OPTIMAL VLSI STRUCTURE FOR HIGH SPEED PIPELINE USING RNS

Giuseppe Alia, Enrico Martinelli

Progetto finalizzato
"Materiali e Dispositivi per l'Elettronica a Stato Solido"

Nota interna B4-42

Ottobre 1990

OPTIMAL VLSI STRUCTURE FOR HIGH SPEED PIPELINE FFT USING RNS

G. Alia
Istituto di Elettronica e Telecomunicazioni - Facoltà di Ingegneria dell'Università di Pisa
Via Diotisalvi, 2 - 56100 Pisa - Italy

and

E. Martinelli
Istituto di Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56100 Pisa - Italy

The work described in this paper has been supported by the National Program on Solid-State Electronics Materials and Devices of the National Research Council of Italy and by the Convention between IET of the University of Pisa and Olivetti S.p.A.

1 - INTRODUCTION

Designing computing structures for FFT with ever lower response time has been an attractive goal for several years and many algorithms and architectural solutions have been introduced. Out of the proposed techniques, the use of residue number systems (RNS) proved to be profitable to speed up computations, not only for FFT [1, 2], but for many other problems as well, in the area of digital signal processing (DSP). Actually, additions and multiplications are mainly involved in DSP, with results expected within known ranges; on the other hand, RNS-based arithmetic units are highly efficient just for addition and multiplication, since they are carry-free operations. Moreover RNS allow to design modular and regular structures and therefore are well suited for VLSI implementations, to vantage of area saving.

In this work a VLSI structure, based on RNS units, to perform the N point FFT on a continuous data stream is proposed, and its performance is evaluated in terms of asymptotic VLSI complexity.

A lower bound on complexity was found for the FFT problem in the form $A(N)T^2(N) = \Omega(N^2\log^2 N)$ [3]; such a bound was reached by constructive designs with a response time $T(N) = \vartheta(\log^2 N)$ [4, 5]. The architecture proposed here is based on a very high degree of processing parallelism and on a communication parallelism tailored to the response time of adders and multipliers used in the design; furthermore, under data pipelining, it works as an optimal design with a pipeline interval $T'(N) = \vartheta(\log\log\log N)$.

The FFT Formulation

Let us consider the discrete Fourier transform over N points $x_0(k)$, k = 0, 1, ..., N-1

$$X(n) = \sum_{k=0}^{N-1} x_0(k) W^{nk}, \quad n=0, 1,\ldots, N-1 \tag{1}$$

where $W = \exp(-\imath 2\pi/N)$. When $N = 2^\gamma$, γ integer valued, and denoting by k_h and n_h, h = 0, ..., γ-1, the h-th bit of the binary representations of k and n, expression (1) can be rewritten as [6]:

$$\begin{aligned} X(n_{\gamma-1}, n_{\gamma-2},\ldots,n_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\cdots\sum_{k_{\gamma-1}=0}^{1} & x_0(k_{\gamma-1}, k_{\gamma-2},\ldots,k_0) \\ & \times W^{2^{\gamma-1} n_0 k_{\gamma-1}} . W^{2^{\gamma-2}(2n_1+n_0)\, k_{\gamma-2}} \cdots \\ & \times W^{2^0(2^{\gamma-1} n_{\gamma-1}+2^{\gamma-2} n_{\gamma-2}+\ldots+n_0)k_0} \end{aligned} \tag{2}$$

Performing the summations in (2) separately, intermediate results X_j, $j = 1, ..., \gamma$, each consisting in a vector of N components $x_j(i)$, $i = 0, ..., N-1$, indexed by proper bits of n and k, can be evaluated by means of the following sets of equations [6], which were formulated by Cooley and Tukey:

$$X_1 = \left\{ x_1(i) = x_1(n_0, k_{\gamma-2}, \ldots, k_0) = \sum_{k_{\gamma-1}=0}^{1} x_0(k_{\gamma-1}, k_{\gamma-2}, \ldots, k_0) \cdot W^{2^{\gamma-1} n_0 k_{\gamma-1}} \right\}$$

$$X_2 = \left\{ x_2(i) = x_2(n_0, n_1, k_{\gamma-3}, \ldots, k_0) = \sum_{k_{\gamma-2}=0}^{1} x_1(n_0, k_{\gamma-2}, \ldots, k_0) \cdot W^{(2n_1+n_0)2^{\gamma-2} k_{\gamma-2}} \right\}$$

$$\vdots \qquad (3)$$

$$X_\gamma = \left\{ x_\gamma(i) = x_\gamma(n_0, n_1, \ldots, n_{\gamma-1}) = \sum_{k_0=0}^{1} x_{\gamma-1}(n_0, n_1, \ldots, k_0) \cdot W^{(2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \ldots + n_0) k_0} \right\}$$

$$X(n_{\gamma-1}, n_{\gamma-2}, \ldots, n_0) = x_\gamma(n_0, n_1, \ldots, n_{\gamma-1})$$

Each set of equations (3) can be evaluated by means of a set of N processing elements $\{PE_{ij}\}$ and then a total of NlogN processing elements arranged as an array of N rows and logN columns is required to compute the Fourier transform.

The VLSI Model

We refer to the following assumptions, generally accepted in the literature, to model technological constraints:

a) $\lambda = \vartheta(\text{const})$ is the minimal wire width and the minimal distance between parallel wires is $\vartheta(\lambda)$;
b) wires run horizontally and vertically and only two wires may cross at any point;
c) one bit of stored information requires area $\vartheta(\lambda^2)$;
d) a transistor of area $\vartheta(\text{const})$ needs a transit time $\tau = \vartheta(\text{const})$ to change its state;
e) the propagation of a binary change along a wire requires $\vartheta(\tau)$; to meet this assumption, wires of length $\vartheta(L)$ must be driven by drivers with area $\vartheta(\lambda) \cdot \vartheta(L)$.

Residue Number Systems (RNS)

In a residue number system an integer $0 \le X \le 2^n-1$ is represented by means of p residue digits α_i, where $\alpha_i = |X|_{m_i} = X-\lfloor X/m_i \rfloor \cdot m_i$, $\lfloor X/m_i \rfloor$ denotes the largest integer not exceeding X/m_i, and $\{m_i\}$ is the set of moduli. If moduli m_i are pairwise relatively prime, it can be shown [7] that there is a unique representation for each number X in the range $0 \le X < \prod_{i=1}^{p} m_i = M$.

