Area-Efficient High-Throughput MAP Decoder Architectures

Seok-Jun Lee, *Member, IEEE*, Naresh R. Shanbhag, *Senior Member, IEEE*, and Andrew C. Singer, *Senior Member, IEEE*

***Abstract*—Iterative decoders such as turbo decoders have become integral components of modern broadband communication systems because of their ability to provide substantial coding gains. A key computational kernel in iterative decoders is the maximum *a posteriori* probability (MAP) decoder. The MAP decoder is recursive and complex, which makes high-speed implementations extremely difficult to realize. In this paper, we present *block-interleaved pipelining* (BIP) as a new high-throughput technique for MAP decoders. An area-efficient symbol-based BIP MAP decoder architecture is proposed by combining BIP with the well-known look-ahead computation. These architectures are compared with conventional parallel architectures in terms of speed-up, memory and logic complexity, and area. Compared to the parallel architecture, the BIP architecture provides the same speed-up with a reduction in logic complexity by a factor of M, where M is the level of parallelism. The symbol-based architecture provides a speed-up in the range from 1 to 2 with a logic complexity that grows exponentially with M and a state metric storage requirement that is reduced by a factor of M as compared to a parallel architecture. The symbol-based BIP architecture provides speed-up in the range M to $2M$ with an exponentially higher logic complexity and a reduced memory complexity compared to a parallel architecture. These high-throughput architectures are synthesized in a 2.5-V 0.25-μm CMOS standard cell library and post-layout simulations are conducted. For turbo decoder applications, we find that the BIP architecture provides a throughput gain of 1.96 at the cost of 63% area overhead. For turbo equalizer applications, the symbol-based BIP architecture enables us to achieve a throughput gain of 1.79 with an area savings of 25%.**

***Index Terms*—Area efficient, block-interleaved pipelining, high throughput, parallel processing, pipeline, symbol-based decoding, turbo decoder, turbo equalizer.**

I. Introduction

THE TURBO principle first introduced in [1] has been adopted in a variety of communication systems. It has been shown that turbo decoding schemes for decoding parallel or serial concatenated convolutional codes [2] and low-density parity check codes [3] approach Shannon's limit for additive white Gaussian noise (AWGN) channels. A soft-input soft-output (SISO) equalizer can be combined with a SISO decoder to form a turbo equalizer [4]. In the presence of intersymbol interference (ISI) caused by frequency selective channels and high-density read channels, a turbo equalizer provides additional coding gain beyond that obtained from separate equalization and decoding [4]–[8]. Turbo multi-user detection schemes have also been presented to combat multiple access interference [9]. These turbo decoding algorithms have shown remarkable receiver performance improvements over hard decision decoding or separate equalization and decoding via soft information exchange between two independent SISO decoding blocks. This performance enhancement has lead turbo codes to be accepted as the coding technique in several standards such as Wideband CDMA (WCDMA), the 3rd Generation Partnership Project (3GPP) for IMT-2000 [10], Consultative Committee for Space Applications (CCSDS) telemetry channel coding [11], UMTS [12], DVB-RCS [13], and IEEE 802.16ab.

Turbo decoders are composed of two or more constituent SISO decoders, which correspond to the component codes employed in the transmitter, and an interconnection of these constituent decoders through an interleaver/deinterleaver. The decoding algorithm employed in the constituent decoders is the maximum *a posteriori* probability (MAP) algorithm. The MAP algorithm provides a reliability metric, known as the log-likelihood ratio (LLR), on the transmitted code symbols. The LLR output is employed by other constituent decoders, which attempt to improve their LLR estimates iteratively. However, the use of iterative processing results in a large computational and storage complexity and hence high power dissipation in the receiver. Therefore, low-power and high-throughput implementations for turbo decoders have recently been investigated [14]–[38] for wireless and broadband applications.

Several high-throughput VLSI architectures of turbo decoders have already appeared [25]–[37]. These high-throughput architectures can be classified into parallel processing [25]–[33], look-ahead computation [34]–[36], and algorithm reformulation approaches [37]. Since the key algorithm employed in the constituent SISO processing blocks of turbo receivers (turbo decoders and equalizers) is the MAP algorithm, most research works have so far focused on high-throughput MAP decoder architectures which directly lead to high-throughput turbo receiver implementations. However, the recursive nature of the ACS kernel in MAP decoding algorithms has limited the reduction of the critical path delay. Hence, parallel processing [25]–[32] and look-ahead computation architectures [34]–[36] are viewed as practical approaches for high-throughput implementations.

Manuscript received July 17, 2003; revised June 2, 2004. This material is based upon work supported by the National Science Foundation under Grant CCR 99-79381 and ITR 00-85929.

S.-J. Lee is with the Communication Systems Laboratory, DSP Solution R&D Center, Texas Instruments Incorporated, Dallas, TX 75243 USA (e-mail: s-lee8@ti.com).

N. R. Shanbhag and A. C. Singer are with the Coordinated Science Laboratory, Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: shanbhag@uiuc.edu; acsinger@uiuc.edu).

Digital Object Identifier 10.1109/TVLSI.2005.853604

In this paper, we propose a new critical path reduction technique referred to as *block-interleaved pipelining* (BIP), which leads to a pipelined ACS kernel with less area overhead compared to conventional high-throughput architectures. Note that the proposed idea of BIP can be applied along with the above mentioned high-throughput techniques [25]–[28], [34]–[37]. We also propose a symbol-based MAP decoding architecture, where multiple bits are processed as a symbol. The symbol-based MAP architecture is derived by applying the look-ahead transform [39], [40] to the ACS recursion loop. In the context of a turbo equalizer, the symbol-based decoder architecture enables us to fold the operations of the SISO MAP equalizer and SISO MAP decoder on to the same hardware platform leading to savings in silicon area. We show that the throughput of the symbol-based decoder architecture can be improved further by applying the BIP technique. Moreover, we also provide a detailed comparison of various high-throughput VLSI architectures in terms of their storage requirements, logic complexity, throughput gain, and silicon area. These analyses are validated via physical design examples for turbo decoders of parallel concatenated convolutional codes (PCCC) and turbo equalizers.

The rest of this paper is organized as follows. Section II describes the algorithm and VLSI architecture of SISO MAP decoding and explains the *warm-up* property which is exploited in both parallel and BIP architectures. In Section III, the proposed BIP transform is presented along with a brief summary of other architectural transforms that result in high-throughput implementations of recursive algorithms. In Section IV, we apply the techniques presented in Section III, to develop several architectural solutions for high-throughput MAP decoding. Section V describes an area-efficient high-throughput turbo decoder for PCCC and turbo equalizer implementations.

II. MAP Decoding

In this section, the algorithm for the SISO MAP decoder is explained first, followed by a description of a baseline VLSI architecture for a SISO MAP decoder.

A. Algorithm

Consider an (n_o, k_o, K) convolutional code. An encoder is a finite state machine with $K-1$ memory elements that encodes a k_0-bit data symbol into an n_0-bit code symbol as shown in Fig. 1(a). The parameter K is called the constraint length of the code, and the code rate is defined as $R = (k_0/n_0)$. Given a current state (s_k) and an input symbol (u_k), there exists a unique pair consisting of a next state variable (s_{k+1}) and an output symbol (c_k). Hence, for an input block of N data symbols $\mathbf{u} \triangleq (u_0, u_1, \ldots, u_{N-1})$, the encoder starts from an initial state s_0 at time $k = 0$ and performs N state transitions while generating N coded output symbols $\mathbf{c} \triangleq (c_0, c_1, \ldots, c_{N-1})$. An efficient method describing all possible state transitions is a trellis diagram. A section of the trellis of Fig. 1(a) is depicted in Fig. 1(c), where solid edges correspond to $u_k = 1$ and dashed edges correspond to $u_k = 0$.

Note that a discrete-time channel model [see Fig. 1(b)] can be considered as a (1,1,3) convolutional encoder, where the coded

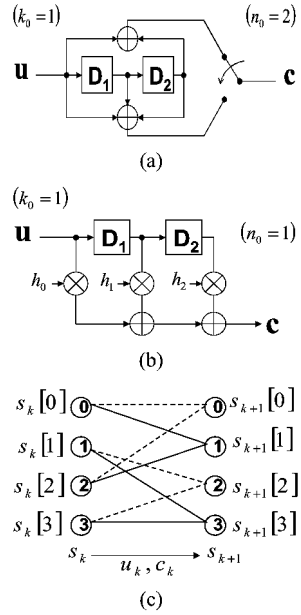


Fig. 1. Various encoder forms for MAP decoding. (a) A (2,1,3) convolutional encoder with two memory elements and modulo 2 adders. (b) A discrete time channel model with two memory elements and the channel vector $\mathbf{h} = [h_0, h_1, h_2]$. (c) A trellis section resulting from state transitions of the encoders in (a) and (b).

symbol is computed as $c_k = \sum_{n=0}^{2} h_n u_{k-n}$. Hence, the discrete-time channel causing ISI can be decoded via the decoding algorithm used for the code in Fig. 1(a).

The observed sequence $\mathbf{y} = (y_0, y_1, \ldots, y_{N-1})$ at the receiver is distorted due to transmission over AWGN and/or ISI channels. Thus, the decoding problem can be defined as determining the transmitted sequence $\mathbf{u}$ given the noisy sequence $\mathbf{y}$. Maximum likelihood (ML) decoding algorithm estimates the transmitted symbols via searching the most likely trellis transition path which maximizes the probability $p(\mathbf{y}|\mathbf{u})$. However, the MAP decoding algorithm determines each of symbols u_k independently via maximizing *a posteriori* probability $p(u_k|\mathbf{y})$. The BCJR algorithm [41] solves the MAP decoding problem in a multiplicative form. From an implementation perspective, a log-domain MAP algorithm, called "log-MAP", has the advantage that it can be formulated in terms of sums rather than products.

