An Approach for Adaptively Approximating the Viterbi Algorithm to Reduce Power Consumption while Decoding Convolutional Codes

Russell Henning and Chaitali Chakrabarti

***Abstract*— Significant power reduction can be achieved by exploiting real-time variation in system characteristics. An approach is proposed and studied herein that exploits variation in signal transmission system characteristics to reduce power consumption while decoding convolutional codes. With this approach, Viterbi decoding is adaptively approximated by varying pruning threshold of the T-algorithm and truncation length while employing trace-back memory management. A heuristic is given for finding and adaptively applying pairs of pruning threshold and truncation length values that significantly reduce power to variations in signal-to-noise ratio, code rate, and maximum acceptable bit-error rate. The power reduction potential of different levels of adaptation is studied. High-level energy reduction estimates of 80% to 97% compared to Viterbi decoding are shown. Implementation insight and general conclusions about when applications can particularly benefit from this approach are given.**

***Index Terms*—Convolutional Code, Low Power, T-algorithm, Viterbi Algorithm.**

I. INTRODUCTION

A Viterbi decoder [1], [2], [3] is an important target for power reduction in a low-power signal processing device. It can account for more than one-third of power consumption during baseband processing in second generation cellular telephones [4]. As integrated circuits continue to become smaller and faster, the appeal of higher complexity Viterbi decoders for higher memory order convolutional codes increases. Higher memory order codes can achieve superior coding performance without requiring precious additional bandwidth. However, to counteract the exponential dependence of Viterbi decoder complexity on memory order in low-power designs, good power reduction methods that exploit the signal transmission system are needed.

To keep complexity reasonable for high memory order codes, a Viterbi decoder is typically implemented with trace-back memory management [5]. A conventional Viterbi decoder implementation employing trace-back memory management consists of three basic building blocks—a branch metric computation unit (BMCU), an add-compare-select unit (ACSU), and a survivor memory unit (SMU). The SMU results in over half of the power consumption in a conventional Viterbi decoder due to expensive memory accesses [6]. The ACSU can consume significant power as well, while BMCU power consumption tends to be relatively insignificant.

To reduce power consumption of Viterbi decoders with trace-back memory management, a variety of techniques are available. The T-algorithm, an approximation of the Viterbi algorithm that prunes the number of stored paths at each trellis stage, can conserve considerable energy [7], [8], [9], [10]. Trace-back memory size can be increased to reduce the frequency with which the trace-back operation is performed [4], [11], [12]. Systolic SMU hardware can be dynamically reconfiguring to take advantage of a shorter truncation length when signal-to-noise ratio (SNR) increases [13]. Clock-gating can be applied to systolic SMU hardware [14]. Trace-back routes can be stored and reused for consecutive trace-back operations [6], [14]. Decoder input can be re-coded to reduce switching activity and computation [15]. Low power ACSU architectures are also available [12], [16].

However, as is the case in many designs today, significant untapped power reduction potential lies in dynamically varying the decoder implementation according to real-time changes in system characteristics. The main overhead of implementing such an approach is typically the power consumption of monitoring the system characteristic variation. Examples include filter design, where coefficient values and the number of filter taps can be varied based on a filter quality factor such as output stopband energy attenuation, mean square error between the output and the

Manuscript received January 23, 2002. This work was supported by the NSF/SIUCRC Center for Low Power Electronics that is jointly funded by NSF, the State of Arizona and the member companies.

R. Henning was with Arizona State University, Tempe. He is now with General Dynamics Decision Systems, Scottsdale, AZ 85257 USA (e-mail: Russell.Henning@ gdds.com).

C. Chakrabarti is with Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: chaitali@asu.edu).

desired signal, SNR of the output, or bit-error rate (BER) of the output [17], [18]. The resulting energy reduction can be 60 to 90%. Other applications in which system characteristics have been exploited include Reed-Solomon channel coding [19], an encryption processor [20], and DCT/IDCT computation in an MPEG-2 video codec [21]. By translating variations in the workload to variations in the supply voltage, significant power savings have been obtained in DSP processors as well [22].

In this paper, an approach is presented that exploits variation in signal transmission system characteristics to reduce power consumption while decoding convolutional codes. In addition to high memory order, a modern Viterbi decoder can encounter multiple code rates (R) in a system where bandwidth availability, E_b/N_o, and/or maximum acceptable BER (MABER) for an application vary real-time. The proposed approach, referred to as Adaptive T-algorithm decoding, adapts two parameters of a T-algorithm decoder employing trace-back memory management to these real-time system changes in order to reduce energy consumption. These parameters are pruning threshold and truncation length. Reducing pruning threshold can reduce the average number of paths stored per stage in a T-algorithm decoder, which can in turn reduce energy consumption. Reducing truncation length can reduce the frequency with which trace-back is performed, which can also significantly reduce energy consumption. Of course, with such energy reductions come losses in coding performance. However, as long as these losses do not cause BER performance to exceed the MABER at any given time, they should be completely acceptable.

This paper studies the potential of adapting pruning threshold and truncation length of the T-algorithm decoder in order to achieve significant energy reduction. There are six main contributions of this paper:

1) A new system-dependent, low-power approach for decoding convolutional codes, called Adaptive T-algorithm decoding, is proposed.
2) A heuristic is given for finding and adaptively applying pairs of pruning threshold and truncation length values to variations in R, Eb/No, and MABER, in order to significantly reduce energy consumption.
3) Variation in the potential of this approach as signal transmission system characteristics (R, Eb/No, MABER) vary is studied.
4) Versatility and power reduction potential of different levels of adaptation are studied.
5) High-level energy reduction estimates of 80% to 97% compared to hardware Viterbi decoding are shown.
6) Implementation insight and general conclusions about when an application can particularly benefit from this approach are given.

The rest of this paper is organized as follows. In Section II the Adaptive T-algorithm decoding approach is developed. Section III describes the energy estimation method used to assess the new approach's energy reduction potential for different levels of adaptation versus Viterbi decoding. Experimental results are then given and analyzed in Section IV that demonstrate the potential of the new approach. Section V summarizes the paper.

II. Approach

The proposed Adaptive T-algorithm decoder essentially consists of 1) a T-algorithm decoder employing trace-back memory management, and 2) control for real-time adaptation of pruning threshold and truncation length. Significant energy is conserved by adapting truncation length and pruning threshold according to real-time system variation in terms of R, E_b/N_o, and MABER. In this section, important background is first given on Viterbi decoder implementation and the T-algorithm. Then, prior T-algorithm work is reviewed. Finally, a heuristic for implementation of Adaptive T-algorithm decoding is discussed.

