This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs

A Novel Area-Efficient VLSI Architecture for Recursion Computation in LTE Turbo Decoders

Arash Ardakani and *Mahdi Shabany*

***Abstract*—Long term evolution (LTE) is aimed to achieve the peak data rates in excess of 300 Mb/s for the next generation wireless communication systems. Turbo codes, the specified channel coding scheme in LTE, suffer from a low-decoding throughput due to its iterative decoding algorithm. One efficient approach to achieve a promising throughput is to use multiple Maximum a-Posteriori (MAP) cores in parallel, resulting in a large area overhead. The two computationally challenging units in an MAP core are α and β recursion units. Although several methods have been proposed to shorten the critical path of these recursion units, their area-efficient architecture with minimum silicon area is still missing. In this paper, a novel relation existing between α and β metrics is introduced, leading to a novel add-compare-select (ACS) architecture. The proposed technique can be applied to both the precise approximation of log-MAP and max-log MAP ACS architectures. The proposed ACS design, implemented in a 0.13 μm CMOS technology and customized for the LTE standard, results in at most 18.1% less area compared to the reported designs to-date while maintaining the same throughput level.**

***Index Terms*—Add-Compare-Select (ACS) unit, Long Term Evolution (LTE), parallel architecture, radix-4, recursion unit, turbo decoder.**

I. Introduction

Many advanced wireless communication standards adopted turbo codes as the channel coding scheme due to its near Shannon error-correcting performance [1]. The decoding procedure is performed in two different half iterations, where the reliability of received bits are computed in the form of extrinsic values using interleavers and soft-input soft-output (SISO) decoders in an iterative way. On even half iterations, the decoding process is performed on the non-interleaved data and parity, while on odd half iterations, the interleaved data is decoded. The extrinsic values, representing the reliability of the information bits, are sent to another half iteration by passing through the interleaver/deinterleaver unit till the acceptable error level is achieved.

Recently, Long Term Evolution (LTE)-advanced has been dominated as the next generation wireless communication standard, which is aimed at higher peak data rates close to 3 Gbps [2]. The turbo decoder, specified in LTE, reveals to be a limiting block toward this goal due to its iterative decoding nature, high latency and significant silicon area consumption. The decoding procedure is performed using the algorithm presented in [3]. Since the implementation of the actual MAP algorithm incurs a very high computational complexity, typically, two modified forms of the MAP algorithm, i.e., the Max-log-MAP and log-MAP algorithms [4], [5], are commonly realized instead.

A. Ardakani and M. Shabany are with the Electrical Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: ardakani.arash@gmail.com; mahdi@sharif.edu).

In these two alternative methods, the MAP core consists of log likelihood ratio (LLR) units, as well as the core units to compute α, β, γ, the forward, backward and branch metrics, respectively. In fact, the α and β units, due to their recursive computation nature, are the most challenging units to implement, occupying almost 40% of the whole MAP core area [6]. The γ unit, on the other hand, is a trivial part of the turbo decoder, consisting of few addition computations. Therefore, an area-efficient architecture for α and β metrics computation is highly desirable, which has always been a challenge in literature.

In order to address this challenge, in this paper, a new relation among α and β metrics is introduced; based on this new relation, a novel add-compare-select (ACS) unit for forward and backward computation is proposed. The proposed scheme results in at most a 18.1% reduction in the silicon area compared to the designs reported to-date.

II. Turbo Decoder Algorithm

The MAP algorithm, which provides the a posteriori probability for each bit is used in iterative decoding of turbo codes. The MAP algorithm provides the probability of the decoded bit u_k being either +1 or -1 for the received symbol sequence y by calculation of the LLR values as follows:

