inmos

discrete Fourier transform with the IMS A100

9.1 Introduction

In the time-domain representation, signals are expressed as a function of time. For example $x = Ae^{-at}$ is a time-domain description of a signal whose amplitude decays exponentially with time.

In the 18th century J.B. Fourier showed any signal that can be generated in a laboratory can be expressed as a sum of sinusoids of various frequencies. In other words each signal can be said to have a frequency spectrum represented by the amplitude and phases of various sinosoidal components. The frequency spectrum of a signal completely specifies it and is referred to as frequency-domain description of the signal.

The Fourier integrals provide the means for obtaining the frequency-domain representation (the spectrum) of a signal from its time-domain representation or vise-versa, i.e.

Fourier Transform:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt \tag{1}$$

Inverse Fourier Transform:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega \tag{2}$$

where $x(t)$ is a time-domain signal, $X(\omega)$ is the frequency spectrum of $x(t)$ and ω is the frequency variable.

These transforms are fundamental to the description of many real world phenomena in the fields of science and engineering. In the area of signal processing the Fourier transform is an important mathematical (and nowadays practical) tool in understanding, analysing and solving system-level problems. The Fourier transform allows us to translate time serial information into the frequency domain in a reversible way. The components of a signal, although dispersed in the time domain, may have restricted occupancy or a characteristic relationship in the frequency domain. In fact many physical processes can be categorised by the frequencies they generate and their relative strength. This is one reason why the Fourier transform, with its ability to segregate frequency components of a signal, has gained such an importance in many signal processing environments.

Apart from the ability of the Fourier transform to provide spectral information, many signal processing functions such as correlation, filtering and beamforming can be expressed in terms of the Fourier transform and its inverse. For these reasons considerable effort has gone into the development of efficient algorithms for evaluating the Fourier transform. The continuous-time Fourier transform, given by equation (1), can be made suitable for digital computation by sampling the time and frequency variables and limiting the computation to a finite set of data points. This modified version of the Fourier Transform is often referred to as the Discrete Fourier Transform (DFT) and will be discussed in more detail in the next section.

Many algorithms have been developed for efficient digital computation of the DFT. Until recently the digital multipliers needed to implement DFT's were costly, large and relatively slow, and the general purpose microprocessors were extremely slow at performing multiplications. Consequently it was necessary to calculate DFT's using a minimum number of multiplications and to use data and coefficient storage economically and this led to development of several Fast Fourier Transform (FFT) (references 1 & 2) algorithms. These FFT algorithms make use of the redundancies, that occur in the DFT, to reduce the arithmetic operations involved. Most FFT algorithms were designed simply to minimise the total number of multiplications required to calculate the DFT, often at the expense of an increase in the number of additions, memory accesses and control complexity. One such algorithm is the Cooley-Tuckey radix-2 FFT algorithm which necessitates a data size equal to a power of two ($N = 2^n$). Winogard FFT algorithms on the other hand requires that the number of data points to be prime. Both algorithms simply minimize the number of multiplications in the DFT by the use of redundancies resulting from the particular choice of the data size. These algorithms are particularly suitable for general-purpose computers and microprocessors where the major limit on processing speed is the time taken to perform the multiply instruction.

Other algorithms have been developed which map the DFT process into particular hardware structures. Two such techniques are the Rader's Prime Number Transform (PNT) (reference 3) and the Chirp-Z Transform (CZT) (reference 4) which convert the DFT into circular correlation/convolutions. These algorithms are particu-

larly suitable for implementation using transversal filter type structures. In the past, CCD and SAW transversal filters have been used to implement high-throughput wide-bandwidth DFT processors using these algorithms. The analogue nature of the CCD and SAW technologies has restricted the precision of these processors.

The availability of the IMS A100, the first high performance cascadable digital transversal filter, means that the same algorithms can now be implemented digitally offering both high speed and high accuracy. This application note deals with the concepts behind these algorithms and their implementations using the IMS A100 signal processor. Generalised mapping techniques which facilitate the DFT evaluation of a long data sequence via a number of short transforms are also discussed (The radix-2 FFT is a special case of these more general partitioning techniques). The approach described here is particularly suitable if a long DFT is to be evaluated with a transversal filter of limited size. Also, these decomposition methods are applicable to concurrent architectures and as such provide the basis for the trade-off between speed and cost involved in a particular implementation. These issues are of major importance when combining the IMS A100(s) with the INMOS transputer family of parallel processors.

Figure 9.1a shows the structure for an N-stage canonical transversal filter where the output is the weighted sum of the N most recent input samples. The IMS A100 implementation of the transversal filter is depicted in figure 9.1b where the multi-input summation of the canonical form has been replaced by a delay and add chain. You should be able to convince yourself that the two structures in figure 9.1 have the same functional behaviour. The main difference is that in figure 9.1b the partial product terms are passed down the delay-and-add chain whilst in figure 9.1a, the input samples are delayed and the sum of products is calculated simultaneously.

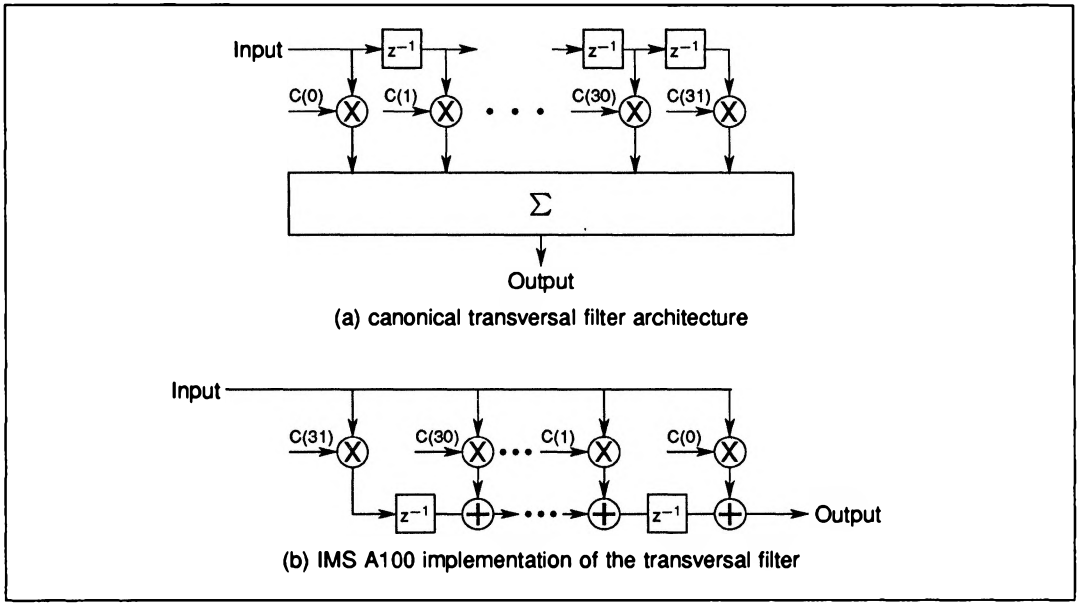

Figure 9.1 Transversal filter architecture

A simplified functional diagram for the IMS A100 is shown in figure 9.2. The major processing part of the chip incorporates 32 multipliers and a 32-stage delay-and-add chain. For the IMS A100 the input data word length in 16 bits. The coefficient word length can be programmed to be 4, 8, 12 or 16 bits. The data throughput ranges from 2.5 million samples/s to 10 million samples/s depending on the coefficient word size. Two complete sets of coefficient memories are provided. At any instant one set of coefficients is applied to the transversal filter, whilst the other set can be accessed via a standard memory interface (capable of 100ns cycle time). The function of the two coefficient memories can be exchanged by writing to control registers. Further this exchange can be made continuous, i.e alternate sets of coefficients can automatically be selected for successive computation cycles. This is particularly useful for complex number processing.

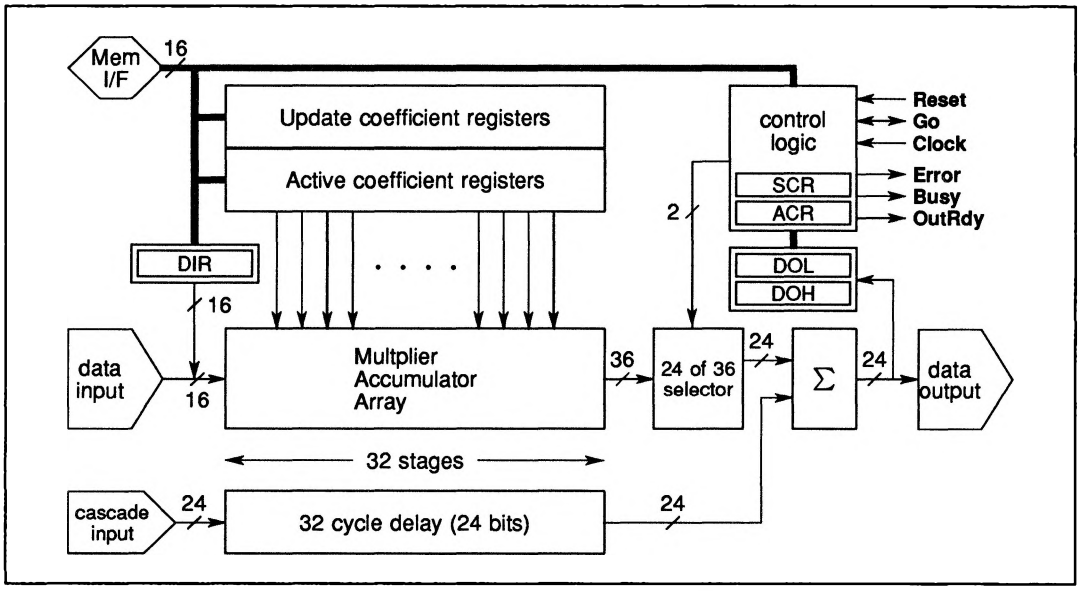

Figure 9.2 User's model of the IMS A100

Data input and output are available both through dedicated ports or via the memory interface. This selection can be programmed via the control and status registers in the IMS A100.

To preserve complete numerical accuracy, no truncation or rounding is performed on the partial products in the multiplications and delay-and-add chain. The output of the chain is calculated with a precision of 36 bits which is sufficient to ensure no overflow occurs (the only time that the output of the delay-and-add chains exceeds 36 bits is when all 32 coefficients and 32 successive input samples have the maximum possible negative value i.e. 1000000000000000 in two's complement binary notation, this is of course highly unlikely). A programmable barrel shifter is located at the output of this chain, which allows 24 bits (starting at bits 7, 11, 15 or 20 of the full 36 bit result) to be selected and rounded for output. To allow devices to be cascaded without any external components, a 32-stage 24-bit wide shift register and a 24-bit adder are included on the chip. For cascading purposes the output of one chip is connected directly to the cascade input of the next.

The control registers accessible via the memory interface allows various operational parameters to be programmed. For the full detail of the specification you are advised to refer to the IMS A100 data sheet.

In the following parts of this application note the basic concepts of DFT will be reviewed and some algorithms for its evaluation will be summarized. This is followed by a detailed description of those DFT algorithms suitable for implementation using the IMS A100 transversal filter. A multi-dimensional mapping technique is also described which allows efficient computations of long-length DFTs via the IMS A100 implementations.