Moreover, for any pair of integers X and Y, the following relations hold:

$$|X \pm Y|_{m_i} = \left| |X|_{m_i} \pm |Y|_{m_i} \right|_{m_i}, \qquad |X \cdot Y|_{m_i} = \left| |X|_{m_i} \cdot |Y|_{m_i} \right|_{m_i}$$

and, consequently, addition and multiplication can be performed for each modulus separately.

It is easy to show that previous notation can be extended to signed arithmetic. In fact, assuming M even for simplicity, an implicit sign representation can be taken by associating to any relative integer X, with $-2^{n-1} \le X \le 2^{n-1}-1$, an integer Y in the range [0, M) defined by the relation:

$$Y = \left\{ \begin{array}{ll} X & \text{if } X \ge 0 \\ M-|X| & \text{if } X < 0 \end{array} \right\}$$

Thus signed arithmetic operations can be carried out in the residue form without needing sign knowledge, which is difficult to obtain rapidly in RNS's. Moreover, after the RNS-to-positional conversion, the resulting non-negative number N, with $0 \le Y < M$, carries an implicit representation of the sign of the actual result X, which can be obtained in its range [-M/2, M/2) as follows:

$$X = \left\{ \begin{array}{ll} Y & \text{if } Y < M/2 \\ Y-M & \text{if } Y \ge M/2 \end{array} \right\}$$

In this article we assume $m_i = \vartheta(\log M)$, $1 \le i \le p$, and, consequently,

$$M = \sum_{i=1}^{p} m_i = \vartheta((\log M)^p) \cdot \vartheta(\text{const}^p)$$

$$\log M = \vartheta(p \log\log M), \qquad p = \vartheta(\log M / \log\log M)$$

This assumption simply states that all moduli are of the same order; it can be noted that the number of bits necessary to represent all residue digits of X, adding up to

$\vartheta(\log\log M)$, is of the same order as the number of bits necessary to represent X in the weighted number system.

Finally, we assume, as in [3], that $M > N$, $\log M = O(\log N)$, that is $\log M = \vartheta(\log N)$.

2 - THE STRUCTURE OF PROCESSING ELEMENTS

Each processing element PE_{ij} of the array introduced in previous section can perform the following computations:

$$x_j(i) = x_{j-1}(s)+x_{j-1}(t)W^r$$

where

$$s = i-i_{\gamma-j}\cdot 2^{\gamma-j}, \quad t = i+(1-i_{\gamma-j})\cdot 2^{\gamma-j}, \quad r = |N|_{2^j}\cdot 2^{\gamma-j}$$

and $i_{\gamma-j}$ is the $(\gamma-j)$th bit in the binary representation of $0 \le i \le N-1$. Note that the computation consists of the sum of one input with the other input multiplied by a constant. Denoting input data by x and y and the output by z and recalling that operands are complex numbers, previous equation can be rewritten as

$$Re(z) = Re(x)+Re(y)Re(W^r)-Im(y)Im(W^r) \tag{4}$$

$$Im(z) = Im(x)+Re(y)Im(W^r)+Im(y)Re(W^r) \tag{5}$$

where all operands may assume non-integer values.

Nevertheless, integer values are to be provided to these equations to perform residue arithmetic. To this purpose, let S be a positive integer, then we can write:

$$Re(z) = \frac{1}{S}\,(Re(x)S+Re(y)Re(W^r)S-Im(y)Re(W^r)S) \tag{4'}$$

$$Im(z) = \frac{1}{S}\,(Im(x)S+Re(y)Im(W^r)S+Im(y)Re(W^r)S) \tag{5'}$$

Assuming that input data are integer valued and substituting $Re'(W^r) = \lfloor Re(W^r)S \rfloor$, $Im'(W^r) = \lfloor Im(W^r)S \rfloor$ for $Re(W^r)S$ and $Im(W^r)S$ respectively we obtain:

$$Re(z) = \left\lfloor \frac{1}{S}\,(Re(x)S+Re(y)Re'(W^r)-Im(y)Im'(W^r)) \right\rfloor \approx \left\lfloor \frac{Re'(z)}{S} \right\rfloor \tag{4''}$$

$$Im(z) = \left\lfloor \frac{1}{S} (Im(x)S+Re(y)Im'(W^r)+Im(y)Re'(W^r)) \right\rfloor \approx \left\lfloor \frac{Im'(z)}{S} \right\rfloor \qquad (5'')$$

Let $[\lfloor -M/2 \rfloor, \lfloor M/2 \rfloor)$ be the range necessary to represent system inputs and outputs. Then Re(z) and Im(z) can be considered mod M:

$$|Re(z)|_M \approx \left| \left\lfloor \frac{Re'(z)}{S} \right\rfloor \right|_M = \left| \frac{Re'(z) - |Re'(z)|_S}{S} \right|_M =$$

$$= \left| \left| \frac{1}{S} \right|_M (|Re'(z)|_M - | |Re'(z)|_S |_M) \right|_M \qquad (4''')$$

Similarly,

$$|Im(z)|_M \approx \left| \left| \frac{1}{S} \right|_M (|Im'(z)|_M - | |Im'(z)|_S |_M) \right|_M \qquad (5''')$$

It must be observed that previous expressions are defined provided that the multiplicative inverse $|1/S|_M$ exists, i.e., GCD(S, M) = 1. Moreover, from equalities (4''') and (5''') it results that $|Re'(z)|_S$ and $|Im'(z)|_S$ must be evaluated. As $|Re'(z)|_S = ||Re'(z)|_{MS}|_S$ and $|Im'(z)|_S = ||Im'(z)|_{MS}|_S$, it is sufficient to perform computations (4''') and (5''') in the range MS, by adding p' moduli m_{p+1}, m_{p+2}, ..., $m_{p+p'}$, such that $\prod_{i=p+1}^{p+p'} m_i = S$, to the residue representation system of data; in this way a base extension is carried out. In the following we assume that $S = \vartheta(M)$.

