Further Improvements in SOVA for High-Throughput Parallel Turbo Decoding

Authors: F. J. Martin-Vega, F. Blanquez-Casado, F. J. Lopez-Martinez, Gerardo Gomez, and J. Tomas Entrambasaguas

Corresponding author: F. J. Martin-Vega (fjmvega@ic.uma.es)

Note: This is a pre-print of a paper published at IEEE Communications Letters

Abstract—In this letter, we present two mechanisms based on soft-output Viterbi algorithm (SOVA) technique that achieve a performance close to maximum a-posteriori (MAP) decoding. First, we propose a new technique called alternating direction SOVA, which allows for improving the performance of SOVA in similar terms as bidirectional SOVA while barely affecting the area complexity. Then, we introduce the idea of parallel turbo decoding based on path metric/state border exchange. Unlike other techniques, overlapping between adjacent subblocks is not required, thus reducing the decoding latency dramatically. We evaluate their performance in the context of 3GPP-LTE, showing that the combination of both mechanisms achieves a BER performance close to parallel MAP decoding while having a lower complexity.

***Index Terms*—BISOVA, LTE, max-log-MAP, parallel decoding, SOVA, turbo codes**

URBO Codes [1] have been widely used in modern mobile communication standards due to the their enormous error correction capability. There exist essentially two families of *soft-input soft-output* (SISO) algorithms for turbo decoding: the *maximum a-poteriori* (MAP) algorithm and the *soft-output Viterbi algorithm* (SOVA). The MAP algorithm offers a coding gain about 0.6 dB better than SOVA [1] at the expense of an increased latency and area occupation even in its simplified forms Max-log-MAP and Log-MAP. Several proposals have been made in order to reduce this gap in coding gain due to the appealing features of SOVA for low complexity HW implementation. In [2] a modification of SOVA that becomes equivalent to Max-Log-MAP was presented, although the proposed algorithm is significantly more complex. An interesting alternative is *bi-directional* SOVA (BISOVA) [3] in which two SOVAs decode the received sequence at each iteration so that the decoding performance is similar to MAP decoding; nevertheless, the complexity is also doubled.

In the context of modern wireless communication systems, parallel decoding becomes necessary to improve the decoding throughput. Thus, each received coded block is divided into several sub-blocks that are decoded by a different SISO decoder. However, it has two main drawbacks: (1) chip area is increased and (2) there is a loss in error correcting capability [4].

Some techniques have been proposed with the aim of alleviating this performance degradation, although they are mostly focused on MAP decoding. In [4] sub-block overlapping is proposed to increase the reliability at sub-block edges; unfortunately, this also increases the decoding latency. In [5] a method for parallel MAP decoding based on exploiting the boundary information between sub-blocks was shown to perform quite close to non-parallel decoding with no major impact in the latency.

Surprisingly, the use of SOVA for parallel decoding has not been considered in depth in the literature at the best of our knowledge: only in [6] the consideration of sub-block overlapping for SOVA was studied. This apparent lack of interest conflicts with the interest on efficient HW implementation of high-throughput parallel turbo decoders in the context of 4G [7]. This motivated us to investigate new modifications of

SOVA for parallel turbo decoding, aiming to keep the advantages of SOVA in terms of complexity while achieving a performance close to MAP decoding.

Specifically, the contribution of this work is twofold. Firstly, we propose a new architecture for non-parallel turbo decoding, namely *alternating direction* SOVA (ALSOVA), that reduces the existing gap in coding gain between conventional SOVA and MAP algorithm; in contrast to BISOVA, ALSOVA does not require to double its complexity. Secondly, we propose a new architecture for parallel turbo decoding, namely *path metric/state border exchange* (PMSBx) that alleviates the performance degradation in terms of BER associated to parallel decoding. Using the fact that the path metrics are reliable at the right side of sub-block edges, they can be used as initialization metric for the following sub-block in the next iteration. Unlike [6], PMSBx does not require for overlapping between subblocks, thus reducing the decoding latency. We also show that the combination of both methods allows for achieving a BER performance close to parallel MAP decoding, while keeping the benefits in terms of reduced complexity inherent to SOVA.