9.2 The basic concepts of DFT

The equation for the Fourier transform and its inverse (equations 1 & 2) can be made suitable for digital processing by discretizing both the time variable t and the frequency variable ω; and by constraining the integration to finite limits. Referring to equation 1, for the forward transform, this can be done by making $t = nT$ and $\omega = k\omega_0$. Where T is the sampling period of the time function and ω_0 is the frequency resolution of the discrete spectrum. If the integration limits are confined over N time samples for which N independent frequency samples can be calculated we have

$$n = 0, 1, 2, 3, ..., N-1$$

$$k = 0, 1, 2, 3, ..., N-1$$

The Fourier-transform integral then becomes a Fourier-transform sum given by:

$$X(k\omega_0) = \sum_{n=0}^{N-1} x(nT)e^{-jk\omega_0 nT} \qquad k = 0, 1, ...N-1 \tag{3}$$

It can be shown that the frequency resolution in the f-domain is given by:

$$\omega_0 = \frac{2\pi}{NT} \tag{4}$$

Substituting this in (3) gives:

$$X(k\omega_0) = \sum_{n=0}^{N-1} x(nT)e^{-2\pi jkn/N} \qquad k = 0, 1, ...N-1 \tag{5}$$

For convenience the terms T and ω_0 are usually dropped from the indicies giving the DFT equation as:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} = \sum_{n=0}^{N-1} x(n)W^{nk}{}_N \qquad k = 0, 1, ...N-1 \tag{6}$$

where

$$W_N = e^{-2\pi j/N} = \cos(\frac{2\pi}{N}) - j\sin(\frac{2\pi}{N}) \tag{7}$$

The inverse discrete Fourier transform (IDFT) can be derived in a similar manner from its corresponding continuous form and is given by:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{2\pi jnk/N} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk} \qquad n = 0, 1, ...N-1 \tag{8}$$

Note that the DFT and its inverse, IDFT, are very similar, the only difference is the factor $\frac{1}{N}$ and the negative exponent in the IDFT. This similarity has important practical significance as it allows an algorithm or a hardware developed for DFT to be used for IDFT with minor modifications. For example an inverse DFT on the data sequence $x(0), x(1)....x(N-1)$ can be carried out by first reversing this sequence to generate a new data set $x'(\,)$ such that $x'(0) = x(N-1)$, $x'(1) = x(N-2)$.....$x'(N-1) = x(0)$ and then performing a DFT and dividing the result by N. This technique works simply because $x'(k) = x(-k)$, (In the DFT both $x(n)$ and $X(k)$ are assumed to be periodic) which converts the positive exponential in (8) into a negative one, representing a DFT. For this reason any algorithm or implementation in the following subsections will only be described for DFT as the extension to an IDFT is trivial.

It is worth noting that some authorities write the DFT and its inverse as:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} \qquad DFT$$

$$x(n) = \sum_{k=0}^{N-1} X(k)e^{2\pi jnk/N} \qquad IDFT$$

i.e. the factor $\frac{1}{N}$ is applied to the DFT rather than its inverse. This version can be seen to have a physical meaning since $X(0)$, as defined, represents the average of the sampled time waveform i.e. the 'd.c.' value. Other authorities express the DFT and its inverse as:

$$X(k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} \qquad DFT$$

$$x(n) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} X(k)e^{2\pi jnk/N} \qquad IDFT.$$

This last formulation is necessary if the power contents of the time-domain and frequency-domain signals are to be identical.

Throughout this application note, the definitions given by equations (6) and (8) are used for DFT and IDFT. However, the techniques described here are applicable to all three formulations of the DFT and its inverse.

9.3 Algorithms for efficient evaluation of DFT

From equation (6), it is apparent that the direct evaluation of the DFT is very much computation intensive. Assuming complex data, $x(n) = xr(n) + j\ xi(n)$, we have

$$X(k) = XR(k) + jXI(k) = \sum_{n=0}^{N-1}[xr(n) + jxi(n)](\cos(\frac{2\pi kn}{N}) - j\sin(\frac{2\pi kn}{N})$$

or

$$XR(k) = \sum_{n=0}^{N-1} xr(n)\cos(\frac{2\pi nk}{N}) + xi(n)\sin(\frac{2\pi nk}{N}) \tag{9a}$$

and

$$XI(k) = \sum_{n=0}^{N-1} xi(n)\cos(\frac{2\pi nk}{N}) - xi(n)\sin(\frac{2\pi nk}{N}) \tag{9b}$$

where $k = 0, 1, 2.....N-1$, $XR(k)$ and $XI(k)$ are the real and imaginary parts of the spectrum respectively.

From equation (9) it can be deduced that the direct evaluation of DFT involves $4N^2$ multiplications and approximately $4N^2$ additions. In these estimates the computations involved in the evaluation of the trigonometric functions, (sin and cos) have been ignored as it is possible to precalculate the trigonometric values in a look-up table and use them appropriately. Historically, multiplications were very slow compared to other operations, therefore algorithms were developed to minimize the number of required multiplications at the expense of other operations. These algorithms made use of the cyclic nature of the exponential $\exp(\frac{-2\pi jnk}{N})$ to reduce the number of multiplications involved.

To demonstrate some of the resulting redundancies, consider the case where N is even. It is fairly straightforward to show that for this case

$$W_{\frac{N}{2}}^{k} = W_N^{2k} \tag{10}$$

and

$$W_N^k = -W_N^{k+\frac{N}{2}} \tag{11}$$

One algorithm which uses these types of redundancies is the Cooley-Tuckey radix-2 FFT (reference 1). This algorithm requires the number of data points to be equal to a power of two, ie $N = 2^m$ where m is an integer. Using the identities given by (10) and (11) the algorithm expresses the N-point DFT in terms of two $\frac{N}{2}$-point DFT's. Then the $\frac{N}{2}$-point DFT's are expressed as two $\frac{N}{4}$-point transforms using identities similar to (10) and (11). This decomposition is carried out until all the DFT's involved are only two-point transforms. The net result of this decomposition is a considerable reduction in the number of multiplications. In fact it can be shown that for the Cooley-Tuckey radix-2 FFT algorithm the number of multiplications involved is approximately $2N\log_2(N)$ and the number of additions is $3N\log_2(N)$. Compared to the $4N^2$ operations involved in the direct DFT calculation, for a large N, the radix-2 FFT algorithm reduces the number of multiplications and additions considerably.

Another algorithm which also minimises the number of multiplications is Winogard's prime-length transform (reference 2). This algorithm is applicable to cases where the data size is a prime number. In practice this algorithm is used only for short-length transforms and mapping techniques are used to extend it to large data sizes (references 5 & 6).

The argument behind the efficiency of these algorithms is only valid if the multiplication time is longer than other operations such as indexing and memory accesses. This is indeed the case for most general-purpose processors.

Todays digital technology is capable of providing extremely powerful processing engines which mean that the minimization of the number of multiplications is not always the best approach. For high performance systems, other issues such as the memory bandwidth, architecture efficiency and parallelism potential have to be seriously considered. The advances in digital technology allow other algorithms particularly those which map the DFT onto special VLSI hardware structures to be exploited. The following sections deal with algorithms that map the DFT into correlation/convolutions, ideal for implementation using the IMS A100 transversal filter. This algorithms make use of the higher level functional nature of the device and its on-chip

memory to minimise the required host's memory bandwidth. For this reason the combination of a medium-speed microprocessor and the IMS A100 device(s) results in a very high performance system capable of competing with bit-slice DSP processors.

9.4 DFT algorithms suitable for the IMS A100 implementation

There are basically two algorithms which map the DFT into a correlation (convolution) process. These are

(i) the Prime Number Transform (PNT)

(ii) the Chirp-Z-Transform (CZT)

The PNT was developed by Radar and is applicable when the number of data points is prime. The CZT on the other hand is applicable to any data size; it can, however, be simplified if the data size is an even number. The following two sections deal with each one of these algorithms and their implementations using the IMS A100 transversal filter. The final part of this application note describes mapping techniques which allow the DFT of a large number of data points to be evaluated via a number of short transforms. This mapping technique is of vital practical significance when implemanting PNT & CZT processors.

9.4.1 Rader's Prime Number Transform

The PNT algorithm has its origin in number theory (reference 7) and consists of three seperate operations. The first is a permutation (re-ordering) of the input data. The second operation is correlation of the permuted input data with permuted discrete cosine and sine samples. The third operation is a repermutation, which yields the DFT components in the conventional order of linear frequency. This final stage may be ignored in applications which also involve an inverse DFT.

In this section the mathematical background for the PNT will be summarized. Where necessary examples are provided to assist in the understanding of the concepts.

If the standard DFT equation, i.e.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} = \sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad k = 0, 1, ...N-1, \tag{12}$$

is to be converted to correlation between $x(n)$ and the twiddle factors W_N's, the nk product needs to be converted to a sum $n+k$. For cases where N is prime number theory allows us to achieve this.

According to number theory (reference 7), for each prime number N, there exist integers r, known as primitive roots, whose successive integer powers modulo-N will generate a permuted version of the sequence $1, 2, 3, ...N-1$.

What this means is that for a prime number N, it is possible to map the sequence $\{p\} = 0, 1, 2, ...N-2$ via the equation

$$q = (r^q) mod\ N \quad where\ \{p\} = \{0, 1, 2, 3, ...N-2\} \tag{13}$$

to a sequence $\{q\}$ where q is a one-to-one map of the original sequence $\{p\}$ and consists of a permuted version of the sequence $\{1, 2, 3, ...N-1\}$. For such a unique map to be possible r must be a primitive root of N. Let us consider $N = 7$, for which one of the primitive roots of 7 is 3. From (13) the mapping equation is

$$q = (3^p) mod\ N \quad where\ p = 0, 1, 2, ..., 5$$

For $p = 3$, $q = (27) mod\ 7 = 6$. Table 9.1 gives the corresponding values of p and q which confirm the one-to-one nature of the mapping.

It should be emphasized that for any prime N, the primitive root r, is not unique. For example table 9.2 illustrates the mapping given by (13) for $N = 7$. From this table you can see that the mapping is unique and cyclic for $r = 3$ and $r = 5$ which are the primitive roots of 7. In most practical cases the smallest primitive root is often selected.

p	0	1	2	3	4	5	6	7
q	1	3	2	6	4	5	1	3

Table 9.1 The mapping corresponding to equation (13) for $r = 3$

$r \setminus p$	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	4	1	2	4	1	2	4	1	2	4	1
3*	1	3	2	6	4	5	1	3	2	6	4	5	1
4	1	4	2	1	4	2	1	4	2	1	4	2	1
5*	1	5	4	6	2	3	1	5	4	6	2	3	1
6	1	6	1	6	1	6	1	6	1	6	1	6	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	2	4	1	2	4	1	2	4	1	2	4	1
10	1	3	2	6	4	5	1	3	2	6	4	5	1
11	1	4	2	1	4	2	1	4	2	1	4	2	1
12	1	5	4	6	2	3	1	5	4	6	2	3	1
13	1	6	1	6	1	6	1	6	1	6	1	6	1
* Note that $r = 3$ and $r = 5$ give unique mapping and are therefore the primitive roots of 7.													

Table 9.2 Values for q in the mapping $q = (r^p) mod 7$

If N is prime and r is primitive root of N then we would like to apply the mapping given by (13) to an N-point DFT. Referring to the DFT equation (12), it can be seen that the subscripts n and k vary from 0 to $N-1$ whilst in the mapping given by (13) the variable q assumes values from 1 to $N-1$ i.e it excludes zero. To overcome this problem we write the DFT equation (12) in the following form

$$X(0) = \sum_{n=0}^{N-1} x(n) \tag{14a}$$

$$X(k) - x(0) = \sum_{n=1}^{N-1} x(n) W_N^{nk} \qquad k = 1, ..., N-1 \tag{14b}$$

i.e. we separate the expressions for the zero-frequency DFT component $X(0)$ (the d.c. term) which is very simple. This expression consists of N additions only and if required can be calculated directly.

