Transactions Briefs

Highly-Parallel Decoding Architectures for Convolutional Turbo Codes

Zhiyong He, Paul Fortier, and Sébastien Roy

***Abstract*—Highly parallel decoders for convolutional turbo codes have been studied by proposing two parallel decoding architectures and a design approach of parallel interleavers. To solve the memory conflict problem of extrinsic information in a parallel decoder, a block-like approach in which data is written row-by-row and READ diagonal-wise is proposed for designing collision-free parallel interleavers. Furthermore, a warm-up-free parallel sliding window architecture is proposed for long turbo codes to maximize the decoding speeds of parallel decoders. The proposed architecture increases decoding speed by 6%–34% at a cost of a storage increase of 1% for an eight-parallel decoder. For short turbo codes (e.g., length of 512 bits), a warm-up-free parallel window architecture is proposed to double the speed at the cost of a hardware increase of 12%.**

***Index Terms*—Decoder, interleaver, parallel architecture, turbo code.**

I. Introduction

Turbo codes [1] have received a lot of interest because of their excellent performance. To apply turbo codes in high-speed digital communications, such as in broadband wireless access based on the IEEE 802.16 standard supporting data rates of up to 70 Mb/s, and in fourth generation cellular systems, which are expected to provide a data rate ranging from 20 to 100 Mb/s for high mobility, high throughput of turbo codes is a critical issue. Since turbo decoders inherently have a large latency and low throughput due to the iterative decoding process, highly parallel decoding architectures are required to achieve speed-up of an order of magnitude.

This paper discusses several key problems in the implementation of highly parallel decoders. One of the most immediate problems is memory conflict during decoding [2], where several component decoders simultaneously get access to the same memory module when reading (writing) the extrinsic information from (into) the memory modules. To avoid collisions in accesses to the memory modules, this paper presents an efficient approach for designing collision-free parallel interleavers in which data is written row-by-row and read diagonal-wise.

The second problem in the implementation of a highly parallel decoder is concerned with limiting both the hardware cost and the decoding delay of the decoder when employing many component decoders. In order to reduce computational complexity, the maximum *a posteriori* probability (MAP) algorithm [3], which is used for decoding convolutional turbo codes is generally implemented in the logarithmic domain, resulting in the so-called log-MAP algorithm. In an M-parallel decoder based on the log-MAP algorithm, an information block is divided into M sub-blocks which are assigned to M component decoders. To initialize the path metrics in each sub-block, these component decoders typically have to perform a warm-up phase which consumes a few clock cycles at each iteration (e.g., [4]). However, the warm-up phase reduces the throughput substantially when the number of sub-blocks is large.

This paper proposes two warm-up-free parallel architectures for decoding short and long turbo codes. A complete analysis of the two architectures for various code lengths is given based on the 8-state turbo code specified by both the 3GPP standard [5] and the IEEE 802.16 standard.

This paper is organized as follows. Section II describes in detail the approach for designing highly parallel interleavers. Section III proposes warm-up-free sliding window (SW) architectures for highly parallel decoders and then compares BER performance and decoding speeds of the warm-up-free SW architectures with that of the conventional SW architecture. Section IV presents a warm-up-free parallel window architecture for implementing highly parallel decoders with short code lengths. Finally, Section V concludes this paper.

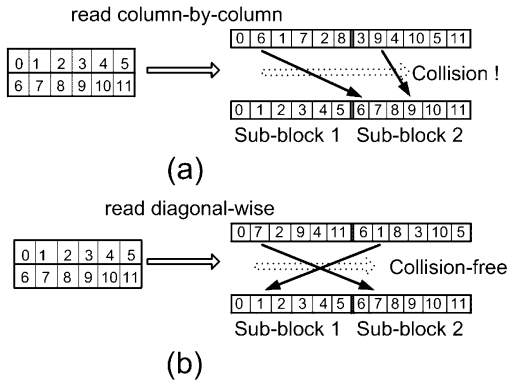

Fig. 1. (a) Interleaver in collision. (b) Collision-free interleaver.

