The Radix-2^k Viterbi Decoding with Transpose Path Metric Processor

Wen-Ta Lee
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R. O. C.

Thou-Ho Chen
Department of Electronic Engineering
Nai-Tai College
Tainan, Taiwan, R. O. C.

Liang-Gee Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R. O. C.

Abstract

In this paper, we present a radix-2^k Viterbi decoding with Transpose Path Metric (TPM) processor. The TPM processor can provide a permutation function for state rearrangement with simple local interconnection. For interconnection realization, the routing complexity is less than that of the delay-commutator reported previously. In addition, a higher memory length Viterbi processor can be constructed with lower radix-2^k modules. With features of modulation and cell regularity, the radix-2^k Viterbi decoding with TPM processor is very suitable for VLSI implementation.

I. Introduction

In recent communication systems, error-correction decoder has been widely used to reduce the bit error rate of transmitted data. By using appropriate error-correcting codes, the required signal-to-noise ration to achieve a specified error probability can be significantly reduced. Among the error-correcting codes, convolutional coding has been widely used in communication system due to its high coding gain. It is typically applied in deep-space and satellite communications [1-3], and used in text/voice [4-5] recognitions. The most popular decoding scheme for convolutional codes is the Viterbi algorithm (VA) [6].

VA is a maximum-likelihood decoder (MLD) that receives information and gets any possible path on the trellis diagram, and than finds an optimal path (indicated by bold line in Fig.1.) with minimum path metric. To describe the Viterbi decoding, it is essential to use a trellis diagram that is a time expansion of state diagram. The nodes of the trellis are called states and their associated numerical values call path metric. Fig.1. shows the process of a four-state trellis. Conventionally, each node in the trellis diagram of VA is fabricated as a processing element with add-compare-select (ACS) function. The ACS elements are organized in pairs to iterate one stage of a two-state trellis.

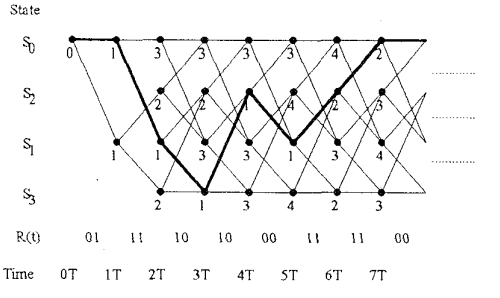

Fig.1. Trellis diagram of a four-state convolutional code

There are two approaches to realize VA, node parallel or node serial. In the node-parallel approach, computations for all nodes are performed simultaneously. The speed performance of such an architecture increases directly with the hardware cost. However, the interconnections will increase the routing complexity. In the node-serial approach, VA computations are carried out sequentially by one

0-7803-2440-4/94/ $4.00

processor. This architecture requires smaller area and less interconnects but suffers from slower decoded speed. A number of VLSI realizations have become available and their implementations have been reported [7-10].

We present a medium-speed VA decoding architecture that used radix-2^k modules with transpose path metric (TPM) processor that can provide a permutation function for state arrangement with simple local interconnection. And the routing complexity of interconnection can be reduced and embedded in the TPM processor. A single radix-2^k processor with TPM processor can be formed a single decoder, and a higher memory length decoder can be constructed with multiple such radix-2^k modules and TPM processors. These features make this architecture attractive for high-coding-gain, moderate-data-rate applications.

The next section describes the Viterbi algorithm with radix-2^k modules and higher radix formulations. Section III discusses the radix-2^k Viterbi processor with TPM processor, and derive a single radix-2^k computation module to form a higher memory length VA decoder. Finally, a conclusion is given in Section IV.

II. The Viterbi Decoding with Radix-2^k Computing Modules

The Viterbi algorithm is an optimum algorithm for estimating any possible path on the trellis diagram and then finds an optimal path. An R=1/2 convolutional encoder is based on a binary shift register. For each input bit, two output bits are generated as a modulo-2 combination of shift register contents and the input. Each node on the corresponding trellis diagram has exactly two inward and one outward branches respectively. The Viterbi algorithm can estimate the state sequence by an encoded sequence. Conventionally, each node in the trellis diagram of Viterbi algorithm is fabricated as a processing element. For an $(n,1,m)$ convolutional code, the computation complexity and storage requirement of Viterbi decoder are proportional to n^m, where n is the code rate $R=1/n$ and m is the memory length. The decoding speed decreases with increasing m.