The organization of any PE_{ij} consists of three parts, as shown in Fig. 1. The base extension structure generates residue digits of input data for moduli from m_{p+1} to $m_{p+p'}$. The second structure computes $|Re'(z)|_{MS}$ and $|Im'(z)|_{MS}$ as defined in equations (4'') and (5''). The scaling structure is necessary to represent results produced by the computation structure in the range M.

The computation structure is shown in Fig. 2 and consists of 2(p+p') computation units, each evaluating $|Re'(z)|_{m_i}$ or $|Im'(z)|_{m_i}$, and of a permutation network which allows feeding signals $|Re(x)|_{m_i}$, $|Im(x)|_{m_i}$, $|Re(y)|_{m_i}$ and $|Im(y)|_{m_i}$, $1 \le i \le p+p'$ to proper computation units.

Each computation unit is arranged as shown in Fig. 3 and computes $|Re'(z)|_{m_i}$ and $|Im'(z)|_{m_i}$ according to equations (4'') and (5''). Three mod m_i multipliers, as the optimal multipliers defined in [8], are used, which require inputs and produce outputs divided into substrings. Three register files store substrings of constants $|Im'(W^r)|_{m_i}$, $|Re'(W^r)|_{m_i}$, $|S|_{m_i}$. Moreover, two mod m_i adders, accepting input data substrings, are cascaded and a FIFO register allows proper pairing of substrings for the second adder inputs. A mod m_i adder is sketched in Fig.4; it consists of two binary adders as

described in [9], the first computing $|x|_{m_i}+|y|_{m_i}$, and the second subtracting m_i to test for overflow. A multiplexer controlled by the sign of the result selects the proper sequence of substrings.

The base extension structure is formed by four substructures computing $|Re(x)|_{m_k}$, $|Im(x)|_{m_k}$, $|Re(y)|_{m_k}$ and $|Im(y)|_{m_k}$, $p+1 \leq k \leq p+p'$ in parallel; in the following only the substructure concerning the evaluation of $|Re(x)|_{m_k}$ is described for simplicity and it is outlined in Fig. 5. This structure is replicated in four copies allowing to extend real and imaginary parts of input operands in parallel. It implements the base extension procedure described in [7] which is arranged as a sequence of p modular subtractions and multiplications by proper constants. Other authors recently proposed a more efficient technique [10], which could be used alternatively, however the complexity of the whole FFT structure would not be affected by such a choice. In our system the implementation consists of a cascade of p arrays of modular addition and multiplication blocks. The j-th array, $1 \leq j \leq p$ contains p-j+p' blocks; each block $C_{j,h}$, $j+1 \leq h \leq p+p'$, uses modular adders and multipliers like those described for the computation unit. Thus each block requires inputs and produces outputs divided into substrings and its outputs can directly feed the computation unit inputs.

Computations involved in (4''') and (5''') are performed by the structure shown in Fig. 6. A preliminary base extension of the quantities $|Re'(z)|_S$ and $|Im'(z)|_S$ to the range M, so providing $||Re'(z)|_S|_M$ and $||Im'(z)|_S|_M$, is required. At this purpose the same structure shown in Fig. 5 can be repeated, except that the extension from p' moduli to p moduli is now to be performed. Once $||Re'(z)|_S|_M$ and $||Im'(z)|_S|_M$ are obtained, subtraction and scaling by S can be carried out on the residue digits separately, by means of two arrays of p blocks like those of Fig. 5; consequently the structure of Fig. 6 can use substrings produced by computation units, as inputs.

3. - COMPLEXITY EVALUATION

3.1 - Area complexity

The overall system is depicted in Fig. 7, from which one can immediately deduce that the vertical dimension is determined by the largest among the vertical extent of N processing elements rows and the vertical extent of the 2^{j-1} connection networks used at stage j, $1 \leq j \leq \gamma = \log N$. Similarly, the horizontal size depends on the horizontal extents of the logN processing elements plus the logN connection stages.

To derive the complexity of the dimensions of processing elements, refer again to Fig. 1. In this figure, complexity measures are reported for each structure. Moreover we refer to the VLSI mod m multiplier proposed in [8] in which each operand is considered as divided into s_M substrings of w/s_M bits, where w is the length of

operands, sequentially fed to the inputs. This multiplier can be designed optimally for any value of $s_M(w)$ within the range $[\vartheta(\log w), \vartheta(\sqrt{w})]$. In such hypothesis, it exhibits multiply time $T_M(w) = \vartheta(s_M)$ and requires area $A_M = \vartheta((w/s_M)^2) = \vartheta((w/T_M)^2)$. Adding and subtracting blocks can be designed using for the structure of Fig. 4 the adder proposed in [9], such blocks also consider operands of length w divided into s_A substrings and exhibit computation time $T_A=\vartheta(s_A+\log(w/s_A))$, and area $A_A = \vartheta(w/s_A\cdot(s_A+\log(w/s_A)))$. Assuming $s_A = s_M$, still ranging in $[\vartheta(\log w), \vartheta(\sqrt{w})]$, the expression for T_A reduces to $T_A =\vartheta(s_M) = \vartheta(T_M)$ and, as for area, $A_A = \vartheta(w/T_M\cdot(T_M+\log(w/T_M))) = \vartheta(w)$.

Area occupancies of the base extension, computation and scaling structures are then mainly determined by modular multipliers, which exhibit $A_M = \vartheta((\log m/T_M)^2)$, and by the number of moduli $p+p' = \vartheta(p)$. In fact also the permutation network of Fig. 2 occupies a similar area $\vartheta((p\log m/T_M)^2)$, so this is the overall area occupancy of each processing element. Choosing $T_M = \vartheta(\log\log m)$, which is the lower extreme of the range of optimality and corresponds to the fastest design, and recalling assumptions on p, m, M and N, we can write:

