

A Power Efficient Reconfigurable Max-Log-MAP Turbo Decoder for Wireless Communication Systems

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

¹University of Edinburgh, School of Engineering and Electronics
Edinburgh, EH9 3JL, Scotland, United Kingdom

²Institute of System Level Integration, The ALBA campus
Livingston, EH54 7EG, Scotland, United Kingdom

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

Abstract

The authors present a reconfigurable soft-input soft-output (SISO) turbo decoder based on Max-Log maximum a posteriori (ML-MAP) algorithm implemented with a sliding window (SW) method. The turbo decoder is designed to support constraint lengths from 3 to 5 and synthesized to a 0.18 μ m standard CMOS cell library. Power and area overheads for the reconfiguration are analyzed and compared with non-reconfigurable ASIC based turbo decoder for each constraint length. Our simulation results demonstrate that the reconfigurable architecture can be applied flexibly to various wireless communication systems without large power and area overheads.

I. Introduction

Demand for turbo codes in wireless communication systems has been increasing since their appearance in the early 1990s, due to their outstanding performance in terms of bit error rate (BER) [1]. For this reason, they have been adopted by various wireless systems such as DVB-RCS, 3GPP UMTS, IEEE 802.16, and CCSDS [2]-[3]. Various turbo decoders have been developed to improve their performance at algorithm and architecture levels. A dual mode decoder for convolutional and turbo codes has also been introduced for multi-standard wireless communication systems [4]. In order to correspond to different standards of wireless communication systems, a reconfigurable architecture is more suitable than an ASIC for better flexibility and more cost saving than a digital signal processor (DSP). There are many works in the literature researching reconfigurable Viterbi decoders for convolutional codes, e.g. [5]-[6]. Whereas, there is not much work targeting reconfigurable turbo decoders.

In this paper, we have implemented a reconfigurable turbo decoder based on ML-MAP with sliding window (SW) method. The application of turbo codes for constraint lengths larger than 5 is not reported in the literature to date. Therefore, the reconfigurable turbo decoder has been designed to support constraint lengths (K) from 3 to 5. For the

reconfigurability, a mapping method for the forward and backward state metric units (SMU) is presented to reallocate the generated state metrics for different constraint lengths. The SMU consists of a branch metric unit (BMU), a branch metric normalization unit (BMNU), add-compare-select (ACS) units, and mapping units (MAU). The log likelihood ratio (LLR) is computed by the LLR computation unit (LCU), which is efficiently implemented with a parallel 'compare' structure. Section II provides a brief introduction to turbo decoding. The reconfigurable architecture and its components are described in section III. We demonstrate the comparisons of power consumption and area usage between the reconfigurable and ASIC turbo decoders in section IV and conclude with section V.

II. TURBO DECODING ALGORITHM

A. ML-MAP Algorithm

The decoding process using the ML-MAP algorithm is performed by the forward and backward processes in order to compute all state metrics [7]. These values are then used to compute the LLR values as shown by the following equation:

$$L_{lr} = L_1 - L_0 \quad (1)$$

$$L_1 = \max_{s', s, u_k = +1} [a_{k-1}(s') + c_k(s', s) + b_k(s)]$$

$$L_0 = \max_{s', s, u_k = -1} [a_{k-1}(s') + c_k(s', s) + b_k(s)]$$

where $a_{k-1}(s')$, $c_k(s', s)$, and $b_k(s)$ represent branch, forward, and backward state metrics, respectively. The subscript k , s' and s denote time and trellis states. In equation (1), each of the metrics is represented as shown below :

$$c_k(s', s) = 1/2(L_e u_k^s + L_c x u_k^s + L_c y_l u_k^p) \quad (2)$$

$$a_k(s) = \max_j [a_{k-1}(s') + c_k(s', s)] \quad (3)$$

$$b_k(s') = \max_j [b_{k+1}(s) + c_k(s', s)] \quad (4)$$

where $c_k(s', s)$ is calculated by the a priori information (L_e), the channel reliability value (L_c), input data (x and y_l), the systematic bit (u_k^s), and the parity bit (u_k^p). The L_e is obtained from the LLR value computed in previous decoding process after

subtracting the input data of the systematic bit and a priori value from the LLR value. The ML-MAP algorithm has a more simplified form than the original MAP algorithm. The simplifying is enabled by a well know approximation, called Jacobi logarithm, using the following equation :

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

The 2nd term of the right hand side in equation (5) is the correction term, which can be implemented through a simple look-up table [8]. However, in this paper, we have implemented ML-MAP decoder in which the correction term is ignored.