Based on the trellis diagram, modulation of VA decoding can be easily processed as a radix-2 butterfly module where two ACS units are organized in pair to calculate the path metrics of each node. Due to the regular geometry, the trellis can be divided into a set of radix-2^k butterfly module as below:

A. Radix-2 Butterfly Module

Since, single input (R=1/2) encoder is widely adopted, our description of the radix-2^k decoding algorithm will concentrate on $(2,1,m)$ design. For a $(2,1,m)$ trellis diagram, the node is indicated as the set of states S_i, i=0 to N-1 ($N=2^m$). With state permutation of trellis diagram, any stage can be explored based on a radix-2 butterfly module as shown in Fig.2. The four states S_i, $S_{i+N/2}$ at one stage and S_{2i}, S_{2i+1} at next stage can form the radix-2 butterfly module of VA, where the subscripts i, $i+N/2$, $2i$, $2i+1$ are the numbers in modulo-2^m system. For a 2^m-state trellis, it can be transformed into a bipartite graph with 2^{m-1} independent radix-2 butterflies.

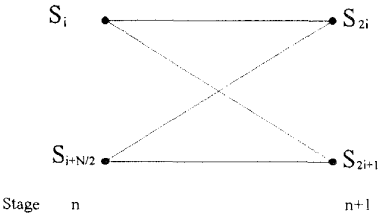

Fig.2. A radix-2 butterfly module

The radix-2 butterfly is a pair of two ACS operations. The ACS operation maintains a set of accumulated weights called path metrics. Each state has a path metric and two input paths. The accumulated weight is calculated by adding the weight associated with the state transition and the path metric of the preceding state. Then the two sums are compared, the smaller of the two is stored as the new path metric of the state, and represents the most likely path into the state.

B. Radix-4 Butterfly Module

Consider a trellis diagram for a (2,1,3) convolutional code as shown in Fig.3(a). Fig.3(b). demonstrate two stages of trellis merged through appropriate state rearrangement. A radix-2^2 butterfly can be derived in Fig.3(c). The four states of *nth* stage, S_i, $S_{i+N/4}$, $S_{i+2N/4}$,$S_{i+3N/4}$, and the four states of $(n+2)th$ stage, S_{4i}, S_{4i+1}, S_{4i+2}, S_{4i+3}, can form the radix-2^2 butterfly module of VA as shown in Fig.4. The ACS operation of radix-4 butterfly module can calculate the four paths entering the state and select the smallest path metric of the four as the survivor path.

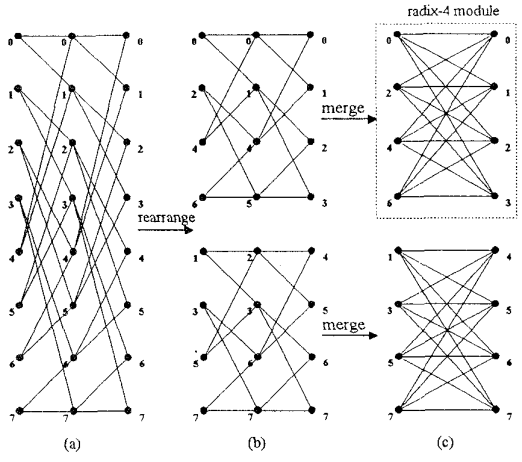

Fig.3. (a). A trellts diagram for a (2,1,3) convolutional code
(b). A 8-state trellis diagram after appropriate rearrangement
(c). Two radix-4 butterfly modules for a (2,1,3) trellis diagram

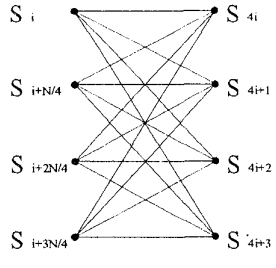

Fig.4 A radix-4 butterfly computing module

C. Radix-2^k Module

To merge three, four or more stages of trellis, we can obtain the radix-2^3, radix-2^4, ..., radix-2^k butterfly module. In case that m stages are merged, each state will be leaded to all the possible 2^k states through branches and a bipartite complete graph can be derived. In Fig.5 the trellis diagram of radix-2^k and its state permutation is listed.