$$A_{PE} = \vartheta\left(\frac{\log^2 N}{(\log\log\log N)^2}\right)$$

As for the networks in Fig. 7 connecting processing elements in adjacent columns, they have the same structure, except the number of inputs and outputs. Denoting the number of inputs of any network by $I = 2^{\log N-j+1}$, each network connects input in position h, $0 \le h \le I-1$, to processing elements in positions h and $|h+I/2|_I$. According to the VLSI model rules, each network can be implemented in a squared area $\vartheta(I)\times\vartheta(I)$. Assume that each connection consists of ω wires, ω depending on the parallelism degree of communications, $\vartheta(\text{const}) \le \omega \le \vartheta(p\log m) = \vartheta(\log N)$, the total area per network is $\vartheta(\omega I)\times\vartheta(\omega I)$. It can be observed that, at each stage the total number of network inputs is N and then the vertical size of the area dedicated to communications is $\vartheta(\omega N)$. However, the width of each network at stage i is $\vartheta(\omega 2^{\log N-i+1}) = \vartheta(\omega N/2^{i-1})$, and therefore the total network horizontal size is

$$\sum_{i=1}^{\log N} \vartheta\left(\omega\frac{N}{2^{i-1}}\right) = \vartheta\left(\omega N\sum_{i=1}^{\log N}\frac{1}{2^{i-1}}\right) = \vartheta(\omega N)$$

As inputs and ouputs of any PE are subdivided into $T_M = \vartheta(\log\log m)$ strings, ω can be given the value $\omega = \vartheta(\frac{p\log m}{\log\log m}) = \vartheta(\frac{\log N}{\log\log\log N})$ and the complexity of the communication area is $\vartheta(\frac{N\log N}{\log\log\log N})\times\vartheta(\frac{N\log N}{\log\log\log N})$.

Then the total area of the FFT architecture is

$$A = \vartheta\left(\frac{N\log N}{\log\log\log N}\right) \times \vartheta\left(\frac{N\log N}{\log\log\log N}\right) = \vartheta\left(\frac{N^2\log^2 N}{(\log\log\log N)^2}\right)$$

3.2 - Time complexity

To evaluate time complexity it is convenient to recall that all modular multipliers and modular adders used in our structure are suited for pipeline operation, that is T_M strings cross $\vartheta(\log m/T_M) = \vartheta(\log m/\log\log m)$ stages, each with a constant delay.

The whole structure can operate in a similar way: as the time complexity is expressed in this case as the stage delay times the sum of the number of strings and the number of stages, the total time is

$$T = \vartheta(\log\log\log N + \frac{\log^2 N}{\log\log\log N})$$

In fact there are $\log N$ columns, each containing $\vartheta(p) = \vartheta(\log N/\log\log N)$ multipliers with $\vartheta(\log\log N/\log\log\log N)$ stages and $\vartheta(p)$ adders of negligible complexity.

It can be observed that the proposed system is very suited to compute the FFT on a stream of sets of N points as it is encountered in several real time applications. Let k be the number of FFT instances to be computed. Total time can be now written as

$$T' = \vartheta(k\log\log\log N + \frac{\log^2 N}{\log\log\log N})$$

If $k = \Omega(\log^2 N/\log\log\log^2 N)$, total time expression reduces to $T' = \vartheta(k\log\log\log N)$, that is the system works as though time necessary to compute an FFT on N points were $\vartheta(\log\log\log N)$.

4 - CONCLUDING REMARKS

The problem of computing FFT in VLSI systems has been considered in [3] with the aim of evaluating the complexity of the problem. In particular a lower bound on VLSI complexity has been obtained in the form $A_0T_0^2 = \Omega(N^2\log^2 N)$ which holds

independently of the adopted data representation system; such a bound is related to the solution of a single instance of the FFT problem.

In this paper a VLSI system based on RNS to compute the FFT has been presented: under the constraint of pipeline operation with a stream of at least one instance every $\vartheta(\log\log\log N)$ time units, for a total of $\Omega(\log^2 N/\log\log\log^2 N)$ instances, the presented system works as an optimal design, that is $AT'^2 = \Omega(N^2\log^2 N)$.

In the literature optimal designs of VLSI systems to compute the FFT have been presented [4, 5, 11, 12]; the major advantage of the solution proposed in this work is the lower computation time ($\vartheta(\log^2 N/\log\log\log N)$ versus $\vartheta(\log^2 N)$), which moreover reduces to $\vartheta(\log\log\log N)$ under pipeline operation. On the other hand the main drawback is a larger area $A = \vartheta(N^2\log^2 N/\log\log\log^2 N)$ to be supplied even for a single FFT computation.

REFERENCES

[1] Taylor, F. J. and C. Huang, A comparison of DFT algorithms using a residue architecture, Comput. and Electr. Eng. **8** (3) (1981) 161-171.

[2] Alia, G., F. Barsi and E. Martinelli, A fast near optimum VLSI implementation of FFT using residue number system, INTEGRATION, The VLSI Journal, **2** (1984) 133-147.

[3] Thompson, C.D., A complexity theory for VLSI , Ph.D. Thesis, Rept. CMU-CS-80-140, Carnegie-Mellon University, Dept. of Computer Science, 1980.

[4] Preparata, F. P. and J. E. Vuillemin, The cube connected cycles: a versatile network for parallel computation, Comm. ACM **24** (5) (1981) 300-309.

[5] Preparata, F. P. and J. E. Vuillemin, Area-time optimal VLSI networks for computing integer multiplication and discrete Fourier transform, ICALP 81, Acre, Israel, July 1981, in Lecture Notes in Computer Science, **115**, Springer-Verlag, Berlin, 1981.

[6] Brigham, E. O. *The fast Fourier transform* (Prentice-Hall Inc., Englewood Cliffs N.J., 1974).

[7] Szabo, N.S. and R.J. Tanaka, *Residue Arithmetic and its Applications to Computer Technology* (McGraw-Hill, New York, 1967).

[8] Alia, G. and E. Martinelli, A VLSI Modulo m Multiplier, under revision for publication on IEEE Trans. Comput.

[9] Brent, R.P. and H.T. Kung, A regular layout for parallel adders, IEEE Trans. Comput. **C31** (3) (1982) 260-264.

[10] Shenoy, A.P. and R. Kumaresan, Fast base extension using a redundant modulus in RNS, IEEE Trans. Comput. **C38** (2) (1989) 292-296.

[11] Bongiovanni, G., Two VLSI structures for the discrete Fourier transform, IEEE Trans. Comput., **C-32** (8) (1983) 750-753.

