

High Speed Max-Log-MAP Turbo SISO Decoder Implementation Using Branch Metric Normalization

J. H. Han¹, A. T. Erdogan^{1,2}, T. Arslan^{1,2}

¹University of Edinburgh, School of Engineering and Electronics

²Institute of System Level Integration, The ALBA campus

j.han@ed.ac.uk, Ahmet.Erdogan@ee.ed.ac.uk, Tughrul.Arslan@ee.ed.ac.uk

Abstract

The authors present a turbo soft-in soft-out (SISO) decoder based on Max-Log maximum a posteriori (ML-MAP) algorithm implemented with sliding window (SW) method. A novel technique based on branch metric normalization is introduced to improve the speed performance of the decoder. The turbo decoder with the proposed technique has been synthesized to evaluate its power consumption and area usage using a 0.18 μ m standard CMOS cell library. It is shown that while power consumption and area usage change slightly with our technique, it achieves up to 58% speed-up compared to a conventional SISO decoder architecture.

1. Introduction

Turbo codes have shown their outstanding performance in terms of bit error rate (BER) at very low signal-noise ratio (SNR) since they appeared in the early 1990s [1]. For this reason, they have been adopted in various standards for wireless communication systems, such as DVB-RCS, IEEE 802.16, 3GPP UMTS, and CCSDS [2]. When the turbo codes were introduced by Berrou et. al in 1993, their decoding complexity was very high for them to be efficiently implemented in hardware when compared with a decoder for convolutional codes like a Viterbi decoder. Two-step Soft output Viterbi algorithm (SOVA) decoder [3] offers a low complexity solution to the turbo decoder. However, the SOVA turbo decoder has less BER performance than the MAP turbo decoder for the same SNR. The MAP algorithm provides the best performance in BER while its complexity is higher than the two-step SOVA. Many researchers have studied the MAP algorithm to simplify its hardware implementation as well as to

improve its performance. Log MAP and ML-MAP algorithms [4] have reduced the implementation complexity. Since the Log MAP and ML-MAP algorithms have been proposed, they are widely used to implement the turbo decoders. Furthermore, the sliding window (SW) method [5] provides efficient area usage by dividing a block of input symbols into a number of sub-blocks. Some earlier works for the implementation of turbo decoders are reported in [6-9]. However, they have contributed less to improve the speed of turbo decoders. High-speed downlink packet access (HSDPA) in 3GPP standard requires more than 10Mbits/s. Even high data rates are required for wireless local area network systems (LAN) [10]. Therefore, the speed is one of the key factors beside the power consumption and area usage for efficient implementation of turbo decoders.

In this paper, we introduce a novel technique that can reduce the critical path of a turbo decoder. This is achieved by normalizing the branch metric values instead of normalizing the state metric values, as is the case in conventional implementations. The ML-MAP SISO decoder architecture with our proposed technique has been implemented to investigate its performance in terms of area usage, power consumption, and timing delay. The constraint length $K = 4$ and the window length 40 have been used. A conventional ML-MAP SISO decoder architecture has also been implemented in order to compare the performance of our proposed architecture.

2. Turbo decoder structure

A turbo decoder consists of two soft-input soft-output (SISO) decoders and one interleaver/deinterleaver between them. Decoding process in a turbo decoder is performed iteratively through the two SISO decoders via the interleaver and the deinterleaver. Figure 1 shows the structure of a

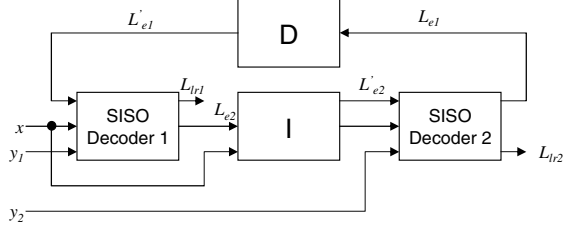


Figure 1. Turbo decoder structure.

turbo decoder, where I and D denote interleaver and deinterleaver, respectively. The input symbols, x and y_1 , and a priori value, L'_{e1} (initial value is zero), are used for decoding process in the SISO decoder 1 that produces log-likelihood ratio, L_{lr1} , and a priori value, L'_{e2} . Then, the input symbol data, x (via interleaver) and y_2 , and a priori value, L'_{e2} (interleaved value of L'_{e2}), are used in the SISO decoder 2 that produces L_{e1} for the SISO decoder 1 and soft-output value, L_{lr2} . Turbo decoder can achieve high performance in terms of BER at very low SNR with iterating these processes.