III. Radix-2^k Viterbi Decoding With Transpose Path Metric Processor

A. Radix-2^k Pipelined Viterbi Decoding

With appropriate transformation of trellis diagram, Viterbi algorithm is compatible with Fast Fourier Transform (FFT). With the developed radix-4 pipelined FFT algorithm, the radix-4, radix-8 and higher radix Viterbi decoding has been proposed by [11-13].

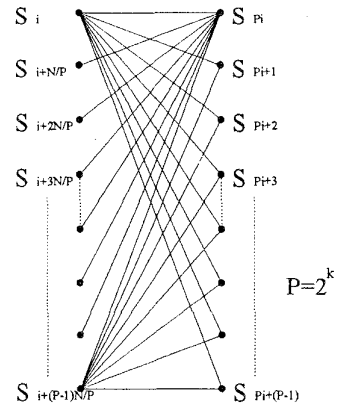

Fig.5. A radix-2^K butterfly computing module

For brevity, a radix-4 module to compute the ACS operation is named as a Vdx-4 processor. For a trellis diagram of (2,1,4) convolutional code using a Vdx-4 computing processor as shown in Fig.6, there are 2^4 states to enter the Vdx-4 processor in such a way that four states at each time are transferred and thus four bipartite complete graphs of radix-4 are formed. The index i of state S_i indicted the location where the Viterbi decoding procedure of the corresponding state are performed. After performing four radix-4 VA operations, the 2^4 ACS computations of one stage are finished and thus the permutation of state is changed. To continue next radix-4 ACS operations using the same Vdx-4 processor, a proper rearrangement of state will be required.

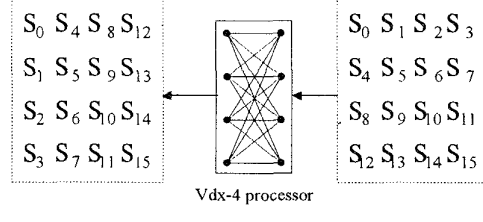

Fig.6. 2^4-state VA decoding by Vdx-4 processor

There is a way using delay commutator (DC) as switching processor [14] to rearrange the states. Fig.7. shows the data flow of a (2,1,4) code by using Vdx-4 and delay-commutator-delay processor. With the delay-commutator-delay operations, the data indicated by labeled number for Vdx-4 is successively transferred to the next iteration, However, the delay commutation approach will require complex

interconnections. To implement the commutator operation, registers and multiplexers are required. To reduce the interconnection complexity, a VA decoding based on a radix-2^k module computing with transpose path metric processor is proposed. It is easy to iterate the radix-2^k operation by TPM processor that is very modular and regular to be suitable for VLSI implementation. It is also easy to construct a higher memory length Viterbi decoding with lower radix-2^k processor.

