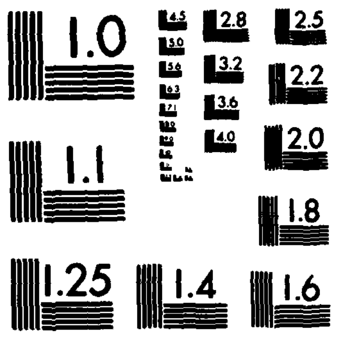
AD-A133 927 FAULT TOLERANT SIGNAL PROCESSING USING FINITE FIELDS AND ERROR-CORRECTING..(U) RENSSELAER POLYTECHNIC INST TROY NY CENTER FOR INTEGRATED ELE.. G R REDINBO JUN 83 1/2

UNCLASSIFIED AFOSR-TR-83-0839 AFOSR-80-0153 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FINAL REPORT

Fault Tolerant Signal Processing Using Finite Fields and Error-Correcting Codes

June 1983

AFOSR 80-0153

by

G. R. Redinbo

Center for Integrated Electronics
Electrical Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, N. Y. 12181
(518) 270-6034

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0839	2. GOVT ACCESSION NO. AD-A133 929	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) "FAULT TOLERANT SIGNAL PROCESSING USING FINITE FIELDS AND ERROR-CORRECTING CODES"		5. TYPE OF REPORT & PERIOD COVERED FINAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) G.R. Redinbo		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0153
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rensselaer Polytechnic Institute Center for Integrated Electronics Troy NY 12181		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A6
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NM Bldg. 410 Bolling AFB DC 20332		12. REPORT DATE June 1983
		13. NUMBER OF PAGES 106
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Finite field arithmetic may be efficiently applied for implementing signal processing architectures. The fundamental theoretical work in this area was done by Matluk and Gill (1). They showed how convolutions over a residue class ring, modulo an integer M, the situation for all implementations using a finite digital representation, may be separated into an equivalent system of serial and parallel sections. If the integer M has the prime factor expansion.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

Finite field arithmetic may be efficiently applied for implementing signal processing architectures. The fundamental theoretical work in this area was done by Matluk and Gill 9 ~~[1]~~. They showed how convolutions over a residue class ring, modulo an integer M, the situation for all implementations using a finite digital representation, may be separated into an equivalent system of serial and parallel sections. If the integer M has the prime factor expansion,

$$M = p_1^{e_i},\ p_2^{e_2},\ \ldots.,\ p_s^{e_s}\ ,\quad p_i \text{ a distinct prime,}$$

each section in this decomposition performs a convolution employing finite field arithmetic. The general decomposition is shown in Figure 1. Furthermore when M contains no primes raised to a power, this decomposition consists only of parallel sections, each using arithmetic associated with a distinctive prime. Recently these techniques have been expanded and fast algorithms for use with signal processing systems have been developed [7].

This report presents several new approaches for incorporating error-correcting codes with signal processing operations yielding fault tolerant systems. Fault tolerant levels can be distributed throughout the system's architecture. Such architectures will become necessary when very sophisticated and dense systems are implemented with Very → P3

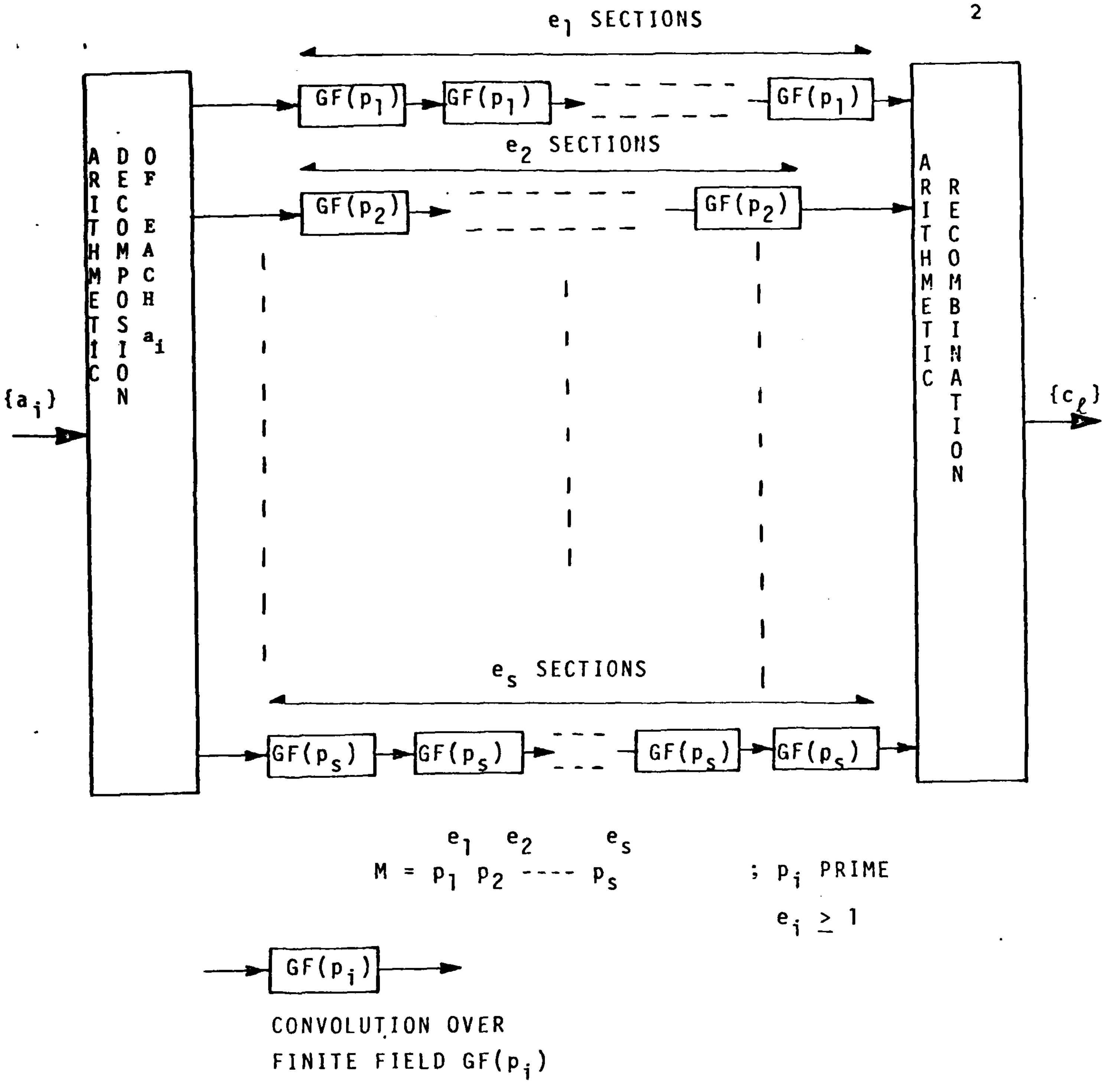


Figure 1

Serial-Parallel System Decomposition of Convolution Over a Residue Class Ring Modulo M

Large Scale Integration (VLSI).

This introduction describes the basic approach for including error-correcting codes with the convolution operation. Several architectures are developed that will protect against permanent and temporary failures in the hardware. The theoretical support for these techniques is given in succeeding sections. Further details are contained in Appendices A-F.

The desired signal processing operation is equivalent to the multiplication of two polynomials a(x) and b(x) where a(x) denotes the input sequence and b(x) corresponds to the desired filter impulse response. It will be assumed that the degree of their product does not exceed an integer, k, and that the coefficients of the polynomials are from a suitably chosen finite field, GF(q). Furthermore the proper sampling and quantization of the input and weighting sequences is considered to have been accomplished in accordance with accepted practices [6]. The required convolution operation is stated in mathematical terms as:

$$a(x)\,b(x) \qquad ; \quad \begin{array}{l} \text{degree } [a(x)b(x)] < k \\ a(x),\ b(x) \in GF(q)[x] \end{array}$$

Cyclic codes are described in terms of polynomial algebras where the code is a special subspace called an ideal [2-5]. All polynomials in the algebra are reduced in a cyclic fashion and an ideal is generated by taking multiples of a single generator polynomial. Polynomial multiplication involving an element of the code space always produces another element in the code. Cyclic codes with their error-protecting structure are ideal for digital filtering because polynomial multiplicaitons are the fundamental operations in both realms. Throughout this presentation the cyclic code will have length n and the integer k will denote the number of information digits represented in each codeword. The polynomial algebra underlying the cyclic code is defined modulo the polynomial $(x^n - 1)$, [2-5].

Suppose that one of the polynomials in the product a(x) b(x) is replaced by a codeword from the cyclic code with generator polynomial g(x). The filter sequence represented by b(x) may be encoded.

$$b(x) \rightarrow [b(x)\ g(x)] \qquad \text{ENCODING OF FILTER RESPONSE}$$

When the encoded filter response is multiplied by a(x), a codeword results. The desired filter result is the information part of the codeword since the degree of a(x) b(x) is less than k and the degree of the generator g(x) is (n -k).

$$a(x)\ [b(x)\ g(x)] = \underbrace{[a(x)\ b(x)]}_{\text{DESIRED FILTER RESULT}}\ g(x) \qquad \text{CODEWORD}$$

The polynomial a(x) represents the input to the filter, but since b(x) corresponds to a fixed filter response and g(x) is a known encoding polynomial, [b(x) g(x)] may be precomputed and stored.

Figure 2 shows the system philosophy for employing the code space. This represents the general case of a non-systematic code since the information digits are not assigned to fixed positions in the codeword [5]. The special case of systematic codes will be discussed shortly. In Figure 2, two extra subsystems that provide the protection are indicated. One is the Error Detector/Corrector which extracts the correct desired product from the output of the polynomial multiplication section while the other is an Error Status Check system which determines if errors have occurred and whether they are from the multiplication section or lie in the Error Detector/Corrector itself. Clearly it is possible for two of the three subsystems to introduce errors that go undetected. However the probability of this type of error is substantially less than an unprotected system.

Error-correcting codes were designed for detecting and correcting additive errors. When hardware failures lead to these types of errors at the system's output, the code can detect and may also be used to correct the errors. However it will also be shown that the code can detect and correct other types of errors that arise in hardware systems.

Systematic Form of the Code:

A systematic form of the cyclic code may be used in the system described above. In such a form the desired filtering operation's output appears in an unaltered format at the out-

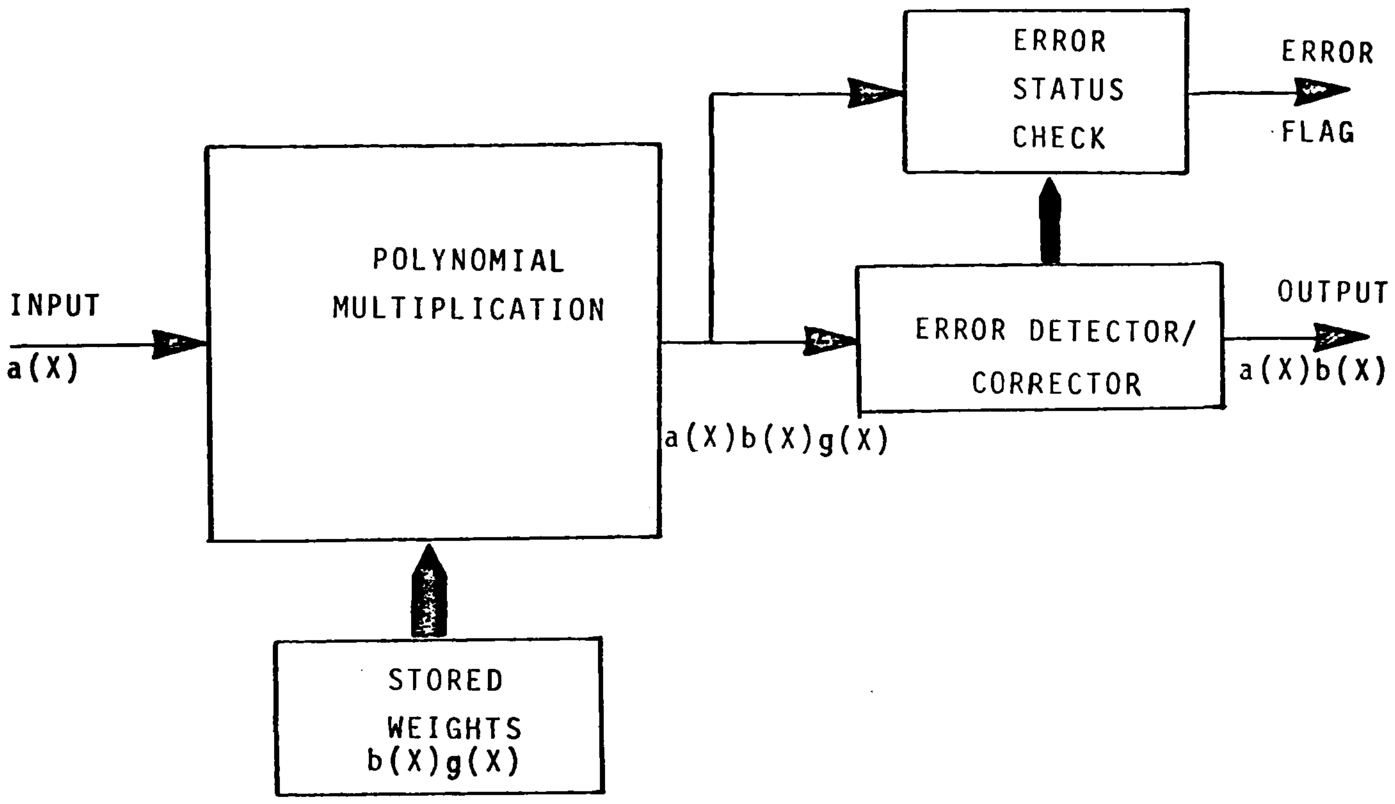


Figure 2

DICHOTOMY OF SIGNAL PROCESSING
SYSTEM EMPLOYING CONCURRENT
ERROR PROTECTION

put of the polynomial multiplication subsystem. Thus no further operations are required to extract the result if no errors have been made. However in this case two parallel similar subsystems are required. One subsystem calculates the direct filter output while the other computes the corresponding parity-check digits.

For the systematic form the stored encoded filter weights, labeled $b(x)g(x)$ in Figure 2, are computed differently. The term $x^{n-k}b(x)$ is reduced modulo $g(x)$ to obtain the parity-check portion of the codeword denoted by $[-p(x)]$. The Euclidean Division Algorithm when applied to $x^{n-k}b(x)$ using $g(x)$ as the divisor [5] gives:

$$x^{n-k}b(x) = f(x)g(x) + p(x) \; ; \; \deg p(x) \leq \deg g(x) = n-k \qquad (1)$$

The polynomial $[x^{n-k}b(x) - p(x)]$ is a codeword, and more importantly the parity-check digits represented by $[-p(x)]$ are confined to the lower ordered $(n-k)$ coefficients. On the other hand the weighting contained in $b(x)$ appears in the higher ordered k positions. Thus the codeword may be separated into two polynomials, one representing the parity-checks and the other, the desired weighting.

$$\underbrace{[x^{n-k}b(x)]}_{\substack{\text{FILTER}\\ \text{WEIGHTING}\\ \text{PORTION}}} + \underbrace{[-p(x)]}_{\substack{\text{PARITY-CHECK}\\ \text{PART}}} = \text{CODEWORD}$$

When multiplied by $a(x)$ the desired result may be segregated into two portions as shown in Figure 3 where two similar subsystems are required. Only linearity is necessary to insure this separation into two parallel subsystems. Since the degree $[a(x)b(x)]$ is less than k, the digits in $[x^{n-k}a(x)b(x)]$ correspond to the desired filter output. On the other hand though $[-a(x)p(x)]$ may have degree greater than $(n-k-1)$. These digits are only necessary when the error detection and correction systems are employed. The two parts need only be combined before entering the error control subsystems as shown in Figure 3.

The systematic and nonsystematic code forms lead to similar architectures. In developing the general system architecture the nonsystematic form will be considered. The basic principles are applicable in both situations, and only minor modifications are necessary for the systematic case. The system shown in Figure 2 will be studied in detail with particular attention devoted to the subsystem labeled polynomial multiplication.

Polynomial Multiplication Using Component Polynomials:

Associated with the code space are irreducible polynomials $g_i(x)$, $i = 1, 2, \ldots, t$, where the number t is determined by the code space. These polynomials are factors of the parity-check polynomial, $h(x)$.

$$h(x) = \prod_{i=1}^{t} g_i(x)$$

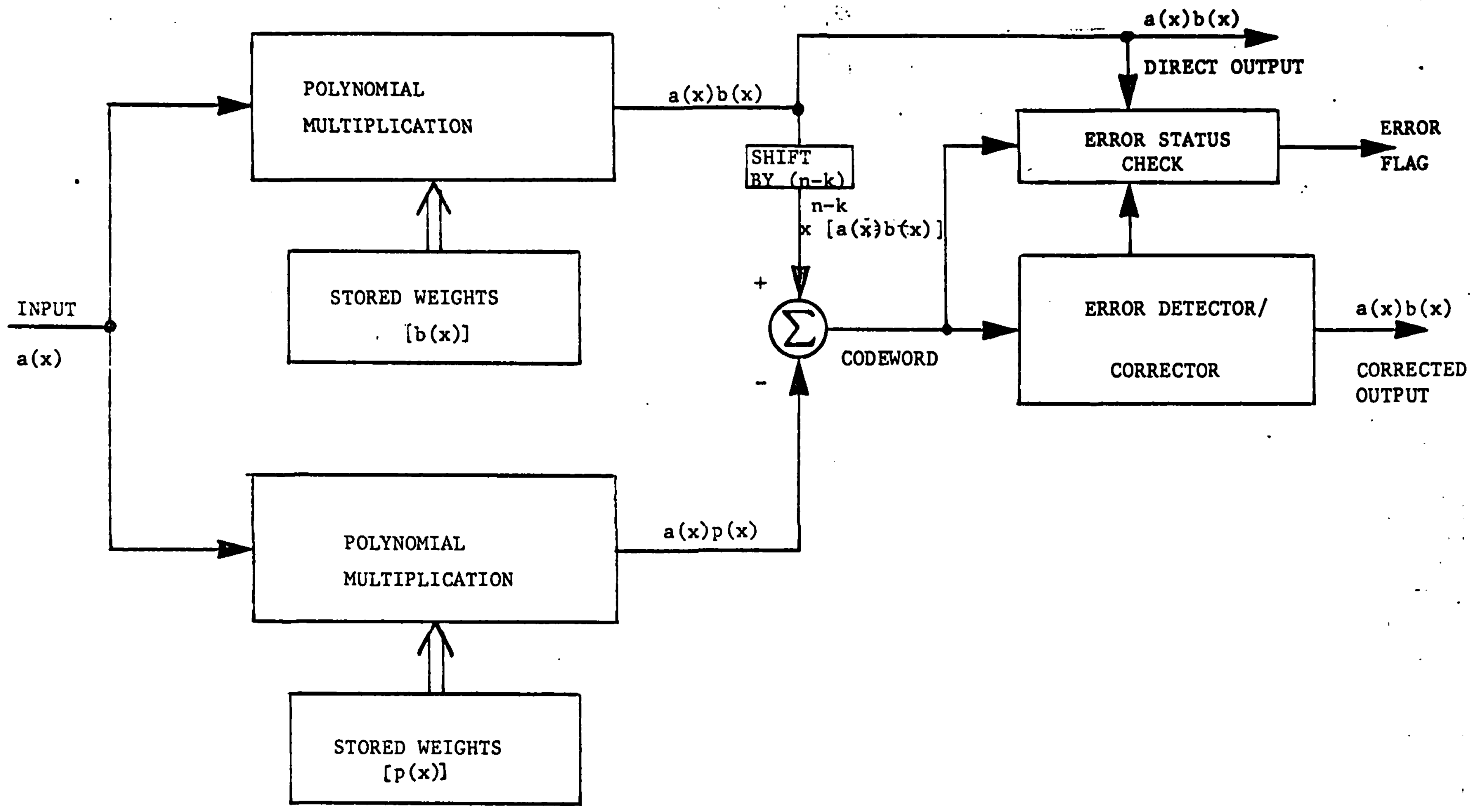


Figure 3

DIGITAL WEIGHTING SYSTEM USING A SYSTEMATIC CODE

Each irreducible factor in turn defines an idempotent $e_i(x)$, with the defining property:

$$e_i^2(x) = e_i(x) \quad ; \; i = 1, 2, \ldots, t \tag{3}$$

These idempotents permit an expansion of $[a(x)b(x)g(x)]$ using component factors modulo the individual factors $g_i(x)$, $i = 1, 2, \ldots, t$.

$$a(x)\; b(x)\; g(x) = \sum_{i=1}^{t} a_i(x)\; b_i(x)\; e_i(x) \tag{4}$$

where

$$a_i(x) \equiv a(x) \bmod g_i(x) \tag{5}$$

and

$$b_i(x) \equiv b(x)\; g(x) \bmod g_i(x) \tag{6}$$

The polynomial multiplication subsystem of Figure 2 may be replaced by t parallel subsystems each involving component multiplications of the respective factors $a_i(x)$ and $b_i(x)$ $e_i(x)$. The t results are recombined according to equation (4).

The terms $b_i(x)$ represent fixed weights which may be stored. The theory shows that each component multiplication $a_i(x)\; b_i(x)$ is equivalent to a product of corresponding field elements from a finite field defined by the irreducible factor $g_i(x)$. The field representation of this product addresses shifted versions of the idempotent $e_i(x)$, and a sum of the necessary versions gives the factor $[a_i(x)b_i(x)e_i(x)]$. This is shown in Figure 4a where the reduction modulo $g_i(x)$ is an integral part of the field operations. The t products each generated in this way are combined according to equation (4). The complete system appears in Figure 4b.

Use of the Transform Domain:

An alternate implementation of the polynomial multiplications can be performed in a transform domain. The mapping into this domain involves a primitive n th root of unity α which lies possibly in an extension field which will be denoted by $GF(q^m)$, $m>1$. However under certain conditions it may be possible to choose n so that the primitive root, α falls in the base field GF(q). The polynomial multiplications of the component polynomials may be transformed to an equivalent series of multiplications of the related transform coefficients. The inverse transform of these results gives the desired product. The number of nonzero transform coefficients needed to implement the product $a_i(x)\gamma_i(x)$ is less than the degree of the correspondingly indexed irreducible factor $g_i(x)$. Furthermore the transform coefficients may be determined by forming successive q th powers of a single transform coefficient. The realization of this device is simpler than a general multiplier system for the coefficient field $GF(q^m)$.

