Non-recursive max* operator with reduced implementation complexity for turbo decoding

Novel Non-Recursive max* Operator with Reduced Implementation Complexity for Turbo Decoding

Stylianos Papaharalabos*, P. Takis Mathiopoulos*, Guido Masera**, and Maurizio Martina**

December 2, 2011

(*) Institute for Space Applications and Remote Sensing (ISARS), National Observatory of Athens, Metaxa and Vas. Pavlou Str., Palaia Penteli, GR-15236, Athens, Greece.

(**) VLSI Lab, Dipartimento di Elettronica, Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino, Italy.

E-mail: spapaha@space.noa.gr

This work was supported by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (contract no. 216715).

Abstract

In this paper, we deal with the problem of how to effectively approximate the max* operator when having $n > 2$ input values, with the aim of reducing implementation complexity of conventional Log-MAP turbo decoders. We show that, contrary to previous approaches, it is not necessary to apply the max* operator recursively over pairs of values. Instead, a simple, yet effective, solution for the max* operator is revealed having the advantage of being in non-recursive form and thus, requiring less computational effort. Hardware synthesis results for practical turbo decoders have shown implementation savings for the proposed method against the most recent published efficient turbo decoding algorithms by providing near optimal bit error rate (BER) performance.

1 Introduction

In the past, several algorithmic approaches have been proposed aiming to simplify the well-known max* operator [1] for decoding turbo codes [2], including Improved Max-Log-MAP, Constant Log-MAP [3], Linear Log-MAP, Average Log-MAP, *etc.* A detailed comparative study among these techniques was published in [4], in which the max* operator was approximated using the max operator and a small number of piecewise linear (PWL) terms, denoted as r. As shown in [4], the implementation of the $r = 3$ approximation is very simple and has similar complexity with the Constant Log-MAP algorithm. Additionally, the $r = 4$ approximation requires higher complexity and it is comparable, in terms of complexity, with the most recent known algorithms, such as the Linear Log-MAP, the Average Log-MAP, *etc.* [4]. The penalty paid for all these well-known reduced complexity implementation techniques is a small bit error rate (BER) performance degradation as compared to the BER performance achieved by the optimal Log-MAP algorithm [5]. Currently, the best trade-off between turbo code performance and complexity is achieved by the Constant Log-MAP algorithm.

Conventionally, for Log-MAP turbo decoding the max* operator is defined only for $n = 2$ input values [1] and for $n > 2$ it is applied $n - 1$ times in a recursive form [5]. In

this paper, we show how the max* operator with $n > 2$ input values can be computed effectively in a non-recursive form with the aim of reducing implementation complexity of the Log-MAP turbo decoder. Performance evaluation results are depicted for both binary and duo-binary turbo codes showing the near-optimal and essentially optimal BER performance of the proposed method, respectively. Furthermore, hardware synthesis results indicate implementation advantages against the most recent published efficient turbo decoding algorithms, such as the $r = 3$ and $r = 4$ approximations [4], which makes our proposed method quite appealing in practical communication systems.

2 Novel Non-Recursive $\max^*$ Operator

The max* operation, i.e. Jacobian logarithm, used in Log-MAP turbo decoding, is defined as [1]

$$\begin{aligned} \max^*(x_1, x_2) \stackrel{\triangle}{=} \log\{\exp(x_1) + \exp(x_2)\} &= \max(x_1, x_2) \\ + \log\{1 + \exp(-|x_1 - x_2|)\} &= \max(x_1, x_2) + f_c(|x_1 - x_2|) \end{aligned} \tag{1}$$

where $f_c(|x_1 - x_2|)$ is a non-linear function referred to as 'correction term' [5] and $|.|$ denotes absolute value. Typically, for more than two input values, the Jacobian logarithm is applied recursively [5]. For example, considering three values, it yields

$$\max^*(x_1, x_2, x_3) = \max^*\{\max^*(x_1, x_2), x_3\} \tag{2}$$

For the Log-MAP algorithm, a look-up table (LUT) substitutes $f_c(|x_1 - x_2|)$, which is usually implemented with eight values [5]. If the LUT is omitted, then the Log-MAP simplifies to the Max-Log-MAP algorithm.

