Recursive Structure for Linear Filtering using Number Theoretic Transform

M. Bhattacharya* and J. Astola*

Abstract – A recursive structure for linear filtering is feasible using number theoretic transform (NTT) for certain choices of modulus. The structure is completely different from the typical FFT structure, although the lengths considered are square numbers and quite composite. Advantages gained are manifold. To mention few, there are no data index management in intermediate stages or twiddle factor multiplications and complex multiplications unlike in typical FFT structure, low multiplication per point compared to FIR filtering, and low power consumption, while some marginal complexity due in modulo arithmetic remains.

1 Introduction

In [14] an excellent exposition was given for realization of recursive digital filters in inputting data in bit serial and bit parallel fashion and use of look up tables (LUT) i.e., accessing preprogrammed (that relates to the coefficients of the filter) output depending on the input bit pattern (i.e., input data stream) was made use of; eliminating the need of multipliers. These realizations call for storing of the finite number of possible outputs of an intermediate arithmetic operation, and using them to obtain the next output sample through repeated addition and shifting operations; no multiplications were needed. The bit serial realization, while being slower, consumed less power, and was of minimal complexity hardware (e.g. requirement of LUTs or ROMs are less). On the other hand, the data throughput was higher in bit parallel realization, but consumed very high power and hardware complexity was much higher and required many ROMs. Here, we make an approach in using such ideas in computing number theoretic transform (NTT).

In general, use of (NTT) [1-3, 5-7, 9-13, 15, 16] for implementation of filtering through transform domain has been attractive always due to its exactness i.e., free of round off errors compared to other methods and implementation by simple and real arithmetic for real sequences. However, it is associated with the stringent relation of convolution length N with the choice of modulus M. Efforts to alleviate this problem is generally associated with increase in computational effort in terms of computational structure. For example, real multiplication is necessary in place of bit shifts being used in Fermat Number Transform (FNT) or Mersenne Number Transform (MNT), or more complex modulo operation due to choice of modulus other than the type $2^{2n+1}$, n an integer, or computation of many transforms modulo few moduli (supporting the same convolution length) followed by combining with Chinese Remainder Theorem (CRT) [1-3, 5-7, 9-13, 15, 16].

Implementation methods of convolution modulo various classes of primes, like FNT, MNT, pseudo-FNT, pseudo-MNT, primes of large class of prime of type $k2^n+1$ [17], submultiple primes of non-primes, etc., are widely found in literature [1-3, 5-7, 9-13, 15, 16].

Here, we consider the large class of primes of type $k2^n+1$ [17]; choice of modulus M being taken as $k2^n+1$, a prime, the maximum convolution length $N_{max}$ is limited to $k2^n$.

Such a choice of M leads to a wide choice of wordlength, with each wordlength associated with fairly large choices of convolution length due to high degree of compositeness of k $2^n$. Under such choice of M, there will be many many transform lengths (either $N_{max}$ or submultiples there of) that are perfect squares.

In this paper, we highlight and emphasize the implementation of these transform lengths and bring out the advantages and benefits of the structure for applications in linear filtering through transform domain.

In Table 1 some choices of modulus M are listed; it also shows the range of wordlength feasible for a given k. Table 2 shows some choices of transform lengths N those are square, showing the factor terms. As $\alpha$, an element of order of transform length N is not simple along with the fact that M is not simple unlike in the case of FNT, MNT, or pseudo-FNT/MNT, the realization will involve some computational effort, requiring real arithmetic modulo M.

Reduction of computational efforts is achieved by employing bit serial mechanization that is described in
The bit serial mechanization is also associated with a recursive structure that is possible when the transform length is a perfect square as in Table 2.

### 2 Implementation

In chirp filtering approach [4] the DFT of a sequence of length \( N \) can be expressed as

\[
X_k = \alpha^{k/2} \sum_{n=0}^{N-1} x_n \alpha^{-n/2} \alpha^{-(k-n)/2},
\]

with \( \alpha^N = 1 \) i.e.,

\( \alpha \) is the Nth root of unity (is an element of order N in the ring of integers under modulus \( M \) in number theoretic domain), and

\( X_k \) is the output for the period \( N \leq n \leq 2N-1 \), when an input sequence \( \{x_n \} \) defined only for the period \( 0 \leq n \leq N-1 \), is passed through a filter of impulse response \( \alpha^{-n/2} \) (defined only for a period \( 0 \leq n \leq 2N-1 \)) and the output of the filter is multiplied by \( \alpha^{k/2} \).