A. Background

This subsection is intended to provide information about the Viterbi decoder [1], [2], [3], [5] and T-algorithm decoder [7] pertinent to understanding the capabilities of the proposed Adaptive T-algorithm decoder. A Viterbi decoder implementation for convolutional codes that employs trace-back memory management can be described in terms of four parameters, **memory order**, **trace-back memory length**, **truncation length**, and **code rates supported**. A T-algorithm decoder is essentially an approximation of the Viterbi decoder that, in general, stores fewer candidate paths than the Viterbi decoder on average. A T-algorithm decoder that employs trace-back memory management can be described with the same four parameters plus one more, **pruning threshold**.

A convolutional code is commonly referred to in the literature as rate $R = k/n$ with **memory order** m [3]. This means the code generates n outputs at a time from some combination of the current block of k inputs and the previous m blocks of k inputs. Codes with large m are popular today because, in general, the larger m is, the better the channel coding performance for the same bandwidth cost. This comes at the expense of increasing Viterbi decoder complexity, which has an exponential dependence on m.

Trace-back memory management is employed in a Viterbi decoder in the case of large m codes because of its reduced implementation complexity, area, and energy consumption versus register-exchange [23]. Figure 1 shows a block diagram of a generic Viterbi decoder architecture employing trace-back memory management. More detailed implementations of the blocks in Figure 1 can be found, for example, in [15]. With trace-back memory management, candidate paths that may correspond to the sequence of states used during encoding are stored in a RAM called trace-back

memory. **Trace-back memory length** is the maximum length of these paths. The trace-back operation is performed periodically such that once a path has been traced back the number of stages equal to the **truncation length** from the most recent stage, it is assumed that the remaining stages in trace-back memory represent the maximum likelihood path. Decoder outputs are taken directly from this remaining path. Note that the decoder latency is increased by the amount of time it takes to perform the trace-back operation.

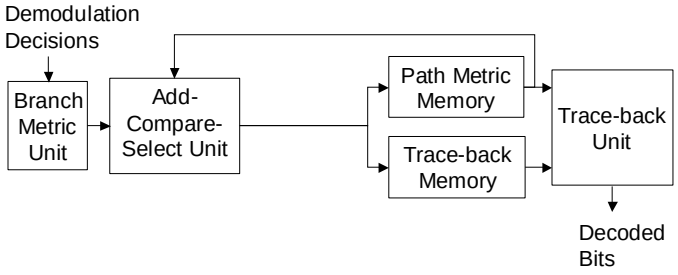

Fig. 1. Block diagram of a generic Viterbi decoder architecture employing trace-back memory management.

The larger the truncation length is, the more probable it is that the maximum likelihood path is indeed found. However, the larger the trace-back memory length is, the higher is the power and area required by the trace-back memory. To minimize trace-back memory length, truncation length can be set as small as possible without causing the decoder to exceed BER constraints, and trace-back memory length can be set at *truncation length* + 1. The drawback of this choice is that the trace-back operation is performed after each stage of path construction during steady state decoding, which costs significant energy if truncation length is large. If truncation length is instead less than *trace-back memory length* – 1, the trace-back operation can be performed at lower frequency because multiple outputs can be decoded per trace-back operation. The proposed Adaptive T-algorithm decoding approach exploits this aspect of trace-back memory management by keeping truncation length significantly less than trace-back memory length whenever possible.

A desirable feature for a decoder to support is multi-rate channel coding. Puncturing is a popular technique for allowing a Viterbi decoder to decode different convolutional code rates with the same trellis structure. When a higher rate code is used, bandwidth is conserved at the expense of code performance. The **code rates supported** by a decoder affect its design in that larger truncation lengths are typically needed to fully exploit the code performance capability of higher rates [24]. As mentioned above, larger truncation lengths have larger trace-back memory length requirements, which can add to trace-back memory power consumption and area.

A T-algorithm decoder implementation with trace-back memory management has similar dependence on trace-back memory length, truncation length, and supported code rates as described for the Viterbi decoder above. By comparing the block diagram of a T-algorithm decoder employing trace-back memory management in Figure 2 to the Viterbi decoder block diagram in Figure 1, it can be seen that the T-algorithm decoder is in fact very similar to the Viterbi decoder. The difference is that the T-algorithm decoder is able to reduce the impact of m on decoding complexity compared to the Viterbi decoder by defining a **pruning threshold**. This pruning threshold is compared to the accumulated branch metric associated with each path stored in the trace-back memory. If the accumulated branch metric minus the smallest accumulated branch metric for the current trellis stage exceeds the threshold, that path is not extended in the next trellis stage. Thus, the number of paths maintained is effectively reduced below that of the Viterbi decoder if pruning threshold is small enough. More detailed implementations of the blocks in Figure 2 can be found, for example, in [9], [25].

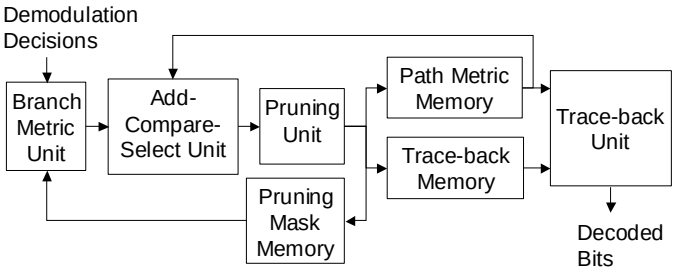

Fig. 2. Block diagram of a generic T-algorithm decoder architecture employing trace-back memory management.

The important aspect of the T-algorithm decoder in Figure 2 that is exploited with the new Adaptive T-algorithm decoding approach is that the lower the pruning threshold is, the less energy consumed during path construction. Energy reduction results in two ways. First, the pruning unit does not write discarded paths to trace-back memory. This is made possible by a trace-back memory that allows one bit to be addressed and written to at a time. Second, corresponding branch metric and add-compare-select computations in the next stage are not performed based on pruning mask memory contents that indicate which paths have been discarded.

Note, however, that the more paths that are pruned, the greater the chance that the maximum likelihood path will be pruned, causing decoding performance to degrade. The greatest drawback of the T-algorithm appears to be that long error events can occur unless the number of paths that may be stored per stage is 2^m and the threshold is large enough [10]. Thus, the Adaptive T-algorithm decoder reduces power by attempting to choose the smallest pruning threshold in real time that will meet the performance constraints of the system, while always allowing up to 2^m paths to be stored per stage.

B. Prior T-algorithm Work

Previous work involving the T-algorithm in [7], [8], [10] is the foundation for the proposed approach. In [7], the relationship between the average number of surviving paths

(an indication of energy consumption) and BER for fixed E_b/N_o and variation in T-algorithm pruning threshold is studied. Rate-1/2 convolutional codes with various constraint lengths are considered. Impressive reductions are achieved for small decreases in BER compared to the Viterbi algorithm. However, the effects of changing truncation length and multiple code rates are not considered. Furthermore, use of threshold optimization to take advantage of real-time variations in system characteristics is not directly discussed.