Figure 5 shows the transform domain approach to polynomial multiplication. The transform coefficients affiliated with the component polynomials $a_i(x)$ are labeled $[\hat{a}_{ij}]$ while those corresponding to the $\gamma_i(x)$ components, by $[\hat{\gamma}_{ij}]$. The number

IF ℓ_i = DEGREE $[g_i(X)]$, COMPONENTS

$a_i(X)$, $b_i(X)$ BEHAVE AS IF FROM GF(p^{ℓ_i})

$[a_i(X)b_i(X) \bmod g_i(X)] \Longleftrightarrow$ FINITE FIELD MULTIPLICATION

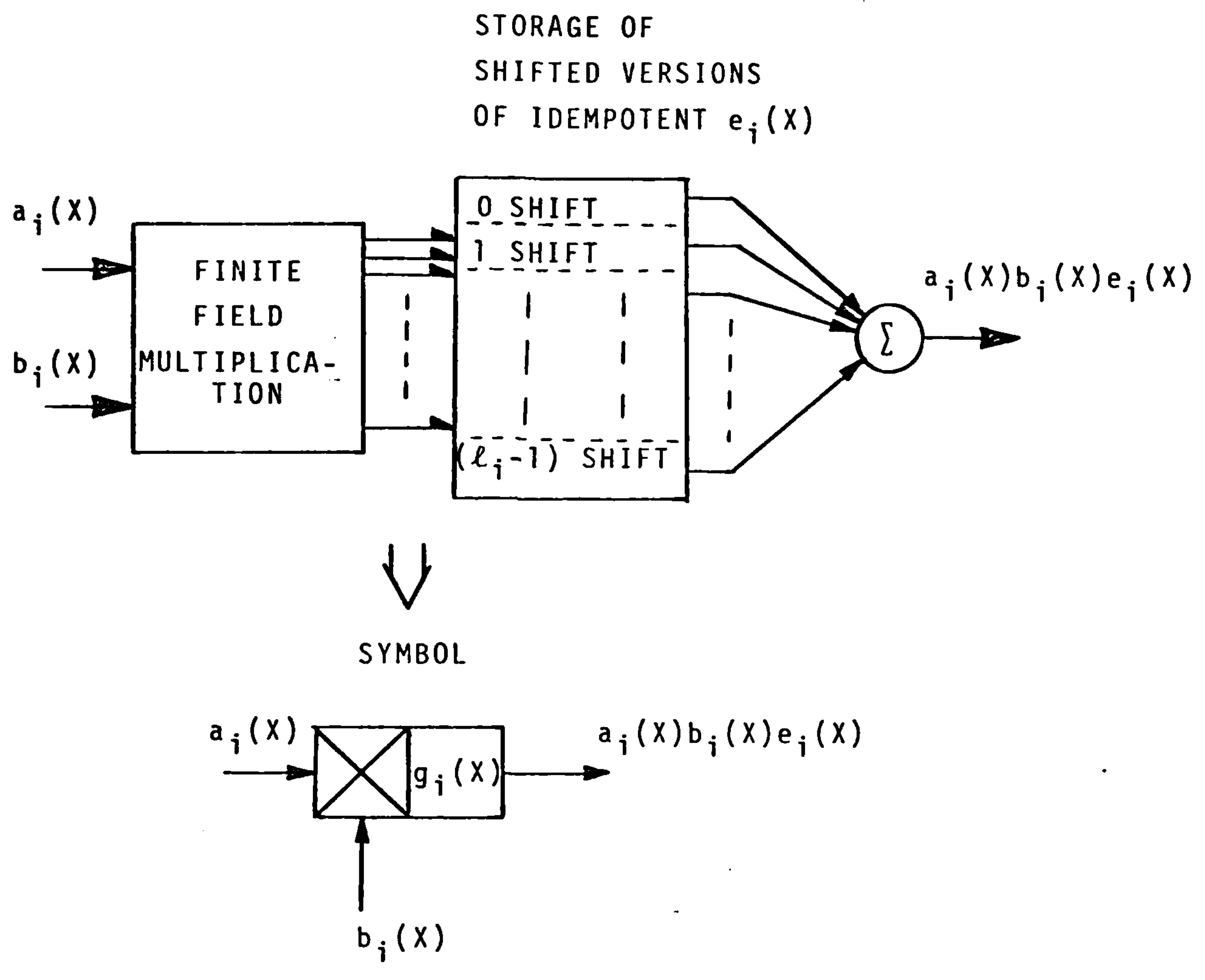


Figure 4a
Multiplication of One Components

Figure 4
Polynomial Multiplication Using Components

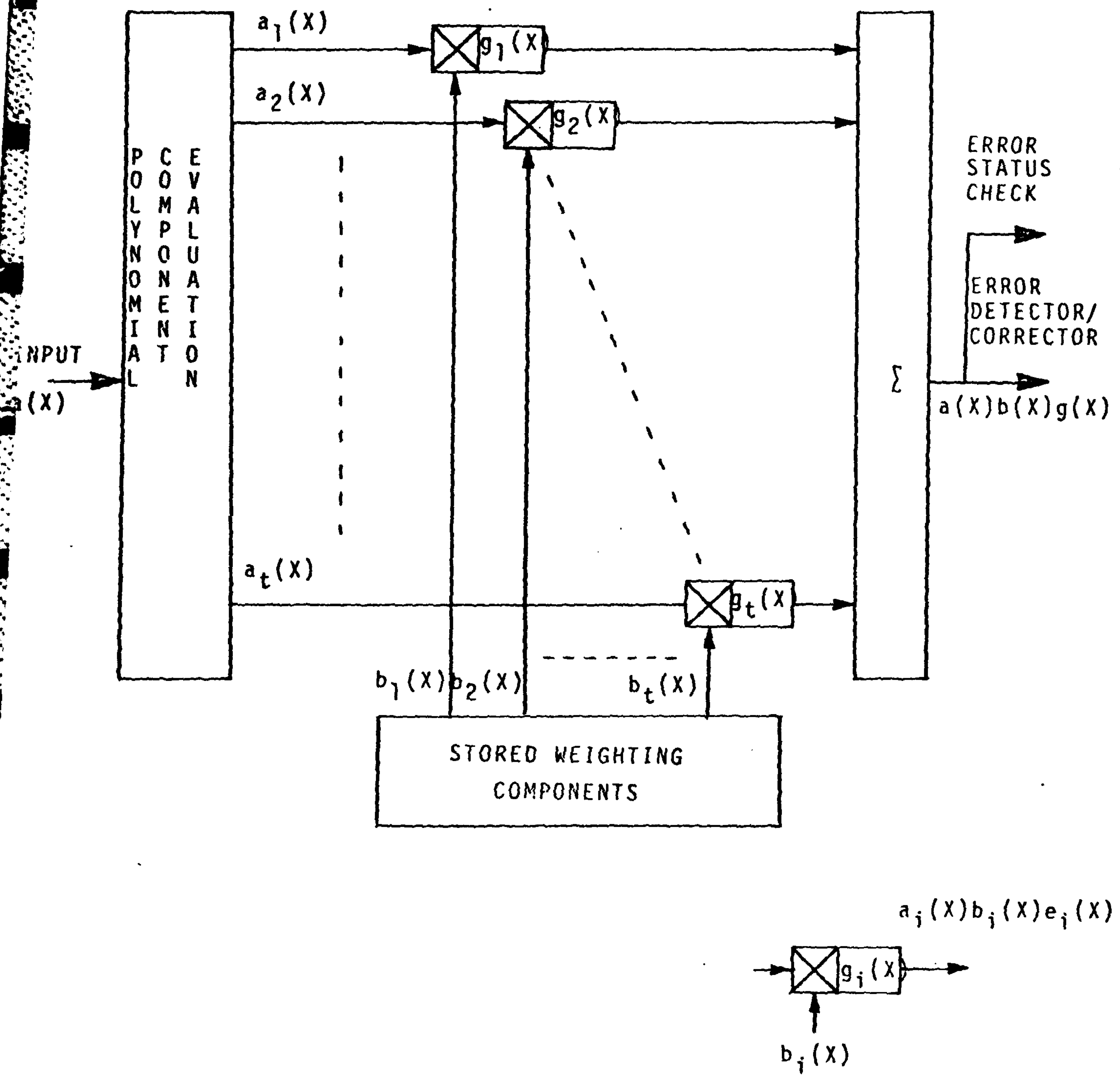
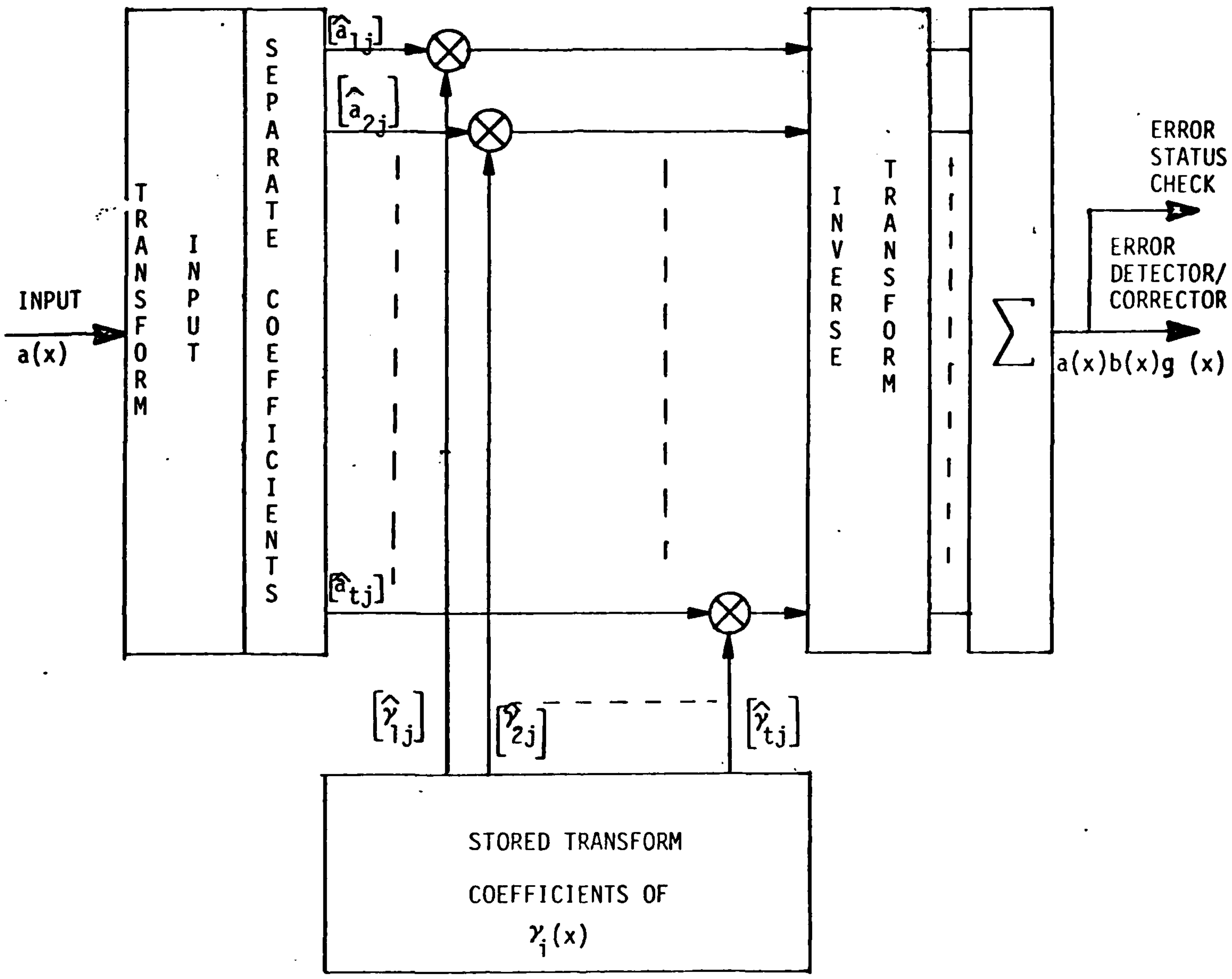


Figure 4b
Combined System



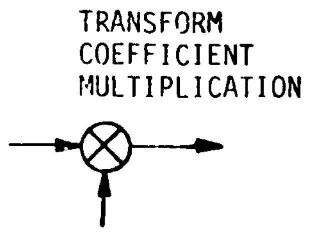


Figure 5

TRANSFORM DOMAIN IMPLEMENTATION OF POLYNOMIAL PRODUCT of nonzero terms in $[\hat{\gamma}_{ij}]$ is guaranteed to be less than the degree of $g_i(x)$ and so only coefficients in the list, $\{a_{ij}\}$, with similar indices need to be determined. The locations of the nonzero terms are known in advance and are determined by index numbers in certain subsets of integers, the cyclotomic subsets modulo n.

Space Decompositions

The section describes the component decompositions of the code space. The general nature of minimal ideal expansions is presented first followed by the application of minimal ideals to cyclic codes. The basic properties displayed in this section are also used in certain aspects of the transform domain implementation. All properties are stated without proofs but the supporting developments are relagated to the appendices.

Minimal Ideal Decompositions:

If the code length n and the size of the ground field q are relatively prime then the polynomial $(x^n - 1)$ may be factored into distinct irreducible factors. (Say for definiteness that there are T factors.)

$$(x^n - 1) = \prod_{i=1}^{T} g_i(x) \quad ; \begin{array}{l} (n,q) = 1 \\ g_i(x) \text{ irreducible} \end{array} \tag{7}$$

The ring of polynomials GF(q) [x], reduced modulo $(x^n - 1)$ is a residue class ring in which the minimal ideals are defined by the irreducible factors in equation (7). The following notational conventions are introduced.

$$A_n = GF(q)\ [x] / \left\|x^n - 1\right\| \quad ; \begin{array}{l} \text{Residue Class Ring} \\ \left\|f(x)\right\| \text{ denotes principal} \\ \text{ideal generated by } f(x). \end{array} \tag{8}$$

$$m_i(x) = \frac{(x^n - 1)}{g_i(x)} \quad ; \quad \text{MINIMAL IDEALS} \qquad (9)$$

$$M_i = \left\|\left(m_i(x)\right)\right\| \qquad i = 1, 2, \ldots, T$$

The fact that each $m_i(x)$ as defined in equation (9), generates a minimal ideal is demonstrated in Appendix A.

Each minimal ideal M_i also contains an idempotent $e_i(x)$ defined by considering units modulo $g_i(x)$ in the minimal ideal M_i. It also generates the same principal ideal. An idempotent obeys the following identity

$$e_i^2(x) = e_i(x) \qquad \text{IDEMPOTENT} \qquad (10)$$

The idempotents and their properties play a central role in the results to follow. For example the components necessary for a direct sum expansion of any element $a(x) \in A_n$ can be easily determined using an orthonormal property.

$$a(x) = \bigoplus_{i=1}^{T} a_i(x)\; e_i(x) \quad ; \quad \oplus \text{ Denotes a direct sum expansion} \qquad (11)$$

$$a_i(x) = a\,(x) \bmod g_i(x); \quad i=1, 2, \ldots, T$$

<u>Minimal Ideals and Cyclic Codes:</u>

A cyclic code is a principal ideal in A_n generated by a polynomial $g(x)$. Furthermore $g(x)$ divides $(x^n - 1)$ and also defines the parity-check polynomial, $h(x)$.

$$(x^n - 1) = g(x)\; h(x) \qquad (12)$$

Since $(n,q) = 1$, there are no repeated factors in $(x^n - 1)$, and $g(x)$ and $h(x)$ contain only distinct irreducible factors. The factors of $h(x)$ will be indexed by $i = 1, 2, \ldots, t$ and those of $g(x)$ by $j = t+1, t+2, \ldots, T$.

$$h(x) = \prod_{i=1}^{t} g_i(x)$$

$$g(x) = \prod_{j=t+1}^{T} g_j(x) \tag{13}$$

The degrees of these polynomials determine the permissible numbers of parity-check digits in each possible code contained within A_n.

$$\text{degree } g(x) = n-k \quad ; \quad \text{degree } h(x) = k \tag{14}$$

The code generated by $g(x)$ will be labeled by G; it is a k-dimensional subspace of A_n (and by construction also a principal ideal).

$$G = ((g(x))) \tag{15}$$

The code space G may be expressed in terms of the t minimal ideals defined by the respective factors $g_i(x)$, $i = 1, 2, \ldots, t$.

$$G = \bigoplus_{i=1}^{t} M_i \tag{16}$$

It is possible to define an idempotent generator for the code space G. If cascaded sections use the same code space, the idempotent property of this generator could be used to guarantee that no decoding would be necessary until the final section. This concept will not be pursued further here, but such possibilities for cascaded configurations could be useful.

The Transform Domain

Definitions and Properties:

The degrees of the polynomials $a(x)$ and $b(x)$, involved in the weighting operation, determine the parameter k while the desired level of error protection determines the size of n for the code. The choices of the pairs of integers n and k are limited by the irreducible factors of $(x^n - 1)$. (See equations (7), (12) and (13).) The field $GF(q)$ may be extended to a larger field $GF(q^m)$ and if n divides $(q^m - 1)$ there will be a primitive n th root of unity in $GF(q^m)$. Let α denote this root. (If n is $(q - 1)$ the root α will lie in the original field $GF(q)$.)

The irreducible polynomials $g_i(x)$ which are factors of $(x^n - 1)$ are constructed using powers of this root α and subsets of the integers modulo n. These cyclotomic subsets index the conjugate roots in $GF(q^m)$ of the irreducible factors. A typical cyclotomic subset Λ_s consists of the integers modulo n of the form $\{s, sq, sq^2, \ldots, s\, q^{\iota-1}\}$ where s and ι are related by the identity $sq^{\iota} \equiv s \bmod n$. s is generally taken as the smallest positive integer in the cyclotomic subset.

$$\Lambda_s = \{s, sq, sq^2, \ldots, sq^{\iota-1}\} \qquad (17)$$
$$\iota,\ s \text{ defined by } s \equiv sq^{\iota} \bmod n$$

Each subset is constructed successively and inductively by selecting an integer, s, modulo n, that is not contained in a previously considered subset and forming each integer s q^j until the identity $s \equiv s\, q^{\ell} \bmod n$ is satisfied. The integers modulo n are partitioned by the cyclotomic subsets. These subsets define irreducible factors $g_i(x)$.

$$g_i(x) = \prod_{j \,\varepsilon\, \Lambda_s(i)} (x-\alpha^j) \qquad ; \Lambda_s^{(i)} \text{ some cyclotomic-subset } i = 1, 2, \ldots, T$$

The degree of $g_i(x)$ is defined by the size, ℓ_i, of the respective cyclotomic subset $\Lambda_s^{(i)}$.

A transform may be defined for the space A_n. It uses the n th root of unity, α, and takes values in $GF(q^m)$ as determined by the choice of this root. The forward transform produces n coefficients in the larger field $GF(q^m)$.

$$\hat{\gamma}_j = \gamma(\alpha^j) \qquad ; j = 0, 1, 2, \ldots, (n-1)$$

$$\gamma(x) = \gamma_0 + \gamma_1 x + \gamma_2 x^2 + \text{--} + \gamma_{n-1} x^{n-1} \quad ; \gamma(x) \;\varepsilon\; A_n \tag{18}$$

The inverse transform has a similar form and is normalized by an element in the prime subfield GF(p) where $q = p^r$, p a prime[1].

$$\gamma_{\ell} = \left(\left(n^{-1}\right)\right)_p \sum_{j=0}^{n-1} \hat{\gamma}_j \, \alpha^{-\ell j} \qquad (19)^1$$

[1] $\left(\left(n^{-1}\right)\right)_p$ denotes the inverse of n modulo p. Note that since $(n,q) = (n,p) = 1$ such an inverse element exists; there are integers x and y such that $xn + yp = 1$; $x = n^{-1} \bmod p$.

The validity of this transform pair is established in Appendix C.

This transform pair has the usual and important relationship between convolution in one domain and pointwise sequence products in the other. In symbols this relationship is:

$$\begin{array}{lll} a(x) \longleftrightarrow \{\hat{a}_j\} & ; & \hat{a}_j = a(\alpha^j) \\ & & \quad j = 0, 1, \ldots, n-1 \\ \gamma(x) \longleftrightarrow \{\hat{\gamma}_j\} & ; & \hat{\gamma}_j = \gamma(\alpha^j) \\ a(x)\,\gamma(x) \longleftrightarrow \{\hat{a}_j\hat{\gamma}_j\} & & \end{array} \qquad (20)$$

CONVOLUTION $\longleftrightarrow$ POINTWISE MULTIPLICATION

This relationship is also demonstrated in Appendix C.

Effects of the Code in Transform Domain:

The transform domain will be used in performing the multiplication of the component factors $a_i(x)$ and $b_i(x)$ $e_i(x)$ as required in each sum term of equation (4). The polynomial $a_i(x)$ is defined in equation (5) while $b_i(x)$ is given in equation (6). For convience the stored components will be labeled by $\gamma_i(x)$.

$$\gamma_i(x) = b_i(x)\ e_i(x) \qquad ; \ i = 1, 2, \ldots, t \qquad (21)$$

The transform coefficients, $\hat{\gamma}_{ij}$, associated with each $\gamma_i(x)$ have an important property.

$$\hat{\gamma}_{ij} = \gamma_i(\alpha^j) = b_i(\alpha^j)\ e_i(\alpha^j) = \begin{cases} b_i(\alpha^j) & ;\ j \in \Lambda_s^{(i)} \\ 0 & ;\ j \notin \Lambda_s^{(i)} \end{cases} \quad \begin{array}{l} ;\ i = 1, 2, \ldots, t \\ ;\ j = 0, 1, \ldots, (n-1) \end{array} \tag{22}$$

They are nonzero only on the cyclotomic subset $\Lambda_s^{(i)}$ because $e_i(x)$, involves all irreducible factors except $g_i(x)$. The required polynomial product $a_i(x)\ \gamma_i(x)$ corresponds to a pointwise product in the transform domain using only those transform coefficients of $a_i(x)$ that have indices in the cyclotomic subset $\Lambda_s^{(i)}$. They are the only ones that can yield a nonzero product.

On the other hand the roots α^{sq^j} associated with each cyclotomic subset may be viewed as an individual piece of the transform where the number of elements in the cyclotomic subset $\Lambda_s^{(i)}$ will be denoted by ℓ_i. The stored weights for the i th component $\gamma_i(x)$ are represented by the ℓ_i nonzero transform coefficients

$$\hat{\gamma}_{ij} \quad , j \in \Lambda_s^{(i)} \ , i = 1, 2, \ldots, t.$$

Note the total number of nonzero coefficients is $\sum_{i=1}^{t} \ell_i = k$. The transform coefficients of the input components $a_i(x)$ need to be determined only for the same set of indices. The pointwise product is formed and the inverse transform using only the roots defined by the respective sets $\Lambda_s^{(i)}$, yields the $a_i(x)$ $\gamma_i(x)$ products.

$$a_i(x)\ \gamma_i(x) = \left\|n^{-1}\right\|_p \sum_{m=0}^{(n-1)} x^m \left[\sum_{j\ \varepsilon\ \Lambda_s^{(i)}} \hat{\gamma}_{ij}\ \hat{a}_{ij}\ \alpha^{-mj} \right] \qquad (23)$$

$$; i = 1, 2, \ldots, t$$

Recall that the $\{\hat{a}_{ij}\}$ coefficients are determined from the input polynomial a(x). The final result [a(x)b(x)g(x)] is given by the sum of these products according to equation (4).