3. ML-MAP algorithm

The ML-MAP algorithm can be used for reduced complexity decoder implementation [4]. In this section we summarize the ML-MAP algorithm used to implement the decoder architecture in this paper. The decoding process in MAP algorithm performs calculations of the forward and backward state metric values to obtain the log likelihood ratio (LLR) values, which have the decoded bit information and reliability values. The LLR values are represented by the following equation :

$$L_{lr} = \ln \frac{\sum_{S_k} \sum_{S_{k-1}} \gamma_1(S_{k-1}, S_k) \alpha(S_{k-1}) \beta(S_k)}{\sum_{S_k} \sum_{S_{k-1}} \gamma_0(S_{k-1}, S_k) \alpha(S_{k-1}) \beta(S_k)} = L_1 - L_0 \quad (1)$$

$$L_1 = \ln \sum_{S_k} \sum_{S_{k-1}} \gamma_1(S_{k-1}, S_k) \alpha(S_{k-1}) \beta(S_k),$$

$$L_0 = \ln \sum_{S_k} \sum_{S_{k-1}} \gamma_0(S_{k-1}, S_k) \alpha(S_{k-1}) \beta(S_k)$$

where γ , α , and β represent the branch, forward, and backward state metric values, respectively. The subscript k and S denote time and state. The LLR value (L_{lr}) is calculated by the metric values at all states (S) of time k and $k-1$. The equation of γ , α , and β can be represented to logarithm form as shown below [4] :

$$\ln \gamma(S_{k-1}, S_k) = 1/2(L_c u'_k + L_c x u'_k + L_c y u'_k) \quad (2)$$

$$\ln \alpha(S_k) = \ln \sum_{S_{k-1}} \exp(\ln \gamma(S_{k-1}, S_k) + \ln \alpha(S_{k-1})) \quad (3)$$

$$\ln \beta(S_k) = \ln \sum_{S_{k+1}} \exp(\ln \gamma(S_k, S_{k+1}) + \ln \beta(S_{k+1})) \quad (4)$$

where the branch metric (γ) is calculated by the a priori information (L_e), channel reliability value (L_c), input symbols (x and y_1), the systematic bit (u_k^s) and the parity bit (u_k^p). As described in previous section, a priori information is obtained from the LLR value computed in previous decoding process after subtracting the input symbol data and a priori values from the LLR value. The MAP algorithm, which uses the above equations, is not suitable for hardware implementation due to the logarithm function. This problem can be addressed using a well know approximation, called Jacobi logarithm, which is given below

$$\ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \quad (5)$$

This approximation is used to implement the state metric unit (SMU) and LLR computation unit (LCU) in Log MAP and ML-MAP SISO decoder. The 2nd term of the right hand side in equation (5) a correction term which can be implemented through a simple look-up table [12]. Log-MAP algorithm includes this correction term. However, in this paper, we have implemented ML-MAP SISO decoder in which the correction term is ignored.

4. SW method decoding process

The conventional MAP decoding process has very high latency due to the processing of forward and backward calculations in all trellis states. Computing the LLR values requires the state metric values generated by the forward and backward processes. Therefore, a large memory size is required to store the state metric values, which in turn depends on the input data block size. To address this problem, the SW method, which uses a sub-block of input data, can be

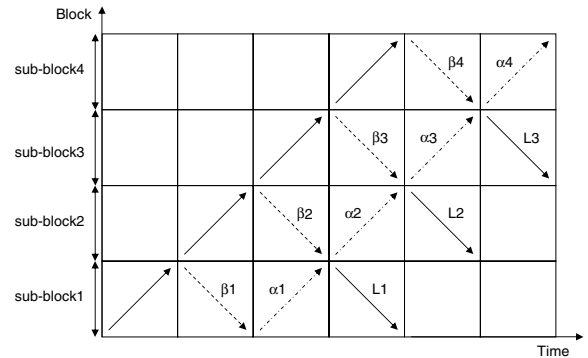


Figure 2. A graph of data flow with SW method.

used [5]. In the SW method, the sub-block size is fixed. Therefore, the SW method not only can reduce the memory size but can also be used with variable block sizes. In this paper the window length (sub-block size) is fixed to 40, which is the smallest block length in UMTS standard specification [12].

