3-D FAST ALGORITHM FOR THE 3-D NEW MERSENNE NUMBER TRANSFORM

O. Alshibami and S. Boussakta
Institute of Integrated Information Systems
School of Electronic and Electrical Engineering
University of Leeds, Leeds, LS2 9JT, UK
E-mail: s.boussakta@ee.leeds.ac.uk

ABSTRACT

The New Mersenne Number Transform (NMNT) has been introduced in order to solve the problem of short transform lengths associated with other Mersenne number transforms (MNTs). In this paper, the three-dimensional NMNT and the 3-D radix-2×2×2 fast algorithm are introduced and discussed. The mathematical derivation of the new algorithm is presented and the number of arithmetic operations is calculated and compared to the row-column approach. Using single and multiple butterflies implementations, the radix-2×2×2 is found to reduce the number of arithmetic operations significantly.

1. INTRODUCTION

The calculation of the three-dimensional (3-D) and multidimensional (m-D) convolution and correlation functions involves a large number of arithmetic operations. Therefore, fast 3-D and m-D transforms are used for their calculations [1]. Among the m-D transforms, the Fermat and Mersenne numbers transforms are considered to be among the best candidates for error-free calculation of m-D convolution and correlation functions [2-5]. However, the main problem associated with their use is the short transform sizes [6]. Thus, the new Mersenne number transform (NMNT), was introduced to solve such problem [7].

The NMNT is defined modulo Mersenne number where arithmetic operations are simple (equivalent to 1's complement arithmetic) and can be used for fast calculation of convolutions, cross-, auto-correlations and related applications without the effects of rounding and/ or truncation errors. It has a large transform size power of two [4,7].

Unlike one and two-dimensional signal processing algorithms, most algorithms in three and higher dimensions are still undeveloped. Hence m-D transforms are usually calculated using algorithms developed for the 1-D case in row-column approach. Although this method has the advantage of using algorithms developed for the 1-D case, it is not efficient and when the transform is not separable as the case for the 3-D NMNT, extra arrays and arithmetic operations are needed.

Therefore, the 3-D radix-2×2×2 algorithm was introduced for the 3-D NMNT [8] and its arithmetic complexity was analysed for a single butterfly implementation. In this paper, a simpler derivation of this algorithm is introduced and the arithmetic complexity is analysed using multiple butterflies to remove trivial operations. The radix-2×2×2 algorithm has simple butterfly structure and can be implemented in-place. Compared to the 3-D row-column approach based on multiple butterflies also, the new algorithm is found to reduce the total number of arithmetic operations significantly.

2. THE THREE-DIMENSIONAL NMNT

The 3-D NMNT of $x(n_1, n_2, n_3)$ of dimensions $N_1 \times N_2 \times N_3$ is defined as [7]:

$$X(k_1,k_2,k_3) = \left\langle \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\sum_{n_3=0}^{N_3-1} x(n_1,n_2,n_3)\,\beta(n_1k_1,n_2k_2,n_3k_3) \right\rangle_{Mp} \quad (1)$$

$$k_i = 0,\ 1,\ 2, ..., N_i - 1; \quad i = 1,\ 2,\ 3$$

where

$$\beta(n_1k_1,n_2k_2,n_3k_3) = \beta_1(n_1k_1,n_2k_2,n_3k_3) + \beta_2(n_1k_1,n_2k_2,n_3k_3) \quad (2)$$

$$\beta_1(n_1k_1,n_2k_2,n_3k_3) = \left\langle \mathrm{Re}(\alpha_1 + j\alpha_2)^{n_1k_1+n_2k_2+n_3k_3} \right\rangle_{Mp} \quad (3)$$

and

$$\beta_2(n_1k_1,n_2k_2,n_3k_3) = \left\langle \mathrm{Im}(\alpha_1 + j\alpha_2)^{n_1k_1+n_2k_2+n_3k_3} \right\rangle_{Mp} \quad (4)$$