B. Sliding Window Method

The sliding window (SW) method [9] can reduce the latency and memory size for state metrics in the turbo decoding process by dividing an input block into several sub-blocks. Fig. 1 shows a graph of data flow for the decoding process in time and block axis. The solid line without notation indicates buffering of input data and the dashed line with b represents the backward process and the number identifies the sub-block. Actually, the backward process of dashed line is not for computing the LLR value, but to provide the state metric to the other backward process. The backward state metrics for the LLR values computation are generated 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 a , of a sub-block is completed before the backward and LLR computation. During the forward process, the state metrics are stored into a LIFO memory block. In this paper, this SW method has been implemented with a latency of 4 times of the sub-block length.

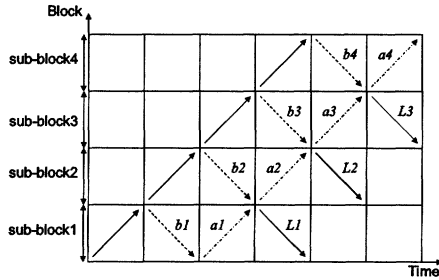


Figure 1. A graph of data flow with SW method.

III. IMPLEMENTATION

A. ML-MAP SISO decoder architecture

The reconfigurable ML-MAP turbo decoder architecture implemented in this paper is illustrated in Fig. 2. The architecture incorporates LIFOs and FIFOs for buffering of input and output data, and storing of state metrics. The SMU, LCU, and LIFO2 blocks are designed to be reconfigured by K and to

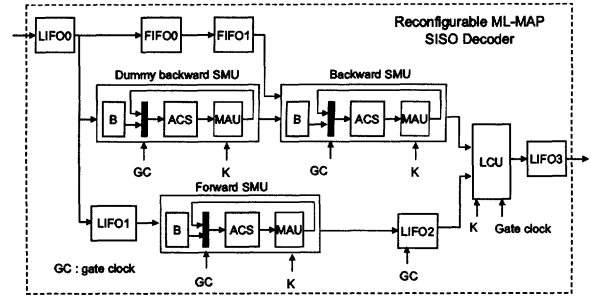


Figure 2. A block diagram of the ML-MAP SISO decoder.

save power consumption by employing clock gating. There are three SMUs in this architecture; one for forward and two for backward state metric calculations. One of the backward SMUs (called dummy backward SMU) produces the state metrics to the other backward SMU that generates the state metrics needed to compute the LLR values. The calculated forward state metric values are stored into the LIFO2, which employs clock gating. While 16 forward state metrics are input to the LIFO2 for $K=5$, 4 and 8 forward state metrics are input to the LIFO2 for $K=3$ and 4. The detailed reconfigurable SMU and LCU structures are described in the next sub sections.

B. State Metric Unit Structure and Mapping

B.1 State metric unit structure

Figure 3 shows the SMU structure used to compute the branch, forward and backward state metrics. The branch metric calculation is performed in two stages using BMU and BMNU with pipeline registers. The ACSs are recursively processed to compute the state metrics through the mapping unit (MAU) that reallocates the state metrics for the next ACS based on the current K value. In Fig. 4 (a), the BMU computes the branch metrics and the BMNU normalizes the branch metrics. The normalization process is performed by substituting the maximum (or minimum) branch metric from each of the

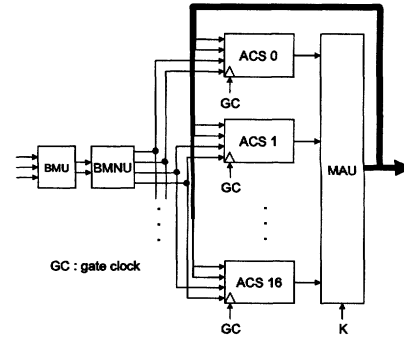


Figure 3. A structure of the SMU for the backward and forward processes.

