A New Transform Algorithm for Viterbi Decoding

KUEI-ANN WEN, TING-SHIUN WEN, MEMBER, IEEE, AND JHING-FA WANG, SENIOR MEMBER, IEEE

Abstract—**Implementation of the Viterbi decoding algorithm has attracted a great deal of interest in many applications, but the excessive hardware/time consumptions caused by the dynamic and backtracking decoding procedures make it difficult to design efficient VLSI circuits for practical applications.**

A new transform algorithm for maximum likelihood decoding is derived from trellis coding and Viterbi decoding processes. Dynamic trellis search operation are paralleled and well formulated into a set of simple matrix operations referred to as the Viterbi transform (VT).

Based on the VT, the excessive memory accesses and complicated data transfer scheme demanded by the trellis search are eliminated. Efficient VLSI array implementations of the VT have been developed. Long constraint length codes can be decoded by combining the processors as the building blocks.

I. Introduction

THE Viterbi decoding algorithm (VA) [1], [2] is a maximum likelihood decoding algorithm for convolutional codes [3]. Its implementation in VLSI circuits is increasingly in demand for diversified applications such as deep-space and satellite communications, text and voice recognitions, and many others [4]–[7]. However, the excessive storage requirements and implementation complexity can be major problems, especially with codes of longer constraint lengths. A number of special architectures for VA have been proposed [8]–[10] and numerous algorithmic variations have been explored [12], [13] to facilitate hardware implementations and reduce computation time. Chang and Yao proposed the combination of several stages with a strongly connected trellis diagram [11], our work follows their merging philosophy, but results in a new algorithm which greatly reduces data transfer and access requirements.

We have analyzed the encoding computations of convolutional codes and developed by integration algorithm for Viterbi decoding operations. The algorithm will be called the Viterbi transform (VT). By computational modifications, the decoding processes of several stages are efficiently transformed into simple matrix operations, and branch code generation is integrated into the decoding processes. The dominant time consumptions in VLSI circuits caused by data access operations are greatly reduced, as this transformation limits the memory access to a loccal access without global memory requirements and the backtracking is eliminated.

A modular processor element is developed to construct the Viterbi decoder for long constraint length codes. An efficient systolic VT processor for (2, 1, 4) code is presented; it may be used as the building block for any m with m being radix 4.

II. Convolutional Coding System

An (n, k, m) convolutional encoder (Fig. 1) is a finite state machine with k inputs, $X_1, X_2, \cdots, X_k$, and n outputs, $Y_1, Y_2, \cdots, Y_n$. Following each input node, there is a shift register. The encoder memory m is the maximum of those register lengths [2]. There are $N = 2^{km}$ states in the trellis diagram [2], [14] of an (n, k, m) code. The branch leading from state S_i to state S_j of the next stage is denoted as $e_{i,j}$ (Fig. 2).

The Viterbi decoding algorithm dynamically searches for the shortest path in the trellis diagram by survivor metric evaluation [15]. For

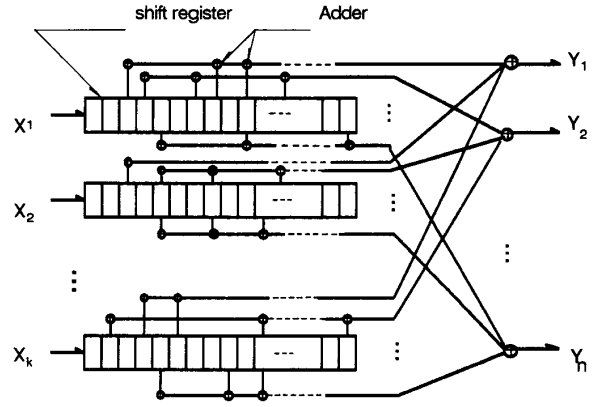

Fig. 1. Convolutional encoder.

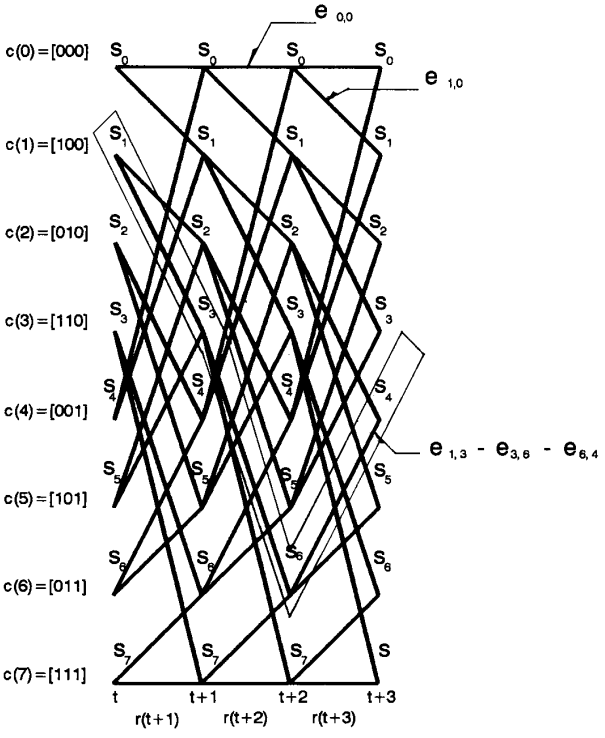

Fig. 2. Trellis diagram of an eight-state convolution code.

Paper approved by the Editor for Coding Theory and Applications of the IEEE Communications Society. Manuscript received July 22, 1987; revised January 13, 1989. This work was supported by the National Science Council under Grant NSC79-0404-E009-26.

K.-A. Wen is with Kwang-Ming New Village, Hsin Chu 30042, Taiwan, Republic of China.

T.-S. Wen was with Industrial Technique Research Institute, Hsin Chu, Taiwan, Republic of China. He is now deceased.

J.-F. Wang is with Research Institute of Electrical and Computer Engineering, National Cheng Kung University, Tainan, Taiwan, Republic of China.

IEEE Log Number 9035785.

a certain state S_j at stage t, there are 2^k candidate paths leading to it. Each path consists of the survivor path leading to S_i at the $(t-1)$th stage from the S_i to S_j at the tth stage by the branch $e_{i,j}$.