where $\langle \bullet \rangle_{Mp}$ means modulo the pth Mersenne number ($Mp = 2^p - 1$, with p an odd prime). To get the maximum transform sizes, Mp should be chosen to be prime of the form 2^p-1 where the maximum transform size is $N_{\max} \times N_{\max} \times N_{\max} = 2^p \times 2^p \times 2^p$. β_1 and β_2 in Eqs. (3) and (4), respectively, are for the transform size $2^{p+1} \times 2^{p+1} \times 2^{p+1}$. For transform size $\frac{2^{p+1}}{d} \times \frac{2^{p+1}}{d} \times \frac{2^{p+1}}{d}$, where d in integer power of two, β_1 and β_2 are given by:

$$\beta_1(n_1k_1,n_2k_2,n_3k_3) = \left\langle \mathrm{Re}\left((\alpha_1 + j\alpha_2)^d\right)^{n_1k_1+n_2k_2+n_3k_3} \right\rangle_{Mp} \quad (5)$$

and

$$\beta_2(n_1k_1,n_2k_2,n_3k_3) = \left\langle \mathrm{Im}\left((\alpha_1 + j\alpha_2)^d\right)^{n_1k_1+n_2k_2+n_3k_3} \right\rangle_{Mp} \quad (6)$$

where α_1 and α_2 are calculated by:

$$\alpha_1 = \pm\left\langle 2^q \right\rangle_{Mp}; \quad \alpha_2 = \pm\left\langle -3^q \right\rangle_{Mp}; \quad q = 2^{p-2} \quad (7)$$

The inverse transform is the same as the forward except for a scale factor $(N_1N_2N_3)^{-1}$ which is equal to 1 for maximum transform size. For other transform sizes, it can be split between the forward and the inverse transforms to make them exactly the same.

3. RADIX-2×2×2 ALGORITHM

The m-D NMNT is usually computed using the row-column approach by adding certain points of the m-D separable transform. The m-D separable transform is computed using fast 1-D NMNT algorithms applied over each dimension successively. The m-D NMNT is then computed from the separable transform at the expense of extra additions and shifts (multiplication by 2^{p-1}).

In this paper, a simple derivation of the previously presented 3-D radix-2×2×2 algorithm [8] is introduced for fast calculation of the 3-D NMNT.

3.1 Mathematical Development

The 3-D new Mersenne number transform defined in Eq. (1) has transform sizes equal to powers of 2×2×2 and hence can be computed using the radix-2×2×2 algorithm. In this algorithm, the three-dimensional new Mersenne number transform summations are decomposed into eight separate summations over the even and odd indices of $x(n_1,n_2,n_3)$ where the $N\times N\times N$-point 3-D NMNT computation is divided into eight $\frac{N}{2}\times\frac{N}{2}\times\frac{N}{2}$ 3-D NMNTs. In the next stage of the algorithm, each $\frac{N}{2}\times\frac{N}{2}\times\frac{N}{2}$-point 3-D NMNT is further divided into eight $\frac{N}{4}\times\frac{N}{4}\times\frac{N}{4}$-point 3-D NMNTs, and the process continues until we obtain 2×2×2 transforms [8]. Therefore, $X(k_1,k_2,k_3)$ in Eq. (1) is decomposed as:

$$\begin{aligned} X(k_1,k_2,k_3) = \Big\langle & X_{000}(k_1,k_2,k_3) + X_{001}(k_1,k_2,k_3) + X_{010}(k_1,k_2,k_3) \\ & + X_{011}(k_1,k_2,k_3) + X_{100}(k_1,k_2,k_3) + X_{101}(k_1,k_2,k_3) \\ & + X_{110}(k_1,k_2,k_3) + X_{111}(k_1,k_2,k_3) \Big\rangle_{Mp} \end{aligned} \tag{8}$$

where

$$\begin{aligned} X_{abc}(k_1,k_2,k_3) = \Big\langle & \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1} x(2n_1+a, 2n_2+b, 2n_3+c) \\ & \times \beta\big((2n_1+a)k_1 + (2n_2+b)k_2 + (2n_3+c)k_3\big) \Big\rangle_{Mp} \end{aligned} \tag{9}$$