There is little information in [7] about the relationship between energy consumption and E_b/N_o when BER expectation is fixed and pruning threshold is varied. A pair of E_b/N_o values is considered at most, showing that fewer average surviving paths are required to achieve the same BER when E_b/N_o is higher. The relationship between the average number of surviving paths and E_b/N_o for a fixed pruning threshold is, however, demonstrated.

In [8], the T-algorithm threshold is adapted as SNR varies for a channel equalizer but not necessarily in an optimal way. The approach taken is to monitor the maximum difference between accumulated path metrics at each stage of the trellis as a measure of whether a higher or lower threshold is needed.

The main improvement to the T-algorithm provided in [10] is implementation insight that allows long error events to be avoided, rather than handled with cumbersome and sometimes inefficient techniques like periodically forcing the path to a known state. By allowing up to 2^m survivors to be stored in trace-back memory at any time and providing a threshold large enough so that the number of survivors can reach 2^m after an error event, such very long error events are avoided.

Energy estimates are not given in [7], [8], [10]. Instead, performance is discussed in terms of the average number of surviving paths, which gives some indication of energy reduction potential. But, not all energy consumption of a T-algorithm implementation is proportional to the average number of surviving paths, especially when trace-back memory management is employed. High-level energy estimation that better indicates the energy reduction potential of the Adaptive T-algorithm decoder is given in Section III.

C. Adaptive T-algorithm Decoding

Figure 3 shows a block diagram of a generic Adaptive T-algorithm decoder architecture employing trace-back memory management. The main difference between this decoder and the T-algorithm decoder in Figure 2 is the addition of a look-up unit that changes the pruning threshold used by the pruning unit and the truncation length used by the trace-back unit according to variations in estimates of E_b/N_o, *R*, and MABER. Also, the Adaptive T-algorithm decoder is always implemented such that the number of survivors that can be stored per stage is allowed to reach 2^m. This constraint and careful adaptation of the pruning threshold allow very long error events that can significantly reduce T-algorithm decoding performance to be avoided. In fact, an additional benefit of adapting pruning threshold with the Adaptive T-algorithm decoder is that without fixing it at a conservatively high value, the threshold can be kept from ever becoming so small that these very long error events occur.

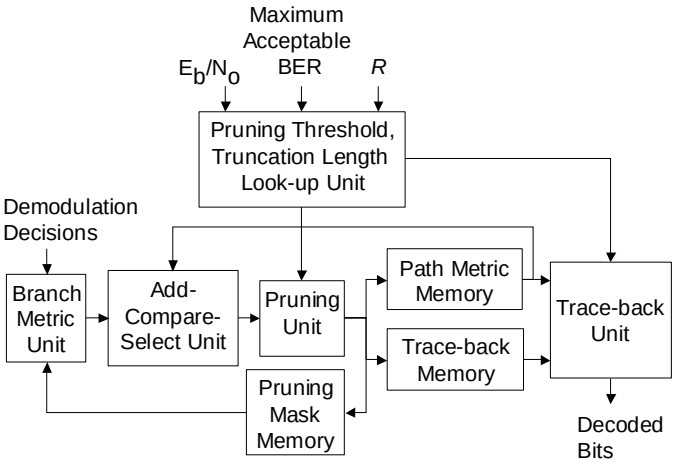

Fig. 3. Block diagram of a generic Adaptive T-algorithm decoder architecture employing trace-back memory management.

Using high-level energy estimation and computer simulation, the following heuristic can be used to efficiently find a trace-back memory length and good pairs of pruning threshold and truncation length for the look-up unit of a typical Adaptive T-algorithm decoder implementation:

1) Choose the MABER that is most important to support for the implementation.
2) Determine the decoder operation point being supported that requires maximum truncation length. This point will typically correspond to the lowest E_b/N_o supported for the highest *R* that does not cause MABER to be exceeded.
3) Choose the trace-back memory length in one of three ways.
   a. Choose the trace-back memory length that minimizes energy of the operation point being supported that requires the largest truncation length, because trace-back memory length will impact this case the most.
   b. If the operation point being supported that requires the largest truncation length will not be encountered very often in practice or trace-back memory size is tightly constrained, it is possible to gain a little more power reduction for operational points not requiring such a large truncation length by setting trace-back memory length at one more than the largest truncation length supported.
   c. Since trace-back operations can be a significant speed bottleneck, a trace-back memory length that is

much larger than the maximum truncation length supported might be necessary to meet speed constraints.

4) For each *R*, use computer simulation to find the minimum pruning thresholds for a reasonable set of E_b/N_o values that do not cause MABER to be exceeded with truncation length fixed at *trace-back memory length* – 1.
5) For each operation point and corresponding pruning threshold found in Step 4, find the minimum truncation length that does not cause MABER to be exceeded.
6) For each operation point in the previous step,
   a. increase the pruning threshold used in the previous step by one,
   b. find the minimum truncation length that does not cause MABER to be exceeded,
   c. compare the energy estimate for this configuration to the same operation point in the previous step,
   d. if the energy estimate is less than in the previous step, repeat Step 6 for the operation point, otherwise the threshold and truncation length of the previous step is the optimal choice for the E_b/N_o, *R*, and MABER of the operational point and this information should be stored in the look-up unit.
7) If it is possible to support multiple MABER settings, repeat Steps 4-6, for a reasonable number of MABER settings.

During operation of the Adaptive T-algorithm decoder, the pruning threshold and truncation length pair can be chosen from the look-up table corresponding to 1) the current *R*, 2) the closest MABER setting in the look-up table that does not exceed the current estimate of MABER, 3) the closest E_b/N_o value in the look-up table that does not exceed the current estimate of E_b/N_o. Note, it is not prudent to search for and store look-up table entries for excessive numbers of E_b/N_o, *R*, and MABER combinations. Thus, some additional investigation into the tradeoffs of look-up table construction for a given application is necessary. In addition to studying the benefits of applying the above heuristic, the examples of Section IV consider the energy reduction of lesser levels of adaptation that reduce the number of look-up table entries.