A log-MAP decoding algorithm estimates the LLR of data symbol u_k denoted as $\Lambda_k(u_k)$ defined below:

$$\Lambda_k(u_k) = \ln \frac{p(u_k = 1|\mathbf{y})}{p(u_k = 0|\mathbf{y})} = \ln \frac{\sum_{\mathbf{u}:u_k=1} p(\mathbf{u}, \mathbf{y})}{\sum_{\mathbf{u}:u_k=0} p(\mathbf{u}, \mathbf{y})}. \quad (1)$$

Defining $P(\mathbf{u}, \mathbf{y}) = \ln p(\mathbf{u}, \mathbf{y})$, we rewrite $\Lambda_k(u_k)$ as

$$\Lambda_k(u_k) = \ln \sum_{\mathbf{u}:u_k=1} e^{P(\mathbf{u},\mathbf{y})} - \ln \sum_{\mathbf{u}:u_k=0} e^{P(\mathbf{u},\mathbf{y})}. \quad (2)$$

Typically, one term will dominate each sum leading to an approximation

$$\Lambda_k(u_k) \approx \max_{\mathbf{u}:u_k=1} P(\mathbf{u}, \mathbf{y}) - \max_{\mathbf{u}:u_k=0} P(\mathbf{u}, \mathbf{y}). \quad (3)$$

The log-domain probability $P(\mathbf{u},\mathbf{y})$ can be reformulated [23] as

$$\begin{aligned} P(\mathbf{u},\mathbf{y}) &= \ln\{p(s_k,\mathbf{y}_{j<k})p(y_k,s_{k+1}|s_k)p(\mathbf{y}_{j>k}|s_{k+1})\} \\ &= \ln p(s_k,\mathbf{y}_{j<k}) + \ln p(y_k,s_{k+1}|s_k) \\ &\quad + \ln p(\mathbf{y}_{j>k}|s_{k+1}), \\ &= \alpha_{k-1}(s_{k-1}) + \lambda_k(s_{k-1},s_k) + \beta_k(s_k) \end{aligned} \tag{4}$$

where $\mathbf{y}_{j<k}$ and $\mathbf{y}_{j>k}$ are the observed sequence before and after the kth trellis section. The first term, $\alpha_{k-1}(s_{k-1})$, referred to as the *forward metric*, is related to the probability that the trellis reaches s_{k-1} given the past observation $\mathbf{y}_{j<k}$. The forward metric $\alpha_k(s_k)$ is computed recursively as

$$\alpha_k(s_k) = \max_{s_{k-1}}\{\alpha_{k-1}(s_{k-1}) + \lambda_k(s_{k-1},s_k)\} \tag{5}$$

by performing a forward sweep on the trellis from s_0. The second term in (4), $\lambda_k(s_{k-1},s_k)$, is referred to as the *branch metric* and is related to the probability that a transition from s_{k-1} to s_k occurs. The branch metric $\lambda_k(s_{k-1},s_k)$ is computed from the channel output, noise statistics, and the error-free output of the branch connecting s_{k-1} and s_k at time k. Note that each trellis section has 2^K possible transitions. The third term $\beta_k(s_k)$, referred to as the *backward metric*, is related to the probability that the trellis reaches state s_k given the future observation $\mathbf{y}_{j>k}$. The backward metric $\beta_k(s_k)$ is computed recursively as

$$\beta_k(s_k) = \max_{s_{k+1}}\{\beta_{k+1}(s) + \lambda_k(s_k,s_{k+1})\} \tag{6}$$

by performing a backward sweep on the trellis from state s_{N-1}. Hence, the decoding process consists of three steps. First, branch metrics in each trellis section are computed. Second, the forward and backward metrics (α_k and β_k) are computed recursively via (5) and (6). Third, the LLR $\Lambda_k(u_k)$ is computed as

$$\begin{aligned} \Lambda_k(u_k) = &\max_{\mathbf{u}:u_k=1}\{\alpha_{k-1} + \lambda_k(s_{k-1},s_k) + \beta_k(s_k)\} \\ &- \max_{\mathbf{u}:u_k=0}\{\alpha_{k-1} + \lambda_k(s_{k-1},s_k) + \beta_k(s_k)\}. \end{aligned} \tag{7}$$

It is known that if the function $\overset{*}{\max}$ defined as

$$\overset{*}{\max}\{x,y\} = \max\{x,y\} + \ln(1+e^{-|x-y|}) \tag{8}$$

is employed instead of $\max$, the performance of the log-domain MAP algorithm approaches that of the BCJR algorithm [23] to within 0.05 dB. The second term in (8) is referred to as the *correction factor*.

B. MAP Decoder Architectures

Numerous architectures can be employed to implement the log-MAP algorithm [43]. However, the trellis sweep over all N observed symbols requires large memory in order to hold the forward and backward metrics until they are used in the LLR computation in (7). Hence, the sliding window log-MAP algorithm, considered in this paper, has become popular as it minimizes the metric storage requirements [23].

The sliding window log-MAP decoding algorithm is derived via the property that the forward and backward metrics α_k and β_k converge after a few constraint lengths have been traversed in the trellis, independent of the initial conditions [23]. We refer to this property as the *warm-up* property and the warm-up period is assumed to have a duration of L symbols. Due to the warm-up property, the state metrics (α_k and β_k) computation can be partitioned into windows of size L. Further, the computations in each window can be done in parallel. Fig. 2 shows an example of a decoding flow where the warm-up property is employed only for computing backward metrics. The warm-up or initialization period is depicted by dashed lines and the computation period by solid lines. This warm-up property will be exploited later in deriving parallel and BIP architectures.

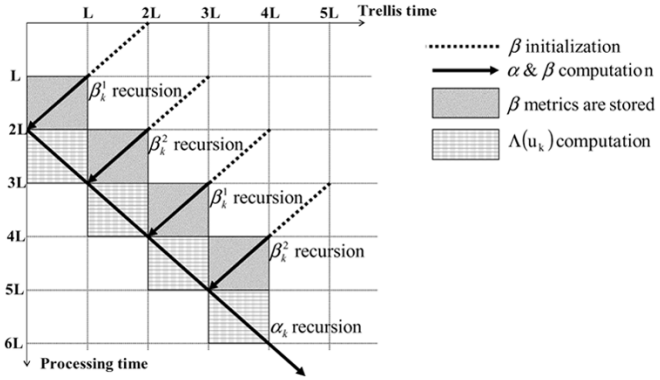


Fig. 2. Scheduling of state metric recursions for α_k and β_k in the sliding window log-MAP. For simplicity, it is assumed that the computation and warm-up period equals L. Shadowed region indicates the computation and storage of β_k. The gridded region indicates the computation of α_k followed by the computation of $\Lambda(u_k)$. Here, β_k^1 and β_k^2 are the first and second β-recursion outputs, respectively, at a time index k.

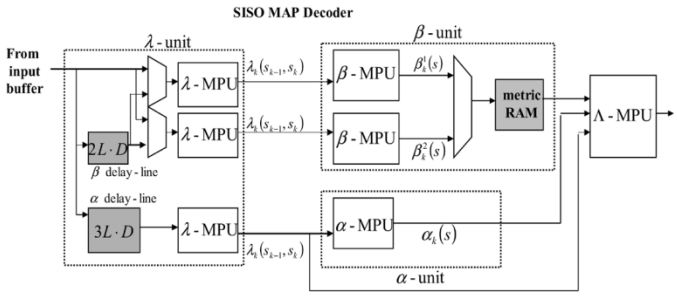


Fig. 3. Baseline VLSI architecture of a sliding window log-MAP decoder.

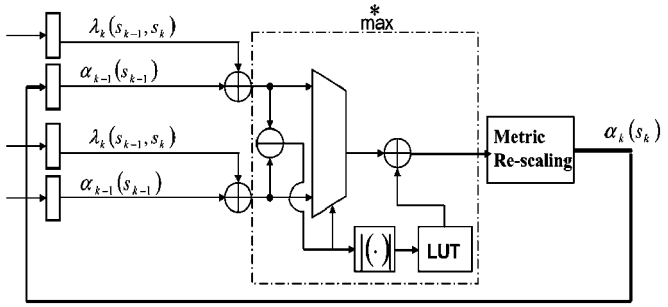


Fig. 4. Add-Compare-Select (ACS) unit.

Fig. 3 shows the VLSI architecture of a decoder whose data-flow graph is shown in Fig. 2. The architecture has units for the computation of branch metrics (λ-unit), one forward

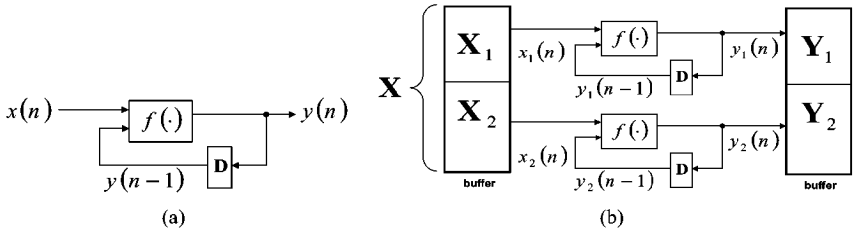


Fig. 5. Parallel processing. (a) Original. (b) Sub-block parallel architecture.

recursion (α-unit), two backward recursions and a buffer to store backward metrics β_k^1 and β_k^2 (β-unit), and the Λ metric processing unit (Λ-MPU). The computations in λ-MPU and Λ-MPU can be implemented in a feedforward manner and thus these do not limit the throughput. However, the forward and backward recursions are computed via an array of ACS kernels in a state-parallel manner. An ACS kernel is depicted in Fig. 4, where the correction factor in (8) is implemented via a look-up-table (LUT) and state metric re-scaling is employed to avoid overflows [14]. Therefore, it is the critical path delay of the ACS unit shown in Fig. 4 that limits the throughput.