$a,b,c = 0 \ \ or \ \ 1$

For $abc = 000$, $X_{000}(k_1,k_2,k_3)$ can be written as:

$$\begin{aligned} X_{000}(k_1,k_2,k_3) &= \Big\langle \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1} x(2n_1, 2n_2, 2n_3) \times \beta(2n_1k_1, 2n_2k_2, 2n_3k_3) \Big\rangle_{Mp} \\ &= D_{000}(k_1,k_2,k_3) \end{aligned} \tag{10}$$

where

$$\begin{aligned} D_{abc}(k_1,k_2,k_3) = \Big\langle & \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1} x(2n_1+a, 2n_2+b, 2n_3+c) \\ & \times \beta(2n_1k_1, 2n_2k_2, 2n_3k_3) \Big\rangle_{Mp} \end{aligned} \tag{11}$$

$a,b,c = 0 \ \ or \ \ 1$

$D_{000}(k_1,k_2,k_3)$ is a 3-D NMNT with size $\frac{N}{2}\times\frac{N}{2}\times\frac{N}{2}$. Similarly, $X_{001}(k_1,k_2,k_3)$ can be written as:

$$\begin{aligned} X_{001}(k_1,k_2,k_3) = \Big\langle & \sum_{n_1=0}^{N/2-1}\sum_{n_2=0}^{N/2-1}\sum_{n_3=0}^{N/2-1} x(2n_1, 2n_2, 2n_3+1) \\ & \times \beta\big(2n_1k_1, 2n_2k_2, (2n_3+1)k_3\big) \Big\rangle_{Mp} \end{aligned} \tag{12}$$

Using the identity:

$$\beta(m+n) = \beta(m)\beta_1(n) + \beta(-m)\beta_2(n) \tag{13}$$

$X_{001}(k_1,k_2,k_3)$ can be written as:

$$X_{001}(k_1,k_2,k_3) = \Big\langle D_{001}(k_1,k_2,k_3)\beta_1(k_3) + D_{001}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_3) \Big\rangle_{Mp} \tag{14}$$

Again $D_{001}(k_1,k_2,k_3)$ and $D_{001}(N/2\text{-}k_1,N/2\text{-}k_2,N/2\text{-}k_3)$ are 3-D NMNTs with size $\frac{N}{2}\times\frac{N}{2}\times\frac{N}{2}$. Following the same development, $X_{010}(k_1,k_2,k_3)$, $X_{011}(k_1,k_2,k_3)$, $X_{100}(k_1,k_2,k_3)$, $X_{101}(k_1,k_2,k_3)$, $X_{110}(k_1,k_2,k_3)$, and $X_{111}(k_1,k_2,k_3)$ can be developed as:

$$X_{010}(k_1,k_2,k_3) = \Big\langle D_{010}(k_1,k_2,k_3)\beta_1(k_2) + D_{010}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_2) \Big\rangle_{Mp} \tag{15}$$

$$X_{011}(k_1,k_2,k_3) = \Big\langle D_{011}(k_1,k_2,k_3)\beta_1(k_2,k_3) + D_{011}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_2,k_3) \Big\rangle_{Mp} \tag{16}$$

$$X_{100}(k_1,k_2,k_3) = \Big\langle D_{100}(k_1,k_2,k_3)\beta_1(k_1) + D_{100}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1) \Big\rangle_{Mp} \tag{17}$$

$$X_{101}(k_1,k_2,k_3) = \Big\langle D_{101}(k_1,k_2,k_3)\beta_1(k_1,k_3) + D_{101}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_3) \Big\rangle_{Mp} \tag{18}$$

$$X_{110}(k_1,k_2,k_3) = \Big\langle D_{110}(k_1,k_2,k_3)\beta_1(k_1,k_2) + D_{110}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_2) \Big\rangle_{Mp} \tag{19}$$

$$X_{111}(k_1,k_2,k_3) = \Big\langle D_{111}(k_1,k_2,k_3)\beta_1(k_1,k_2,k_3) + D_{111}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_2,k_3) \Big\rangle_{Mp} \tag{20}$$