II. Parallel Interleaver Design

A. Collisions in a Highly Parallel Decoder

To observe the problem of memory conflict in a parallel decoder, Fig. 1 illustrates an example for an interleaver having a block length of 12 bits. In a two-parallel turbo decoder, each block is divided into two sub-blocks, each consisting of 6 bits. Let 12 bits be written row-by-row into a 2×6 matrix. Fig. 1(a) shows the designed interleaver when the bits are read column-by-column from the matrix. The solid arrows which show the mapping of the second bits in two sub-blocks, highlight an obvious collision. Indeed, the component decoders have to WRITE the data into the same storage module after decoding the second bits.

Collisions in a highly parallel decoder are further analyzed by using two commonly-used approaches to the design of interleavers for turbo codes, the so-called S-random interleaver (e.g., [6]) and the interleaver specified in the 3GPP standard [5]. Let the input signals

Manuscript received October 28, 2005; revised December 8, 2005 and May 10, 2006. The material in this paper was presented in part at the IEEE International Symposium on Circuits and Systems, Kobe, Japan, May 23–26, 2005, and the Canadian Workshop on Information Theory, Montreal, Canada, June 6–8, 2005. This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), by Le Fonds québécois de la recherche sur la nature et les technologies (FQRNT), and by the Canadian Microelectronics Corporation (CMC) under its System-on-Chip Research Network (SOCRN).

The authors are with the Department of Electrical and Computer Engineering, Laval University, Quebec City, QC G1K 7P4, Canada.

Digital Object Identifier 10.1109/TVLSI.2006.884172

and the extrinsic information be stored in a series of modules in which each module has 32 addresses for storage. The number of collisions is defined to be $m-1$ when m component decoders attempt access simultaneously to a given memory module. When the S-random interleaver with a block length of 1024 is used, the number of collisions increases from 45 for a four-parallel decoder to 514 for a 32-parallel decoder. On the other hand, the number of collisions increases from 67 for a four-parallel decoder to 424 for a 32-parallel decoder when the 3GPP interleaver with a block length of 1024 is used. To avoid collisions, collision-free management of these modules can be performed by adding small buffers to assist memory accesses [7]. This approach is adequate for low-degree parallel decoders, e.g., $M \leq 4$. For highly-parallel decoders, a collision-free parallel interleaver is needed.

B. High-Performance Design of Collision-Free Interleaver

The basic principle behind the design of collision-free parallel interleavers is illustrated in Fig. 1(b) which shows a collision-free interleaver with a block length of 12 bits. By writing row-by-row and reading diagonal-wise, the data at a given position relative to the beginning of a sub-block is mapped to the same position within another sub-block. The solid arrows in Fig. 1(b) show a collision-free mapping of the second bits in two sub-blocks. The collision-free approach shown in Fig. 1(b) by reading diagonal-wise, does not optimize interleaver spread which guarantees a good performance of turbo code. To maximize the overall weight of a code word, the interleaver of a turbo code should associate a small weight sequence from one component encoder with a large weight sequence from the other component encoder, i.e., the interleaver should make the duo-distance, defined as [8]

$$d(n_1, n_2) \equiv |n_1 - n_2| + |\pi(n_1) - \pi(n_2)| \tag{1}$$

between position n_1 and position n_2 as large as possible, where $\pi(n_1)$ and $\pi(n_2)$ are the interleaved positions of position n_1 and position n_2. Then, the spread associated with position n_1 is

$$D(n_1) = \min_{n2, n2 \neq n1} [d(n_1, n_2]. \tag{2}$$

Likewise, the overall spread is defined as

$$D = \min_{n1} [D(n_1)]. \tag{3}$$

Based on the previous discussion, we propose a two-level mapping approach for designing high-performance and collision-free parallel interleavers. Let an M-parallel decoder be used to decode an information block of length $M \times W \equiv N$, where each component decoder decodes W bits. After N information bits are written row-by-row into an $M \times W$ matrix, the two-level mapping process is performed in three steps described as follows.