From (2) it is evident that for n input values the max* operator is applied recursively $n - 1$ times. Let us consider the Chebyshev inequality [6, p. 186, Eq. 36.11]

$$\left\{\sum_{i=1}^{n} a_i\right\}\left\{\sum_{i=1}^{n} b_i\right\} \leq n \sum_{i=1}^{n} a_i b_i \tag{3}$$

where $a_1 \geq a_2 \geq \cdots \geq a_n$ and $b_1 \geq b_2 \geq \cdots \geq b_n$. For $a_1 = \exp(x_1)$, $a_2 = \exp(x_2)$, ..., $a_n = \exp(x_n)$ and $b_1 = n$, $b_2 = n-1$, ..., $b_n = 1$ the Chebyshev inequality yields

$$\left\{\sum_{i=1}^{n} \exp(x_i)\right\} \{n + (n-1) + \ldots + 1\} \leq n \{\exp(x_1)n + \exp(x_2)(n-1) + \ldots + \exp(x_n)\} \quad (4)$$

Taking the logarithm on both sides of the above inequality and considering only two terms in the right-hand side (RHS), the following approximation is obtained

$$\log\left\{\sum_{i=1}^{n} \exp(x_i)\right\} + \log\{n(n+1)/2\} \approx \log(n) + x_1 + \log\{n + \exp(x_2 - x_1)(n-1)\} \quad (5)$$

or equivalently

$$\max_{i=1:n}{}^{*}(x_i) \approx \log\{2n/(n+1)\} + \max_{i=1:n}(x_i) + \log\{[1 + \exp(-\delta)(n-1)/n]\} \quad (6)$$

where $x_1 = \max_{i=1:n}(x_i)$, $\delta = x_1 - x_2$, and x_2 is the second maximum value among n values.

Since the first term in the RHS of (6) is positive constant it can be ignored, and because for large values of n, $(n-1)/n \approx 1$, (6) is further simplified to

$$\max_{i=1:n}{}^{*}(x_i) \approx \max_{i=1:n}(x_i) + \log\{[1 + \exp(-\delta)]\} \quad (7)$$

denoted as Log-MAP Delta. From (7) it is evident that the max* operator having n input values can be computed non-recursively since it requires only knowledge of the maximum among n values and an additive correction term depending on the second maximum value among n values.

3 Turbo Code Performance Evaluation Results

Performance evaluation results have been obtained by means of computer based simulations for both binary and duo-binary turbo codes, in terms of BER against bit energy E_b in an

additive white Gaussian noise (AWGN) channel having one-sided power spectral density N_o. The binary turbo code used in simulations is a 16-state code with overal coding rate $R = 1/2$ and generator polynomials $(1, 33/23)_o$ in octal form representing the feed-forward and backward polynomials, respectively. An information sequence length of $N = 10^3$ bits is assumed together with a pseudo-random turbo interleaver. The modulation type is binary phase shift keying (BPSK). Additionally, the duo-binary turbo code used in simulations is an 8-state code proposed in the Digital Video Broadcasting by Satellite-Return Channel (DVB-RCS) [7] and Worldwide Interoperability for Microwave Access (Wi-MAX) [8] standards with overall coding rates $R = 1/3$, $2/3$, and $4/5$, and information sequence length of $N = 752$ bit couples (MPEG packets). The modulation type in this case is quadrature phase shift keying (QPSK). At the receiver, a maximum of 10 decoding iterations are performed for both turbo codes. In our of computer based simulations, BPSK/QPSK modulation is implemented with antipodal baseband signaling representation, i.e. no carrier frequency is assumed.

The performance of the newly proposed algorithm given in (7) is compared with the least complex Max-Log-MAP algorithm and also with the best performing Log-MAP algorithm. As shown in Figs. 1 and 2, Log-MAP Delta achieves near-optimal BER performance, which is far better than the equivalent performance achieved by Max-Log-MAP algorithm. For instance, in Fig. 1 and the 16-state turbo code, the BER performance degradation of Log-MAP Delta against Log-MAP algorithm is only 0.05 dB at BER of 10^{-5}. Similarly, in Fig. 2 and the 8-state duo-binary turbo code, Log-MAP Delta has negligible performance degradation against Log-MAP algorithm at BER of 10^{-6}.

