A 30-MHz Trellis Codec Chip for Partial-Response Channels

C. Bernard Shung, Paul H. Siegel, *Senior Member, IEEE*,
Hemant K. Thapar, *Member, IEEE*, and Razmik Karabed

Abstract—**We present a rate 8/10 matched-spectral-null (MSN) trellis codec chip which can increase noise tolerance in partial-response channels applicable to digital magnetic recording. The Viterbi detector in this codec features an area-efficient pipelined architecture and a modulo metric normalization technique. The chip was implemented in a 1.2-μm CMOS process with a die size of 22 mm^2. It offers a 12-Mb/s data rate when operating at 30 MHz. Experimental results verified the predicted coding gain of 2.8 dB relative to the uncoded system at a bit-error rate of 10^{-7}.**

Index Terms—**Codec chip, trellis code, partial response.**

I. Introduction

PARTIAL-response signaling has been widely used in data transmission since its introduction in the 1960's. A partial-response channel with a transfer polynomial $(1 \pm D^N)$ transforms the input sequence $\{x_i\}$ into the sequence $\{y_i\}$ where $y_i = x_i \pm x_{i-N}$. The *dicode* channel, with transfer polynomial $(1 - D)$, and the *class-IV* channel, with transfer polynomial $(1 - D^2)$, were shown by Kobayashi and Tang [1] to be attractive models for magnetic recording channels. Kobayashi [2] also gave an algorithm for maximum-likelihood detection for these channels, which is essentially equivalent to the trellis-based Viterbi algorithm [3].

To improve the reliability of such partial-response maximum-likelihood (PRML) channels, a new class of codes —matched spectral null (MSN) codes—has recently been proposed [4]. MSN codes are constructed to have a power spectrum that has nulls at precisely those frequencies where the channel transfer function has a null. The transfer function of the dicode channel has a first-order null at zero frequency (dc), meaning that the transfer function and its first derivative vanish at this frequency. The class-IV partial response channel has first-order nulls at zero frequency (dc) and at the Nyquist frequency. By matching the channel null frequencies, the MSN codes exploit the channel memory, enhancing minimum distance properties of received symbol sequences, while reducing the complexity of the Viterbi detectors [4]. For a dicode channel, the theory of MSN codes predicts that a MSN-coded system in which the MSN code has a first-order null at dc should tolerate close to 3 dB more noise power than the uncoded PRML channel.

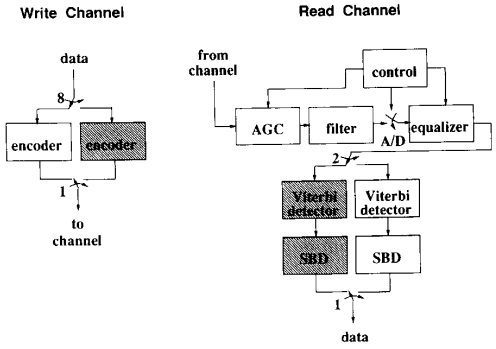


Fig. 1. System block diagram of a class-IV partial-response channel.

In this paper we present the first chip implementation of a rate 8/10 MSN trellis codec for the dicode partial-response channel. Since the class-IV channel is equivalent to an interleaved dicode channel, the benefits of the code can be extended to the former channel by interleaving. In Fig. 1 we show the system block diagram of the rate 8/10 trellis-coded PRML channel. The chip reported in the paper contains the shaded boxes in Fig. 1, including the encoder, a Viterbi detector, and a sliding block decoder.

The remainder of this paper is organized as follows. In Section II, we give a brief overview of the trellis code and the chip block diagram. In Section III, two architectural features used in the chip—area-efficient pipelining and modulo normalization—are discussed. In Section IV we review the design tools that were used in the physical design of the chip. The experimental setup and performance evaluation results are described in Section V. We conclude the paper in Section VI.