Step 1) Bottom-Level S-Random Mapping: In order to increase the randomness and the spread of the designed interleaver, the S-random approach is used to interleave the M sub-blocks and interleave W bits within each sub-block.

Step 2) Top-Level Collision-Free Mapping: Assume that the ith information bit in the jth sub-block is mapped to the i'th information bit in the j'th sub-block, i.e., the information bit at position $n \equiv j \times W + i$ is mapped to position $n' \equiv \pi(n) \equiv j' \times W + i'$. The top-level mapping functions are expressed as

$$i' = \pi_W(i) \tag{4}$$

$$j' = \pi_M([i + j]_M) \tag{5}$$

where π_W and π_M are the bottom-level mappings obtained in the first step for the W information bits and the M sub-blocks, respectively. $[x]_m$ denotes x modulo-m arithmetic.

Step 3) Inter-Sub-Block Collision-Free Swapping: Calculate the spread $D(n)$ associated with position n ($n = 0, 1, 2, \ldots, N-1$) and the overall spread D according to (2) and (3). Then perform inter-sub-block swapping for all positions n with $D(n) \leq (D+1)$ as follows.

Let the spread of the ith information bit in the jth sub-block be of a value less than $(D+1)$, i.e., $D(n) \leq (D+1)$ and $n \equiv j \times W + i$. Swap this bit with the ith information bit in the j'th sub-block, where $j' = 0, 1, \ldots, M-1$ and $j' \neq j$. After swapping, the overall spread D is calculated according to (3). If the value of D is decreased, this swapping is rejected.

The inter-sub-block swapping is repeated several hundred times in order to obtain a high spread. Since the swapping process is much faster than the selection process in the S-random approach, the proposed approach designs a collision-free parallel interleaver with a given value of D within a considerably smaller time than the dividable interleaver [9] in which the S-approach was employed to design a parallel interleaver by adding collision-free constraint.

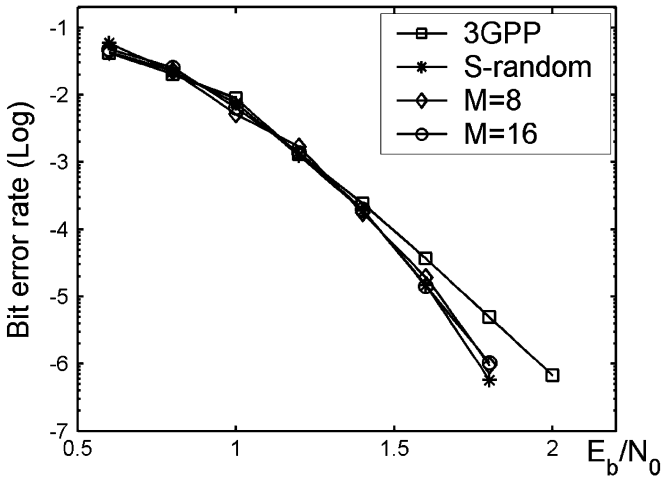

Fig. 2. BER at the tenth iteration for turbo code with a code rate of 1/2 and a block length of 1024 bit. Four interleavers are compared.

C. Bit Error Rate (BER) Performance Analysis

We performed simulations for the eight-state turbo code described in the 3GPP standard for a code rate of 1/2 and an interleaver length of 1024, assuming an AWGN channel and BPSK modulation. The BER at the tenth iteration versus the signal-to-noise ratio (SNR) per bit E_b/N_0 are compared in Fig. 2 for the S-random interleaver, the 3GPP interleaver, the eight-parallel interleaver ($M = 8$) and the 16-parallel interleaver ($M = 16$), respectively. The performances of the proposed collision-free parallel interleavers are competitive with the collision-prone S-random interleaver. Moreover, they slightly outperform the interleaver specified by the 3GPP standard.