Figure 2 shows a graph of data flow for decoding process in time and block axis. The solid line without notation indicates input data stored into LIFO block. The dashed line with β represents the backward calculation process and the number identifies the sub-block. In fact, the backward process of dashed line is not to compute the LLR value, but to provide the state metric value for the backward process. The backward process for the LLR values is performed on the solid line with L. On the solid, line the backward process and LLR computation is executed simultaneously. The forward process, the dash-dotted line with α , is done before the backward and LLR computation. During the forward process the state metric values are stored into LIFO memory blocks. For this SW method several FIFO and LIFO memory blocks are used in the ML-MAP decoder. In the SW method the latency is 4 times of the window length.

5. ML-MAP SISO decoder architecture

The ML-MAP SISO decoder architecture presented in this paper consists of the forward and backward state metric, LLR computation, and memory (LIFO and FIFO) blocks. Figure 3 shows a block diagram of the SISO decoder architecture. LIFO and FIFO memory blocks are used to control the flow of input symbol data following the SW method given in Figure 2. In Figure 3, the LIFO 1 and 2, and the FIFO 1 and 2 are used to buffer the input data symbols. The LIFO 3 and 4 are to store the forward state metric and the LLR values, respectively. The SISO decoder has been built with two backward state metric units, $\beta 1$ and $\beta 2$, where $\beta 1$ (called ‘dummy logic’) is used to provide the state metric values into $\beta 2$, which generates the backward

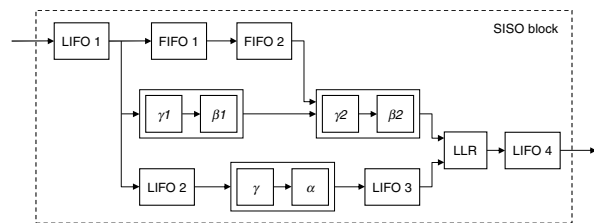


Figure 3. A block diagram of SISO decoder based on ML-MAP algorithm with the SW method.

state metric to compute the LLR values. The α and γ denote the forward state and branch metric units to calculate the forward state and branch metric values. The detailed structure of the main components of the SISO decoder is described below.

5.1 Branch and State Metric Units

The branch and state metric units (BMU and SMU) have been implemented following the ML-MAP algorithm using the Jacobi logarithm approximation shown in section 3. As shown in Figure 4 (a), the conventional BMU and SMU consists of branch metric calculation, add, compare, select, and normalization processes. In Figure 4 (a) input symbol data is computed to generate the branch metric values ($\gamma_{0,k}$, $\gamma_{1,k}$). The branch metric values are added to the state metric values (α_{k-1} (or β_{k+1}), α'_{k-1} (or β'_{k+1})) to generate the new state metric value (α_k (or β_k)) for the next cycle. The general SMU in turbo SISO decoder must include the normalization process to avoid overflow of the state metric values [6]. In Figure 4 (a), the branch metric values are obtained from input data symbols. Then, the branch metric and state metric values are used to make new state metric values through add, compare (C), select, and normalization (N) processes. These processes are performed in one clock cycle recursively. Therefore, critical path delay is determined by these processes. Figure 4 (b) and (c) show the recursive

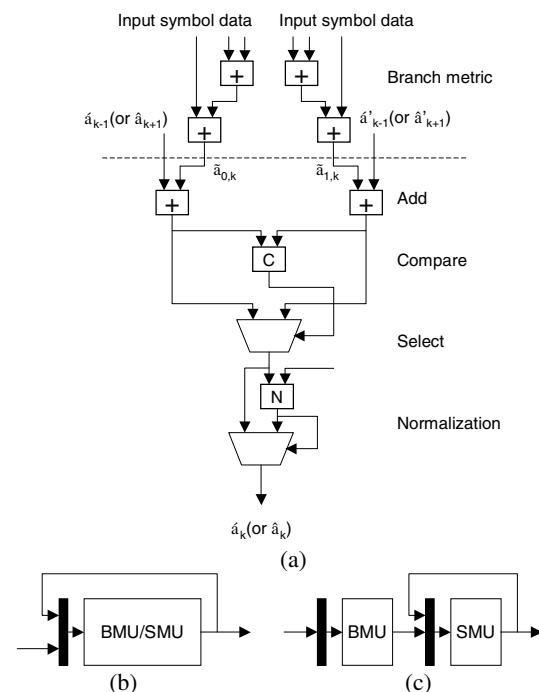


Figure 4. (a) Conventional structure of the branch and state metric units (b) without and (c) with pipelined.