III. Energy Estimation

High-level energy estimation can be used to assess general hardware implementation of the Adaptive T-algorithm decoder. Estimates for the Adaptive T-algorithm decoder included in Section IV consist of energy consumption contributions from the following decoder operations based on Figure 3:

1) Branch metric calculation for a three-bit soft decision decoder input
2) Add-compare-select (ACS) operation
3) Compares to determine which paths should be pruned
4) Writing the one-bit transition of each path that is not pruned to the trace-back memory (T-SRAM)
5) Reading one-bit path transitions during trace-back from T-SRAM
6) Writing accumulated metrics to the path metric memory (A-SRAM)
7) Reading path metrics from A-SRAM
8) Writing the mask that determines which paths are pruned to the pruning mask memory (M-SRAM)
9) Reading the mask that determines which paths are pruned from M-SRAM

Each of these operations can be broken down into 8-bit adds, 8-bit compares, memory reads, memory writes, and associated control. It is assumed that three different SRAM memories are employed in this estimate to store trace-back paths (T-SRAM), accumulated metrics (A-SRAM), and the pruning mask (M-SRAM). The additions and compares are assumed to consume about the same amount of energy. The energy consumption of memory accesses relative to additions can be extrapolated from [26], [27]. The resulting energy consumption values for each basic operation and associated control, normalized by the energy of the 8-bit add, are shown in Figure 4 for the decoder energy estimates in Section IV. Note that the relatively large power consumption of T-SRAM reads and writes in Figure 4 contribute to the strong impact that variation in pruning threshold and truncation length have on power consumption.

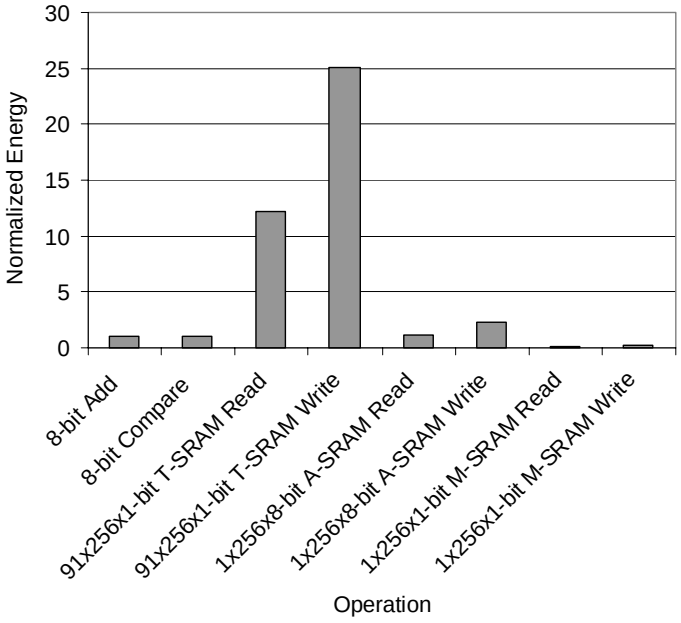

Fig. 4. Relative energy consumption of basic operations used for Adaptive T-algorithm decoder energy estimates in Section IV.

High-level estimates of Adaptive T-algorithm decoder energy consumption are calculated by simply determining how many times each operation is executed on average per

stage and multiplying each value by the corresponding energy estimate for the operation. The average number of times nearly all calculations are performed per stage is proportional to the average number of surviving paths per stage. As pruning threshold is reduced by the Adaptive T-algorithm decoder, the average number of surviving paths per stage is reduced, resulting in significant computation reduction due to this dependency. This is one of the keys to the Adaptive T-algorithm decoder's power reduction ability.

The other key to power reduction with the Adaptive T-algorithm decoder is that the average number of trace-back memory reads performed per stage is proportional to the number of trellis stages that are accessed during trace-back, *trace-back memory length*, divided by the number of stages that are processed between trace-back operations, *trace-back memory length – truncation length*. Thus, as truncation length is reduced by the Adaptive T-algorithm decoder from *trace-back memory length* – 1 to just one stages less, the number of trace-back memory reads is cut in half, which significantly impacts power reduction. On the other hand, if truncation length is already much less than *trace-back memory length* – 1, then it must be reduced by many more stages to have a significant impact on power consumption.

The overhead of an adaptive versus fixed T-algorithm implementation comes from monitoring R, E_b/N_o, and BER expectation and then choosing the appropriate threshold and truncation length to use. This overhead can vary from implementation to implementation depending on how often the system characteristics are sampled and how many low energy configurations are stored. However, because none of these overhead operations are expected to be performed at a high rate relative to decoder operations in practice, their energy consumption can be assumed negligible.

R and E_b/N_o should already be monitored in most implementations. BER expectation depends on the user and the information being transmitted. Choosing the appropriate threshold and truncation length to use can be implemented as a lookup table that is accessed based on the current estimates of E_b/N_o, R, and MABER. It is desirable, to keep the size of this lookup table small to minimize overhead by only storing a reasonable number of threshold and truncation length pairs.

The Viterbi decoder energy estimates in Section IV assume the same architecture and implementation as the Adaptive T-algorithm decoder estimates, except for the blocks in Figure 3 that are not in Figure 1 and the slightly smaller 85X256X1-bit T-SRAM used in the Viterbi decoder architecture. Note that the generic architectures and relative energy estimation methods used in this paper are intended to provide a more generally applicable view of Adaptive T-algorithm decoding potential than a particular implementation would. However, it is important to remember that energy reduction results used to gauge potential of the Adaptive T-algorithm decoding approach in the next section may be better or worse than for a particular implementation due to differences in implementation method and integration technology.

IV. Experimental Results and Analysis

In this section, the energy reduction potential of the Adaptive T-algorithm decoder is demonstrated through experimental results for additive white Gaussian noise channels. Two example cases involving punctured codes are studied. The first case is a system with fixed MABER and variable E_b/N_o. The second case is a system with fixed E_b/N_o and variable MABER. The results presented were found with the help of the heuristic presented at the end of Section II and the energy estimation method described in Section III. Computer simulation was performed via C programming and a PC. Versatility and power reduction potential of different levels of adaptation are studied. Accompanying analysis provides insight and guidance for implementation of the approach.

A. Fixed Maximum Acceptable BER, Variable E_b/N_o Example

First, consider a Viterbi decoder implementation employing trace-back memory management that decodes R = {1/2, 2/3, 3/4}, m = 8 punctured convolutional codes. Assume that with this decoder, acceptable quality is achieved for E_b/N_o as low as 2 dB while decoding the rate-1/2 code. The BER achieved in this case, approximately 0.005, is the fixed MABER this implementation is designed to handle.

A truncation length of about 81 stages is needed to decode the rate-3/4 code with negligible quality loss. This implementation uses a trace-back memory length of 85 trellis stages to support the truncation length of 81 stages while minimizing energy consumption for the rate-3/4 code. (Note that the rate-1/2 and rate-2/3 codes only require truncation lengths of 45 and 72 stages, respectively; however, using these truncation lengths for this implementation provides negligible energy reduction compared to simply using 81 stages for all code rates.) For trace-back memory access speeds as slow as 150 ns, throughput of coded bits up to about 600 kbps can be supported with this design by performing trace-back once for every 8 bits received by the decoder.