III. Parallel Decoding Architectures With Sliding Window Approach

Based on the principle that the Viterbi algorithm can start cold in any state at any time, the sliding window (SW) approach has been proposed to reduce the storage requirements of the path metrics (e.g., [10]). The SW approach consists of dividing a block into several SWs and processing each SW sequentially by using three path processors for the preliminary backward path, the forward path and the backward path

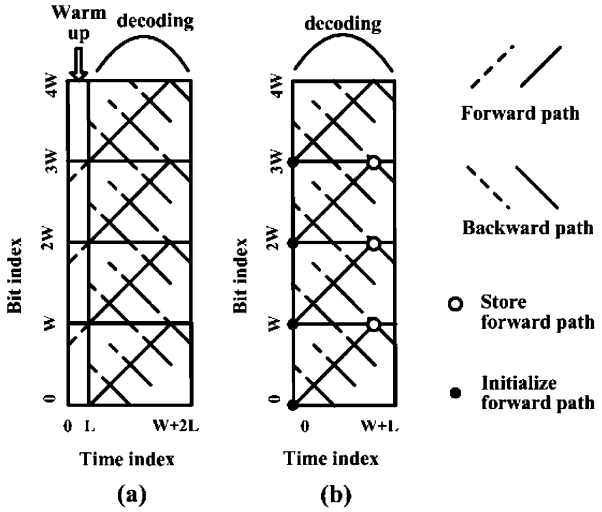

Fig. 3. Parallel decoding architectures with sliding window approach: (a) with warm-up process and (b) without warm-up process.

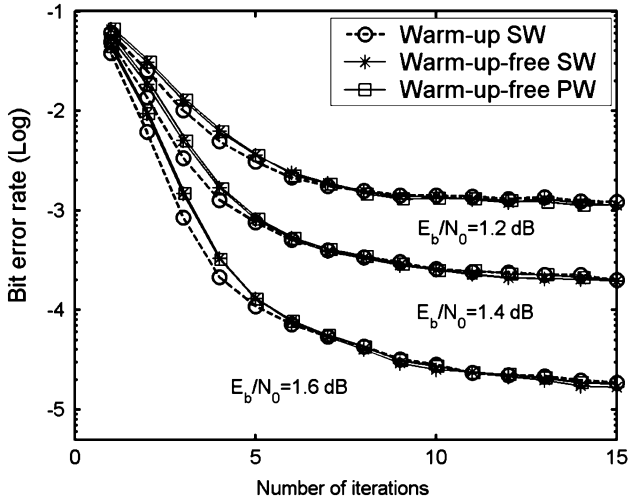

Fig. 4. BER performances of eight-parallel decoders ($M = 8$) for turbo codes with a block length of 1024 and a code rate of 1/2.

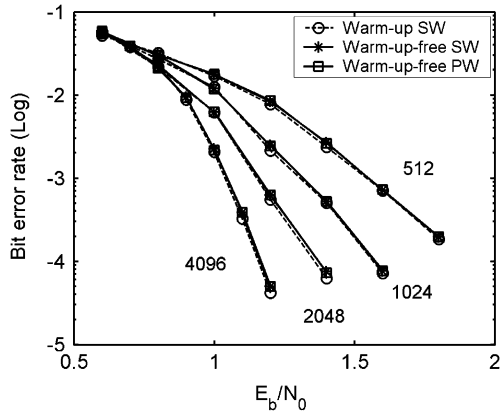

Fig. 5. BER performances of eight-parallel decoders ($M = 8$) for turbo codes with a code rate of 1/2, and block lengths of 512, 1024, 2048, and 4096 bits.

computations, respectively. By storing the backward path metrics of the previous iteration to avoid the preliminary backward path computation, only two processors are required [11]. In this section, we discuss parallel SW architectures. Since the SW approach with two path processors proposed in [11], increases dramatically the storage requirement for long turbo codes, the SW approach with three path processors, is favored throughout the section.