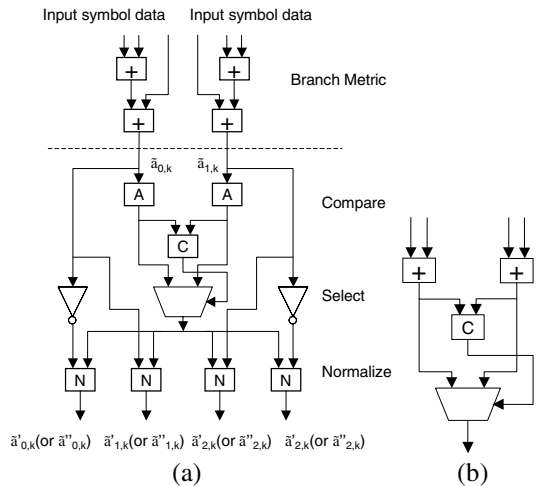


Figure 5. Our proposed structure of (a) BMU and (b) SMU.

process of BMU and SMU without and with pipelined method to reduce the critical path delay. However, the critical path delay can still be reduced further.

Figure 5 (a) and (b) show our proposed architecture of the BMU and the SMU. In our proposed architecture we have chosen to normalize the branch metric values instead of the state metric values as is the case in the conventional implementations. In Figure 5 (a), the generated branch metric values (γ_{0-1}) are converted into absolute values (A), which are then compared (C) to select the maximum or minimum branch metric value. The branch metric normalization (N) can be represented as following :

$$\gamma'(S'_{k-1}, S_k) = \gamma(S'_{k-1}, S_k) - \max(\gamma(S'_{k-1}, S_k)) \quad (6)$$

$$\gamma''(S'_{k-1}, S_k) = \gamma(S'_{k-1}, S_k) - \min(\gamma(S'_{k-1}, S_k)) \quad (7)$$

where γ is the branch metric values, and γ' and γ'' are the normalized values by maximum or minimum values of $\gamma - \max(\gamma(S'_{k-1}, S_k))$ or $\min(\gamma(S'_{k-1}, S_k))$. Therefore, the normalized value of γ' is always equal to zero or less than zero. Similarly, the normalized value of γ'' is always equal to zero or larger than zero. In our decoding process, these normalized branch metric values were used to compute state metric values. The output state metric values are used to determine whether it is normalized by minimum or maximum branch metric value. The decision rule is that if all state metric values are larger than zero, the maximum branch metric value is used for the normalization and if all state metric values are less than zero, the minimum branch metric value is used for the normalization. This normalization method leads to a simplified SMU, but also to a more complex BMU as shown in Figure 5. However, the

total hardware resources of BMU and SMU are not much different than the conventional architecture as shown in Table 1. More importantly, as can be seen in Figure 5 (a) and (b), our proposed architecture reduces the critical path delay significantly by eliminating the state metric normalization process used in the conventional SMU.

Table 1. Comparison of conventional and proposed BMU and SMU architectures for constraint length $K=4$.

	Conven.	Proposed
Input data word lengths	Signed 4bits	
Internal word lengths	Signed 10bits	
no. of adders in BMU	4	14
no. of adders in SMU	32	24

5.2 LLR Computation Unit

In order to compute the LLR values (L_I or L_O), forward (α_{0-7}) and backward (β_{0-7}) states, and branch metric (γ_{0-1}) values of all states are required as shown in Figure 6. The LLR computation unit (LCU) has a similar architecture to the SMU. In a conventional architecture, a total of 16 adders are used to obtain the LLR values for a decoder with 8 states. Then, a 3-stage compare and select process is used to identifying the max LLR value. This 3-stage process results in a long critical path delay. Therefore, the LCU can be pipelined in order to reduce the critical path delay, as shown in Figure 6. However, a high level of pipelining can increase area usage and power consumption due to increased number of pipeline registers.

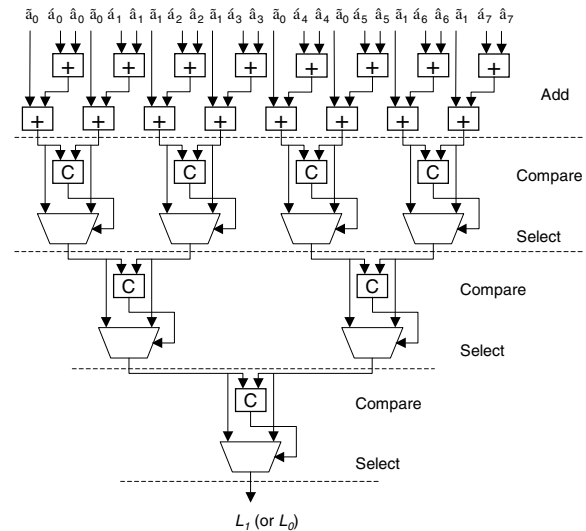


Figure 6. A conventional LCU structure.