To save significant energy, T_f, a T-algorithm decoder with fixed pruning threshold and truncation length, can be substituted for the Viterbi decoder in this implementation. In this case, energy consumption is minimized for the rate-3/4 code by employing a trace-back memory length of 91 trellis stages to support the truncation length of 81 stages. To avoid very long error events, trace-back memory width, or the number of paths that can be stored per decoder trellis stage, is set at 2^m = 256 in this example, as it is for the Viterbi decoder. By employing a fixed pruning threshold of 15, the 0.005 MABER is not exceeded for E_b/N_o greater than 2.1 dB while decoding the rate-1/2 code. Thus, with this pruning threshold, a slight coding loss of about 0.1 dB is incurred

when decoding the rate-1/2 code. The loss resulting from the fixed pruning threshold of 15 for rate-2/3 and rate-3/4 codes is negligible. Figure 5 shows decoder performance for all three codes. Only points that achieve a BER below the MABER of 0.005 are shown in Figure 5, since it is assumed that this constraint governs operation of the multi-rate system.

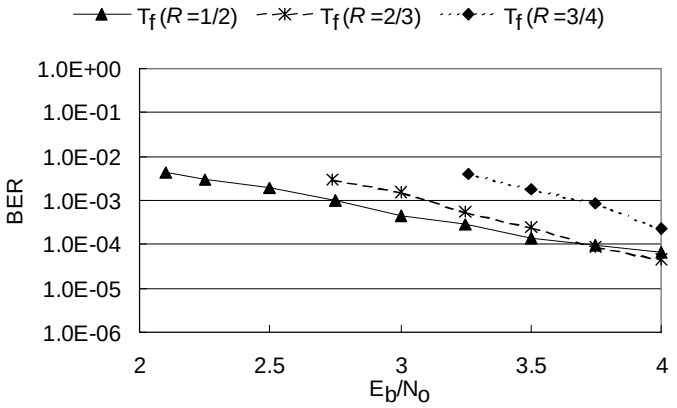

Fig. 5. BER versus E_b/N_o curves for T_f, a T-algorithm decoder with fixed pruning threshold and truncation length, as *R* varies. Pruning threshold = 15, truncation length = 81, and trace-back memory length = 91 for all points shown. Points shown do not exceed the MABER of 0.005.

For its relatively small coding loss, the normalized hig level energy estimates in Figure 6 indicate that T_f c consume 52% to 94% less energy than the multi-rate Vite decoder when E_b/N_o is between 2.1 dB and 4.0 dB. practice, energy reduction achieved will depend on how *R* a E_b/N_o vary with time. To keep results as independent integration technology as possible, all energy estimates in tl figure and the rest of this section are normalized by t energy estimate for the Viterbi decoder implementation. T breakdown of normalized energy consumption for the Vite decoder estimate based on Figure 1 is 0.056 for the Bran Metric Unit, 0.204 for the Add-Compare-Select Unit and Pau Metric Memory accesses, and 0.740 for Trace-back Memory accesses and the Trace-back Unit.

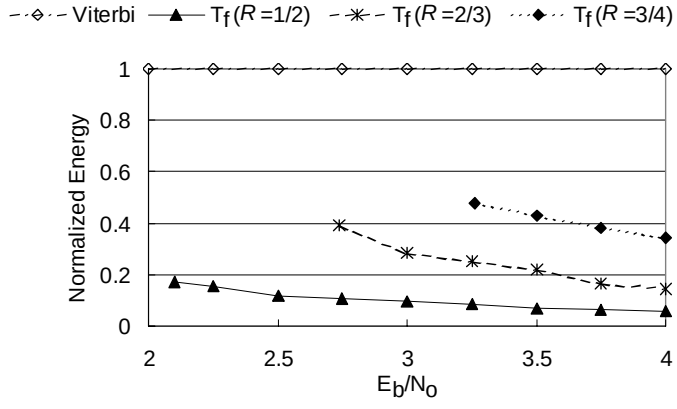

Fig. 6. Normalized energy estimates for the Viterbi decoder and T_f, a T-algorithm decoder with fixed pruning threshold and truncation length, as *R* and E_b/N_o vary. Truncation length = 81 and trace-back memory length = 85 for all Viterbi decoder points shown. Pruning threshold = 15, truncation length = 81, and trace-back memory length = 91 for all T_f points shown. Points shown do not exceed the MABER of 0.005.

Additional energy reduction is possible for higher E_b/N_o. However, energy consumption slowly converges to the same point for all code rates as the average number of paths kept by the T-algorithm per decoding stage converges to one. Thus, the upper limit on energy reduction for T_f relative to the Viterbi decoder is about 97%, based on high-level estimation.

For only slightly more control complexity, negligible energy overhead, and coding loss that still does not allow BER to exceed 0.005, Tra, an Adaptive T-algorithm decoder that adapts pruning threshold and truncation length to *R* only, can save even more energy than T_f, while employing the same basic architecture. High-level energy estimation indicates that when E_b/N_o is between 2.1 dB and 4.0 dB, T_{ra} consumes 4% to 15% less energy than T_f for the rate-1/2 code, 51% to 53% for the rate-2/3 code, and 63% to 66% for the rate-3/4 code, as shown in Figure 7.

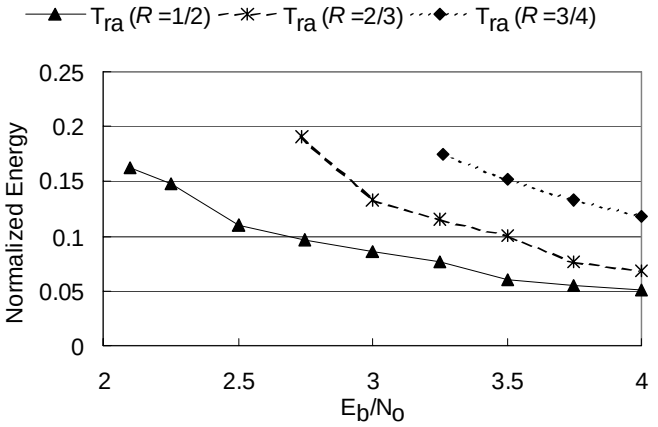

Fig. 7. Normalized energy estimates for T_{ra}, an Adaptive T-algorithm decoder that adapts pruning threshold and truncation length to *R* only, as *R* and E_b/N_o vary. For the rate-1/2 code points, pruning threshold = 15 and truncation length = 54. For the rate-2/3 code points, pruning threshold = 12 and truncation length = 72. For the rate-3/4 code points, pruning threshold = 11 and truncation length = 81. Trace-back memory length = 91 for all points. All points shown do not exceed the MABER of 0.005.