A. Warm-Up-Free Parallel SW Architecture

First, the conventional SW approach often used in the literature (e.g., [4]) is briefly reviewed. Shown in Fig. 3(a), an information block of length $4W$ is divided into 4 sub-blocks, where each sub-block consists of W information bits. Then, each sub-block is divided into three SWs for SW processing. In order to initialize the boundary distributions of the path metrics for each sub-block, an overlapping of L information bits between adjacent sub-blocks is arranged for the warm-up phase of decoding. The length L for the warm-up phase is normally chosen to be six times the constraint length of the code.

Since the length W of each sub-block decreases linearly with an increasing number of sub-blocks, the warm-up process represents a large portion of the decoding delay in a high-speed parallel implementation. Instead of the warm-up process using two overlapped sliding windows, a warm-up-free parallel SW architecture is proposed by using the forward path metrics that were computed in the previous iteration of the adjacent sub-block to initialize the path metrics for each sub-block in the next iteration. Fig. 3(b) shows a four-parallel decoder with a warm-up-free architecture. Shown by open circles in Fig. 3(b), the forward path metrics of the last information bit in sub-block i were stored for initialization of the forward path metrics of sub-block $(i+1)$ in the next iteration. In the first iteration when there is no forward path metrics from the previous iteration to use, the forward path metrics are initialized to zero.

To analyze the performance of the proposed warm-up-free architecture, we performed simulations by assuming binary phase-shift keying (BPSK) modulation and an AWGN channel. The BER of three eight-parallel decoders versus iterations are compared in Fig. 4 for the turbo code with a block length of 1024 and a code rate of 1/2, where the so-called warm-up-free parallel window (PW) architecture will be discussed in the next section. Since the forward path metrics are initialized to zero in the first iteration, the performances of the warm-up-free architectures degrade slightly for the first five iterations, i.e., at a given iteration, warm-up architectures outperform slightly warm-up-free architectures. No difference between warm-up and warm-up-free architectures is observed after five iterations.

The BER of three parallel decoders at the sixth iteration are compared in Fig. 5 for turbo codes with a code rate of 1/2 and code lengths of 512, 1024, 2048, and 4096. It is shown clearly that the warm-up-free architectures do not affect the BER performance of the decoder.

B. Tradeoffs Between Speeds and Hardware Costs

To compare decoding speeds and hardware costs between warm-up and warm-up-free parallel architectures, we implemented a series of parallel decoders for the eight-state turbo code specified by the 3GPP standard into Xilinx FPGA Spartan-3 device. The vector lengths of each quantity within the algorithm were empirically optimized to minimize complexity while maintaining close to optimal performance, e.g., the path metric was represented by a bit vector of 10 bits by employing modulo normalization [12], and the extrinsic information was represented by a vector of 7 bits. Both the length of the warm-up window and that of the sliding window were taken to be 32, i.e., $L = 32$ in Fig. 3.

The decoding speeds of the parallel decoders with warm-up and warm-up-free SW architectures are compared in Fig. 6 as a function of the block size, where the decoding speed was calculated by using a decoding clock frequency of 100 MHz and six decoding iterations. Few

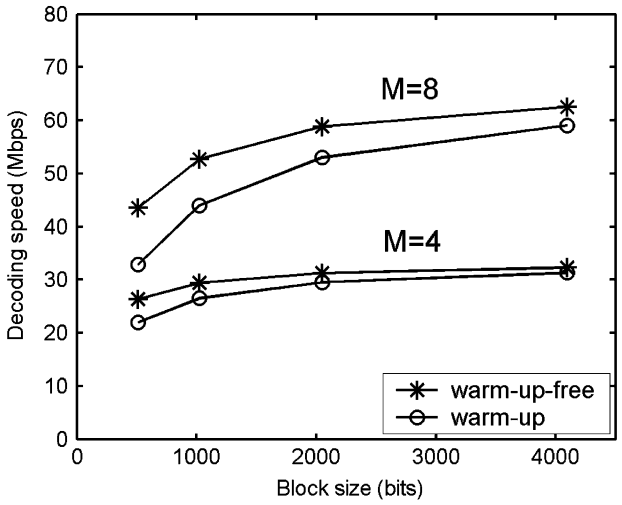

Fig. 6. Decoding speeds of four-parallel and eight-parallel decoders with warm-up and warm-up-free architectures.