[12] Bongiovanni, G., A VLSI network for variable size FFT's, IEEE Trans. Comput., **C-32** (8) (1983) 756-760.

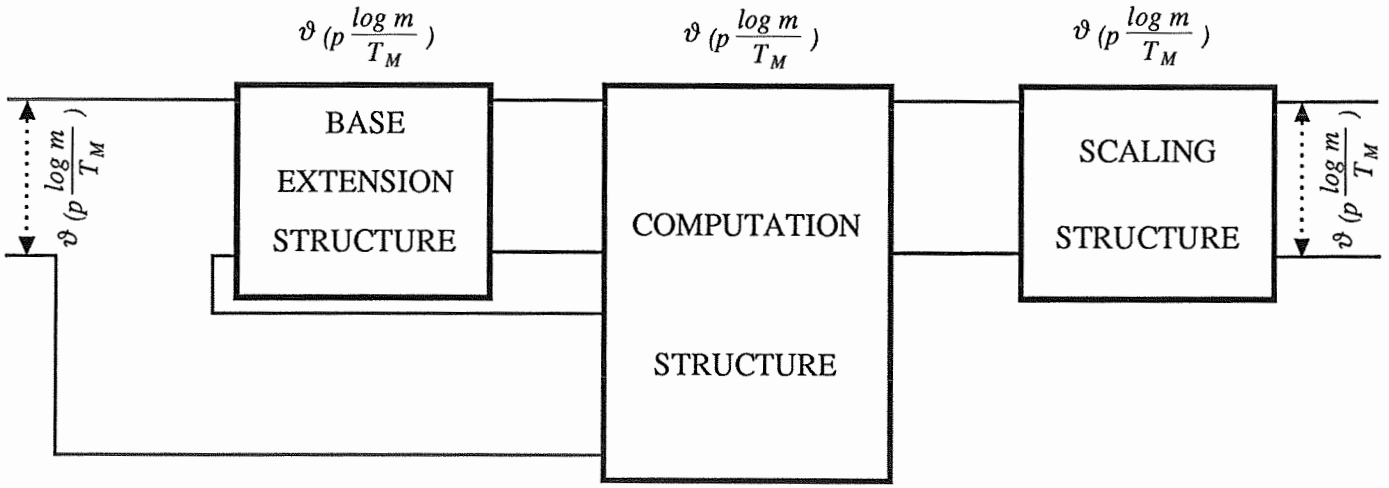


Fig. 1 - The organization of a Processing Element (PE)