Replacing Eqs. (10) and (14)-(20) into (8), leads to the general formula of the radix-2×2×2 algorithm for the 3-D NMNT:

$$\begin{aligned} X(k_1,k_2,k_3) = \Big\langle & D_{000}(k_1,k_2,k_3) \\ & + \left[D_{001}(k_1,k_2,k_3)\beta_1(k_3) + D_{001}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_3)\right] \\ & + \left[D_{010}(k_1,k_2,k_3)\beta_1(k_2) + D_{010}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_2)\right] \\ & + \left[D_{011}(k_1,k_2,k_3)\beta_1(k_2,k_3) + D_{011}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_2,k_3)\right] \\ & + \left[D_{100}(k_1,k_2,k_3)\beta_1(k_1) + D_{100}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1)\right] \\ & + \left[D_{101}(k_1,k_2,k_3)\beta_1(k_1,k_3) + D_{101}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_3)\right] \\ & + \left[D_{110}(k_1,k_2,k_3)\beta_1(k_1,k_2) + D_{110}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_2)\right] \\ & + \left[D_{111}(k_1,k_2,k_3)\beta_1(k_1,k_2,k_3) + D_{111}\left(\tfrac{N}{2}-k_1, \tfrac{N}{2}-k_2, \tfrac{N}{2}-k_3\right)\beta_2(k_1,k_2,k_3)\right] \Big\rangle_{Mp} \end{aligned} \tag{21}$$

Combining eight points together gives the 3-D radix-2×2×2 algorithm as:

$$\begin{bmatrix} X(k_1,k_2,k_3) \\ X(k_1,k_2,k_3+N/2) \\ X(k_1,k_2+N/2,k_3) \\ X(k_1,k_2+N/2,k_3+N/2) \\ X(k_1+N/2,k_2,k_3) \\ X(k_1+N/2,k_2,k_3+N/2) \\ X(k_1+N/2,k_2+N/2,k_3) \\ X(k_1+N/2,k_2+N/2,k_3+N/2) \end{bmatrix} = \left\langle \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} X_{000}(k_1,k_2,k_3) \\ X_{001}(k_1,k_2,k_3) \\ X_{010}(k_1,k_2,k_3) \\ X_{011}(k_1,k_2,k_3) \\ X_{100}(k_1,k_2,k_3) \\ X_{101}(k_1,k_2,k_3) \\ X_{110}(k_1,k_2,k_3) \\ X_{111}(k_1,k_2,k_3) \end{bmatrix} \right\rangle_{Mp} \tag{22}$$

3.2. Arithmetic Complexity

The relation for the radix-2×2×2 in Eq. (22) calculates eight points, which can be represented by single butterfly. For in-place calculation, two butterflies are needed to be combined as shown in Figure 1. The in-place butterfly calculates sixteen points and involves 28 real multiplications and 62 real additions. The 3-D transform needs $\log_2 N$ stages. Therefore, the calculation of the whole transform using one butterfly requires $\lceil \frac{7}{4}N^3 \log_2 N$ integer multiplications and $\lceil \frac{31}{8}N^3 \log_2 N$ integer additions.

The above calculation includes a large number of trivial multiplications and additions, which can be eliminated using multiple butterflies. If multiple butterflies are used, the total number of arithmetic operations will be reduced to: $\lceil \frac{7}{4}N^3 \log_2 N - 7N^3 + 14N^2$ integer multiplications, $\lceil \frac{31}{8}N^3 \log_2 N - \frac{21}{8}N^3 + \frac{7}{2}N^2$ integer additions, and $\lceil \frac{7}{4}N^3 - 7N^2 \rceil$ shifts.

On the other hand, if the 3-D NMNT is calculated using the row-column approach, it will involve $\lceil 3N^3 \log_2 N$ integer multiplications and $\lceil \frac{9}{2}N^3 \log_2 N + 3N^3$ integer additions for a single butterfly implementation and $\lceil 3N^3 \log_2 N - 12N^3 + 24N^2$ integer multiplications, $\lceil \left(\frac{9}{2}\right)N^3 \log_2 N - \left(\frac{3}{2}\right)N^3 + 6N^2$ integer additions, and $\lceil \frac{5}{2}N^3 - 6N^2$ shifts using multiple butterflies to remove trivial arithmetic operations

Figures 2, 3 and 4 show a comparison between the new algorithm and the row-column approach based on multiple butterflies implementations. It is obvious that the radix-2×2×2 algorithm offers a substantial saving in both total number of multiplications and additions.