II. ARCHITECTURES FOR TURBO DECODING

A. SISO Decoding Algorithms

1) MAP Algorithm: We consider trellis terminated codes as in the case of LTE [8]. Given a received sequence of codewords **y** for the binary code (n, 1, v), being K the message length and T the number of stages required to come back to the final state, the MAP1 algorithm calculates the *a-posteriorilog likelihood ratio* (LLR) value $L(uk \mid \mathbf{y})$ for the bit uk at stage k. In order to do that, the forward $Ak(\cdot)$ and backward $Bk(\cdot)$ metrics [1] are calculated inside a window [7] whose length W must guarantee that the metric converges to reliable values. These metrics are initializated as $A_0(s) = \log(\delta(s))$ and $B_{K+T}(s) = \log(\delta(s))$ where $\delta(s)$ is the Kronecker function and $s = 0$ the zero state.

2) SOVA: This algorithm consists of two stages [9]: (*a*) calculation of weights of the ML path and (*b*) updating of those weights in order to obtain *a-posteriori* LLR values. The first step involves the calculation of the *path metric* (PM). Then the *trace back* (TB) process is carried out inside the decoding window whose length *Wd* guarantees the convergence of the surviving paths into the ML path. Once this path is known, the decoded bit sequence is obtained. In order to associate a reliability measure to each decoded bit, a preponderation of each decision (weight) is performed. When the PMs and the weights for the decoding window *Wd* have been calculated, the TB is carried out in order to obtain the weights of the ML path *Sk*. In the second step, the weights are updated as explained in detail in [1] using a updating window of length *Wu*. The initialization implies $PM_0(s) = \log(\delta(s))$ and $S_{K+T} = 0$.

B. Parallel Decoding

Due to implementations constraints, the maximum binary rate of non-parallel turbo decoders is around 40 Mbps. This is incompatible with the over 100 Mbps required for some technologies like 3GPP-LTE [7], and therefore parallel turbo decoding is necessary. This involves that each coded block of length $K + T$ is divided into P sub-blocks of length $M = K/P$ that are decoded in parallel, so the trellis termination does not hold for the nth sub-block due to the uncertainty at sub-block edges. Hence, for SOVA $PMkl(s) = \log(2^{-v})$ and the TB is initiated at stage $kr = nM$ from *any* state, being $kl = (n-1)M + 1$ the left side of nth sub-block and kr the right side. In case of MAP, $Akl(s) = \log(2^{\neg v})$ and $Bkr(s) = \log(2^{\neg v})$. For the first and last sub-blocks the initial and final states are known, respectively. The existing performance gap between MAP and SOVA grows in the parallel case as P is increased. The reason that explains this behavior is related to the direction of the metrics involved in the calculation of $L(uk)$. For the MAP algorithm, the forward metrics are not reliable at the left side of the nth sub-block due to uncertainty at sub-block edges. However, the backward metrics are reliable as long as $M \geq W$. Hence, the lack of the correct forward metric has a negative

effect in $L(uk|\mathbf{y})$ at sub-block edges, but at least it keeps a certain degree of reliability thanks to the backward metric. The opposite situation happens at the right side. In the case of SOVA, the PM will have reliable values close to sub-block edges in the right side provided that $M \geq W$. However, when the TB process is

initiated from the right side, the uncertainty in the final state causes many errors in the decoding sequence at the right side. Since the PM close to the right side is reliable, the $L(uk|\mathbf{y})$ will take high values for wrong decoded bits after updating. This overestimation provokes the appearance of errors at the subblock edges independently of the SNR.

In order to reduce the uncertainty at sub-block edges, overlapping between adjacent sub-blocks is performed [4]. Although this solution gives good results in terms of BER, it dramatically decreases the binary rate. Moreover, as the uncertainty in the sub-block edges has a greater impact on SOVA performance, a larger overlapping is required compared to MAP