The computations required for each state are to find the survivor metrics and determine the survivor path among all the candidate paths to S_j. Let $J(j)$ denotes the set of indexes i of the 2^k states of previous stage S_i leading to S_j, each with a branch $e_{i,j}$ on the trellis. In Fig. 2, for state S_4 at $t+3$, $J(4)=\{2, 6\}$. Path metrics for all the candidate paths of S_j, denoted by $pm(i, j, t)$ for all $i \in J(j)$, will be evaluated via the likelihood measurement function f, and the maximum metric among them will be chosen as the survivor metric $sm(j, t)$. Let $r(t)$ be the n-bit word received at the tth stage and $w(i, j)$ the valid codeword of transition $e_{i,j}$. The $pm(i, j, t)$ and $sm(j, t)$ are related as

$$pm(i, j, t) = f\{r(t), w(i, j), sm(i, t-1)\} \tag{1}$$

$$sm(j, t) = X(pm(i, j, t)) \quad (= \max(pm(i, j, t);\ \forall i \in J(j))) \tag{2}$$

where X is the maximum comparator and is applied over all S_i with $i \in J(j)$. From (2), the operands relating to survivor metric evaluation for S_j are $r(t)$, $\{w(i, j) | i \in J(j)\}$ and $\{sm(i, t-1) | i \in J(j)\}$.

III. Trellis Diagram Integration

By topological modifications, p stages of the trellis diagram may be integrated with each p-branch path selected as a single branch in the reconfigured trellis diagram. Two variations of the trellis diagram for $(n, 1, 3)$ codes are illustrated in Fig. 3(a) and (b). In Fig. 3(a), every two stages are integrated and each branch is exactly a 2-branch path in the original trellis (Fig. 2). In Fig. 3(b), the merged 3-branch paths are illustrated. The advantages of mapping Viterbi decoding algorithm onto the integrated trellis diagrams are analyzed as follows.

Referring to Fig. 2, by assuming that $pm(1, 3, t+1)$ is determined as the survivor metric for S_3 at the $(t+1)$th stage and $pm(3, 6, t+2)$ as the survivor metric for S_6 at $(t+2)$th stage, (as circled in Fig. 2). Then the 3-branch path $p_{6,4,t+3}$ (as highlighted in Fig. 2) composed of $e_{1,3,}$, $e_{3,6}$, and $e_{6,4}$ is one of the candidate paths of S_4 at the $(t+3)$th stage. In the conventional VA, $pm(6, 4, t+3)$ is iteratively calculated stage by stage as

$$\begin{aligned} pm(6, 4, t+3) &= f(r(t+3), w(6, 4), sm(6, t+2)) \\ &= f(r(t+3), w(6, 4), X(pm(i, 6, t+2))) \\ &= f(r(t+3), w(6, 4), pm(3, 6, t+2)) \\ &= f(r(t+3), w(6, 4), f(r(t+2), w(3, 6), \\ &\quad sm(3, t+1))) \\ &= f(r(t+3), w(6, 4), f(r(t+2), w(3, 6), \\ &\quad X(pm(i, 3, t+1)))\cdots). \end{aligned}$$

In this stage-by-stage dynamic process, survivor metrics for all S_i with $i \in J(j)$ should be available to update the survivor metric of S_j. This requires excessive data access operations which are the most time consuming in VLSI circuits.

While with p stages merged into a single one, m successive branches on the trellis diagram will be taken into an integrated branch. If the sum of the branch metrics of these m successive branches can be calculated in a single computation procedure and added with the survivor metrics evaluated m stages before to form the new path metrics, then the survivor metric for each state will be updated every p stages and the number of data access operations can be reduced p times.

However, in order for the branch/path/survivor evaluations defined on the integrated trellis to be compatible with those on the original VA trellis, integration of trellis diagram has the constraint that a merged branch on the integrated trellis must be in one-to-one correspondence to a p-branch path in the original trellis.

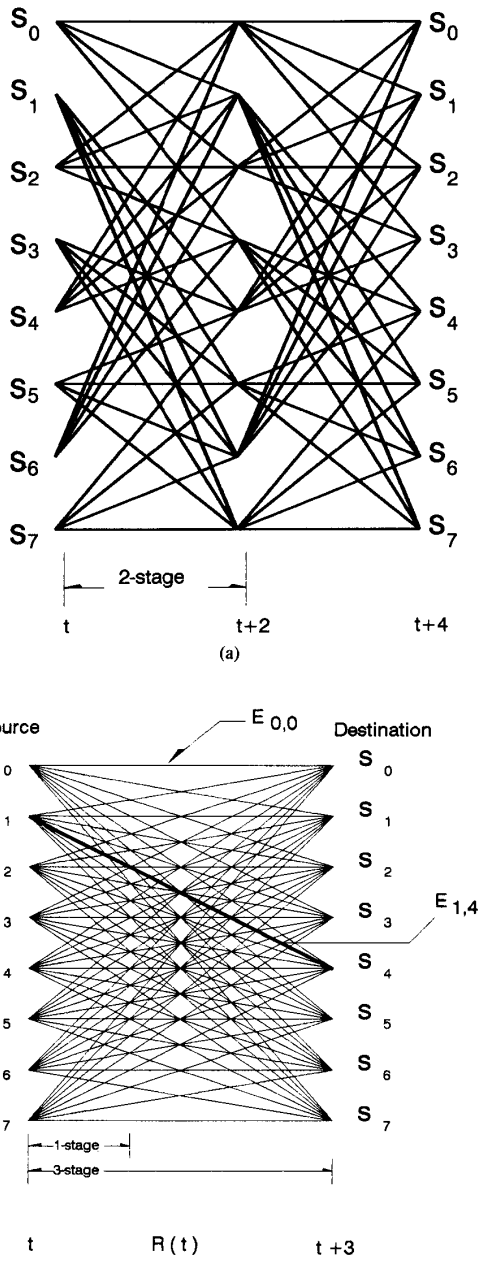

Fig. 3. (a) The integrated trellis diagram ($p = 2$). (b) The integrated trellis diagram ($p = 3$).

A brief proof given in Appendix A shows that for a p-branch path which is uniquely specified by its source and destination states, p should be no larger than m. For the extreme case $p = m$, the trellis diagram with m stages integrated is in the form of a bipartite complete $K_{N,N}$ graph [16]. The Viterbi transform is derived according to the modified $K_{N,N}$ trellis diagram. We use $E_{i,j}$ to denote an m-branch path linking S_i and S_j in the $K_{N,N}$ graph. The trellis diagram of an $(n, 1, 3)$ code with three stages integrated, as shown in Fig. 3(b), is a $K_{8,8}$ graph. The path marked as $E_{1,4}$ is exactly the 3-branch path highlighted in Fig. 2.