II. Trellis Codec Chip Overview

The chip block diagram is shown in Fig. 2. The encoder is a four-state finite-state machine that encodes an 8-b data input into a 10-b code word. The decoder consists of two parts: a Viterbi detector that produces from the noisy

Manuscript received May 2, 1991; revised August 19, 1991.

C. B. Shung is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

P. H. Siegel and R. Karabed are with IBM Research Division, San Jose, CA 95120.

H. K. Thapar is with IBM Corporation, San Jose, CA 95193.

IEEE Log Number 9103594.

0018-9200/91/1200-1981$01.00 ©1991 IEEE

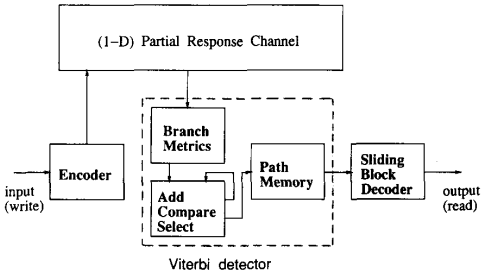


Fig. 2. Block diagram of trellis codec chip.

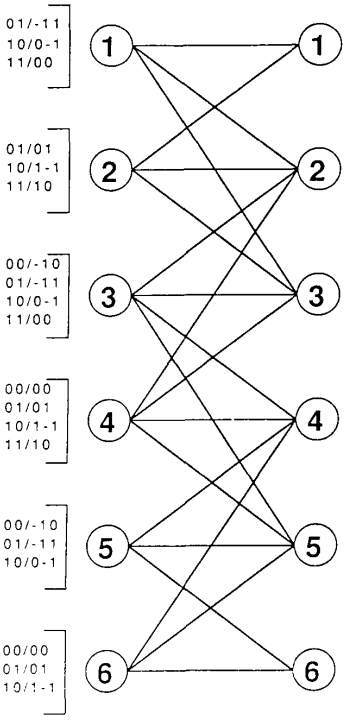


Fig. 3. Trellis diagram of the rate 8/10 MSN trellis code.

channel output sequence a near maximum-likelihood estimate of the correct code sequence, and a sliding-block decoder that uses two adjacent 10-b code-word estimates supplied by the Viterbi detector to determine the 8-b decoded data.

Fig. 3 shows the Viterbi-detector trellis structure for the rate 8/10 MSN trellis code. It represents the possible sequences at the output of the trellis-coded dicode channel. The edges are labeled by u_1u_2/v_1v_2 where u_1u_2 are code symbols and v_1v_2 are channel output symbols. For each of the six trellis states, the Viterbi detector recursively generates a code sequence, called the survivor. Among all sequences generated by a trellis path ending in a specified state, the output sample sequence corresponding to the survivor provides the best fit (in the sense of squared error) to the noisy received output sequence.

The Viterbi detector contains a branch metric calculator, two pipelined add–compare–select (ACS) processors, and a path memory. The branch metric calculator calculates the branch metric, which is the squared-Euclidean distance between the channel output symbols v_1v_2 and the received signals s_1s_2 (in 6-b soft decision format); that is,

$$bm_{v_1v_2} = (v_1 - s_1)^2 + (v_2 - s_2)^2$$

is the branch metric of the branch labeled v_1v_2. Note that since s_1^2 and s_2^2 are common to all branch metrics, they can be ignored in the subsequent minimization. The branch metrics then become

$$bm_{v_1v_2} = v_1^2 + v_2^2 - 2v_1s_1 - 2v_2s_2.$$

Since $v_1, v_2 \in \{0, 1, -1\}$, only adders and single-bit shifters are required in the branch metric calculation.

The path metric for each state is the accumulation of branch metrics of the survivor sequence ending in that state. Path metric update is the key operation to determine the extension of the survivor sequence. It involves adding the old path metric with the corresponding branch metric of all incident branches, comparing their sums, and selecting the minimum (hence the name add–compare–select, or ACS, for the unit that performs this operation). For example, for state 1 in Fig. 3,

$$new\ pm_1 = \min(pm_1 + bm_{-1\,1}, pm_2 + bm_{0\,1})$$

where pm_i is the path metric of state i. The ACS design will be discussed in more detail in the next section.

