AREA-EFFICIENT HIGH-THROUGHPUT VLSI ARCHITECTURE FOR MAP-BASED TURBO EQUALIZER

Seok-Jun Lee, Naresh R. Shanbhag, and Andrew C. Singer

Coordinated Science Laboratory, ECE Dept.
University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, IL 61801
Email: [slee6,shanbhag,acsinger]@uiuc.edu

ABSTRACT

We present an area-efficient MAP-based turbo equalizer VLSI architecture by proposing a symbol-based soft-input soft-output (SISO) kernel which processes one multi-bit symbol in every clock cycle. The symbol-based SISO hardware can be shared by the equalizer and decoder, thereby reducing silicon area. Further, by introducing block-interleaved computation in the add-compare-select recursions, the critical path delay is reduced thereby improving throughput. Experimental results with QPSK modulation and $K = 3$ encoder demonstrate that the proposed area-efficient architecture achieves area savings of 47% with 11% throughput gain in 0.25 μm CMOS process. It is also shown that the throughput is improved by 79% via block-interleaved computation with an area savings of 25%.

1. INTRODUCTION

The turbo decoding technique has found numerous applications in decoding of turbo codes [1], turbo equalization [2], and low-density parity check codes (LDPC) [3]. The soft-input soft-output (SISO) module is a core computational kernel for turbo decoding. Thus, efficient implementation of SISO algorithms is of interest. Extensive research has already be done on implementations of turbo code decoders and turbo equalizers [4]–[12]. These include low-power design [4]–[7], memory optimization [8]–[9], and high-throughput implementation, [10]–[12].

Suppose a quadrature phase shift keying (QPSK) modulation and recursive systematic convolutional (RSC) encoder is employed with block-based transmission (see Fig. 1). Then, in each iteration, the maximum *a posteriori* (MAP) SISO equalizer processes one QPSK symbol, but the SISO MAP decoder decodes one bit. Therefore, two different hardware platforms are required for the equalizer and decoder architectures as they are designed for different trellis structures. Furthermore, there is inefficient hardware utilization since the block-based equalization and decoding are carried out alternatively between the two SISO blocks. The SISO decoder is in an idle state while the SISO equalizer is computing reliability values on each transmitted symbol in a block and vice and versa as shown in Fig. 2. In this paper, we present a symbol-based SISO architecture which can be utilized by both the equalizer and the decoder, thereby reducing silicon area. Further, it is expected that the critical path delay will increase since the symbol-based architecture processes multiple bits within a clock cycle. In order to reduce the critical path delay, we apply block-interleaving computation to the add-compare-select (ACS) recursion thereby achieving high-throughput architecture at the expense of reduced area savings. Thus, the block-interleaved symbol-based SISO decoder architectures enable us to trade-off area with throughput. But in all cases, the proposed architecture has significantly better area and throughput compared to existing turbo equalizer architectures.

The rest of this paper is organized as follows. Section 2

This material is based upon work supported by the National Science Foundation under Grant CCR 99-79381 and ITR 00-85929.

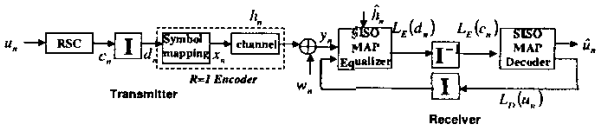


Fig. 1. Transmitter and receiver model in MAP-based turbo equalizer, where $\mathbf{I}$ and $\mathbf{I}^{-1}$ denote block interleaving and de-interleaving, respectively.

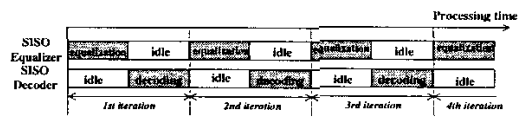


Fig. 2. Decoding flow of the conventional turbo equalizer implementation.

gives a brief description of the MAP-based turbo equalizer and SISO algorithm for equalization and decoding. Then, in Section 3, the proposed symbol-based SISO decoding scheme is derived and its VLSI architecture is described. In Section 4, the architectural performance of the proposed method is evaluated in a 0.25 μm CMOS process when it is applied to turbo equalizer implementation.

2. REVIEW ON MAP-BASED TURBO EQUALIZER.

In the MAP-based turbo equalizer [2], the intersymbol interference (ISI) channel and symbol mapping (modulation) block in Fig. 1 are regarded as a rate one code which is serially concatenated to the channel encoder Thus, the ISI-channel can be decoded by a SISO equalizer. We can use the same SISO algorithm for equalization and decoding and apply the turbo principle [1] [2] to iteratively equalize and decode the transmitted information as shown in the receiver model of Fig. 1.

