Table 3. Elapsed time of conventional OSA and proposed OSA on PC (μs)

M	conventional	proposed	savings(%)
512	14.69	10.89	25.87
1024	32.72	24.99	23.62
2048	67.40	54.30	19.44
4096	139.24	117.52	15.60

Table 4. Elapsed time of conventional OSA and proposed OSA on DSP (μs)

M	conventional	proposed	savings(%)
512	43.39	37.49	13.61
1024	121.49	83.12	31.58
2048	255.80	209.32	18.17
4096	613.51	423.84	30.92

5. CONCLUSIONS

We have proposed a new OSA that needs half size FFT compared to he conventional OSA. Since the FFT is the main process in the OSA, the proposed algorithm requires less computations and memory access time than the conventional method. The computations for the previous and current blocks in OSA are completely separated, and each block is efficiently computed by QDFT. The real part of QDFT domain convolution for current block plus the imaginary part of previous is shown to be the linear convolution result for the current block.

6. ACKNOWLEDGEMENT

This study is accomplished as a fundamental research project (UD080015FD) of Defence Acquisition Program Administration (DAPA) and Agency for Defence Development (ADD).

7. REFERENCES

[1] A. V. Oppenhiem, R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.

[2] R. C. Agarwal, C. S. Burrus, "Number theoretic transforms to implement fast digital convolution," *Proceedings of IEEE*, Vol. 63, No. 4, Apr. 1978, pp. 550 - 560.

[3] Z.-J. Mou, P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Processing*m Vol. 39, No. 6, June 1991, pp. 1322 - 1332.

[4] M. Vetterli, "Runni ng FIR and IIR filtering using multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 36, No. 5, May 1988, pp. 730 -738.

[5] W. G. Gardner, "Efficient convolution without input-output delay," *Journal of Audio Engineering Society*, Vol. 43, No. 3, March 1995, pp. 127 - 136.

[6] A. Torger, A. Farina, "Real-time partitioned convolution for ambiophonics surround sound," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 21-24 October 2001, New Paltz, New York.

[7] J. J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Magazine*, Jan. 1992, pp. 14 - 37.

[8] R. Matusiak, "Implementing fast fourier transform algorithms of real-valued sequences with the TMS320 DSP family" *Application Report of Texas Instruments*, 1997

[9] *Intel Performance Libraries. Intel Integrated Performance Primitives website.* http://www3.intel.com/cd/software/products/asmona/eng/perflib/302910.htm