The path memory is implemented using the register-exchange scheme. The survivor sequences are stored in six shift registers each with a length of 32 stages (64 b). The shift register i always holds the survivor sequence for state i. This is accomplished by updating the entire contents of every shift register during every decoding cycle (which corresponds to one trellis stage), using multiplexers which are controlled by the comparison operation in the corresponding ACS.

An alternative architecture for path memory design is the trace-back scheme [5]. It has the advantage that regular RAM's can be used instead of shift registers and multiplexers, at the expense of a larger number of stages being stored. The trace-back approach is attractive when the number of states is large and the trellis is regular (e.g., convolutional codes). In our case we have only six states and a relatively irregular trellis. Therefore, we chose to use the register-exchange scheme.

III. ACS Architecture

A. Area-Efficient Pipelining

Most hardware realizations of the Viterbi detector in the literature [6], [7] achieve high data rate with a state-parallel approach, i.e., devoting one ACS to each state. The six-state detector trellis for the rate 8/10 MSN code,

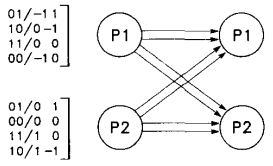


Fig. 4. A two-state trellis that emulates the six-state trellis in Fig. 3.

shown in Fig. 3, requires four 4-input ACS's (for states 2, 3, 4, and 5) and two 2-input ACS's (for states 1 and 6). The throughput rate can be increased by means of pipelined processing of independent interleaved data sources, but this approach is applicable only in the context of memoryless channels.

For partial-response channels, with memory in the form of intersymbol interference, we have developed a new technique that achieves the dual objectives of high throughput and area efficiency. We let several states share the same ACS, thus reducing the required silicon area. The partitioning and scheduling of the states into ACS's are carefully arranged to create an internal structure that can be exploited by pipelining to reduce the throughput penalty. The area–time trade-off performed by the new area-efficient architecture is, in some sense, the converse of that achieved by the proposed high-speed architecture [8], which uses a block processing technique to achieve higher throughput rate at the expense of increased area. For several applications, it has been shown that the area-efficient architecture achieves a smaller overall area–time product than the high-speed architecture [9].

Our area-efficient technique is based on graph emulation, which has also been found to be useful in other coding applications [9]. Roughly speaking, the partition and scheduling of the states are guided through a folding of the original trellis into a smaller trellis. For example, the six-state trellis in Fig. 3 can be folded into the two-state trellis shown in Fig. 4, with states 1, 3, and 5 combined into the single node represented by processor $P1$, and states 2, 4, and 6 combined into the single node represented by processor $P2$. Note that there are two branches connecting any two states in the smaller trellis. Hence, $P1$ and $P2$ are implemented by two 4-input ACS's.

The key idea is that the states sharing the same ACS can be processed in a pipelined fashion, even for channels that have memory. We refer to this structure as internal parallelism to differentiate it from external parallelism, where independent data sources are interleaved at the input to a single (memoryless) channel. To a large extent, the throughput penalty from using fewer ACS's is overcome by pipelining the ACS's. Specifically, for a four-input ACS, we put a pipeline latch after the adder, the first compare select, and the second compare select, as shown in Fig. 5. Such an arrangement is natural because the adder delay is roughly equal to the compare–select delay. As soon as the first state finishes adding the old path