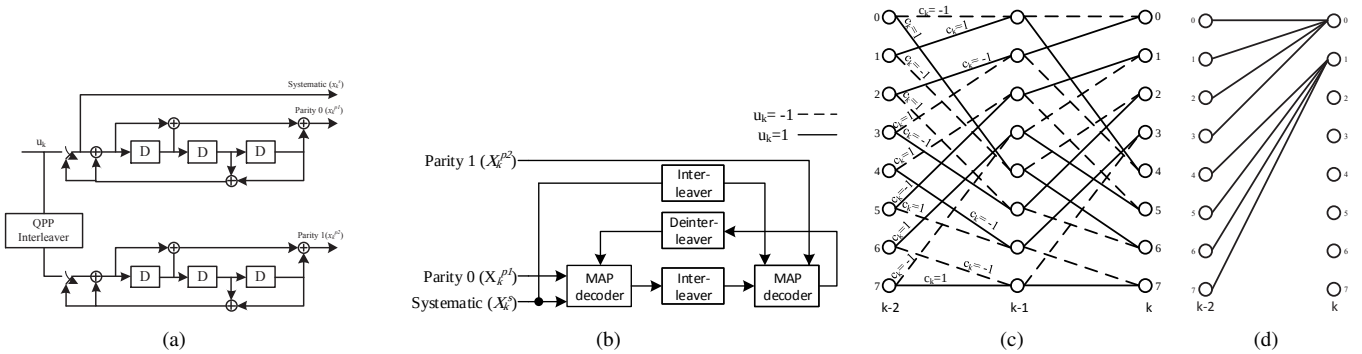
$$L(u_k|y) = \log[\frac{p(u_k = +1|y)}{p(u_k = -1|y)}], \tag{1}$$

where $p(u_k = +1|y)$ and $p(u_k = -1|y)$ denote the probability of bit u_k being +1 and -1, respectively.

The turbo decoder specified in LTE consists of two recursive convolutional encoders, an interleaver and a feed-through path, as shown in Fig. 1(a). The feed-through passes one block of K information bits, called systematic bits x_k^s, where $k = 0, 1, \ldots, K-1$. The parity generated by the convolutional encoder is denoted by x_k^{p1}. By permuting the systematic bits via the interleaver, the second sequence of parity is generated by passing through the second convolutional encoder, denoted by x_k^{p2}. On the receiver side, the reliability of bits is computed iteratively by exchanging the extrinsic LLRs between two SISO decoders based on (1), as depicted in Fig. 1(b). Another representation of a convolutional encoder is by using a trellis diagram, as shown in Fig. 1(c), depicting two steps of the LTE turbo encoder.

Applying few mathematical manipulations on (1), leads to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs



(a) (b) (c) (d)

Fig. 1. (a) Turbo encoder, (b) Turbo decoder, (c) Radix-2 trellis diagram, (d) Partial radix-4 trellis diagram.

the following:

$$LLR(u_k) = \log\left(\frac{\sum_{u_k=+1} \tilde{\alpha}_{k-1}(s')\ \tilde{\beta}_k(s)\ \tilde{\gamma}_k(s',s)}{\sum_{u_k=-1} \tilde{\alpha}_{k-1}(s')\ \tilde{\beta}_k(s)\ \tilde{\gamma}_k(s',s)}\right), \quad (2)$$

where $\tilde{\alpha}_k(s)$, $\tilde{\beta}_k(s)$ and $\tilde{\gamma}_k(s',s)$ denote the forward, backward and branch metrics, respectively. The s and s' indexes are also associated with trellis steps k and $k-1$, respectively. The MAP algorithm traverses in both forward and backward directions to get state metrics $\tilde{\alpha}_k(s)$ and $\tilde{\beta}_k(s)$, respectively. The transmission value in the k-th stage from the state s' to state s is denoted by $\tilde{\gamma}_k(s',s)$. The calculation of $\tilde{\alpha}_k(s)$, $\tilde{\beta}_k(s)$ and $\tilde{\gamma}_k(s',s)$ metrics are performed as follows:

$$\tilde{\alpha}_k(s) = \sum_{s'} \tilde{\gamma}_k(s',s)\ \tilde{\alpha}_{k-1}(s'), \quad (3)$$