Furthermore, performance evaluation results have been obtained assuming the two efficient decoding algorithms proposed recently in [4], i.e. $r = 3$ and $r = 4$ approximations, and also the Constant Log-MAP algorithm, which currently provides the best trade-off between BER performance and complexity. Following [9], to further improve the BER performance scaling was applied in the extrinsic information and the best performing values were found by trial and error for all investigated algorithms. The corresponding performance evaluation results for both binary and duo-binary turbo codes are summarized in Table 1, without

scaling, and Table 2, using scaling, respectively. From Table 1, it is noticed that Log-MAP Delta performs similarly with $r = 3$ approximation, whereas both $r = 4$ approximation and Constant Log-MAP algorithm achieve near optimal BER performance. Furthermore, from Table 2, it is noticed that scaling improves performance and Log-MAP Delta performs similarly with $r = 4$ approximation and Constant Log-MAP algorithm. Hence, Log-MAP Delta achieves esentially optimal BER performance.

4 Hardware Architecture Description

Several architectures have been proposed for the implementation of (1) and (2) based on the two-input max* structure [4, 10]. In principle, two main implementation approaches are feasible for $n > 2$, that is: serial and parallel architecture. On the one hand, the serial architecture employs only one two-input max* structure and a bank of registers but it requires $n - 1$ clock cycles to complete the computation. On the other hand, to reduce the latency of a generic n-input max* structure, parallel architectures employing $n-1$ two-input max* structures and operating concurrently in a tree-based architecture are usually preferred. Notice that experimental results comparing serial and parallel architectures will be discussed in Section 5. Furthermore, the computation of (7) requires knowledge of the first two maximum values among n elements, therefore another solution is necessary. From the implementation point of view, (7) can be split into two sub-problems: i) Finding the maximum x_1 and x_2; and ii) Computing the correction term $\log\{[1 + \exp(-\delta)]\}$. These two sub-problems are presented next.

4.1 Maximum Finding

The most straightforward approach in finding x_1 and x_2 among x_i with $i = 1, 2, ..., n$ is to sort x_i. As suggested in [11, Chapter 28.5, pp. 1-2], to obtain a parallel sorter a merge sort architecture can be deployed. However, this approach will result in an increased area overhead, due to the large number of comparators required for finding x_1 and x_2 being equal to $n/4 \cdot \{[\log_2(n) + 1] \cdot \log_2(n)\}$. An alternative solution is obtained by adapting the

first-two-minimum-finder architecture (denoted as M2), which was proposed in [12] for low-density parity-check (LDPC) decoding. This architecture is based on a tree structure using Maximum-Value Generators (MVG). The structure for n inputs is derived recursively from two MVG architectures for $n/2$ values and a connection unit, based on Maximum-Value Units (MVU) as shown in Figs. 3 (a) and (b) where

$$s = \begin{cases} 0 & \text{if} \quad A \geq B \\ 1 & \text{if} \quad A < B \end{cases} \tag{8}$$

With the M2 architecture the number of comparators required for finding x_1 and x_2 is $2n - 3$.

4.2 Correction Term Computation

A simple implementation of the correction term in (7) is obtained by exploiting the procedure adopted in [10] for the two-input max* approximation. Namely, the correction term is stored into a LUT accessed by δ. The size of the LUT, denoted as m, is the minimum positive integer value that satisfies

$$\log\{[1 + \exp(-m/2^p)]\} \leq 2^{-(p+1)} \tag{9}$$

and p is the number of fractional bits to represent δ as a fixed point value. Then, the LUT content is obtained by computing $\log\{[1 + \exp(-i/2^p)]\}$, $\forall i \in [0, m-1]$ and converting the result as a fixed point value on p fractional bits.

5 Hardware Synthesis Results

Post synthesis results obtained by implementing the max* operator and its approximations on 90 nm standard cell technology are shown in Table 3. The synthesis was performed with Synopsys Design Compiler [13] for a target clock frequency of 200 MHz representing the data on $n_b = 8$, 12, 16 bits. The design space been explored includes the parameters

n = 4, 8, 16 and p = 1, 2, 3 being the number of inputs for the max* operation and the fractional bits, respectively. Furthermore, the n input max* operator is approximated as: (i) Simple recursive application of the 2-input max operation (denoted as MX and corresponding to Max-Log-MAP algorithm); (ii) Recursive application of the Jacobian logarithm with correction function implemented by means of a LUT [5, 10] (denoted as JL and corresponding to Log-MAP algorithm); (iii) Adoption of M2 architecture for the maximun finding, according to (7), which corresponds to Log-MAP Delta; (iv) Constant Log-MAP architecture as proposed in [3]; and (v) $r = 3$, $r = 4$ architectures as proposed in [4]. As it can be observed, the proposed solution with M2 architecture leads to significantly lower complexity than JL. In particular, the area of M2 is from 50% to 70% of the area required to implement JL. Furthermore, the proposed M2 solution is simpler than the approximations recently proposed in [4] for both $r = 3$ and $r = 4$, whereas it has nearly the same complexity with Constant Log-MAP.