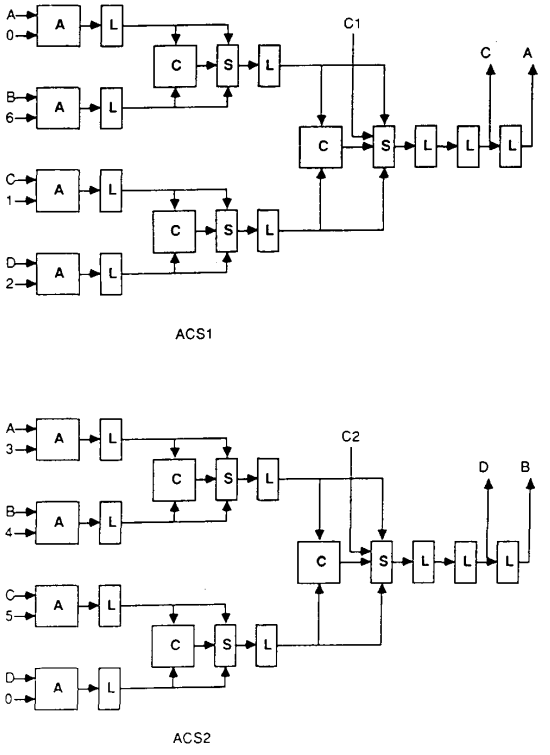


Fig. 5. Circuit schematics of the area-efficient pipelined ACS's. Legend: L = pipeline latch; A = adder; C = comparators; S = selector.

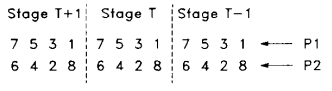


Fig. 6. Time schedule of the states sharing the ACS's.

metrics and branch metrics and proceeds to the compare select, the second state can be activated. Therefore, this pipelining scheme can increase the throughput rate by a factor of 3. (This assumes that the delay associated with the pipeline latches is negligible and that the pipeline delay is evenly partitioned among the pipeline stages.)

The scheduling of the states in $P1$ and $P2$ is shown in Fig. 6. Two dummy states, 7 and 8, are added into the schedule to assure proper time dependency between states in the current stage and the past stage. As in Fig. 6, each stage is partitioned into four minor cycles. At the second minor cycle (see Fig. 7(b)), for example, state 2 is processed in $P1$ while state 3 is processed in $P2$. They both require the old path metrics from states 1–4, as can be seen from Fig. 3. At the next minor cycle (see Fig. 7(c)), states 4 and 5 are processed, requiring the old path metrics from states 3–6. In a pipelined structure, the moving windows in Fig. 7(b) and (c) simply correspond to a set of fixed feedback connections from the output to the

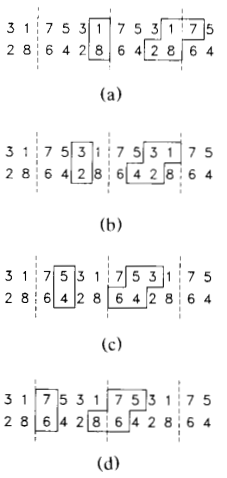


Fig. 7. (a) First minor cycle. States 1 and 8 are processed, which takes old path metrics of states 7 (wrong stage), 1, 8, and 2. (b) Second minor cycle. States 3 and 2 are processed, which takes old path metrics of states 1, 3, 2, and 4. (c) Third minor cycle. States 5 and 4 are processed, which takes old path metrics of states 3, 5, 4, and 6. (d) Fourth minor cycle. States 7 and 6 are processed, which takes old path metrics of states 5, 7, 6, and 8 (wrong stage).

input of the pipelined ACS. These connections are illustrated by net $A-D$ in Fig. 5.

The need for the two dummy states can be seen when the pipeline ACS's are at minor cycles 1 and 4 (Fig. 7(a) and (d)). In both cases, the moving windows take one old path metric from the wrong stage. Since states 1 and 6 depend only on two states (whose old path metrics remain in the correct stage), the operation is correct if we disable half of the ACS's at minor cycles 1 and 4. These are accomplished by the control signals $C1$ and $C2$ shown in Fig. 5.

We can summarize the overall area–time trade-off of the area-efficient pipelining architecture as follows. The speed is 33% slower than a six-state nonpipelined implementation, mainly due to the additional dummy states. On the other hand, the area is substantially less because of the reduction in: 1) the number of ACS's; 2) the number of feedback connections between ACS's; and 3) the number of interconnections between the branch metric calculators and the ACS's, stemming from the fact that each branch metric is now used in one ACS only. Although these area improvements are slightly offset by the incorporation of the pipeline latches in the ACS units, the net area reduction is measured to be roughly 50%, thereby providing a favorable overall area–speed trade-off.