We are left with DFT components $X(k)$ corresponding, to $k = 1, 2, ..N-1$, which are given by (14b). Note that in this equation we have taken $x(0)$ to the left-hand side so as the summation over n is from $n = 1$ to $n = N-1$. We can now apply the mapping given by (13) to equation (14) via the following transformations,

$$n = (r^m) \ mod \ N \qquad m = 0, 1, ..., N-2 \tag{15}$$

$$k = (r^l) \ mod \ N \qquad l = 0, 1, ..., N-2 \tag{16}$$

which result in a permutation of the terms in the summation and a change in the order of the equations. Equation (14b) then becomes:

$$X[(r^l) mod N] - x(0) = \sum_{m=0}^{N-2} x[(r^m) mod N] W_N^{[(r^m) mod N][(r^l) mod N]} \tag{17}$$

where $l = 0, 1, 2, ...N-2$

Remember that the twiddle factor, W_N, is cyclic in N, therefore we have

$$X[(r^l)modN] - x(0) = \sum_{m=0}^{N-2} x[(r^m)modN]W_N^{r^{(m+l)}} \tag{18}$$

where $l = 0, 1, 2, ...N-2$ This equation indicates that the sequence $X[(r^i)modN] - x(0)$ can be calculated via a circular correlation of the permuted input sequence $x[(r^i)mod\ N]$ and the sequence $e^{-2\pi j(r^i)/N}$. To see this clearly let us consider in detail an example for a DFT of length 7.

The expression for a 7-point DFT is given by

$$X(k) = \sum_{n=0}^{6} x(n)W_N^{nk} = \sum_{n=0}^{6} x(n)e^{-2\pi jnk/7} \tag{19}$$

where $n, k = 0, 1, 2, ...N-1$

This DFT equation can be expressed in matrix form as: (The subscript 7 has been dropped from W_7 for convenience)

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 \\ W^0 & W^2 & W^4 & W^6 & W^1 & W^3 & W^5 \\ W^0 & W^3 & W^6 & W^2 & W^5 & W^1 & W^4 \\ W^0 & W^4 & W^1 & W^5 & W^2 & W^6 & W^3 \\ W^0 & W^5 & W^3 & W^1 & W^6 & W^4 & W^2 \\ W^0 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \end{bmatrix} \tag{20}$$

The superscript in each W^{nk} is evaluated mod 6. Noting that $W^0 = 1$ and separating the equation for $X(0)$ we can rewrite equation (20) as:

$$\begin{bmatrix} X(1) - x(0) \\ X(2) - x(0) \\ X(3) - x(0) \\ X(4) - x(0) \\ X(5) - x(0) \\ X(6) - x(0) \end{bmatrix} = \begin{bmatrix} W^1 & W^2 & W^3 & W^4 & W^5 & W^6 \\ W^2 & W^4 & W^6 & W^1 & W^3 & W^5 \\ W^3 & W^6 & W^2 & W^5 & W^1 & W^4 \\ W^4 & W^1 & W^5 & W^2 & W^6 & W^3 \\ W^5 & W^3 & W^1 & W^6 & W^4 & W^2 \\ W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \end{bmatrix} \tag{21}$$

and

$$X(0) = \sum_{n=0}^{6} x(n) \tag{22}$$

The expression for $X(0)$ i.e equation (22) is a simple summation and is assumed to be evaluated separately.

Dealing with the computationally intensive part of the transform i.e. equation (21), we can apply the mapping given by (13) to this equation which would convert equation 21 into a cyclic correlation suitable for implementation using IMS A100 transversal filter. We choose $r = 3$ which is the smallest primitive root of 7. The mapping would thus correspond to that given by table 9.1. We first apply the permutation given by this mapping to the input sequence of $x(n)'s$. This would correspond to a column permutation of the twiddle matrix as shown in (23).

$$\begin{bmatrix} X(1) - x(0) \\ X(2) - x(0) \\ X(3) - x(0) \\ X(4) - x(0) \\ X(5) - x(0) \\ X(6) - x(0) \end{bmatrix} = \begin{bmatrix} W^1 & W^3 & W^2 & W^6 & W^4 & W^5 \\ W^2 & W^6 & W^4 & W^5 & W^1 & W^3 \\ W^3 & W^2 & W^6 & W^4 & W^5 & W^1 \\ W^4 & W^5 & W^1 & W^3 & W^2 & W^6 \\ W^5 & W^1 & W^3 & W^2 & W^6 & W^4 \\ W^6 & W^4 & W^5 & W^1 & W^3 & W^2 \end{bmatrix} \begin{bmatrix} x(1) \\ x(3) \\ x(2) \\ x(6) \\ x(4) \\ x(5) \end{bmatrix} \tag{23}$$

Note that the matrix equations (23) and (21) are essentially the same and their difference is only in the order of the terms. Next we apply the same mapping to the column matrix containing $X(k) - x(0)$ terms in equation

(23), this would of course correspond to a similar permutation of the rows of twiddle matrix in (23); and the result is given as equation (24).

$$\begin{bmatrix} X(1)-x(0) \\ X(3)-x(0) \\ X(2)-x(0) \\ X(6)-x(0) \\ X(4)-x(0) \\ X(5)-x(0) \end{bmatrix} = \begin{bmatrix} W^1 & W^3 & W^2 & W^6 & W^4 & W^5 \\ W^3 & W^2 & W^6 & W^4 & W^5 & W^1 \\ W^2 & W^6 & W^4 & W^5 & W^1 & W^3 \\ W^6 & W^4 & W^5 & W^1 & W^3 & W^2 \\ W^4 & W^5 & W^1 & W^3 & W^2 & W^6 \\ W^5 & W^1 & W^3 & W^2 & W^6 & W^4 \end{bmatrix} \begin{bmatrix} x(1) \\ x(3) \\ x(2) \\ x(6) \\ x(4) \\ x(5) \end{bmatrix} \tag{24}$$

Referring to equation (24) it can be seen that the twiddle-factor matrix has the property that each row can be obtained by a left-shift (rotate) of the previous row. This means that the sequence $\{X(1)-x(0),\ X(3)-x(0),\ X(2)-x(0),\ X(6)-x(0),\ X(4)-x(0),\ X(5)-x(0)\}$ can be obtained by performing a circular convolution between the sequence $\{x(1), x(3), x(2)x(6), x(4), x(5)\}$ and a permuted twiddle factor set given by $\{W^1, W^3, W^2, W^6, W^4, W^5\}$.

Figure 9.3 shows how this circular convolution can be implemented using a transversal filter structure. For the moment let us confine our attention to the canonical transversal filter structure and assume that the transversal filter is capable of complex processing i.e. both input data $x(n)$ and the twiddle factors are complex. It will be shown later how a single IMS A100 device can be used to implement this complex processing. Two implementations are shown in figure 9.3.

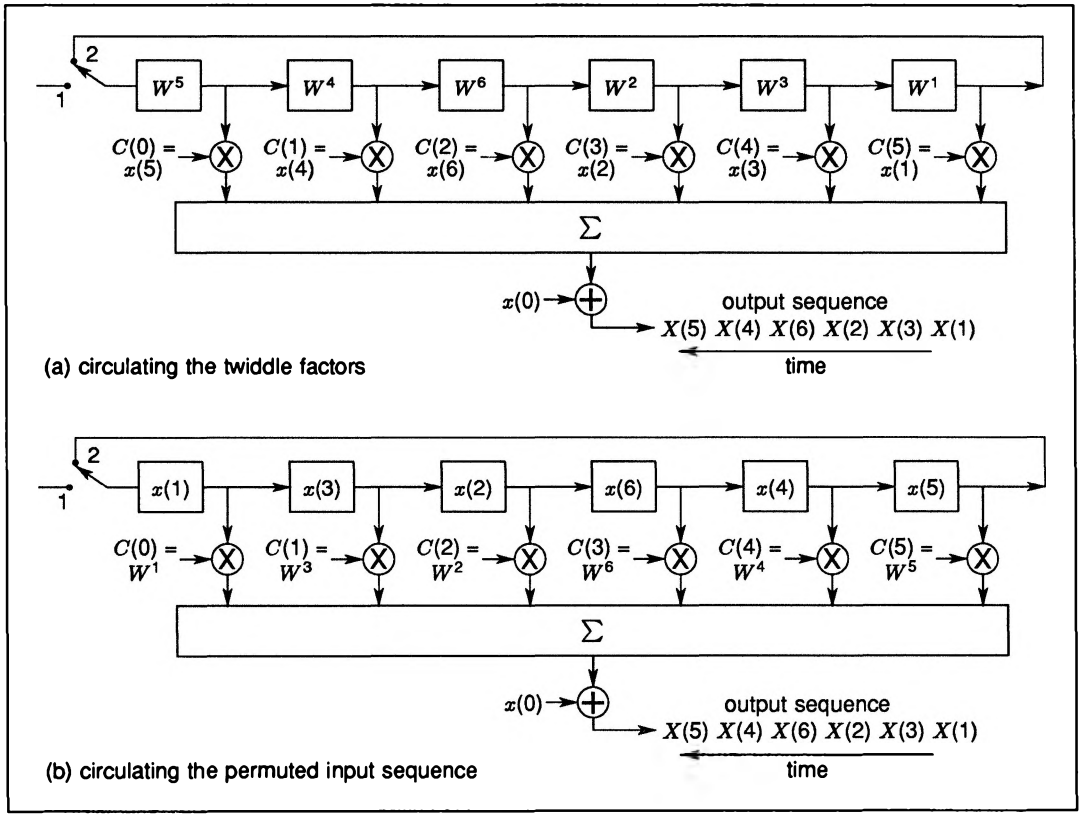

Figure 9.3 Prime number transform implementation based on the transversal filter structures

In the first implementation, figure 9.3a, the permuted twiddle factors are used as the inputs to the transversal filter. These twiddle factors are first loaded into the filter with the input switch at position 1 and then circulated

with the input switch at position 2. The output samples, as shown in figure 9.3a, would correspond to the DFT of the input sequence $x(n)$. You should be able to confirm this by referring to the matrix equation given by (24). For the arrangement in figure 9.3a the allocation of the data sequence $x(n)$ to the coefficient memory can be formulated as:

$$C(i) = x[(r^{N-2-i}) mod\ N] \qquad i = 0, 1, ..., N-2 \tag{25}$$

where N is 7 in this case. Similarly the twiddle factor sequencing, at the input of figure 9.3a, can be mathematically expressed as:

$$Input(i) = W^{[(r^i) mod\ N]} \qquad i = 0, 1, ..., N-2 \tag{26}$$

The output sequence for figure 9.3a is given by:

$$Output(i) = X[(r^i) mod\ N] \qquad i = 0, 1, ..., N-2 \tag{27}$$

Figure 9.3b shows the second possible implementation in which the coefficient memory contains the permuted twiddle factors and the permuted data sequence is loaded at the input of the filter and is circulated to generate the DFT of the input samples. For this implementation the generalized equations for the input and output sequences and the allocation of coefficient memory are:

$$C(i) = W^{[(r^i) mod\ N]} \qquad i = 0, 1, ..., N-2 \tag{28}$$

$$Input(i) = x[(r^{N-2-i} mod\ N] \qquad i = 0, 1, ..., N-2 \tag{29}$$

$$Output(i) = X[(r^i) mod\ N] \qquad i = 0, 1, ..., N-2 \tag{30}$$

Equation (25) to (27) and (28) to (30) define the required permutation and sequencing for a generalised prime number transform based on the canonical transversal filter structure.

It was argued earlier that the IMS A100 implementation of the transversal filter structure (figure 9.1b) is identical in behaviour to that of the canonical form (figure 9.1a). The only difference is that in the canonical form the first coefficient, $C(0)$, is associated with the left most memory location (see figure 9.1a) while in the IMS A100 implementation the right most coefficient register is allocated to $C(0)$. We will now show how our 7-Point DFT can be implemented in complex form using the IMS A100 transversal filters. The input data samples $x(n)$, the twiddle factors W^n, and the DFT output samples $X(n)$ can be expressed in terms of their real and imaginary parts as:

$$x(n) = xr(n) + jxi(n) \tag{31}$$

$$W_N^n = e^{-2\pi/N} = \cos(\frac{2\pi n}{N}) + j\sin(\frac{-2\pi n}{N}) = WR(n) + jWI(n) \tag{32}$$

$$X(n) = XR(n) + jXI(n) \tag{33}$$

As mentioned earlier the IMS A100 device contains two sets of coefficient memories; at any instant one set of the coefficients is used in the computation whilst the other set can be accessed via a standard memory interface. One very important feature of the device is that the two memory banks can be exchanged automatically at the beginning of every computation cycle. i.e. alternate set of coefficients are applied to the filter successively. This feature allows complex convolution and correlation to be performed in a single device. (This is unlike most conventional realizations of complex convolution/correlation where, as shown in figure 9.4, four transversal filters are often used to implement these complex functions). This is achieved by appropriately loading the two coefficient memories with combinations of real and imaginary samples of the reference signal and using the continuous memory-swap mode to implement complex processing. The real and imaginary parts of the signal to be correlated (or convolved) with the reference signal are then applied alternatively to the input of the IMS A100 device. An application note entitled 'Complex Processing Using the IMS A100 Transversal Filters' covers this topic in detail and is available from INMOS. The remainder of this section gives an overview of the topic in relation to complex DFTs.