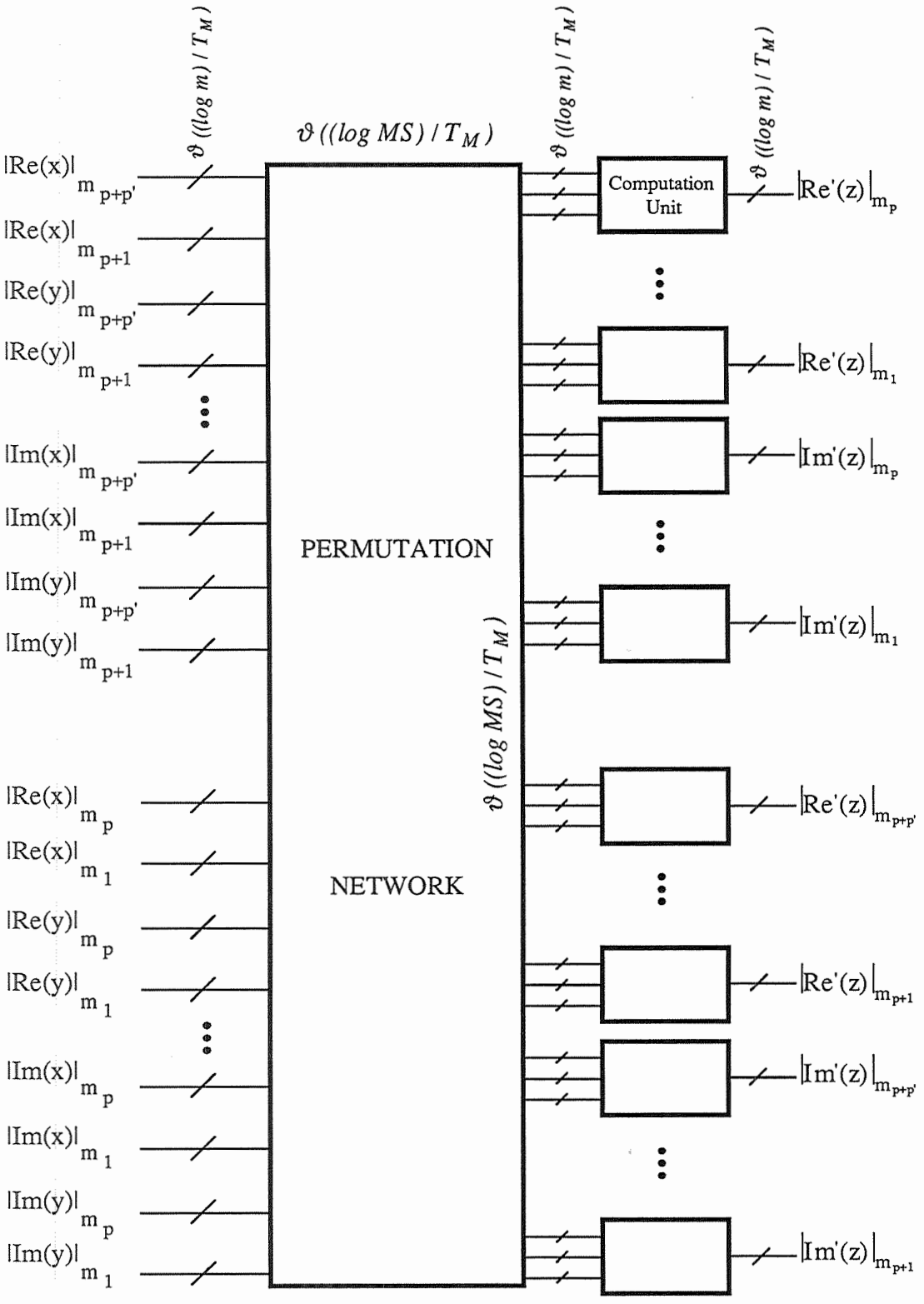


Fig. 2 - The computation structure

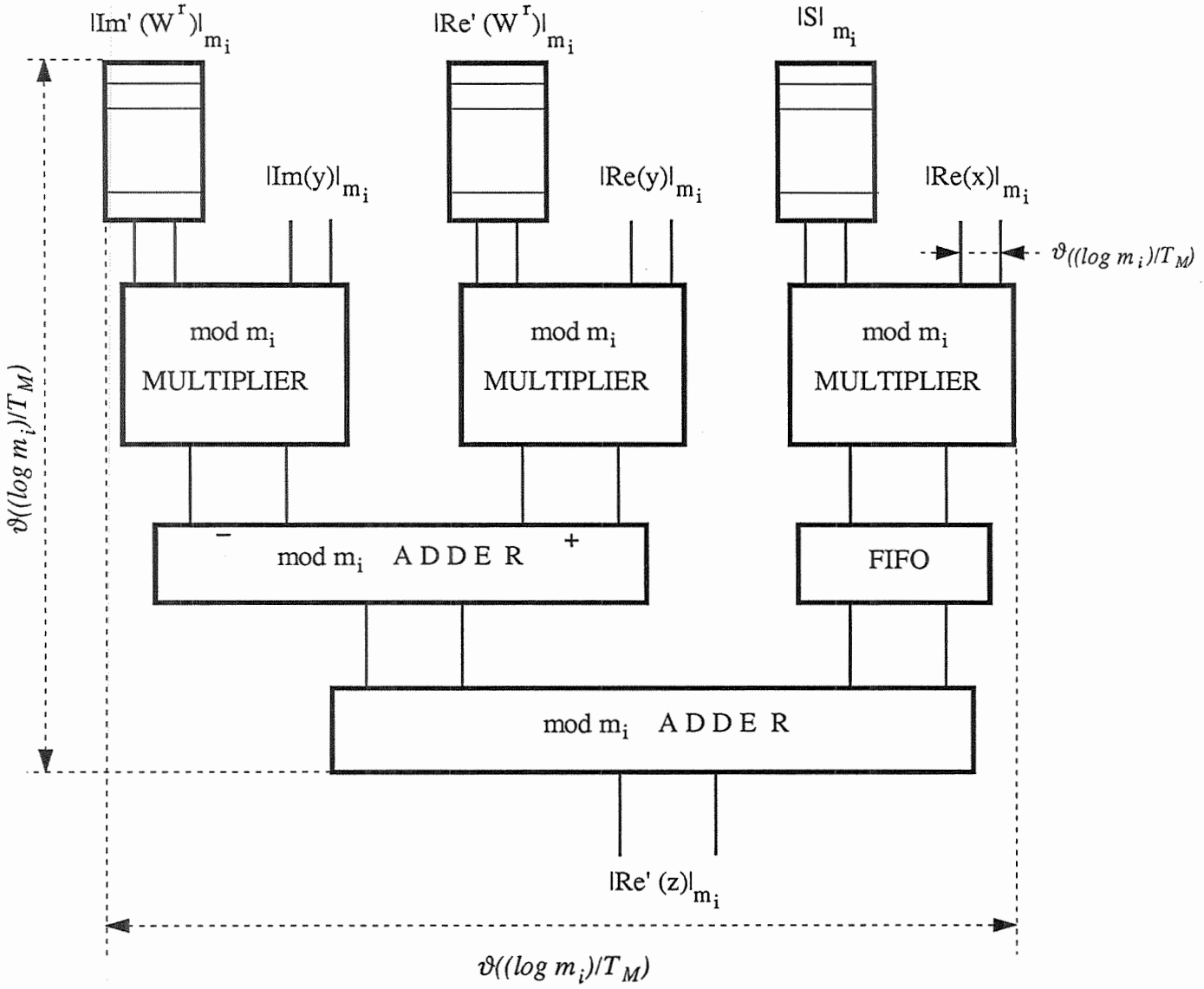