III. NOVEL ALGORITHMS FOR SOVA TURBO DECODING

A. Alternating Direction SOVA (ALSOVA)

In [3] it was shown that BISOVA has a coding gain about 0.35 dB at a BER of 10^{-4} compared to SOVA. This algorithm requires two SOVA decoders at each iteration. At iteration i, one of the decoders computes *a-posteriori* measures in the forward direction $L(i)\, f\,(uk|\mathbf{y})$, and the other one does the same calculation in the backward direction $L(i)\ b\ (uk|\mathbf{y})$. Then, the *a-posteriori* measures used to obtain the extrinsic information are calculated as $L(i)(uk|\mathbf{y}) = uk \min(|L(i)\, f\,(uk|\mathbf{y})|, |L(i)\ b\ (uk|\mathbf{y})|)$. Despite the fact that the ML path is the same for the same inputs, the *a-posteriori* measures obtained in the forward and backward directions are different. The updating process in BISOVA is improved since more competing paths are taken into account [3]. This is due to the fact that some paths that are discarded before merging to theML path in the forward direction could not be discarded in the backward direction an vice-versa. This reduces the overestimation of *a-posteriori* measures associated to wrong decoded bits. However, the main drawback of BISOVA is that it requires twice as much area as the conventional SOVA. Our technique ALSOVA aims to overcome this drawback: instead of comparing forward and backward LLR values at the same iteration we combine them in different iterations. What we propose is to compute forward *a-posteriori* measures at odd iterations and backward measures at even iterations as:

$$L^{(i)}(u_k|\mathbf{y}) = \begin{cases} L_f^{(i)}(u_k|\mathbf{y}) & \text{if } \mathrm{mod}(\lfloor i \rfloor, 2) = 1 \\ L_b^{(i)}(u_k|\mathbf{y}) & \text{if } \mathrm{mod}(\lfloor i \rfloor, 2) = 0 \end{cases} \quad (1)$$

where $\lfloor i \rfloor$ is the floor function. We consider in (1) that in a complete iteration the message is decoded, whereas in a half-iteration only the interleaved message is decoded. As in [7], iterations begin from $i = 1$ and half-iterations from $i = 1.5$, respectively. The benefits of using (1) in the decoding process rely on those paths which are discarded before merging to the ML path in one direction, but not in the other one. As opposed to BISOVA, instead of using these paths to reduce overestimation of *a-posteriori* measures in the same iteration, in ALSOVA they are used along different iterations. In particular, let us consider that forward SOVA at stage k makes a wrong hard decision over the bit uk with a high overestimated LLR measure. Let us assume that backward SOVA makes a wrong hard decision as well, but without this overestimation. A turbo decoder using conventional SOVA might propagate this error since it has a high LLR measure. On the contrary, ALSOVA might propagate this error just until the first half iteration; i.e.

the overestimation is mitigated as the second iteration goes in backward direction and the error does not propagate any further.

Furthermore, we must note that this strategy does not increase the area occupation since only one SOVA decoder is used per iteration and the same decoder can be used to perform decoding in both directions.

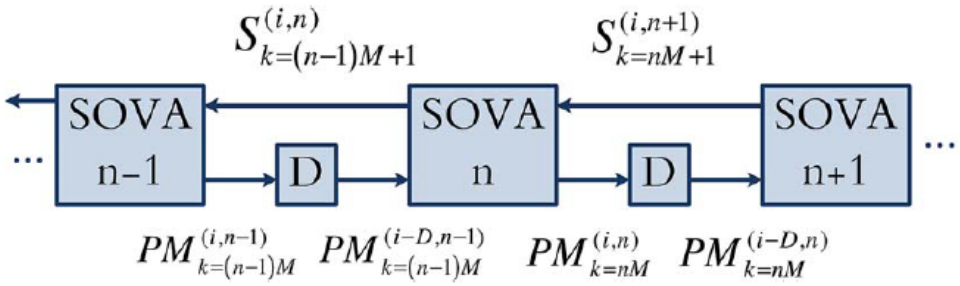

Fig. 1. Bank of SOVA decoders using PMSBx for parallel turbo decoding.

B. Path Metric/State Border Exchange (PMSBx)

In the previous section, we stated that the performance of parallel turbo decoding using SOVA is degraded due to the uncertainty at the sub-block edges. Specifically, the PM at the right side of sub-block edges is reliable whereas the state used to initiate TB is unreliable. For the left side of sub-block edges, the situation is just the opposite. Here, we propose a new architecture that takes advantage of the different reliability of PM and states of the ML path in order to improve the decoding performance. This technique, depicted in Fig. 1 will be denoted as PMSBx, and it is based upon two main premises:

1) Convergence of Path Metrics: As the left side of one subblock is the right side of the previous sub-block, the reliable PM at the right side of one sub-block can be used as initial PM for the following sub-block in the next iteration. This idea was exploited in [5], and is referred to as sub-block boundary information [10].