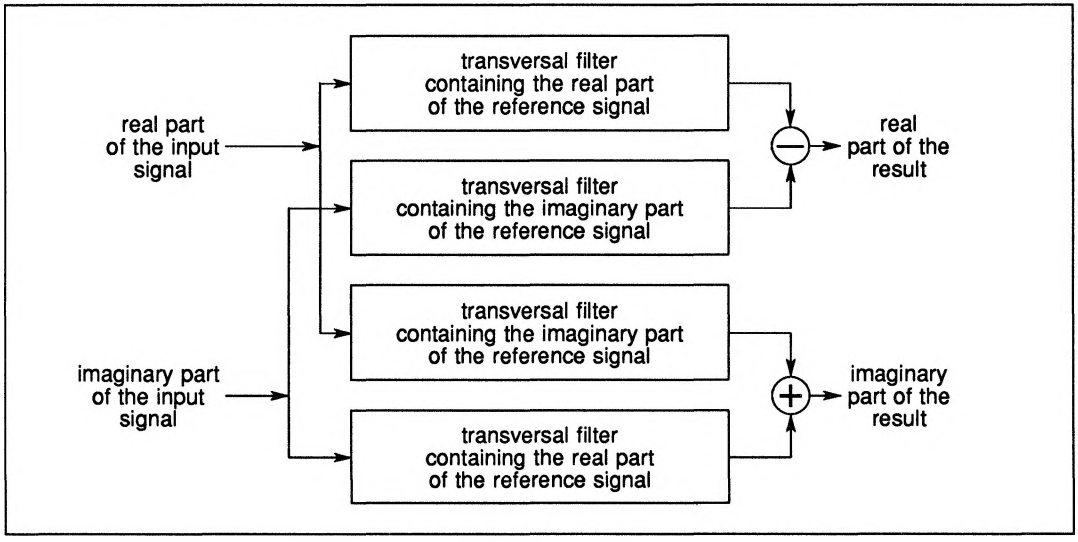

Figure 9.4 Conventional complex correlator/convolver involving four transversal filters

Figure 9.5 shows the IMS A100 implementation of the 7-point DFT example, corresponding to figure 9.3a, where the twiddle factors are circulated. The notation used for real and imaginary parts of the signals is that given by equations (31) to (33). The twiddle factors are applied to the input in a sequence identical to that used in figure 9.3a, with the real part of each sample followed by its imaginary part. The output sequence is also shown. It is assumed that the delay-and-add chain in the IMS A100 is cleared first by writing several zero's to the input. (Note that this is only needed once and any further transforms do not need this flushing). The memory banks are set in their continuous-swap mode. In the first computation cycle, with $WR(1)$ on the input, the memory bank 'A' is used in the computation; in the second cycle, when $WI(1)$ is applied to the input, the memory bank 'B' is used in the computation; in the third cycle $WR(3)$ is the input sample and the memory bank A is used in the computation and so on. Note also that for each output sample an external addition (with either $xr(0)$ or $xi(0)$ depending on whether the output corresponds to real or imaginary part of the result) has to be carried out as dictated by equation (24). This is a negligible overhead compared to the computation performed by the transversal filter and can easily be carried out by the host processor.

In figure 9.5 the first eleven output samples (denoted with $*$) are partial results and as such are not fully valid. This is due to the inherent delay associated with any transversal filter implementation. In continuous processing, however, it is possible to avoid these undefined output samples and to achieve a duty cycle of 100% by updating two coefficient memory locations with new data samples. For example in figure 9.5, assumming that we have applied the first cycle of the twiddle factor values i.e. $WR(1), WI(1), WR(3), WI(3)$.....$WR(5), WI(5)$ to the input, it is possible to update the coefficient locations corresponding to $xi(1)$ and $xr(1)$ in memory bank A with the imaginary and real parts of the first sample of the new input data block. This can be done during the latest computation cycle (with $WI(5)$ as the input twiddle factor) when the memory bank A is free. In the next cycle when $WR(1)$ is applied to the input, $xr(1)$ and $-xi(1)$ in the memory bank B can be updated with new data values while this memory bank is free. In the following computation cycle when $WI(1)$ is the input twiddle factor, the coefficient memory locations corresponding to $xi(3)$ and $xr(3)$ in the memory bank A are updated and so on. This technique removes the undefined output samples and achieves a duty cycle of 100%.

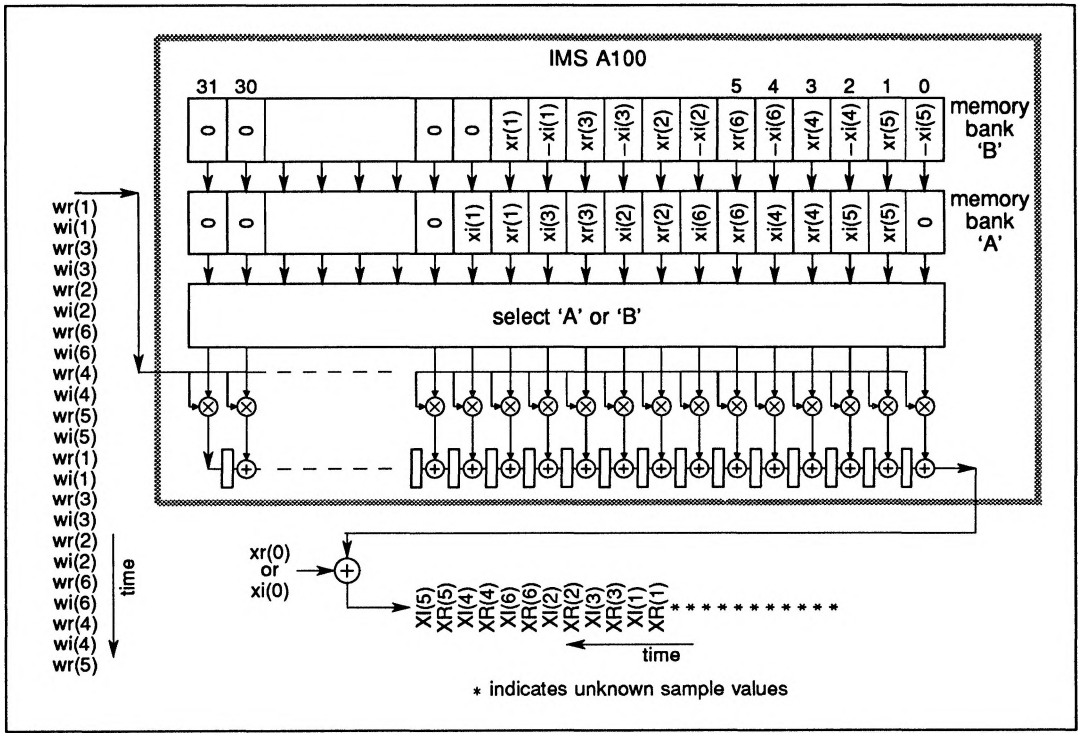

Figure 9.5 IMS A100 implementation of a 7 point complex DFT, corresponding to the canonical transversal filter realization of figure 9.3a

Figure 9.6 depicts the IMS A100 implementation of the 7-point DFT, corresponding to figure 9.3b, where the input data samples are recirculated and the twiddle factors are stored in the coefficient memories. The input data samples are applied to the input in a sequence identical to that used in figure 9.3b, with real part of each sample followed by its imaginary part. The output sequence is also shown. Other characteristics of this implementation are identical to the previous case with the exception that in this implementation it is impossible to avoid initial undefined output samples even when several continuous transforms are to be performed.

The IMS A100 devices can be cascaded without any external components by simply connecting the output of the first device to the cascade-input of the second device. This simple cascading allows transversal filters/correlators with many stages to be easily implemented.

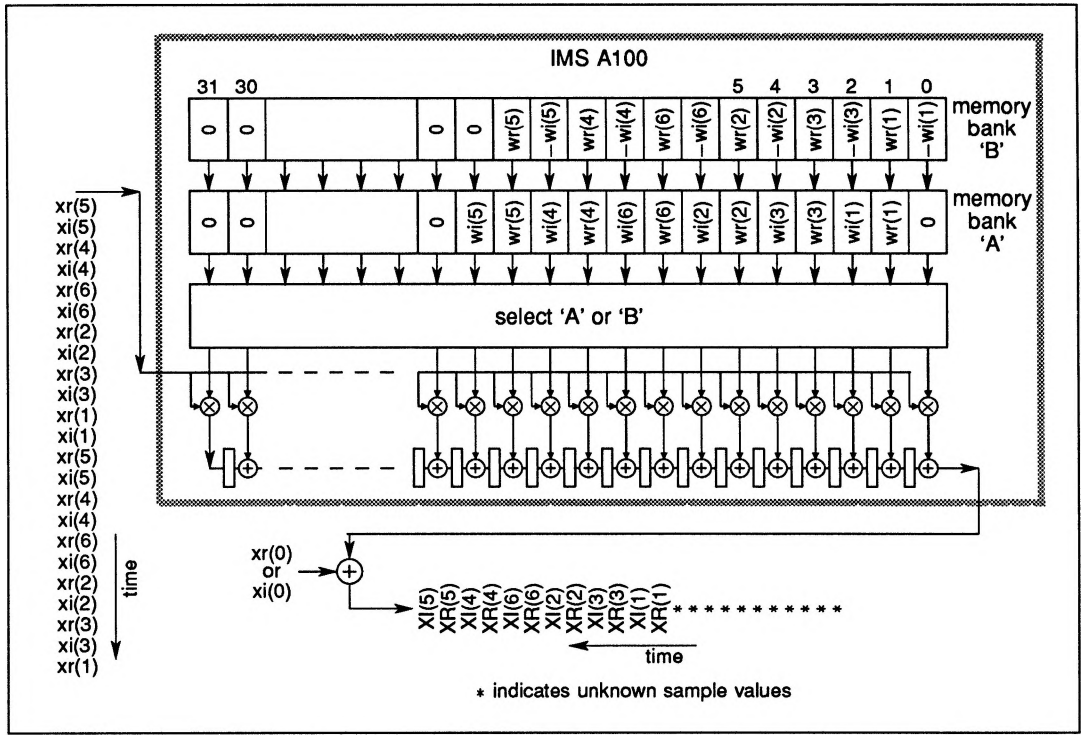

Figure 9.6 IMS A100 implementation of a 7 point complex DFT corresponding to the canonical transversal filter realization of figure 9.3b

Using prime-number algorithm there are basically two ways to implement A100 based DFT processors capable of handling long data blocks. The obvious approach is to cascade several devices resulting in a sufficiently large correlator/convolver capable of dealing with the whole data block size. This approach is only acceptable for moderate block sizes and becomes impractical if the data size is very large. The second approach is based on mapping techniques which convert a large DFT into several independent short transforms. These short transforms can then be evaluated either concurrently or sequentially, depending on the required performance. This means that the decomposition techniques described here are particularly useful as they provide the basis for trade-offs between cost and speed. This subject will be discussed in detail in section 5 of this application note.