Fig. 3 - The computation unit for $|Re'(z)|_{m_i}$

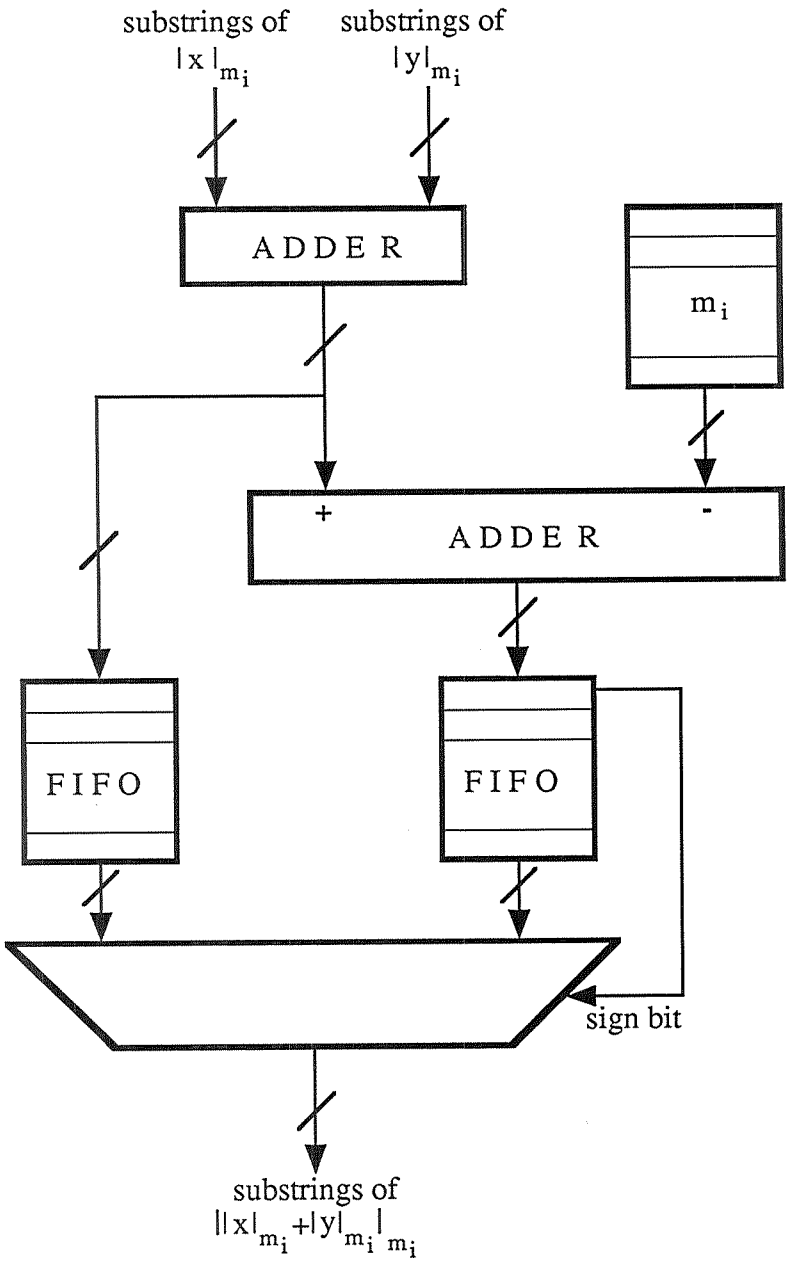
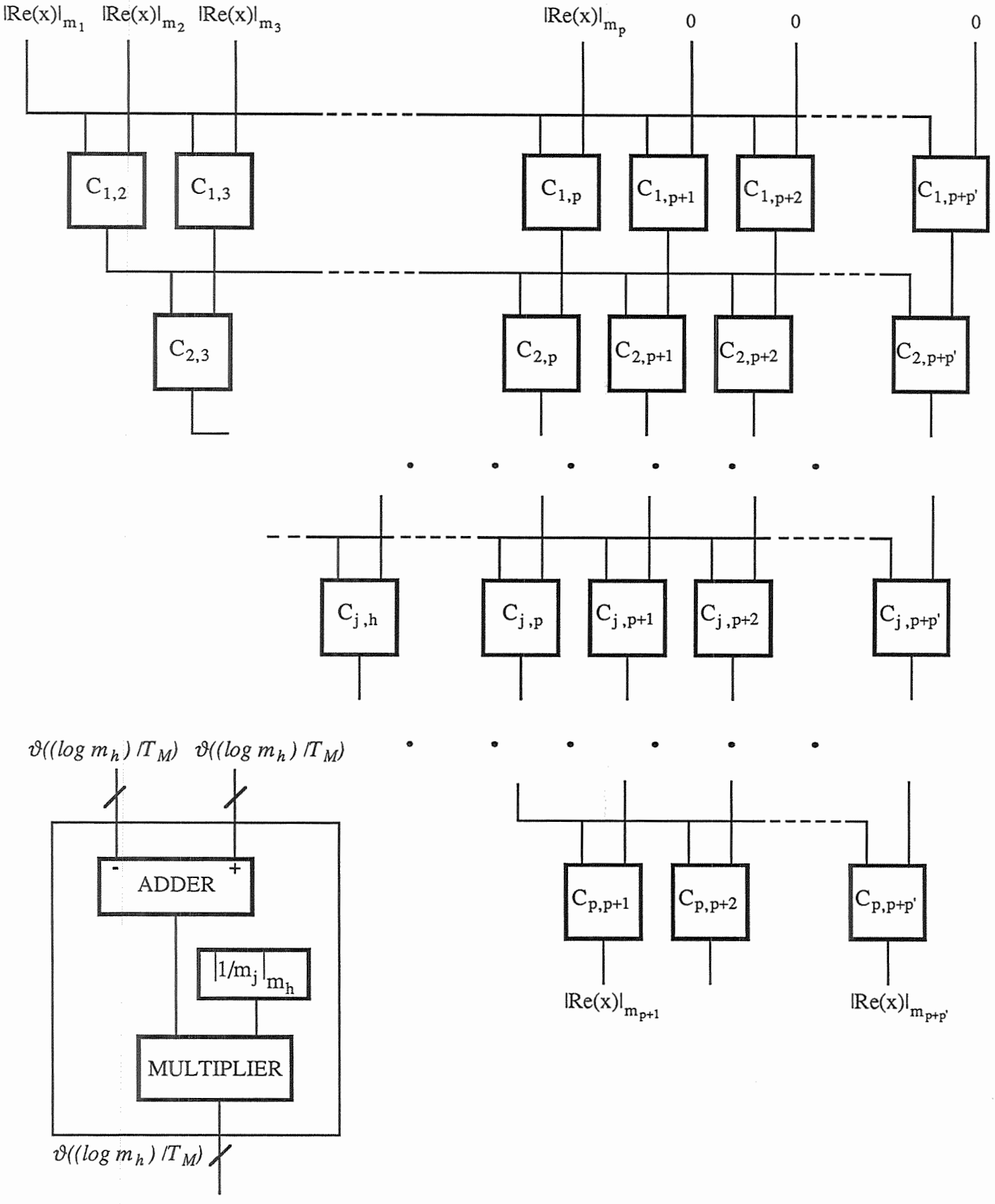