III. Block-Interleaved Pipelining Technique

In this section, we present techniques for improving the throughput of recursive data-flow graphs. First, we review the existing techniques of parallel processing [25], [26] and look-ahead transform [39], [40]. Then, we present the BIP technique, which is one of the contributions of this paper. These techniques are applied to design high-throughput MAP decoder architectures in Section IV.

A. Parallel Processing

In general, pipelining or parallel processing becomes difficult for a recursive datapath. However, if the data is being processed in blocks and the processing satisfies the following two properties: 1) computation between blocks are independent and 2) computation within a block is recursive, then a block parallel processing architecture can be achieved. Further, if a block can be segmented into computationally independent sub-blocks, parallel processing can be applied at the sub-block level leading to high-throughput architectures presented in [25] and [26].

Consider the recursive architecture in Fig. 5(a). Note that the architecture in Fig. 5(a) cannot be easily pipelined or parallelized due to the presence of the feedback loop. However, if a data block of length N is processed independent of other blocks and the computations within a block can be segmented into computationally independent sub-blocks, then one can parallelize the architecture as shown in Fig. 5(b), where the parallelization factor $M = 2$ and a block $\mathbf{X}$ is divided into $M = 2$ sub-blocks, $\mathbf{X}_1$ and $\mathbf{X}_2$. It is obvious that the critical path is not affected and the throughput is increased by a factor of M at the expense of a factor of M increase in hardware complexity.

B. Look-Ahead Transform

Another transform to achieve high-throughput for recursive data-flow graphs is look-ahead computation [39]. Look-ahead

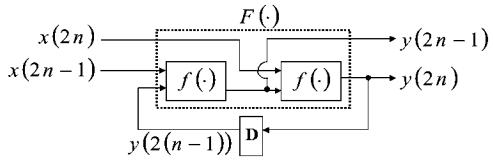


Fig. 6. Look-ahead transform.

leads to an increase in the number of symbols processed at each time step as shown in Fig. 6, where two symbols are processed in one clock cycle. If $\mathbf{x}(n) = [x(2n-1), x(2n)]$ and $\mathbf{y}(n) = [y(2n-1), y(2n)]$, then look-ahead results in the output being expressed as $\mathbf{y}(n) = F(\mathbf{x}(n), y(2(n-1))$. Note that $F(\cdot)$ will have a longer critical path delay than the original computation of Fig. 5(a). However, it has been shown that the function $F(\cdot)$ can be optimized via logic minimization so that an overall increase in throughput can be achieved. For example, in the context of Viterbi decoding, it has been shown that a 1.7 times increase in throughput is feasible via radix-4 computation [40].

C. Block Interleaved Pipelining (BIP)

The parallel architecture in Fig. 5(b), where the level of parallelism is equal to 2, has two identical computation units processing on two independent input symbols. Therefore, the hardware complexity increases linearly with the level of parallelism M. To achieve an area-efficient implementation, we propose the block interleaved pipelining (BIP) technique. First, the data-flow of Fig. 5(b) is folded [39] to a single computation unit as shown in Fig. 7(a) where two independent computations are carried out in a single computational unit. Note that the resulting BIP architecture in Fig. 7(a) is inherently pipelined. Therefore, an application of retiming [39] [see Fig. 7(b)] results in reduction of the critical path delay by a factor of two over that of the original architecture in Fig. 5(a). It is clear that the retimed BIP architecture in Fig. 7(b) leads to high-throughput at the cost of a marginal increase in memory due to pipelining latches when compared to the architecture in Fig. 5(a).

The BIP technique is applicable to general recursive data-flow graphs that satisfy the following two properties: 1) computation is block-based with computation between the blocks being independent and 2) operations within a block is recursive. Further, if block computation can be partitioned into sub-block computation via algorithmic transforms while maintaining independence between sub-blocks, the BIP can be applied in a sub-block level. In such a case, processing M sub-blocks in a parallel manner leads to the BIP architecture which is inherently pipelined.

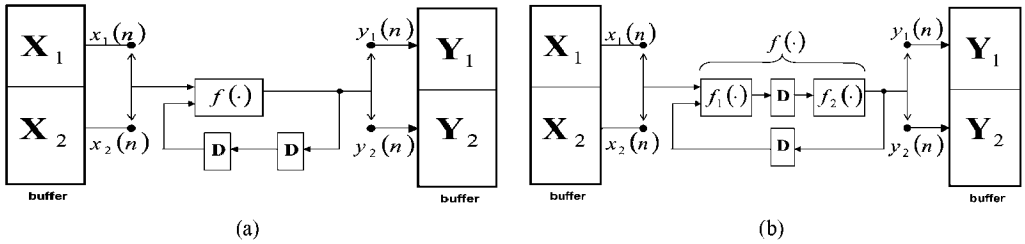


Fig. 7. Block-interleaved pipelining (BIP). (a) BIP architecture. (b) Retimed BIP architecture.

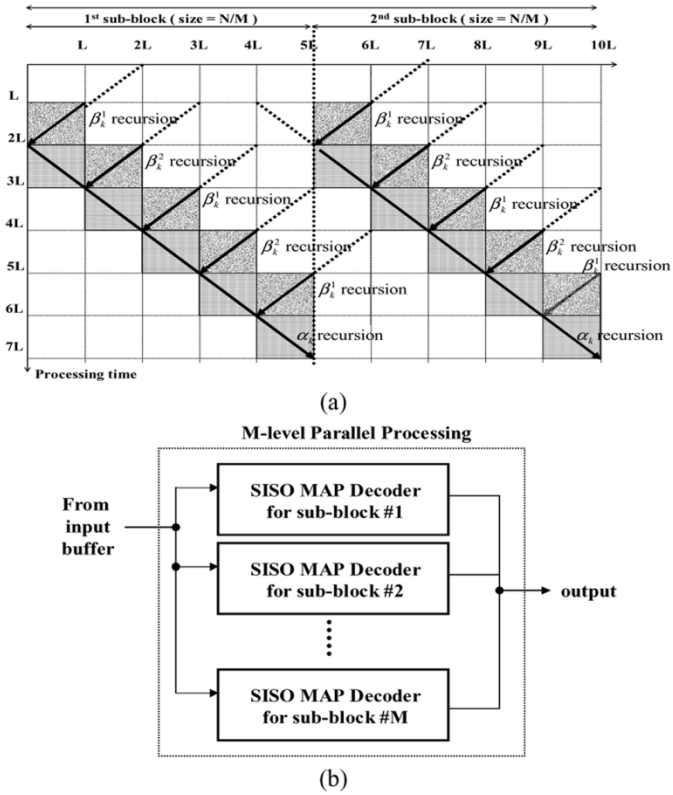


Fig. 8. Parallel architecture. (a) Scheduling for parallel sub-block processing with $M = 2$. (b) VLSI architecture with M-level parallel sub-block processing.

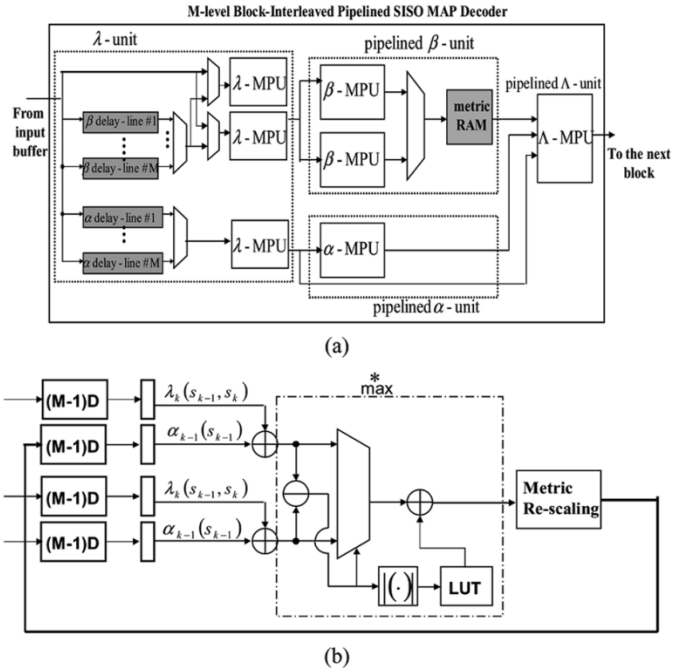


Fig. 9. BIP architecture. (a) M-level BIP architecture. (b) BIP ACS architecture.

IV. High-Throughput MAP Decoder Architectures

In this section, we apply the techniques in Section III to develop parallel, BIP, symbol-based, and symbol-based BIP decoder architectures.

A. Parallel Architecture

Due to the warm-up property of MAP decoding described in Section II-B, a block of N information bits can be segmented into M sub-blocks with the warm-up period L overlapped among sub-blocks. Thus, by employing the parallelization technique in Section III-A, M identical processing units are implemented in parallel. This sub-block parallel architecture was presented in [25]–[28] and leads to a linear increase in hardware complexity for a linear speedup in throughput. For example, a block size of $N = 10L$ can be divided into $M = 2$ sub-blocks of size $5L$. Thus, the L trellis sections between the $4L$th and $5L$th trellis nodes are used as the warm-up period for the α recursion of the second sub-block. Fig. 8(a) shows the data-flow for an example with $N = 10L$ and $M = 2$. Note that the beginning and ending states of the trellis are known to the decoder and hence warm-up period is not required for the first and last computation windows. Fig. 8(b) depicts an M-level parallel architecture.

B. BIP Architecture

Following the approach described in Section III-C, the BIP architecture is obtained by processing M sub-blocks of size N/M. For simplicity, we refer to this architecture as a BIP architecture even though the interleaving is done at the sub-block level and we assume that the block-size (N) and the sub-block size (N/M) is a multiple of the warm-up period L.