<u>The Transforms of the Polynomial Components</u>:

A property of the transform pieces relates the zeros of the polynomial with the location of the zero transform coefficients and conversely. Consider a typical piece of the transform of component polynomial, say $c_i(x)$, over its related cyclotomic subset $\Lambda_s^{(i)}$.

$$\begin{aligned} c_{i,j} &= c_i(\alpha^j) \qquad ; j\ \varepsilon\ \Lambda_s^{(i)},\ i = 1, 2, \ldots, t \\ &= \sum_{r=0}^{n-1} c_{i,r}\ \alpha^{sq^\ell r} \qquad ; j = sq^\ell \\ &\qquad\qquad\qquad \ell = 0, \ldots, (\ell_i - 1) \end{aligned} \qquad (24)$$

where $c_i(x) = c_{i,0} + c_{i,1}\, x + c_{i,2}\, x^2 + \ldots + c_{i,(n-1)}\, x^{n-1}$. The size of cyclotomic subset $\Lambda_s^{(i)}$ is designated by ℓ_i.

A useful result which is a special case of a general property for elements of A_n is from a Lemma in Appendix D.

Annihilator Property

$c_i(x)$ in A_n has as roots all α^j except for $j \in \Lambda_s^{(i)}$ if and only if the transform coefficients $\hat{c}_{ij} = 0$ for all $j \notin \Lambda_s^{(i)}$.

In another vein, since the polynomials in A_n have coefficients in GF(q) while the transform may take values in GF(q^m) there is a constraint on the values of the respective coefficients. This leads to the Chord Property of the transform coefficients.

Chord Property

$c_i(x) \; \varepsilon \; GF(q)\,[x]$ if and only if $(\hat{c}_{ij})^q = \hat{c}_{i|jq|}, \; j \; \varepsilon \; \Lambda_s^{(i)}$

This is also a special case of another general result from Appendix D. The Chord Property simplifies the computation of the transform coefficients affiliated with each polynomial component. The piece of the transform indexed by the cyclotomic subset $\Lambda_s^{(i)}$ can be computed by repeatedly forming the q th powers of a single coefficients. This continues until all exponents from $\Lambda_s^{(i)}$ have been used.

Calculation in the Transform Domain:

The product [a(x) b(x) g(x)] can be realized principally in the transform domain. An inverse transform yields the final result (See equation (23).) However as was noted earlier the product of transform coefficients $\hat{\gamma}_{ij}\;\hat{a}_{ij}$, corresponding with the polynomial product $\gamma_i(x)\; a_i(x)$, i=1, 2, ..., t are nonzero only in nonoverlapping segments in the transform domain. Thus it is important to investigate the properties of the inverse transform of a sequence of transform coefficients indexed by a single cyclotomic subset.

In this regard let $\delta(x)$ denote the sum below and let $\hat{\Delta}_r$ label its transform coefficients which were produced by multiplying the terms $\hat{\gamma}_{ij}$ and $\hat{a}_{ij}$ indexed by the cyclotomic subset $\Lambda_s^{(i)}$.

$$\delta(x) = \sum_{i=1}^{t} a_i(x)\, \gamma_i(x)$$

$$\hat{\Delta}_j = \hat{\gamma}_{ij}\, \hat{a}_{ij} \qquad ; \; j \in \Lambda_s^{(i)} \quad i = 1, 2, \ldots, t \tag{25}$$

The Annihilator Property shows that the transform coefficients, $\hat{\Delta}_j$, are zero for indices in the cyclotomic subsets related to the irreducible polynomial factors of g(x). The combined effect on these transform coefficients may be stated as:

$$\hat{\Delta}_j = 0 \qquad ; \; j \in \Lambda^{(g)}$$

$$\Lambda^{(g)} = \bigcup_{j=t+1}^{T} \Lambda_s^{(j)} \tag{26}$$

The remainder of the indices are contained in the complement set, $\Lambda^{\overline{(g)}}$, defined by the set difference operator in the following.

$$\Lambda^{\overline{(g)}} = \left[\{0,1,2,\ldots,(n-1)\} - \Lambda^{(g)}\right] \tag{27}$$

The inverse transform determines the coefficients of $\delta(x)$.

$$\delta_j = \left(\left(n^{-1}\right)\right)_p \sum_{r \in \Lambda^{\overline{(g)}}} \hat{\Delta}_r\, \alpha^{-jr} \; ; \; j=0,1,2,\ldots,(n-1) \tag{28}$$

Let Ψ denote a generic root of g(x). It may be expressed as a power of α using the appropriate index from $\Lambda^{(g)}$.

$$\Psi = \alpha^{sq^{\ell}} \quad ; \; sq^{\ell} \in \Lambda^{(g)} \tag{29}$$

The general result, Lemma D1, Appendix D, shows that ψ is a root of $\delta(x)$, and if $\delta(x) \in A_n$, the closure property of the individual cyclotomic subsets comprising the subset $\Lambda^{\overline{(g)}}$ coupled with the general chord property Lemma D2 gives that

$$\hat{\Delta}_r^q = \hat{\Delta}_{rq} \quad \text{for all } r \in \overline{\Lambda^{(g)}} \qquad (30)$$

Then g(x) divides $\delta(x)$ and the error detector/corrector subsystem will be unable to distinguish any errors that were introduced in $\delta(x)$.

A word of caution is in order. Care must be taken to insure that any hardware implementation does not satisfy the two conditions (26) and (30), thus preempting the error detecting capability of the code. For example implementing relationship (30) in the architecture to avoid computing all of the products $[\hat{\gamma}_{ij}\ \hat{\alpha}_{ij}]$ as prescribed in equation (25) could mask errors that are introduced by hardware failures. Fortunately the transform domain contains redundancy that protects against such errors.

<u>Error Protection in The Transform Domain:</u>

If an error occurs in the transform domain, numerous errors can be introduced in the polynomial components. Their number could exceed the error-detecting or correcting capability of the code. Two general sources of errors in the transform domain that arise internal to the architecture are of concern. On the other hand any errors appearing momentarily in a straightforward implementation of the inverse transform operation affect only one polynomial coefficient which may be treated as a normal additive error. Further it can be safely assumed that the stored transform coefficients are held in the memory correctly.

One source of internal errors is the calculation of the transform coefficients from the input a(x), i.e., the $\hat{a}_{ij}$ terms may be incorrectly computed. The second form of errors results from the componentwise multiplication in this transform domain. These products $\{\hat{a}_{ij}\ \hat{\gamma}_{ij}\}$ are necessary in the inversion formula (23) and can propagate numerous errors into the A_n domain. It is possible to model both types of errors by considering noise transform coefficients $\hat{\varepsilon}_{ij}$ along with the desired products $\hat{c}_{ij}$.

$$\hat{d}_{ij} = \hat{c}_{ij} + \hat{\varepsilon}_{ij} \qquad ; \begin{array}{l} i = 1, 2, \ldots, t \\ j \in \Lambda_s^{(i)} \end{array} \tag{31}$$

Most reasonable hardware implementations using the transform domain will employ an inversion formula involving a sum only over the individual cyclotomic subset as shown in equation (23). Therefore the polynomials resulting from the inversion of the product terms will obey the Annihilator Property. Thus it is reasonable to assume that the Annihilator Property is satisfied. However it is shown in Appendix E that if the noise transform coefficients $\hat{\varepsilon}_{ij}$ do <u>not</u> obey the Chord Property, the result of the inversion formula will yield at least one polynomial coefficient with a value strictly in the extension field $GF(q^m)$. Then of course the polynomial $d_i(x)$ obtained from $\{\hat{d}_{ij}\}$ has at least one coefficient also in $GF(q^m)$, a fact that is easy to detect.

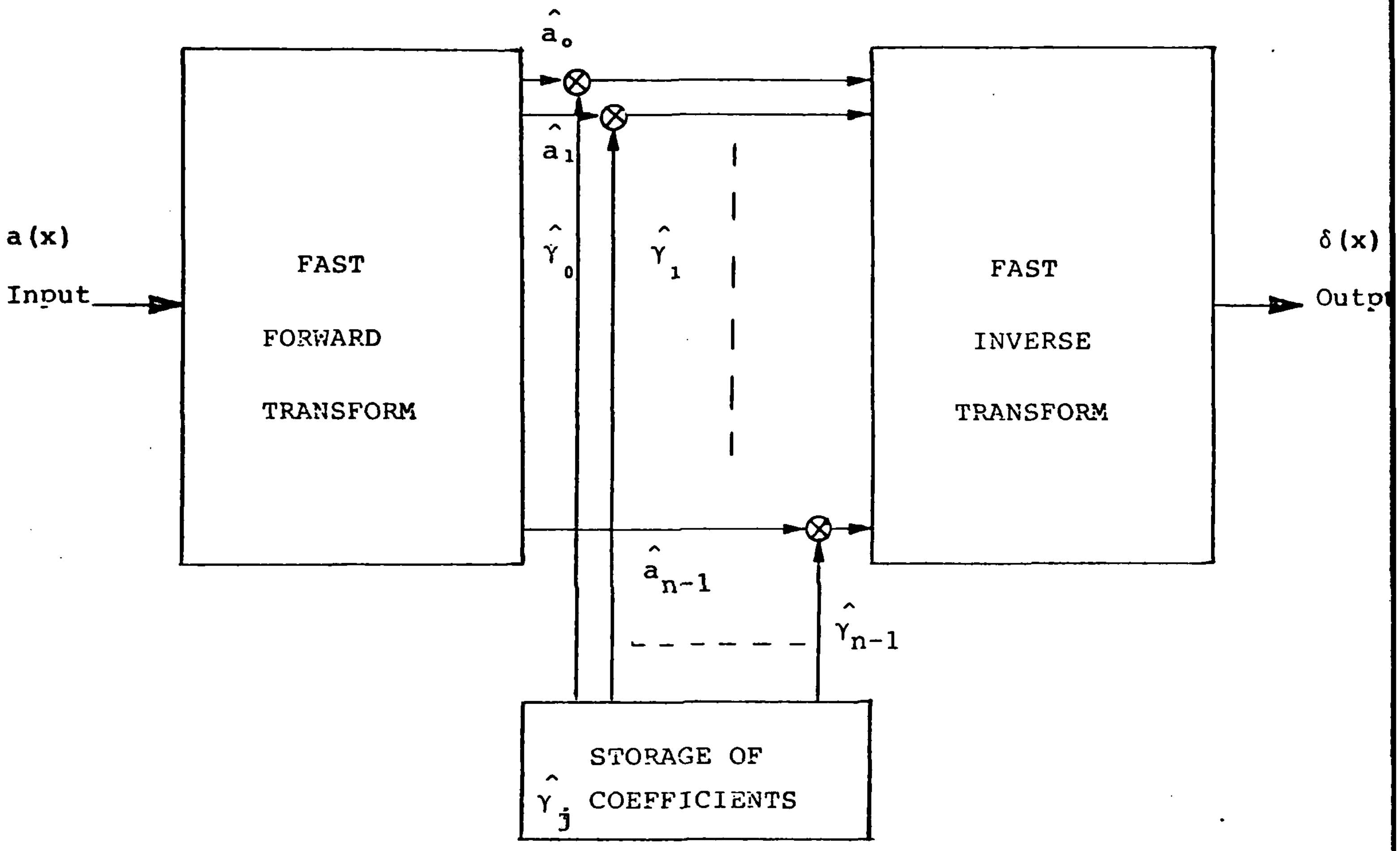
It is shown in Appendix E that the error-detecting capability of this approach allows up to $(\ell_i - 1)$ transform coefficients in error where the cyclotomic subset $\Lambda_s^{(i)}$ has ℓ_1 indices. Error detection is easily implemented by checking that the inversion formula (23) only produces elements in GF(q). There is an error-correcting technique also available. It is outlined and discussed in Appendix E. The sequence of transform coefficients $\hat{d}_{ij}$ is compared with ℓ_i different test sequences. Each test sequence is generated by forming all $q^{\underline{th}}$ powers of each $\hat{d}_{ij}$ element. If a coefficient obeys the Chord Property then it is correct; otherwise it can be replaced using the Chord Property and known correct elements in the $\{\hat{d}_{ij}\}$ sequence. This scheme will correct up to $\frac{\ell_i - 1}{2}$ errors in the transform coefficients. It also can correct errors introduced by the implementation of the scheme itself.

The error-protecting levels in the transform domain are directly related to the size ℓ_i of each cyclotomic subset $\Lambda_s^{(i)}$. Thus the choice of error-correcting codes for digital filtering applications should have components with large cyclotomic subsets (or equivalently large degrees of the affiliated irreducible polynomials). However the overall efficiency of the space decompositions depends on having numerous but small component factors. The classic trade-off between complexity and error-protecting capability is encountered.

Imbedding Error Protection in Fast Transform Implementation

The classical method of using the transform domain employs fast transform algorithms for producing the frequency domain coefficients which in turn are producted with the stored weighting coefficients, this result being inverted by another fast algorithm [6]. This traditional approach is depicted in Figure 6. However the transform coefficients in this situation are not sorted according to their minimal ideal index as shown in Figure 5. The fast transform method has a speed advantage especially when pipelining is incorporated in the realization. A major disadvantage is the potential for internally generated errors to propagate extensively throughout the implementation. In fast transform algorithms each intermediate variable at one stage is used in several intermediate variables at the succeeding stage. Thus even single errors can contaminate many final output values.

The purposes of this section are to develop fast forward and inverse transforms, analyze the propagation and penetration of internally generated errors and demonstrate methods and restrictions which insure safe operation of the overall system shown in Figure 6. The techniques described below will concentrate on containing and controlling the effects of single isolated internal errors, primarily through error detection followed by re-execution of the defective step.



$$\delta(x) = a(x)\gamma(x)$$

TRANSFORM DOMAIN IMPLEMENTATION EMPLOYING FAST ALGORITHMS

Figure 6

It will be clear that other procedures are possible also and that the detection and correction of multiple errors are easily implemented with more powerful codes and additional restrictions.

The finite field forward transforms is used to obtain the transform coefficients $\hat{a}_j$, j = o, 1, ..., (n-1) from the polynomial a(x).

$$\hat{a}_j = \sum_{i=0}^{n-1} a_i \alpha^{ij} \quad ; j = 0, 1, \ldots, (n-1) \qquad (32)$$

$$q = p^r$$

$$n \mid (q^m - 1); (n,q) = 1$$

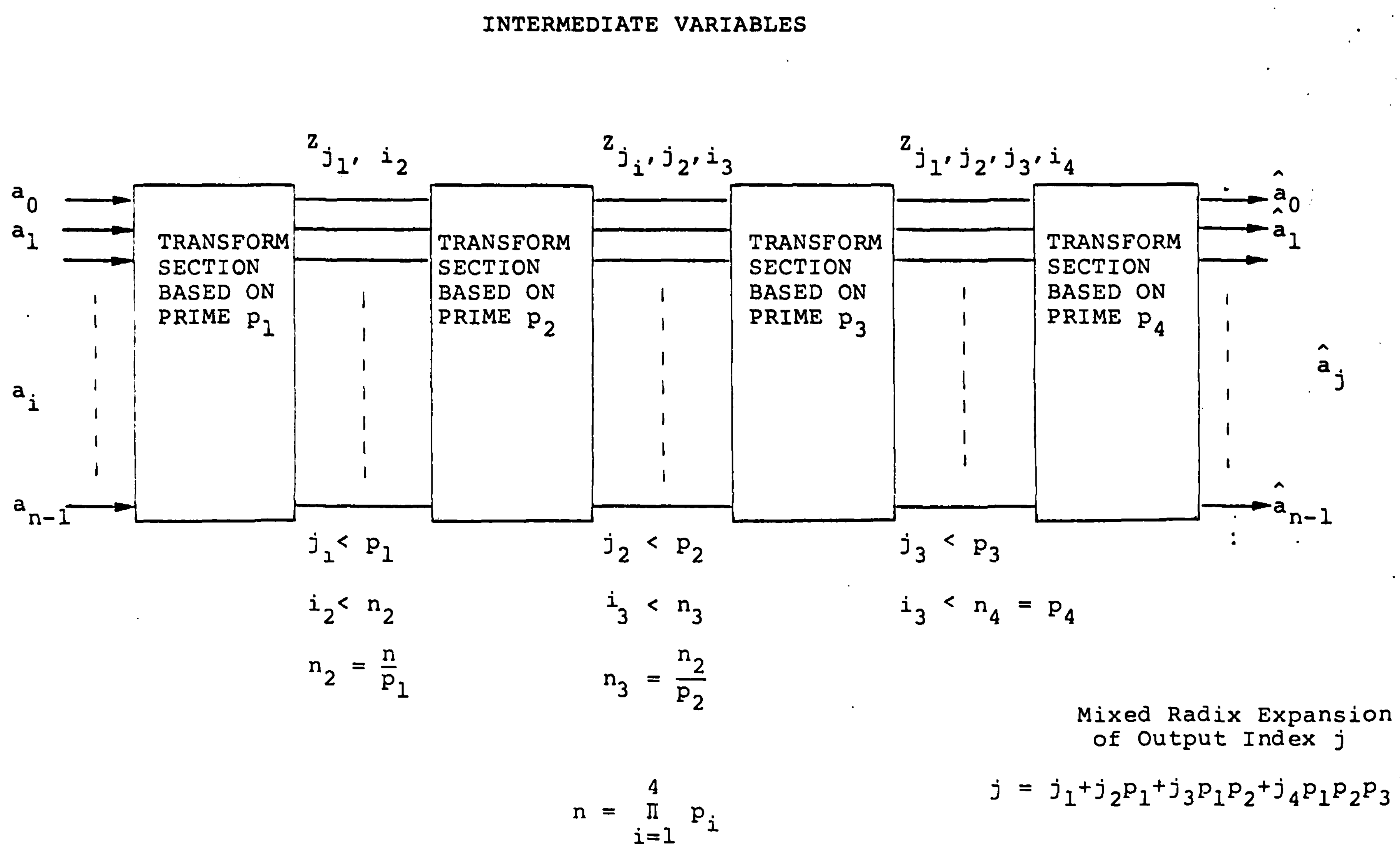
$$\alpha \ n^{\underline{th}} \text{ root of unity}$$

$$a(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

A prime factor fast transform algorithm may be developed starting with the prime decomposition of the integer n. Repeated occurrences of the same prime are permitted.

$$n = \prod_{i=1}^{M} p_i \quad ; p_i = \text{PRIME} \qquad (33)$$

Since n and q are relatively prime, p cannot appear in the product (33). There are two forms of the fast algorithm depending on the initial decomposition of the indices i and j appearing in (32): decimation in frequency and decimation in time. Figure 7 shows the general form of the fast algorithm for the case of n having four factors. Both forms never use


INTERMEDIATE VARIABLES
z_{j_1, i_2}
z_{j_i, j_2, i_3}
z_{j_1, j_2, j_3, i_4}
a_0
a_1
a_i
a_{n-1}
TRANSFORM SECTION BASED ON PRIME p_1
TRANSFORM SECTION BASED ON PRIME p_2
TRANSFORM SECTION BASED ON PRIME p_3
TRANSFORM SECTION BASED ON PRIME p_4
$\hat{a}_0$
$\hat{a}_1$
$\hat{a}_j$
$\hat{a}_{n-1}$
$j_1 < p_1$
$i_2 < n_2$
$n_2 = \frac{n}{p_1}$
$j_2 < p_2$
$i_3 < n_3$
$n_3 = \frac{n_2}{p_2}$
$j_3 < p_3$
$i_3 < n_4 = p_4$
Mixed Radix Expansion of Output Index j
$j = j_1 + j_2 p_1 + j_3 p_1 p_2 + j_4 p_1 p_2 p_3$
$n = \prod_{i=1}^{4} p_i$

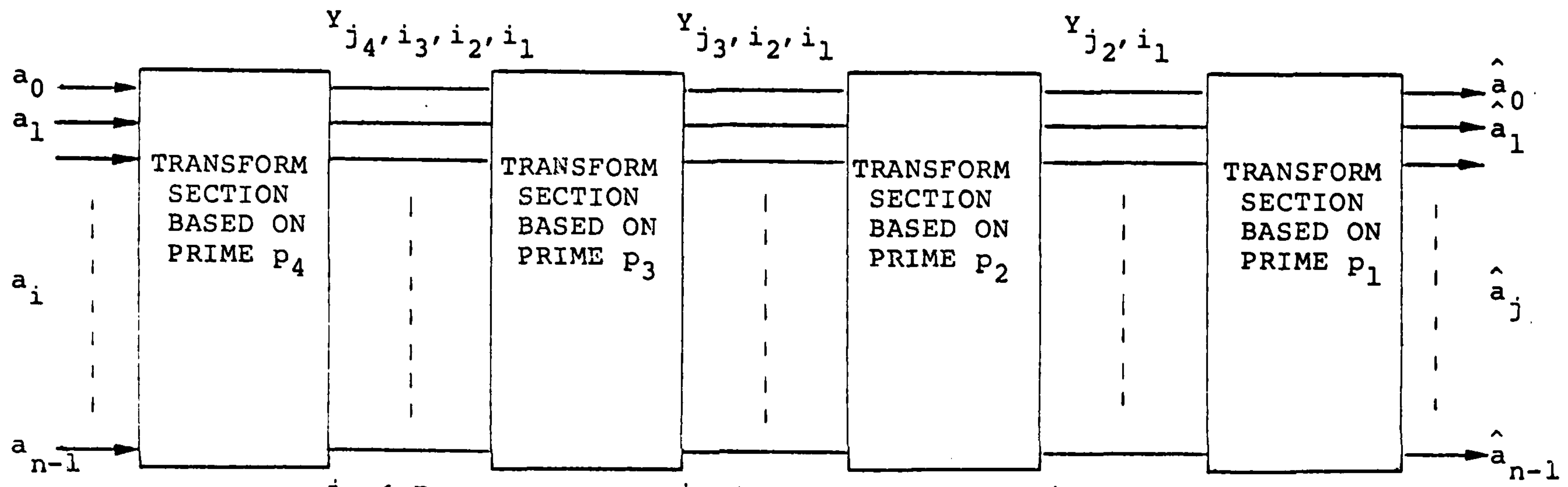
DECIMATION IN FREQUENCY FORM

FIGURE 7a

GENERAL FORM OF FAST FORWARD ALGORITHM

FIGURE 7

INTERMEDIATE VARIABLES


Y_{j_4, i_3, i_2, i_1}
Y_{j_3, i_2, i_1}
Y_{j_2, i_1}
a_0
a_1
a_i
a_{n-1}
TRANSFORM SECTION BASED ON PRIME p_4
TRANSFORM SECTION BASED ON PRIME p_3
TRANSFORM SECTION BASED ON PRIME p_2
TRANSFORM SECTION BASED ON PRIME p_1
$\hat{a}_0$
$\hat{a}_1$
$\hat{a}_j$
$\hat{a}_{n-1}$

$j_4 < n_4 = p_4$ $\quad$ $j_3 < n_3$ $\quad$ $j_2 < n_2$

$i_3 < p_3$ $\quad$ $i_2 < p_2$ $\quad$ $i_1 < p_1$

$n_4 = \frac{n_3}{p_2}$ $\quad$ $n_3 = \frac{n_2}{p_2}$ $\quad$ $n_2 = \frac{n}{p_1}$

Mixed Radix Expansion
of Input Index i

$$i = i_1 + i_2p_1 + i_3p_1p_2 + i_4p_1p_2p_3$$

$$n = \prod_{i=1}^{4} p_i$$

DECIMATION IN TIME FORM

FIGURE 7b

more than n intermediate variables at any stage of the algorithm. The straightforward mathematical implementation of formula (32) requires on the order of n^2 multiplications and a similar order of additions. Either form of the fast algorithm needs on the order of

$$n\left(\sum_{i=1}^{M} p_i\right)$$

such operations. This is one source of the speed advantage of fast algorithms.