B. Modulo Normalization

Metric normalization is another important design consideration for a Viterbi decoder. The normalization is required to prevent errors due to overflow or underflow during the updating and storage of the path metrics in a finite precision implementation of the Viterbi algorithm. Several normalization techniques have been proposed and used in the past. Most of them attempt to exploit the fact that, despite the potential unbounded magnitude of the path metrics, pairwise differences between them remain bounded in magnitude by a fixed quantity Δ, independent of the number of ACS recursion. We have developed an improved method for exact calculation of Δ [10] which leads to smaller ACS word-length requirements than those based upon previous bounds.

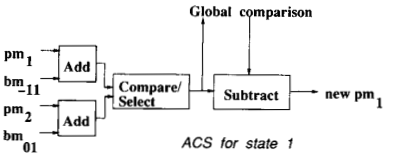


Fig. 8. A conventional metric normalization scheme.

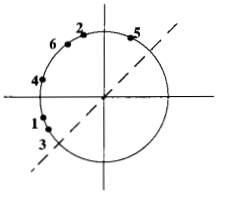


Fig. 9. Modulo normalization illustrated by "runners in a race."

One of the most frequently used metric normalization techniques is shown in Fig. 8. Updated path metrics from all ACS's are applied to a global minimization circuit whose output is broadcast back to every ACS to be subtracted. The result then defines the normalized path metric for each state. This technique introduces global wiring as well as significant delay in the critical ACS path. In addition, it is not compatible with the pipelined ACS architecture discussed previously.

In our chip we used a method called modulo normalization, which is best explained by a "runners-in-a-race" analogy. Suppose in a competitive track race, the maximum distance between any two runners is less than half the circumference, then it is straightforward to tell who is leading. In Fig. 9, for example, if all the runners are running counterclockwise, then 3 is leading and 5 is trailing. Therefore, with the circumference being twice the size of the difference bound Δ, the relative magnitude of the metrics can be determined by modulo arithmetic. This technique builds the normalization directly into the ACS operation, and it provides the most local and uniform design [11]. Furthermore, modulo normalization is especially well suited to the pipelined ACS architecture.

IV. Physical Design

Computer-aided design tools were heavily used in the simulation and physical design of the chip. The tool set [12] contains general-purpose place-and-route tools and

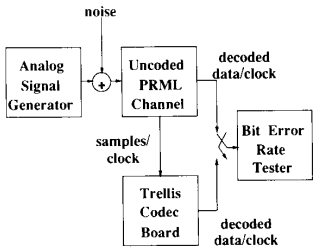


Fig. 10. Performance evaluation setup.