Fig. 4 - The organization of a modular adder



the structure of block $C_{j,h}$

Fig. 5 - The Base Extension Structure for $|Re(x)|_{m_k}$, $p+1 \le k \le p+p'$

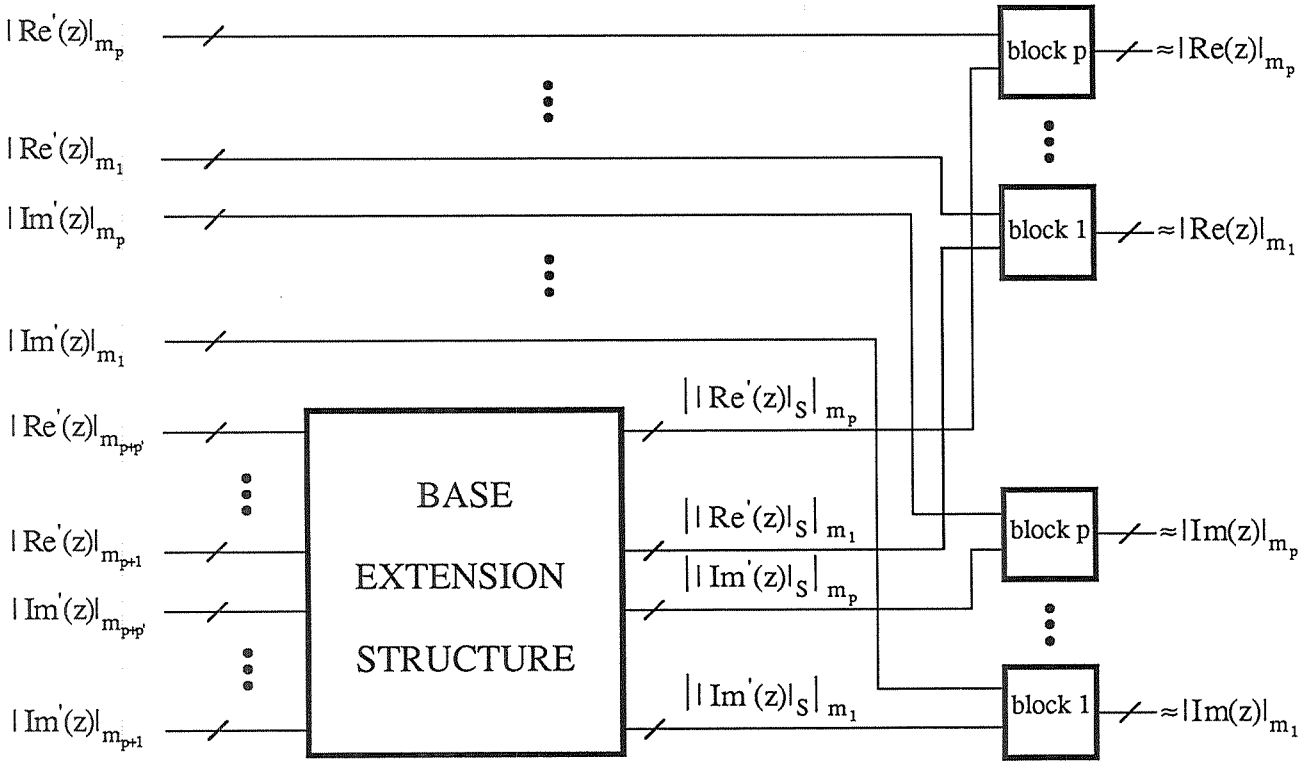


Fig. 6 - Scaling structure

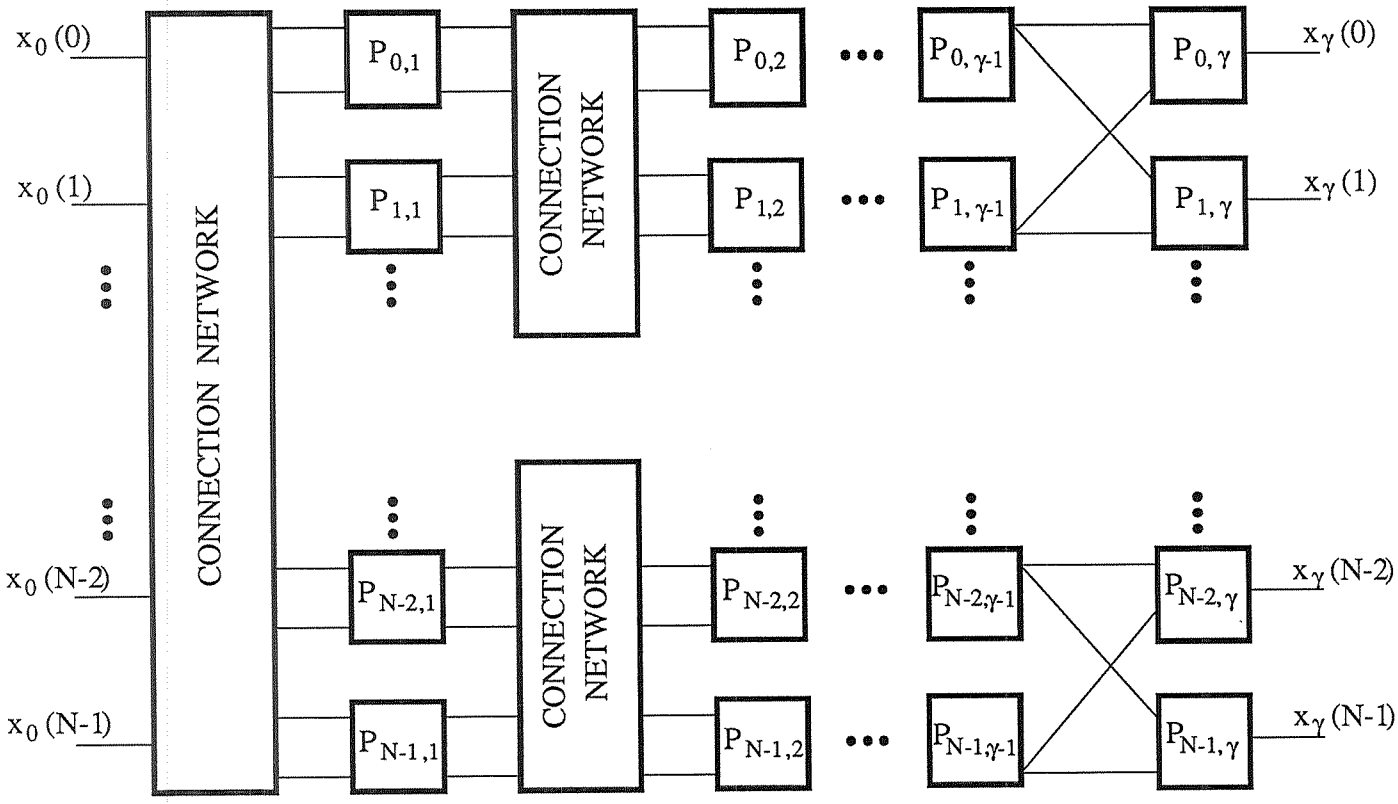


Fig. 7 - The overall system