A Reconfigurable, Power-Efficient Adaptive Viterbi Decoder

Russell Tessier, Sriram Swaminathan, Ramaswamy Ramaswamy, Dennis Goeckel and Wayne Burleson

***Abstract*—Error-correcting convolutional codes provide a proven mechanism to limit the effects of noise in digital data transmission. Although hardware implementations of decoding algorithms, such as the Viterbi algorithm, have shown good noise tolerance for error-correcting codes, these implementations require an exponential increase in VLSI area and power consumption to achieve increased decoding accuracy. To achieve reduced decoder power consumption, we have examined and implemented decoders based on the reduced-complexity adaptive Viterbi algorithm (AVA). Run-time dynamic reconfiguration is performed in response to varying communication channel noise conditions to match minimized power consumption to required error-correction capabilities. Experimental calculations indicate that the use of dynamic reconfiguration leads to a 69% reduction in decoder power consumption over a non-reconfigurable field-programmable gate array (FPGA) implementation with no loss of decode accuracy.**

I. Introduction

As the error-correcting capability of convolutional codes is improved by employing codes with larger constraint lengths K, the complexity of decoders [1] is increased. The Viterbi algorithm [1], which is the most extensively employed decoding algorithm for convolutional codes, is effective in achieving noise tolerance, but the cost is an exponential growth in memory, computational resources, and power consumption. To address this issue, the reduced-complexity adaptive Viterbi algorithm (AVA) [2], [3] has been developed. The average number of computations per decoded bit for this algorithm is substantially reduced versus the Viterbi algorithm, while comparable bit-error rates (BER) are preserved.

SRAM-based FPGA devices offer both hardware-level specialization and the capability to dynamically modify decoder hardware functionality at run-time. For power-sensitive systems, this flexibility can be exploited to achieve desired decoding accuracy, while minimizing decoder power consumption. During system operation, the constraint length of the convolutional encoder (and corresponding decoder) employed in the system is updated every few seconds based on channel noise characteristics. At the same time, the decoder parameters are optimized. Both the constraint length of the encoder and the decoder are chosen to maintain a prespecified decoder accuracy (bit error rate) and decoding rate with minimum power consumption at the receiver. This *slow* adaptation fits the target application of wireless communications, where current values of the channel path-loss and shadowing can be fed back to the transmitter with high reliability [4]. Additionally, power consumption due to FPGA decoder reconfiguration is amortized across thousands of received data values. Through experimentation, it is shown that when dynamic reconfiguration is applied to decoders mapped to Xilinx XC4036 and XCV1000 devices, a power savings of 27% and 69%, respectively, is achieved versus non-reconfigurable implementations.

In Section II, an overview of communication coding and the adaptive Viterbi algorithm is provided. Our AVA architecture is outlined in Section III and the experimental approach used to evaluate it is described in Section IV. The benefits of our AVA architecture are highlighted by experimental results presented in Section V. Section VI summarizes our efforts and offers directions for future work.

The authors are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.

II. Background

Numerous reduced complexity decoding methods have been introduced over the past 40 years, and many of these can result in reduced decoder power consumption in communication systems employing trellis-based codes [1]. These methods range from techniques that simplify the receiver trellis in a fixed manner for a given complexity (e.g. [5], [6]) to those that modify trellis searching in some way based on the specific received values [7]. Here, our interest is in the path-pruning techniques of [2], [3], which fall in the latter category. The adaptive Viterbi algorithm was introduced with the goal of reducing the *average* computation and path storage required by the Viterbi algorithm. Instead of computing and retaining all 2^{K-1} possible paths, only those paths which satisfy certain path cost conditions are retained at each stage, where a path's "cost" is defined as the Euclidean distance between the path and the received sequence. Path retention is based on the following criteria [2].

1) A threshold T indicates that a path is retained if its path cost is less than $d_m + T$, where d_m is the minimum cost among all surviving paths in the previous trellis stage.
2) The total number of survivor paths per trellis stage is limited to a fixed number, N_{max}, which is pre-set prior to the start of communication.