The details of the fast algorithms are developed in Appendix F, but a list of the intermediate variables and the ranges of the indices are displayed above the interstage gaps in Figure 7. Figure 8 gives the sets of variables and their interrelationship for both forms of the algorithm. Pieces of the appropriate signal graphs are given in Appendix F where it becomes clear that an error in one operation rapidly spreads to many subsequent intermediate variables. However in both forms the intermediate variables, the Y and Z subscripted variables, obey properties which allow chords to be used at each stage. These properties may be listed where the ranges of the indices are given in Figure 8.

$$\begin{aligned}
(Z_{j_1, i_2})^q &= Z_{qj_1, i_2} &&;\ qj_1 \bmod p_1 \\
(Z_{j_1, j_2, i_3})^q &= Z_{qj_1, qj_2, i_3} &&;\ qj_2 \bmod p_2 \qquad (34) \\
(Z_{j_1, j_2, j_3, i_4})^q &= Z_{qj_1, qj_2, qj_3, i_4} &&;\ qj_3 \bmod p_3 \\
(Y_{j_4, i_3, i_2, i_1})^q &= Y_{qj_4, i_3, i_2, i_1} &&;\ qj_4 \bmod n_4
\end{aligned}$$

$$\begin{tikzcd}
\{a_j\} \arrow[r, "\text{FIRST STAGE}"] & \{z_{j_1,i_2}\} \arrow[r, "\text{SECOND STAGE}"] & \{z_{j_1,j_2,i_3}\} \arrow[r, "\text{THIRD STAGE}"] & \{z_{j_1,j_2,j_3,i_4}\} \arrow[r, "\text{FOURTH STAGE}"] & \{\hat{a}_{j_1+j_2p_1+j_3p_1p_2+j_4p_1p_2p}\} \\
0\le j<n & 0\le j_1<p_1 & 0\le j_2<p_2 & 0\le j_3<p_3 & 0\le j_4<p_4 \\
 & 0\le i_2<n_2 & 0\le i_3<n_3 & 0\le i_4<n_4 &
\end{tikzcd}$$

DECIMATION IN FREQUENCY FORM

FIGURE 8a

$$\begin{tikzcd}
\{a_{i_1+i_2p_1+i_3p_1p_2+i_4p_1p_2p_3}\} \arrow[r, "\text{FIRST STAGE}"] & \{Y_{j_4,i_3,i_2,i_1}\} \arrow[r, "\text{SECOND STAGE}"] & \{Y_{j_3,i_2,i_1}\} \arrow[r, "\text{THIRD STAGE}"] & \{Y_{j_2,i_1}\} \arrow[r, "\text{FOURTH STAGE}"] & \{\hat{a}_j\} \\
0\le i_4<p_4 & 0\le j_4<n_4 & 0\le j_3<n_3 & 0\le j_2<n_2 & 0\le j<n \\
 & 0\le i_3<p_3 & 0\le i_2<p_2 & 0\le i_1<p_1 &
\end{tikzcd}$$

DECIMATION IN TIME FORM

FIGURE 8b

$$n = p_1p_2p_3p_4 \quad ; \quad n_2 = \frac{n}{p_1} \quad ; \quad n_3 = \frac{n_2}{p_2} \quad ; \quad n_4 = \frac{n_3}{p_3} = p_4$$

RELATIONSHIP OF INTERMEDIATE VARIABLES IN FOUR FACTOR ALGORITHM

FIGURE 8

$$\begin{aligned}(Y_{j_3, i_2, i_1})^q &= Y_{qj_3, i_2, i_1} \quad &; qj_3 \bmod n_3 \\ (Y_{j_2, i_1})^q &= Y_{qj_2, i_1} \quad &; qj_2 \bmod n_2\end{aligned} \qquad (35)$$

The procedures, outlined in the previous section on error protection through chords in the transform domain, can be used for detection and correction as detailed in Appendix E. Note from equations (34) and (35) that the chord lengths are now determined modulo integers which are divisors of n. Therefore it is important to insure chord lengths greater than one so that detection is at least possible. Two principles which are demonstrated in Appendix F may be used in this regard. One concerns the combination of chord lengths while the other is constraints on the code length n.

The first principle applies to the overall chord length when more than one index and prime number is involved as in equations (34). For example, say that the chords corresponding to index j_1 mod p_1 is ℓ_1 while the one for index j_2 mod p_2 is ℓ_2.

$$\begin{aligned}j_1 q^{\ell_1} &\equiv j_1 \bmod p_1 \quad &; \ell_1 \text{ least} \\ j_2 q^{\ell_2} &\equiv j_2 \bmod p_2 \quad &; \ell_2 \text{ least}\end{aligned} \qquad (36)$$

Then the overall chord length applicable to Z_{j_1, j_2, i_3}, for i_3 fixed, is the least common multiple of ℓ_1 and ℓ_2.

$$L_{12} = \ell.\text{c.m.}\ (\ell_1, \ell_2) \tag{37}$$

This means by successively forming $q^{\underline{th}}$ powers of Z_{j_1, j_2, i_3}, its value is repeated first on the L_{12} st power.

The second principle concerns sufficient conditions insuring that all chord lengths for nonzero indices exceed length ℓ. The conditions are

$$\left|n,\ (q^s-1)\right| = 1 \quad \text{for all } s \leq \ell. \tag{38}$$

They guarantee that all chord lengths t which satisfy

$$j\ q^t \equiv j \bmod p_i \qquad i = 1, 2, \ldots, M \tag{39}$$

obey the inequality

$$\ell \leq t. \tag{40}$$

With this principle it is possible to know when chord lengths are favorable to detection or correction. The sufficient conditions can only be satisifed up to the integer m because $n|(q^m-1)$, equation (32).

Even when the restriction (38) are used to guarantee error detection or correction through the chord properties, there are situations where errors can still propagate from one stage to another. The properties (34) and (35) show that any intermediate variables with all j indices equal to zero must lie in GF(q), e.g.,

$$Z_{0,\ 0,\ i_3} \ \varepsilon\ GF(q) \qquad ;\ 0 \le i_3 < n_3$$

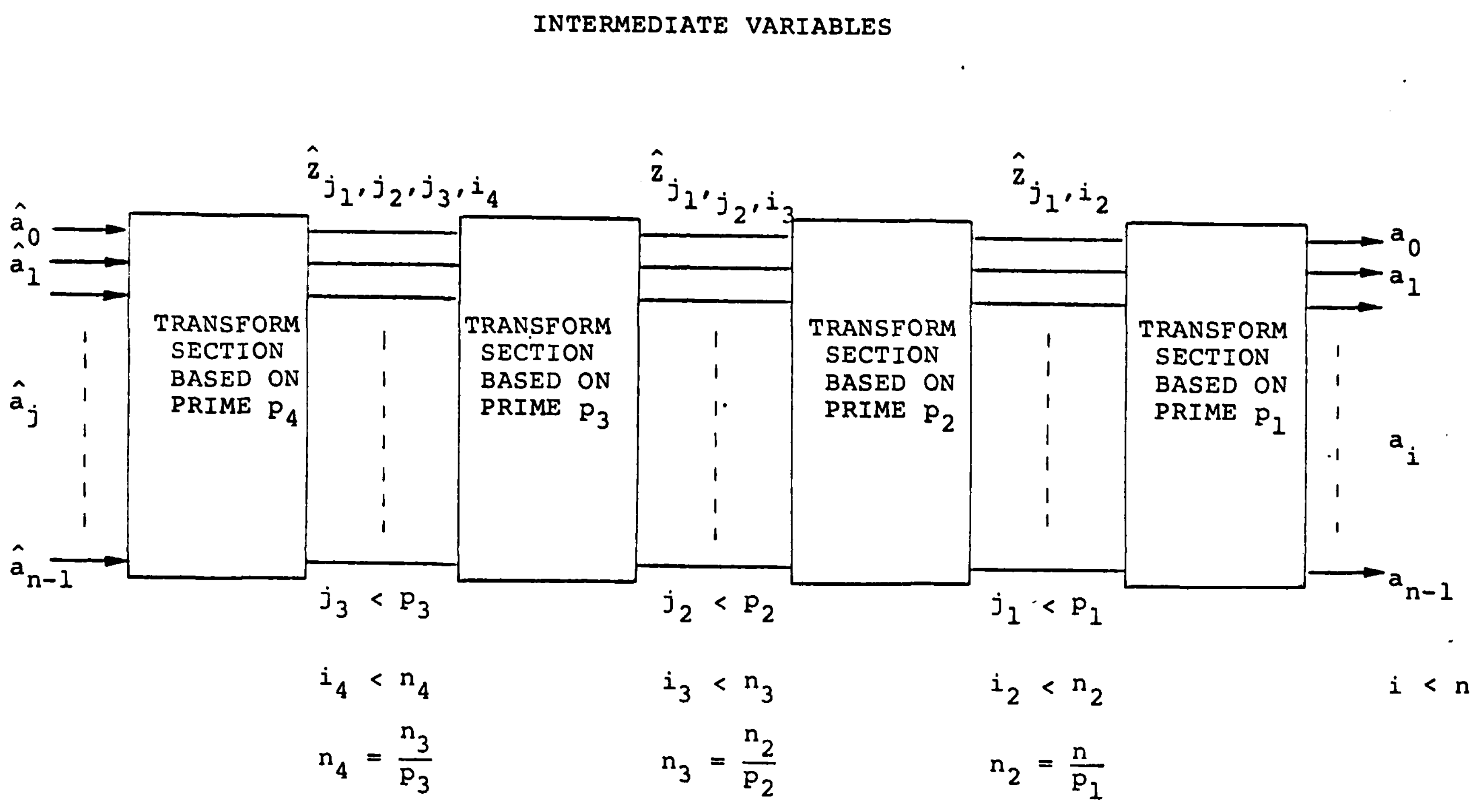
$$Y_{0,\ i_2, i_3} \ \varepsilon\ GF(q) \qquad ;\ i_2 < p_2,\ i_1 < p_1$$

At each stage there are fewer such terms with the DC term, $\hat{a}_0$, being the only one in the transform domain belonging to GF(q). Furthermore intermediate variables with zero j subscripts may propagate unchecked errors only to those new variables at the next stage which have analogous j indices also equal to zero. Thus uncovered errors will only reach the DC coefficeint, $\hat{a}_0$. However this one term may be protested by redundant duplication methods.

The fast inverse transform is developed in an analogous manner. It will have the usual symmetry with respect to the fast forward transform [6]. Figure 9 contains the general structure of both the time and frequqncy decimation forms. The intermediate variables passed between stages are shown also. These variables have properties similar to those in equations (34) and (35) with Appendix F containing the formal development of the algorithm and the properties. They are shown below.

$$(\hat{z}_{j_1,\ i_2})^q = \hat{z}_{qj_1,\ i_2} \qquad ;\ qj_1 \bmod p_1$$

$$(\hat{z}_{j_1,\ j_2,\ i_3})^q = \hat{z}_{qj_1,\ qj_2,\ i_3} \qquad ;\ qj_2 \bmod p_2 \qquad (41)$$



Mixed Radix Expansion of Input Index j

$$n = \prod_{i=1}^{4} p_i$$

$$j = j_1 + j_2p_1 + j_3p_1p_2 + j_4p_1p_2p_3$$

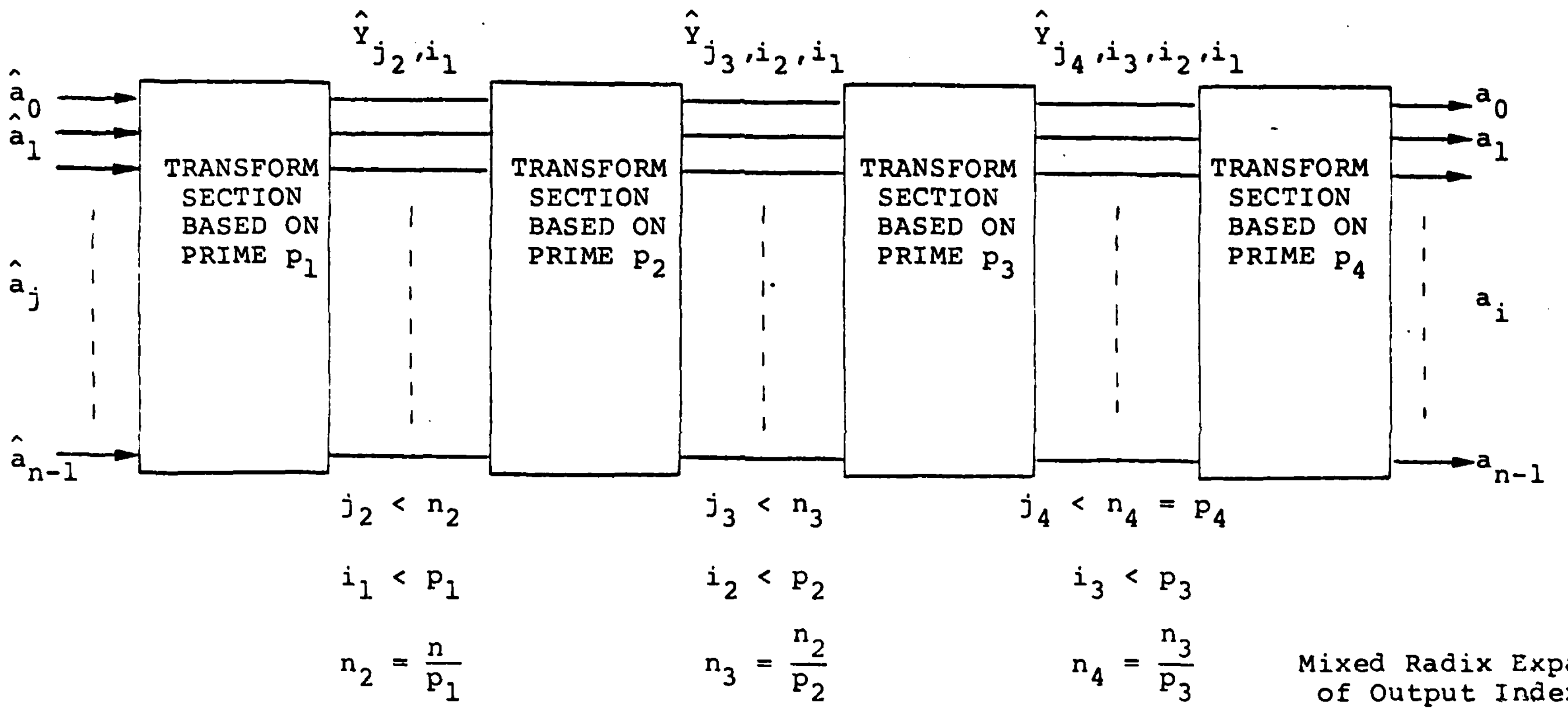
DECIMATION IN FREQUENCY FORM

FIGURE 9a

GENERAL FORM OF FAST INVERSE ALGORITHM

FIGURE 9

INTERMEDIATE VARIABLES



Mixed Radix Expansion of Output Index i

$$i = i_1 + i_2p_1 + i_3p_1p_2 + i_4p_1p_2p_3$$

DECIMATION IN TIME FORM

FIGURE 9b

$$\begin{aligned}
(\hat{Z}_{j_1, j_2, j_3, i_4})^q &= \hat{Z}_{qj_1\ qj_2, qj_3, i_4} &&; qj_3 \bmod p_3 \\
(\hat{Y}_{j_2, i_1})^q &= \hat{Y}_{qj_2, i_1} &&; qj_2 \bmod n_2 \\
(\hat{Y}_{j_3, i_2, i_1})^q &= \hat{Y}_{qj_3, i_2, i_1} &&; qj_3 \bmod n_3 \\
(\hat{Y}_{j_4, i_3, i_2, i_1})^q &= Y_{qj_4, i_3, i_2, i_1} &&; qj_4 \bmod n_4
\end{aligned} \tag{42}$$

Properties (41) and (42) may be used to establish chords affording error detection and correction as outlined earlier. The sufficient conditions (38) clearly insure that all chords associated with nonzero indices have at least length ℓ. But any intermediate variables whose j indices are all zero lie in base field GF(q), and may impact other subsequent GF(q) variables. Hence it is possible to propagate errors from one stage forward. In addition, an examination of the appropriate signal flow graphs shows that each such variable affects numerous GF(q) variables in the next stage. These errors penetrate to the final stage possibly affecting many output coefficients.

As an example of this internal error propagation error polynomials, generically labeled e(x), which are added to the final result $\delta(x)$ are consider. They are developed for the situations where a single additive error, with value ε, occurs within a stage. Table 1 summarizes the results of these analyses. These error polynomials have the same value

ε added to many positions in the desired output $\delta(x)$. Nevertheless it is possible to detect such errors in the final output if the code generator polynomial, $g(x)$, is probably chosen. It must not divide the special forms of $e(x)$ as shown in Table 1. Note that each error polynomial is formed by effectively removing certain nth roots of unity from the factor (x^n-1). Hence if $g(x)$ contains at least some of the roots which are removed it could not divide any of the $e(x)$ terms thus providing an error detection capability.

For example, in the decimation in frequency form an error polynomial

$$e(x) = x^{i_3}\left[\frac{(x^n-1)}{(x^{n_3}-1)}\right]$$

can appear at the output. But if $g(x)$ contains as one of its roots an n_3 root of unity (and of course its conjugate roots), all errors of this form are detectable. Therefore proper code selection is critical for protecting against internal errors in the fast inverse system. Since all $(x^{n_i}-1)$ factors are divisible by $(x-1)$, so if $g(x)$ contains 1 as a root, every possible single error failure may be detected.

ERROR DESCRIPTION		ERROR POLYNOMIAL AT OUTPUT
STAGE	VARIABLE WITH ADDITIVE ERROR ε	$e(x)$
1	$\hat{a}_0$	$\varepsilon \left[\frac{(x^n-1)}{(x-1)}\right]$
2	$\hat{z}_{0,0,0,i_4}$ $i_4 < p_4$	$\varepsilon\, x^{i_4} \left[\frac{(x^n-1)}{(x^{n_4}-1)}\right]$
3	$\hat{z}_{0,0,i_3}$ $i_3 < n_3$	$\varepsilon\, x^{i_3} \left[\frac{(x^n-1)}{(x^{n_3}-1)}\right]$
4	$\hat{z}_{0,i_2}$ $i_2 < n_2$	$\varepsilon\, x^{i_2} \left[\frac{(x^n-1)}{(x^{n_2}-1)}\right]$

DECIMATION IN FREQUENCY FORM

TABLE 1a

OUTPUT ERROR POLYNOMIALS DUE TO SINGLE INTERNAL ERRORS IN THE FAST INVERSE ALGORITHM

TABLE 1

ERROR DESCRIPTION		ERROR POLYNOMIAL AT OUTPUT
STAGE	VARIABLE WITH ADDITIVE ERROR ε	$e(x)$
1	$\hat{a}_0$	$\varepsilon \left[\frac{(x^n-1)}{(x-1)}\right]$
2	$\hat{Y}_0, i_1$ $i_1 < p_1$	$\varepsilon x^{i_1} \left\{\frac{(x^n-1)}{(x^{\frac{n}{n_2}}-1)}\right\}$
3	$\hat{Y}_0, i_2, i_3$ $i_2 < p_2$ $i_3 < p_3$	$\varepsilon x^{i_1} x^{i_2 p_1} \left\{\frac{(x^n-1)}{(x^{\frac{n}{n_3}}-1)}\right\}$
4	$\hat{Y}_0, i_3, i_2, i_1$	$\varepsilon x^{i_1} x^{i_2 p_1} x^{i_3 p_1 p_2} \left\{\frac{(x^n-1)}{(x^{\frac{n}{n_4}}-1)}\right\}$

DECIMATION IN TIME FORM

TABLE 1b

Error Detection/Correction and Error Status

The approach to error control presented in this chapter has intrinsic opportunities for distributed error detection and correction throughout the architecture. Not all possibilities will be explored here but an obvious and important aspect of error protection involves the minimal ideals. Each minimal ideal, M_i, which contains the component polynomials is a cyclic code in its own right [4]. The number of parity-check positions is equal to the degree of the related irreducible factor of $(x^n - 1)$, $g_i(x)$. Cyclic codes have powerful burst error-protecting capabilties proportional to the number of parity-check positions. Thus error detection or correction can be performed on the individual components $[a_i(x)b_i(x)e_i(x)]$ before the final combination as required by equation (4) is completed. This protects against errors introduced in the last stages of the component polynomial manipulations. This technique allows errasure decoding to be mixed with the usual decoding methods [2, 5]. The overall detecting and correcting performance levels increase accordingly.

Suppose that the output of the filter mechanization of $a(x)\ b(x)\ g(x)$ can be modeled as containing additive errors. The output of the filter section is given by $r(x)$.

$$r(x) = [a(x)\ b(x)\ g(x)] + \nu(x) \quad ; \nu(x) \quad \text{error components} \qquad (43)$$

Standard error-control techniques [2-5] may be applied under these conditions. For example the same transform domain used in the filter's implementation is also applicable. In this situation the roots of g(x) as indexed by the set $\Lambda^{(g)}$, equation (26), are involved. They define the syndromes. (Recall g(x) has degree (n-k) and $\Lambda^{(g)}$ has (n-k) elements.) A typically syndrome is s_β where

$$\begin{aligned} s_\beta &= r(\alpha^\beta) \qquad ; \beta \in \Lambda^{(g)} \\ &= \nu(\alpha^\beta) \end{aligned} \tag{44}$$

Of course there is the problem that the Error Detector/ Corrector system shown in figure 2 could suffer errors similar to the section that it was designed to protect. The Error Status device is designed to signal the occurance of such a problem. Say that the decoder produces a polynomial, f(x), which represents the desired product a(x) b(x). An error status may be made by encoding f(x) through multiplication by g(x) and subtracting it from a copy of r(x), equation (43).

$$\mu(x) = r(x) - f(x)g(x)$$

The weight of the status polynomial $\mu(x)$ determines whether the decoded element f(x) is the closest choice to r(x). This gives a check on the Error Detector/Corrector and the Status Check circuit.

The Special Case of Systematic Codes

All of the previous discussions are applicable to the case of systematic codes. As was mentioned in the introduction, the parity-check portion, $[-p(x)\ a(x)]$, when added to a shifted version of the desired result, $a(x)\ b(x)$, yields a codeword. Therefore the techniques described for nonsystematic codes can be employed for the parity-check portion of the codeword; only the components from the t minimal ideals that define the code G need to be computed. Linearity quarantees that no other components are required.