The goal of SISO algorithm is to estimate the *a posteriori* log-likelihood ratio (LLR) value as

$$L_E(d_k) = \ln \frac{P(d_k = 1|\mathbf{y})}{P(d_k = 0|\mathbf{y})}$$
$$L_D(u_k) = \ln \frac{P(u_k = 1|L_{E1}(c_1), L_{E1}(c_{11}), \cdots, L_{E1}(c_{M-1}))}{P(u_k = 0|L_{E1}(c_1), L_{E1}(c_{11}), \cdots, L_{E1}(c_{M-1}))},$$

where $d_\pi = \Pi(c_\pi)$ (interleaved version of c_π), E and D denote equalization and decoding, respectively, $\mathbf{y}$ is the observed channel sequence, and N is the transmission block size. Numerous algorithms [13] exist that can be employed to estimate LLR values. The sliding window log-domain MAP algorithm is popular as it minimizes the metric storage requirements. The forward metric ($a_k(s)$), the backward metric ($b_k(s)$), and branch metric ($\gamma_k(s', s)$) are as [9]:

$$\gamma_k(s', s) = \ln p(q_k|s, s') + \ln p(s|s') \quad (1)$$
$$a_k(s) = \max_{s'}[a_{k-1}(s') + \gamma_k(s', s)] \quad (2)$$
$$b_{j-1}(s') = \max_{s}[b_j(s) + \gamma_j(s', s)], \quad (3)$$

where s and s' are trellis states and the max* operation is implemented as

$$\max^*(x, y) \approx \max(x, y) + \ln(1 + e^{-|x-y|}). \quad (4)$$

Note y_k in (1) equals the channel output y_k for the equalizer and it equals $L_{E1}(c_\pi)$ for the decoder. The LLR of the k-th trellis section data v_k is approximated as

$$L(v_k) \approx \max^*_{s',s:v_k=1}[a_{k-1}(s') + \gamma_k(s', s) + b_k(s)] - \max^*_{s',s:v_k=0}[a_{k-1}(s') + \gamma_k(s', s) + b_k(s)], \quad (5)$$

where $u_k = d_k$ for the equalizer and $v_k = u_k$ for the decoder. In a turbo equalizer algorithm, the updated LLRs are passed to the next SISO block after being deinterleaved or interleaved.

Since the equalizer takes symbols received from a channel and LLR values on each bit from SISO decoder, the branch metric of SISO equalizer is computed as

$$\gamma_k(s', s) = -\frac{1}{2\sigma_w^2}|y_k - \sum_{i=0}^{M-1} h_i x_{k-1}|^2 + \frac{1}{2}\sum_{i=1}^{r-1} L_D(d_k)(2d_k - 1), \quad (6)$$

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

$$\gamma_k(s', s) = \frac{1}{2}\sum_{i=0}^{m} L_E(c_k)(2c_i - 1), \quad (7)$$

where the code rate (R) is $\frac{m}{m+1}$. Further, note that the trellis of SISO equalizer is traversed by one symbol step while that of SISO decoder is processed at the bit-level.

3. SYMBOL-BASED DECODING

In this section, we derive a symbol-based decoding scheme for binary RSC, describe the proposed area-efficient VLSI architecture, and introduce the block-interleaved computation to reduce the critical path delay. The predicted area saving and throughput improvement are also provided.

3.1. Derivation

In order to share a hardware platform for both SISO equalizer and decoder, bit-level operations in SISO decoding need to be transformed into symbol operations. In this paper, we apply a look-ahead transform to the trellis of the binary encoder so that ACS recursion may be carried out on multiple trellis sections [14]. For simplicity, a 2-bit symbol decoding, where two trellis sections are grouped (see Fig. 3), is described. Extension to multi-bit symbol cases is straightforward. To compute the LLR $L(v_k)$ for 2-bit symbol decoding, we substitute a_{k-1} with x_{k-2} in (5) as follows

$$L(u_k) = \max^*_{s'',s:v_k=1}[\max^*_{s'}[a_{k-2}(s'') + \gamma_k(s'', s') + \gamma_k(s', s) + b_k(s)] - \max^*_{s'',s:v_k=0}[\max^*_{s'}[a_{k-2}(s'') + \gamma_k(s'', s') + \gamma_k(s', s) + b_k(s)], \quad (8)$$