The first criterion allows high-cost paths that likely do not represent the transmitted data to be eliminated from consideration early in the decoding process. In the case of many paths with similar cost, the second criterion restricts the number of paths to N_{max}, which is important architecturally. At each stage, the minimum cost of the previous stage d_m, threshold T, and maximum survivors N_{max} are used to prune the number of surviving paths. Careful calculation of T and N_{max} is the key to effective use of the AVA algorithm. If threshold T is set to a small value, the average number of paths retained at each trellis stage will be reduced. This can result in an increased BER since the decision on the most likely path has to be taken from a reduced number of possible paths. Alternately, if a large value of T is selected, the average number of survivor paths increases and results in a reduced BER. As a result, increased decode accuracy comes at the expense of additional computation and a larger path storage memory. The maximum per-trellis stage number of survivor paths, N_{max}, has a similar effect on BER as T. As a result, an optimal value for T and N_{max} should be chosen so that BER is within allowable limits while matching the resource capabilities of the hardware. In previous work [8], we have experimentally determined appropriate values for T and N_{max} for a range of K values.

Several power-sensitive implementations of adaptive Viterbi architectures have been proposed. In Henning and Chakrabarti [9], a high-level architectural model of an adaptive Viterbi decoder is described. The threshold T and truncation length TL of the decoder is varied based on the desired BER, SNR, and code transmission rate. Although the authors mention potential power savings of up to 97% for their high-level architecture versus standard Viterbi decoders, a detailed hardware implementation of the approach is not described. The architecture does not take advantage of survivor path limits, N_{max}, or dynamic reconfiguration in determining potential power savings.

A second proposed high-level AVA implementation [10] uses a systolic architecture with a strongly-connected trellis. This architecture provides storage for up to 2^{K-1} paths, but only calculates and stores paths whose costs meet threshold T. Power savings are achieved through reduced storage and computation. Since a detailed discussion of a potential hardware implementation is not provided, it is not possible to evaluate the scalability and feasibility of the approach for other K values. An earlier discussion of the AVA architecture described in this manuscript was presented in [8]. Although the basic architecture

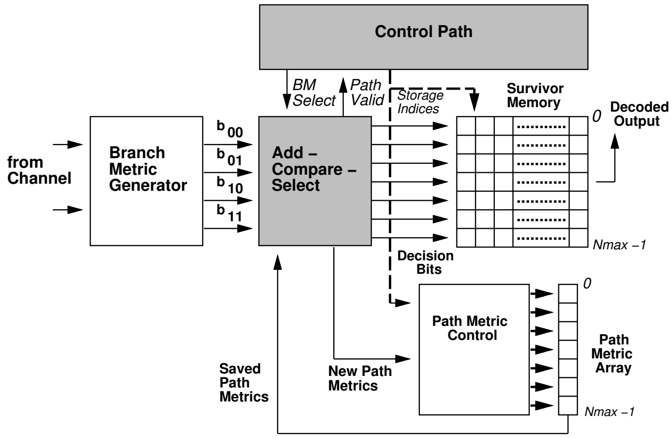

Fig. 1. Adaptive Viterbi decoder architecture

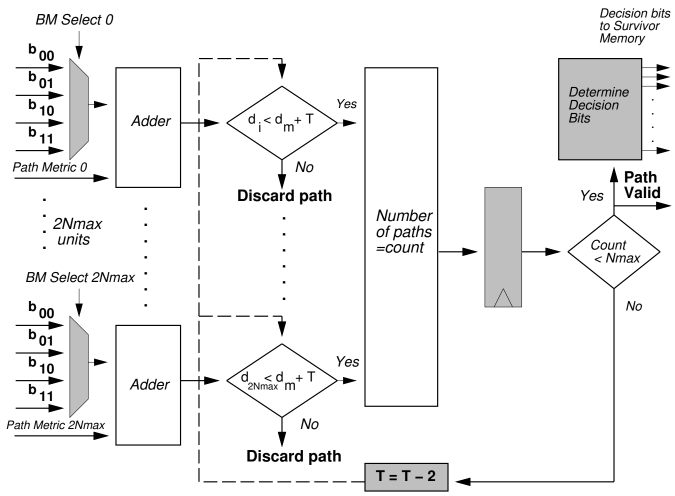

Fig. 2. ACS unit of adaptive Viterbi decoder

of the AVA decoder is the same, this earlier work focused on decoder performance improvement rather than power reduction. Unlike previous AVA approaches [2], the standard operation of eliminating the largest-metric path when two survivor paths enter the same trellis state was not implemented in the approach outlined in this manuscript due to hardware complexity. The implemented algorithm more closely resembles a predecessor of the AVA known as the Simmons T-algorithm [3].