The desired result $a(x)\ b(x)$ has degree less than k and thus can be computed in a residue class ring modulo $(x^k - 1)$ which has been labeled as A_k in this report. All the minimals ideals in the algebra A_k are employed now because no code is involved. However the use of the minimal ideal decomposition and the transform techniques can be advantageously applied. One may think of using a rate-one, (k,k) code in this case. Nevertheless there are intrinsic error detection and correction opportunities even in the no coding ideals and because of the properties of the transform domain.

References

[1] Matluk, M.M. and Gill, A., "Decomposition of Linear Sequential Circuits over Residue Class Rings", Journal of the Franklin Institute, vol. 294, No. 3, pp. 167-180, Sept., 1972.

[2] Peterson, W.W. and Weldon, E.J., Jr., Error-Correcting Codes, Second Edition, Cambridge, Massachusetts: The MIT Press, 1972.

[3] Mac Williams, F.J. and Sloane, N.J.A., The Theory of Error-Correcting Codes, Amsterdam: North-Holland Publishing Company, 1977.

[4] Van Lint, J.H., Coding Theory, New York: Springer-Verlag, 1971.

[5] Lin, S., An Introduction to Error-Correcting Codes, Englewood Cliffs, New Jersey: Prentice Hall, 1970.

[6] Nussbaumer, H.J., Fast Fourier Transform and Convolution Algorithms, New York: Springer Verlag, 1981.

[7] Herstein, I.N., Topics in Algebra, New York: Blaisdell Publishing Company, 1964.

[8] G.R. Redinbo, D.O. Carhoun and B.L. Johnson, "Fast Algorithms for Signal Processing Using Finite Field Operations", 1982 IEEE International Conference on Acoustics Speech and Signal Processing, pp. 40-43.

Appendix A

Properties of Minimal Ideal

A minimal ideal only contains itself and the trivial ideal $\left(\left(0\right)\right)$. The fact that the ideals defined by $m_i(x)$, equations (9), are minimal in the space A_n will be proved first; the concepts used in this proof will appear in developing other properties.

Suppose that there is a smaller ideal S_i contained within M_i. Then there is a polynomial $s_i(x)$ that generates S_i, i.e., $S_i = \left(\left(s_i(x)\right)\right)$. Since $s_i(x) \varepsilon M_i$ it follows by the definition of $m_i(x)$ that all irreducible factors except $g_i(x)$ divide $s_i(x)$. (For if $g_i(x)$ were also a factor of $s_i(x)$ then $s_i(x)=0$ in A_n.) Thus $s_i(x)$ may be written in the form

$$s_i(x) = f(x) \prod_{\substack{j \neq i \\ j=1}}^{n} g_j^{\lambda_j(x)} \quad ; \quad \lambda_j \geq 0 \tag{A-1}$$

where f(x) contains all other irreducible factors of $s_i(x)$ not contained in the factorization of (x^n-1). But then f(x) is relatively prime to (x^n-1), indicating that f(x) is a unit in A_n. $\left((f(x), x^n-1)=1\right.$ implies that there are items y(x) and z(x) such that $y(x)f(x) + z(x)(x^n-1)=1$; y(x) is the inverse of f(x) in $\left.A_n\right)$. Hence it is possible to take

$$s_i(x) = \prod_{\substack{j \neq i \\ j=1}}^{n} g_j^{\lambda j}(x) \tag{A-2}$$

Clearly $\|s_i(x)\| \subset \|m_i(x)\|$. To show the opposite inclusion it will be demonstrated that $s_i(x)$ divides $m_i(x)$. Since $g_i(x)$ is excluded from $s_i(x)$, they are relatively prime. Then so is $g_i(x)$ and the term, $\prod_{j\neq i} g_j^{(\lambda j-1)}(x)$, which has all irreducible factors of $s_i(x)$ with single power λ_j's removed.

$$\left| g_i(x),\ \prod_{j\neq i} g_j^{(\lambda_j-1)}(x) \right| = 1 \qquad \text{(A-3)}$$

Thus these are polynomials a(x) and b(x) for which it is possible to write

$$a(x) \prod_{j\neq i} g_j^{(\lambda_j-1)}(x) + b(x)\, g_i(x) = 1 \qquad \text{(A-4)}$$

Then $g_i(x)$ divides $\left[1-a(x) \prod_{j\neq i} g_j^{(\lambda_j-1)}(x)\right]$ and it follows from the definition of the irreducible factors of (x^n-1) that

$$\prod_{\substack{j\neq i\\ j=1}}^{n} g_j(x) \left[1-a(x) \prod_{k\neq i} g_k^{(\lambda_k-1)}(x)\right] \equiv 0 \bmod (x^n-1) \qquad \text{(A-5)}$$

Equivalently in A_n,

$$m_i(x) = a(x) \prod_{\substack{j\neq i\\ j=1}}^{n} g_j^{\lambda_j}(x) \qquad \text{(A-6)}$$

The generating polynomials of the minimal ideals have an interesting property in A_n:

$$m_i(x)\, m_j(x) = 0 \qquad ;\ i\neq j,\ i,j, = 1, 2, \ldots, T \qquad \text{(A-7)}$$

It is easy to show from this that the T minimal ideals form a direct sum decomposition of A_n. Thus any $a(x) \varepsilon A_n$ can be expanded uniquely as a sum of elements each possibly from one of the minimal ideals:

$$a(x) = \sum_{j=1}^{T} a_j(x)\ m_j(x) \tag{A-8}$$

Each nonzero term $[a_j(x)\ m_j(x)]$ represents the component of a(x) in minimal ideal M_j.

The orthogonality of the ideal generates $m_i(x)$ and $m_j(x)$ clearly demonstrates that any distinct ideals share only the zero element. Thus to show that the minimal ideals M_i can be used for a direct sum decomposition of A_n, it remains to prove that any element in A_n can be written using components from some of the ideals. The set of polynomials $m_i(x)$, i = 1, 2, ..., T have their greatest common divisor as unity. Then there are polynomials $\nu_i(x)$ such that

$$1 = \sum_{j=1}^{T} \nu_j(x)\ m_j(x) \tag{A-9}$$

Multiplying both sides by any $a(x) \varepsilon A_n$ gives:

$$a(x) = \sum_{j=1}^{T} [a(x)\ \nu_j(x)\ m_j(x)] \tag{A-10}$$

Since none of the nonzero terms, $[a(x)\ \nu_j(x)\ m_j(x)]$, can be in more than one ideal, the direct sum expansion in equation (A-8) is established where $a_j(x) = a(x)\nu_j(x)$.

The idempotents have several important properties which will be listed and then proved.

$$e_i(x)\ a(x) = a(x) \quad ; \ a(x) \varepsilon M_i \qquad \text{(A-11)}$$

UNIT ON M_i

$$e_j(x) = \begin{cases} 1 & j = i \\ 0 & j \neq 1 \end{cases} \bmod g_i(x) \qquad \text{(A-12)}$$

ORTHONORMAL IDENTITY

$$\left(\left(e_i(x)\right)\right) = \left(\left(m_i(x)\right)\right) \qquad \text{IDEAL GENERATOR} \qquad \text{(A-13)}$$

$$\left(e_i(x), g_i(x)\right) = 1 \qquad \text{(A-14)}$$

RELATIVELY PRIME RELATIONSHIP

Each minimal ideal M_i contains an idempotent that also generates the same principal ideal. The idempotent is determined by noting that $m_i(x)$ and $g_i(x)$ are relatively prime. Hence there are two polynomials $s_i(x)$ and $r_i(x)$ for which the following is true.

$$1 = s_i(x)\ m_i(x) + r_i(x)\ g_i(x) \ ; \ i=1, 2, \ldots, T \qquad \text{(A-15)}$$

The desired idempotent $e_i(x)$ is defined by

$$e_i(x) = s_i(x)\ m_i(x) \ ; \ i = 1, 2, \ldots, T \qquad \text{(A-16)}$$

We now detail the proofs of the properties listed above. Firstly the basic idempotent property equation (10) follows by multiplying identity (A-15) by $e_i(x)$.

$$\begin{aligned} e_i(x) &= e_i^2(x) + s_i(x)\ m_i(x)\ r_i(x)\ g_i(x) \\ &\equiv e_i^2(x) + 0 \qquad \text{mod } (x^n-1) \end{aligned} \tag{A-17}$$

The property of a unit also is demonstrated by choosing any $a(x)$ in M_i and multiplying equation (A-15) by it. However, since $a(x) = a_i(x)m_i(x)$ because it is in the principal ideal generated by $m_i(x)$, a simplification is possible.

$$\begin{aligned} a(x) &= a(x)\ e_i(x) + a_i(x)\ m_i(x)\ r_i(x)\ g_i(x) \\ &\equiv a(x)\ e_i(x) + 0 \qquad \text{modulo } (x^n-1) \end{aligned} \tag{A-18}$$

Reducing both sides of identity (A-15) modulo $g_i(x)$ gives the top part of equation (A-12) while definition (A-16) clearly shows that $e_i(x)$ contains all other irreducible factors and the remaining part of the orthonormal property is demonstrated.

We observe that by the definition of $e_i(x)$, equation (A-15), $\|(e_i(x))\| \subset \|(m_i(x))\|$.

While on the other hand combining identities (A-15) and (A-16) and multiplying by $m_i(x)$ yields:

$$\begin{aligned} m_i(x) &= m_i(x)\ e_i(x) + m_i(x)\ r_i(x)\ g_i(x) \\ &\equiv m_i(x)\ e_i(x) + 0 \quad \text{modulo } (x^n-1) \end{aligned} \tag{A-19}$$

Hence $m_i(x)$ is contained in $\|e_i(x)\|$ and therefore

$$\|m_i(x)\| \subset \|e_i(x)\|.$$

Both $m_i(x)$ and $e_i(x)$ are legitimate generators for M_i as shown in equation (A-13).

Finally the relatively prime property in equation (A-14) can be shown by assuming that $g_i(x)$ divides $e_i(x)$. Since $e_i(x)$ is a unit, property (A-11), the following is true.

$$e_i(x)m_i(x) = m_i(x) \qquad \text{(A-20)}$$

But by the definition of $m_i(x)$, equation (9), the left side is zero in A_n, a contradiction to fact that $m_i(x)$ has degree strictly less than n.

Another interesting and useful property of the minimal ideals is their relationship with the appropriately sized finite fields. Consider the following mapping from M_i into the cosets of the principal ideal $\|g_i(x)\|$.

$$\begin{aligned} F_i \quad : \quad M_i &\longrightarrow GF(q)[x]/\|g_i(x)\| \\ a(x)\ e_i(x) &\longmapsto a(x) \bmod g_i(x) \end{aligned} \qquad \text{(A-21)}$$

FIELD ISOMORPHISM

The range space is known to be a finite since $g_i(x)$ is irreducible. Note that in M_i we may always write each element b(x) as a(x) $e_i(x)$ where degree a(x) < degree $g_i(x)$. Also note that in the minimal ideal, $e_i(x)$, acts as the field identity, but it is not the identity of the ring A_n.

To establish the isomorphism (A-21) we first show that every $b(x)$ is equivalent to another polynomial $a(x)$ with degree less than degree $g_i(x)$. The Eucledian algorithm can be used to show that

$$b(x) = a_i(x)\, g_i(x) + a(x) \qquad \text{where } \deg a(x) < \deg g_i(x) \tag{A-22}$$

It then follows that because of the unit property of $e_i(x)$,

$$\begin{aligned} b(x) &= b(x)\, e_i(x) \\ &= a_i(x)\, g_i(x)\, e_i(x) + a(x)\, e_i(x) = a(x)\, e_i(x) \end{aligned} \tag{A-23}$$

Consider any $\alpha(x) \varepsilon GF(q)[x]/ \left\|g(x)\right\|$. The element $\alpha(x)\, e_i(x)$ in M_i clearly is mapped to it by F_i. Since the number of distinct elements in each space is equal the mapping is one-to-one and onto. The mapping F_i preserves both sums and products. Note since $\left|g_i(x), e_i(x)\right| = 1$ the item $a_1(x)\, e_i(x)\, b_1(x)\, e_i(x) = a_1(x)\, b_1(x)\, e_i(x)$ maps into $a_1(x)\, b_1(x) \bmod g_i(x)$. The mapping (A-21) gives $M_i \bmod g_i(x)$ all of the properties of a field and also explains why $e_i(x)$ acts as a unit on the space $M_i \bmod g_i(x)$.

Appendix B

Idempotent Expansion of the Code Ideal

The direct sum decomposition of G, equation (16), may be demonstrated in the following manner. Since $g(x) \in A_n$ it may be expressed using the minimal ideal expansion as per equation (11).

$$g(x) = \bigoplus_{i=1}^{T} \gamma_i(x)\ e_i(x)$$

$$\gamma_i(x) = g(x) \bmod g_i(x) \qquad \text{(B-1)}$$

However from the construction of g(x) as given in equation (13), (T-t) of the $\gamma_i(x)$ coefficients are zero.

$$\gamma_i(x) = 0 \bmod g_i(x) \qquad i = t+1,\ t+2,\ \ldots,\ T \qquad \text{(B-2)}$$

Thus g(x) is in a smaller subspace of A_n; it is defined by the first t minimal ideals.

$$g(x) \in \bigoplus_{i=1}^{T} M_i$$

Since the finite direct sum of ideals is an ideal, it follows that

$$\left\|g(x)\right\| \subset \bigoplus_{i=1}^{t} M_i$$

On the other hand, the generator of each M_i, $e_i(x)$, i=1, 2, ..., t, by its respective definition (A-16) is contained in the ideal $\left\|g(x)\right\|$.

It then follows that

$$\bigoplus_{i=1}^{t} M_i \subset \left\|g(x)\right\|$$ so that the desired identity (16)

is established.

The code G also has an idempotent generator which will be denoted by $g_I(x)$. It satisfies the usual property:

$$g_I^2(x) = g_I(x) \qquad \text{(B-3)}$$

Furthermore it is equivalent to the generator g(x).

$$G = \left\|g_I(x)\right\| = \left\|g(x)\right\| \qquad \text{(B-4)}$$

The code idempotent acts as an identity on the ideal G.

$$\gamma(x) = \gamma(x)\ g_I(x) \quad ; \quad \text{All } \gamma(x) \in G \qquad \text{(B-5)}$$

These results are proven below where a method for determining $g_I(x)$ is given. Unlike g(x) its degree is not necessarily (n-k).

One possible application for the code idempotent in representing the code involves the cascading of filter sections. Say the same code is used in each of several filter sections which follow one another in a cascade. Further suppose that no decoding is performed until the final section is reached. If the stored filter weighting in each of the s sections is given by the codeword $[b_i(x)\ g_I(x)]$, i = 1, 2, ..., s, then the output would be of the form

$$a(x) \prod_{i=1}^{s} \left[b_i(x)\ g_I(x)\right] = a(x)\ g_I(x) \prod_{i=1}^{s} b_i(x) \quad ; \text{ a(x) input to cascade}$$

by repeated applications of property (B-3).

The idempotent form of the generator polynomial $g(x)$, labeled by $g_I(x)$, may be constructed by noting that $g(x)$ and $h(x)$ are relatively prime. (They share no common irreducible factors.) Thus there are polynomials $d(x)$ and $f(x)$ for which it is possible to write:

$$d(x)\ g(x) + f(x)\ h(x) = 1 \tag{B-6}$$

The idempotent generator $g_I(x)$ is the product of $d(x)$ and $g(x)$

$$g_I(x) = d(x)\ g(x) \tag{B-7}$$

It is easy to see that $g_I(x)$ and $g(x)$ both generate the same ideal.

$\|g_I(x)\| \subset \|g(x)\|$ on one hand while $g(x) = g(x)$ $[g_I(x) + f(x)\ h(x)] = g(x)\ g_I(x) + 0$ on the other; thus $g_I(x)$ divides $g(x)$ and $\|g(x)\| \subset \|g_I(x)\|$.

$$\|g_I(x)\| = \|g(x)\| \tag{B-8}$$

It is possible using the definition of $g_I(x)$ and the basic equation (B-6) above to demonstrate the facts in equations (B-3) and (B-5). Their proofs parallel those associated with the similar results for minimal ideal idempotents contained in Appendix A.

Appendix C

Finite Field Transform

The n th root of unity, α, has the following properties:

$$\alpha^n = 1 \quad , \alpha^k \neq 1 \text{ for all } k < n$$
$$\alpha \in GF(q^m) \quad , \quad n \text{ divides } (q^m - 1) \tag{C-1}$$

The validity of the transform pair, equations (18) and (19), rests upon the following identities concerning sums of roots of unity.

$$\sum_{\ell=0}^{n-1} \alpha^{d\ell} = \sum_{\ell=0}^{n-1} 1 = ((n))_p \quad ; \ d \equiv 0 \bmod n \tag{C-2}$$

$$(1-\alpha^d) \sum_{\ell=0}^{n-1} \alpha^{d\ell} = (1-\alpha^n) = 0 \ ; \ d \not\equiv 0 \bmod n \tag{C-3}$$

The first equation is a consequence of the characteristic of the field $GF(q^m)$ while the second one is the geometric sum. Since $d \not\equiv 0 \bmod n$, $(1-\alpha^d) \neq 0$, and equation (C-3) implies that the sum is zero

$$\sum_{\ell=0}^{n-1} \alpha^{d\ell} = 0 \quad ; \ d \not\equiv 0 \bmod n \tag{C-4}$$

The relationship between equations (18) and (19) may be examined by substituting one into the other.

$$((n^{-1}))_p \sum_{j=0}^{n-1} \hat{\gamma}_j \, \alpha^{-\ell j} = ((n^{-1}))_p \sum_{j=0}^{n-1} \left(\sum_{m=0}^{n-1} \gamma_m \, \alpha^{mj} \right) \alpha^{-\ell j} \tag{C-5}$$

$$= ((n^{-1}))_p \sum_{m=0}^{n-1} \gamma_m \left[\sum_{j=0}^{n-1} \alpha^{(m-\ell) j} \right]$$

However the sum in brackets in the last expression may be evaluated using identities (C-2) and (C-4).

$$\sum_{j=0}^{n-1} \alpha^{j(m-\ell)} = \begin{cases} ((n))_p & ; m \equiv \ell \bmod n \\ 0 & ; m \not\equiv \ell \bmod n \end{cases}$$

This verifies that equation (19) is the inverse of equation (18) and vice versa. The uniqueness of the transform follows immediately from the linearity of the formulas. The inversion formula, equation (19), guarantees that the transform with all zeros also has zero coefficients in the polynomial domain A_n.

The relationship between convolution and pointwise coefficient multiplication in the transform domain, equation (20) can be established by considering the inversion of the product $\{\hat{a}_j \hat{\gamma}_j\}$.

$$\begin{aligned} \delta_m &= ((n^{-1}))_p \sum_{j=0}^{n-1} \hat{a}_j \hat{\gamma}_j \alpha^{-jm} \qquad m=0, 1, \ldots, (n-1) \\ &= ((n^{-1}))_p \sum_{j=0}^{n-1} \alpha^{-jm} \left[\sum_{k=0}^{n-1} a_k \alpha^{jk}\right]\left[\sum_{\ell=0}^{n-1} \gamma_\ell \alpha^{j\ell}\right] \\ &= ((n^{-1}))_p \sum_{k=0}^{n-1} \sum_{\ell=0}^{n-1} a_k \gamma_\ell \left[\sum_{j=0}^{n-1} \alpha^{j(k+\ell-m)}\right] \end{aligned} \tag{C-6}$$

Identities (C-2) and (C-4) permit the evaluation of the sum on j.

$$\sum_{j=0}^{n-1} \alpha^{j(k+\ell-m)} = \begin{cases} ((n))_p & ; k+\ell \equiv m \bmod n \\ 0 & \text{otherwise} \end{cases}$$

The value of δ_m is determined by a convolution of the terms a_i and γ_j where the indices are reduced modulo n.

$$\begin{aligned}\delta_m &= \sum_{\ell=0}^{n-1} a_{m-\ell}\, \gamma_\ell \\ &= \sum_{k=0}^{n-1} a_k\, \gamma_{m-k}\end{aligned} \qquad \text{(C-7)}$$

However this is completely equivalent with polynomial multiplication in A_n because of the reduction modulo $(x^n - 1)$.

$$\delta(x) = a(x)\gamma(x) \qquad \text{(C-8)}$$

where $\delta(x) = \delta_0 + \delta_1 x + \ldots + \delta_{n-1} x^{n-1}$.

Appendix D

Annihilator and Chord Properties of Transform Coefficients

The Annihilator Property relates the zeros of a polynomial $c(x) \in A_n$ with the indices of the zero transform coefficients $\hat{c}_j$. Using the notation of the text and in particular the usual labeling of the polynomial and transform coefficients, it is possible to state the following Lemma.

Lemma D1

$c(x)$ has a α^j as a root if and only if $\hat{c}_j = 0$ for $j \notin \Lambda$ where Λ denotes a subset of the indices $\{0, 1, \ldots, (n-1)\}$.

Proof: Since by definition $\hat{c}_j = c(\alpha^j)$, the necessary part is obvious. On the other hand the inversion formula (19) and the requirement that $\hat{c}_j = 0$ for all $j \notin \Lambda$ allows the following form for $c(x)$.

$$c(x) = \sum_{\ell=0}^{n-1} ((n^{-1}))_p \, x^\ell \sum_{m\in\Lambda} \hat{c}_m \alpha^{-\ell m} \tag{D-1}$$

Evaluating this at α^j, $j \notin \Lambda$ and rearranging the terms gives:

$$c(\alpha^{-j}) = \sum_{m\in\Lambda} ((n^{-1}))_p \, \hat{c}_m \sum_{\ell=0}^{n-1} \alpha^{\ell(j-m)} \tag{D-2}$$

However since $j \notin \Lambda$, $(j-m) \not\equiv 0 \bmod n$ and so identity (C-4) applies showing that the last sum in (D-2) is zero. Thus

α^j, $j \not\in \Lambda$ is a root of c(x) where its transform coefficients are zero for those indices not in subset Λ.

The Chord Property describes a special relationship among the indices of equal transform coefficients. The closure property of cyclotomic subsets resembles this useful property.

Lemma D2

$c(x) \in A_n$ if and only if $(\hat{c}_j)^q = \hat{c}_{jq}$, $j=1, 2, \ldots, n-1$

Proof: For any $c \in GF(q)$, $c^q = c$.

$$(\hat{c}_j)^q = \sum_{\ell=0}^{n-1} c_\ell^q \ \alpha^{jq\ell} = \sum_{\ell=0}^{n-1} c_\ell \ \alpha^{(jq)\ell} = \hat{c}_{jq} \qquad \text{(D-3)}$$

$$; j = 0, 1, 2, \ldots, (n-1)$$

The coefficients of a polynomial in A_n are in GF(q). Then the inversion formula (19) and the special index property on the right of the Lemma lead to the following development

$$(c_\ell)^q = [((n^{-1}))_p] \sum_{j=0}^{n-1} (\hat{c}_j)^q \ \alpha^{-jq\ell} \ ; \ \ell=0, 1, \ldots, (n-1) \qquad \text{(D-4)}$$

The normalizing factor $((n^{-1}))_p$ is by definition in GF(p) $\subset$ GF(q).

$$(c_\ell)^q = ((n^{-1}))_p \sum_{j=0}^{n-1} \hat{c}_{jq} \ \alpha^{-jq\ell} \qquad \text{(D-5)}$$

Since $(n,q) = 1$, a change of variables $m = jq$ in the summation is possible and an equivalent expression for c_ℓ emerges on the right of (D-5).