The energy consumption of T_{ra} can be further improved upon, while continuing to use the same basic architecture as T_f, by additionally adapting pruning threshold to variations in E_b/N_o, without allowing BER to exceed 0.005. This Adaptive T-algorithm decoder shall be referred to as T_a. High-level energy estimation indicates that when E_b/N_o is between 2.1

dB and 4.0 dB, T_a consumes 0% to 40% less energy than T_{ra} for the rate-1/2 code, 0% to 30% for the rate-2/3 code, and 0% to 47% for the rate-3/4 code, as shown in Figure 8.

Note that truncation length can also be adapted to variations in both *R* and E_b/N_o but, for the best case in this example, only provides 53% less energy than T_{ra} for the rate-3/4 code as opposed to the 47% reduction achieved by T_a, based on high-level energy estimation. Thus, it is probably not necessary to find and store optimal truncation lengths in memory according to *R* and E_b/N_o in this case. However, in situations where it is necessary to minimize trace-back memory length such that truncation length for the Adaptive T-algorithm decoder is within a few stages of trace-back memory length much of the time, adapting truncation length to variations in both *R* and E_b/N_o can be worth the added effort [28].

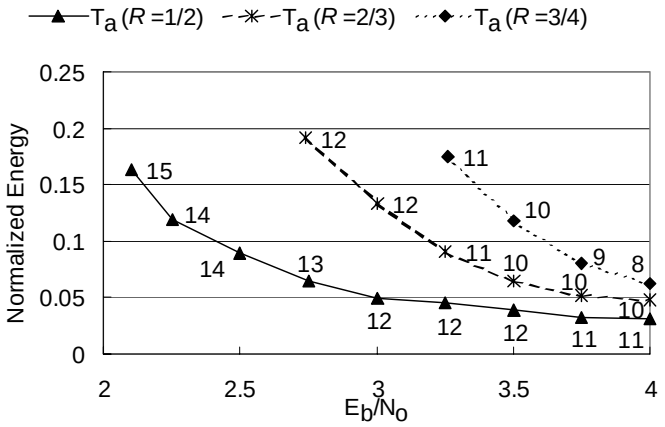

Fig. 8. Normalized energy estimates for T_a, an Adaptive T-algorithm decoder that adapts pruning threshold to both *R* and E_b/N_o while adapting truncation length to *R* only, as *R* and E_b/N_o vary. Each point is labeled with its correspondi pruning threshold. For the rate-1/2 code points, truncati length = 54. For the rate-2/3 code points, truncation length 72. For the rate-3/4 code points, truncation length = 8 Trace-back memory length = 91 for all points. All poi shown nearly reach, but do not exceed, the MABER of 0.00

In practice, the system in which the T_a decoder is employ might conserve bandwidth whenever possible by alwa choosing the highest *R* that will not cause the MABER to exceeded for a given E_b/N_o. Though this policy does r significantly affect the energy consumption of the Vite decoder, it can cause T_a, as well as T_f, to expend more ener In these decoders, energy consumption of a lower rate code is generally lower than a higher rate code because fewer paths on average are kept by the T-algorithm for the same E_b/N_o. Therefore, in some cases, significant energy might be conserved if a lower *R* (higher bandwidth) than necessary is used when surplus bandwidth is available. For example, if E_b/N_o = 3.25 dB, rather than conserve bandwidth by employing a rate-3/4 code, high-level energy estimates indicate that about 48% less energy is required by the T_a decoder if the rate-2/3 code is used and about 74% less is consumed for the rate-1/2 code. Note, however, that these energy reduction gains do decrease as E_b/N_o increases.

B. Fixed E_b/N_o, Variable Maximum Acceptable BER Example

Further application of the proposed Adaptive T-algorithm decoding approach can allow a T-algorithm decoder to adapt energy consumption to variation in MABER as well. MABER typically depends on user expectations for an application. MABER could increase if, for example, a user has control over it and understands that battery life can be significantly increased for a portable device by sacrificing some amount of quality. Conversely, MABER could decrease if increased quality is desired at the expense of additional energy consumption. At the same time, *R* and E_b/N_o can vary like the previous example; however, for simplicity, E_b/N_o is assumed to be fixed in the following example.

Figure 9 now shows the performance of T_f in the case where E_b/N_o is fixed at 3.75 dB. Though high-level energy estimates indicate that energy consumption is reduced by 62% for the rate-3/4 code, 84% for the rate 2/3 code, and 94% for the rate-1/2 code compared to the Viterbi decoder, there is little versatility available in terms of adapting to variation in MABER. If MABER is greater than about 0.00133, a rate-3/4 code can be used to minimize channel bandwidth required or more bandwidth can be sacrificed to further reduce power consumption with the rate-1/2 or rate-2/3 codes. If MABER is between 0.0001 and 0.00133, only the rate-1/2 and rate-2/3 codes are available. The lack of versatility comes from the fact that no matter what MABER is, the energy consumption for the chosen *R* is fixed.

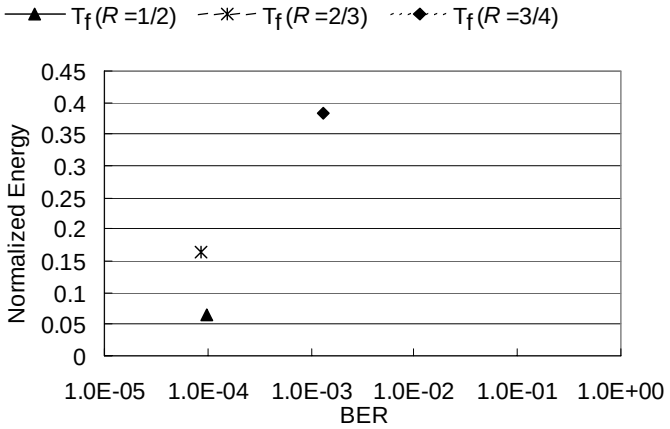

Fig. 9. Normalized energy estimates for T_f, a T-algorithm decoder with fixed pruning threshold and truncation length, as *R* and BER vary. E_b/N_o is fixed at 3.75 dB. Pruning threshold = 15, truncation length = 81, and trace-back memory length = 91 for all points shown.

As can be seen from Figure 10, a similar versatility problem is encountered with T_{ra}. Although, in this case, energy

consumption is reduced by 87% for the rate-3/4 code, 92% fc the rate 2/3 code, and 94% for the rate-1/2 code compared t the Viterbi decoder, based on high-level energy estimatior Recall that the additional energy reduction achieved by T compared to T_f comes at the price of some coding gain, as i reflected best by comparing the rate-2/3 code points in Figur 9 and Figure 10. The rate-2/3 code cannot be used fc MABER between 0.0001 and 0.00075 when employing T for decoding like T_f can.