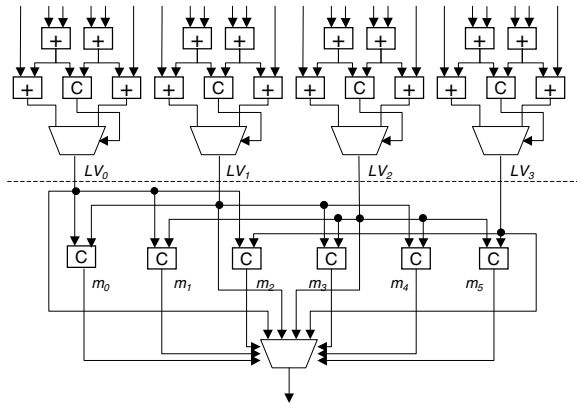


Figure 7. Our proposed LCU structure.

Our proposed LCU architecture is given in Figure 7. This architecture was developed to reduce the critical path delay without increasing power consumption and area usage. Our proposed architecture is based on two pipeline stages. In the first stage, four LLR values (one corresponding to each state) are obtained through four parallel add, compare, and select processes. However, note that in this architecture the comparisons are based on addition of forward and backward state metric values only, without addition of branch metric values. In the second stage the resulting four values ($LV_{0,3}$) are compared with each other using six comparators and a selector. Therefore, our architecture uses 3 additional comparators and replaces 3 small selectors with a larger one when compared with the conventional architecture, shown in Figure 6.

6. Simulation results

The ML-MAP turbo SISO decoder presented in this paper was initially simulated at high level to verify its functionality. In our simulation a 1024 size block type interleaver [13] has been used. The simulation results are shown in Figure 8. To simplify the simulation, AWGN channel model with variance 1, and BPSK modulation was assumed. Total number of source bits and the number of maximum iteration were 10M and 8, respectively.

After verifying the functionality with high-level simulations, the turbo decoder has been implemented using Verilog HDL at RTL level and synthesized for a 0.18um standard cell library using Synopsys DesignCompiler™. RTL and gate level simulations have been performed using Cadence Verilog-XL™.

Figure 9 (a) and (b) show the power and area results, where scheme 1, 2, and 3 represent the SISO decoder architectures of conventional without pipeline, conventional with pipeline, and our proposed

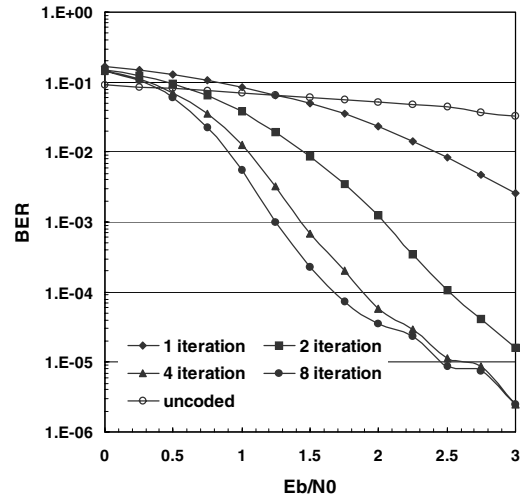
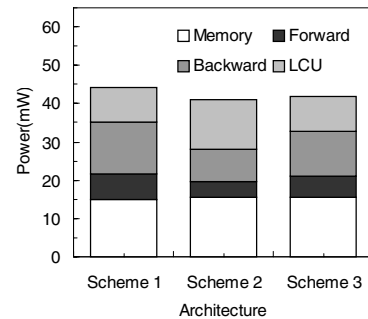
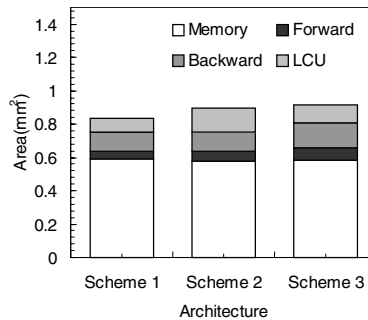


Figure 8. BER simulation results of our ML-MAP turbo decoder with constraint length $K=4$.

architecture with pipeline, respectively. Power results were obtained with Synopsys DesignPower™ for a clock speed of 50MHz. Our proposed architecture (scheme 3) has achieved small saving in total power consumption while scheme 1 has the least area usage. In conventional pipeline architecture (scheme 2), the components for backward and forward state metric consumed less power compared to scheme 1 and



(a)



(b)

Figure 9. (a) Power and (b) area simulation results of different architectures.

scheme 3. However, the LCU of scheme 2 consumed more power and area compared to scheme 1 and 3. In overall, our architecture (scheme 3) achieved 5% power saving with 9% area increase compared to scheme 1. On the other hand, when compared to scheme 2, our architecture had 2% increase in power consumption and area usage.