Appendix E

Error Correction in the Transform Domain

Suppose that the desired output from the componentwise multiplications in the transform domain for component i, i = 1, 2, ..., t is denoted generically by $\{\hat{c}_{ij}\}$. Let $c_i(x)$ designate the corresponding polynomial obtained from the inversion process.

$$\hat{c}_{ij} = \hat{a}_{ij}\,\hat{\gamma}_{ij} \longleftrightarrow c_i(x) \;;\; i = 1, 2, \ldots, t$$
$$j \in \Lambda_s^{(i)}$$

Errors can be introduced in several operations. When the stored transform coefficients $\{\hat{\gamma}_{ij}\}$ are delivered to the multiplier system, they could contain errors $\{\hat{\sigma}_{ij}\}$; the input coefficients $\{\hat{a}_{ij}\}$ could contain errors modeled by the terms $\{\hat{\eta}_{ij}\}$.

$\{\hat{\gamma}_{ij} + \hat{\sigma}_{ij}\}$	Erroneous Stored Sequence
$\{\hat{a}_{ij} + \hat{\eta}_{ij}\}$	Erroneous Input Sequence

Label the final result of the pointwise coefficient multiplications by $\{\hat{d}_{ij}\}$. The most general form of these coefficients including all types of errors is given by:

$$\hat{d}_{ij} = (\hat{\gamma}_{ij} + \hat{\sigma}_{ij})\ (\hat{\alpha}_{ij} + \hat{\eta}_{ij}) + \nu_{ij}$$

Thus it is possible to include all of these types of errors in the following expression where the noise coefficients $\hat{\varepsilon}_{ij}$ are added to the desired coefficients $\hat{c}_{ij}$

$$\hat{d}_{ij} = \hat{c}_{ij} + \hat{\varepsilon}_{ij} \qquad ; \ \hat{\varepsilon}_{ij} \text{ Noise Terms}$$

$$i = 1, 2, \ldots, t \qquad \text{(E-1)}$$

$$j \in \Lambda_s^{(i)}$$

where

$$\hat{c}_{ij} = \hat{a}_{ij}\hat{\gamma}_{ij} \qquad ; \ \hat{a}_{ij} \text{ Input Terms}$$

$$\hat{\gamma}_{ij} \text{ Stored Terms} \qquad \text{(E-2)}$$

The coefficients of $d_i(x)$ are related to the d_{ij} through the inverse transform:

$$d_{i\ell} = ((n^{-1}))_p \sum_{j \in \Lambda_s^{(i)}} \hat{d}_{ij}\, \alpha^{-\ell j} \quad i = 1, 2, \ldots, t \qquad \text{(E-3)}$$

$$\ell = 0, 1, \ldots, (n-1)$$

where

$$d_i(x) = \sum_{\ell=0}^{n-1} d_{i\ell} x^{\ell} \qquad \text{(E-4)}$$

The inversion process satisfies the Annihilator Property, Lemma D1, Appendix D. (This guarantees that $d_i(x) \in M_i$ provided $d_i(x) \in A_n$).

It will be assumed that the size of the cyclotomic subset $\Lambda_s^{(i)}$, denoted by ℓ_i, is greater than one throughout the remainder of the appendix. The ramifications of this assumption will become apparent as the development progresses. The transform domain errors, $\hat{\varepsilon}_{ij}$, are reflected to the polynomial components as an error polynomial $\varepsilon_i(x)$ by the inversion formula.

$$d_i(x) = c_i(x) + \varepsilon_i(x) \quad ; \begin{array}{ll} \varepsilon_i(x) & \text{Error Polynomial} \\ c_i(x) & \text{Desired Result} \\ i = 1, 2, \ldots, t & \end{array} \qquad \text{(E-5)}$$

If the error coefficients $\hat{\varepsilon}_{ij}$ do not satisfy the Chord Property, Lemma D2 indicates that not all coefficients of $\varepsilon_i(x)$ are in GF(q), an easily detectable condition.

There are simple sufficient conditions for violating the Chord Property.

Lemma E1

If at least one transform coefficient is zero while others are nonzero then these coefficients do not satisfy the Chord Property.

Proof: Let $\hat{\varepsilon}_{ij_0} = 0$ while $\hat{\varepsilon}_{im_0} \neq 0$ for $j_0, m_0 \in \Lambda_s^{(i)}$. Then by the construction of the cyclotomic subset $\Lambda_s^{(i)}$, there is a r_0 such that $j_0 \equiv m_0 q^{r_0} \bmod n$. Assume that the Chord Property holds. Then it follows that

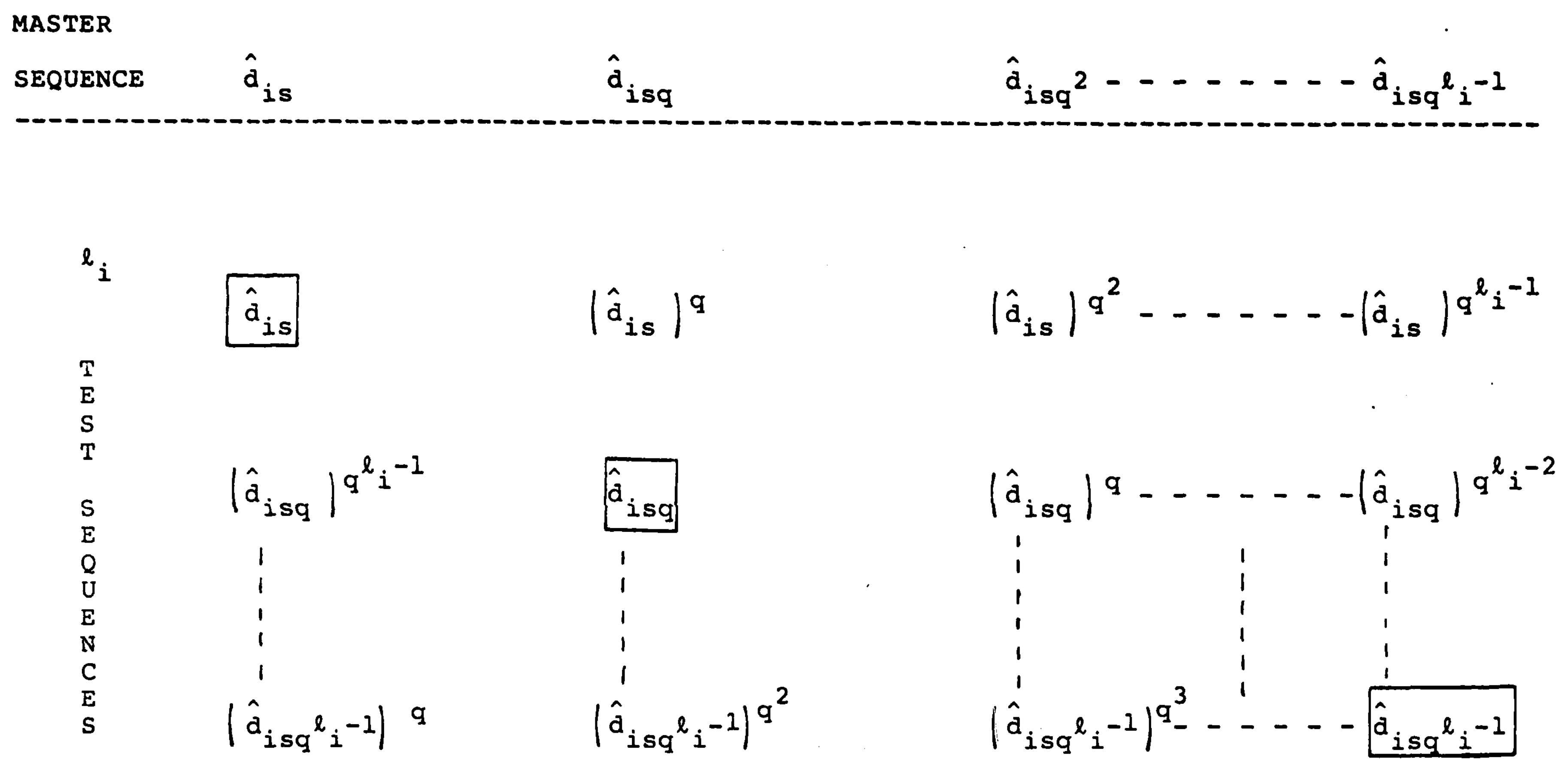
$$(\hat{\varepsilon}_{im_0})^{q^{r_0}} = \hat{\varepsilon}_{im_0 q^{r_0}} = \hat{\varepsilon}_{ij_0} = 0$$

But this is a contradiction of the fact that

$$(\hat{\varepsilon}_{im_0})^{q^{r_0}} \neq 0$$

This Lemma insures that the Chord Property is the basis for detecting up to $(\ell_i - 1)$ errors in the noise coefficients. Thus the size of the cyclotomic subset takes on the same importance as minimum distance does in normal bounded distance decoding [3-5]. The analogy does not stop there. The error-correcting performance in the transform domain which will be developed next extends up to $(\ell_i - 1)/2$ errors in the coefficients.

For error-correcting purposes the transform coefficients $\hat{d}_{ij}$ which are the inputs to the inversion formula (E-3) will be called the Master Sequence. If all of its items are correct, each coefficient could be used, by invoking the Chord Property, to generate ℓ_i different Test Sequences by repeatedly forming the q$\underline{\text{th}}$ power of each successive element. This concept of a Master Sequence and ℓ_i Test Sequences is shown in Figure E-1 where each element in a Test Sequence is listed directly under its correspondingly indexed term in the Master Sequence. The generating coefficient in each Test Sequence is surrounded by a box.



$\hat{d}_{isq^{\ell_i-1}}$

$(\hat{d}_{is})^{q^{\ell_i-1}}$
$(\hat{d}_{isq})^{q^{\ell_i-1}}$

$(\hat{d}_{isq})^{q^{\ell_i-2}}$
$(\hat{d}_{isq^{\ell_i-1}})^q$
$(\hat{d}_{isq^{\ell_i-1}})^{q^2}$
$(\hat{d}_{isq^{\ell_i-1}})^{q^3}$
$\boxed{\hat{d}_{isq^{\ell_i-1}}}$

Array Indexed By $\Lambda_s^{(i)}$

$\boxed{\hat{d}}$ DESIGNATES GENERATING TERM FOR TEST SEQUENCE

The Relationships Between the Master and Test Sequences

Figure E-1

Suppose now that some of the coefficients in the Master Sequence have nonzero error coefficients. Say there are ν where the integer ν is less or equal to $(\ell_i - 1)/2$.

$$\{\hat{\varepsilon}_{ij_1}, \hat{\varepsilon}_{ij_2}, \ldots, \hat{\varepsilon}_{ij_\nu}\} = \text{NONZERO NOISE COEFFICIENTS}$$
$$\nu \leq \frac{\ell_i - 1}{2} \tag{E-5}$$

Further assume, only momentarily though, that the formation of the ℓ_i Test Sequence is error-free. Most test sequences, except those generated by $\hat{d}_{ij_1}, \hat{d}_{ij_2}, \ldots, \hat{d}_{ij_\nu}$ which do not obey the Chord Property, will have more than $\frac{\ell_i}{2}$ correct terms in common with the Master Sequence. The remaining Test Sequences represent questionable positions in the Master Sequence. However the incorrect positions can be determined using the Chord Property and the known correct positions. Alternately a majority decision on a particular position using all the Test Sequences and the Master Sequence gives the correct value for that position.

Now remove the requirement, introduced directly above, that the generation of the Test Sequences be error-free. Say a position in the Master Sequence is correct but that the Test Sequence generated from it contains errors due to the generation process. It is easy to see that as long as the additional number of erroneous Test Sequences combined with the number of original errors does not exceed $\frac{\ell_i - 1}{2}$, then the correction technique using a majority decision will not affect the correct positions.

Appendix F

Fast Finite Field Transform Algorithm:

Development, Properties and Error Propagation

Prime factor fast finite field forward and inverse transform algorithms are developed in this appendix. The intermediate variables used in each of the algorithms possess useful properties concerning chord lengths. Such properties can be employed for distributed error control in system realizations. The propagation and penetration of internal errors are investigated, and, in the case of the inverse transform, the output error polynomials due to single errors within the structure are explicitly determined.

The forward transform will be considered first where both decimation in frequency and decimation in time forms of fast algorithms are developed. Some parts of typical signal flow charts are shown. Analogous developments for the inverse transform are presented in the latter half of the appendix.

The decimation in frequency form of the forward transform stems from the following decomposition of the i and j indices in equation (32).

$$\begin{aligned} i &= i_1 n_2 + i_2 \qquad & i_1, j_1 &= 0,1,\ldots(p-1) \\ j &= j_1 + j_2 p_1 \qquad & i_2, j_2 &= 0,1,\ldots(n_2-1) \end{aligned} \tag{F-1}$$

The prime p_1 is a factor in equation (33) while

$$n_2 = \prod_{i=2}^{M} p_i \qquad \text{(F-2)}$$

Intermediate variables Z_{j_1, i_2} is defined.

$$Z_{j_1,i_2} = \alpha^{j_1 i_2} \sum_{i_1=0}^{p_1-1} a_{i_1 n_2 + i_2} \, (\alpha^{n_2})^{i_1 j_1}$$

The transform coefficient, $\hat{a}_j$, may be expressed as:

$$\hat{a}_{j_1 + j_2 p_1} = \sum_{j_2=0}^{n_2-1} Z_{j_1,i_2} \, (\alpha^{p_1})^{i_2 j_2}; \; j = j_1 + j_2 p_1$$

The development continues in this fashion refining any index whose range limits contain composite numbers. Table F-1a summarizes the straightforward development of a fast transform when n contains four prime factors.

The decimation in time approach may be developed starting with the expansion of indices as:

$$\begin{aligned} i &= i_1 + i_2 p_2 \quad &&; i_1, j_1 = 0,1,\ldots(p_1-1) \\ & && \qquad\qquad\qquad\qquad \text{(F-3)} \\ j &= j_1 n_2 + j_2 \quad &&; i_2, j_2 = 0,1,\ldots(n_2-1) \end{aligned}$$

Thus the desired output variable, say $\hat{a}_j$, may be written in terms of intermediate variables, Y_{j_2, i_1}

INTERMEDIATE VARIABLE DEFINITION	INDEX RANGES
$z_{j_1, i_2} = \alpha^{j_1 i_2} \sum_{i_1=0}^{p_1-1} \alpha_{i_1 n_2 + i_2} (\alpha^{n_2})^{i_1 j_1}$	$0 \le j_1 < p_1,\ 0 \le i_2 < n_2$
$z_{j_1, j_2, i_3} = (\alpha^{p_1})^{j_2 i_3} \sum_{i_2=0}^{p_2-1} z_{j_1, i_2 n_3 + i_3} (\alpha^{p_1 n_3})^{i_2 j_2}$	$0 \le j_2 < p_2,\ 0 \le i_3 < n_3$
$z_{j_1, j_2, j_3, i_4} = (\alpha^{p_1 p_2})^{j_3 i_4} \sum_{i_3=0}^{(p_3-1)} (\alpha^{\frac{n}{p_3}})^{i_3 j_3} z_{j_1, j_2, i_3 n_4 + i_4}$	$0 \le j_3 < p_3\ ,\ 0 \le i_4 < n_4$
$\hat{a}_{j_1 + j_2 p_1 + j_3 p_1 p_2 + j_4 p_1 p_2 p_3} = \sum_{i_4=0}^{(n_4-1)} (\alpha^{p_1 p_2 p_3})^{i_4 j_4} z_{j_1, j_2, j_3, i_4}$	$0 \le j_4 < p_4$

DECIMATION IN FREQUENCY FORM

TABLE F-1a

$$n = p_1 p_2 p_3 p_4$$

$$n = p_1 n_2 \ ; \quad n_2 = p_2 n_3 \ ; \quad n_3 = p_3 n_4 \ ; \quad n_4 = p_4$$

INTERMEDIATE VARIABLES IN A FAST FORWARD TRANSFORM

TABLE F-1

INTERMEDIATE VARIABLE DEFINITION	INDEX RANGES
$Y_{j_4,i_3,i_2,i_1} = (\alpha^{p_1p_2})^{i_3j_4} \sum_{i_4=0}^{n_4-1} (\alpha^{p_1p_2p_3})^{i_4j_4} a_{j_1+i_2p_1+i_3p_1p_2+i_4p_1p_2p_3}$	$0 \leq j_4 < n_4$ $0 \leq i_1 < p_1$, $0 \leq i_2 < p_2$, $0 \leq i_3 < p_3$
$Y_{j_3,i_2,i_1} = (\alpha^{p_1})^{i_2j_3} \sum_{i_3=0}^{p_3-1} (\alpha^{p_1p_2p_4})^{i_3j_3'} Y_{j_4', i_3, i_2, i_1}$	$0 \leq j_3 < n_3$, $0 \leq j_4' < n_4$
where $j_3 = j_3'n_4 + j_4'$	
$Y_{j_2,i_1} = \alpha^{i_1j_2} \sum_{i_2=0}^{p_2-1} (\alpha^{p_1n_3})^{i_2j_2'} Y_{j_3', i_2, i_1}$	$0 \leq j_2 < n_2$, $0 \leq j_3' < n_3$
where $j_2 = j_2'n_2 + j_3'$	
$\hat{a}_{j_1n_2 + j_2} = \sum_{i_1=0}^{p_1-1} (\alpha^{n_2})^{i_1j_1} Y_{j_2, i_1}$	$0 \leq j_1 < p_1$, $0 \leq j_2 < n_2$

DECIMATION IN TIME

TABLE F-1b

$$\hat{a}_{j_1 n_2 + j_2} = \sum_{i_1=0}^{p_1-1} (\alpha^{n_2})^{i_1 j_1} \; Y_{j_2, i_1} \quad ; \; 0 \le j_2 < n_2, \; 0 \le i_1 < p_1 \qquad \text{(F-4)}$$

The next step is to define other intermediate variables by further decomposing indices which range up to n_2 using the prime p_2 and

$$n_3 = \prod_{i=3}^{M} p_i .$$

Table F-1b shows the interrelationship between intermediate variables for a four factor value of n. The input data with indices expressed in a mixed radix notation are used to define intermediate variables Y_{j_4, i_3, i_2, i_1} which in turn are to define the Y_{j_3, i_2, i_1}, etc. See Figure 8b in the text for the general flow of the defining relationships.

The intermediate variables obey the interesting properties typified by equations (34) and (35). They equate the q<u>th</u> power of variables with other variables having specially related indices. Such properties might accurately be called limited chord properties since the related indices are reduced modulo divisors of n. They are based upon the input data lying in the field GF(q). These properties allow distributed error protection within the algorithm.

A proof of one of the properties will be outlined; similar proofs are easily constructed for all of them possibly involving finite induction. Take for example, Z_{j_1, i_2}, and consider its q<u>th</u> power.

$$(Z_{j_1,i_2})^q = \alpha^{(j_1q)i_2} \sum_{i_1=0}^{p_1-1} a_{i_1n_2+i_2} (\alpha^{n_2})^{(j_1q)i_1}; \quad 0 \le j_1 < p_1, \; 0 \le i_2 < n_2 \qquad \text{(F-5)}$$

$$= Z_{j_1q,i_2} \qquad j_1q \text{ modulo } p_1$$

The index (j_1q) is reduced modulo p_1 because of the range of the j_1 variable.

Parts of the signal flow graph for the first stage of the fast forward algorithm are shown in Figure F-1. Note that an error in one input datum can propagate to either p_1 or p_4 new intermediate variables depending on the form. However, because of the limited chord properties, it is possible to perform detection and correction as outlined earlier using $q^{\underline{th}}$ powers of the variables. There is one possible difficulty, though. The chord length must exceed one so as to offer any error control capability.

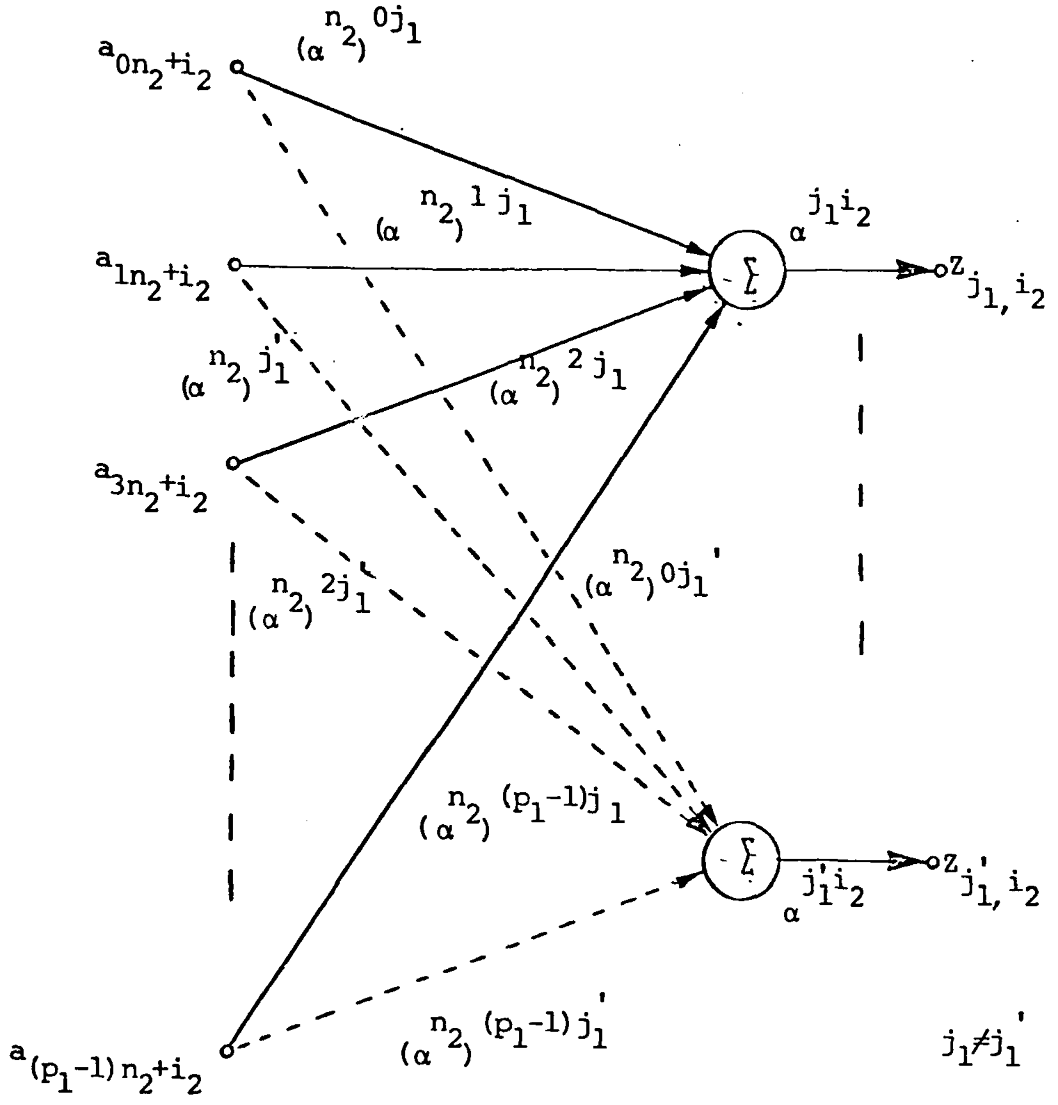
There are constraints that guarantee that all chord lengths, involving non-zero indices, exceed a known integer. This general result will be stated in the nature of a theorem.