IV. From Viterbi Algorithm to Viterbi Transform

The dynamic decoding procedures of the VA will be modified for $E_{i,j}$, so that the data access/transfer for metric evaluations demanded by the component branches can be eliminated. For clarification, only $(n, 1, m)$ convolutional codes will be discussed for the derivation of the VT, and we start our discussion with the encoding process.

A. State Code Assignment

For the $(n, 1, m)$ codes, there is a single input, m registers and n outputs. Since the number of registers is m, there are $N = 2^m$ states for the $(n, 1, m)$ encoder corresponding to the 2^m possible bit combinations in the original m encoder registers. We represent each state S_j by an m-bit state code which is exactly the reversed binary representation of j, that is,

$$C(j) = \text{State code of } S_j = \text{reversed binary of } j.$$

B. Branch Code Generation

1) Formulation of Single-Stage Branch Code Generation: The prime task of an encoder is to generate the branch code $w(i, j)$ according to a possible state transitions from S_i to S_j. Based on the encoding process, the operand vector in (i, j) for generating the branch code is composed of the contents of the m registers $(C(i))$ and the current input bit which causes the state transition from S_i to S_j. Since the input bit is loaded into the encoder from the left side and the bit combination in the m registers is changed from $C(i)$ to $C(j)$, the left-most bit of $C(j)$ [i.e., LMB(j)] is exactly the input bit causing the state transition from S_i to S_j. Hence,

$$in(i, j) = [LMB(j)|C(i)] \tag{4}$$

where $[\,|\,]$ denotes the concatenation and $w(i, j)$ is generated as

$$w(i, j) = in(i, j)*G \tag{5}$$

where "*" is the matrix multiplication defined on GF(2). G is a $k(m+1) \times n$ matrix defining the code.

$$G = \begin{bmatrix} G_0 \\ G_i \\ \cdots \\ G_m \end{bmatrix} \quad \text{where } G_r = \begin{bmatrix} g_{1,r^1} & g_{1,r^2} & g_{1,r^3} & \cdots & g_{1,r^n} \\ g_{2,r^1} & g_{2,r^2} & g_{2,r^3} & \cdots & g_{2,r^n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{k,r^1} & g_{k,r^2} & g_{k,r^3} & \cdots & g_{k,r^n} \end{bmatrix}.$$

Element g_{i,r^j} is the rth bit of g_{i^j}, the generator sequence relating X_i and Y_j [2]. For example, if a (3, 1, 3) convolutional code is defined by

$$G = \begin{bmatrix} 110 \\ 011 \\ 101 \\ 001 \end{bmatrix}.$$

Then

$$w(1, 3) = [LMB(3)|C(1)]*G = [1|100]*G = [101] \tag{6}$$

$$w(3, 6) = [LMB(6)|C(3)]*G = [0|110]*G = [110] \tag{7}$$

$$w(6, 4) = [LMB(4)|C(6)]*G = [0|011]*G = [100]. \tag{8}$$

2) Formulation of m-Stage Branch Code Generation: For $E_{i,j}$, m input bits are entered during the combined m stages and the content in the m registers is changed to $C(j)$. In other words, $C(j)$ is exactly composed of the m input bits. We use $C(j)$ as the operand in the evaluation of the m-stage branch code $W(i, j)$ for $E_{i,j}$, that is,

$$IN(i, j) = [C(j)|C(i)] \tag{9}$$

$$W(i, j) = IN(i, j)*Gm \tag{10}$$

where Gm is a $(2m \times mn)$ extended matrix composed of m G's with each G being located one row lower than the previous one:

$$Gm = \begin{pmatrix} G_0 & & \\ G_1 & G_0 & \\ G_2 & G_1 & \\ \vdots & G_2 & \\ \vdots & \vdots & \\ G_{m-1} & \vdots & G_0 \\ G_m & G_{m-1} & G_1 \\ & G_m & G_2 \\ & & \vdots \\ & & \vdots \\ & & G_{m-1} \\ & & G_m \end{pmatrix}.$$

The extended m-stage branch code derived from (10) is composed of all the branch codes of its m component branches. The extended generator matrix for the previous example is

$$Gm = \begin{pmatrix} 110 & & \\ 011 & 110 & \\ 101 & 011 & 110 \\ 001 & 101 & 011 \\ & 001 & 101 \\ & & 001 \end{pmatrix}.$$

It can be seen that $W(1, 4) = [001|100]*Gm = [100110101]$. Comparing to (6), (7), and (8), $E_{1,4}$ is composed of $e_{1,3}$, $e_{3,6}$, and $e_{6,4}$; and $W(1, 4)$ is constructed with $w(1, 3)$, $w(3, 6)$, and $w(6, 4)$ which are concatenated in reverse. That is, $W(1, 4) = [w(6, 4)|w(3, 6)|w(1, 3)]$.

C. Computational Modification for m-Stage Branch Code Generation

For further reduction, Gm is decomposed into two $(m \times nm)$ matrixes: Gm_u and Gm_L which are the upper and lower halves of Gm, respectively. When calculating $W(i, j)$'s for all the inward branches of S_j, the partial product of $C(j)$ and Gm_u is fixed, being dependent on S_j only, and can therefore be isolated in (10) as $T(j)$, that is,

$$\begin{aligned} W(i, j) &= IN(i, j)*Gm \\ &= [C(j)|C(i)]* \begin{pmatrix} Gm_u \\ \hline Gm_L \end{pmatrix} \\ &= [C(j)*Gm_u] \oplus [C(i)*Gm_L] \\ &= T(j) \oplus [C(i)*Gm_L] \end{aligned} \tag{11}$$

where $\oplus$ is the exclusive-OR operator. Equation (11) may be further

integrated by adding an augmented dimension to the multiplicand vector:

$$W(i, j) = [C(i)|1] * \begin{pmatrix} Gm_L \\ \hline T(j) \end{pmatrix}. \tag{12}$$