Since, as it can be seen from Table 3 the proposed M2 and the Constant Log-MAP architectures have comparable complexity, it is interesting to further investigate these two solutions. On the one hand, the proposed Log-MAP Delta approximation is intrinsically parallel for an n-input max* operator. On the other hand, the Constant Log-MAP approximation can be employed to obtain either a serial or a parallel implementation. Serial implementation of the Constant Log-MAP approximation for an n-input max* operator has been carried out for the cases shown in Table 3, namely $n = 4, 8, 16$ and $n_b = 8, 12, 16$. The minimum delay of this serial architecture ranges from 0.8 ns to 0.9 ns and the area from 1246 μm^2 to 5597 μm^2. Unfortunately, the low delay of such serial architecture can not be fully exploited in the design of a complete MAP decoder mainly, due to the limited maximum clock frequency achieved by memories. For this type of technology, such maximum clock frequency is about 500 MHz. As a consequence, the low area figures offered by serial architectures come at the expense of a dramatic throughput reduction as compared with parallel implementations. For instance, if we consider the Universal Mobile Telecommunication System-Long Term Evolution (UMTS-LTE) turbo code and a serial implementation, each half iteration in the decoding process requires a number of clock cycles,

which is about twenty times the number of clock cycles required by a parallel implementation. Therefore, in the following we will concentrate on area synthesis results achieved with parallel implementation. To better compare the proposed parallel architecture and the Constant Log-MAP based one, we have investigated the minimum delay achieved by both of them. In Table 4 both area (denoted as A) and minimum delay (denoted as D) are shown for proposed parallel architecture when $p = 1$ and the Constant Log-MAP, respectively. As it can be observed, in several cases the proposed architecture achieves lower delay and complexity as compared to the Constant Log-MAP for maximum achievable clock frequency been equal to $1/D$.

In order to evaluate the impact of the proposed solution on the complexity of a complete Soft-In-Soft-Out (SISO) module [10] both a UMTS-LTE [14] and a Consultative Committee for Space Data Systems (CCSDS) [15] SISO module were implemented, in which binary turbo codes are deployed. The UMTS/LTE SISO module requires two 8-input max* operators to compute the a posteriori information (APO) [16]. Similarly, the CCSDS SISO module requires two 16-input max* operators to compute the APO [17]. In both cases, the n-input max* operators were implemented as M2, MX, JL, Constant Log-MAP, and $r = 3$, $r = 4$ architectures. Moreover, the intrinsic and extrinsic information were represented with six and eight bits, respectively with $p = 3$, whereas state metrics were represented with ten bits.

Post synthesis results for the UMTS-LTE/CCSDS turbo codes, as shown in Table 5, depict that the area required to compute APO with the proposed M2 architecture is about 74% of the area occupied by JL-based solution. If we consider the area occupied by the logic of a whole SISO module, then the proposed M2 architecture features an area saving that ranges from about 12% to 15% with respect to a JL-based SISO. We have similarly investigated the DVB-RCS/Wi-MAX duo-binary turbo code and the post synthesis results are shown in Table 6. In this case, M2 architecture offers 21% area savings with respect to JL-based SISO. Furthermore, the area required to compute APO/SISO modules with the proposed M2 architecture is less than that required by both the $r = 3$ and $r = 4$ approximations [4]. Lastly, Constant Log-MAP requires the smallest area to compute

APO/SISO modules. It is thus, the most efficient algorithm, in terms of computational complexity.

6 Conclusion

It has been shown how the max* operator with n input values can be approximated effectively without recursive computation, in order to reduce implementation complexity of practical Log-MAP turbo decoders. For the case of a 16-state binary turbo code, 0.05 dB of performance degradation was observed at BER of 10^{-5} but with 15% complexity savings. In another case, for an 8-state duo-binary turbo code neglibible performance degradation was observed at BER of 10^{-6}, while maintaining 21% complexity savings. If scaling is additionally used, then negligible performance degradation is observed against Log-MAP algorithm for both binary and duo-binary turbo codes. In terms of complexity comparison with other state-of-the-art reduced complexity algorithms, the proposed solution is simpler than the approximations recently published in [4] for both $r = 3$ and $r = 4$, and it is slightly more complex than Constant Log-MAP algorithm.