It was further shown that when the transform length is a perfect square, the z-transform of the filter can be expressed as,

\[
H(z) = (1 - z^{-2N}) \sum_{t=0}^{L-1} \alpha^{t/2} z^{-t} \frac{1 + \alpha^{dt} z^{-t}}{1 + \alpha^{dt}} \tag{2}
\]

where, \( N=L^2 \) and \( t = 0, 1, 2, \ldots L-1 \).

From (1) and (2) it is clear that convolution can be implemented by a bank of 2L recursive filters as shown in Fig. 1. The block to implement the \((1 - z^{-2N})\) is not required as input sequence is from zero to \( N-1 \) period and the output sequence is from \( N \) to \( 2N-1 \) period.

In [13] an approach similar to this was made using modulus as Fermat/Mersenne and pseudo-Fermat/Mersenne number, by choosing \( \sqrt{\alpha} \) as power of 2 so that multiplications can be performed by word shifts and addition operations. This resulted in reduction of transform length by half; also, the choice of modulus was highly limited to a few only, unlike here. However, under under choice of this class of \( M \), there will be a requirement of \( 4\sqrt{N} + 3 \) multiplier stages. Such a requirement is avoided by bit serial realization.

### Table 1. Some choices of modulus \( M \) of the type \( k.2^n+1 \), a prime.

<table>
<thead>
<tr>
<th>( k )</th>
<th>( n )</th>
<th>Range of wordlengths in bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8, 12, 18, 30, 36, 41, 66</td>
<td>9-67</td>
</tr>
<tr>
<td>5</td>
<td>13, 15, 25, 39, 55, 75, 85</td>
<td>15-87</td>
</tr>
<tr>
<td>7</td>
<td>14, 20, 26, 50, 52, 92</td>
<td>16-94</td>
</tr>
<tr>
<td>9</td>
<td>11, 14, 17, 33, 42, 43, 63, 65, 67, 81</td>
<td>14-84</td>
</tr>
<tr>
<td>15</td>
<td>9, 10, 12, 27, 37, 38, 44, 48, 78</td>
<td>12-81</td>
</tr>
<tr>
<td>25</td>
<td>10, 20, 22, 52, 64, 78</td>
<td>14-82</td>
</tr>
<tr>
<td>27</td>
<td>16, 19, 20, 22, 26, 40, 44, 46, 47, 50, 56, 59, 64, 92</td>
<td>20-96</td>
</tr>
<tr>
<td>45</td>
<td>9, 12, 14, 23, 24, 29, 60</td>
<td>14-65</td>
</tr>
<tr>
<td>49</td>
<td>10, 30, 42, 54, 66</td>
<td>15-71</td>
</tr>
<tr>
<td>63</td>
<td>9, 10, 14, 17, 18, 21, 25, 37, 38, 44, 60, 65, 94</td>
<td>14-99</td>
</tr>
<tr>
<td>75</td>
<td>10, 12, 34, 43, 51, 57, 60, 63, 67, 87</td>
<td>16-93</td>
</tr>
<tr>
<td>81</td>
<td>12, 15, 16, 21, 25, 27, 32, 35, 36, 39, 48, 89</td>
<td>25-101</td>
</tr>
<tr>
<td>99</td>
<td>10, 11, 22, 31, 33, 34, 41, 42, 53, 58, 65, 82</td>
<td>16-88</td>
</tr>
</tbody>
</table>