$$\tilde{\beta}_{k-1}(s') = \sum_{s} \tilde{\gamma}_k(s',s)\ \tilde{\beta}_k(s), \quad (4)$$

$$\tilde{\gamma}_k(s',s) = \exp\left[\frac{1}{2}L_e(u_k)u_k + \frac{1}{2}L_c X_k^s u_k + \frac{1}{2}L_c X_k^p c_k\right], \quad (5)$$

where X_k^s and X_k^p are the received soft inputs corresponding to transmitted bits x_k^s and x_k^p, respectively. The value of $L_e(u_k)$ denotes the extrinsic value of u_k, and L_c is the channel reliability measure. The u_k and c_k are transmitted values of the systematic and parity bits, respectively, which can be either +1 or -1.

Due to the high computational complexity of the MAP algorithm, which is as a result of the exponential and multiplication calculations, typically an equivalent logarithmic form is employed, where a multiplication is converted to an addition. In this case, the corresponding equations in (2), (3), (4) and (5) can be reformulated as follows:

$$\alpha_k(s) = \log[\sum_{s'} \exp(\gamma'_k(s',s) + \alpha_{k-1}(s'))], \quad (6)$$

$$\beta_{k-1}(s') = \log[\sum_{s} \exp(\gamma'_k(s',s) + \beta_k(s))], \quad (7)$$

$$\gamma'_k(s',s) = \frac{1}{2}L_e(u_k)\ u_k + \frac{1}{2}L_c\ X_k^s\ u_k + \frac{1}{2}L_c\ X_k^p\ c_k. \quad (8)$$

where α, β and γ' are defined as follows:

$$\alpha_k(s) = \log(\tilde{\alpha}_k(s)), \quad (9)$$

$$\beta_k(s) = \log(\tilde{\beta}_k(s)), \quad (10)$$

$$\gamma'_k(s',s) = \log(\tilde{\gamma}_k(s',s)). \quad (11)$$

The above logarithmic formulation of the MAP algorithm is used to make the implementation of this algorithm feasible. The γ' values, according to (8), can be readily realized through few additions, not critical in hardware. In fact, the computation of α, β and LLR values makes the major computation part of the algorithm, occupying the major fraction of the silicon area. In order to implement the logarithmic computations efficiently in hardware, two common approaches are normally used, namely the max-log MAP and precise approximation of log-MAP algorithms, described in the sequel.

Consider the following equation, used to implement the logarithm:

$$\begin{aligned}\max*(z,t) &= \log(e^z + e^t) \\ &= \max(z,t) + \log(1 + e^{-|z-t|}),\end{aligned} \quad (12)$$

where the $\max$ function denotes the maximum value. In the precise approximation of log-MAP method, the first term in (12), i.e. $\max(z,t)$, can be easily implemented by a comparator while the second term, i.e. $\log(1+e^{-|z-t|})$, is implemented using a lookup table (LUT), see [7]. On the other hand, the max-log MAP method relies on the approximation of $\log(e^z + e^t)$ by the maximum of z and t, i.e., $\max*(z,t) \approx \max(z,t)$, see [6]. The hardware realization of the max-log MAP results in a lower critical path and computational complexity compared to the precise approximation of log-MAP implementation, while its performance loss is an inevitable side effect.

By using either the max-log MAP or precise approximation of log-MAP algorithm, the recursive computation is performed as follows:

$$\alpha_k(s) = \max*[\sum_{s'} \gamma'_k(s',s) + \alpha_{k-1}(s')], \quad (13)$$

$$\beta_{k-1}(s') = \max*[\sum_{s} \gamma'_k(s',s) + \beta_k(s)]. \quad (14)$$

In order to improve the processing speed of the decoding algorithm, a radix-4 architecture is generally used [8] by incorporating two stages of the trellis diagram, as partially shown in Fig. 1(d). In this case, two LLR values are produced simultaneously per clock cycle, increasing the throughput by a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs

TABLE I
SETS OF EQUAL γS FOR RADIX-4 COMPUTATION

$(u_{k-1}, c_{k-1}, u_k, c_k)$	Set of equal gammas $[\gamma_k(s'', s)]$	Symbol
(+1,+1,+1,+1)	$\gamma_k(1,4), \gamma_k(7,7)$	$\gamma_k(1)$
(+1,+1,+1,-1)	$\gamma_k(0,6), \gamma_k(6,5)$	$\gamma_k(2)$
(+1,+1,-1,+1)	$\gamma_k(0,2), \gamma_k(6,1)$	$\gamma_k(3)$
(+1,+1,-1,-1)	$\gamma_k(1,0), \gamma_k(7,3)$	$\gamma_k(4)$
(+1,-1,+1,+1)	$\gamma_k(2,0), \gamma_k(4,3)$	$\gamma_k(5)$
(+1,-1,+1,-1)	$\gamma_k(5,1), \gamma_k(3,2)$	$\gamma_k(6)$
(+1,-1,-1,+1)	$\gamma_k(3,6), \gamma_k(5,5)$	$\gamma_k(7)$
(+1,-1,-1,-1)	$\gamma_k(2,4), \gamma_k(4,7)$	$\gamma_k(8)$
(-1,+1,+1,+1)	$\gamma_k(3,0), \gamma_k(5,3)$	$-\gamma_k(8)$
(-1,+1,+1,-1)	$\gamma_k(4,1), \gamma_k(2,2)$	$-\gamma_k(7)$
(-1,+1,-1,+1)	$\gamma_k(4,5), \gamma_k(2,6)$	$-\gamma_k(6)$
(-1,+1,-1,-1)	$\gamma_k(5,7), \gamma_k(3,4)$	$-\gamma_k(5)$
(-1,-1,+1,+1)	$\gamma_k(0,4), \gamma_k(6,7)$	$-\gamma_k(4)$
(-1,-1,+1,-1)	$\gamma_k(7,5), \gamma_k(1,6)$	$-\gamma_k(3)$
(-1,-1,-1,+1)	$\gamma_k(7,1), \gamma_k(1,2)$	$-\gamma_k(2)$
(-1,-1,-1,-1)	$\gamma_k(0,0), \gamma_k(6,3)$	$-\gamma_k(1)$

factor of two [9]. The transmission metrics are also computed as follows:

$$\gamma_k(s'', s) = \gamma'_{k-1}(s'', s') + \gamma'_k(s', s), \tag{15}$$

where s'' denotes the $(k-2)$-th stage of the trellis diagram and the recursion computations are performed according to the transmission values of each step.

III. A FORMULATION FOR RADIX-4 RECURSION COMPUTATION

The encoder and trellis diagram of the LTE standard, consisting of 8 states, are shown in Fig. 1(a) and 1(c), respectively. The radix-4 trellis diagram is partially shown in Fig. 1(d). According to (15) and the fact that u_k and c_k can take one of the two values of +1 or -1, the γ values of each stage in the trellis diagram have 32 possible values, with half of them being negative, which are summarized in Table I. According to γ values in Table I and (14), the backward state values in the k-th stage can be written as follows:

$$\begin{aligned}\beta_{k-2}(0) = \max*\{&\beta_k(0) - \gamma_k(1), \beta_k(4) - \gamma_k(4),\\ &\beta_k(2) + \gamma_k(3), \beta_k(6) + \gamma_k(2)\},\end{aligned} \tag{16}$$

$$\begin{aligned}\beta_{k-2}(1) = \max*\{&\beta_k(0) + \gamma_k(4), \beta_k(4) + \gamma_k(1),\\ &\beta_k(2) - \gamma_k(2), \beta_k(6) - \gamma_k(3)\},\end{aligned} \tag{17}$$

$$\begin{aligned}\beta_{k-2}(2) = \max*\{&\beta_k(0) + \gamma_k(5), \beta_k(4) + \gamma_k(8),\\ &\beta_k(2) - \gamma_k(7), \beta_k(6) - \gamma_k(6)\},\end{aligned} \tag{18}$$