III. AVA Architecture

To explore the power benefits of AVA use we have developed a hardware implementation of the algorithm. This architecture exhibits significant parallelism and supports dynamic reconfiguration to adapt decoder hardware to changing channel noise characteristics. Hardware reconfiguration provides the key mechanism to achieve decoder power savings.

A high-level view of the implemented adaptive Viterbi decoder architecture is shown in Figure 1. The decoder contains a datapath and an associated control path. Like most Viterbi decoders [11], the datapath is split into four parts: the branch metric generators (BMG), add-compare-select (ACS) units, the survivor memory unit, and path metric storage and control. A BMG unit determines distances between received and expected symbols. The ACS unit determines path costs and identifies lowest-cost paths. The survivor memory stores lowest cost bit-sequence paths based on decisions made by the ACS units and the path metric array holds per-state path metrics. The flow of data in the datapath and the storage of results is determined by the control path.

Two distinctive features of our decoder are the parallel computation of all ACS units and the per-symbol dynamic adjustment of T. In the implemented decoder, the expected symbol value ($BMselect$) is used to select the appropriate branch metric from the BMG, as shown at the left in Figure 2. This branch metric value is combined with the path metric of its parent present state to form a new path metric, d_i. At each trellis stage, the minimum-value surviving path metric among all path metrics for the preceding trellis stage, d_m, is computed. New path metrics are compared to the sum $d_m + T$ to identify path metrics with excessive cost. Comparators are then used to determine the life of each path based on the threshold, T. If the threshold condition is not satisfied by path metric $d_m + T$, the corresponding path is discarded.

Once the paths that meet the threshold condition are determined, the lowest-cost N_{max} paths are selected. Sorting circuitry is eliminated by allowing feedback adjustments to the parameter T for each received symbol. If the number of paths that survive the threshold is less than N_{max}, no iteration is required. As show in Figure 2, for stages when the number of paths surviving the threshold condition is greater than N_{max}, T is iteratively reduced by 2 for the current trellis stage until the number of paths surviving the threshold condition is equal to or less than N_{max}. The T value is reset to its original value prior to the processing of the next trellis stage. Appropriate values for T and N_{max} were determined in previous work [8], so that T reduction is needed infrequently (for less than 5% of symbols). The output of the ACS units includes *path valid* signals which indicate which of the 2 * N_{max} paths have survived pruning. Details regarding other decoder components can be found in [8].

Most communication systems desire links with predictable performance, which is usually specified by a fixed BER. Although desired decoder accuracy remains constant, channel signal-to-noise ratios can vary widely due to factors such as the propagation distance and the shadowing of the transmitted signal by large objects. In the presence of increased noise power (equivalently, a decreased SNR due to a weaker signal), a higher constraint length code is required to maintain a constant BER. As will be shown in Section V, AVA decoders for higher constraint-length codes require a larger amount of logic resources and consume more power than decoders for codes with smaller constraint lengths. If the encoder and decoder hardware can be reconfigured to exactly match the constraint length required at a specific time instant, power consumption can be minimized. In the presence of increased noise, a high-constraint length encoder and decoder (larger K) can be swapped in at the cost of increased power consumption. If the noise power is reduced, a low-constraint length encoder and associated lower-power AVA decoder can be used in its place. If swapping is not allowed, a high-constraint length decoder must always be used. Since channel noise statistics do not generally change instantaneously, reconfiguration based on channel noise statistics can be performed at a coarse timescale, once every few seconds.