D. The m-Stage Branch Metric Evaluation

In the case of a binary symmetric channel (BSC), $w(i, j)$ and $r(t)$ are n-tuple vectors and branch metric $bm(i, j, t)$ may be measured from the Hamming distance between $w(i, j)$ and $r(t)$. A maximum likelihood decoder will choose $w(i, j)$ as the codeword sent if $bm(i, j, t)$ is the minimum among the Hamming distances between $r(i)$ and all the codewords. Branch metric can be implemented by inner product operation "x" as

$$bm(i, j, t) = [in(i, j)|1] * \begin{pmatrix} G \\ \hline r(t) \end{pmatrix} \times I^T \tag{13}$$

where I is the n-tuple unit vector $I = [111 \cdots 1]$. Extending (13) for $E_{i,j}$, the received code for the successive m stages should be considered. Let $R(t) = [r(t)|r(t-1)| \cdots r(t-m+1)]$, and $Bm(i, j, t)$ the m-stage branch metric of $E_{i,j}$, then

$$\begin{aligned} Bm(i, j, t) &= (W(i, j) \oplus R(t)) \times I_m^T \\ &= \left([C(i)|1] * \begin{pmatrix} Gm_L \\ \hline T(j) \end{pmatrix} \oplus R(t) \right) \times I_m^T \\ &= [C(i)|11] * \begin{pmatrix} Gm_L \\ T(j) \\ \hline R(t) \end{pmatrix} \times I_m^T \end{aligned} \tag{14}$$

where I_m is the nm-tuple unit vector extended from I.

E. The m-Stage Path Metric Evaluation

After evaluation of branch metric, path metric must be calculated. The path metric $pm(i, j, t)$ is defined as the sum of $bm(i, j, t)$ and $sm(i, t-1)$. That is,

$$\begin{aligned} sm(j, t+1) &= X(pm(i, j, t+1)) \\ pm(i, j, t+1) &= f(r(t), w(i, j), sm(i, t)) \\ &= bm(i, j, t+1) + sm(i, t). \end{aligned}$$

With m-stages integrated, $J(j)$ for each state contains all the numbers from 0 to $N-1$. Path metric and survivor metric for the integrated trellis diagram are denoted as $Pm(i, j, t)$ and $Sm(j, t)$ which can be evaluated analogously to $pm(i, j, t)$ and $sm(j, t)$, with X operated from $i = 0$ to $N-1$

$$Sm(j, t) = X(Pm(i, j, t)) \tag{15}$$

$$\begin{aligned} Pm(i, j, t) &= Bm(i, j, t) + Sm(i, t-m) \\ &= [C(i)|11] * \begin{pmatrix} Gm_L \\ T(j) \\ \hline \mathrm{Re}(t) \end{pmatrix} \times I_m^T + Sm(i, t-m) \\ &= [C(i)|11] * \left(\begin{array}{c|c} Gm_L & 0 \\ T(j) & \\ \hline \mathrm{Re}(t) & 1 \end{array} \right) \times [I_m|Sm(i, t-m)]^T. \end{aligned} \tag{16}$$

We define a matrix operator "" to be: $A[B|C] = A*B \times C^T$, then

$$Pm(i, j, t) = [C(i)|11] \left(\begin{array}{cc|c} Gm_L & 0 & \\ T(j) & & \mathrm{Im} \\ \hline \mathrm{Re}(t) & 1 & Sm(i, t-m) \end{array} \right). \tag{17}$$

Thus, the codeword generating, branch metric and path metric calculations for each state in the integrated trellis can be done by the simple matrix operation given in (17).

F. Elimination on Backtracking

In standard Viterbi decoding, the final decoded symbol should be traced out with backtracking procedures. In evaluating $Sm(j, t)$, we record the code of the source state of the path processing the largest path metric, that is,

$$U(j, t) = C(q) \text{ iff } X(Pm(i, j, t)) = Pm(q, j, t) \tag{18}$$

where $U(j, t)$ is the maximum likelihood decoded code for $Pm(q, j, t)$. By defining the H operator as the comparator which yields the source state code of the path processing the maximum path metric, then

$$U(j, t) = H(Pm(i, j, t)). \tag{19}$$

For example, if $Pm(1, 4, 6)$ is chosen to be the survivor metric of S_4 at $t = 6$,

$$Pm(1, 4, 6) = X(Pm(i, 4, 6)) \tag{20}$$

then $C(1)$ (= [100]) is regarded as the maximum likelihood source code of the path leading to S_4 at $t = 6$ and will be selected by the H operator

$$U(4, 6) = H(Pm(i, 4, 6)) = C(1) = [100]. \tag{21}$$

Thus, if $Sm(j, t)$ is the maximum among the survivor metrics for all states, the most likely decoded code may be directly read out from $U(j, t)$. The usual backtracking processes in the VA is then eliminated.

G. Transformation Properties

We denote

$$\begin{pmatrix} Gm_L & 0 & \\ T(n) & 0 & I_m \\ R(t) & 1 & Sm(i, t-1) \end{pmatrix}$$

as $V_t^{i,n}$ which is in a form resembling the twiddle factor $W_N^{i,n}$ in FFT [21], that is,

$$\begin{aligned} V_t^{i,n} &= V_t^{i,(n+N)} = V_t^{i,n(i+N)} \\ W_N^{i,n} &= W_N^{i,(n+N)} = V_N^{i,n(i+N)}. \end{aligned}$$

The Viterbi decoding procedures for the successive m stages can be formulated as

$$R(t) \rightarrow \underbrace{\boxed{\begin{array}{l} H(Pm(i, j, t)) \\ = H([C(i)|11]V_t^{i,n}) \end{array}}}_{\text{VT processor}} \rightarrow U(j, t).$$

In fact, compatibilities between the FFT and the VA can be established [18] with respect to computational properties, periodic properties, signal flow graph, $\cdots$, etc. It suggests that existing FFT processors may be adapted for VT computations.