Fig. 9(a) depicts the BIP architecture, where the α, β, and Λ MPUs are pipelined in order to reduce the critical path delay and M delay-line blocks are required to compute M sub-blocks in a block-interleaved order. The α and β units have the folded block-interleaved ACS architecture shown in Fig. 9(b), where the block-interleaving factor is M. The BIP architecture processes trellis sections $k, k+(N/M), k+2(N/M) \ldots k+(M-1)(N/M)$ in consecutive clock cycles. Compared with the conventional implementation in Fig. 4, the block-interleaved ACS architecture in Fig. 9(b) has M delays in the feedback loop of the ACS recursion. Thus, retiming can be employed to pipeline

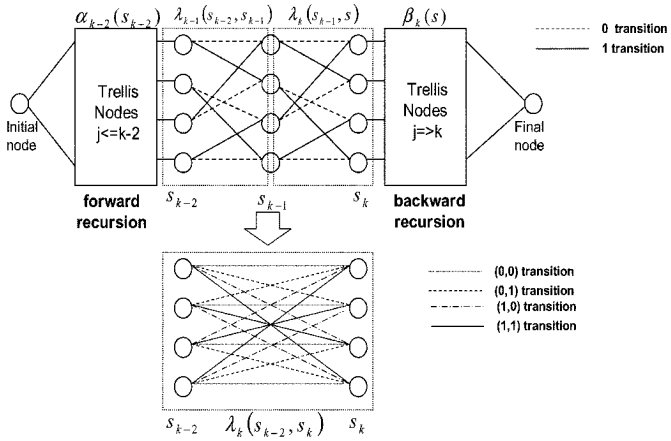


Fig. 10. Proposed symbol-based decoding scheme for a example with $[5, 7]_8$ RSC encoder.

the critical path by M levels in order to achieve speed-up by a factor of M. The price we pay is an increase in the pipelining registers. Hence, the BIP architecture will consume less area than the parallel architecture.

C. Symbol-Based Architecture

We apply an r-level look-ahead transform to the α and β recursions. Doing so enables us to compute the α and β metrics by merging r trellis sections and to decode r bits per each clock cycle [40]. We refer to this approach as *symbol-based* MAP decoding. For simplicity, a 2-bit symbol decoding, where two trellis sections are grouped (see Fig. 10), is described next. Extension to the multi-bit symbol case is straightforward.

We reformulate the α recursion in (5) by applying a two-level look-ahead transform as follows:

$$\begin{aligned}\alpha_k(s_k) &= \overset{*}{\max_{s_{k-1}}}\{\alpha_{k-1}(s_{k-1}) + \lambda_k(s_{k-1}, s_k)\} \\ &= \overset{*}{\max_{s_{k-1}}}\Big\{\overset{*}{\max_{s_{k-2}}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_{k-1})\} \\ &\qquad + \lambda_k(s_{k-1}, s_k)\Big\} \\ &= \overset{*}{\max_{s_{k-2}, s_{k-1}}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k)\} \qquad (9)\end{aligned}$$

since the $\overset{*}{\max}$ operation is associative and addition distributes over $\overset{*}{\max}$. Here, $\lambda_k(s_{k-2}, s_k)$ is the branch metric of the path connecting s_{k-2} and s_k. In a similar way, the β recursion in (6) is rewritten as

$$\beta_k(s_k) = \overset{*}{\max_{s_{k+1}, s_{k+2}}}\{\beta_{k+2}(s_{k+2}) + \lambda_k(s_k, s_{k+2})\}. \qquad (10)$$

In order to compute $\Lambda_k(u_k)$ via 2-bit symbol decoding, either α_{k-1} or β_k in (7) can be extended to α_{k-2} or β_{k+1}. Here, we extend the α recursion. Employing (5), we substitute for α_{k-1} in (7) as follows:

$$\begin{aligned}\Lambda(u_k) = &\overset{*}{\max_{s_{k-2}, s_k : u_k = 1}}\Big\{\overset{*}{\max_{s_{k-2}}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_{k-1})\} \\ &\qquad + \lambda_k(s_{k-1}, s_k) + \beta_k(s_k)\Big\} \\ &- \overset{*}{\max_{s_{k-2}, s_k : u_k = 0}}\Big\{\overset{*}{\max_{s_{k-2}}}\{\alpha_{k-2}(s_{k-2}) \\ &\qquad + \lambda_k(s_{k-2}, s_{k-1})\} \\ &+ \lambda_k(s_{k-1}, s_k) + \beta_k(s_k)\Big\} \qquad (11)\end{aligned}$$

and since $\overset{*}{\max}$ operation is associative and addition distributes over $\overset{*}{\max}$, we have

$$\begin{aligned}\Lambda(u_k) = &\overset{*}{\max_{s_{k-2}, s_k : u_k = 1}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k) \\ &\qquad + \beta_k(s_k)\} \\ &- \overset{*}{\max_{s_{k-2}, s_k : u_k = 0}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k) \\ &\qquad + \beta_k(s_k)\}. \qquad (12)\end{aligned}$$

Similarly, $\Lambda(u_{k-1})$ can be computed as

$$\begin{aligned}\Lambda(u_{k-1}) = &\overset{*}{\max_{s_{k-2}, s_k : u_{k-1} = 1}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k) \\ &\qquad + \beta_k(s_k)\} \\ &- \overset{*}{\max_{s_{k-2}, s_k : u_{k-1} = 0}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k) \\ &\qquad + \beta_k(s_k)\}. \qquad (13)\end{aligned}$$

The first term in (13) can be expressed as

$$\begin{aligned}&\overset{*}{\max}\Big\{\overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{1,0}) + \beta_k(s_k)\}, \\ &\overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{1,1}) + \beta_k(s_k)\}\Big\} \qquad (14)\end{aligned}$$

where $\tilde{u}_{(i,j)}$ corresponds to the transition with $u_{k-1} = i$ and $u_k = j$. By defining symbol reliability metric $\Gamma_{u_{k-1}u_k}$ as

$$\begin{aligned}\Gamma_{00} &= \overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{0,0}) + \beta_k(s_k)\} \\ \Gamma_{01} &= \overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{0,1}) + \beta_k(s_k)\} \\ \Gamma_{10} &= \overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{1,0}) + \beta_k(s_k)\} \\ \Gamma_{11} &= \overset{*}{\max_{s_{k-2}, s_k}}\{\alpha_{k-2}(s_{k-2}) + \lambda_k(s_{k-2}, s_k, \tilde{u}_{1,1}) + \beta_k(s_k)\} \qquad (15)\end{aligned}$$

we can compute $\Lambda(u_k)$ and $\Lambda(u_{k-1})$ as follows:

$$\begin{aligned}\Lambda(u_{k-1}) &= \overset{*}{\max}\{\Gamma_{10}, \Gamma_{11}\} - \overset{*}{\max}\{\Gamma_{00}, \Gamma_{01}\} \\ \Lambda(u_k) &= \overset{*}{\max}\{\Gamma_{01}, \Gamma_{11}\} - \overset{*}{\max}\{\Gamma_{00}, \Gamma_{10}\}. \qquad (16)\end{aligned}$$

Note that LLR outputs of (16) equal those computed from the bit-wise decoding algorithm in (7). The modified ACS architecture and the Λ-MPU are depicted in Fig. 11. In general, (16) can be derived for an r-bit symbol-based decoding as follows:

$$\Lambda(u_j) = \overset{*}{\max_{\forall\, \Gamma : u_j = 1}}\Gamma_{b_{k-r+1}\ldots b_k} - \overset{*}{\max_{\forall\, \Gamma : u_j = 0}}\Gamma_{b_{k-r+1}\ldots b_k} \qquad (17)$$

where $\Gamma_{b_{k-r+1}\ldots b_k}$ is the symbol made up of bits from b_{k-r+1} to b_k and $j = k - r + 1, \ldots, k$. Although the symbol-based decoding increases the number of decoded bits per clock cycle, the critical path delay is increased as shown in Fig. 11. However, this architecture reduces the storage requirement for state metrics by a factor of r.

In general, assuming that T_A and T_{MAX} are equal to delays of an adder and a $\overset{*}{\max}$, respectively, and the delay of the re-scaling block is negligible, the critical path delay of r-bit symbol-based algorithm equals $T_A + rT_{\mathrm{MAX}}$ as shown in Fig. 11. The expected speed-up η is written as

$$\eta = \frac{N(T_A + T_{\mathrm{MAX}})}{\frac{N}{r}(T_A + rT_{\mathrm{MAX}})} = \frac{T_A + T_{\mathrm{MAX}}}{\frac{T_A}{r} + T_{\mathrm{MAX}}} \qquad (18)$$

TABLE I
STORAGE REQUIREMENT, LOGIC COMPLEXITY, CLOCK FREQUENCY, AND SPEED-UP OF HIGH-THROUGHPUT ARCHITECTURES

	Parallel	BIP	Symbol-based	Symbol-based BIP
Storage Requirement (number of bits)	$4L(2B_y+B_{LLR})M$ $+LN_sB_{pm}M$	$4L(2B_y+B_{LLR})M$ $+LN_sB_{pm}M$	$4L(2B_y+B_{LLR})$ $+LN_sB_{pm}\frac{1}{M}$	$4L(2B_y+B_{LLR})M$ $+LN_sB_{pm}$
Number of 1-bit adder	$24B_{bm}M+$ $10N_sB_{pm}M$	$24B_{bm}+$ $10N_sB_{pm}$	$3(3M-1)2^{2M}B_{bm}+$ $5N_s2^MB_{pm}$	$3(3M-1)2^{2M}B_{bm}+$ $5N_s2^MB_{pm}$
Number of B_{pm}-bit $\overset{*}{\max}$	$(5N_s-2)M$	$(5N_s-2)$	$2^M(4N_s+M-1)$ $-2M-3N_s$	$2^M(4N_s+M-1)$ $-2M-3N_s$
Number of 1-bit latches	$4B_{bm}M+$ $(5N_s-2)B_{Pm}M$	$4B_{bm}+$ $3N_sB_{Pm}(2M-1)$ $+2(N_s-1)B_{pm}M$	$2^{2M}B_{bm}+$ $2^M(N_s+M-1)B_{Pm}$ $+(3N_s-2M)B_{pm}$	$2^{2M}B_{bm}+$ $2^M(4N_s+M-1)B_{Pm}$ $-(2M+3N_s)B_{pm}$
Clock Frequency	$\frac{1}{T_A+T_{MAX}}$	$\frac{M}{T_A+T_{MAX}}$	$\frac{1}{T_A+MT_{MAX}}$	$\frac{M}{T_A+MT_{MAX}}$
Speed-up (η)	M	M	$\frac{T_A+T_{MAX}}{\frac{T_A}{M}+T_{MAX}}$	$M\cdot\frac{T_A+T_{MAX}}{\frac{T_A}{M}+T_{MAX}}$