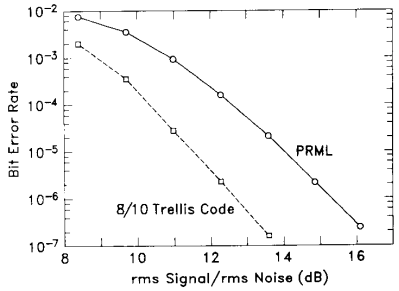
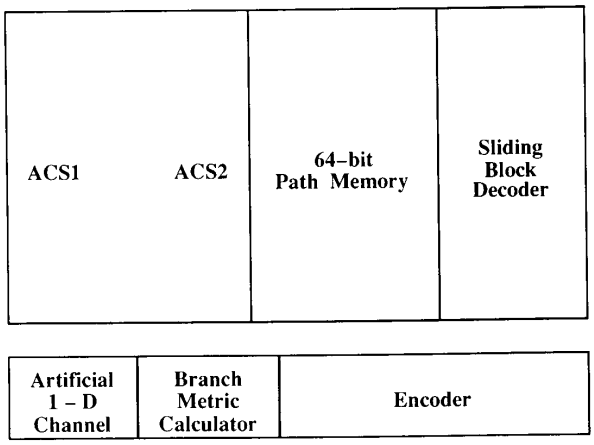
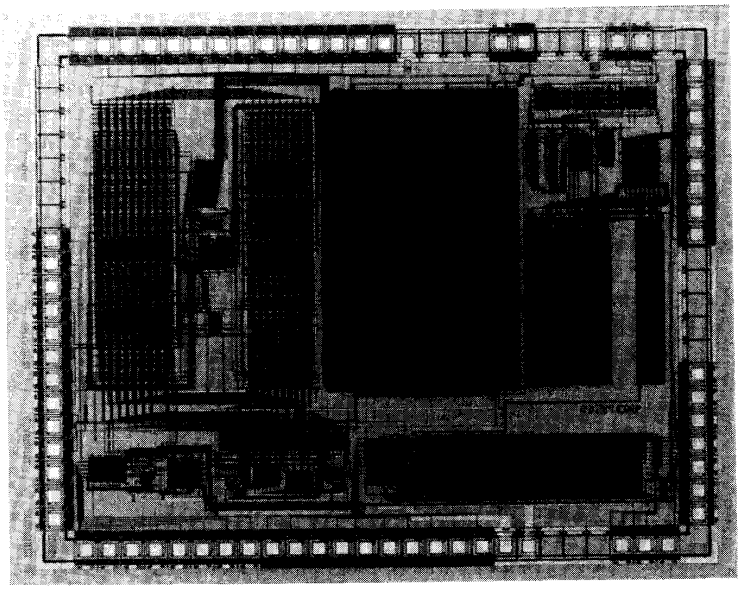


Fig. 11. Measured performance of trellis-coded partial response and uncoded PRML in additive Gaussian noise.



(a)



(b)

Fig. 12. (a) Chip floorplan. (b) Chip die micrograph.

module generators dedicated to three different types of modules: array modules, bit-sliced data paths, and random logic blocks. Bit-sliced data paths were used in the pipelined ACS's, the branch metric calculators, and a part of the sliding-block decoder. Array modules such as PLA were used to generate the encoder and decoder tables. Random logic blocks were used in various local control units.

High-level specifications and parameterization of these module generators greatly expedite the physical design of the chip. For example, a PLA can be generated by providing the size information and a personality matrix. Random logic blocks are compiled from a high-level hardware description language into standard cells. Bit-sliced data paths are generated from a netlist description of data-path cells and their interconnections. This netlist is described for one bit, and the actual word-length parameter of the data path is provided at run time. This makes possible a quick generation of the ACS's for global floorplanning before the exact word length is determined (e.g., from the calculation of Δ).

The path memory in the Viterbi detector, on the other hand, was customized by manual layout editing because it involves substantial replication of a basic component that depends heavily upon the trellis structure. The overall goal was to balance the design time and layout quality. The complete design (including architectural modification and final simulation) was finished in three man-months.

To aid in testing, an on-chip 1-D channel was incorporated into the design. In chip-test mode, the encoder output is fed to the internal (noiseless) 1-D channel, with the channel output then directed to the Viterbi detector, followed by the sliding-block decoder. The resulting decoded output should match a delayed version of the original encoder input. This feature proved to be very useful in the functional testing of the chip.

V. Experimental Results

The performance of the rate 8/10 MSN code was experimentally evaluated using the VLSI chip described in the previous sections. This section presents the experimental results.

Two chips were configured according to Fig. 1 to simulate a trellis-coded class-IV partial-response system. As mentioned previously, the encoder, detector, and decoder functions were implemented on the chip. The interleaving, deinterleaving, serial-to-parallel conversion, and parallel-to-serial conversion were implemented using off-the-shelf fast TTL logic. The AGC, filter, control, and equalizer functions shown in Fig. 1 were derived from an experimental partial-response maximum-likelihood (PRML) receiver.