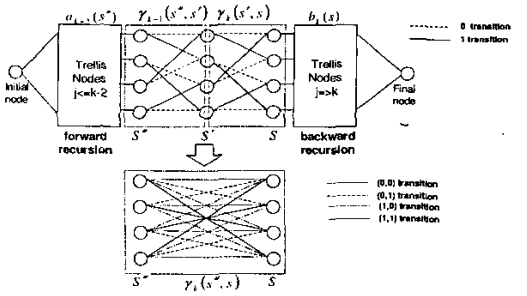


Fig. 3. Proposed symbol-based decoding scheme for an example with $[5, 7]_8$ RSC encoder.

and since max* operation is associative,

$$L(v_k) = \max^*_{s',s:v_k=1}[a_{k-2}(s') + \gamma_k(s', s) + b_k(s)] - \max^*_{s',s:v_k=0}[a_{k-2}(s') + \gamma_k(s', s) + b_k(s)], \quad (9)$$

where $\gamma_k(s', s)$ is the branch metric from s' to s. Similarly, $L(v_{k-1})$ can be computed as

$$L(v_{k-1}) = \max^*_{s'',s:v_{k-1}=1}[a_{k-3}(s'') + \gamma_{k-1}(s'', s) + b_k(s)] - \max^*_{s'',s:v_{k-1}=0}[a_{k-3}(s'') + \gamma_{k-1}(s'', s) + b_k(s)]. \quad (10)$$

The first term (10) can be expressed as

$$\max^*\{\max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{1,1}) + b_k(s)], \max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{1,0}) + b_k(s)]\}, \quad (11)$$

where $\tilde{v}_{(i,j)}$ corresponds to the transition with $v_{k-1} = i$ and $v_k = j$. In a similar manner, each term (9) and (10) is computed by grouping two bits as a symbol. By defining symbol reliability metrics ($\lambda_{v_{k-1}v_k}$) as

$$\begin{aligned}
\lambda_{10} &= \max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{1,0}) + b_k(s)] \\
\lambda_{11} &= \max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{1,1}) + b_k(s)] \\
\lambda_{10} &= \max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{0,0}) + b_k(s)] \\
\lambda_{11} &= \max^*_{s'',s}[a_{k-2}(s'') + \gamma_k(s'', s, \tilde{v}_{0,1}) + b_k(s)],
\end{aligned} \quad (12)$$

we can compute $L(v_k)$ and $L(v_{k-1})$ as shown below,

$$\begin{aligned}
L(v_{k-1}) &= \max^*(\lambda_{10}, \lambda_{11}) - \max^*(\lambda_{00}, \lambda_{01}) \\
L(v_k) &= \max^*(\lambda_{11}, \lambda_{01}) - \max^*(\lambda_{10}, \lambda_{00}).
\end{aligned} \quad (13)$$

Table 1. Logic requirement of SISO decoding where B_{bm}, B_{pm}, B_{λ}, and B_{buffer} are precisions of branch metrics, path metrics, received symbol(y), and LLR values.

	buffer
γ-unit	$3L(2B_\lambda + B_{LLR})$
β-unit	$2LN_s B_{pm}\frac{1}{2}$
	Adder
γ-unit	$3(3n-1)2^{2n}B_{bm}$
α,β-unit	$3N_s r B_{pm}$
Λ-unit	$2^{n+1}N_s B_{pm}$
	max*
α,β-unit	$3N_s(2^r-1)B_{pm}$
Λ-unit	$(2^n(N_s-1)+2r(2^{n-1}-1))B_{pm}$
	Latch
γ-unit	$2^{2n}B_{bm}$
α,β-unit	$3N_s B_{pm}$
Λ-unit	$(2^n(N_s-1)+2r(2^{n-1}-1))B_{pm}$

This is equivalent to (5) derived via bit-based decoding. In general, (13) can be rederived for n-bit symbol-based decoding,

$$L(v_j) = \max^*_{\forall\lambda:v_j=1}[\lambda_{b_{k-r+1}\cdots b_k}] - \max^*_{\forall\lambda:v_j=0}[\lambda_{b_{k-r+1}\cdots b_k}], \quad (14)$$

where $\lambda_{b_{k-r+1}\cdots b_k}$ is the symbol made up of bits from b_{k-r+1} to b_k and $j = k-r+1, \cdots, k$.