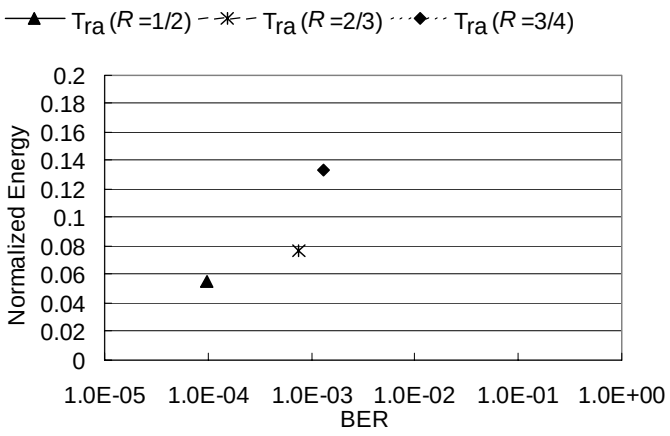

Fig. 10. Normalized energy estimates for T_{ra}, an Adaptive T-algorithm decoder that adapts pruning threshold and truncation length to *R* only, as *R* and MABER vary. E_b/N_o is fixed at 3.75 dB. For the rate-1/2 code point, pruning threshold = 15 and truncation length = 54. For the rate-2/3 code point, pruning threshold = 12 and truncation length = 72. For the rate-3/4 code point, pruning threshold = 11 and truncation length = 81. Trace-back memory length = 91 for all points.

On the other hand, if the pruning threshold of T_a is additionally allowed to adapt to MABER, the versatility problem of T_f and T_{ra} is solved, as shown in Figure 11. These results show that energy consumption can now be significantly further reduced for each *R* as MABER increases. An additional benefit of this adaptation is that code rates can generally be employed for lower MABER with T_a compared to T_f and T_{ra}, because higher pruning thresholds are available. In some cases, this feature can allow less bandwidth to be consumed for higher MABER. For example, if MABER is 0.00025, T_a allows this constraint to be met with the rate-3/4 code, whereas T_{ra} forces rate-1/2 code usage.

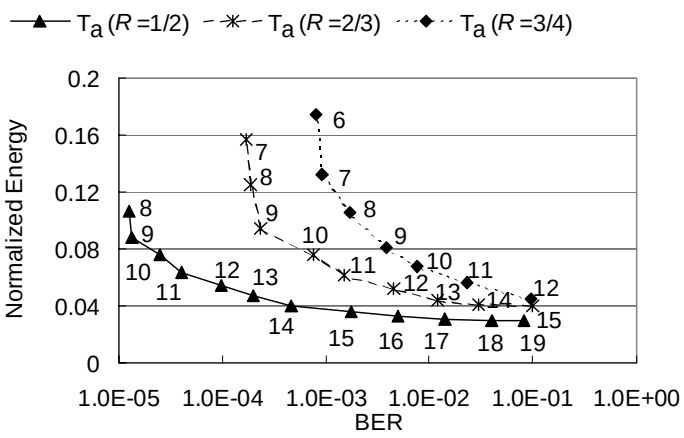

Fig. 11. Normalized energy estimates for T_a, an Adaptive T-algorithm decoder that adapts pruning threshold to *R*, E_b/N_o, and MABER while adapting truncation length to *R* only, as *R* and MABER vary. E_b/N_o is fixed at 3.75 dB. Each point is labeled with its corresponding pruning threshold. For the rate-1/2 code points, truncation length = 54. For the rate-2/3 code points, truncation length = 72. For the rate-3/4 code points, truncation length = 81. Trace-back memory length = 91 for all points.

V. Conclusions

The Adaptive T-algorithm decoder adaptively approximates the Viterbi decoder according to variations in convolutional code rate, E_b/N_o, and MABER. The Adaptive T-algorithm decoder adapts the pruning threshold of the T-algorithm and truncation length while employing trace-back memory management. It is shown in this paper that, by substituting an Adaptive T-algorithm decoder for a Viterbi decoder or fixed T-algorithm decoder, significant energy reduction can potentially be achieved, without exceeding the MABER.

A heuristic is given for finding and adaptively applying pairs of pruning threshold and truncation length values to variations in *R*, E_b/N_o, and MABER, in order to significantly reduce energy consumption is given in this paper. However, the level of adaptation implemented in an Adaptive T-algorithm decoder must consider the trade-off between the versatility and power reduction potential it provides and the expense of the associated look-up table size.

The potential of two levels of Adaptive T-algorithm decoding are studied in this paper with two examples. The results for the fixed MABER example are summarized in Table I. This table shows high-level, incremental % energy reduction estimates for each decoder considered in this example. By incremental, it is meant that the T_f decoder % energy reduction estimate is with respect to the Viterbi Decoder, while the T_{ra} decoder estimate is with respect to the T_f decoder, and so on. The same set of results is summarized in Table II for the fixed E_b/N_o example.

TABLE I
SUMMARY OF HIGH-LEVEL, INCREMENTAL % ENERGY REDUCTION ESTIMATES FOR EACH DECODER CONSIDERED IN THE FIXED MABER EXAMPLE

Decoder	Incremental % Energy Reduction
Viterbi Decoder	NA
T_f Decoder	52-94% less than Viterbi Decoder
T_{ra} Decoder	4-66% less than T_f Decoder
T_a Decoder	0-47% less than T_{ra} Decoder

TABLE II
SUMMARY OF HIGH-LEVEL, INCREMENTAL % ENERGY REDUCTION ESTIMATES FOR EACH DECODER CONSIDERED IN THE FIXED E_B/N_O EXAMPLE

Decoder	Incremental % Energy Reduction
Viterbi Decoder	NA
T_f Decoder	62-94% less than Viterbi Decoder
T_{ra} Decoder	14-65% less than T_f Decoder
T_a Decoder	0-66% less than T_{ra} Decoder

In general, the greater E_b/N_o and MABER are, the higher the energy reduction afforded by an Adaptive T-algorithm decoder. However, as both E_b/N_o and MABER increase from the minimum levels that can be supported by an Adaptive T-algorithm decoder implementation, a point of diminishing returns is rapidly converged upon. This behavior is contributed to by both truncation length reduction and pruning threshold reduction. The convergence rate due to truncation length reduction is higher than that of pruning threshold reduction, causing adaptation in truncation length to become less important as truncation length becomes several stages less than trace-back memory length. Similar conclusions are expected for convolutional codes having other code rates, trace-back memory lengths, and *m* values than studied here, as well as software decoder implementations.

REFERENCES

[1] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Trans. Information Theory*, vol. IT-13, pp. 260-269, 1967.

[2] G. D. Forney, Jr., "The Viterbi Algorithm", *Proc. IEEE*, vol. 61, no. 3, pp. 218-278, Mar. 1973.

[3] S. Lin and D. J. Costello, Jr., *Error Control Coding*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

[4] I. Kang and A. Willson Jr., "Low-Power Viterbi Decoder for CDMA Mobile Terminals", *IEEE J. of Solid-State Circ*., vol. 33, no. 3, pp. 473-482, Mar. 1998.