The evaluation of the trellis-coded system was based on the experimental configuration shown in Fig. 10. The analog signal generator was used to synthesize class-IV partial-response waveforms for trellis-coded or uncoded sequences based on a random data input. Noise was added to the waveform, which was then applied to the PRML receiver. For the uncoded system, the decoded output from the PRML receiver was used to determine the error rate as a function of signal-to-noise ratio (SNR). For evaluation of the trellis-coded system performance, the PRML receiver was used only for the purpose of performing A/D conversion, and for timing and gain recovery. The sample sequence and the recovered clock from the receiver were applied to the trellis codec chips. The decoded data were used to measure the error rate. All performance comparisons were performed at a channel data rate of 30 Mb/s.

Fig. 11 shows measured performance of the uncoded and the trellis-coded system in the presence of additive noise. As shown, the trellis-coded system tolerates approximately 2.8 dB more noise than the uncoded system at an error rate of 10^{-7}. This measured improvement is in good agreement with analytic and simulated results reported in [4]. This added immunity to noise provided by trellis coding may be used to improve performance or increase the recording density, or both, in magnetic recording systems.

VI. Summary

The chip floorplan and micrograph are shown in Fig. 12(a) and (b). The die size is 4.2 mm $\times$ 5.2 mm in a 1.2-μm CMOS technology. As can be seen from the die photo, a six-ACS nonpipelined implementation would have significantly increased the size of the chip. Our area-efficient technique in fact makes it possible for a class-IV trellis codec to be implemented in a single chip, including two copies of the dicode codec and the analog front end shown in Fig. 1. Because the encoder and sliding block decoder are running at one-fifth of the speed of the Viterbi detector, they can be shared by the two dicode codecs for additional area savings.

The chip has been tested to correctly operate at a data rate of 12 Mb/s using a 30-MHz clock, with 0.55-W power dissipation. For the complete class-IV channel, the throughput rate is 24 Mb/s. Close to 3-dB coding gain was verified.

Even though our chip was designed for a specific partial-response channel, we believe that some of the novel architectural techniques described in this paper are applicable to the design of Viterbi detectors for other applications.

Acknowledgment

The authors wish to thank J. Sanz, H. Sussner, and I. Traiger for their assistance in initiating this project. We are also grateful to R. Hoyt, D. Petkovic, and R. Brodersen for their continued encouragement and support, and to P. Ziperovich and J. Wolf for their help with the experimental setup.

REFERENCES

[1] H. Kobayashi and D. Tang, "Application of partial-response channel coding to magnetic recording systems," *IBM J. Res. Develop.*, vol. 14, pp. 368–375, July 1970.

[2] H. Kobayashi, "Application of probabilistic decoding to digital magnetic recording systems," *IBM J. Res. Develop.*, vol. 15, pp. 64–74, Jan. 1971.

[3] G. Forney, Jr., "Maximum likelihood sequence detection in the presence of inter-symbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 3, pp. 363–378, May 1972.

[4] R. Karabed and P. Siegel, "Matched-spectral-null codes for partial response channels," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pt. II, pp. 818–855, May 1991.

[5] C. Rader, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. COM-29, no. 9, pp. 1399–1401, Sept. 1981.

[6] T. Ishitani, K. Tansho, N. Miyahara, S. Kubota, and S. Kato, "A scarce-state-transition VLSI viterbi-decoder for bit error correction," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 575–582, Aug. 1987.

[7] R. Orndorff, R. Chou, J. Krcmarik, T. Doak, and R. Koralek, "Viterbi decoder VLSI integrated circuits for bit error correction," in *Proc. 16th Annual Asilomar Conf. Circuits, Syst. Comput.* (Monterey, CA), Nov. 1983.

[8] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785–790, Aug. 1989.

[9] C. Shung, H. Lin, P. Siegel, and H. Thapar, "Area-efficient architectures for the Viterbi algorithm," in *Proc. 1990 IEEE Global Conf. Commun.*, vol. 3 (San Diego, CA), Dec. 1990, pp. 1787–1793.