References

[1] Viterbi, A. J.: 'An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes', *IEEE J. Sel. Areas Commun.*, 1998, **16**, (2), pp. 260–264.

[2] Berrou, C., Glavieux A., and Thitimajhima, P.: 'Near Shannon limit error correcting coding and decoding: Turbo codes', Proc. IEEE Int. Conf. Commun. (ICC), Geneva, Switzerland, 1993, pp. 1064–1070.

[3] Gross, W. J., and Gulak, P. G.: 'Simplified MAP algorithm suitable for implementation of turbo decoders', *IEE Electron. Lett.*, 1998, **34**, (16), pp. 1577-1578.

[4] Papaharalabos, S., Mathiopoulos, P. T., Masera, G., and Martina, M.: 'On optimal and near-optimal turbo decoding using generalized max* operator', *IEEE Commun. Lett.*, 2009, **13**, (7), pp. 522–524.

[5] Robertson, P., Villebrun, E., and Hoeher, P.: 'A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain', Proc. IEEE Int. Conf. Commun. (ICC), Seattle, USA, 1995, pp. 1009–1013.

[6] Spiegel, M. R.: 'Mathematical handbook of formulas and tables', *McGraw-Hill*, 1968.

[7] 'Digital Video Broadcasting (DVB): Interaction channel for satellite distribution systems', ETSI EN 301 790, v 1.3.1, Mar. 2003.

[8] 'Air interface for fixed and mobile broadband wireless access sytems: Physical and medium access control layers for combined fixed and mobile operation in licensed bands', IEEE P802.16e-2005 Amendment 2, Feb. 2006.

[9] Vogt, J., and Finger, A.: 'Improving the Max-Log-MAP turbo decoder', *IEE Electron. Lett.*, 2000, **36**, (23), pp. 1937-1939.

[10] Montorsi, G., and Benedetto, S.: 'Design of fixed-point iterative decoders for concatenated codes with interleavers', *IEEE J. Sel. Areas Commun.*, 2001, **19**, (5), pp. 871-882.

[11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.: 'Introduction to algorithms', *The MIT Press*, 2nd ed., 2001.

[12] Wey, C. L., Shieh M. D., and Lin, S. Y.: 'Algorithms of finding the first two minimum values and their hardware implementation', *IEEE Trans. Circuits Syst. I*, 2008, **55**, (11), pp. 3430-3437.

[13] http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCUltra/Pages/default.aspx, accessed 2011.

[14] '3GPP TS 36.212 v8.0.0: Multiplexing and channel coding', 2007-2009.

[15] 'Consulative Committee for Space Data Systems (CCSDS): Telemetry channel coding', ser. Blue Book, May 1999.

[16] Martina, M., Nicola, M., and Masera, G.: 'A flexible UMTS-WiMax turbo decoder architecture', *IEEE Trans. Circuits Syst. II*, 2008, **55**, 4, pp. 369-373.

[17] Miyauchi, T., Yamamoto, K., Yokokawa, T., Kan, M., Mizutani, Y., and Hattori, M. M.: 'High-performance programmable SISO decoder VLSI implementation for decoding turbo codes', *Proc. IEEE Global Telecommunications Conference*, San Antonio, USA, 2001, pp. 305-309.

Table 1: Required E_b/N_o (in dB) at BER $= 10^{-5}$ for binary turbo code and at BER $= 10^{-6}$ for duo-binary turbo code.

Decoding Algorithm	Binary Turbo Code $R = 1/2$	Duo-binary Turbo Code $R = 1/3$	Duo-binary Turbo Code $R = 2/3$	Duo-binary Turbo Code $R = 4/5$
Max-Log-MAP	2.1	1.45	2.65	3.85
$r = 3$ Approx. [4]	1.75	1.35	2.55	3.75
$r = 4$ Approx. [4]	1.7	1.35	2.55	3.75
Log-MAP Delta	1.75	1.35	2.55	3.75
Constant Log-MAP	1.7	1.35	2.55	3.75
Log-MAP	1.7	1.3	2.55	3.75

Table 2: As in Table 1 but with the extrinsic information scaled by a factor of sc.