TABLE I
COMPUTATION RESULT OF VA

Index	i	j	t/	1	2	3	4	5	6	7	8	9	10
bm	0	0		2	2	1	1	2	0	2	1	2	2
	2	0				1	0	2	2	0	1	2	0
	0	1		2	0	3	1	0	2	2	3		
	2	1				1	3	2	2	2	1		
	1	2			2	1	3	2	2	2	1	0	
	3	2				3	1	0	2	2	3	2	
	1	3			2	1	1	2	2	2	1		
	3	3				1	1	2	0	2	1		
pm	0	0		2	4	5	6	5	5	7	6	8	10
	2	0				5	3	6	6	5	8	10	8
	0	1		2	2	7	6	3	7	7	8		
	2	1				5	6	6	6	7	8		
	1	2		2	4	3	8	8	5	9	8	8	
	3	2				6	4	4	8	7	10	10	
	1	3		2	4	3	6	8	5	9	8		
	3	3				5	4	6	6	7	8		
sm	0			2	4	5	3	5	5	5	6	8	8
	1			2	2	5	6	3	7	7	8		
	2				4	3	4	4	5	7	8	8	
	3				4	3	4	6	5	7	8		

U(j,t)	t/	1	2	3	5	8	10
0		0	00	000	01000	01001000	0100100100
1		1	01	101	..01001...	01001001.....	
2			10	010	11110	01001110	
3			11	111	11111	01001011	

Final survivor = [000 110 011 101 110 011 101 110 011 101]
Decode information sequence = [0 1 0 0 1 0 0 1 0 0]

TABLE II
COMPUTATION RESULT OF VT

Index	i	j	t/	2	4	6	8	10
pm	0	0		4	6	5	8	10
	1	0			3	8	10	8
	2	0			4	6	6	12
	3	0			5	8	8	10
	0	1		2	6	7	10	
	1	1			6	8	10	
	2	1			6	8	8	
	3	1			8	8	8	
	0	2		4	10	5	8	
	1	2			4	8	8	
	2	2		4	6	8	10	
	3	2			6	8	8	
					8	5	8	
	1	3			4	6	10	
	2	3			4	8	8	
	3	3			6	6	8	
Sm	0			4	3	5	6	8
	1			2	6	7	8	
	2			4	4	5	8	
	3			4	4	5	8	

U(j,t)	t/	2	4	8	10
0		00	0100	01001000	0100100100
1		01	0101	.. 01001001	
2		10	1110	01001110	
3		11	1111	01001011	

Final survivor= [000110 011101 110011 101110 011101]
Decoded information sequence = [01 00 10 01 00]

H. Example

Feasibility of VT will be demonstrated by a simple example for a $(3, 1, 2)$ code with

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

By assuming that a received sequence r = [011 110 001 100 110 000 101 001 011 101] is obtained at the decoder, $w(i, j)$ is computed as:

$$w(0, 0) = [0, 0, 0]*G1 = 000 \quad w(2, 0) = [0, 0, 1]*G1 = 101$$

$$w(0, 1) = [1, 0, 0]*G1 = 110 \quad w(2, 1) = [1, 0, 1]*G1 = 011$$

$$w(1, 2) = [0, 1, 0]*G1 = 011 \quad w(3, 2) = [0, 1, 1]*G1 = 110$$

$$w(1, 3) = [1, 1, 0]*G1 = 101 \quad w(3, 3) = [1, 1, 0]*G1 = 000.$$

Branch metrics, path metrics and survivor metrics will be evaluated stage by stage as listed in Table I.

Via VT procedures, $Gm = \begin{pmatrix} 110 \\ 011110 \\ 101011 \\ \hline 101 \end{pmatrix}$ and $T(j)$ is calculated as

$$T(0) = [00]*\begin{pmatrix} 110 \\ 011110 \end{pmatrix} = [000000];\ \text{likewise, } T(1) = [110000]$$

$$T(2) = [011110];\ T(3) = [101110].$$

$Sm(i, 0)$s for all $i = 0$ to $N - 1$ are initialized to be 0 and the path metric is calculated as

$$Pm(0, 1, 2) = [C(0)|11]*\left(\begin{array}{c|c} Gm_L & 0 \\ T(1) & \\ \hline R(2) & 1 \end{array}\right) \times [I_m|Sm(0, 0)]^T$$

$$= [C(0)|11]*\begin{pmatrix} 101011 & 0 \\ 000101 & 0 \\ 110000 & 0 \\ 110011 & 1 \end{pmatrix} \times [111111 \quad 0]^T$$

$$= 2.$$

Computations are listed in Table II. The decoded information sequence obtained via the VA and the VT procedures are both [0100100100].

V. Systolic Array Processor for the Viterbi Transform

By using the VT, a number of features can be obtained for the VLSI array design [19], [20].

Locality: According to the matrix operation (17), systolic array implementation for the VT may be realized with local data access. Thus the complicated wiring problem incurred by the long distance data transfer in the original trellis is eliminated.

Regularity and Asynchrony: In the VT, branch code generation is integrated with the decoding processes. Hence, there is no need for additional branch code generation circuitry or memories [8], [10]. Evaluation for branch/path/survivor metrics may also be rhythmically operated without extra time delays for branch code access [8], [10].

Modularity: Computations relating to $C(i)$ and $C(j)$, i.e., $C(i)*Gm_L$ and $C(j)*Gm_u$, can be isolated and be operated in parallel by modular processing blocks to reduce processing time performance.

A. The Systolic (2, 1, 4) Processor Design

Due to the triangular property of Gm_L and Gm_u, as shown in Fig. 4(a), Gm may be installed in a 2-D array processor as shown