As mentioned earlier the IMS A100 transversal filter has an on-chip industry standard memory interface which allows the part to be fully memory-mapable. Figure 9.7 shows a schematic diagram of a simple system making use of this memory interface. When implementing the prime transform algorithm on this system, the IMS A100 (or arrays of them) will perform the bulk of the computation and the host processor will be responsible for data permutation (using look-up tables), evaluating the X(0) term (equation 22), and performing the auxiliary addition involving either $xr(0)$ or $xi(0)$ (see figures 9.5–9.6).

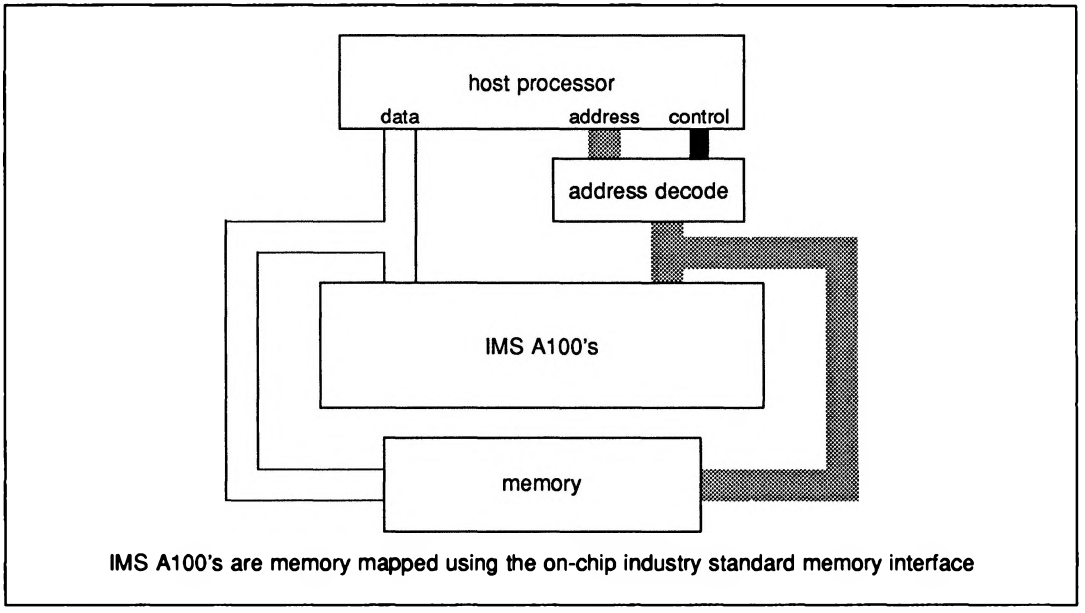

Figure 9.7 Schematic diagram of a simple IMS A100 based system

Another possible system configuration is shown in figure 9.8. This is particularly suitable for the arrangement of figure 9.5. The real and imaginary parts of the twiddle factors are pre-loaded in the memory MEM1 and are supplied to the A100 via the dedicted input port. The sequencer shown in figure 9.8 could be a simple counter. The processor accesses the coefficient memories and the output result via the IMS A100's memory interface. Other system configurations are possible. For example figure 9.9 shows the schematic of a high performance signal processing system using a dedicated controller.

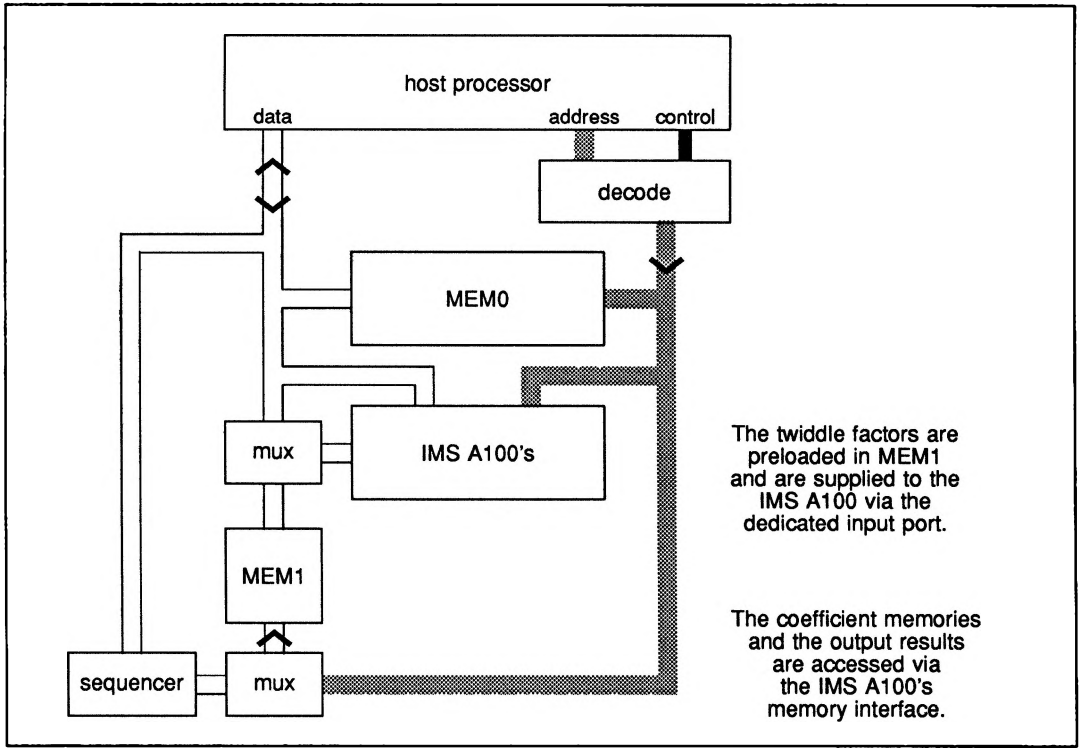

Figure 9.8 An implementation particularly suitable for the arrangement in figure 9.5

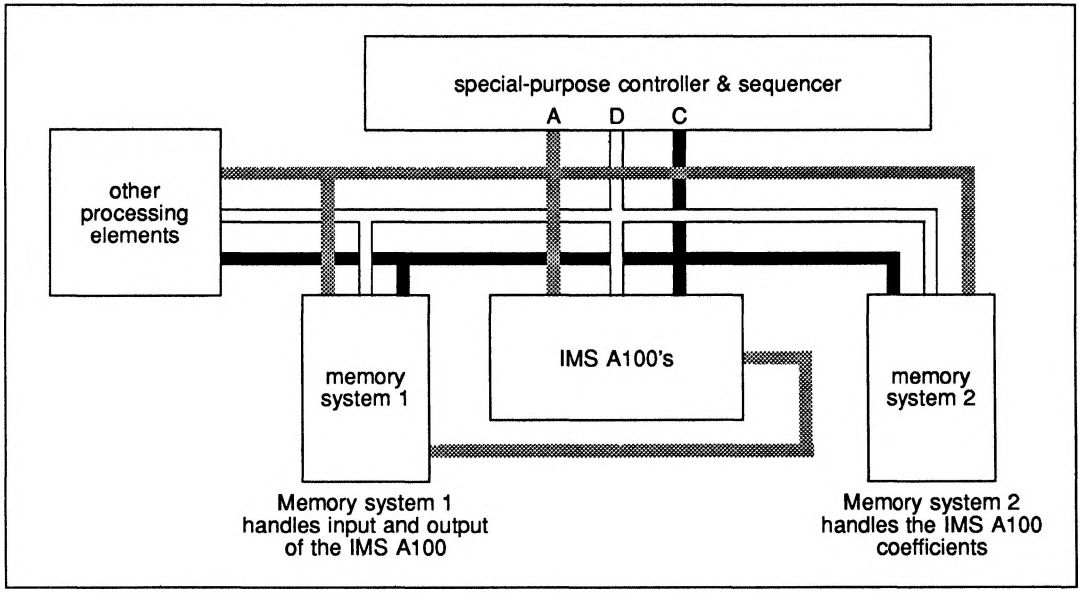

Figure 9.9 Schematic block diagram of a high performance involving a special purpose controller

9.4.2 The chirp-z transform

Another algorithm which converts the DFT equation into a convolution (or correlation) is the chirp-z transform. The reason for the name chirp is that the transform uses a sampled linear frequency-modulated carrier which in signal processing is often termed a 'chirp signal'. In the previous section we saw that the prime number transform consisted of three operations, namely

(i) Data input permutation.

(ii) Convolution (or correlation) of the permuted data with a permuted sequence of twiddle factors.

(iii) A final permutation to obtain the correct output sequence.

Figure 9.10 summarizes the principles of the prime number transform algorithm. The auxillary computation for zeroth input sample and evaluation of $X(0)$ are also shown in this schematic diagram. The structure of the chirp-z DFT algorithm is similar to that of the prime number transform technique and consists of the following sequence of three operations:

(i) Premultiplication of the input sequence $x(n)$ by the chirp $e^{(-\pi jn^2/N)}$.

(ii) Convolution (or correlation) of the resulting sequence with a second chirp signal.

(iii) Post-multiplication of the resulting sequence by the chirp signal $e^{(-\pi jk^2/N)}$.

These operations are summarized in figure 9.11. Comparing figure 9.10 and figure 9.11, it can be seen that the major difference between the two algorithms is that in the chirp-z transform the permutation operations are replaced by multiplications.

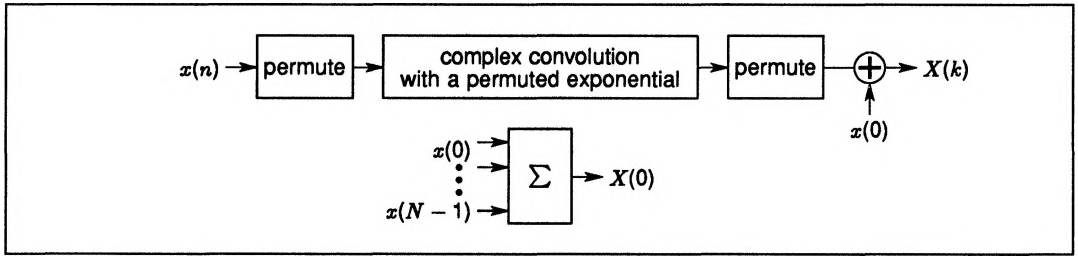

Figure 9.10 The principle of the prime number transform

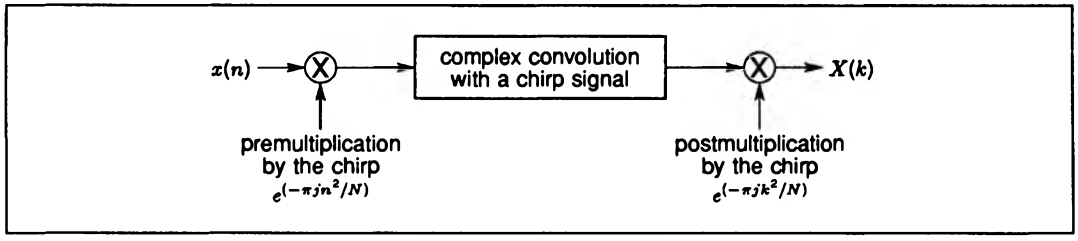

Figure 9.11 The basic principle of the CZT

In the CZT the convolution (correlation) operations can be implemented using the IMS A100 transversal filter, but the pre- and post-multiplications have to be done externally. In many applications data permutation may be preferred to multiplication in which case the prime numbers approach may be considered more advantageous. However there are applications (e.g beamforming) where the CZT, in particular a simplified version of it referred to as sliding CZT, is preferred.

To understand the CZT algorithm we start from the DFT equation

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} = \sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad k = 0, 1, \ldots N-1 \tag{34}$$

and replace the term $-2nk$ in the complex exponential with the seemingly more complicated expression

$$-2nk = (k-n)^2 - k^2 - n^2 \tag{35}$$

hence

$$X(k) = e^{-j\pi k^2/N} \sum_{n=0}^{N-1} [x(n)e^{-j\pi n^2/N}][e^{j\pi(k-n)^2/N}] = e^{-j\pi k^2/N} \sum_{n=0}^{N-1} y(n)[e^{j\pi(k-n)^2/N}] \tag{36}$$

where $k = 0, 1, \ldots\ldots, N-1$.