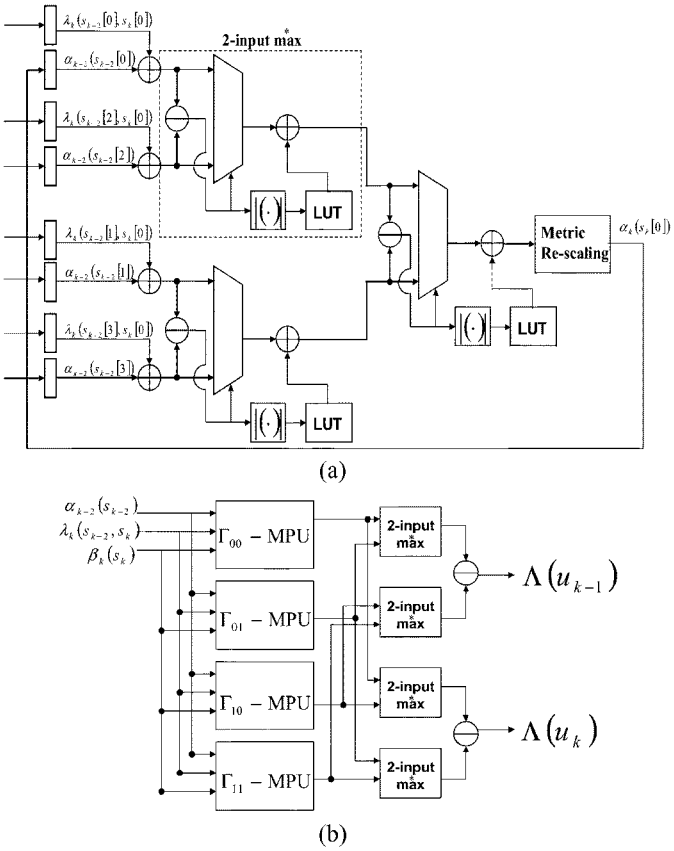


Fig. 11. Symbol-based architectures with $r = 2$ for: (a) ACS recursion, and (b) the Λ-MPU.

where the numerator and the denominator in (18) are the computation times for processing a block of N bits using the ACS architectures of Fig. 4 and Fig. 11, respectively. In general, $T_A < T_{\rm MAX}$ and hence a speed-up in the range of $1 < \eta < 2$ is expected at the cost of an increase in logic complexity due to the look-ahead transform. In a typical scenario, without any logic optimization, we get $2T_A = T_{\rm MAX}$. Hence, η approaches 1.5 asymptotically as r increases. Employing appropriate logic optimizations, it is possible to reach $\eta = 1.7$ as shown in [40] for a Viterbi decoder.

D. Symbol-Based BIP Architecture

In order to increase the throughput gain of the symbol-based decoding architecture, we apply an r-level BIP transform along with an r-level look-ahead transform. First, a block is divided into r sub-blocks using the warm-up property and a r-bit symbol-based decoder is employed to process each sub-block. Then, the r identical symbol-based decoders are folded on to a single hardware unit resulting in a r-level pipelined r-bit symbol-based decoder. Retiming the r latches results in a symbol-based BIP decoder architecture. The symbol-based BIP architecture has a speed-up of

$$\eta = r \cdot \frac{T_A + T_{\rm MAX}}{\frac{T_A}{r} + T_{\rm MAX}}. \tag{19}$$

Thus, the symbol-based BIP is expected to achieve the highest speed-up among the above mentioned architectures. The symbol-based BIP architecture is same as the architecture of Fig. 9(a) except for the fact that the λ-MPU, α-MPU, β-MPU, and Λ-MPU have symbol-based datapaths.

E. Comparison

Table I summarizes the storage requirement, logic complexity, clock frequency, and expected speed-up of each high-throughput architecture, where M-level parallelism is considered, N_s is the number of states in the trellis, B_{bm}, B_{pm}, B_y, and $B_{\rm LLR}$ are precisions of branch metrics, state metrics, received symbol (y), and LLR, respectively. As mentioned previously, the hardware complexity including the storage requirement and logic complexity of the parallel processing architecture increases linearly with speed-up M. Compared to the parallel architecture, the BIP architecture provides the same speed-up with a reduction in logic complexity by a factor of M. The symbol-based architecture provides a speed-up in the range 1 to 2, with a logic complexity that grows exponentially with M but with a state metric storage requirement that decreases by a factor of M as compared to a parallel architecture. The symbol-based BIP architecture has a speed-up in the range M to $2M$ with a logic complexity that increases exponentially with M and a state metric memory complexity that decreases roughly by a factor of M.

We estimate the area based on a complexity analysis in Table I. A standard cell-based design methodology is considered. Actual areas of a 1-bit adder, 1-bit latch, $\max^*$, and 1-bit buffer cell are obtained from a 0.25-μm CMOS standard cell library, and precisions are determined via computer simulations ($B_{\rm LLR} = 6$, $B_y = 8$, $B_{bm} = 8$, and $B_{pm} = 12$). The interconnect area is not included. For $K = 3$ and 4, the

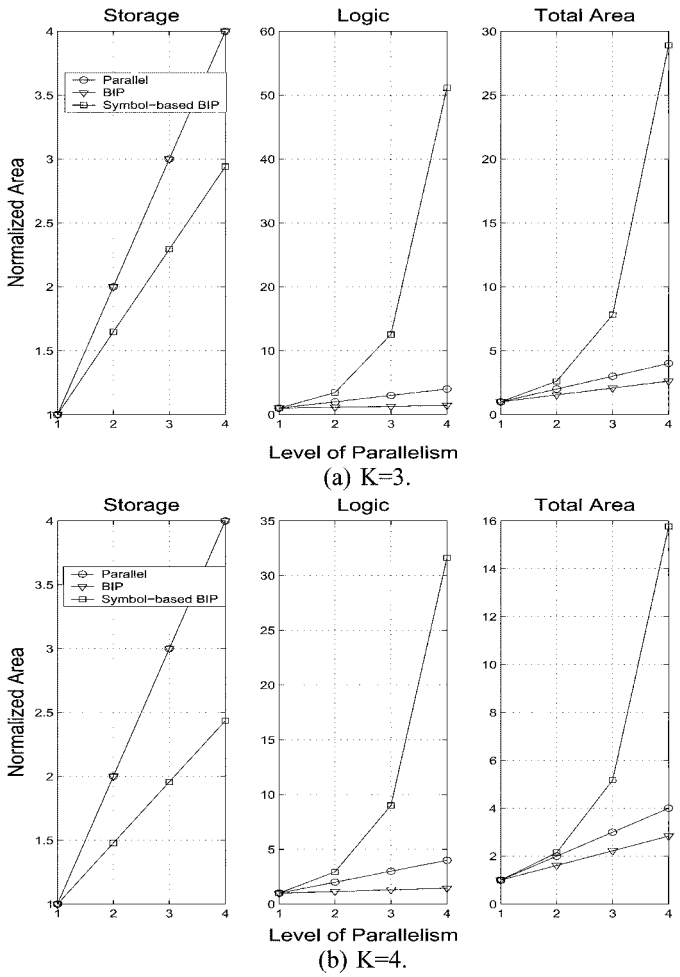


Fig. 12. SISO MAP decoder area comparison.

estimated silicon area, computed from Table I, is shown in Fig. 12 in terms of the normalized area, which is defined as the ratio of the area of the high-throughput architecture over that of the original architecture $(M = 1)$. The BIP architecture consumes 65%–80% of the area of the parallel architecture for $M = 2, 3$, and 4. The symbol-based BIP architecture occupies 30% and 7% more area than the parallel architecture for $K = 3$ and 4, respectively, for $M = 2$. However, the normalized area increases exponentially with M and hence the symbol-based BIP architecture is not considered as practical for $M \geq 3$.

Different memory access patterns for input and output memories (interleavers/deinterleavers in turbo decoders and equalizers) should be considered for each different high-throughput MAP architecture. In case of an M-level parallel architecture, the clock frequency is $1/(T_A + T_{\mathrm{MAX}})$, which is same as that of a nonparallel MAP decoder architecture, but at each clock M symbols need to be decoded in parallel. Thus, M memory accesses are executed within $T_A + T_{\mathrm{MAX}}$ period unless input and output memories are partitioned and accessed in parallel such that no access conflict occurs. The symbol-based architecture has the same problem that M memory accesses are executed during $T_A + MT_{\mathrm{MAX}}$ period, but the clock period is increased approximately by a fact of M. On the other hand, the BIP architecture needs one memory access per one clock cycle, but the clock period is reduced by a fact of M in comparison with

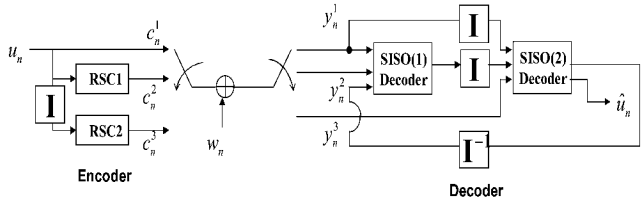


Fig. 13. Block diagram of a PCCC encoder and a turbo decoder. Here $\mathbf{I}$ and $\mathbf{I}^{-1}$ denote block interleaving and deinterleaving, respectively.