TABLE I
TOTAL MEMORY COSTS IN KILOBITS FOR EIGHT-PARALLEL DECODERS WITH SLIDING-WINDOW (SW) AND PARALLEL-WINDOW (PW) ARCHITECTURES

Block length (bits)	256	512	1024	2048	4096
Warm-up SW	27.5	33.9	46.7	72.3	123.5
Warm-up-free SW	28.1	34.5	47.3	72.9	124.1
Warm-up-free PW	16.8	33.4	66.7	133.3	266.3

differences between the decoding speeds of the warm-up and warm-up-free architectures are observed for the four-parallel decoders for the long turbo codes of more than 1000 bits. However, for the highly parallel decoders with $M = 8$, the proposed warm-up-free architecture increases the speed by 6%–34% compared to the warm-up architecture.

The total memory costs for storing the input signals, the extrinsic information and the path metrics in the eight-parallel decoders with parallel SW architectures are listed in Table I. The warm-up-free SW architecture increases the memory requirement by only 1% compared to the warm-up SW architecture.

IV. WARM-UP-FREE PARALLEL WINDOW ARCHITECTURE

First considered in 1993 [13] and later modified in [14] to minimize the storage of the path metrics, the parallel window (PW) architecture consists of dividing a block into several windows and processing these windows in parallel. Since the memories for storing the path metrics in the PW architecture are proportional to the block length, there are few real applications of this architecture for medium or long turbo codes. In this section, an optimal version, the so-called warm-up-free PW architecture, is proposed for implementing a highly-parallel decoder for short turbo codes. The proposed architecture increases dramatically decoding speeds at the cost of a little increase in hardware.

Fig. 7 gives an example of a two-parallel decoding scheme with a warm-up-free PW architecture. An information block of length $2W$ is divided into two windows, where each window consists of W information bits. Just like the warm-up-free SW architecture, the proposed PW architecture uses the path metrics that were computed in the previous iteration to initialize the path metrics for each window in the next

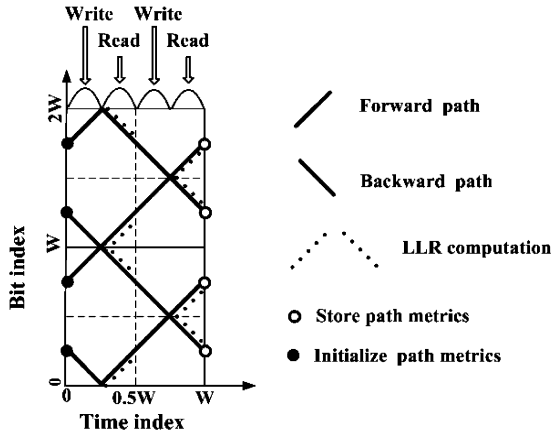

Fig. 7. Warm-up-free parallel window architecture.

TABLE II
APPLICATION FOR TWO WARM-UP-FREE ARCHITECTURES

Parallel architecture	SW	PW
Sub-block length (bits)	≥ 64	≤ 64

iteration. During the time periods from $t = 0$ to $t = W/4$ and from $t = W/2$ to $t = 3W/4$, the path metrics of $W/4$ bits are computed and stored in memory. From $t = W/4$ to $t = W/2$ and from $t = 3W/4$ to $t = W$, the stored path metrics are READ and soft outputs are computed. Since the memory is used twice, total memory requirements are halved. It is interesting to note that the forward and backward path recursions start and stop at the same bits. Thus, the forward and backward path metrics of the bits that are computed last in each window are stored to initialize the path metrics in the next iteration. For an M-parallel decoder, $(M-1)$ sets of forward path metrics and $(M-1)$ sets of backward path metrics are stored. The simulation results shown in Figs. 4 and 5 indicate that no difference between the BER performances of the warm-up-free parallel SW architecture and the warm-up-free PW architectures is observed.