Decoding Algorithm	Binary Turbo Code $R = 1/2$	Duo-binary Turbo Code $R = 1/3$	Duo-binary Turbo Code $R = 2/3$	Duo-binary Turbo Code $R = 4/5$
Max-Log-MAP	1.7 ($sc = 0.65$)	1.3 ($sc = 0.75$)	2.55 ($sc = 0.75$)	3.75 ($sc = 0.75$)
$r = 3$ Approx. [4]	1.65 ($sc = 0.75$)	1.25 ($sc = 0.85$)	2.45 ($sc = 0.85$)	3.65 ($sc = 0.85$)
$r = 4$ Approx. [4]	1.6 ($sc = 0.75$)	1.25 ($sc = 0.9$)	2.45 ($sc = 0.9$)	3.65 ($sc = 0.9$)
Log-MAP Delta	1.6 ($sc = 0.75$)	1.25 ($sc = 0.85$)	2.45 ($sc = 0.85$)	3.65 ($sc = 0.85$)
Constant Log-MAP	1.6 ($sc = 0.85$)	1.25 ($sc = 0.9$)	2.45 ($sc = 0.9$)	3.65 ($sc = 0.9$)
Log-MAP	1.6 ($sc = 0.9$)	1.25 ($sc = 0.9$)	2.45 ($sc = 0.9$)	3.65 ($sc = 0.9$)

Table 3: Post synthesis area results [μm^2] obtained by implementing the max* operator used in Log-MAP and its approximations for a target clock frequency of 200 MHz.

n	n_b	p	Max-Log-MAP (MX) μm^2	Log-MAP Delta (M2) μm^2	Log-MAP (JL) μm^2	$r = 3$ [4] μm^2	$r = 4$ [4] μm^2	Constant Log-MAP μm^2
		1		859.42	1302.54			
	8	2	347.86	891.88	1379.45	996.84	1353.68	834.72
		3		918.69	1508.57			
		1		1396.38	2088.58			
4	12	2	518.62	1418.26	2121.74	1605.40	2118.19	1287.72
		3		1452.12	2304.49			
		1		2209.23	3211.89			
	16	2	700.66	2213.47	3354.42	2318.16	3078.61	1984.85
		3		2135.85	3558.34			
		1		1873.37	3045.37			
	8	2	805.09	1879.72	3254.23	2326.02	3158.43	1946.04
		3		1951.69	3728.39			
		1		3614.79	5126.18			
8	12	2	1244.68	3505.42	5285.65	3745.98	4942.21	3206.95
		3		3697.34	5958.09			
		1		4888.40	8245.64			
	16	2	2026.48	5091.61	8513.77	5408.89	7183.42	5270.13
		3		4906.04	8496.84			
		1		4338.73	7046.12			
	8	2	1728.01	4242.77	7594.37	4984.24	6768.54	4273.62
		3		4487.62	8959.00			
		1		8007.85	12200.53			
16	12	2	3009.38	7914.72	12968.22	8026.93	10590.86	7245.81
		3		7935.18	13632.90			
		1		10708.89	18678.64			
	16	2	4235.01	10490.16	20118.07	11590.35	15393.64	12212.53
		3		11106.85	19384.24			

Table 4: Post synthesis area results [μm^2] and minimum latency [ns] obtained by implementing the max* operator for parallel Log-MAP Delta and Constant Log-MAP architectures. A notes area and D denotes latency, respectively. The target clock frequency is $1/D$.

n	n_b	Log-MAP Delta		Constant Log-MAP	
		A	D	A	D
4	8	1642	1.30	1704	1.20
	12	2292	1.50	2307	1.45
	16	2850	1.55	3735	1.45
8	8	3320	1.70	4296	1.80
	12	5008	1.85	6765	2.05
	16	6354	1.95	9435	2.15
16	8	6611	2.05	9639	2.35
	12	10294	2.25	14721	2.65
	16	13191	2.35	21936	2.80

Table 5: Post synthesis area results [μm^2] for turbo code used in UMTS-LTE and CCSDS standards with a target clock frequency of 200 MHz.