<u>Theorem</u> If $(n, (q^i-1)) = 1$ for all $i \le \ell$, there are no integers j and s, $0<j<t$, $s \le \ell$ such that

$$jq^s \equiv j \mod t \qquad \text{(F-6)}$$

where $t|n$

Proof: Since $t|n$, $(t, (q^i-1)) = 1$, $0<i\le\ell$ also.



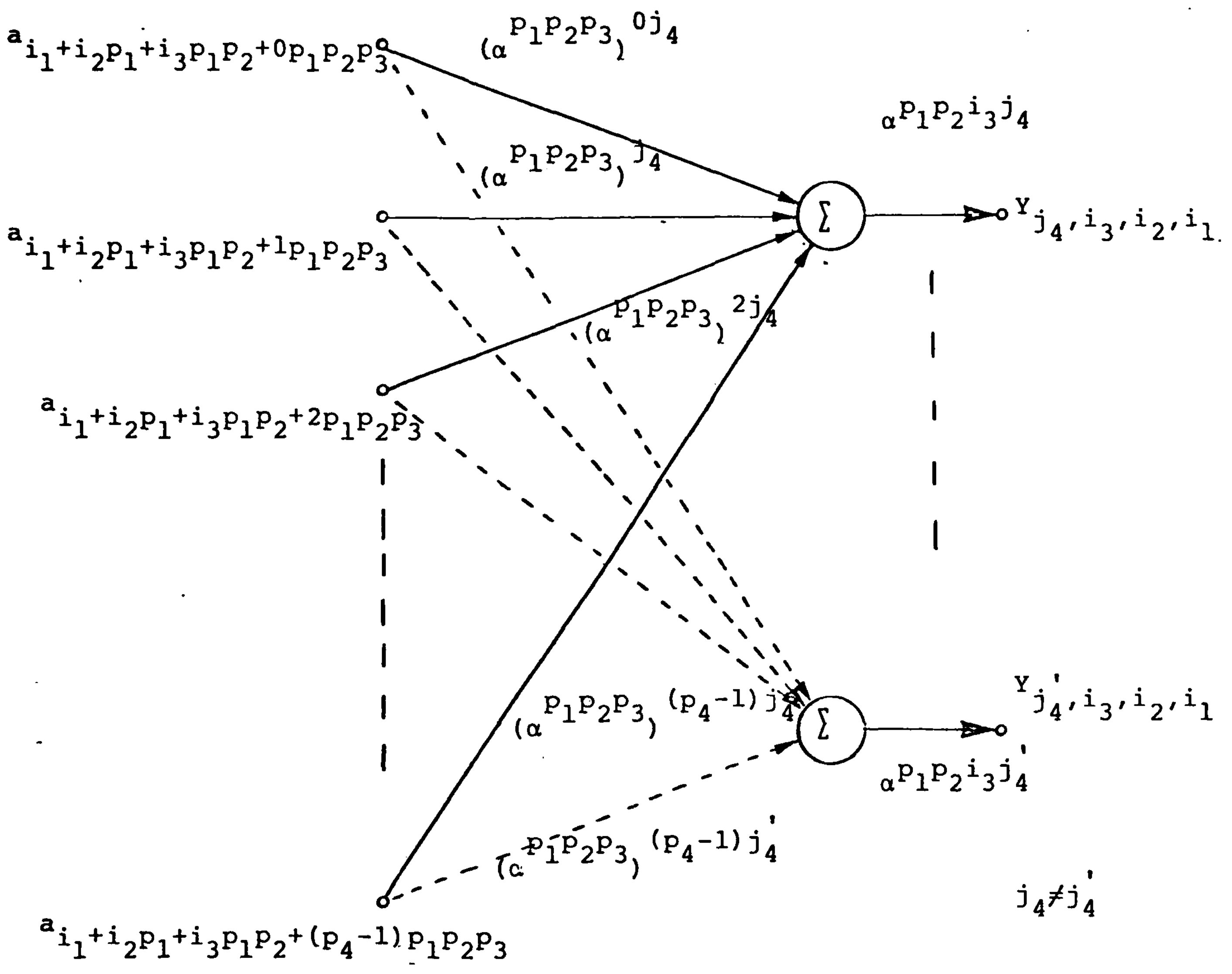
Each input datum enters into the formation of p_1 intermediate variables for each $i_2 = 0, 1, \ldots, (n_2-1)$.

DECIMATION IN FREQUENCY FORM

FIGURE F-1a

PART OF FIRST STAGE, FAST FORWARD ALGORITHM

FIGURE F-1


$a_{i_1+i_2p_1+i_3p_1p_2+0p_1p_2p_3}$
$a_{i_1+i_2p_1+i_3p_1p_2+1p_1p_2p_3}$
$a_{i_1+i_2p_1+i_3p_1p_2+2p_1p_2p_3}$
$a_{i_1+i_2p_1+i_3p_1p_2+(p_4-1)p_1p_2p_3}$
$(\alpha^{p_1p_2p_3})^{0j_4}$
$(\alpha^{p_1p_2p_3})^{j_4}$
$(\alpha^{p_1p_2p_3})^{2j_4}$
$(\alpha^{p_1p_2p_3})^{(p_4-1)j_4}$
$(\alpha^{p_1p_2p_3})^{(p_4-1)j_4'}$
$\alpha^{p_1p_2i_3j_4}$
$\alpha^{p_1p_2i_3j_4'}$
Σ
Σ
Y_{j_4,i_3,i_2,i_1}
Y_{j_4',i_3,i_2,i_1}
$j_4 \neq j_4'$

Each input datum enters into the formation of p_4 intermediate variables for each set of indices $i_3 = 0, 1, \ldots, (p_3-1)$; $i_2 = 0, 1, \ldots, (p_2-1)$; $i_1 = 0, 1, \ldots, (p_1-1)$.

DECIMATION IN TIME FORM

FIGURE F-1b

Suppose that j_o and s_o exist which satisfy the equation. Then $t|j_o(q^{s_o}-1)$. But since $(t, (q^{s_o}-1))=1$ it follows that $t|j_o$ which is impossible because of the range of j.

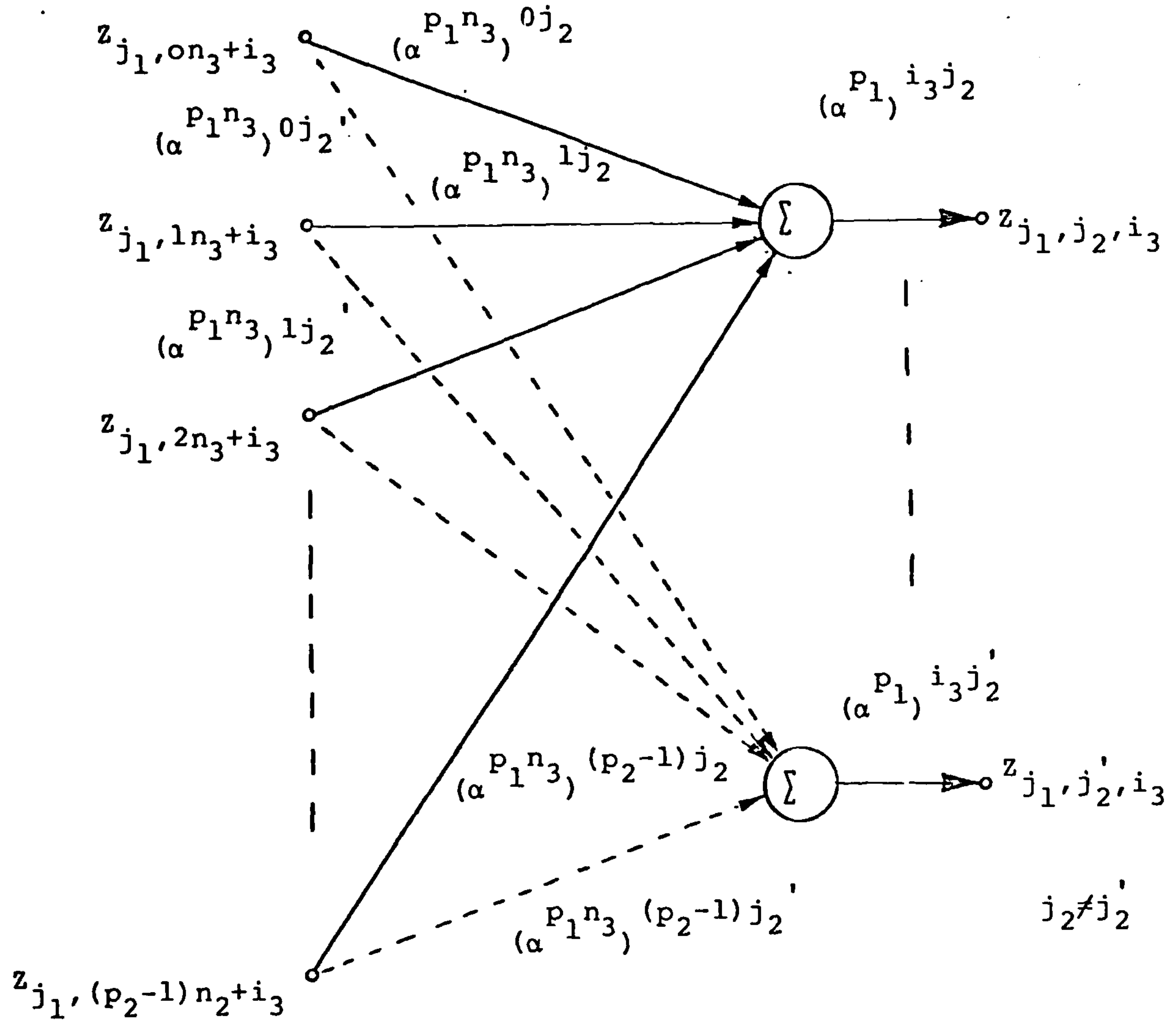
Thus by constraining the length n so that it shares no prime factors with

$$(q^i-1) \qquad ;i\leq \ell \qquad \text{(F-7)}$$

all intermediate variables with *non-zero indices* will be part of chords with lengths exceeding ℓ. The error detection and correction procedures have been outlined earlier. Note, though, that in equation (32), $n|(q^m-1)$ and so ℓ must be less than m.

The intermediate variables with zero indices are in the field GF(q) giving chord lengths of one. Thus, no error protection through chord lengths can be applied. Nevertheless, at the next stage, each such variable has a direct impact on newly defined intermediate variables which are part of error-protected chords. The next stage in the algorithms will be discussed so that this will be apparent.

Figure F-2 displays parts of the second state of each form of the algorithm. It is an easy matter to demonstrate the properties in equations (34) and (35) for the new intermediate variables appearing at the conclusion of this stage. For example, in the case of the decimation in frequency form, the limited chord property applies to the two indices,


z_{j_1,on_3+i_3}
$(\alpha^{p_1n_3})^{0j_2}$
$(\alpha^{p_1n_3})^{0j_2'}$
$(\alpha^{p_1n_3})^{1j_2}$
$(\alpha^{p_1})^{i_3j_2}$
$z_{j_1,1n_3+i_3}$
Σ
z_{j_1,j_2,i_3}
$(\alpha^{p_1n_3})^{1j_2'}$
$z_{j_1,2n_3+i_3}$
$(\alpha^{p_1})^{i_3j_2'}$
$(\alpha^{p_1n_3})^{(p_2-1)j_2}$
Σ
z_{j_1,j_2',i_3}
$(\alpha^{p_1n_3})^{(p_2-1)j_2'}$
$j_2 \neq j_2'$
$z_{j_1,(p_2-1)n_2+i_3}$

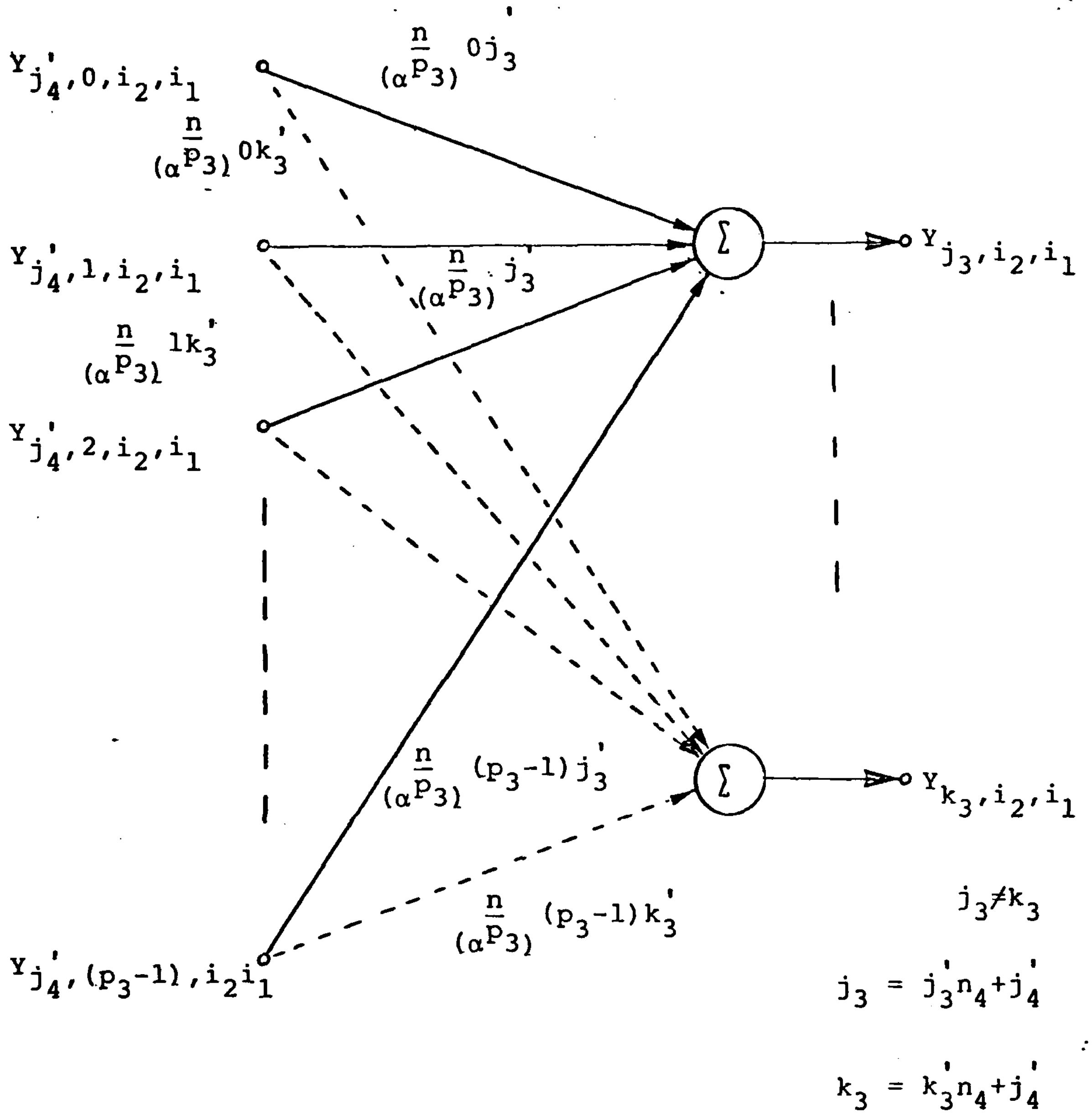
Each Input Variable Affects p_2 New Variables for Each $j_1 = 0, 1, \ldots, (p_1-1)$; $i_3 = 0, 1, \ldots, (n_3-1)$.

DECIMATION IN FREQUENCY FORM

FIGURE F-2a

Part of Second Stage, Fast Forward Algorithm

FIGURE F-2


$Y_{j_4',0,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{0j_3'}$
$(\alpha^{\frac{n}{p_3}})^{0k_3'}$
$Y_{j_4',1,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{j_3'}$
$(\alpha^{\frac{n}{p_3}})^{1k_3'}$
$Y_{j_4',2,i_2,i_1}$
Σ
Y_{j_3,i_2,i_1}
$(\alpha^{\frac{n}{p_3}})^{(p_3-1)j_3'}$
Σ
Y_{k_3,i_2,i_1}
$(\alpha^{\frac{n}{p_3}})^{(p_3-1)k_3'}$
$Y_{j_4',(p_3-1),i_2 i_1}$
$j_3 \neq k_3$
$j_3 = j_3' n_4 + j_4'$
$k_3 = k_3' n_4 + j_4'$

Each Input Variable Affects p_3 New Variables for Each $j_4' = 0, 1, \ldots, (p_4-1)$; $i_2 = 0, 1, \ldots, (p_2-1)$; $i_1 = 0, 1, \ldots, (p_1-1)$.

DECIMATION IN TIME FORM

FIGURE F-2b

j_1 and j_2, of Z_{j_1, j_2, i_3}. In this situation, each index may have different chord lengths, a case to be considered next.

Suppose two non-zero indices, j_1 and j_2 have respective limited chord lengths ℓ_1 and ℓ_2 established by the following requirements

$$\begin{aligned} j_1 q^{\ell_1} &\equiv j_1 \bmod p_1 \quad ; \ \ell_1 \text{ least} \\ j_2 q^{\ell_2} &\equiv j_2 \bmod p_2 \quad ; \ \ell_2 \text{ least} \end{aligned} \tag{F-8}$$

The overall chord length of Z_{j_1, j_2, i_3} describing the behavior when taking repeated qth powers, is easily seen to be the least common multiple of ℓ_1 and ℓ_2.

$$L_{12} = \text{L. C. M. } (\ell_1 ; \ell_2); \quad \text{Overall Chord Length} \tag{F-9}$$

This same principe can be generalized to intermediate variables involving more than two indices in their limited chord properties.

If a constraint similar to indentity (F-7) is imposed on the transform length n, all chord lengths at the second stage will permit error protection except for the case when $j_1=0=j_2$. However, in such a case the defining equation for Z_{j_1, j_2, j_3, i_4}, Table F-1a, shows that any errors in $Z_{0,0,i_3}$, i_3 fixed, propogate to terms like $Z_{0,0,j_3,i_4}$, $0<j_3<p_3$ which appear in nontrival chords. Furthermore there could be numerous chords at this level allowing the source

$z_{0,\,0,\,i_3}$ to be identified.

The general development and error protection available in the fast tranforms are now clear. The fast inverse trnasform will be considered next.

The inverse transform is given by

$$a_j = ((n^{-1}))_p \sum_{j=0}^{n-1} \hat{a}_j \, (\alpha)^{-ij} \;;\; ((x))_p \equiv x \bmod p \qquad \text{(F-10)}$$

where the letters a are used for convenience. In the context of signal processing the inverse transform would be applied to the $\hat{\Delta}_j$ transform coefficients to yield the coefficients in $\delta(x)$ (see Figure 6). The chord property, equation (30), generically stated as,

$$(a_j)^q = a_{jq} \quad ; \; jq \bmod n \qquad \text{(F-11)}$$

imposes constraints on intermediate variables within a fast algorithm. Such algorithms have two forms and may be developed as before. Table F-2 gives the definitions for a four factor version.

A proof of the limited chord properties listed in equations (41) and (42) relies upon the condition $(n,q) = 1$. For example to show that

$$(z_{j_1,\, j_2,\, j_3,\, i_4})^q = z_{j_1 q,\, j_2 q,\, j_3 q,\, i_4} \qquad \text{(F-12)}$$

$$; \; j_1 q \bmod p_1, \; j_2 q \bmod p_2, \; j_3 q \bmod p_3$$

INTERMEDIATE VARIABLE DEFINITION	INDEX RANGES
$\hat{z}_{j_1,j_2,j_3,i_4} = (\alpha^{p_1p_2})^{j_3i_4} \sum_{j_4=0}^{n_4-1} (\alpha^{p_1p_2p_3})^{i_4j_4} \hat{a}_{j_1+j_2p_1+j_3p_1p_2+j_4p_1p_2p_3}$	$0 \le j_4 < n_4$ $0 \le j_3 < p_3$ $0 \le j_2 < p_2$ $0 \le j_1 < p_1$
$\hat{z}_{j_1,j_2,i_3} = \alpha^{p_1j_2i_3} \sum_{j_3=0}^{p_3-1} (\alpha^{p_1p_2p_4})^{i_3'j_3} \hat{z}_{j_1,\ j_2,\ j_3,\ i_4'}$ where $i_3 = i_3'n_4 + i_4'$	$0 \le i_3 < n_3$ $0 \le i_4' < p_4$
$\hat{z}_{j_1,i_2} = \alpha^{j_1i_2} \sum_{j_2=0}^{p_2-1} (\alpha^{p_1n_3})^{i_2'j_2} \hat{z}_{j_1,\ j_2,\ i_3'}$ where $i_2 = i_2'n_3 + i_3'$	$0 \le i_2 < n_2$ $0 \le i_3' < n_3$
$a_{i_1,n_2+i_2} = ((n^{-1}))_p \sum_{j_1=0}^{p_1-1} (\alpha^{n_2})^{i_1j_1} \hat{z}_{j_1,\ i_2}$	$0 \le i_1 < p_1$ $0 \le i_2 < n_2$

$$n = p_1p_2p_3p_4$$

$$n = p_1n_2 \quad ; \quad n_2 = p_2n_3 \quad ; \quad n_3 = p_3n_4 \quad ; \quad n_4 = p_4$$

DECIMATION IN FREQUENCY FORM

TABLE F-2a

INTERMEDIATE VARIABLES IN A FAST INVERSE TRANSFORM

TABLE F-2

INTERMEDIATE VARIABLE DEFINITION	INDEX RANGES
$\hat{Y}_{j_2, i_1} = \alpha^{i_1 j_2} \sum_{j_1=0}^{p_1-1} (\alpha^{n_2})^{i_1 j_1} \hat{a}_{j_1 n_2 + j_2}$	$0 \le j_2 < n_2$ $0 \le i_1 < p_1$
$\hat{Y}_{j_3, i_2, i_1} = (\alpha^{p_1})^{i_2 j_3} \sum_{j_2=0}^{p_2-1} (\alpha^{p_1 n_3}) \hat{Y}_{j_2 n_3 + j_3, i_1}$	$0 \le j_3 < n_3$ $0 \le i_2 < p_2$
$\hat{Y}_{j_4, i_3, i_2, i_1} = (\alpha^{p_1 p_2})^{i_3 j_4} \sum_{j_3=0}^{p_3-1} (\alpha^{p_1 p_2 n_4})^{i_3 j_3} \hat{Y}_{j_3 n_4 + j_4, i_2, i_1}$	$0 \le j_4 < n_4$ $0 \le i_3 < p_3$
$a_{i_1 + i_2 p_1 + i_3 p_1 p_2 + i_4 p_1 p_2 p_3} = ((n^{-1}))_p \sum_{j_4=0}^{(p_4-1)} (\alpha^{p_1 p_2 p_3})^{i_4 j_4} \hat{Y}_{j_4, i_3, i_2, i_1}$	$0 \le i_4 < p_4$

DECIMATION IN TIME FORM

TABLE F-2b

the defining equation from Table F-2a is used.

$$(\hat{z}_{j_1,\, j_2,\, j_3,\, i_4})^q = \alpha^{(j_3 q) i_4 \, p_1 \, p_2} \sum_{j_4=0}^{n_4-1} \alpha^{i_4 (q j_4) p_1 \, p_2 \, p_3} \, \hat{\alpha}_{q j_1 + q j_2 p_1 + q j_3 p_1 p_2 + (q j_4) p_1 p_2 p_3} \tag{F-13}$$

However $(q,n) = 1$ implies $(q,n_4) = 1$ because $n_4 | n$.
So $((q^{-1}))_{n_4} \equiv q^{-1} \bmod n_4$ exists and the change of variables

$$\xi = ((q))_{n_4} \, j_4$$

is one to one mod n_4 since $j_4 = ((q^{-1}))_{n_4} \, \xi$ is the inverse mapping.

$$(\hat{z}_{j_1,\, j_2,\, j_3,\, i_4})^q = \alpha^{j_3 q \; i_4 p_1 p_2} \sum_{\xi=0}^{p_4-1} (\alpha^{p_1 p_2 p_3})^{i_4 \xi} \, \hat{\alpha}_{q j_1 + q j_2 p_1 + q j_3 p_1 p_2 + \xi p_1 p_2 p_3} = \hat{z}_{q j_1,\, q j_2,\, q j_3,\, i_4} \tag{F-14}$$

A proof of $(Y_{j_2, i_2})^q = Y_{q j_2, i_2}$ uses the same reasoning and similarly for the remaining properties in equations (41) and (42).