IV. Experimental Approach

A. Test Platform

To test the practicality of our reconfigurable AVA architecture, a hardware implementation of the decoder was tested as part of a communication system. The communication system model used for experimentation is shown in Figure 3. The *Random Bit Generator* is a *C* module that generates a randomized bit sequence to model transmitted data. The *convolutional encoder*, can be parameterized to assorted constraint lengths. The *modulator* converts a coded bit to a real number: 0 -> 1, 1 -> -1 for the binary phase-shift keyed (BPSK) system

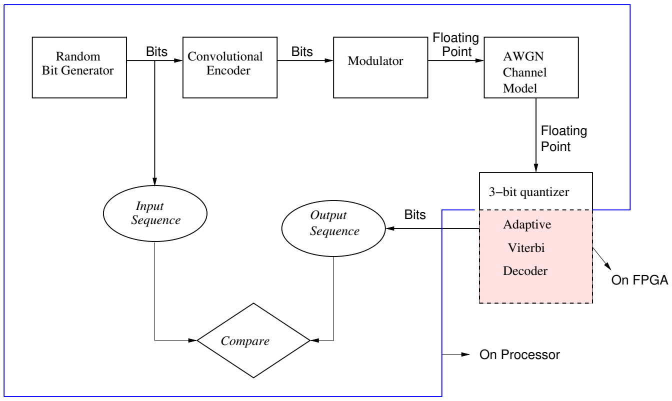

Fig. 3. System model

K	N_{max}	T	TL	XC4036XL-08		XCV1000-04	
				CLBs	FFs	CLB Slices	FFs
4	4	14	20	553	278	436	278
5	7	14	25	1194	540	922	540
6	8	18	30	1206	724	1260	724
7	8	17	35	1215	756	1401	756
8	8	17	40	1284	788	1443	788
9	9	18	45	1296	820	1469	819
10	21	20	50	NA	NA	3371	1911
11	25	23	55	NA	NA	3643	2137
12	25	23	60	NA	NA	3668	2170
14	41	24	70	NA	NA	6741	2446

TABLE I

FPGA RESOURCE UTILIZATION FOR THE ADAPTIVE VITERBI DECODER FOR BER OF 10^{-5}

employed. The output of the modulator is input to the *AWGN channel simulator*. This block simulates a noisy channel where Gaussian noise is added to the transmitted signal. The amount of noise depends on the signal-to-noise ratio preset by the user. The symbols obtained from the AWGN channel model are quantized before being sent to the *decoder* as its input. On receiving the input, the decoder attempts to recover the original sequence. All software modeling of the communication system was performed using a 366 MHz Celeron PC.

B. Hardware Implementation

The AVA decoder architecture was mapped to a Xilinx XC4036XL-08 FPGA located on an Annapolis Micro Systems WildOne board [12]. This mapping allowed for in-field testing of AVA designs for constraint lengths up to K=9. An RTL level description of the adaptive Viterbi decoder was written in VHDL that could be mapped to FPGA devices. The VHDL code was simulated using Cadence *Affirma* tools. All designs were synthesized using Synplicity *Synplify* and mapped to Xilinx hardware using Xilinx Foundation M2.1 tools with timing constraints. The operating frequencies of the FPGA were obtained from the Xilinx *TRACE* timing analyzer tool.

Power consumption values for the AVA decoders implemented in the XC4036XL were determined with the following equation from [12]:

$$Power = ((0.02 * f) + 0.09) * A * V \tag{1}$$

where f is the design frequency, A is the percentage of used device flip flops and I/Os multiplied by their switching activity, and V is the supply voltage. A switching activity of 30% [13] was used with a supply voltage of 5 V. To account for power consumption during XC4036XL reconfiguration, the power associated with reading the configuration bitstream from SDRAM and storing it in the FPGA was calculated. It was determined that approximately 5 mW of power are needed during reconfiguration to read the 832,480 XC4036XL configuration bits from 2Mx32 Micron MT48LC2M32B2 SDRAM [14]. This value was determined by scaling the specified maximum power dissipation at 200 MHz by the required 4 MHz FPGA configuration speed. The amount of power required to reconfigure the XC4036XL using the on-chip reconfiguration shift chain was determined by calculating the energy dissipated by a single chain shift in 0.35 μm technology with SPICE. This value was scaled by the required 832,480 shifts and divided by configuration time to calculate FPGA reconfiguration power. It was calculated that 1.9 mW are required to reprogram the configuration bits of the XC4036XL at 4 MHz. Total XC4036XL reconfiguration time is 40 ms [15].