In equation 36, the term $X(k)$ consists of three operations:

(i) Multiplications of the samples $X(n)$ with a complex linear frequency-modulated signal $e^{-j\pi n^2/N}$ to form a new set of samples $y(n)$; This operation is often referred to as premultiplication by a chirp,

(ii) the convolution of $y(n)$ with a second-linear frequency-modulated signal (the term $e^{j\pi(k-n)^2/N}$ and

(iii) post multiplication by $e^{-j\pi k^2/N}$.

Note that if only the power spectrum of the signal is required the final operation can be omitted, since $e^{-j\pi k^2/N}$ represent only a phase-shift and $|e^{-j\pi k^2/N}| = 1$. Also in operation (ii) the term $(k-n)^2$ in the complex exponential is equal to $(n-k)^2$ so that a convolution operation, in this case, is identical to that of a correlation. Figures 9.12 and 9.13 show examples for a 6-point CZT implemented using the canonical transversal filter structure. In these diagrams it is assumed that the transversal filter is capable of complex processing. As described in the previous section the complex conolution/correlation can easily be implemented using the IMS A100 transversal filter chip.

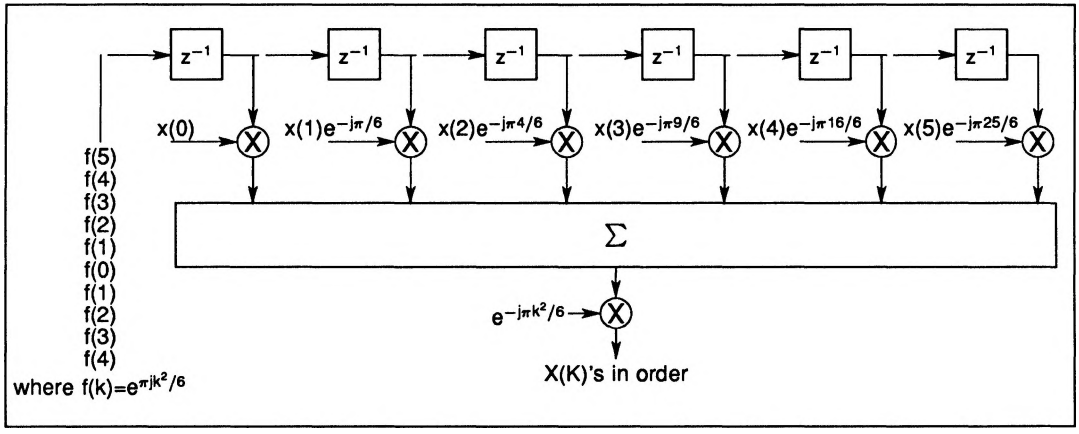

Figure 9.12 Schematic diagram for 6 point CZT using canonical transversal filter structure

In figure 9.12, samples corresponding to the product of the input data and the premultiplying chirp are stored as the filter coefficients and a second sampled chirp is fed to the input of the convolver. Note that the convolver has N-complex points. Figure 9.13 shows an alternative implementation where the product of the input data samples and the premultiplying chirp samples are the inputs to the convolver and a chirp signal is used as the reference signal in the coefficient store. Note that in this arrangement the convolver has to have $2N-1$ complex points. However when the number of points in the transform are even (N =even), it can easily be shown that the sampled chirp signal $f(n) = e^{j\pi n^2/N}$ has the following properties:

(i) it is periodic with a periodicity equal to N i.e.

$$f(n) = f(n+N) \tag{37}$$

(ii) It is symmetrical about $n = \frac{N}{2}$ i.e.

$$f(n) = f(N-n) \tag{38}$$

These properties convert the convolution in figure 9.13 into a circular one which can be implemented with an N-point complex transversal filter.

In many applications the PNT may be preferred to the CZT because of the requirement of pre- and post-multiplications in the latter. (Remember that in the PNT, permutation operations are involved rather than multiplications).

As there are considerable amount of literature available on CZT, it will not be considered here in any more detail.

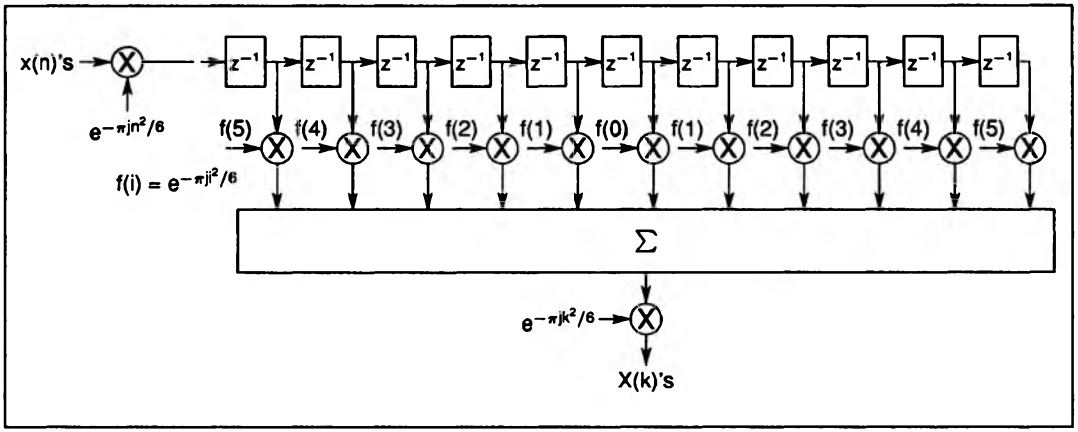

Figure 9.13 Alternative implementation of the 6 point CZT

9.5 Multidimensional index mapping for DFT decomposition

In the previous sections, algorithms which convert the DFT into convolution/correlation operations were presented. These algorithms, particularly the PNT, are suitable for implementation using the IMS A100 transversal filter.

In order to compute the DFT of long data sequences, one approach is to cascade several the IMS A100 devices so that sufficient convolution/correlation points are made available. This approach is only acceptable for moderate data sizes, and does not provide the optimum performance for a given number of devices.

In this section, index mapping techniques are described which allow long DFT's to be decomposed into several shorter transforms. These shorter transforms can then be efficiently implemented by using IMS A100 transversal filters. The decomposition techniques described here can be viewed as generalised algorithms, with radix-2 FFT being a special case of these more general partitioning techniques. These mapping techniques provide the basis for designing highly concurrent systems and optimization in terms of performance and cost.

9.5.1 Basic concepts of index mapping

The essence of these mapping techniques is that by a simple change of variable, the original complex problem is converted into several easy ones. Before applying these techniques to the DFT, let us consider a few examples which should help to familiarize the reader with the terminologies used in general index mapping.

Consider a one-dimensional array of data $x(n)$, $n = 0$ *to* $N-1$, where N is the total number of elements in the array. For $N = 6$, the array elements will be given by

$$\{x(0)\ \ x(1)\ \ x(2)\ \ x(3)\ \ x(4)\ \ x(5)\}$$

Let us rearrange this one-dimensional array into a two-dimensional array as shown below:

$$\begin{bmatrix} x(0) & x(1) & x(2) & x(3) & x(4) & x(5) \end{bmatrix} \overset{map}{\Longleftrightarrow} \begin{bmatrix} x(0) & x(1) & x(2) \\ x(3) & x(4) & x(5) \end{bmatrix} = \begin{bmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \end{bmatrix} \quad (39)$$

We have 'mapped' the original one-dimensional array, $x(n)$, into a two-dimentional array $y(n_1, n_2)$. It can be seen that in this example the mapping is given by the following linear equation.

$$n = 3n_1 + n_2 \quad , \quad x(n) = y(n_1, n_2) \tag{40}$$

where $n_1 = 0, 1$ and $n_2 = 0, 1, 2$.

The mapping is said to be one-to-one (unique) as all the elements in the original array, $x(n)$, appear in the two-dimensional array $y(n_1, n_2)$.

As a second example consider the following mapping:

$$\begin{bmatrix} x(0) & x(1) & x(2) & x(3) & x(4) & x(5) \end{bmatrix} \overset{map}{\Longleftrightarrow} \begin{bmatrix} x(0) & x(2) & x(4) \\ x(3) & x(5) & x(1) \end{bmatrix} = \begin{bmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \end{bmatrix} \tag{41}$$

This mapping has been obtained from

$$n = (3n_1 + 2n_2) \ mod \ 6 \tag{42}$$

Note that in equation (42) the index n is evaluated modulo N (in this case N=6) and it is therefore cyclic in N.

As a final example let us apply the following mapping

$$n = (2n_1 + 2n_2) \ mod \ 6 \tag{43}$$

to our 6-element array, which gives;

$$\begin{bmatrix} x(0) & x(1) & x(2) & x(3) & x(4) & x(5) \end{bmatrix} \overset{map}{\Longleftrightarrow} \begin{bmatrix} x(0) & x(2) & x(4) \\ x(2) & x(4) & x(0) \end{bmatrix} = \begin{bmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \end{bmatrix} \tag{44}$$

Obviously this map is not unique as $x(1), x(3)$ and $x(5)$ are not represented in the matrix $y(n_1, n_2)$.

9.5.2 Generalisation and conditions for uniqueness

Let us now generalize the ideas developed in the previous examples. We are interested in mapping a one-dimensional array of length $N = N_1 \times N_2$ into a two-demensional array that is N_1, by N_2 in size. In other words, the one-dimensional array

$$x(n) \ \ for \quad n = 0, 1, ...N - 1$$

is to be mapped into a two-dimensional array

$$y(n_1, n_2) \ \ for \ \ n_1 = 0, 1, ...N_1 - 1, \ \ and \ n_2 = 0, 1, ...N_2 - 1.$$

The major requirement is that the mapping must be unique. This mapping can be represented by

$$\begin{bmatrix} x(0) & x(1) & x(2) & \dots & x(N-1) \end{bmatrix} \overset{map}{\Longleftrightarrow} \begin{bmatrix} y(0,0) & y(0,1) & \dots & y(0,N_2-1) \\ y(1,0) & y(1,1) & \dots & y(1,N_2-1) \\ \vdots & \vdots & \ddots & \vdots \\ y(N_1-1,0) & y(N_1,1) & \dots & y(N_1-1,N_2-1) \end{bmatrix} \tag{45}$$

where

$$x(n) = y(n_1, n_2) \tag{46}.$$

This map, in general, can assume many different forms, but the one particularly useful to the DFT is the linear form,

$$n = (M_1 n_1 + M_2 n_2) \ mod \ N. \tag{47}$$

Note that n is evaluated modulo N, making the map cyclic in n. In order for this map to be unique and one-to-one, the mapping constants M_1 and M_2 must satisfy certain conditions. In the literature (references 5 & 6) these conditions have been derived from number theory for two cases which are described in the following two subsections.

Relatively prime case

In this case N_1 and N_2 are relatively prime and have no common factor. In the literature this case is often denoted by:

$$(N_1, N_2) = 1 \tag{48}$$

which means that the greatest common divisor of N_1 and N_2 is unity. For example $(5,7) = 1$, $(8,9) = 1$ and $(6,25) = 1$. For this case the conditions on M_1 and M_2 which make the mapping, given by (47), unique and one-to-one are: (references 5 & 6)

$$[(M_1 = \alpha N_2) \; and/or \; (M_2 = \beta N_1)] \; and \; (M_1, N_1) = (M_2, N_2) = 1 \tag{49}$$

where α and β are integers.In other words, to ensure a unique maping for this case:

(M_1 must be a multiple of N_2)
or (M_2 must be a multiple of N_1)
or (M_1 and M_2 must be multiples of of N_2 and N_1 respectively)

and M_1 and N_1 must be relatively prime.

and M_2 and N_2 must be relatively prime.