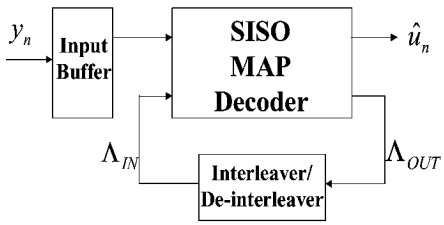


Fig. 14. Serial VLSI turbo decoder architecture.

a parallel architecture. Since the memory and logic clock networks are synchronized with one clock source, the BIP architecture requires more simple clock networks than the parallel and symbol-based architectures.

In summary, the parallel and symbol-based decoder architectures improve throughput at the expense of increased area. The area increases linearly and exponentially with the speed-up M for the parallel and symbol-based architectures, respectively. The BIP reduces this area penalty without sacrificing the throughput gains.

V. Applications

In this section, the proposed high-throughput MAP decoder architectures are employed for high-throughput implementations of turbo decoders and turbo equalizers, and architectural benefits of each approach are compared.

A. Turbo Decoder

The turbo code considered in this paper is made up of two recursive systematic convolutional (RSC) encoders concatenated in parallel as shown in Fig. 13. The bit sequences transferred from one encoder to the other are permuted by an interleaver. The decoder contains two SISO MAP decoders which are associated with two RSC encoders as depicted in Fig. 13. The decoding of the observed sequences is performed iteratively via the exchange of soft output information $(\Lambda(u_k))$ between the constituent decoders. The decoding process is repeated iteratively until a proper stopping criterion terminates the iterations.

The turbo decoder can be implemented via a serial architecture as shown in Fig. 14, where one SISO MAP decoder is time-shared. Hence, increasing the throughput of the SISO MAP decoder directly leads to an overall improvement in throughput of the turbo decoder. Based on the area estimation results in Table I, it is predicted that BIP MAP decoder architecture achieves the best speed-up at the minimum area cost for the turbo decoder implementation.

In order to validate our analysis, three high-throughput MAP decoders were designed using the techniques in Section IV.

TABLE II
TURBO DECODER APPLICATION RESULTS

		Parallel	BIP	Symbol-based BIP
Critical path delay ($nsec$)	$K=3$	13.994	7.126	13.011
	$K=4$	17.767	9.793	15.517
Speed-up (η)	$K=3$	2	1.96	2.15
	$K=4$	2	1.81	2.29
Normalized area	$K=3$	2	1.63	2.03
	$K=4$	2	1.76	1.83
Measured area (mm^2)	$K=3$	5.3(2.87/2.43)	4.32(2.87/1.45)	5.4(2.36/3.04)
(Memory/Logic)	$K=4$	10.9(7.46/3.44)	9.62(7.46/2.16)	9.99(5.58/4.41)

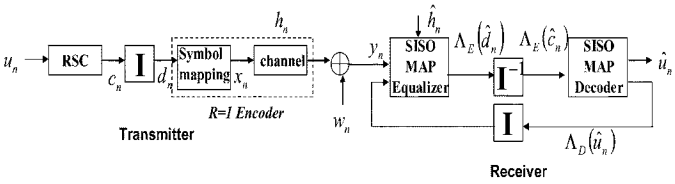


Fig. 15. Transmitter and receiver model of turbo equalizer. The superscripts E and D denote equalizer and decoder, respectively.

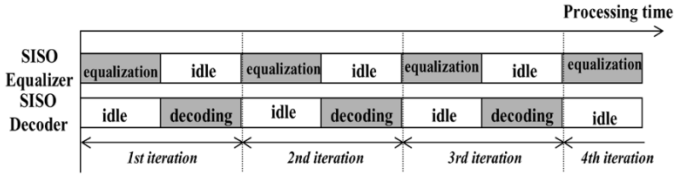


Fig. 16. Decoding flow of the conventional turbo equalizer implementation.

These decoder architectures were implemented in VHDL for $K=3$ and 4 and their functionality was verified. The speed-up factor was $M=2$ in all cases. Encoder polynomials were chosen to be $[5,7]_8$ $(L=16)$ and $[13,15]_8$ $(L=32)$ for $K=3$ and $K=4$, respectively. The design was synthesized using the *Synopsys Design Compiler* using 2.5-V 0.25-μm CMOS technology standard cell library. The synthesized design was placed and routed using the *Cadence Silicon Ensemble*. *Pathmill* was used to determine the critical path from post layout simulations. Measured results on the critical path delay, speed-up, silicon area, and throughput are summarized in Table II. The measured normalized area is close to the predicted results in Fig. 12, and the measured memory and logic areas clearly show that the BIP technique reduces the logic complexity nearly by a factor of M keeping the memory complexity same as the parallel processing. As we predicted previously from the analysis, the BIP architecture achieves high-throughput with minimal area cost. It should be noticed that the BIP symbol-based decoding architecture shows the maximum throughput gain for two-level parallelism (M) as expected in Section IV-D. However, as M is increased, the logic complexity is expected to increase exponentially (see Fig. 12). Hence, the BIP symbol-based decoding architecture does not provide a good tradeoff between the area overhead and the throughput gain for large values of M.

B. Turbo Equalizer

In the case where the transmitted symbols are protected by RSC encoders (see Fig. 15), the equalizer and decoder modules can be related in the same way as in turbo decoding of serially concatenated convolutional codes [2], [4]. Hence, in order to achieve iterative processing gain, the SISO MAP equalizer is employed and provides the SISO MAP decoder with soft outputs $(\Lambda^E(c_k))$ as depicted in Fig. 15. If a binary phase shift keying (BPSK) modulation is concatenated with an RSC encoder in Fig. 15, one SISO MAP decoder can be shared to implement both SISO MAP equalizer and decoder as in a serial architecture (Fig. 14). However, if quadrature phase shift keying (QPSK) modulation is employed, then the MAP SISO equalizer processes one QPSK symbol, composed of two bits, but the SISO MAP decoder decodes one bit at each clock. Hence, the equalizer and the decoder need separate hardware platforms as they have different trellis structures. Furthermore, since block-based equalization and decoding are carried out iteratively between two SISO blocks, one of the blocks, either the equalizer or the decoder, is in an idle state as shown in Fig. 16. This inefficient use of computational resources can be overcome by employing the proposed symbol-based MAP decoder.

Fig. 17 depicts the proposed area-efficient VLSI architecture for turbo equalizers, where instead of two separate MAP equalizer and decoder a single symbol-based MAP decoder is employed thereby both achieving the dual goals of saving area and fully utilizing the hardware resources. Since the equalizer processes the received symbols from the channel and LLR values on each bit from the SISO decoder, the branch metric of the SISO MAP equalizer is computed as

$$\begin{aligned}\lambda_k(s_{k-1},s_k) &= \ln p(y_k,s_k|s_{k-1}) \\ &= \ln p(y_k|s_k,s_{k-1}) + \ln p(s_k|s_{k-1}) \\ &= -\frac{1}{2\sigma_w^2}\left|y_k - \sum_{i=0}^{l-1} h_i x_{k-1}\right|^2 \\ &\quad + \frac{1}{2}\sum_{i=0}^{r-1} L_D(d_k)(2d_k-1) \qquad (20)\end{aligned}$$

$$L_D(d_k) = \begin{cases} \ln\{1+\exp(\Lambda_D(d_k))\}, & \text{if } d_k = 0 \\ \Lambda_D(d_k) \\ \quad -\ln\{1+\exp(\Lambda_D(d_k))\}, & \text{else } (d_k = 1) \end{cases} \qquad (21)$$

where a symbol x_k is made up of r bits, $d_0,\ldots,d_{r-1}$, $(d_i \in \{0,1\})$ and the channel length is l. On the other hand, the branch metric of SISO MAP decoder is defined as

$$\lambda_k(s_{k-1},s_k) \triangleq \frac{1}{2}\sum_{i=0}^{n_0-1}\Lambda_E(c_i)(2c_i-1) \qquad (22)$$

where one code symbol is composed of n_0 bits. Thus, Fig. 17 has two branch metric computation units, but other computation units such as α, β, and Λ are shared.

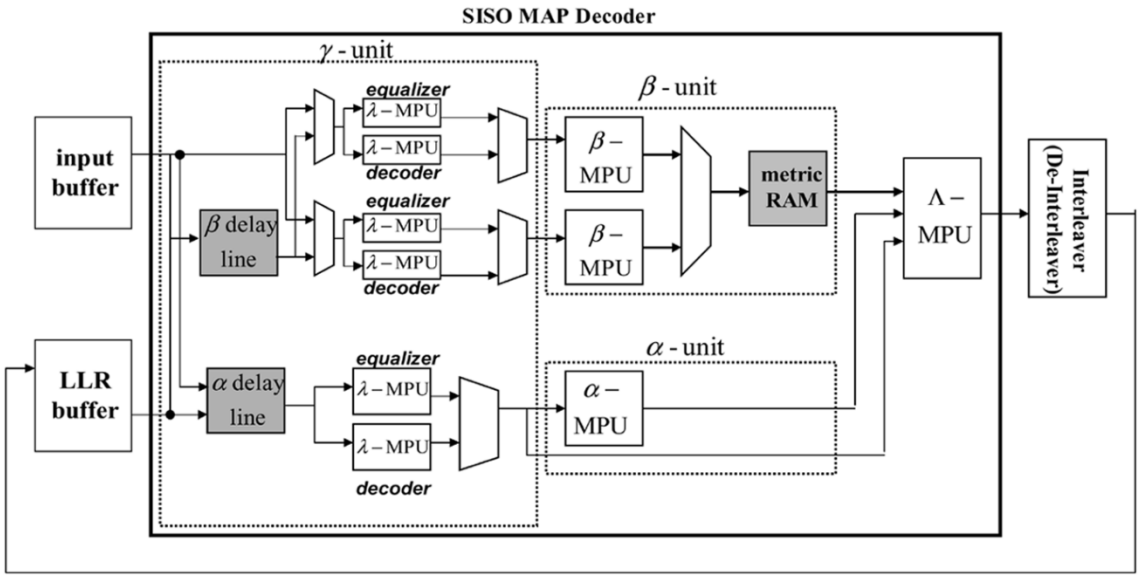


Fig. 17. Proposed area-efficient VLSI architecture for turbo equalizers.