The limited chord properties can be used for error control in the inverse transform algorithm just as before, and the sufficient conditions (F-7) guarantee a minimum length for all the chords involving nonzero indices. However errors in terms with zero indices can be a source of difficulty because they are elements in GF(q). The fast inverse transform algorithm systematically produces some intermediate values in GF(q) at each stage and uses them in the formation of new intermediate values which are also in GF(q) at the next stage. Therefore in some instances there is no useful chord property available for error control. On the other hand though the error-correcting code over the final n output values can handle these propagating errors.

Typical parts of the last two stages of the decimation in frequency form are shown in Figure F-3, When an error, ε, occurs in intermediate variable $\hat{Z}_{0,\, i_2'}$, at the input to the last stage errors appear in p_1 terms

$$a_{i_1' n_2 + i_2'} + \varepsilon \quad ; \; i_1' = 0, 1, \ldots, (p_1 - 1).$$

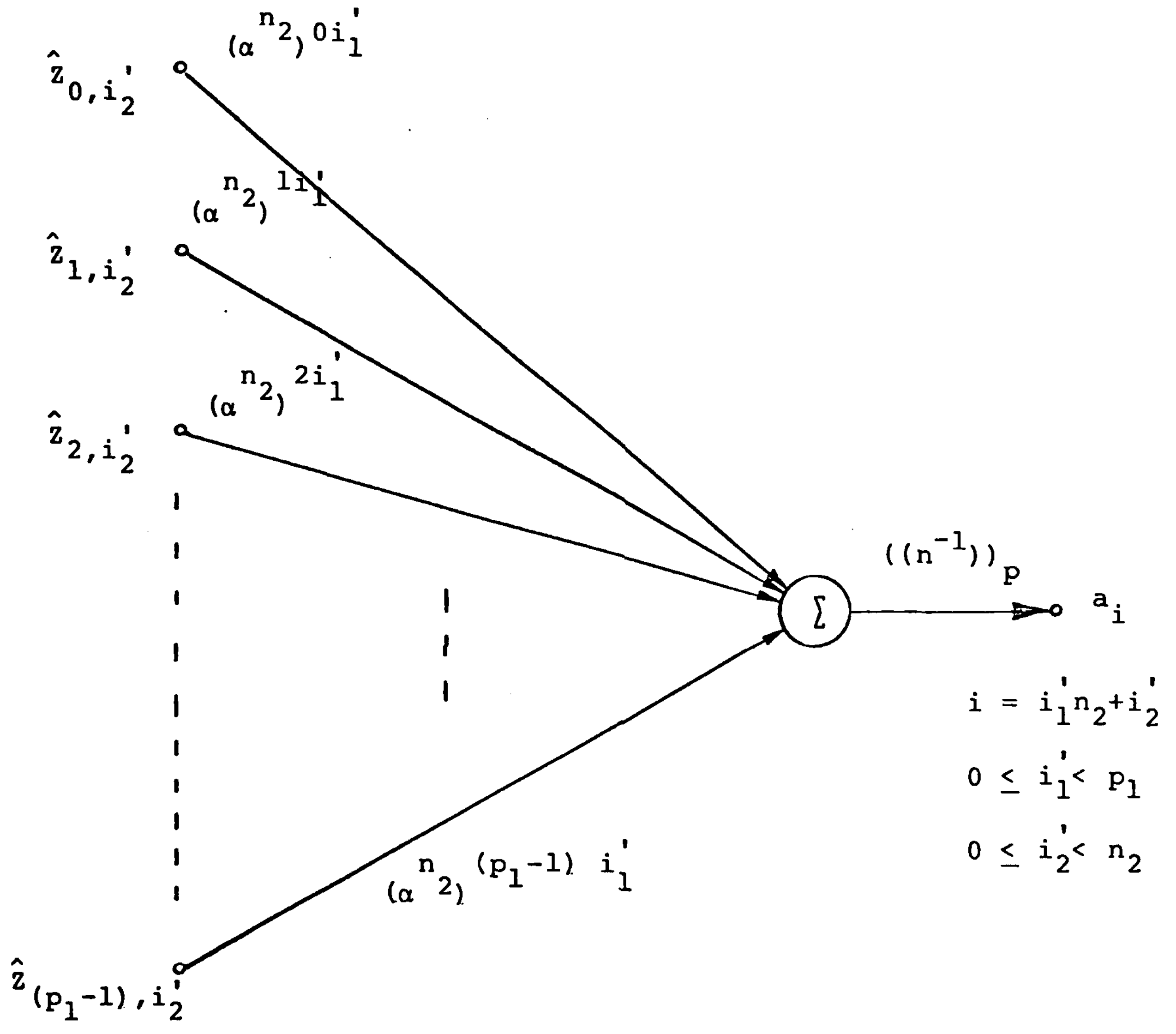
The error components in the output are additive and can be expressed as an error polynomial.

$$e(x) = \varepsilon\, x^{i_2'} \sum_{i=0}^{p_1-1} x^{in_2} = \varepsilon\, x^{i_2'} \left[\frac{x^n - 1}{x^{n_2} - 1} \right] \qquad \text{(F-15)}$$

If this single additive error, ε, occurred in the next to last stage, say in term $\hat{Z}_{0,\, 0,\, i_3}$, the same additive error value ε will appear in the $(p_1 p_2)$ output values

$$a_{i_1 n_2 + i_2 n_3 + i_3} + \varepsilon \quad ; \; \begin{array}{l} i_1 = 0, 1, \ldots, (p_1 - 1) \\ i_2 = 0, 1, \ldots, (p_2 - 1) \end{array}$$

The error polynomial corresponding with these $(p_1 p_2)$ errors is



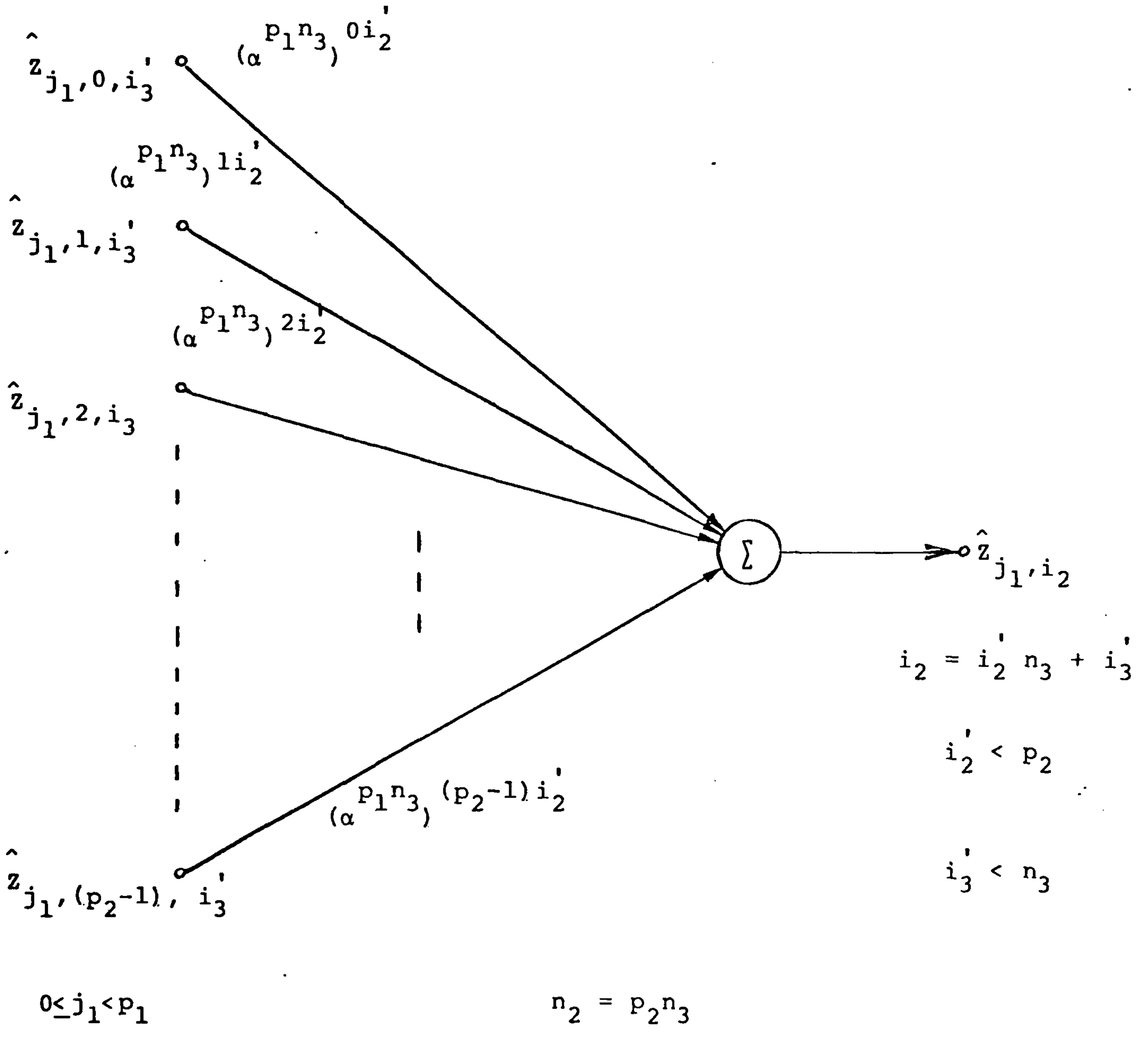
$n = p_1 n_2$

PART OF LAST STAGE

FIGURE F-3a

PART OF DECIMATION IN FREQUENCY FORM, FAST INVERSE TRANSFORM

FIGURE F-3



PART OF NEXT TO LAST STAGE

FIGURE F-3b

$$e(x) = \varepsilon\, x^{i_3} \sum_{i_1=0}^{p_1-1} \sum_{i_2=0}^{p_2-1} (x^{i_2 n_3})\,(x^{i_1 n_2})$$

$$= \varepsilon\, x^{i_3} \left[\frac{x^n-1}{x^{n_3}-1}\right] \quad ; \quad \frac{n}{n_3} = p_1 p_2 \qquad \text{(F-16)}$$

For the decimation in frequency form, the output errors resulting from internal errors in various stages is summarized in Table 1 of the text.

The propagation of errors in the decimation in time form behaves in an analogous manner. The last two stages are exemplified by the parts in Figure F-4 from which error penetration effects may be studied. Carefully analyzing terms with first index set to zero shows how internal errors in GF(q) terms pass through to the output. The error polynomials associated with such error are also given in Table 1.

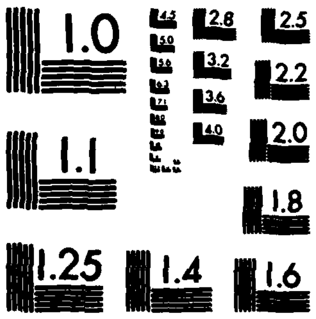
The error polynomials for either form represent very special types of error patterns, one with identical errors in well-defined locations. The few errors that can propagate to the output can be detected and corrected by the overall cyclic codes if its generator polynomial, g(x), is properly chosen. It must not divide any of the possible error polynomials. Since the inverse algorithm is additive, all possible output errors, caused by internal failure producing GF(q) errors in vulnerable intermediate variables, can be constructed by linear combinations of the error polynomials already given.

AD-A133 927 FAULT TOLERANT SIGNAL PROCESSING USING FINITE FIELDS AND ERROR-CORRECTING..(U) RENSSELAER POLYTECHNIC INST TROY NY CENTER FOR INTEGRATED ELE.. G R REDINBO JUN 83 2/2

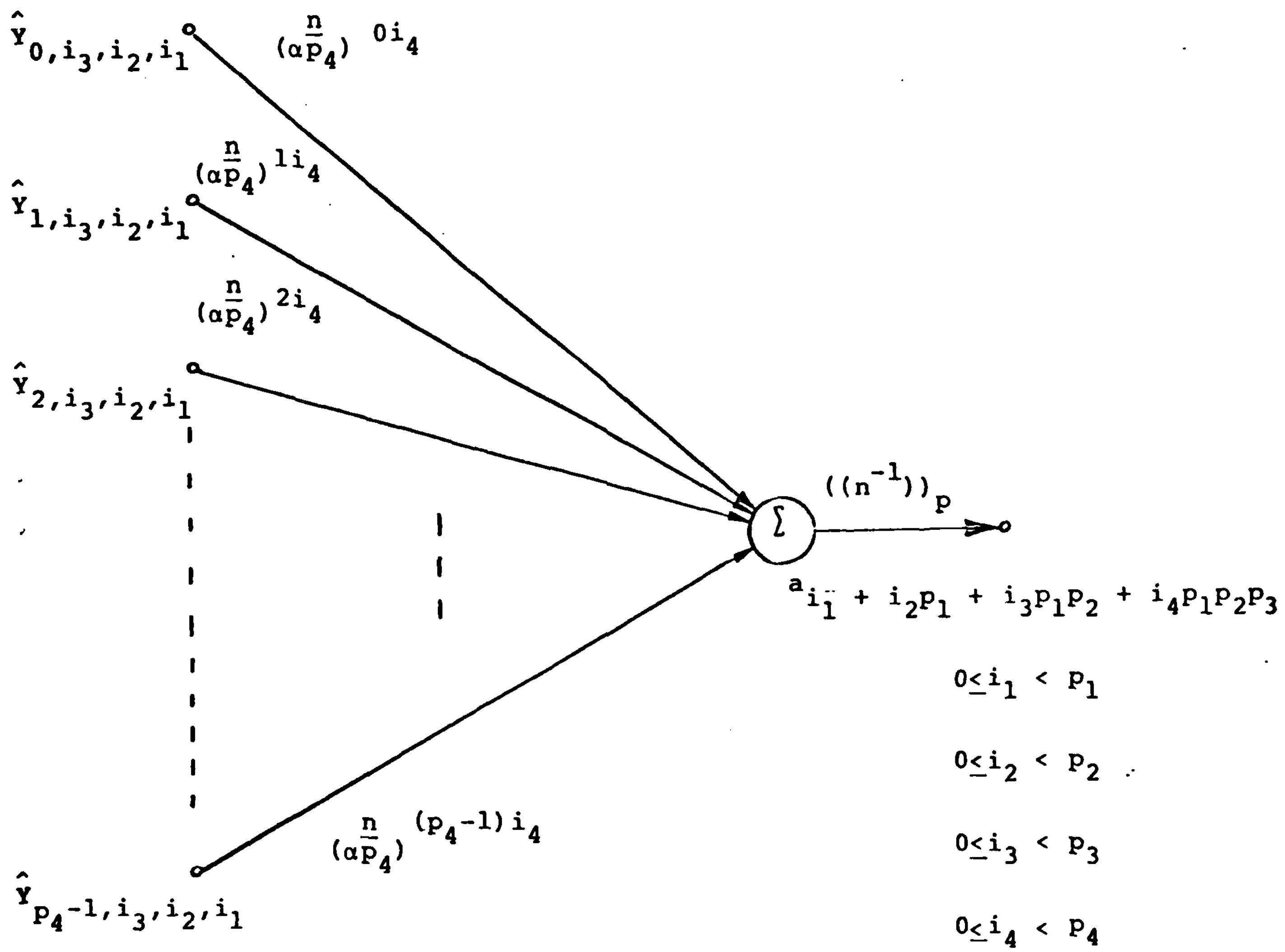
UNCLASSIFIED AFOSR-TR-83-0839 AFOSR-80-0153 F/G 12/1 NL

END

FILMED



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



$\hat{Y}_{0,i_3,i_2,i_1}$
$(\alpha_{\overline{p_4}}^{n})^{0i_4}$
$\hat{Y}_{1,i_3,i_2,i_1}$
$(\alpha_{\overline{p_4}}^{n})^{1i_4}$
$\hat{Y}_{2,i_3,i_2,i_1}$
$(\alpha_{\overline{p_4}}^{n})^{2i_4}$
$\hat{Y}_{p_4-1,i_3,i_2,i_1}$
$(\alpha_{\overline{p_4}}^{n})^{(p_4-1)i_4}$
Σ
$((n^{-1}))_p$
$a_{i_1 + i_2p_1 + i_3p_1p_2 + i_4p_1p_2p_3}$

$0 \le i_1 < p_1$

$0 \le i_2 < p_2$

$0 \le i_3 < p_3$

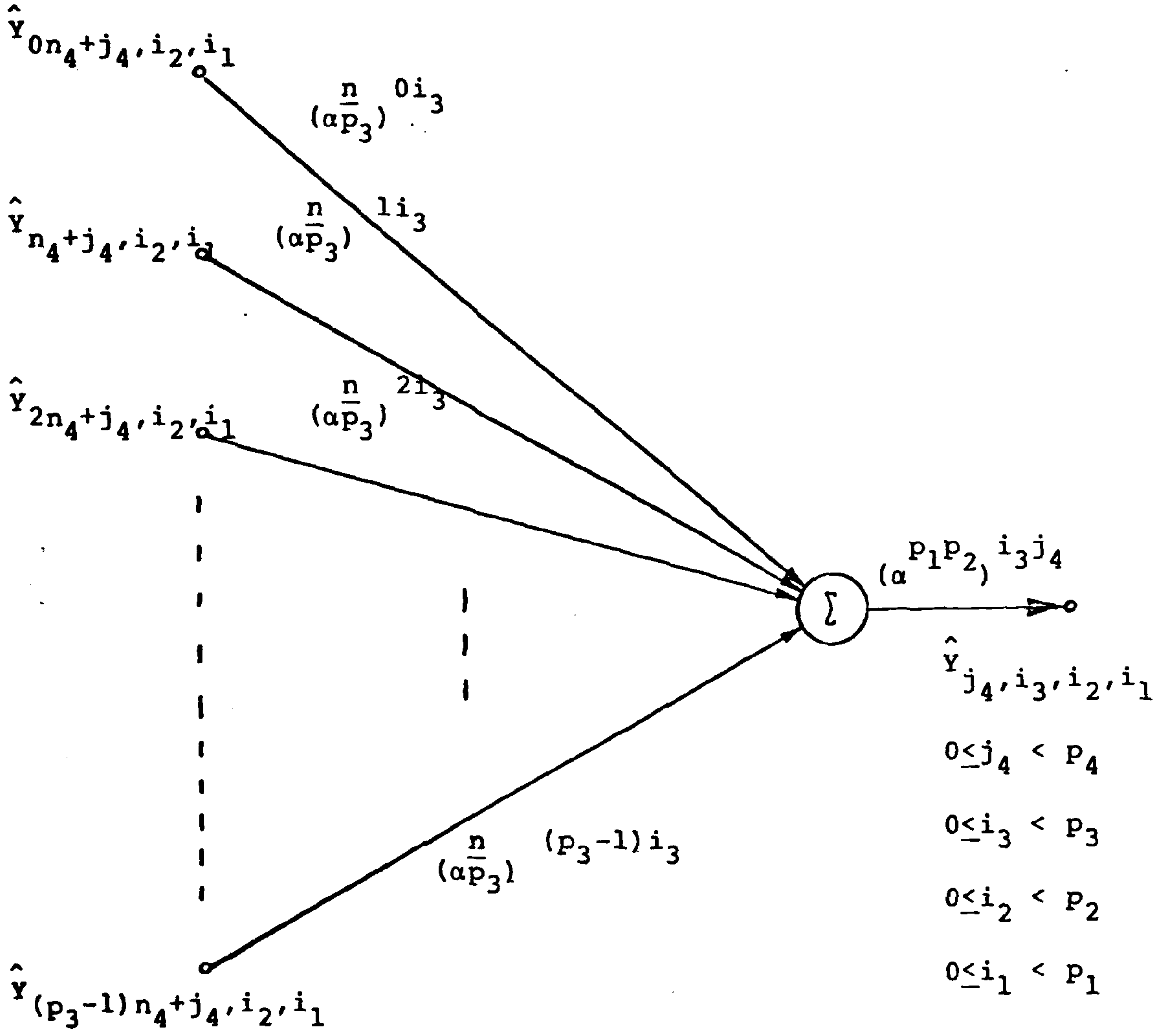
$0 \le i_4 < p_4$

PART OF LAST STAGE

FIGURE F-4a

PART OF DECIMATION IN TIME FORM, FAST INVERSE TRANSFORM

FIGURE F-4


$\hat{Y}_{0n_4+j_4,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{0i_3}$
$\hat{Y}_{n_4+j_4,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{1i_3}$
$\hat{Y}_{2n_4+j_4,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{2i_3}$
$\hat{Y}_{(p_3-1)n_4+j_4,i_2,i_1}$
$(\alpha^{\frac{n}{p_3}})^{(p_3-1)i_3}$
Σ
$(\alpha^{p_1p_2})^{i_3j_4}$
$\hat{Y}_{j_4,i_3,i_2,i_1}$
$0 \le j_4 < p_4$
$0 \le i_3 < p_3$
$0 \le i_2 < p_2$
$0 \le i_1 < p_1$

PART OF NEXT TO LAST STAGE

FIGURE F-4b