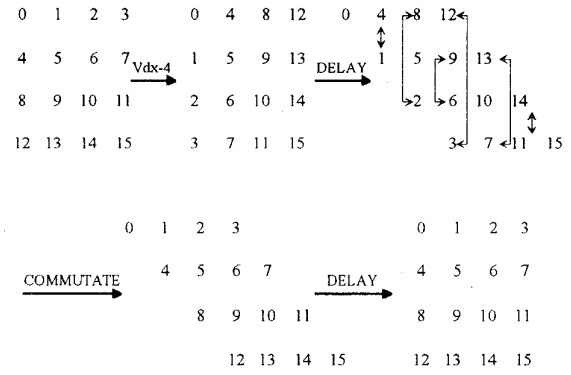

Fig.7. The (2,1,4) code dataflow using Vdx-4 & Delay-Commutator

B. Radix-4 Butterfly with TPM Processor

The permutation of state will change after each Vdx-4 computation. Thus states for input for next stage of VA operations should be rearranged. The proposed TPM processor can provide a permutation function with simple local interconnection. Consider a radix-2^k VA process with a $p*p$ TPM processor, where p is 2^k (k is an integer). The $p*p$ TPM processor is composed of $p*p$ memory cells that are used to store the path metric of each state. A radix-2^k VA computation module with TPM element is named as VAT-2^k processor state. For a $(2,1,m)$ convolutional code, the path metrics of all 2^m states are saved in a $p*p$ TPM processor (where $2^m=p^2$). Each memory cell can input data from right or down cells and output data to left or up cells as shown in Fig.8. The Vdx-2^k decoding with TPM processor for a $(2,1,m)$ convolutional code can be described as below:

Step 1: At beginning of each stage, path metrics of 2^m states are stored in the $p*p$ TPM processor with initial value. For a radix-2^k module, there are 2^k path metrics indicated with S_i, $S_{i+N/p}$, $S_{i+2N/p}$,…, $S_{i+(p-1)N/p}$, are saved in each i column, where $N=2^m$, $p=2^k$, and $i=0$ to $2^{m-k}-1$.

Step 2: For a Vdx-2^k computation, cells of the most left column will provide p path metrics of state for the Vdx-2^k processor by left-shift. Simultaneously, the other data of each column will shift to left column.

Step 3: After a computation of Vdx-2^k processor, the new path metrics generated from Vdx-2^k processor for next stage are transferred to the most right column of TPM processor.
If one stage of trellis is finished (after 2^{m-k} Vdx computations), go to step 4.
Else go to step 2.

Step 4: Now a new stage is beginning again, cells in the most up row will provide p path metrics for the Vdx-2^m processor instead of the column. Simultaneously, the other data of each row will shift data to up row.

Step 5: After a computation of Vdx-2^k processor, the new path metric generated from Vdx-2^k processor are transferred to the most down row of TPM processor.
If one stage of trellis is finished, go to step 2.
Else go to step 4.

There is an example for a (2,1,4) code using Vdx-4 processor with 4*4 TPM element to perform the VA decoding as shown in Fig.8.

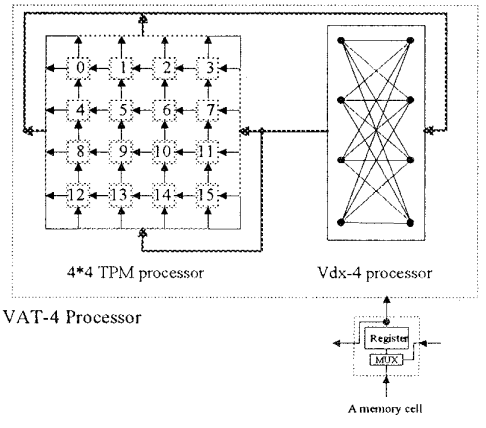

Fig.8. 2^4-state VA decoding by Vdx-4 processor with TPM processor

For brevity, the path metrics of 16 states stored in 4*4 TPM processor is indicated by the labeled number. At the beginning of *nth* stage, the initial permutation of

state is as shown in Fig.9(a), while the Vdx-4 VA computation is done, the newer path metrics generated by Vdx-4 processor are shift to the most right column of TPM processor, and each column of data is to shift left. After four Vdx-4 operations are performed, all states ACS computations of *nth* stage are finished, and the labeled number in memory cells is shown in Fig.9(b). At $(n+1)th$ stage the Vdx-4 VA processor goes as usual except the data of memory cells are shift up not left. Again at the end of stage, the labeled number in memory cells has the same state permutation as nth stage as shown in fig.9(c) and the Vdx operation can perform recursively using the same Vdx-4 processor.