4. CONCLUSION

In this paper, a simple derivation of the radix-2×2×2 algorithm has been introduced for fast calculation of the 3-D MNT. The number of arithmetic operations of the developed algorithm has been calculated and compared to the familiar row-column approach. It has been found that, the new algorithm offers substantial savings over the row-column approach in terms of multiplications, additions and shifts.

References

1. CHEN, T., March 1998. The past, present, and future of image and multidimensional signal processing. *IEEE Signal Processing magazine*, **15** (2), 21-58.
2. BECKER, R.I. AND MORRISON, N., August 1996. The errors in FFT estimation of the Fourier transform. *IEEE Trans. on Signal Processing*, SP-44 (8), 2073-2077.
3. PITAS, I. AND STRINTZIS, M.G., January 1988. Floating point error analysis of two-dimensional fast Fourier transform algorithms. *IEEE Trans. on Circuits and Systems*, vol. CAS-35 (1), 112-115.
4. BOUSSAKTA, S AND HOLT, A.G.J., 1999. Number theoretic transforms and their applications in image processing. *In: Advances in Imaging and Electron Physics.* Academic Press, 111, 1-90.
5. MORANDI, C., PIAZZA, F. AND DOLCETTI, A., May 1988. Image registration using Fermat transforms. *Electronics Letters*, 24 (11), 678-680C.
6. LU, H. AND LEE, S.C., June 1991. A new approach to solve the sequence-length constraint problem in circular convolution using number theoretic transform. *IEEE Trans. on Signal Processing*, SP-39 (6), 1314-1321.
7. BOUSSAKTA, S. AND HOLT, A.G.J., December 1995. New transform using the Mersenne numbers. *IEE Proc.-Vision, Image, and Signal Processing*, 142 (6), 381-388.
8. BOUSSAKTA, S., ALSHIBAMI, O., AZIZ, M., AND HOLT, A.G.J., APRIL 2001. 3-D vector radix algorithm for the 3-D new Mersenne number transform. *IEE Proc.-Vision, Image, and Signal Processing*, 148 (2), 115-125.

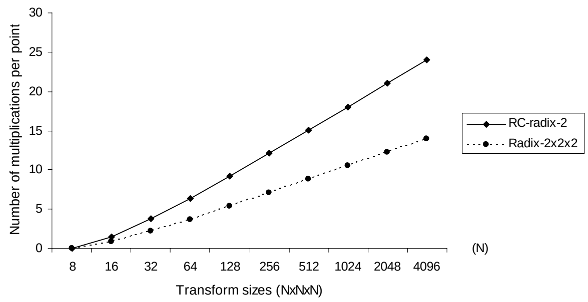

Figure 2. Comparison between the row-column approach and the radix-2×2×2 using multiple butterflies (number of multiplications/point).

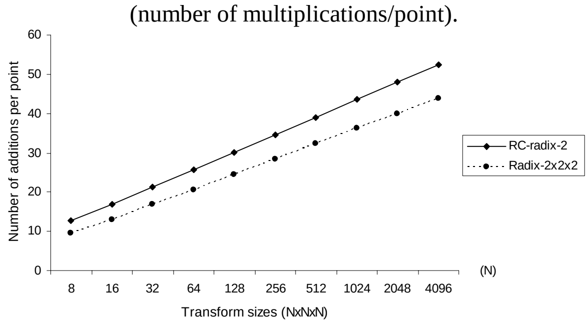

Figure 3. Comparison between the row-column approach and radix-2×2×2 algorithm using multiple butterflies (number of additions/point).

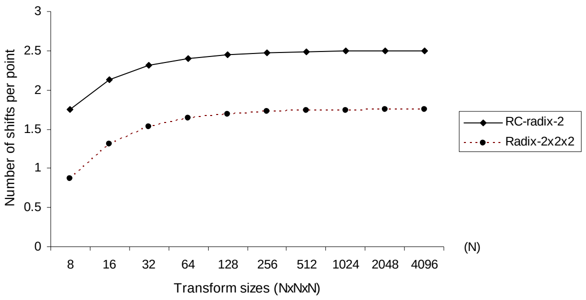

Figure 4. Comparison between the row-column approach and the radix-2×2×2 algorithm using multiple butterflies (number of shifts/point).