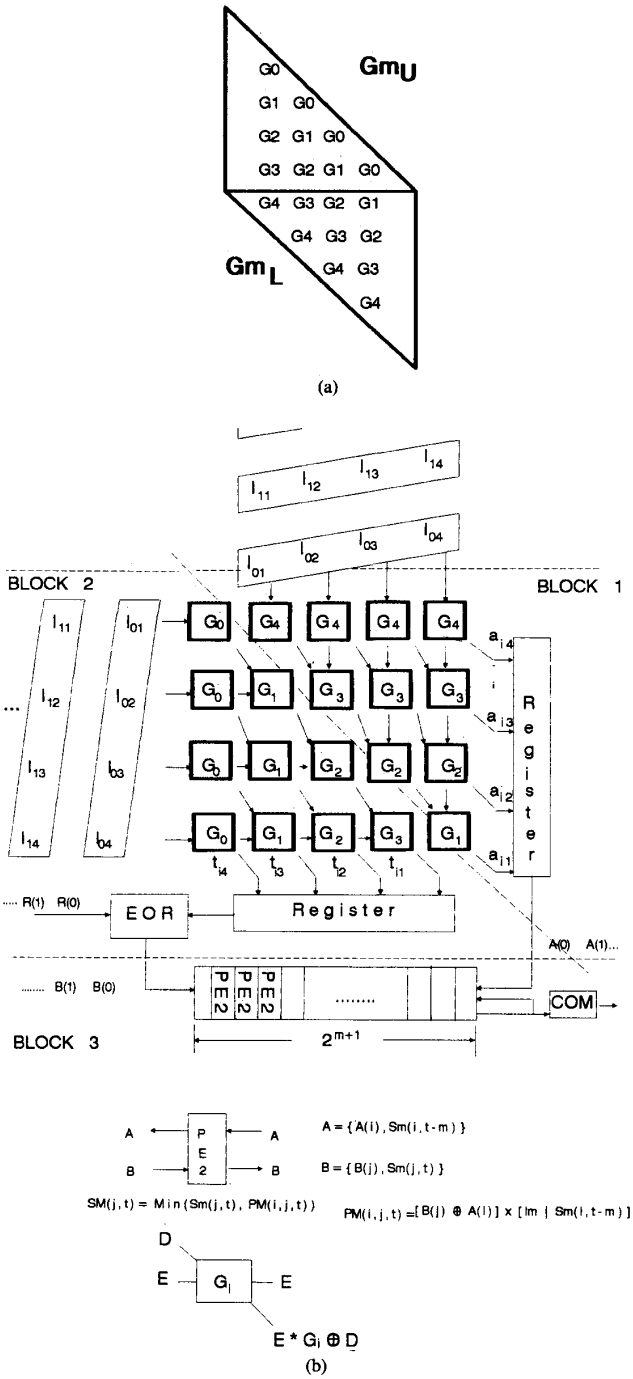

Fig. 4. (a) The generator matrix for (2, 1, 4) code. (b) The (2, 1, 4) VT processor.

in Fig. 4(b). Computations relating to $C(i)$ and $C(j)$, as listed in (22) and (23), are isolated and are concurrently calculated such that the processing time may be reduced:

$$C(i)*Gm_L = A(i) \tag{22}$$

$$C(j)*Gm_u \oplus R(t) = B(j). \tag{23}$$

Path metric computation can be implemented as

$$Pm(i, j, t) = [C(i)|11]* \begin{pmatrix} Gm_L & 0 \\ T(j) & 0 \\ R(t) & 1 \end{pmatrix} \times [I_m|Sm(i, t-m)]^T$$

$$= [A(i) \oplus B(j)|1] \times [I_m|Sm(i, t-m)]^T \tag{24}$$

where $A(i)$ and $B(j)$ are 8-bit vectors. They are independently generated from the upper right (Block 1) and lower left (Block 2) triangle array processors and are simultaneously sent to the linear array (Block 3) for path/survivor metric evaluations.

Hamming distance between $A(i)$ and $B(j)$ should be calculated and be added to $Sm(i, t-m)$. These operations are implemented in the linear array (Block 3).

- $A(i)$ will be transmitted together with $Sm(i, t-m)$ from the right of the linear array.
- Whenever $B(j)$ and $A(i)$ meet in the linear array, three operations will be sequentially operated:

a) $B(j) \oplus A(i)$.

b) $[B(j) \oplus A(i)] \times [I_m | Sm(i, t-m)] (= Pm(i, j, t))$.

c) $Sm(j, t) = \text{Min}\ (Sm(j, t), Pm(i, j, t))$ (where min is the minimum comparator).

$Sm(j, t)$ will be transmitted with $B(j)$ and iteratively updated through the linear array. As soon as the $Sm(j, t)$ emerges from the right, it will be recirculated into the linear array for the evaluation of $Sm(j, t+m)$. At the same time, all the $Sm(j, t)$ will be compared sequentially in the COM processor to obtain the maximum survivor metrics. Within 2^m execution cycles of $PE2$, m decoded symbols will be sent out from COM processor.

B. System Extension by Modularization

To reserve the architecture of the (2, 1, 4) VT processor, the generator matrix of a (2, 1, 8) code [Fig. 5(a)] is folded, as shown in Fig. 5(b). Module A and B can be installed into two (4×5) cell matrices, as shown in Fig. 5(c), with each cell storing $2n$-bit vector. The 8-bit vector generated from the counter is also folded and enters the cell matrix for branch code generation. Hence, based on the architecture of a (2, 1, 4) VT processor, the (2, 1, m) Viterbi decoders with $m = 4K$ can be similarly constructed with G being divided into K modules.

C. Performance Comparison

Computations required for the branches in a single stage using the VA processor and for the m-stage branches in an integrated stage using the VT processor will be analyzed as follows.

a) Computations required in a single stage with the VA processor:
Codeword generation: 2^{m+1} [$m+1$ by n] matrix multiplications.
Branch metric: 2^{m+1} n-tuple EOR operations.
Branch metric: 2^{m+1} n-tuple Hamming distance evaluations.
Path metric: 2^{m+1} additions.
Survivor metric: 2^m comparisons.

b) Computations required in an integrated m-stage with the VT processor:
Path metric: 2^{2m} [$2m$ by mn] matrix multiplications.
Survivor metric: 2^m comparisons.

Based on the dynamic programming property of the standard VA, the decoding procedures should be operated stage by stage. Therefore, to decode m symbols with the VA processors, m iterations should be operated and m times the computations as listed in a) are required. While, the VT has the advantage that parallel processing for m stages is realized through the concurrency algorithm and the computations listed in b) are exactly what are needed to decode m symbols with the VT processors.

As listed in a), there are four computation modules in the VA. In most Viterbi decoder designs, these four modules are always implemented with four building blocks: the codeword generation mechanism, the branch metric calculation mechanism, the path metric calculation mechanism, and the survivor metric decision mechanism. The major design complexity of Viterbi decoder is caused by the data interconnection within these building blocks. With the VT procedures derived here, although more path metrics are to be calculated than with the VA, the computation procedures are more concise and hence the building blocks and the routing schemes in most Viterbi decoders [8], [11], [18], [23]–[25] can be greatly simplified.