2) Convergence of States From ML Path: With reliable initial metric values, the states of the ML path will be reliable at the left side of sub-block edges. These states produced by one SOVA decoder can be used to do the last TB from a reliable state in the previous SOVA decoder, thus reducing the dramatic overestimation in *a-posteriori* LLR values. As the states at the left side are available before the TB at the right side, these states can be exchanged in the same iteration. Note that this does not hold for the PMs, which have to be exchanged between full iterations. Interestingly, PMSBx can be combined with either SOVA, BISOVA or the proposed ALSOVA.

The initialization of PM at sub-block *n* and iteration *i* follows

$$\mathrm{PM}_{k_p}^{(i,n)}(s) = \begin{cases} \mathrm{PM}_{k_p}^{(i-D,n-1)}(s), & \text{if } n \neq 1 \\ \log\left(\delta(s)\right), & \text{if } n = 1 \end{cases} \qquad (2)$$

being $kp = (n-1)M$ the stage at which PM should be initialized and D the delay that is applied in terms of full iterations (i.e. two half-iterations). For SOVA-PMSBx, we have $D = 1$as all iterations go in the same direction. The final state at the nth sub-block where the TB is initiated is

$$\mathrm{S}_{k_s}^{(i,n)} = \begin{cases} \mathrm{S}_{k_s}^{(i,n+1)}, & k_s = nM+1 \quad \text{if } n \neq P \\ 0, & k_s = K+T \quad \text{if } n = P \end{cases} \tag{3}$$

Notice that ks is the stage at which TB should be initiated at the right side of sub-block n. The architecture illustrated in Fig. 1 is valid for a bank of SOVA decoders in both forward and backward directions. In the case of BISOVA-PMSBx and ALSOVA-PMSBx there would be two different kinds of metrics and TBs, one in the forward direction and another one in the backward direction. For BISOVA this issue does not lead to any modification with respect to SOVA-PMSBx since the initialization is carried out inside two independent banks of SOVA decoders. Each of them has the structure presented in Fig. 1 and uses $D = 1$. However, for ALSOVA-PMSBx we have to use $D = 2$in equation (2) since the PMs in the previous iteration have been calculated in a different direction.

3) Throughput Enhancement and Area Overhead: In order to assess the benefits of the proposed parallel method, throughput and area occupation metrics need to be carefully analyzed. The throughput of a parallel turbo decoder can be expressed as $\Gamma(bps) = Kf0/2I(K/P + O)$ where $f0$ is the system clock frequency in Hz, I is the number of iterations and O the number of overlap cycles. Hence, the throughput enhancement of PMSBx over OL can be expressed as $\Delta\Gamma = O \cdot P/K$. We see how this $\Delta\Gamma > 1$, and depends linearly on the number of parallel stages P. Taking some values from LTE standard, we have that K ranges from 40 to 6144, and P is expected to be high in order to fulfil the throughput requirements. As an example, and assuming an overlap of 60 cycles, we have that for values $K = 160$ bits and $P = 4$, the throughput improvement is 150%. If we consider $K = 2048$ and $P = 8$, this yields a throughput increment of about 23%. In the limit case of $K = 6144$, we see that $\Delta\Gamma \approx P$%, i.e. a throughput improvement of an 8% for $P = 8$.

The increment in area is mostly related to the increment in memory, which varies depending on the particular algorithm (i.e. SOVA, BISOVA or ALSOVA). In the case of SOVA, the main factor consuming memory are the PMs, or equivalently, the weights that must be stored for each state. If we consider a bank of P parallel SOVAs, we haveWd^2vqpP bits to be stored, being qp the word length in bits. In the case of SOVA-PMSBx, the exchanged PMs must be additionally stored, yielding a total of $2 \cdot 2vqq(P-1)$ bits. In the case of BISOVA-PMSBx, the memory consumption is just the double of SOVA-PMSBx since it is composed of two independent banks of SOVA decoders.