	UMTS-LTE		CCSDS	
	APO	SISO	APO	SISO
Log-MAP (JL)	17820.63	37861.08	42483.47	79942.36
Log-MAP Delta (M2)	13299.13 (74.6%)	33339.58 (88.1%)	31088.03 (73.2%)	68546.92 (85.7%)
$r = 3$ Approx. [4]	13396.41 (75.2%)	33436.86 (88.3%)	31271.53 (73.6%)	68730.42 (86.0%)
$r = 4$ Approx. [4]	15788.87 (88.6%)	35829.32 (94.6%)	36399.39 (85.7%)	73858.28 (92.4%)
Constant Log-MAP	12318.35 (69.1%)	32358.80 (85.5%)	29709.29 (69.9%)	67168.18 (84.0%)
Max-Log-MAP (MX)	8393.81	28434.26	21236.43	58695.32

Table 6: As in Table 5 but for turbo code used in DVB-RCS and Wi-MAX standards. In this table, α and β denote the forward and backward metrics, respectively.

	DVB-RCS/Wi-MAX		
	α/β	APO	SISO
Log-MAP (JL)	22338.12	40289.06	98270.09
Log-MAP Delta (M2)	16547.72 (74.1%)	31246.06 (77.6%)	77646.29 (79%)
$r = 3$ Approx. [4]	17083.93 (76.5%)	31440.62 (78.0%)	78913.27 (80.3%)
$r = 4$ Approx. [4]	20821.55 (93.2%)	36225.54 (89.9%)	91173.44 (92.8%)
Constant Log-MAP	16066.20 (71.9%)	29284.50 (72.7%)	74721.69 (76.0%)
Max-Log-MAP (MX)	10593.88	21435.42	55927.97

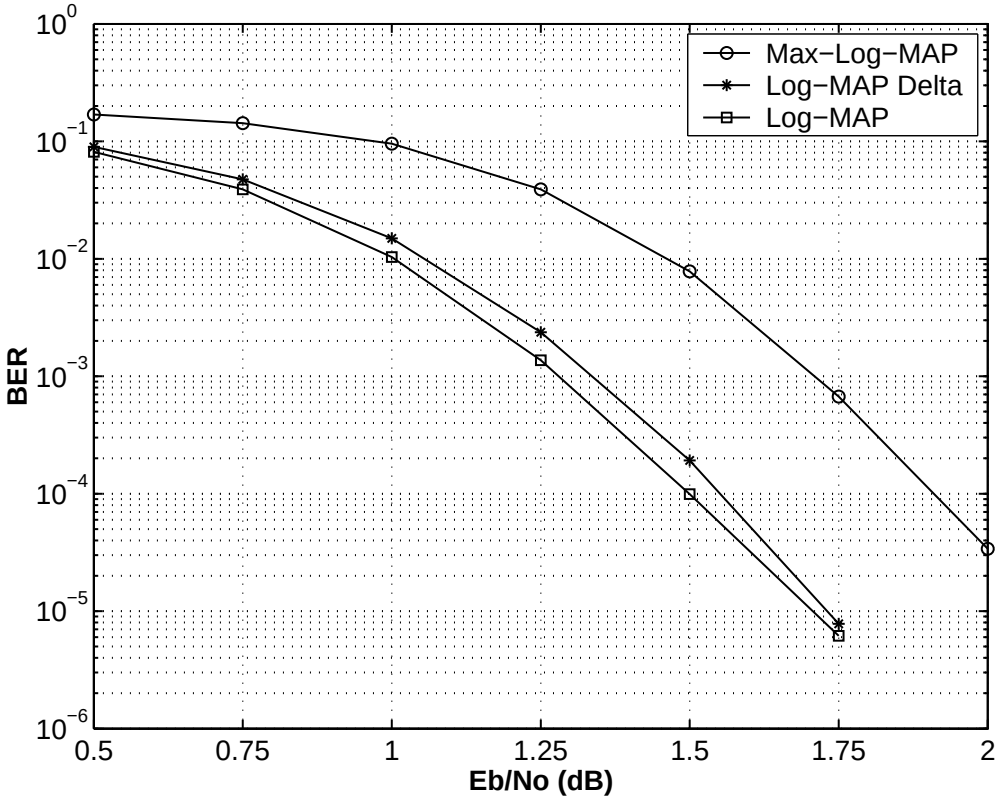

Figure 1: BER performance of binary turbo code with Max-Log-MAP, Log-MAP and the proposed decoding algorithm (denoted as Log-MAP Delta).