To test larger AVA implementations, decoders with constraint lengths up to K=14 were mapped to a Xilinx XCV1000-04 FPGA. Although the XCV1000 designs were not physically implemented in hardware, cycle periods from TRACE were used in conjunction with cycle counts from HDL simulation to estimate decode speed. Power consumption values for the AVA decoders mapped to the XCV1000 were determined using the Xilinx *XPower* tool [16]. These values and a switching activity value of 30% were used by XPower to determine XCV1000 operational power for each AVA decoder. It was determined that approximately 62.5 mW of power are needed during reconfiguration to read the 6,127,744 XCV1000 configuration bits from 2Mx32 Micron MT48LC2M32B2 SDRAM [14]. This value was determined by scaling the specified maximum power dissipation at 200 MHz by the required 50 MHz FPGA configuration speed. The amount of power required to reconfigure the XCV1000 using the on-chip reconfiguration shift chain was determined by calculating the energy dissipated by a single chain shift in 0.22 μm technology with SPICE. This value was scaled by the required 6,127,744 shifts and divided by configuration time to calculate FPGA reconfiguration power. It was calculated that 27.4 mW are required to reprogram the configuration bits of the XCV1000 at 50 MHz. Total XCV1000 reconfiguration time is 15.3 ms [17].

V. EXPERIMENTAL RESULTS

A. FPGA Resource Usage and Non-Reconfigurable Performance

The logic resources used by the adaptive Viterbi decoder architecture described in Section III was measured in terms of logic block (CLB) usage. Table I summarizes the resource utilization of the adaptive Viterbi decoder on an XC4036XL for constraint lengths K = 4 to 9 and on an XCV1000 for constraint lengths K = 4 to 14. An adaptive Viterbi decoder with K=9 utilized 100% of XC4036 CLB resources (85% LUT utilization), while a K=14 AVA decoder fits in a single XCV1000 device (52% LUT utilization). For the AVA hardware, implementation size is affected by T via the comparison between path costs, d_i and the sum $d_m + T$. Optimum values of T range between 14 and 24, as determined in [8] and shown in Table I. This narrow range of variation in T does not substantially affect comparator size. Since the number of comparators is directly related to N_{max}, the impact of T is limited. If T is selected to be optimal, the number of surviving paths will almost always be close to N_{max}.

In most communication systems, both decode rate and maximum BER are fixed across all decoder constraint lengths. We have followed these guidelines in evaluating the decode rate for the XC4036-based decoders. As shown in Table II, due to an increase in required cycles per decoded bit and increased critical path length, *maximum* decode

	Decode Rate - 105.9 Kbps			Max Decode Rate	
K	XC4036XL clock (MHz)	SNR range (dB)	Power (mW)	XC4036XL (Kbps)	SA-1100 (Kbps)
4	12.9	6.3-6.5	45.7	333.7	22.2
5	13.0	6.1-6.3	56.2	164.2	17.4
6	13.0	5.5-6.1	79.8	162.3	10.3
7	13.0	3.9-5.5	125.1	160.8	9.4
8	13.0	3.7-3.9	130.4	143.6	8.7
9	13.0	3.1-3.7	135.7	141.1	7.0

TABLE II

PERFORMANCE AND POWER CONSUMPTION FOR A XC4036XL-08 AND A STRONGARM SA-1100 AT A BER OF 10^{-5}.

	Decode Rate - 61.7 Kbps			Max Decode Rate
K	XCV1000 clock (MHz)	SNR range (dB)	Power (mW)	XCV1000 (Kbps)
4	7.5	6.3-6.5	241	415.0
5	7.5	6.1-6.3	287	303.2
6	7.5	5.5-6.1	319	300.0
7	7.5	3.9-5.5	331	283.6
8	7.6	3.7-3.9	336	277.9
9	7.6	3.1-3.7	339	240.3
10	15.5	3.0-3.1	853	101.5
12	16.1	2.8-3.0	937	94.9
14	17.2	2.5-2.8	1611	82.3

TABLE III

PERFORMANCE AND POWER CONSUMPTION FOR A XCV1000-04 AT A BER OF 10^{-5}.

rate capability per decoder decreases with increasing decoder constraint length. As a result, for our analysis the *fixed* decode rate of all decoders is set to be 75% of the maximum decode rate of the K=9 decoder (105.9 Kbps). This 25% rate overhead is sufficient to account for non-infinite queueing of received samples and PCI-bus data transfer overhead. The FPGA clock rates required to achieve this fixed decode rate are shown in the second column of Table II. The SNR column in the table indicates the range of channel noise statistics over which a given decoder is the minimum constraint length decoder that achieves the target BER of 10^{-5}. In column 4 of Table II, for a fixed decode rate, decoder power consumption increases with constraint length due primarily to increased circuit size.