3.2. Area-Efficient VLSI Architecture

By employing n-bit symbol-based SISO processing kernel derived previously, we can implement both SISO equalizer and decoder on a single hardware platform leading to area saving. The proposed area-efficient VLSI architecture is depicted in Fig. 4. The architecture is composed of four units: γ-unit for computing branch metrics, α-unit for computing the forward metrics (a_k), β-unit for computing the backward metrics (b_k), and Λ-unit for computing LLRs ($L(v_k)$) [7], [9]. Note that two γ-units are needed for each equalizer and decoder to produce 2^{2r} different metrics. The ACS data-path of α,β-units and Λ-unit for decoding n-bit symbol are described in Fig. 5 and 6 for $n = 2$. Since the four, γ, α, β, and Λ units are implemented using 2-input adder, 2-input max*, latches, and internal buffers, we can analyze the hardware complexity in term of precisions, the number of bits in a symbol (r), the number of states in a trellis (N_s), and the processing window size (L) [9]. Table 1 summarizes results. As n is increased, the logic complexity increases exponentially. However, the backward metric buffer size will be decreased by a factor of $\frac{1}{n}$. Note the size of the delay line buffer is independent of r.

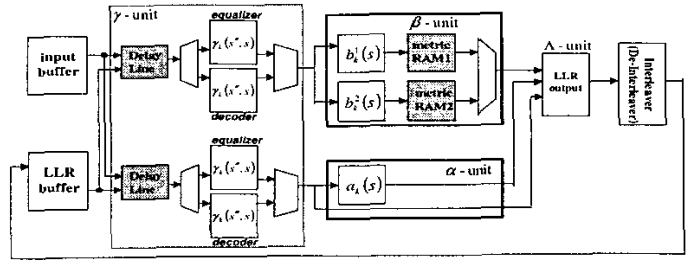


Fig. 4. Proposed area-efficient VLSI architecture.

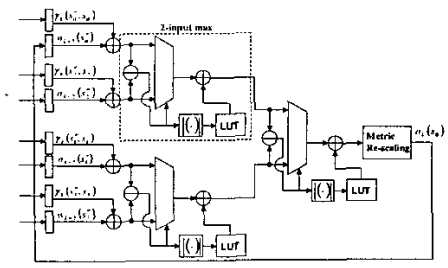


Fig. 5. ACS recursion in symbol-based decoding.

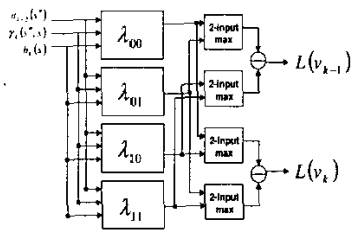


Fig. 6. Λ-unit in symbol-based decoding.

3.3. High-Throughput Architecture

The critical path delay of the symbol-based decoding architecture may be increased as shown in Fig. 5. In order to reduce the delay caused by the look-ahead transform, a block-interleaved computation method is exploited leading to a pipelined ACS data-path at the cost of marginal buffer area increase. Since SISO block processing in turbo equalization satisfies three properties: 1.) computation between blocks are independent, 2.) computations between sub-blocks within a block are independent, and 3.) computation within a sub-block is recursive, the recursive loop delay of ACS can be reduced via interleaved computation, folding, and retiming transform. Consider the recursive architecture in Fig. 7. Note that the architecture in Fig. 7 cannot be easily pipelined or processed in parallel due to the presence of the feedback loop. However, if the processing

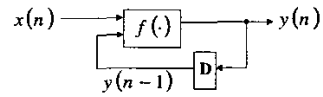


Fig. 7. Recursive computation.

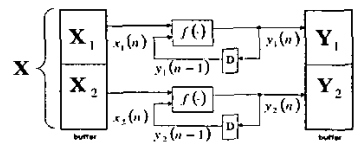


Fig. 8. Two level parallel block architecture for block-independent computation.

is block-independent, and the computations between sub-blocks within a block are independent, then one can parallelize the architecture as shown in Fig. 8.

If we now fold the parallel architecture in Fig. 8 by a factor of 2, we obtain the folded block-interleaved architecture. Note that the folded block-interleaved architecture is inherently pipelined. Therefore, an application of retiming (see Fig. 9) results in reduction of the critical path delay by a factor of two over that of the original architecture in Fig. 7.