Finally, in the case of ALSOVA-PMSBx, the exchange of PMs requires more memory since they are stored in four half-iterations, thus the memory consumption is $4 \cdot 2vqq(P-1)$. For $P>>1$, this lead to the following increments in memory for SOVA-PMSBx, BISOVA-PMSBx and ALSOVA-PMSBx compared to overlap versions of each algorithm: $\Delta S = \Delta B = 2/Wd$ and $\Delta A = 4/Wd$, respectively. For a typical window of 20 stages, the memory requirement is increased $\Delta S = \Delta B = 10\%$, and $\Delta A = 20\%$, respectively.2window of 20 stages, the memory requirement is increased $\Delta S = \Delta B = 10\%$, and $\Delta A = 20\%$, respectively.

IV. SIMULATION RESULTS

We now assess the performance of the proposed algorithms in the context of 3GPP-LTE technology. We consider a message length of $K = 4096$ bits and BPSK modulation over an AWGN channel. In order to

obtain a performance close to real implementations, we use a sliding window with $W = 20$ stages. for MAP, whereas for SOVA we use decoding and updating windows of $Wd = 20$ and $Wu = 10$ respectively. Results are shown for 6 iterations as BER is not reduced significantly after the 6*th* iteration.

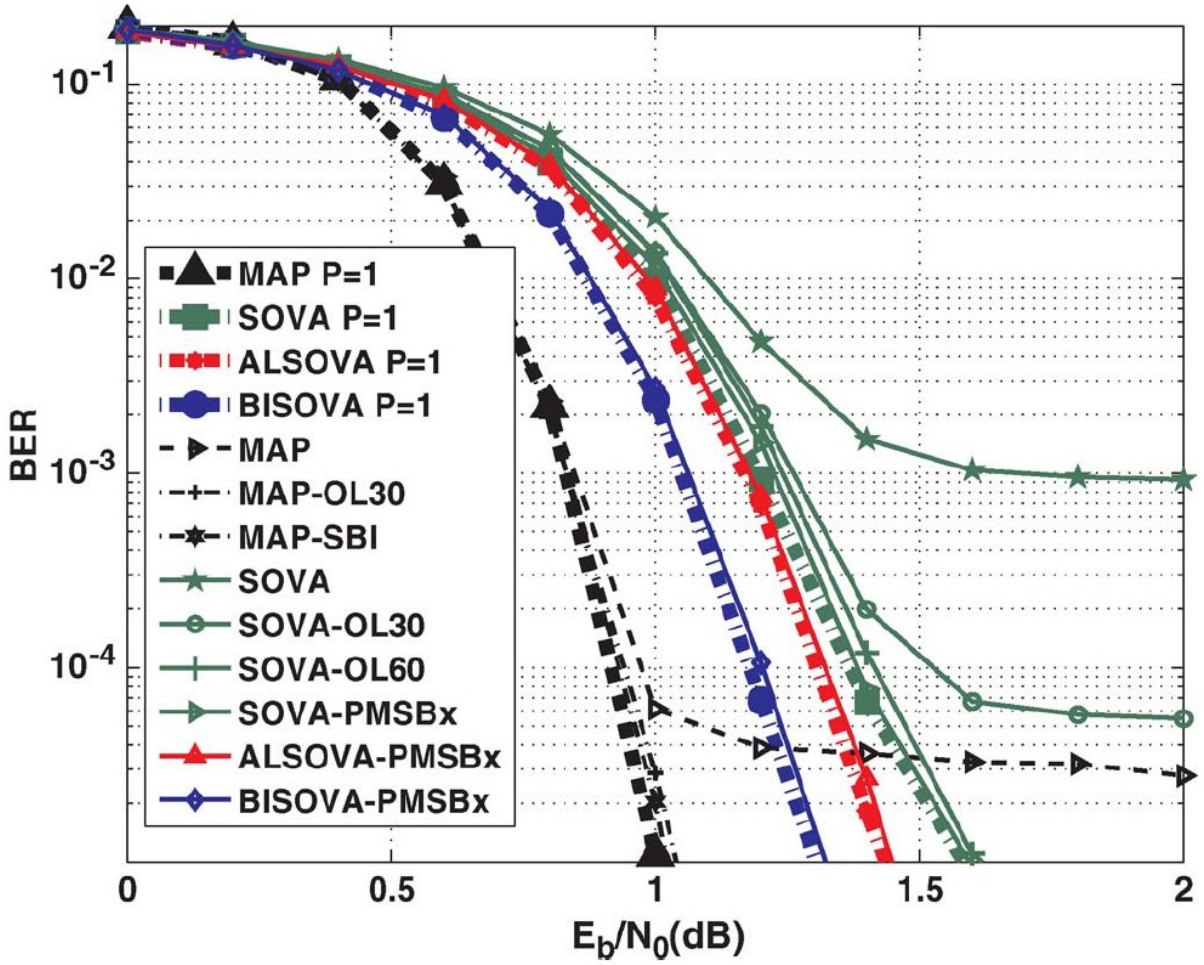

Fig. 2. BER performance vs. E_b/N_0 at $6th$ iteration for different algorithms in the parallel (thin line, $P = 4$) and non-parallel (thick line, $P = 1$) cases.

Fig. 2 shows the BER for different algorithms in the parallel ($P = 4$, thin line) and non-parallel ($P = 1$, thick line) cases. We first see that the performance in terms of BER for ALSOVA is in-between BISOVA and SOVA, but conversely to BISOVA, ALSOVA does not require more area occupation. We also observe that the performance gain between SOVA and MAP in parallel turbo decoding is increased since SOVA overestimates the obtained LLR values. This confirms that neither SOVA nor MAP should be used without a proper method to reduce uncertainty at sub-block edges. The performance of *overlap* (OL) with 30 cycles (15 per each of the two edges of each sub-block) is illustrated for SOVA and MAP. We notice that 30 cycles is not enough for SOVA-OL since the BER tends to saturate around *Eb/N*0 = 1.5 dB. SOVA-OL with 60 cycles of overlap solves this issue at the expense of an even higher latency. Additionally, Fig. 2 shows that the BER performance of SOVA, BISOVA and ALSOVA using PMSBx reaches the performance of the non-parallel case for $P = 4$. Unlike OL based methods, this occurs without any increase in terms of latency, and thus obtaining higher throughput.