In comparison to an XC4036XL implementation, the *maximum* possible decode rate of a software AVA implementation on a 206 MHz StrongARM SA-1100 microprocessor is nearly 5× slower than the desired 105.9 Kbps rate (Table II). Additionally, SA-1100 power values, calculated with JouleTrack [18], range between 350 and 400 mW for the constraint lengths listed in Table II, more than 2× greater than the most power-hungry FPGA decoder. The SA-1100 is implemented in the same technology (0.35 μm) as the XC4036XL-08 and uses a lower supply voltage (1.5 V versus 3.3 V).

Table III summarizes the performance and power dissipated by AVA decoders in the XCV1000-04 FPGA for a fixed decode rate. In this experiment, the fixed decode rate is set to 61.7 Kbps, 75% of the maximum $K = 14$ FPGA decode rate. Like the XC4036XL decoders, power consumption increases with increased K, as BER and decode rate remains fixed. Maximum possible decode rates for each decoder are listed for reference.

B. Dynamic Reconfiguration

In the second set of experiments, channel noise, as indicated by SNR, was used to indicate when the encoder and decoder could be reconfigured to match a fixed BER of 10^{-5}. Power savings is achieved by using a lower constraint length encoder and, hence, lower constraint length and lower-power decoder for high SNR, and a higher constraint length encoder and, hence, higher constraint length and higher-power decoder for low SNR. In all evaluations, it is assumed that the current channel SNR is perfectly fed back from the receiver to the transmitter with zero delay. Such feedback of the path-loss and shadowing is commonly done in modern wireless communication systems [4].

Experiments requiring reconfiguration of the XC4036XL were performed by varying the SNR of transmitted data and reconfiguring the AVA hardware based on (K, N_{max}) values that were required to achieve the desired BER and decode rate. A set of 10,000 SNRs were generated using a log-normal shadowing distribution [1] for a total transmission length of 2.5 billion bits. Based on the assumption that SNR can be sampled successfully every 250,000 bits [1], FPGA hardware was periodically reconfigured during the transmission process. Channel SNR values were varied between 3.1 and 6.5 dB (requiring K values between 4 and 9) and AVA configurations based on Table II were chosen. The power consumption for a dynamically-reconfigured versus a static XC4036XL decoder for a fixed decode rate (105.9 Kbps) and BER (10^{-5}) appears in Table IV. For the generated set of SNRs, FPGA reconfiguration was performed 7065 out of 10,000 possible times leading to a total reconfiguration time of 282.6 seconds. To maintain a decode rate of 105.9 Kbps while taking into account the 282.6 seconds of decode inactivity, each individual decoder was run at a clock rate 2% higher than the value listed in Table II. For the static decoder case, a K=9 decoder must be used at all times to maintain the desired BER. The use of dynamic reconfiguration leads to a 27% reduction in power consumption over the duration of the decoding period.

The benefits of coarse-grained dynamic reconfiguration for a constraint length of K = 4 to 14 was considered by targeting an XCV1000-04 FPGA. The sequence of 2.5 billion bits applied to the XC4036XL was re-evaluated for the XCV1000. Channel SNR values were varied between 2.5 and 6.5 dB, requiring K values between 4 and 14 and AVA configurations from Table III. For the generated set of SNRs, FPGA reconfiguration was performed 7007 out of 10,000 possible times leading to a total reconfiguration time of 107.2 seconds. To maintain a decode rate of 61.7 Kbps while taking into account the 107.2 seconds of decode inactivity, each individual decoder was run at a clock rate 1% higher than the value listed in Table III. If reconfiguration is not used, a static, K=14, decoder must be used at all times to maintain the desired BER. The use of dynamic reconfiguration leads to a 69% reduction in power consumption over the duration of the decoding period. It is apparent that as a broader range of constraint lengths is considered, the amount of possible power savings due to dynamic reconfiguration increases.