Table 2. Some choices of transform length.
From (2) the input and output relation for the t-th recursive stage, when the sequences are represented by B bits one obtains
\[
y_t(n) = a_t \sum_{j=0}^{B-1} x_j(n-t)2^j - b_t \sum_{j=0}^{B-1} y_j(n-L)2^j
\]
(3)
where \( a_t = \alpha^{-t^2/2} \) and \( b_t = \alpha^{-tL} \). We define a function \( F_t \) with two binary arguments as,
\[
F_t(x_j, y_j) = (a_x, x_j - b_y, y_j) \mod M
\]
(4)
From (3) and (4) we obtain,
\[
y_t(n) = \sum_{j=0}^{B-1} 2^j F_t(x_j(n-t), y_j(n-L))
\]
(5)
The function \( F_t \) can take only \( 2^2 = 4 \) distinct values depending on the values of its arguments and can be realized by read only memory (ROM). The implementation of the recursive stage only by shifting and additions modulo M, is illustrated in Fig. 2.

Data enters serially in the shift register SR1 with least significant bit (LSB) leading. At each shift a new address vector depending on bits of \( x_j(n-t) \) and \( y_j(n-L) \) addresses the ROM. The output \( F_t \) is loaded in the register R1 and fed to the Adder modulo M block. The other input to the adder is the output of the register R2 shifted right by one as shown. After B such shifts, the output of R2 register is the value of \( y_t(n) \) reduced modulo M, for the t-th recursive stage and the process for computing \( y_t(n+1) \) can be initiated.

![Figure 1. Recursive stage for computation of transform.](image)

The requirement of ROM is 4 words of B bits for each recursive stage. One transform stage will require 4L words. For \( N = 4096 \), the requirement is only 256 words leading to a requirement of 512 words considering the forward and inverse transform stages. A reduction is possible noting that when both arguments of \( F_t \) are zero, the value of \( F_t \) is zero and can be realized by a simple logic circuit to load the input line, to Adder from ROM, with zeros, thereby reducing the word storage requirement by twenty-five percent, i.e., 6L instead of 8L.

It is possible to reduce other hardware in the recursive stages. It is due to the fact that as all \( y_t(n) \)'s are summed before further processing, it is possible to combine more stages by having \( F_t \) function with more binary arguments. For example, if two recursive stages are combined, the function \( F_{12} \) will have six binary arguments requiring 64 words for these stages. While such approach would need more ROM, it will help in reducing other hardware such as Adder modulo M blocks, registers, shift registers and shift control circuits. Using such increased numbers of ROM, the throughput rate can be increased as the number of additions will be decreased proportionately.

As mentioned earlier in this section the stages are not operating continuously. The input multiplier is operative for period 0 to \( N-1 \) and the summer of \( y_t(n) \)'s are operating from period \( N \) to \( 2N-1 \). In such case two consecutive blocks of data can be processed and data throughput can be doubled with doubling the recursive stages and positioning switches to route the data path alternately for continuous operation.

![Figure 2. Bit serial implementation of the recursive stage.](image)

3 Concluding Remarks

The structure for linear filtering purpose using this bit serial implementation in number theoretic domain has many interesting features. From Table 2 we find that the
range of transform length covers almost entire range of convolution lengths for practical implementation and there exists many choices of wordlength.

We see that only three multiplications per point irrespective of the convolution length are required observing that the multiplication \( \alpha k^{2/2} H(k) \alpha n^{2/2} \) in between the forward and inverse transform stages can be implemented by one multiplier stage. Of course, the multipliers are slightly more complex as they are modulo M type, where M can be represented by a few bits.

In case of linear phase FIR filtering N/2 multiplications per output point (we consider N even for purpose of illustration) are required. Hence, one may state that these structures can be considered as substitute for FIR filtering.

Also, such implementations would be free of round off noise due to modulo integer arithmetic, and due to bit serial mechanization power consumption will be fairly low.

Compared to typical FFT type structure it is seen that no complex operations are required, no data index management (that involves quite a bit of complexity at each intermediate stage of FFT computation) along with no multiplications by twiddle factors at intermediate stages.

Another point to be noted is that while we have considered a large class of prime, the realization is also possible with other prime moduli (or submultiple prime of non-prime integer) where transform length is a perfect square. It is visualized that the structure mentioned in this paper will find wide practical application in fields of digital signal processing.

4 References