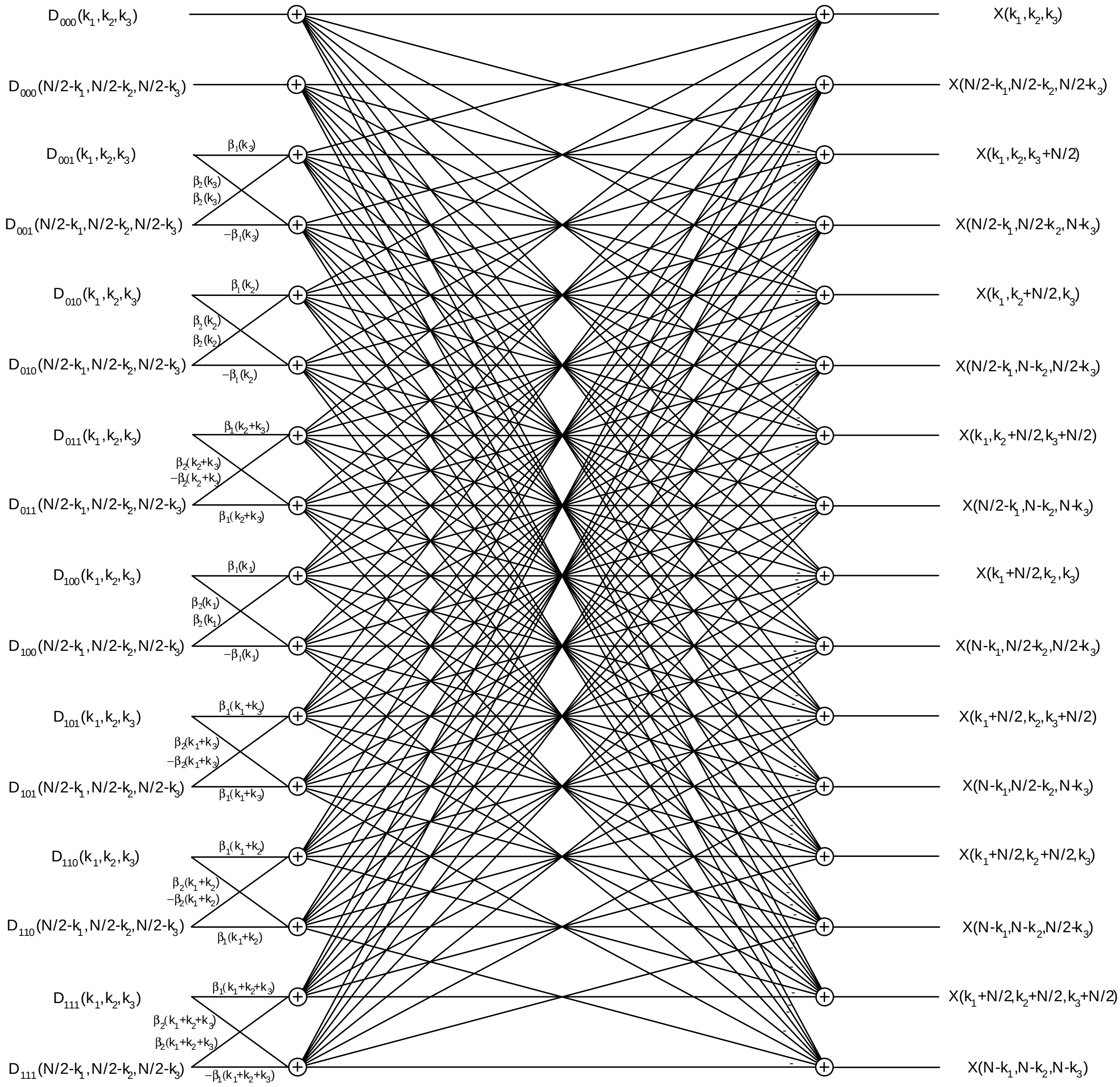

Figure 1. The in-place butterfly for the 3-D NMNT radix-2×2×2 algorithm.