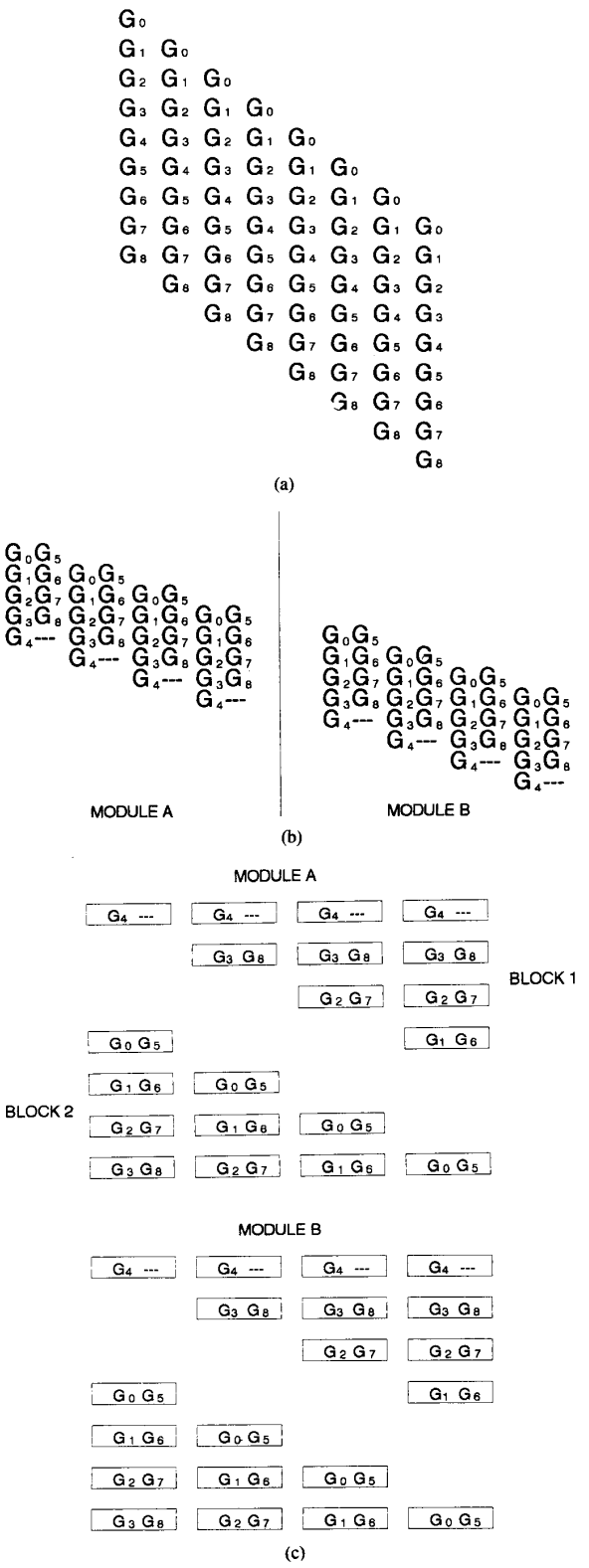

Fig. 5. (a) The generator matrix for (2, 1, 8) code. (b) The folded generator matrix for (2, 1, 8) code. (c) Data modularization for (2, 1, 8) VT processor.

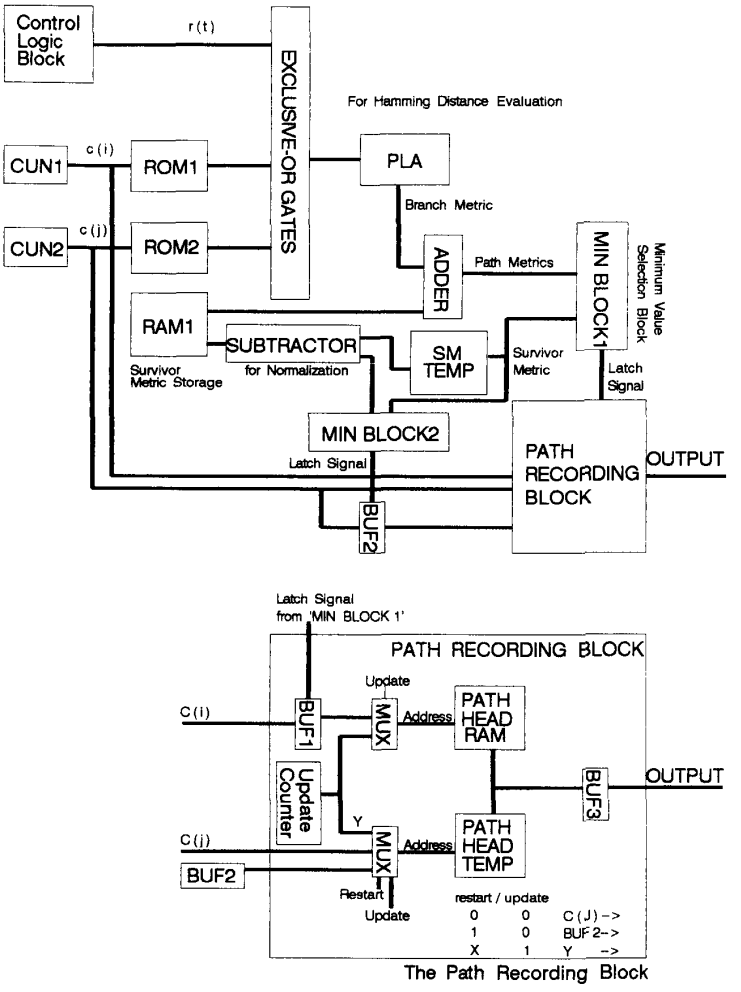

Fig. 6. Function block of the single chip design of VT processor.

For small m (≤ 4), the concise computation of the VT can be implemented with a single chip design of a VT processor as illustrated in Fig. 6, in which the routing scheme is rather simple. For larger m, the concise computation can be realized by the systolic $(2, 1, 4m)$ VT processor design as introduced in Section V. In the systolic $(2, 1, 4m)$ VT processor, data access per symbol is reduced and confined to a local access and branch codes are efficiently generated by a systolic array as shown in Fig. 4(b). Survivor metrics and survivor paths are all stored with local memory and are locally transferred. The processing time required to decode m symbols with the VT processor [Fig. 4(b)] is only the processing time of the pipeline composed of 2^{m+1} $PE2$'s as shown in BLOCK 3 [Fig. 4(b)]. That is, the sum of the execution cycle for 2^{m+1} Hamming distance evaluations for nm-tuple vectors, 2^{m+1} additions and 2^{m+1} comparisons.

VII. Conclusion

The Viterbi transformation can be regarded as a VLSI oriented algorithm because it is developed to take advantages of the high-density, low-cost, and high-speed VLSI's. The dynamic decoding procedures in successive m stages are paralleled. The integrated decoding operations are formulated into matrix operations. For longer constraint lengths, the Viterbi decoders can be derived from the combination of basic $m = 4$ VT processors such that a single standard VLSI design can be used.