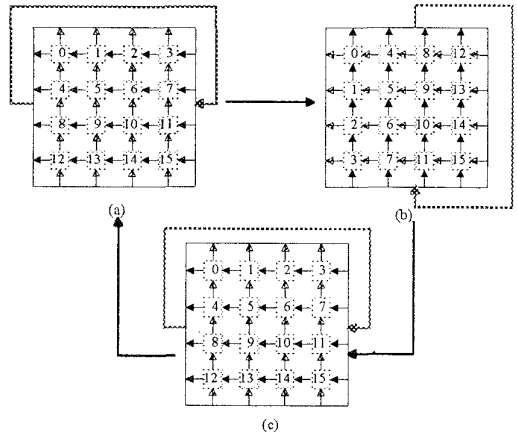

Fig.9. (a). State permutation in the 4*4 TPM processor before Vdx-4
(b). State permutation after Vdx operation by left shift
(c). State permutation after Vdx operation by up shift

With TPM processor, the radix-2^k VA decoding can perform the VA decoding without any other permuting states, like delay commutator, memory in-place updating or any other interconnecting processors [14-15].

C. Extend Higher Memory Length by VAT-2^k Processor

Through a slight modification of two radix-2^k processors with TPM processor, we can construct a high order memory length Vdx processor. For example, we can construct a (2,1,5) VAT-4 processor with two (2,1,4) VAT-4 processor as shown in Fig.10. While *nth* stage , the data of state saved in A VAT-4 processor can be labeled by number i, $i+N/4$, $i+2N/4$, $i+3N/4$ each column and in B VAT-4 is labeled as $(i+1)$, $(i+1)+N/4$, $(i+1)+2N/4$, $(i+1)+3N/4$ for i=0 to 3, $N=2^5$. Each column enter the radix-4 computation processor, the output data of $4i$, $4i+2$, $4(i+1)$, $4(i+1)+2$ (even states) are all gathered to be saved in a A-TPM processor and $4i+1$, $4i+3$, $4(i+1)+1$, $4(i+1)+3$ (odd states), in B-TPM processor.

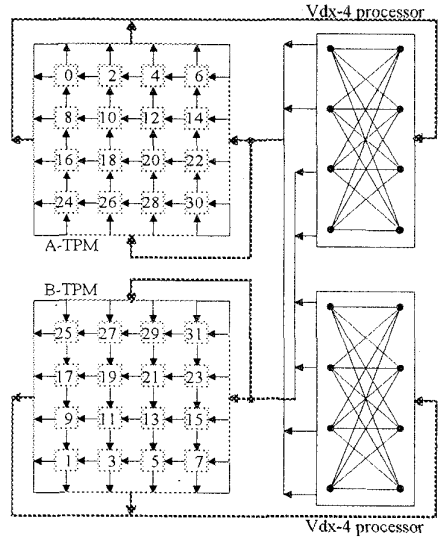

Fig.10. Construct a (2,1,5) VAT-4 processor by two (2,1,4) VA

At the end of one stage VAT computation by shifting data left, the permutation of state is shown in fig.11(b). Again with the same VAT procedure but by up shift, the state permutation of $(n+2)th$ stage will be the same permutation as *nth* stage shown in Fig.11(c). The VAT operations then go recursively. By the same way, a higher memory length m=6 VAT decoder can be constructed with four VAT-4 processors. However, a (2,1,6) VAT-4 decoder can also be constructed by a radix-2^3 with 8*8 TPM processor. The same technique can be easily used recursively within each high memory length VAT decoder by combining VAT-2^k processors. And by choosing appropriate radix and combining several VAT processors, the decoding speed and hardware cost can be adjusted. It is demonstrated in this section as shown in Fig.8. and Fig.10, combination of VAT-2^k modules may be easily realize with no complex routing problem. The state rearrangements and data routings are embedded in the TPM processor with simple local interconnections without any other internal switching processor. This architecture also provides a flexible way to develop a radix-2^k VAT decoder and the properties of VAT-2^k are suitable for a VLSI implementation.