VI. CONCLUSION AND FUTURE WORK

The use of error-correcting codes has proven to be an effective way to overcome data corruption in digital communication channels. In this manuscript, a power-efficient implementation of an adaptive Viterbi decoder has been described. To measure its power consumption, the AVA architecture has been implemented in two contemporary FPGA architectures for a range of constraint lengths. For a given, fixed bit-error and decode rate, power savings is achieved by adapting the constraint length of the convolutional code employed, with the goal of employing a lower-power decoder when allowable. The dynamically re-

	Avg. Speed (Kbps)	Decode time (sec)	Reconfigs. required (out of 10,000)	Reconfig. Overhead (sec)	Avg. Power (mW)
XC4036XL-08					
Static	105.9	23617	0	0	135.7
Dynamic	105.9	23617	7065	282.6	98.8
XCV1000-04					
Static	61.7	40521	0	0	1611.0
Dynamic	61.7	40521	7007	107.2	505.3

TABLE IV
STATIC DECODER VERSUS DYNAMICALLY-RECONFIGURABLE DECODER POWER CONSUMPTION

configurable FPGA implementation is shown to consume significantly less power than a static FPGA implementation.

In the future, we plan to consider the decoding benefits of using a hybrid microprocessor and FPGA device. The tight integration of sequential control with parallel decoding may provide further run-time power benefits.

VII. ACKNOWLEDGMENTS

This work was sponsored by National Science Foundation grants CCR-0081405, CCR-9988238, NCR-9714597 and CCR-9875482. The authors wish to thank Frank Honoré for providing the JouleTrack software.

REFERENCES

[1] J. Proakis, *Digital Communications*. New York, N.Y.: McGraw-Hill, 1995.
[2] F. Chan and D. Haccoun, "Adaptive Viterbi decoding of convolutional codes over memoryless channels," *IEEE Transaction on Communications*, vol. 45, no. 11, pp. 1389–1400, Nov. 1997.
[3] S. J. Simmons, "Breath-first trellis decoding with adaptive effort," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
[4] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, no. 1, pp. 54–64, Jan. 2000.
[5] D. Matolak and S. Wilson, "Variable-complexity trellis decoding of binary convolutional codes," *IEEE Transactions on Communications*, vol. 44, no. 2, pp. 121–126, Feb. 1996.
[6] S. Simmons, "An error bound for reduced-state Viterbi decoding of TCM codes," *IEEE Communications Letters*, vol. 3, no. 9, pp. 266–268, Sept. 1999.
[7] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Transactions on Communications*, vol. 32, no. 2, pp. 169–176, Feb. 1984.
[8] S. Swaminathan, R. Tessier, D. Goeckel, and W. Burleson, "A dynamically reconfigurable adaptive Viterbi decoder," in *Proceedings, ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 2002, pp. 227–236.
[9] R. Henning and C. Chakrabarti, "Low power approach to decoding convolutional codes with adaptive Viterbi algorithm approximations," in *Proceedings, IEEE/ACM International Symposium on Low Power Electronics and Design*, Monterey, CA, Aug. 2002, pp. 68–71.
[10] M. Guo, M. O. Ahmad, M. Swamy, and C. Wang, "An adaptive Viterbi algorithm based on strongly connected trellis decoding," in *Proceedings, IEEE International Symposium on Circuits and Systems*, Scottsdale, AZ, May 2002, pp. 137–140.
[11] G. Fettweis and H. Myer, "High-speed parallel Viterbi decoding: Algorithm and VLSI-architecture," *IEEE Communications Magazine*, vol. 29, no. 5, pp. 46–55, May 1991.
[12] *WILD-ONE Reference Manual*, Annapolis Microsystems, Inc., 1999.
[13] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in *Proceedings, ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, Ca., Feb. 2002, pp. 157–164.
[14] *MT48LC2M32B2 SDRAM Data Sheet*, Micron Technologies, Inc., 2003.
[15] *Xilinx XC4000 Data Sheet*, Xilinx Corporation, 2001, http://www.xilinx.com.
[16] *ISE Manual*, Xilinx Corporation, 2001, http://www.xilinx.com.
[17] *Xilinx Virtex Data Sheet*, Xilinx Corporation, 2001, http://www.xilinx.com.
[18] A. Sinha and A. Chandrakasan, "JouleTrack - a web based tool for software energy profiling," in *Proceedings, ACM/IEEE 35rd Design Automation Conference*, June 2001, pp. 220–225.