Appendix A

A trellis diagram is derived from a tree graph. There are $N(= 2^m)$ m-level trees embedded in a trellis diagram. They are rooted from each of the N states and will get to all of the possible N states m stages later. Hence, there is exactly one path between any pair of nodes which are m stages apart.

Over p stages ($p < m$), there are either one or no paths between any arbitrary pair of nodes. Hence, each p-branch path may be uniquely specified by its source state and destination state.

Acknowledgment

In memory of Prof. Ting-Shiun Wen

This work was first brought to K.-A. Wen's attention by her father, Prof. Ting-Shiun Wen. Throughout the research Prof. T.-S. Wen contributed many valuable ideas and brought to the work his indepth knowledge in related fields. It is most unfortunate that the acceptance of this work was received just two weeks after Prof. T.-S. Wen passed away.

The authors would like to thank Prof. A. C.-J. Tsui for grammatical correction on the manuscript. The suggestion of the reviewers were extremely helpful during revision and the authors want to appreciate Dr. A. H. Levesque, Editor for coding theory and applications, for his thoughtful revision and connection on the manuscript.

References

[1] A. J. Viterbi and J. K. Omura, *Principle of Digital Communication and Coding*. New York: McGraw-Hill, 1979.

[2] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983, ch. 11.

[3] G. D. Forney, Jr., "Convolutional codes II: Maximum likelihood decoding," *Inf. Contr.*, vol. 25, pp. 222–266, July 1974.

[4] S. Haykin and J. P. Reilly, "Maximum-likelihood receiver for low-angle tracking radar, Part I: The symmetric case," *IEE Proc.*, vol. 129, Pt. F, pp. 261–272, 1982.

[5] J. P. Reilly and S. Haykin, "Maximum-likelihood receiver for low-angle tracking radar, Part 2: The nonsymmetric case," *IEE Proc.*, vol. 129, Pt. F, pp. 261–272, 1982.

[6] J. K. Skwirzynski, *The Impact of Processing Techniques on Communications*. The Netherlands: Martinus Nijhoff, 1985, pp. 159–192.

[7] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 835–848, Oct. 1971.

[8] P. G. Gulak and E. Shwedyk, "VLSI structures for Viterbi receivers: Part I—General theory and applications," *IEEE J. Select. Areas Commun.*, vol. SAC-4, Jan. 1986.

[9] S. Mohan and A. K. Sood, "A multiprocessor architecture for the (M, L)-algorithm suitable for VLSI implementation," *IEEE Trans. Commun.*, vol. COM-34, Dec. 1986.

[10] T. Ishitani, K. Tansho, N. Miyahara, S. Kubota, and S. Kato, "A scarce-state-transition Viterbi-decoder VLSI for bit error correction," *IEEE J. Solid-State Circuits*, vol. SC-22, Aug. 1987.

[11] C. Y. Chang and K. Yao, "Viterbi decoding by systolic array," in *23rd Ann. Allerton Conf. Commun., Contr., Comput.*, 1985, pp. 430–439.

[12] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. IT-9, pp. 64–73, Apr. 1963.

[13] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 23, pp. 675–685, Nov. 1969.

[14] R. E. Blahut, *Theory and Practice of Error Control Codes*. New York: Addison-Wesley, 1983, pp. 347–353.

[15] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Rockville, MD: Computer Science, 1978.

[16] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[17] F. Hemmati and D. J. Costello, Jr., "Truncation error probability in Viterbi decoding," *IEEE Trans. Commun.*, vol. COM-25, pp. 530–532, May 1977.

[18] K. A. Wen, J. F. Wang, and J. Y. Lee, "Design of VLSI implemented signal processing systems based on signal flow graph analyses," in *Proc. 1988 IEEE Int. Symp. Circuits Syst.*, Helsinki, June, 1988.

[19] H. T. Kung, "Let's design algorithms for VLSI systems," in *Proc. Conf. VLSI: Architect., Design, Fabrication*, Caltech., Jan. 1979, pp. 65–90.

[20] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988, p. 199.

[21] C. Mead and L. Conway, *Introduction to VLSI Systems*. New York: Addison-Wesley, 1980, ch. 8.

[22] L. R. Robiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975, p. 363.

[23] P. G. Gulak and T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm," *IEEE J. Select. Areas Commun.*, vol. 6, Apr. 1988.

[24] K. A. Wen and J. Y. Lee, "Parallel processing for Viterbi algorithm," *Electron. Lett.*, vol. 24, no. 7, p. 1098, Aug.

[25] J. F. Wang, K. A. Wen, J. Y. Lee, and C. S. Wu, "Embedding interconnections into processing elements for VLSI array design: The $(n, k, m)^m$ maximum likelihood decoder," *The Comput. J.*, to be published.

★

Kuei-Ann Wen was born in Keelung, Taiwan, Republic of China, in 1961. She received the B.E.E., M.E.E., and Ph.D. degrees from the Department of Electrical Engineering and Institute of Electrical and Computer Engineering at National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1983, 1985, and 1988, respectively.

She is presently a professor in the Department of Electrical Engineering, National Chiao Tung University, HsinChu, Taiwan, Republic of China, where she has joined the Center for Telecommunications Research. Her current teaching and research interests are in the areas of high-speed digital signal processing, parallel processing and VLSI circuit design and error-correcting coding.

★

Ting-Shiun Wen (M'75) was born in Kung-Su, China, on July 24, 1927. He was graduated from Chiao Tung University, Shang-Hai, China, in 1947. From 1962 to 1979, he was a professor at National Chiao Tung University where he had held various research and educational positions. His main research areas are telecommunication, radar, and microwave.

From 1979 to 1988, he was with the Industrial Technique Research Institute (ITRI) where he held the position as the vice president. He was also the member of the board of directors of the United Microelectronics Cooperation (UMC), Taiwan Semiconductor Manufacturing Company (TSMC), ···, etc. On Aug. 24, 1989, he died of liver cancer.

★

Jhing-Fa Wang (S'82–M'83–SM'88) was graduated from National Cheng Kung University, Tainan, Taiwan, Republic of China in 1974 and received the Ph.D. degree in electrical engineering and computer science from Stevens Institute of Technology, Hoboken, in 1983.

He is presently a professor in the Department of Electrical engineering at National Cheng Kung University, where he has participated in the development of CAD tool for VLSI. His current teaching/research interests include graph theory, coding theory, speech/pattern recognition and ASIC design.