Compared to the parallel SW architecture shown in Fig. 3(b), the main benefit of the warm-up-free PW architecture is that no extra path processor is needed for computing the preliminary backward path. However, two processors for computing LLR outputs have to be used for the PW architecture. By implementing a series of eight-parallel decoders with PW architecture into Xilinx FPGA Spartan-3 device, we observed that the proposed PW architecture increases the logic resource usage by 12% with respect to the parallel SW architecture.

Since no preliminary computation is required for the forward and backward path metrics, the warm-up-free PW architecture provides a dramatic speed increase. For an eight-parallel decoder with a block length of 512, the decoding speed is doubled from 33 Mb/s with a warm-up parallel SW architecture to 61 Mb/s with a warm-up-free PW architecture, assuming a decoding clock frequency of 100 MHz and six decoding iterations. The decoding latency is decreased from 15.5 to 8.4 μs.

Considering the tradeoff between hardware cost and decoding speed, the warm-up-free PW architecture is a good candidate suitable for decoding short turbo codes, because of high decoding speed and low decoding latency. Since the total storage requirements in the warm-up-free PW architecture listed in Table I increase with block length, the warm-up-free parallel SW architecture is better suitable for decoding long turbo codes. Table II summarizes the application of two warm-up-free architectures for various sub-block lengths.

V. Conclusion

We have proposed a novel approach, in which data is written row-by-row into a matrix and read diagonal-wise, for designing collision-free parallel interleavers. To improve the performance of the designed interleaver, a random mapping and swapping scheme has been used to augment the spread distance of the interleaver. The proposed collision-free parallel interleavers are competitive with the collision-prone S-random interleaver and slightly outperform the interleaver specified by the 3GPP standard. To minimize the decoding delay in a highly-parallel decoder, two warm-up-free parallel architectures, the parallel SW architecture, and the PW architecture, have been proposed for long and short turbo codes, respectively. Compared to the warm-up parallel SW architecture, the proposed warm-up-free parallel SW architecture increases the speed by 6%–34% at a cost of a hardware increase of 1% for an 8-parallel decoder, while the proposed warm-up-free PW architecture doubles the speed at a cost of hardware increase of 12%.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. Int. Conf. Commun. (ICC)*, 1993, pp. 1064–1070.

[2] A. Giulietti, L. van der Perre, and A. Strum, "Parallel turbo coding interleavers: Avoiding collisions in accesses to storage elements," *Electron. Lett.*, vol. 38, no. 5, pp. 232–234, Feb. 2002.

[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. IT-2, pp. 284–287, Mar. 1974.

[4] Z. Wang, Z. Chi, and K. K. Parhi, "Area-efficient high-speed decoding schemes for turbo decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 6, pp. 902–912, Dec. 2002.

[5] *3GPP2 (3rd Generation Partnership Project 2) Specifications*, [Online]. Available: http://3gpp2.com/

[6] S. Dolinar and D. Divsalar, "Weight distributions for Turbo codes using random and nonrandom permutations," *Telecommun. Data Acquisition (TDA) Progress Rep.*, vol. 42, no. 122, pp. 56–65, Aug. 15, 1995.

[7] Z. Wang, Y. Tang, and Y. Wang, "Low hardware complexity parallel turbo decoder architecture," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2003, pp. 53–56.

[8] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *Proc. IEEE 54th Veh. Technol. Conf.*, 2001, pp. 2394–2398.

[9] J. Kwak and K. Lee, "Design of dividable interleaver for parallel decoding in turbo codes," *Electron. Lett.*, vol. 38, no. 22, pp. 1362–1364, Oct. 2002.

[10] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 260–264, Feb. 1998.

[11] F. Raouafi, A. Dingninou, and C. Berrou, "Saving memory in turbo-decoders using the max-log-MAP algorithm," *IEE Colloq. Turbo Codes Dig. Broadcasting—Could It Double Capacity?*, pp. 14/1–14/4, 1999, (Ref. No. 1999/165).