SISO decoders can exploit the property that the forward and backward metrics α_k and β_k converge after a few constraint lengths (K) have been traversed in the trellis, independent of the initial conditions [9]. By using this property, one block can be segmented into several sub-blocks as shown in Fig. 8 [11]. Thus, the symbol-based decoding can reduce the critical path delay via interleaved computation

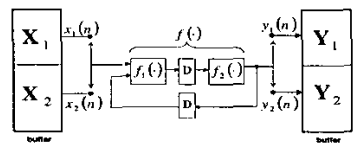


Fig. 9. Retimed folded block-interleaved architecture.

Table 2. Complexity for high-throughput VLSI architecture. The adder and max* computations remain the same.

	buffer
γ-unit	$3(r-1)(2B_A + B_{LLR})$
β-unit	$2(r-1)N_s B_{pm}$
	Latch
α, β-unit	$3N_s(2^\nu - 1)B_{Pm}$

leading to high-throughput implementation. The extra hardware complexity due to pipelining is summarized in Table 2. Only buffer size and the number of latches are affected.

3.4. Area Saving and Throughput Improvement

Based on the analysis results in Table 1 and 2, we can predict the area savings of the proposed area-efficient MAP-based turbo equalizer over the conventional approach. The standard cell design is considered. Actual areas of a 1-bit adder, 1-bit latch, max*, and 1-bit buffer cell are obtained from a 0.25μm CMOS standard cell library, and precisions are determined via computer simulations ($B_{LLR} = 6, B_A = 8, B_{bm} = 8$, and $B_{pm} = 12$). The interconnection area is not included in this analysis. The area savings can be expressed as

$$A_{saving} = \frac{A_E + A_D - A_P}{A_E + A_D} \times 100\%, \tag{15}$$

where A_P are the area of the proposed area-efficient architectures and A_E and A_D are the area of SISO equalizer and decoder. Since $A_E \approx \max(A_E, A_D)$,

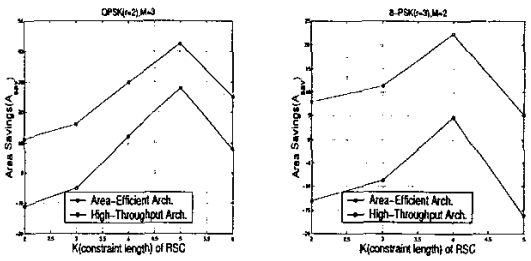
$$A_{saving} \approx \frac{1}{1 + \frac{\max(A_E, A_D)}{\min(A_E, A_D)}} \times 100\%, \tag{16}$$

and the area savings are plotted in Fig. 10 for two examples where SISO equalizer is considered for QPSK modulation with the channel length $M = 3$ ($4^{M-1} = 16$ states) and 8-PSK modulation with the channel length $M = 2$ ($8^{M-1} = 8$ states). It is observed that the area savings are maximized when the number of states in the equalizer and decoder is equal.

The processing time required for one block iteration of equalization and decoding for the convolutional architecture is:

$$T_B = \frac{N}{n}\tau_E + NR\tau_D, \tag{17}$$

where, τ_E and τ_D are the critical path delays of SISO equalizer and decoder, respectively and R is the code rate. Assuming $\tau_E = n\tau_D$, the throughput gain is the ratio (η) of



(a) QPSK (b) 8-PSK

Fig. 10. Predicted area savings for QPSK and 8-PSK.

Table 3. Area after place & route and critical path delay measured via *Synopsys Pathmill*.

	Area(mm^2)	Delay($nsec$)
Conventional	24.39	$\tau_E = 28.513$ $\tau_D = 20.315$
Area-Efficient	12.81	$\tau_P = \tau_D = 28.513$
High-Throughput	18.34	$\tau_P = \tau_D = 17.713$

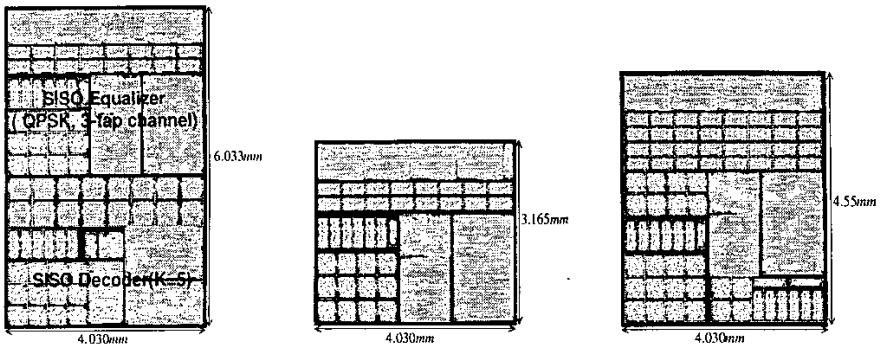
T_B of the conventional approach over the proposed high-throughput architecture,