As an example consider $N_1 = 5$, $N_2 = 7, N = 35$. From (49) we have to choose M_1 a multiple of N_2 or M_2 a multiple of N_1 or both. Let us make M_1 the simplest multiple of N_2 i.e. $M_1 = \alpha N_2 = N_2 = 7$, this also satisfies $(M_1, N_1) = (7,5) = 1$. Then noting that we must have $(M_2, N_2) = (M_2, 7) = 1$, possible values for M_2 are: $1,2,3,4,5,6,8,9,10,11,12,13,15$. While $M_2 = 7,14,21...$ are not allowed as they are not relatively prime with N_2. (Note also that for $M_1 = 5,10,15$, we also have $M_2 = \alpha N_1$, which is allowed.)

If M_1 is chosen to be $M_1 = 2 \times N_2 = 14$, then again the same values of M_2 as above are valid.

This example shows that a large class of unique mappings exist for this case.

Common factor case

In this case N_1 and N_2 have a common factor r. i.e. their greatest common divisor is r and we have:

$$(N_1, N_2) = r \tag{50}$$

For example $(10,5) = 5$, $(9,12) = 3$, and $(7,21) = 7$. For this case the conditions on M_1 and M_2, making the mapping given by (47) unique, have been shown to be: (references 5 & 6)

$$1) \quad (M_1 = \alpha N_2) \; and \; (M_2 \neq \beta N_1) \; and \; (\alpha, N_1) = (M_2, N_2) = 1 \tag{51a}$$

or

$$2) \quad (M_1 \neq \alpha N_2) \; and \; (M_2 = \beta N_1) \; and \; (\beta, N_2) = (M_1, N_1) = 1 \tag{51b}$$

where α and β are integers.

As an example consider $N_1 = 9, N_2 = 15, N = 135$. From (51a) we can choose, $M_1 = \alpha N_2 = N_2 = 15$. The condition $(\alpha, N_1) = (1,9) = 1$ is already satisfied. From (51a), values of M_2 which are allowed are those which satisfy $(M_2 \neq \beta \times 9)$ and $(M_2, 15) = 1$. Therefore following values of M_2 are allowed

$$M2 = 1,2,4,7,8,11,13,14,16,17,19,22,.....$$

$M_2 = 3,5,6,9,10,...$ are not allowed as they do not satisfy $(M_2, 15) = 1$.

Alternatively we could have chosen $M_1 = \alpha N_2 = 2 \times 15 = 30$, then possible values of M_2 would again be

$$M_2 = 1,2,4,7,8,11,13,14,.....$$

However we could not have chosen $M_1 = \alpha N_2 = 3 \times N_2$ since this violates the requirement $(\alpha, N_1) = 1$.

9.5.3 Application of index mapping to DFT decomposition

Having covered the basic principle of index mapping and the required conditions for uniqueness, let us apply the mapping given by (47) to the DFT and investigate its consequences.

Remember that the DFT equation is given by

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi jnk/N} = \sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad k = 0, 1, \ldots N-1 \tag{52}$$

and that the exponent of W is evaluated modulo N since

$$W_N^{nk} = W_N^{nk+N}. \tag{53}$$

Let us apply the following mappings to the indices n and k; i.e. to both the input data array $x(n)$, and the result array $X(k)$;

$$n = (M_1 n_1 + M_2 n_2) \ mod \ N \tag{54}$$

$$k = (L_1 k_1 + L_2 k_2) \ mod \ N \tag{55}$$

Where n_1, and k_1, are indexed to $(N_1 - 1)$ and n_2 and k_2 to $(N_2 - 1)$. As shown below these mappings convert the one dimensional arrays $x(n)$ and $X(k)$ into two-dimensional matrices $y(n_1, n_2)$ and $Y(n_1, n_2)$ respectively.

$$\left[x(0)\,x(1)\ldots x(n)\ldots x(N-1)\right] \overset{map}{\Longleftrightarrow} \begin{bmatrix} y(0,0) & y(0,1) & \ldots\ldots & \ldots y(0,N_2-1) \\ y(1,0) & y(1,1) & \ldots\ldots & \ldots y(1,N_2-1) \\ \vdots & \vdots & \ddots\ddots & \ddots\vdots \\ \vdots & \vdots & \ddots\, y(n_1,n_2) & \ddots\vdots \\ \vdots & \vdots & \ddots\ddots & \ddots\vdots \\ y(N_1-1,0) & y(N_1,1) & \ldots\ldots & \ldots y(N_1-1,N_2-1) \end{bmatrix}$$

where $y(n_1, n_2) = x(n)$.

And

$$\left[X(0)\,X(1)\ldots X(k)\ldots X(N-1)\right] \overset{map}{\Longleftrightarrow} \begin{bmatrix} Y(0,0) & Y(0,1) & \ldots\ldots & \ldots Y(0,N_2-1) \\ Y(1,0) & Y(1,1) & \ldots\ldots & \ldots Y(1,N_2-1) \\ \vdots & \vdots & \ddots\ddots & \ddots\vdots \\ \vdots & \vdots & \ddots\, Y(k_1,k_2) & \ddots\vdots \\ \vdots & \vdots & \ddots\ddots & \ddots\vdots \\ Y(N_1-1,0) & Y(N_1,1) & \ldots\ldots & \ldots Y(N_1-1,N_2-1) \end{bmatrix}$$

where $Y(k_1, k_2) = X(k)$.

Applying these mappings to the DFT equation gives:

$$X(L_1 k_1 + L_2 k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(M_1 n_1 + M_2 n_2) W_N^{nk} \tag{56}$$

or

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_N^{nk} \tag{57}$$

with

$$W_N^{nk} = W_N^{M_2 L_2 n_2 k_2} \, W_N^{M_1 L_2 n_1 k_2} \, W_N^{M_1 L_1 n_1 k_1} \, W_N^{M_2 L_1 n_2 k_1} \tag{58}$$

Let us now partially define the maps in (54) and (55) by setting;

$$M_2 = \beta N_1 \quad and \quad L_1 = \gamma N_2 \tag{59}$$

Where β and γ are integers.
These assignments make the last term in (58) i.e $W_N^{M_2 L_1 n_2 k_1}$ equals to unity.Let us now separately consider each one of the two cases studied earlier and investigate the effect on the three remaining terms in (58).

Case 1 – prime factor decomposition

For this case N_1 and N_2 are assummed to be relatively prime. From (49) it can be seen that it is possible to also set:

$$M_1 = \alpha N_2 \quad and \quad L_2 = \delta N_1. \tag{60}$$

This makes the second term in (58) i.e. $W_N^{M_1 L_2 n_1 k_2}$ also equal to unity. The remaining two terms in (58) can be written as:

$$W_N^{M_2 L_2 N_2 k_2} = W_N^{\beta N_1 \delta N_1 n_2 k_2} = W_{N_2}^{\beta\delta N_1 n_2 k_2} \tag{61}$$

$$W_N^{M_1 L_1 N_1 k_1} = W_N^{\alpha N_2 \gamma N_2 n_1 k_1} = W_{N_1}^{\alpha\gamma N_2 n_1 k_1} \tag{62}$$

The DFT equation will therefore become:

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_{N_2}^{\beta\delta N_1 n_2 k_2} \right] W_{N_1}^{\alpha\gamma N_2 n_1 k_1} \tag{63}$$

where $k_1 = 0, 1, 2, ..., N_1 - 1$ and $k_2 = 0, 1, 2, ..., N_2 - 1$
The advantage of this equation is that it uncouples the DFT calculations in the sense that the N-point DFT can be mapped into two completely separate sets of short DFT's. In evaluating the $Y(k_1, k_2)$'s, in equation (63), the inner summations over n_2 are operations involving separate rows of the matrix $y(n_1, n_2)$. The outer summations over n_1, on the other hand, are column operations and can be carried out after the row operations are completed. By suitable choice of α, β, γ, and δ, each one of these summations can be expressed as a DFT of the corresponding row or column. Goods (reference 8) suggested:

$$\begin{aligned} \alpha &= \beta = 1 \\ \gamma &= (N_2^{-1}) \ mod \ N_1 \\ \delta &= (N_1^{-1}) \ mod \ N_2 \end{aligned} \tag{64}$$

[Note that in modulo arithmetic the reciprocal of a number $(g) \ mod \ N$ is denoted by $(g^{-1}) \ mod \ N$ and is defined as:

$$[(g) \ mod \ N][(g^{-1} \ mod \ N] = 1.$$

For example

$$(3^{-1}) \ mod \ 7 = (5) \ mod \ 7$$

since $5 \times 3 = 15$ which is 1 modulo 7.]

Applying (64) to (63) gives:

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_{N_2}^{n_2 k_2} \right] W_{N_1}^{n_1 k_1} = \sum_{n_1=0}^{N_1-1} \left[u(n_1, k_2) \right] W_{N_1}^{n_1 k_1} \tag{65}$$

This is now a true two-dimensional DFT with the mapping of n and k given by:

$$n = (N_2 n_1 + N_1 n_2) \ mod \ N \tag{66a}$$

$$k = \{[(N_2^{-1}) \ mod \ N_1] N_2 k_1 + [(N_1^{-1}) \ mod \ N_2] N_1 . k_2\} \ mod \ N \tag{66b}$$

In this example the mapping of n is of the simplest form and that of k is the so called Chinese Remainder Theorem (CRT).

Having mapped the original sequence $x(n)$ into the two dimensional array $y(n_1, n_2)$, equation (65) indicates that the desired DFT can be evaluated by the following two steps:

(i) Performing an N_2-point DFT on each row of matrix $y(n_1, n_2)$. This corresponds to a total of N_1 N_2-point row DFT's and would convert the matrix $y(n_1, n_2)$ into the matrix $u(n_1, k_2)$ as shown in figure 9.14.

(ii) Performing an N_1-point DFT on each column of the resultant matrix, $u(n_1, k_2)$, to yield $Y(k_1, k_2)$. The desired output sequence $X(k)$ is related to $Y(k_1, k_2)$ via the mapping given by (66b).

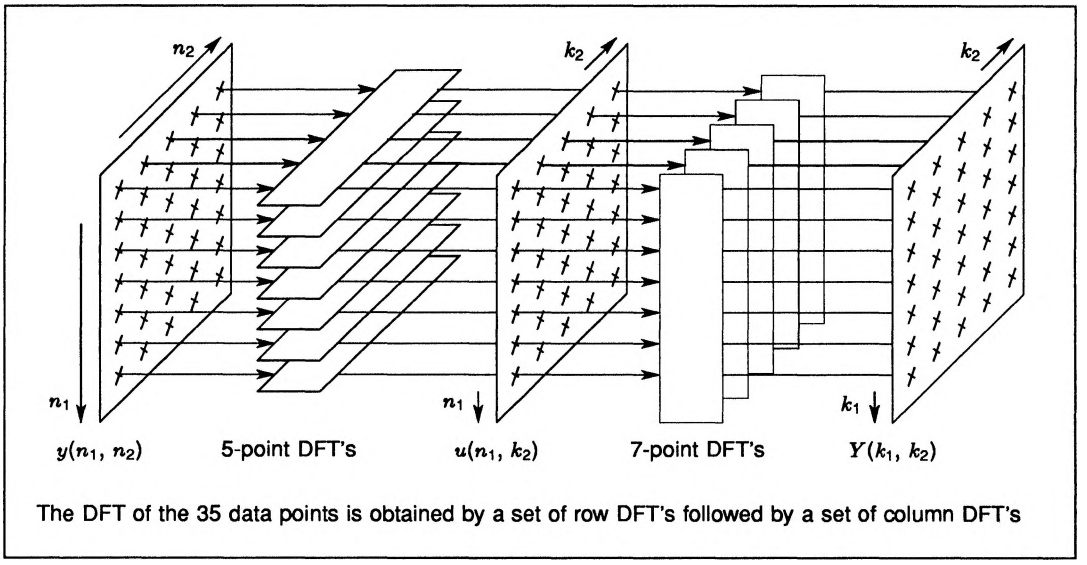

Figure 9.14 Schematic representation of equation 65 for $N = N_1 \times N_2 = 7 \times 5$

Example:

In this example we consider the evaluation of the DFT of a 35-element data array $x(n)$ ($n = 0$ *to* 34) via the mapping techniques discussed so far. Let us take $N = N_1 \times N_2 = 7 \times 5 = 35$ i.e. $N_1 = 7$ and $N_2 = 5$. The data array $x(n)$ is first mapped into a two dimensional array $y(n_1, n_2)$ via the mapping given by equation (66a) i.e.

$$n = (5n_1 + 7n_2) \; mod \; 35 \tag{67}$$

The array $y(n_1, n_2)$ would thus be as follows:

$$y(n_1, n_2) = \begin{bmatrix} x(0) & x(7) & x(14) & x(21) & x(28) \\ x(5) & x(12) & x(19) & x(26) & x(33) \\ x(10) & x(17) & x(24) & x(31) & x(3) \\ x(15) & x(22) & x(29) & x(1) & x(8) \\ x(20) & x(27) & x(34) & x(6) & x(13) \\ x(25) & x(32) & x(4) & x(11) & x(18) \\ x(30) & x(2) & x(9) & x(16) & x(23) \end{bmatrix}$$

The next step is to perform the DFT of each row of this matrix. Obviously in this example this involves seven 5-point DFT's as shown in figure 9.14. The result of these row DFT's is a new matrix denoted by $u(n_1, k_2)$.