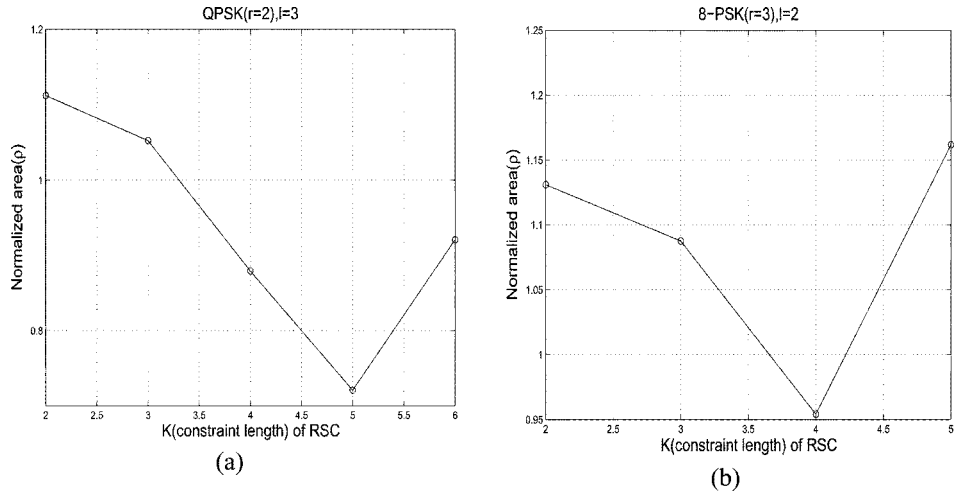


Fig. 18. Predicted area ratio ρ for two different scenarios: (a) QPSK $(r = 2)$, $l = 3$, and (b) 8-PSK $(r = 3)$, $l = 2$.

Though the parallel processing transform can be applied in addition to symbol-based decoding for improving throughput, the BIP symbol-based decoder consumes much less silicon area, and hence provides the better trade off between throughput gain and area cost. In this paper, we compare the area-efficient high-throughput turbo equalizer architecture with the conventional architecture where two SISO MAP equalizer and decoder are separately implemented.

1) Area: By following the approach of Section IV-E, we can predict the normalized area of the area-efficient high-throughput turbo equalizer over that of the conventional architecture. The normalized area ρ is given by

$$\rho = \frac{A_p}{A_E + A_D} \tag{23}$$

where A_p is the area of the proposed area-efficient high-throughput architecture employing the symbol-based BIP processing and A_E and A_D are the area of MAP equalizer and decoder for the separate implementation of two SISO MAP blocks. The predicted normalized area is computed using Table I and plotted in Fig. 18 for two examples where a MAP equalizer is considered for QPSK modulation with the channel length $l = 3$ ($4^{l-1} = 16$ states) and 8-PSK modulation with the channel length $l = 2$ ($8^{l-1} = 8$ states). It is observed that the normalized area is minimized when the number of states in the equalizer and decoder are equal. Further, as the difference between the numbers of states of two SISO blocks is increased, the normalized area increases. This is because the proposed symbol-based SISO decoder is designed to accommodate the larger of the two trellises.

2) Speed-Up: The processing time required for one block iteration of equalization and decoding is expressed as

$$T_B = \frac{N}{r}\tau_E + NR\tau_D \tag{24}$$

where τ_E and τ_D are the critical path delays of SISO MAP equalizer and decoder, respectively. Assuming that the equalizer processes r-bit symbols and hence, $\tau_E = r\tau_D$, the speed-up η is

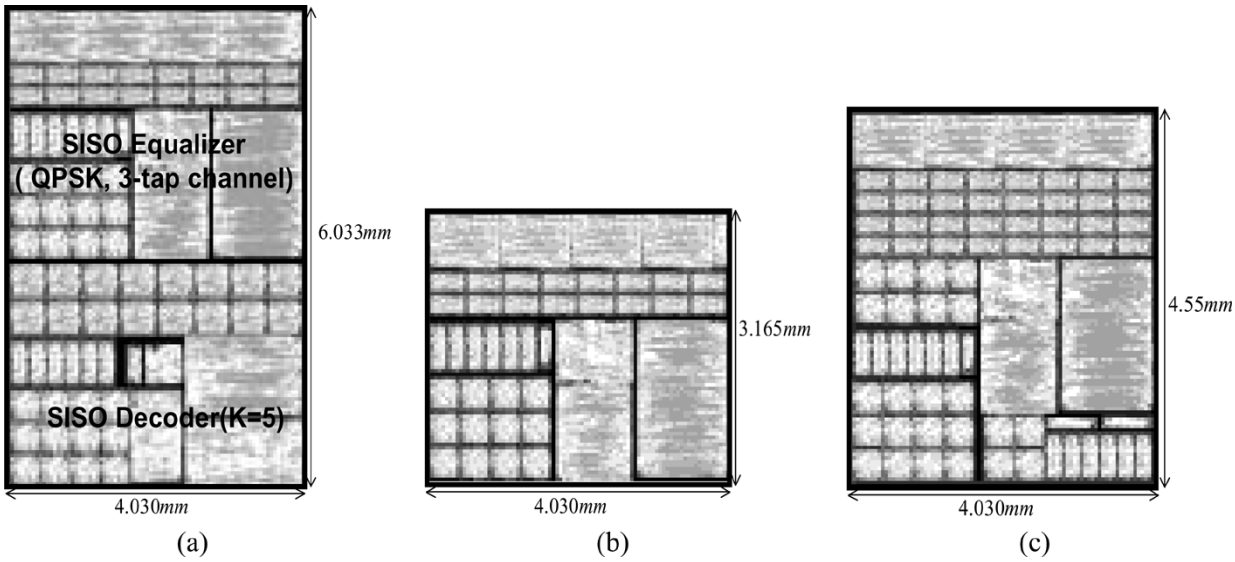


Fig. 19. Turbo equalizer core layouts. (a) Conventional architecture with separate equalizer and decoder. (b) Symbol-based architecture with $r = 2$. (c) Symbol-based BIP architecture with $M = r = 2$.

TABLE III
TURBO EQUALIZER APPLICATION RESULTS

	Area(mm^2)	Delay($nsec$)	Throughput gain (η)
Conventional architecture (MAP equalizer and MAP Decoder)	24.39	$\tau_E = 28.513, \tau_D = 20.015$	1
Symbol-based Architecture ($r = 2$)	12.81	$\tau_E = \tau_D = 28.513$	1.13
Symbol-based BIP architecture($r = 2, M = 2$)	18.13	$\tau_E = \tau_D = 17.739$	1.79
Parallel architecture ($M = 2$)	48.78	same as conventional architecture	2

the ratio of T_B of the conventional approach over the proposed high-throughput architecture

$$\eta = \frac{\frac{N}{r}r\tau_D + NR\tau_D}{\frac{1}{r}(N + 2(r-1)L + NR)\tau_P} = \frac{N(1+R)}{(N + 2(r-1)L + NR)}\frac{r\tau_D}{\tau_P}$$

where $2(r-1)L$ extra cycles are caused by a block-interleaved computation and τ_P is the critical path delay of the proposed high-throughput architecture. Note that the speed-up η becomes larger as τ_P gets closer to τ_D.

3) Experimental Results: We employ a RSC encoder at the transmitter with a generator polynomial $(23, 35)_8$ $(K = 5)$. The coded bit stream is mapped to four-level pulse amplitude modulation signals. We considered a static channel model, $\mathbf{H}(z) = 0.407z + 0.815 + 0.407z^{-1}$, and hence 16 states exist in each SISO equalizer and decoder. The conventional and proposed architectures were designed in VHDL, then synthesized via the *Synopsys Design Compiler*, and placed and routed via the *Cadence Silicon Ensemble* by using 2.5-V 0.25-μm CMOS standard cell library. The core layouts are shown in Fig. 19. The area and the critical path delay of each architecture are summarized in Table III, where the parallel architecture occupies two times area of Fig. 19(a). The throughput is improved by 1.79 while 25% less area is consumed compared with the conventional approach. The measured normalized area matches well with results of Fig. 18.

VI. CONCLUSION

In this paper, we have presented BIP and symbol-based BIP architectures as being area efficient and providing high-throughput. Further, we synthesized the VLSI architectures in a 2.5-V 0.25-μm CMOS process and demonstrated that the proposed architectures achieved the dual goals of high throughput and area- efficiency for turbo decoders and turbo equalizers.

Future work needs to be directed toward further improving the area-efficiency and throughput gains exploiting circuit-level optimization techniques. Developing efficient interleaver architectures is of interest. New architectures based on factor-graph representations of iterative decoders are also a promising area of research.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

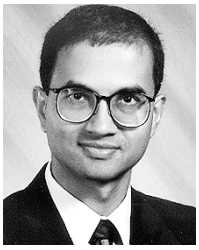
[2] S. Benedetto *et al.*, "A soft-input soft-output maximum *a posteriori* (MAP) module to decode parallel and serial concatenated codes," Jet Propulsion Lab., Pasadena, CA, TDA Progress Rep. 42-127, 1996.