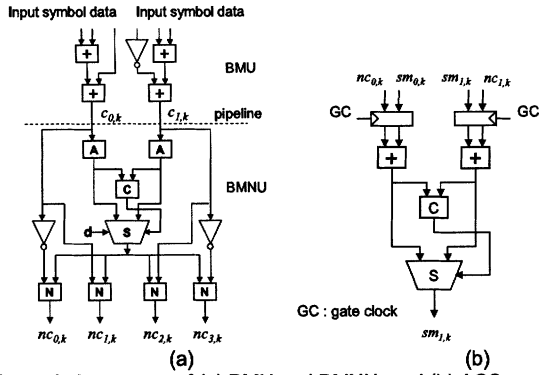


Figure 4. A structure of (a) BMU and BMNU, and (b) ACS.

branch metrics (c_{0k} , c_{1k}). The maximum (or minimum) branch metric is obtained by the compare (C) and select (S) after converting the branch metrics into absolute values (A). In the process, the decision (d) that is input to the S is determined by the state metrics. The decision rule is that if all state metrics are larger (or less) than zero, the maximum (or minimum) branch metric is used for the normalization. If the rule is not satisfied, the branch metrics are provided to ACSs without the normalization process. This normalization can lead to a simplified ACS as shown in Figure 4 (b) and can prevent overflow of the state metric without the normalization process for the state metric [10].

B.2 Mapping of state metric

In the reconfigurable structure, the state metrics must be mapped by different K before they are input to the ACS in the next clock cycle. The mappings for the forward and backward processes for K=3 and K=4 are shown in Fig. 5. As can be seen, the state metrics of the forward and backward processes are reordered by K after the mapping. For constraint length K=3, for example, while the state metrics (a_0 , a_1 , a_8 , and a_9) generated by the ACS 0, 1, 8 and 9 are input to ACS 0, 1, 2 and 3 for the forward process, the state metrics (b_0 , b_1 , b_2 , and b_3) generated by the ACS 0, 1, 2 and 3

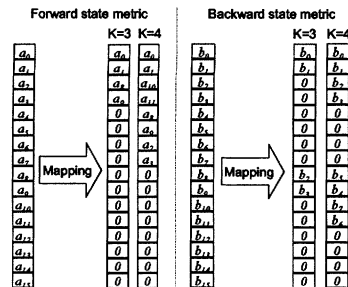


Figure 5. A mapping method of the state metric values.

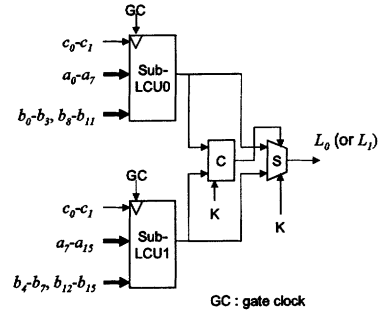


Figure 6. A structure of the reconfigurable LCU.

are input to ACS 0, 1, 8 and 9 the backward process. The rest of the ACS inputs are set to zero. A similar mapping is employed for K=4. For K=5, the ACS process is performed without the mapping.

C. LLR Computation Unit Structure

Figure 6 shows the structure of the LCU consisting of two sub-LCUs, one compare (C), and one select (S) units. Each sub-LCU can be reconfigured by K and activated or de-activated by a gated clock. Therefore, when the LCU is configured for K=4, the clock input of the sub-LCU1 is blocked by the clock gating method. Inputs of the sub-LCUs are the mapped forward (a_0 - a_{15}) and backward (b_0 - b_{15}) state values, which are obtained as shown in Figure 5, and the branch metrics (c_0 - c_1). The output of the LCU is determined by compare (C) and select (S) units with K and associated with L_0 or L_1 in equation (1). A conventional LCU is implemented by a tree structure of compare and select units. However, this results in a long critical path. Therefore, our sub-LCU structures have been implemented with a parallel compare (C) structures as shown in Fig. 7. The sub-LCU architecture is based on two pipeline stages. The output of the sub-LCU is determined by the constraint length K, using six compares (C) and a select (S) after computing the input metric values at the first pipeline stage.