Next the DFT of each column of the matrix u is evaluated. As shown in figure 9.14 this involves five 7-point DFT's and yields the matrix $Y(k_1, k_2)$. The two dimensional array $Y(k_1, k_2)$ contains desired DFT results $X(k)$, with the allocations governed by the mapping given by equation (66b) i.e. $X(k) = Y(k_1, k_2)$ with

$$\begin{aligned} k &= \{[(N_2^{-1}) \bmod N_1]N_2k_1 + [(N_1^{-1}) \bmod N_2]N_1.k_2\} \bmod N \\ k &= \{[3]\ 5\ k_1 + [3]\ 7\ k_2\} \bmod 35 \\ &= \{15k_1 + 21k_2\} \bmod 35 \end{aligned}$$

The array $Y(k_1, k_2)$ would therefore have the following arrangement:

$$Y(k_1, k_2) = \begin{bmatrix} X(0) & X(21) & X(7) & X(28) & X(14) \\ X(15) & X(1) & X(22) & X(8) & X(29) \\ X(30) & X(16) & X(2) & X(23) & X(9) \\ X(10) & X(31) & X(17) & X(3) & X(24) \\ X(25) & X(11) & X(32) & X(18) & X(4) \\ X(5) & X(26) & X(12) & X(33) & X(19) \\ X(20) & X(6) & X(27) & X(13) & X(34) \end{bmatrix}$$

In a practical implementation, the IMS A100 transversal filter can be used to perform these short row and column DFT's via the prime number transform algorithm described earlier. The important fact to note here is that each set of row (or column) DFT's consists of a number of totally independent short transforms (see figure 9.14). This allows various degrees of parallelism to be exploited very easily in acheiving the required specification.

For example a single A100 based DFT processor can be used to sequentially perform all the row DFT's followed by the column DFT's, or when extremeley high processing speed is essential, several such DFT processors can be employed in parallel to complete the independent row (or column) DFT's. In the extreme case, it is possible to compute all row and column DFT's concurrently in a pipelined system arrangement. The INMOS concurrent processor family (transputers) when combined with the IMS A100(s) provide an ideal environment for exploiting these algorithms.

In arriving at equation (65) we applied the conditions given by (64) to equation (63). This resulted in the mapping given by (66) on which the last example was based . It is possible to use other values for α, β, γ, and δ than those given by (64).

For example we could have used:

$$\begin{aligned} \gamma &= \delta = 1 \\ \alpha &= (N_2^{-1}) \bmod N_1 \\ \beta &= (N_1^{-1}) \bmod N_2 \end{aligned} \tag{68}$$

This would have resulted in the mappings for n and k to be interchanged i.e.

$$n = \{[(N_2^{-1}) \bmod N_1]N_2n_1 + [(N_1^{-1}) \bmod N_2]N_1.n_2\} \bmod N \tag{69a}$$

$$k = (N_2k_1 + N_1k_2) \bmod N \tag{69b}$$

Another interesting possibility is

$$\alpha = \beta = \gamma = \delta = 1 \tag{70}$$

This would result in the simple mapping for both n and k i.e.

$$n = N_2n_1 + N_1n_2 \tag{71a}$$

$$k = N_2k_1 + N_1k_2 \tag{71b}$$

but requires a modification in the W. We can see this by substituting (70) into (63) which gives:

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_{N_2}^{N_1n_2k_2} \right] W_{N_1}^{N_2n_1k_1} \tag{72}$$

By defining

$$W'_{N_2} = W_{N_2}^{N_1} = e^{-2\pi j N_1/N_2}$$

and

$$W'_{N_1} = W_{N_1}^{N_2} = e^{-2\pi j N_2/N_1}$$

Equation (72) can be rewritten as:

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W'_{N_2}{}^{n_2 k_2} \right] W'_{N_1}{}^{n_1 k_1} \tag{73}$$

This equation is very similar to (65), with the exception that a modified W is used. Equation (73) also maps into an arrangement similar to figure 9.14. The DFT's of course, would have to be calculated with the modified W. This still can be done via the prime number transforms by simply replacing W with W' in the transform.

Case 2 – common factor decomposition

Having covered the case where N_1 and N_2 were relatively prime, we now go back to equation (58) and consider the case where N_1 and N_2 have a common factor r, i.e.

$$(N_1, N_2) = r$$

Remember that we applied (59) to (58) which made the last term in (58) equal to unity i.e. we have :

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_{N_2}^{\beta L_2 n_2 k_2} \right] W_N^{M_1 L_2 n_1 k_2} W_{N_1}^{M_1 \gamma n_1 k_1} \tag{74}$$

Unlike the previous case we cannot use equation (60) to make the second terms in (74) equal to unity. This is because the equations in (60) would violate the necessary requirement, specified by (51), for one-to-one mapping.

The term $W_N^{M_1 L_2 n_1 k_2}$ is referred to as a 'twiddle factor'. Referring to (51) and remembering that we have already used the conditions given by (59), we can set

$$M_1 = L_2 = \beta = \gamma = 1 \tag{75}$$

which gives

$$Y(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} y(n_1, n_2) W_{N_2}^{n_2 k_2} \right] W_N^{n_1 k_2} W_{N_1}^{n_1 k_1} \tag{76}$$

with the mapping given by:

$$n = n_1 + N_1 n_2 \tag{77a}$$
$$k = N_2 k_1 + k_2 \tag{77b}$$

Note that equation (76) is very similar to equation (65) with the exception of the existence of the twiddle term. Equation (76) can be interpreted as shown in figure 9.15. It can be seen that when N_1 and N_2 have a common divisor, N-complex multiplications have to be performed between the row and column DFT operations. In the previous case where N_1 and N_2 were assumed to be relatively prime no such multiplications were needed making the former mapping more efficient and easier to implement.

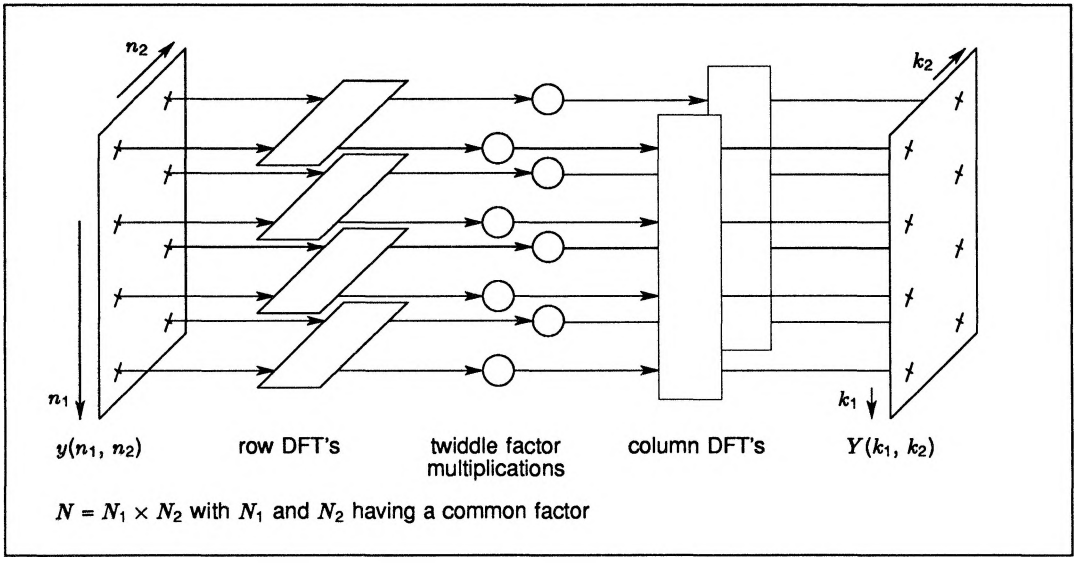

Figure 9.15 Mapping of an N point DFT into dimensions

9.5.4 Extension to multiple dimensions

The concepts presented in this application note were concentrated around a two-dimensional mapping. There is no reason why the same concepts cannot be extended to more dimensions. For example if

$$N = N_1 \times N_2 \times N_3$$

where N_1, N_2 and N_3 are relatively prime. The original N-point transform can be carried out via $N_2 \times N_3$, N_1-point, transforms followed by $N_1 \times N_3$, N_2-point, transforms followed by $N_1 \times N_2$, N_3-point, transforms. The easiest way to see this is to first map the N-point transform into a two-dimensional one with dimensions

$$N_1' = N_3 \qquad and \qquad N_2' = N_1 N_2$$

This consists of N_3, $N_1 \times N_2$-point, transforms (row DFT's) followed by $N_1 \times N_2$, N_3-point, transforms (column DFT's). Each one of the $N_1 \times N_2$-point transforms can then be decomposed into N_1, N_2-point, transforms followed by N_2, N_1-point, transforms.

Note that these multidimensional index mappings apply to both prime factor and common factor decompositions. In fact radix-2 FFT is nothing more than a common factor decomposition where all the factors N_1, N_2, N_3, N_4..... are made equal to 2. The advantage of the prime factor over the common factor decomposition is in that no twiddle matrix multiplications are needed for the prime factor case.

9.6 References

1 *An algorithm for the machine calculation of complex Fourier series*, Cooley J.W., Tukey J.W., Math. Comput., Vol.19, pp 297-301, April 1965.

2 *On computing the discrete Fourier transform*, Winograd S., Proc. Nat. Acad. Sci. USA, Vol.73, No. 4, pp. 1005-1006, April 1976.

3 *Discrete Fourier transforms when the number of data samples is prime*, Rader C.M., Proc. IEEE, Vol.56, pp 1107-1109, June 1968.

4 *The chirp z-transform algorithm and its application*, Rabiner L.R., et al, The Bell System Technical Journal, May-June 1969, pp 1249-1292.

5 *Index mappings for multidimensional formulation of the DFT an convolution*, Burrus C.S., IEEE Trans. on ASSP, Vol.25, June 1977, pp 239-242.

6 *DFT/FFT and convolutional algorithms-theory and implementation*, Burrus C.S., Parks T.W., Wiley-interscience Publication, New York 1985.

7 *Number theory in digital signal processing*, McClellan J.H., Rader C.M., Englewood Cliffs, NJ, Prentice-Hall Inc, 1979.

8 *The relationship between two fast Fourier transforms*, Good I.J., IEEE Trans. on Comput., Vol. C-20, pp 310-317, March 1971.