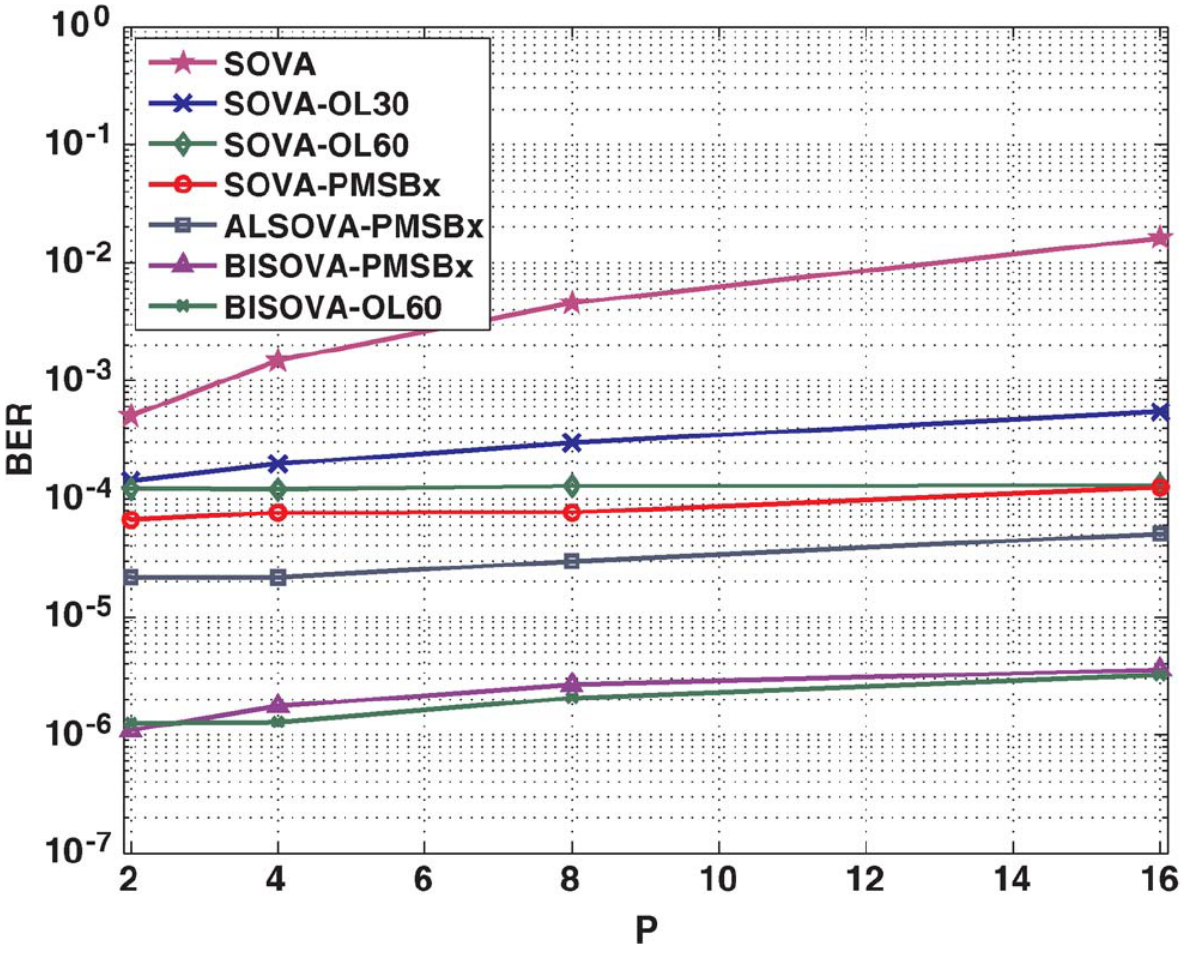

Fig. 3. BER performance vs. P at $6th$ iteration for different algorithms in the parallel case ($E_b/N_0 = 1.4$ dB).

Fig. 3 illustrates the BER performance at *Eb/N*0 = 1.4 dB for different parallelization levels *P*. The benefits of increasing the parallelization level are clear since the throughput of the turbo decoder is increased with *P*. Nevertheless, note that *P* >8 is not recommended in practice due to area constraints [7]. We observe that the performance slightly worsens as *P* increases. However, PMSBx techniques exhibit much better performance than OL-based techniques regardless of the type of SOVA. The benefits of BISOVA-PMSBx and ALSOVAPMSBx are considerable, being their performance quite stable with *P*.

V. CONCLUSION

We presented two novel techniques for SOVA turbo decoding: 1) ALSOVA, which improves the correction capabilities of conventional SOVA without any increase in area occupation or latency, and which can be used both for non-parallel and parallel decoding; 2) PMSBx, which brings improved performance for parallel decoding over classical methods based on sub-block overlapping, both in terms of BER and latency. PMSBx can be combined with existing SOVA, BISOVA and the proposed ALSOVA, allowing the designers to have new mechanisms to improve the performance of parallel decoding depending on their area constraints.

REFERENCES:

[1] J. Woodard and L. Hanzo, “Comparative study of turbo decoding techniques: An overview,” *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.

[2] M. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, “On the equivalence between SOVA and max-log-MAP decodings,” *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137–139, May 1998.

[3] J. Chen, M. Fossorier, S. Lin, and C. Xu, “Bi-directional SOVA decoding for turbo-codes,” *IEEE Commun. Lett.*, vol. 4, no. 12, pp. 405–407, Dec. 2000.

[4] J.-M. Hsu and C.-L.Wang, “A parallel decoding scheme for turbo codes,” in *Proc. IEEE ISCAS*, May/Jun. 1998, pp. 445–448.

[5] S. Yoon and Y. Bar-Ness, “A parallel MAP algorithm for low latency turbo decoding,” *IEEE Commun. Lett.*, vol. 6, no. 7, pp. 288–290, Jul. 2002.

[6] Z. Wang and K. Parhi, “High performance, high throughput turbo/SOVA decoder design,” *IEEE Trans. Commun.*, vol. 51, no. 4, pp. 570–579, Apr. 2003.

[7] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, “Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE,” *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 8–17, Jan. 2011.

[8] “3GPP Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding,” Sophia Antipolis Cedex, France, TS 36.212, Sep. 2008.

[9] C. Berrou, P. Adde, E. Angui, and S. Faudeil, “A low complexity soft-output Viterbi decoder architecture,” in *Proc. IEEE ICC*, May 1993, pp. 737–740.

[10] L.Wang, H. Yang, and H. Yang, “A novel method for parallel decoding of turbo codes,” in *Proc. 4th IEEE ICCSC*, May 2008, pp. 768–772.