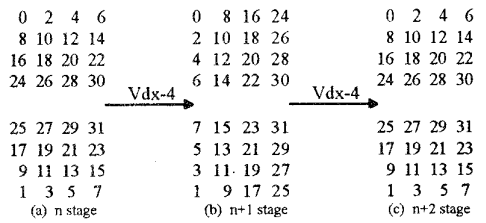

Fig.11. Data flow of Fig.10.

VI. Conclusion

The states for input to the next stage of VA processing must be arranged due to the fact that the state order always changes after each Vdx-2^k computation. The transpose path metric processor described in this paper can rearrange the state permutation for a radix-2^k Viterbi decoding. With simple local interconnection, the routing complexity will be reduced significantly when realizing. Besides, a higher memory length Viterbi processor can be constructed with lower VAT-2^k modules. With the regular and cellular structure, TPM processor is very suitable for VLSI implementation and thus provides a flexible way to develop a radix-2^k VAT decoder.

References

[1] J. A. Heller, and I. M. Jacobs," Viterbi decoding for satellite and space communication," *IEEE Trans. on Commun.*, vol. COM-19, no. 5, pp. 835-848, Oct. 1971.

[2] P. J. McLANE," The Viterbi receiver for correlative encoder MSK signals," *IEEE Trans. on Commun.*, vol. COM-31, no. 2, pp. 290-295, Feb. 1983.

[3] Normand J. P. Frenette, Peter J. Mclane, Lloyd E. Peppard, and Francis Cotter," Implementation of a Viterbi processor for a digital communications system with a time-dispersive channel," *IEEE J. Selected Areas Commun.*, vol. SAC-4,no. 1,pp. 160-167, Jan. 1986.

[4] L. R. Bahl, F. Jelink, and R. L. Mercer, " A maximum likelihood approach to continuous speech recognition," *IEEE Trans. on Pattern Analysis And Machine Intelligence*, vol. PAMI-5, no. pp. 179-190, Mar. 1983.

[5] A. Stölzle, S. Narayanaswamy, H. Murveit, J. M. Rabaey, and R. W. Brodersen," Integrated circuits for a real-time large-vocabulary continuous speech recognition system," *IEEE J. Solid-State Circuits*, vol. SC-26, no.1, pp. 2-11,Jan. 1991.

[6] G.D. Forney, Jr., " The Viterbi Algorithm," in *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[7] 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, pp. 575-582, Aug. 1987.

[8] P. J. Black and T. H. Meng," A 140Mb/s, 32-state, Radix-4 Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. SC-27, no. 12, pp. 1877-1885, Dec. 1992.

[9] M. A. Bree, D. E. Dodds, R. J. Bolton, S. Kumar and B. L. F. Daku," A modular bit-serial architecture for large-constraint-length Viterbi decoding," *IEEE J. Solid-State Circuits*, vol. SC-27, no. 2, pp. 184-189, Feb. 1992.

[10] W. T. Lee, L. G. Chen and M. C. Lin," The implementation of a single-chip Viterbi decoder with adaptive algorithm," in *Proc. of fourth VLSI DESIGN/CAD workshop*, pp. 120-124, Aug. 1993.

[11] H. Burkhardt and L. C. Barbosa," Contributions to the application of the Viterbi algorithm," *IEEE Trans. on Inform. Theory*, vol. IT-31, no. 5, pp. 626-634, Sept. 1985.

[12] K. A. Wen, J. F. Wang, J. Y. Lee and M. Y. Lin," Implementation of the array structured maximum likelihood decoder," in *Proc. of ICSA*, U.S.A.,May 1988.

[13] W. T. Lee, J. F. Wang, J. Y. Lee and K. A. Wen," VLSI architectures of pipelined Viterbi decoder," Master thesis, Department of Electrical Engineering, National Cheng Kung University, R.O.C., 1989.

[14] E. E. Swartzlander, Jr., K. W. Young and S. J. Joseph," A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 702-709, Oct. 1984.

[15] C. M. Rader," Memory management in a Viterbi decoder," *IEEE Trans. on Commun.* vol. COM-29, no. 9, pp. 1399-1401, Sep. 1981.