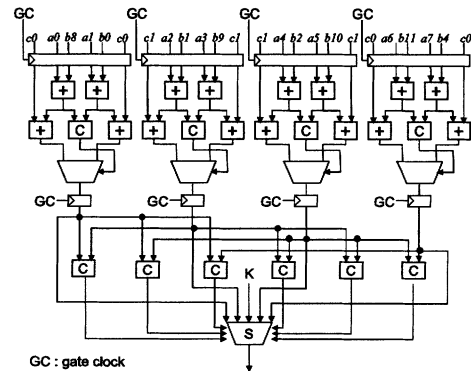


Figure 7. A structure of the sub-LCU.

IV. SIMULATION RESULTS

The reconfigurable ML-MAP turbo decoder has been implemented using Verilog HDL at RTL level and synthesized to a 0.18um standard CMOS cell library with Synopsys DesignCompiler™. RTL and gate level simulation has been performed using Cadence Verilog-XL™. Synopsys DesignPower™ was used to evaluate power consumption. Using the same procedure, we have implemented non-reconfigurable ML-MAP turbo decoders for fixed constraint lengths of K=3, 4, and 5.

Figure 8 illustrates the power contribution of the main blocks for the reconfigurable and the ASIC turbo decoders at a clock speed of 50MHz. The power results are obtained for reconfigurable (R) and ASIC (A) implementations for each K. As can be seen, the reconfigurable architecture has been implemented without a significant increase of power consumption when compared to the ASIC implementations. Table 1 summarizes the power and area comparisons for ASIC and reconfigurable turbo decoders. The area overhead for the reconfigurable architecture are 121%, 57.6%, and 0.5% when compared with ASIC of K=3, 4, and 5, respectively. Clearly, the area overhead for reconfiguration is very large for K=3 and 4. Whereas, the power overhead for the reconfigurable architecture is only 12.7%, 2.3%, and 0.3% for K=3, 4, and 5, respectively.

Table 1. Comparison of power and area results of the reconfigurable and ASIC turbo decoder.

	Power (mW)	Overhead	Area(mm ²)	Overhead
K3-A	22.8	-	0.634	-
K3-R	25.7	12.7%	1.405	121%
K4-A	33.6	-	0.891	-
K4-R	34.4	2.3%	1.405	57.6%
K5-A	56.3	-	1.398	-
K5-R	56.5	0.3%	1.405	0.5%

V. Summary and Conclusions

The authors have presented a reconfigurable ML-MAP turbo decoder architecture with the sliding window method. After comparison with the non-reconfigurable, fixed constraint length ASIC implementations, the power overhead for the reconfiguration was found to be less than 13%, although the area overhead was up to 121%. Therefore, it can be concluded that our flexible turbo decoder can be used for multi-standard wireless communication systems without significant power increase.

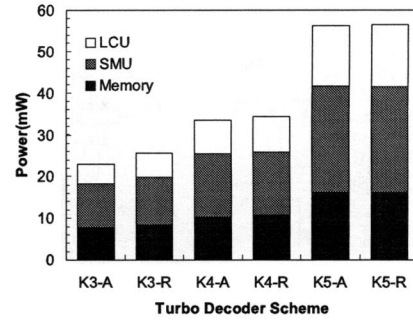


Figure 8. Comparison of power simulation results.

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. Mag.*, Aug. 2003, pp. 132-140.
- (3) C. Berrou, "The Ten-Year-Old Turbo Codes are Entering into Service," *IEEE Commun. Mag.* vol. 41, no. 8, Aug. 2003, pp.110-116.
- (4) M. Bickerstaff, et al, "A unified turbo/viterbi channel decoder for 3GPP mobile wireless in 0.18/spl mu/m CMOS," *IEEE J. Solid State Circuits*, vol. 2, Feb. 2002 pp.90 – 414.
- (5) Chadha, K., and Cavallaro, J. R., "A reconfigurable Viterbi decoder architecture," *Asilmolar Conf. Signals, Systems, and Computers*, vol. 1, Nov. 2001, pp.66 – 71.
- (6) Kelly, P. H., and Chau, P. M, "A flexible constraint length, foldable Viterbi decoder," *IEEE Global Telecom. Conf.*, Nov. 1993, pp. 631 – 635.
- (7) Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal decoding algorithm," *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 1009-1013.
- (8) 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.
- (9) H. Dawid, and H. Meyer, "Real-time algorithms and VLSI architectures for soft output MAP convolutional decoding," *IEEE Int. Sym. PIMRC*, vol. 1, Sep. 1995, pp. 193-197.
- (10) 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.