$$\begin{aligned}\eta &= \frac{\frac{N}{n}n\tau_D + NR\tau_D}{\frac{1}{n}(N + 2(r-1)L + NR)\tau_P} \\ &= \frac{N(1+R)}{(N+2(r-1)L+NR)}\frac{n\tau_D}{\tau_P},\end{aligned} \tag{18}$$

where $2(r-1)L$ extra cycles are necessary because of block-interleaved computation and τ_P is the critical path delay of the proposed high-throughput architecture. Note that the throughput gain, η, becomes larger as τ_P gets closer to $n\tau_D$.

4. EXPERIMENTAL RESULTS AND DISCUSSION

We employ a RSC encoder at the transmitter with a generator polynomial $(23, 35)_8$. The coded bit stream is mapped to 4-level pulse amplitude modulation signals. We considered a static channel model, $H(z) = 0.407z + 0.815 + 0.407z^{-1}$, and hence 16 states exist in each SISO equalizer and decoder. The conventional and proposed architectures are designed in VHDL, synthesized via *Synopsys Design Compiler*, and placed and routed via *Cadence Silicon Ensemble* by using a TSMC 0.25 μm CMOS standard cell library. The layouts are shown in Fig 11 and the area and the critical path delay of each architecture are summarized in Table 3. We see that the area-efficient architecture results in 47% area savings, while the high-throughput architecture



(a) Conventional architecture

(b) Area-efficient architecture

(c) High-throughput architecture

Fig. 11. Area after place and route.

provides 25% area savings. Those values are very close to those predicted by (15) and Table 1 and 2 as can be seen in Fig. 10 (a). Note further that the high-throughput architecture achieves 79% improvement in throughput while the area-efficient architecture results in a 11% improvement in throughput. Thus, the block-interleaved symbol-based SISO decoder architectures enable us to trade-off area with throughput.

5. REFERENCES

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

[2] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization,"

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

[4] Z. Wang, H. Suzuki, and K. K. Parhi, "VLSI implementation issues of turbo decoder design for wireless applications," in *Proc. of IEEE Signal Processing Systems(SiPS): Design and Implementation*, October 1999, pp. 503–512.

[5] D. Garrett, B. Xu, and C. Nicol, "Energy efficient turbo decoding for 3G mobile," in *Proc. of IEEE Int. Symp. Low Power Electronics Design (ISLPED'01)*, Huntington Beach, CA, 2001, pp. 328-333.

[6] 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. of IEEE Int. Symp. Low Power Electronics Design (ISLPED'99)*, San Diego, CA, 1999, pp. 36–41.

[7] S. Lee, N. Shanbhag, and A. Singer, "Low-power turbo equalizer architecture," in *Proc. of IEEE Signal Processing Systems(SiPS): Design and Implementation*, October 2002, pp. 33–38.

[8] C. Schurgers, F. Catthoor, and M. Engels, "Energy efficient data transfer and storage organization for a MAP turbo decoder module," in *Proc. of IEEE Int. Symp. Low Power Electronics Design (ISLPED'99)*, San Diego, CA, 1999, pp. 76–81.

[9] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE Journal on Selected areas in comm.*, vol. 16, no. 2, pp. 260–264, Feb. 1998.

[10] Z. Wang, Z. Chi, and K. Parhi, "Area-efficient high speed decoding schemes for turbo/MAP decoders," in *Proc. 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001, vol. 4, pp.2633 – 2636.

[11] J. Hsu and C. Wang "A parallel decoding scheme for turbo codes," in *Proc. of 1998 IEEE International Conference on Circuits and Systems*, 1998, vol. 4, pp.445 – 448.

[12] M. M. Mansour and N. R. Shanbhag, "Design methodology for high-speed iterative decoder architectures," in *Proc. 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, vol. 3, pp.3085 – 3088.

[13] G. Bauch and V. Franz, "A comparison of soft-in/soft-out algorithms for turbo detection," in *Proc. of IEEE Int. Conf. on Telecommunications(ICT'98)*, June 1998, pp. 259–263.

[14] G. Fettweis and H. Meyr, "High-speed parallel Viterbi decoding: Algorithm and VLSI-Architecture" *IEEE Comm. Magazine*, pp. 46-55, May 1991.