We have also measured the critical path delays for each scheme. As can be seen from Table 2, our proposed architecture has a critical path delay of only 2.85ns, resulting in 43% and 58% speed-up compared to the conventional architecture with and without pipeline methods, respectively. For scheme 1 the maximum delay was caused by LCU. However, for pipelined architectures (scheme 2 and 3) the maximum delay was due to the SMU.

Table 2. Power, area, and speed comparison of different architectures.

Architecture	Power(mW)	Area(mm ²)	Max. delay
Scheme 1	44.08	0.836	6.75ns (LCU)
Scheme 2	40.97	0.895	5.03ns (SMU)
Scheme 3	41.73	0.914	2.85ns (SMU)

7. Conclusions

The authors have presented a high speed ML-MAP turbo SISO decoder. A new technique was introduced for calculating the forward and backward state and branch metric values, where normalize operation was applied to branch metric values instead of to state metric values. In addition, a new architecture was introduced for performing LLR value computations. It has been shown that while our decoder achieved a slight reduction in power consumption and a slight increase in area usage, it has achieved 43% and 58% speed-up compared to pipelined and non-pipelined conventional decoder, respectively. Therefore, by adopting our technique, the turbo decoder can be applied to wireless communication systems requiring high data rates and low power consumption.

8. References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correction coding and decoding: Turbo codes", *Proc. 1993 Inter. Conf. Commun.*, May 1993, pp. 1064-1070.

[2] E. Yeo, B. Nikolic, and V. Anantharam, "Iterative Decoder Architectures", *IEEE Commun. Magazine*, August 2003, pp. 132-140.

[3] Olaf J. Joeressen, Martin Vaupel, and Heinrich Meyr, "High-Speed VLSI Architectures for Soft-Output Viterbi Decoding", *Proc. IEEE Int. Conf. Application Specific Array Processors*, August 1992, pp.373-384.

[4] Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal decoding algorithm", *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 1009-1013.

[5] H. Dawid, and H. Meyer, "Real-time algorithms and VLSI architectures for soft output MAP convolutional decoding", *IEEE Int. Sym. PIMRC, Vol. 1*, Sept. 1995, pp. 193-197.

[6] Z. Wang, H. Suzuki, and K. K. Pahari, "VLSI implementation issues of turbo decoder design for wireless applications", *Proc. IEEE Int. Workshop Signal Processing Syst.*, 1999, pp. 503-512.

[7] Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architecture Strategies for Low-Power VLSI Turbo Decoders", *IEEE Tran. on VLSI Sys.* Vol. 10, No. 3, June 2002, pp. 279-285.

[8] Bickerstaff, M.A.; Garrett, D.; Prokop, T.; Thomas, C.; Widdup, B.; Gongyu Zhou; Davis, L.M.; Woodward, G.; Nicol, C.; Ran-Hong Yan, "A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless inn 0.18nm CMOS", *IEEE J. Solid State Circuits*, Vol. 37, No. 11, Nov. 2002, pp. 1555-1564.

[9] S. J. Lee, N. R. Shanbhag, and A. C. Singer, "A Low-Power VLSI Architecture for Turbo Decoding", *ISLPED03*, August 2003, pp.366-371.

[10] C. Thomas, M. A. Bickerstaff, and L. M. Davis, T. Prokop, Be. Widdup, G. Zhou, D. Barret, and C. Nicol, "Integrated Circuits for Channel Coding in 3G Cellular Mobile Wireless Systems", *IEEE Commun. Magazine*, August 2003, pp.150-159.

[11] G. Montorsi, and S. Benedetto, "Design of Fixed-Point Iterative Decoders for Concatenated Codes with Interleavers", *IEEE J. on Selected Areas in Commun.* Vol. 19, No. 5, May 2001, pp.871-882.

[12] 3GPP TS 25.212, "Multiplexing and Channel Coding (FDD)", v. 4.6.0, Sept. 2002.

[13] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes", *IEEE Tran. Info. Theory*, Vol. 42, March 1996, pp. 429-445