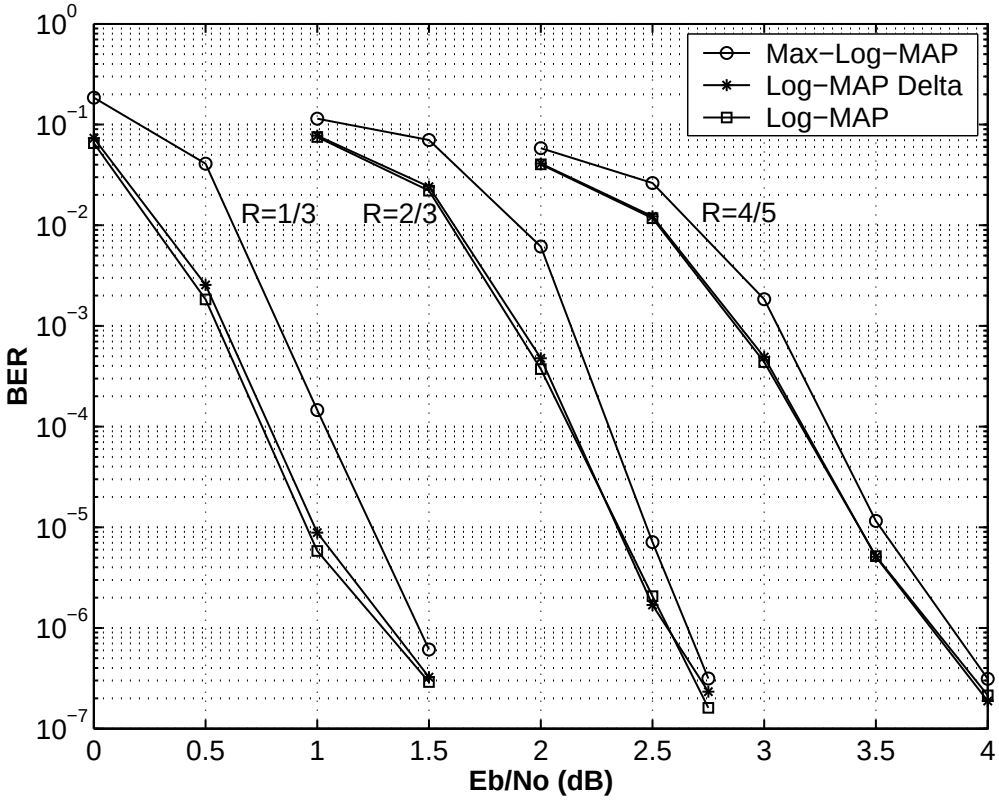

Figure 2: As in Fig. 1 but with duo-binary turbo code used in DVB-RCS and Wi-MAX standards.

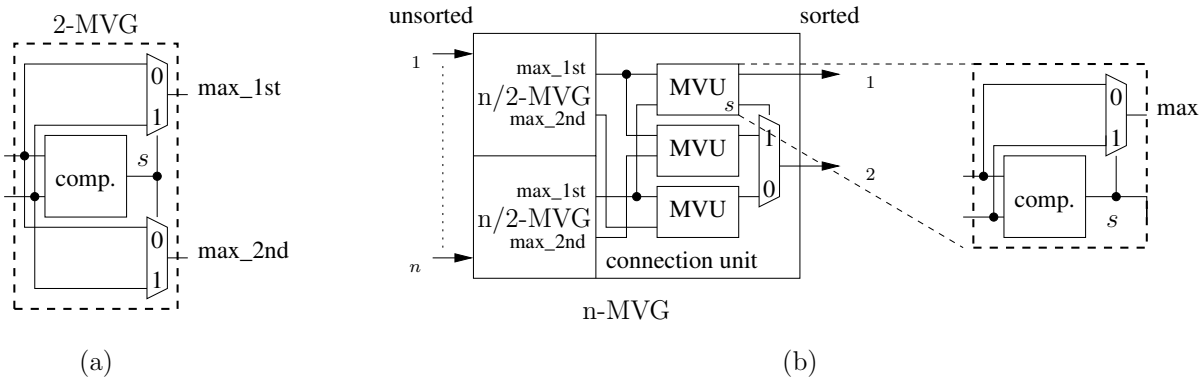

Figure 3: Block diagram of (a) 2-MVG; and (b) M2 architecture.