[10] P. Siegel, C. Shung, T. Howell, and H. Thapar, "Exact bounds for Viterbi detector path metric differences," in *Proc. 1991 IEEE Int. Conf. Acoustics, Speech and Signal Processing* (Toronto, Canada), May 1991, pp. 1093–1096.

[11] C. Shung, G. Ungerboeck, P. Siegel, and H. Thapar, "VLSI architectures for metric normalization in the Viterbi algorithm," in *Proc. 1990 Int. Conf. Commun.* (Atlanta, GA), Apr. 1990, pp. 1723–1728.

[12] C. Shung *et al.*, "An integrated CAD system for algorithm-specific IC design," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 447–463, Apr. 1991.

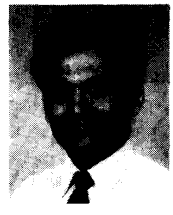
C. Bernard Shung received the B.S. degree in electrical engineering from National Taiwan University in 1981, and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley in 1985 and 1988, respectively.

From 1988 to 1990 he was a Visiting Scientist at IBM Research Division, Almaden Research Center, San Jose, CA, where he was engaged in the development of prototyping integrated circuits for pattern recognition and magnetic recording. In 1990 he joined the Department of Electronics Engineering, National Chiao Tung University in Hsinchu, Taiwan, Republic of China, where he is now an Associate Professor. His research interests include VLSI circuits and systems design for communications and signal processing, and computer-aided design for integrated circuits.

Paul H. Siegel (M'82–SM'90) was born in Berkeley, CA, in 1953. He received the B.S. degree in mathematics in 1975 and the Ph.D. degree in mathematics in 1979, both from the Massachusetts Institute of Technology, Cambridge. He held a Chaim Weizmann fellowship during a year of postdoctoral study at the Courant Institute, New York University.

He joined the research staff at IBM in 1980. He is currently Manager of the Signal Processing and Coding project at the IBM Almaden Research Center in San Jose, CA. His primary research interest is the mathematical foundations of signal processing and coding, especially as applicable to digital data storage channels. He holds several patents in the area of coding and detection for digital recording systems. He has taught courses in information and coding at the University of California, Santa Cruz, and at Santa Clara University, and was a Visiting Associate Professor at the University of California, San Diego, while at the Center for Magnetic Recording Research during the 1989–1990 academic year.

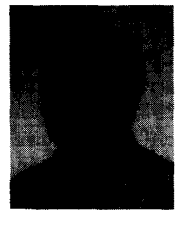
Dr. Siegel was elected to Phi Beta Kappa in 1974. He is currently a member of the Board of Governors of the IEEE Information Theory Society. He was a co-Guest Editor of the May 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY, and was a member of the Program Committee for the 1991 International Symposium on Information Theory.



Hemant K. Thapar (S'78–M'79), received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in August 1979.

He worked at the Bell Telephone Laboratories, Holmdel, NJ, from October 1979 to September 1984, first on the design method for AT&T's dynamic nonhierarchical routing network, and subsequently on signal processing and coding methods for high-speed data transmission. Since 1984 he has been with IBM, San Jose, CA, where he is involved in developing and prototyping advanced signal processing and coding techniques for digital magnetic recording systems. He is also an Adjunct Lecturer at Santa Clara University, where he regularly teaches courses in communication theory, digital communication, and signal processing.

Dr. Thapar is a member of Tau Beta Pi and Eta Kappa Nu, and was a co-recipient of the best paper award at Interface 1984.



Razmik Karabed received the B.S. degree in electrical engineering, the M.S. degrees in mathematics and in computer information and control engineering, and the Ph.D. degree in computer information and control engineering, all from the University of Michigan, Ann Arbor, in 1979, 1980, 1983, and 1985, respectively.

He joined IBM Research in 1985, and is presently with Almaden Research Center in San Jose, CA. His main research areas are Shannon theory, coding, and symbolic dynamics.