[12] Y. Wu, B. D. Woerner, and T. K. Blankenship, "Data width requirements in SISO decoding with modulo normalization," *IEEE Trans. Commun.*, vol. 49, no. 11, pp. 1861–1868, Nov. 2001.

[13] H. Dawid, G. Gehnen, and H. Meyr, "MAP channel decoding: Algorithm and VLSI architecture," in *Proc. Workshop VLSI Signal Process., VI*, 1993, pp. 141–149.

[14] A. Worm, H. Lamm, and N. Wehn, "A high-speed MAP architecture with optimized memory size and power consumption," in *Proc. IEEE Workshop Signal Process. Syst.*, 2000, pp. 265–274.

DTMOS Technique for Low-Voltage Analog Circuits

Mohammad Maymandi-Nejad and Manoj Sachdev

***Abstract*—In this paper, the application of dynamic threshold MOS (DTMOS) technique for low-voltage analog circuits is explored. The body terminal of PMOS transistors in bulk CMOS technology can be used as the forth terminal to enhance the performance of low-voltage analog circuits. To show the effectiveness of this technique, we have designed a continuous time common mode feedback (CMFB) circuit for a sub 1-V opamp and a new sub 1-V, 1-bit quantizer. A 0.8-V opamp with embedded CMFB and a 0.8-V, 1-bit quantizer for low-voltage $\Delta\Sigma$ modulators are implemented in 0.18-μm CMOS technology. The simulation results as well as the measurement data of these blocks are presented in this paper.**

***Index Terms*—1-bit quantizer, CMOS analog circuits, comparator, $\Delta\Sigma$ modulator, dynamic threshold MOS (DTMOS), operational amplifier.**

I. Introduction

THE continuous trend toward smaller feature size for transistors in the CMOS technology, demands for lower supply voltages [1]. This is due to the very thin gate oxide in advanced technologies. In addition, in many applications such as implantable biomedical devices, hearing aids [2], etc., the circuit should be operated with a miniature battery. Often, commercial miniature batteries provide a voltage in the range of 0.9 to 1.5 V with limited energy capacity [3]. In such applications, the volume and the weight of the battery is one of the primary concerns. These concerns force analog circuit designers to look for low-voltage, low-power circuit architectures and techniques. Reducing the supply voltage in analog circuits is not trivial compared to digital circuits. In particular, supply voltage reduction in analog circuits reduces its dynamic range which in turn, degrades the signal–to–noise ratio (SNR) of signals. Moreover, as CMOS technology scales down the output resistance of MOS transistors is reduced. As a consequence, the maximum achievable gain from a MOS amplifier is reduced. Similarly, reduced output resistance makes the design of supply independent biasing network a challenging task. As a result, analog designers must continuously find low-voltage circuit techniques in order to be consistent with technology trends [4], [5]. In this context, the dynamic threshold MOS (DTMOS) technique, which was originally used in digital circuits, has the potential to enhance the performance of a low-voltage analog circuit [6].

In this paper, we show how the DTMOS technique can be helpful in the design of low-voltage analog circuit blocks. We designed a low-voltage, fully differential amplifier with a common mode feedback (CMFB) circuit and a low-voltage comparator, incorporating the DTMOS technique in bulk CMOS technology [7], [8]. In the case of CMFB, the DTMOS technique helps in reducing the circuit complexity while not consuming extra power. Similarly, in the case of the comparator, the DTMOS technique makes it possible to get a rail-to-rail input range.

This paper is organized as follows. In Section II, an overview of the DTMOS technique is given. The common mode feedback circuit and

Manuscript received September 22, 2005.

M. Maymandi-Nejad is with the Electrical Engineering Department, Ferdowsi University of Mashhad, Mashhad 9177948944, Iran (e-mail: maymandi@um.ac.ir).

M. Sachdev is with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo N2L 3G1, ON, Canada (e-mail: msachdev@ece.uwaterloo.ca).

Digital Object Identifier 10.1109/TVLSI.2006.884174