[5] R. Cypher and C. B. Shung, "Generalized Trace-Back Techniques for Survivor Memory Management in the Viterbi Algorithm", *IEEE J. VLSI Sig. Proc*., vol. 5, no. 1, pp. 85-94, Jan. 1993.

[6] D. Oh and S. Hwang, "Design of a Viterbi Decoder with Low Power using Minimum-Transition Traceback Scheme", *Electronics Letters*, vol. 32, no. 24, pp. 2198–2199, Nov. 1996.

[7] S. J. Simmons, "Breadth-First Trellis Decoding with Adaptive Effort", *IEEE Trans. on Comm*., vol. 38, no. 1, pp. 3-12, Jan. 1990.

[8] K. C. Chang and W. H. Lam, "An Adaptive Reduced-State Channel Equalizer with T-Algorithm", in *Proc. of VTC*, vol. 2, pp. 1237-1240, 1994.

[9] W.-T. Lee, et al., "A Single-Chip Viterbi Decoder for a Binary Convolutional Code using an Adaptive Algorithm", *IEEE Trans. Consumer Electronics*, vol. 41, no. 1, pp. 150-159, Feb. 1995.

[10] F. Chan and D. Haccoun, "Adaptive Viterbi Decoding of Convolutional Codes over Memoryless Channels", *IEEE Trans. Comm*., vol. 45, no. 11, pp. 1389-1400, Nov. 1997.

[11] C. M. Rader, "Memory Management in a Viterbi Decoder", *IEEE Trans. Comm*., vol. COM-29, no. 9, pp. 1399-1401, Sep. 1981.

[12] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-State 10-mW Rate-1/3 Viterbi Decoder", *IEEE J. Solid-State Circ*., vol. 35, no. 6, pp. 826-834, Jun. 2000.

[13] K. Takahashi, et al., "Reconfigurable Systolic Viterbi Decoder", in *Proc. of VTC*, vol. 3, pp. 1629-1632, 1999.

[14] J. H. Ryu, et al., "Low Power Viterbi Decoder Architecture with a New Clock-Gating Trace-Back Unit", in *Proc. of ICVC*, pp. 297-300, 1999.

[15] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi Decoder VLSI Implementation and its Performance" *IEEE Trans. on Comm*., vol. 41, no. 8, pp. 1170-1178, Aug. 1993.

[16] C. Tsui, R. Cheng, and C. Ling, "Low Power ACS Unit Design for the Viterbi Decoder" in *Proc. of ISCAS'99*, vol. 1, pp. 137-140, 1999.

[17] J. Ludwig, S. Nawab, and A. Chandrakasan, "Low Power Digital Filtering using Approximate Processing," *IEEE J. Solid-State Circ*., vol. 31, no. 3, pp. 395-400, Mar. 1996.

[18] M. Goel and N. Shanbhag, "Dynamic Algorithm Transformations (DAT): A Systematic Approach to Low power, Reconfigurable Signal Processing", *IEEE Trans. on VLSI Systems*, vol. 7, no. 4, pp. 463-476, 1999.

[19] M. Goel and N. Shanbhag, "Low-power Channel Coding via Dynamic Reconfiguration," in *Proc. of ICASSP*, pp. 1893-1896, Mar. 1999.

[20] J. Goodman, A. Chandrakasan, and A. Dancy, "Design and Implementation of a Scalable Encryption Processor with Embedded Variable DC/DC Converter", *ACM Design Automation Conference (DAC)*, pp. 855-860, June 1999.

[21] R. Henning and C. Chakrabarti, "An Approach for Enabling DCT/IDCT Energy Reduction Scalability in MPEG-2 Video Codecs", in *Proc. ICASSP*, pp. 1209-1212, May 2001.

[22] V. Gutnik and A. Chandrakasan, "Embedded power supply for low power DSPs," *IEEE Trans on VLSI Systems*, pp. 425-435, Dec 97.

[23] P. J. Black and T. H. Meng, "Hybrid Survivor Path Architectures for Viterbi Decoders", in *Proc. of ICASSP*, vol. 1, pp. 433-436, 1993.

[24] Y. Yasuda, K. Kashiki, and Y. Hirata, "High Rate Punctured Convolutional Codes for Soft-Decision Viterbi Decoding," *IEEE Trans. Comm*., vol. COM-32, no. 3, pp. 315-319, Mar. 1984.

[25] M.-H. Chan, et al., "IC Design of an Adaptive Viterbi Decoder", *IEEE Trans. Consumer Electronics*, vol. 42, no. 1, pp. 52-62, Feb. 1996.

[26] W.-T. Shiue, S. Udayanarayanan, C. Chakrabarti, "Data Memory Design and Exploration for Low Power Embedded Systems," *ACM Transactions on Design Automation of Electronic Systems*, 2000.

[27] F. Catthoor et al., *Custom Memory Management Methodology – Exploration of Memory Organization for Embedded Multimedia System Design*. Norwell, MA: Kluwer, 1998.

[28] R. Henning and C. Chakrabarti, "Low-Power Approach for Decoding Convolutional Codes with Adaptive Viterbi Algorithm Approximations", in *Proc. ISLPED*, pp. 68-71, Aug. 2002.

Russell Henning received the B.S., M.S., and Ph.D. degrees in electrical engineering from Arizona State University, Tempe, Arizona, in 1995, 1997, and 2001, respectively. He also graduated with Upper Division Honors from Barrett Honors College at Arizona State University in 1995.

His work during summer internships with Intel in 1997, 1999, and 2000 resulted in one patent with another one pending. Since 2001, he has been with General Dynamics Decision Systems in Scottsdale, AZ, developing secure communications technologies. His research interests are in the area of low power, system-level design for digital signal processing and communications applications.

Chaitali Chakrabarti received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India in 1984, and the M.S. and Ph.D degrees in electrical engineering from the University of Maryland at College Park, USA, in 1986 and 1990 respectively. She has been with the Department of Electrical Engineering, Arizona State University, Tempe, since August 1990 where she is now a Professor. Her

research interests are in the areas of low power systems design including memory optimization, high level synthesis and compilation, and VLSI architectures and algorithms for signal processing, image processing and communications.

Dr. Chakrabarti is a member of the Center for Low Power Electronics, the Consortium for Embedded and Inter-Networking Technologies and Connection One. She received the Research Initiation Award from the National Science Foundation in 1993, a Best Teacher Award from the College of Engineering and Applied Sciences, ASU, in 1994, and the Outstanding Educator Award from the IEEE Phoenix section in 2001. She has served on the program committees of ICASSP, ISCAS, SIPS, ISLPED and DAC. She is currently an Associate Editor of the IEEE Transactions on Signal Processing and the Journal of VLSI Signal Processing Systems.