[3] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[4] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, pp. 507–511, Sep.–Oct. 1995.
[5] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Sep. 1997, pp. 96–102.
[6] M. Tüchler, R. Koetter, and A. Singer, "Turbo-equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 754–767, May 2002.
[7] T. Souviginer *et al.*, "Turbo decoding for PR4: Parallel versus serial concatenation," in *Proc. IEEE Int. Conf. Communications*, Vancouver, Canada, Jun. 1999, pp. 1638–1642.
[8] Z. Wu and J. M. Cioffi, "Turbo decision aided equalization for magnetic recording channels," in *Proc. Global Telecommunication Conf.*, 1999, pp. 733–738.
[9] X. Wang and H. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.
[10] Japan's Proposal for Candidate Radio Transmission Technology on IMT-2000: W-CDMA. Japan Ass. of Radio Industries and Business (ARIB). [Online]. Available: http://www.arib.or.jp/IMT-2000/proponent
[11] "Telemetry Channel Coding," Consultative committee for space data systems (CCSDS), Blue Book 101.0-B-4, 1999.
[12] "Multiplexing and Channel Coding (FDD) (3G TS2 5.212 Version 3.3.0, Release 1999)," Universal Mobile Telecommunications Systems (UMTS), http://www.etsi.org, 1999.
[13] C. Douillard, M. Jezequel, C. Berrou, N. Brengarth, J. Tousch, and N. Pham, "The turbo codec standard for DVB-RCS," presented at the 2nd Int. Symp. Turbo Codes and Related Topics, Brest, France, Sep. 2000.
[14] Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI implementation issues of turbo decoder design for wireless applications," *Proc. IEEE Signal Processing Systems (SiPS): Design and Implementation*, pp. 503–512, Oct. 1999.
[15] D. Garrett, B. Xu, and C. Nicol, "Energy efficient turbo decoding for 3G mobile," in *Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED'01)*, Huntington Beach, CA, 2001, pp. 328–333.
[16] O. Leung, C. Yue, C. Tsui, and R. Cheng, "Reducing power consumption of turbo code decoder using adaptive iteration with variable supply voltage," in *Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED'99)*, San Diego, CA, 1999, pp. 36–41.
[17] A. Matache, S. Dolinar, and F. Pollara, "Stopping rules for turbo decoders," Jet Propulsion Lab., Pasadena, CA, TDA Progress Rep. 42-142, 2000.
[18] P. H. Wu and S. M. Pisuk, "Implementation of a low complexity, low power, integer-based turbo decoder," in *Proc. Global Telecommunications Conf.*, vol. 2, 2001, pp. 946–951.
[19] S. Lee, N. Shanbhag, and A. Singer, "Low-power turbo equalizer architecture," *Proc. IEEE Signal Processing Systems (SiPS): Design and Implementation*, pp. 33–38, Oct. 2002.
[20] D. Garrett and M. Stan, "Low power architecture of the soft-output Viterbi algorithm," in *Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED'98)*, Monterey, CA, 1998, pp. 262–267.
[21] G. Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural strategies for low-power VLSI turbo decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 3, pp. 279–285, Jun. 2002.
[22] C. Schurgers, F. Catthoor, and M. Engels, "Energy efficient data transfer and storage organization for a MAP turbo decoder module," in *Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED'99)*, San Diego, CA, 1999, pp. 76–81.
[23] 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.
[24] M. M. Mansour and N. R. Shanbhag, "Low-power VLSI decoder architecture for LDPC codes," in *Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED'02)*, Monterey, CA, 2002, pp. 284–289.
[25] Z. Wang, Z. Chi, and 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.
[26] J. Hsu and C. Wang, "A parallel decoding scheme for turbo codes," in *Proc. IEEE Int. Conf. Circuits and Systems*, vol. 4, 1998, pp. 445–448.
[27] B. Bougard, A. Ciulietti, L. V. d. Perre, and F. Catthoor, "A class of power efficient VLSI architectures for high speed turbo-decoding," in *Proc. IEEE Global Telecommunication Conf.*, vol. 1, 2002, pp. 553–549.
[28] B. Bougard *et al.*, "A scalable 8.7 nJ/bit 75.6 Mb/s parallel concatenated convolutional (turbo-) codec," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2003, pp. 152–153.
[29] A. Worm, H. Lamm, and N. Wehn, "A high-speed MAP architecture with optimized memory size and power consumption," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS2000)*, 2000, pp. 265–274.
[30] ——, "Design of low-power high-speed maximum *a priori* decoder architectures," in *Proc. Design, Automation and Test in Eur. Conf. Exhibition*, Apr. 2001, pp. 258–265.
[31] M. M. Mansour and N. R. Shanbhag, "Design methodology for high-speed iterative decoder architectures," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, 2002, pp. 3085–3088.
[32] ——, "VLSI architectures for SISO-APP decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 4, pp. 627–650, Aug. 2003.
[33] A. Giulietti *et al.*, "Parallel turbo code interleavers: Avoiding collisions in accesses to storage elements," *Electron. Lett.*, vol. 38, no. 5, pp. 232–234, Feb. 2002.
[34] T. Miyauchi, K. Yamamoto, and T. Yokokawa, "High-performance programmable SISO decoder VLSI implementation for decoding turbo codes," in *Proc. IEEE Global Telecommunications Conf.*, vol. 1, 2001, pp. 305–309.
[35] M. El-Assal and M. Bayoumi, "A high-speed architecture for MAP decoder," *Proc. IEEE Signal Processing Systems (SiPS): Design and Implementation*, pp. 69–74, Oct. 2002.
[36] M. Bickerstaff, L. Davis, C. Thomas, D. Garret, and C. Nicol, "A 24 Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless," *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp. 150–151, 2003.
[37] E. Yeo, S. Augsburger, W. R. Davis, and B. Nikolic, "Implementation of high throughput soft output Viterbi decoders," *Proc. IEEE Signal Processing Systems (SiPS): Design and Implementation*, pp. 146–151, Oct. 2002.
[38] F. Cathoor, S. Wuytack, E. de Greef, F. Balasa, L. Nachtergaele, and A. Vandecapelle, *Custom Memory Management Methodology, Exploration of Memory Organization for Embedded Multimedia System Design*. Norwood, MA: Kluwer, 1998.
[39] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.
[40] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.
[41] L. Bahl *et al.*, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
[42] J. Vogt, K. Koora, A. Finger, and G. Gettweis, "Comparison of different turbo decoder realization for IMT-2000," *Proc. IEEE Globecom*, pp. 2704–2708, 1999.
[43] G. Bauch and V. Franz, "A comparison of soft-in/soft-out algorithms for turbo detection," in *Proc. IEEE Int. Conf. Telecommunications (ICT'98)*, Jun. 1998, pp. 259–263.

Seok-Jun Lee (S'96–M'04) was born in Pusan, Korea, in 1973. He received the B.S. and M.S. degrees from the School of Electrical Engineering, Seoul National University (SNU), Seoul, Korea, in 1996 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC) in 2004.

During the summer of 2001 and 2002, he worked at the Communication Systems Laboratory, DSP Solution R&D Center, Texas Instruments Inc., Dallas, TX, where he was involved in IEEE 802.11 and 802.16 modem development. Since September 2004, he has been working on DVB-H and ISDB-T modem development at the Communication Systems Laboratory, DSP Solution R&D Center, Texas Instruments Inc., Dallas, TX. His research interests include VLSI architectures for communication and signal processing systems.



Naresh R. Shanbhag (M'88–SM'00) received the B.Tech. degree from the Indian Institute of Technology, New Delhi, India, the M.S. degree from the Wright State University, Dayton, OH, and the Ph.D. degree from the University of Minnesota, Minneapolis, in 1988, 1990, and 1993, respectively, all in electrical engineering.

From 1993 to 1995, he worked at AT&T Bell Laboratories, Murray Hill, NJ, where he was the Lead Chip Architect for AT&T's 51.84 Mb/s transceiver chips over twisted-pair wiring for ATM-LAN and VDSL chip-sets. Since August 1995, he has been with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, where he is presently a Professor. His research interests are in the design of integrated circuits and systems for broadband communications including low-power/high-performance VLSI architectures for error-control coding, equalization, as well as digital integrated circuit design. He has published more than 90 journal articles, book chapters, and conference publications in this area and holds three U.S. patents. He is also a coauthor of the research monograph *Pipelined Adaptive Digital Filters* (Norwood, MA: Kluwer, 1994).

Dr. Shanbhag received the 2001 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award, the 1999 IEEE Leon K. Kirchmayer Best Paper Award, the 1999 Xerox Faculty Award, the National Science Foundation CAREER Award in 1996, and the 1994 Darlington Best Paper Award from the IEEE Circuits and Systems Society. Since July 1997, he has been a Distinguished Lecturer for the IEEE Circuits and Systems Society. From 1997 to 1999 and from 1999 to 2002, he served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, respectively. He has served on the technical program committees of various conferences.

Andrew C. Singer (S'92–M'95–SM'05) was born in Akron, OH, in 1967. He received the S.B., S.M., and Ph.D. degrees, all in electrical engineering and computer science, from the Massachusetts Institute of Technology, Cambridge, in 1990, 1992, and 1996, respectively.

Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently an Associate Professor in the Electrical and Computer Engineering Department, and a Research Associate Professor in the Coordinated Science Laboratory. During the academic year 1996, he was a Post-Doctoral Research Affiliate in the Research Laboratory of Electronics at MIT. From 1996 to 1998, he was a Research Scientist at Sanders, A Lockheed Martin Company, Manchester, NH. His research interests include statistical signal processing and communication, universal prediction and data compression, and machine learning.

Prof. Singer was a Hughes Aircraft Masters Fellow and was the recipient of the Harold L. Hazen Memorial Award for excellence in teaching in 1991. In 2000, he received the National Science Foundation CAREER Award and in 2001 he received the Xerox Faculty Research Award. He is currently a member of the MIT Educational Council, Eta Kappa Nu, and Tau Beta Pi.