$$\begin{aligned}\beta_{k-2}(3) = \max*\{&\beta_k(0) - \gamma_k(8), \beta_k(4) - \gamma_k(5),\\ &\beta_k(2) + \gamma_k(6), \beta_k(6) + \gamma_k(7)\},\end{aligned} \tag{19}$$

$$\begin{aligned}\beta_{k-2}(4) = \max*\{&\beta_k(3) + \gamma_k(5), \beta_k(7) + \gamma_k(8),\\ &\beta_k(1) - \gamma_k(7), \beta_k(5) - \gamma_k(6)\},\end{aligned} \tag{20}$$

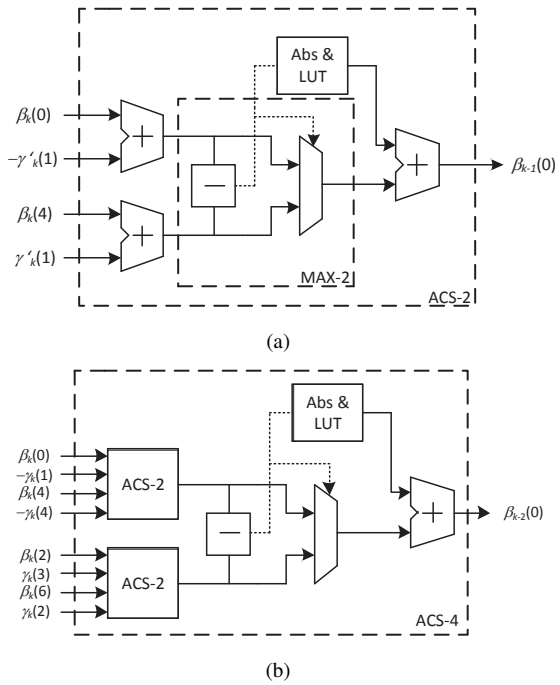


Fig. 2. (a) Conventional radix-2 ACS unit, (b) conventional radix-4 ACS unit.

$$\begin{aligned}\beta_{k-2}(5) = \max*\{&\beta_k(3) - \gamma_k(8), \beta_k(7) - \gamma_k(5),\\ &\beta_k(1) + \gamma_k(6), \beta_k(5) + \gamma_k(7)\},\end{aligned} \tag{21}$$

$$\begin{aligned}\beta_{k-2}(6) = \max*\{&\beta_k(7) - \gamma_k(4), \beta_k(3) - \gamma_k(1),\\ &\beta_k(5) + \gamma_k(2), \beta_k(1) + \gamma_k(3)\},\end{aligned} \tag{22}$$

$$\begin{aligned}\beta_{k-2}(7) = \max*\{&\beta_k(7) + \gamma_k(1), \beta_k(3) + \gamma_k(4),\\ &\beta_k(5) - \gamma_k(3), \beta_k(1) - \gamma_k(2)\}.\end{aligned} \tag{23}$$

Similar equations hold true for α values.

In this paper, a smart relation among these formulas is introduced, leading to an optimized hardware implementation, which is described in the sequel.

IV. ACS ARCHITECTURE

The common approach to implement the recursion unit is by using the ACS architecture. In this case, the radix-2 recursion unit is implemented by using an adder, a comparator unit and a selector unit dictated by (12), shown in Fig. 2(a), where common approximations of the logarithmic term in $\log(1 + e^{-|z-t|})$ is used for implementing the LUT. Few designs such as the one in [10] have been proposed to reduce the latency of this architecture, all in radix-2. However, in recent wireless communication systems with a clear demand for a high-throughput framework, a radix-4 architecture is a common approach [11] and should be efficiently designed.

Fig. 2(b) shows the architecture of a radix-4 ACS unit consisting of three radix-2 ACS units. The main advantage of using a radix-4 architecture is its concurrent computation of two-bit recursion metrics leading to a higher throughput. However, compared to its radix-2 counterpart, it has a higher latency and silicon area overhead. Therefore, due to the nature

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs

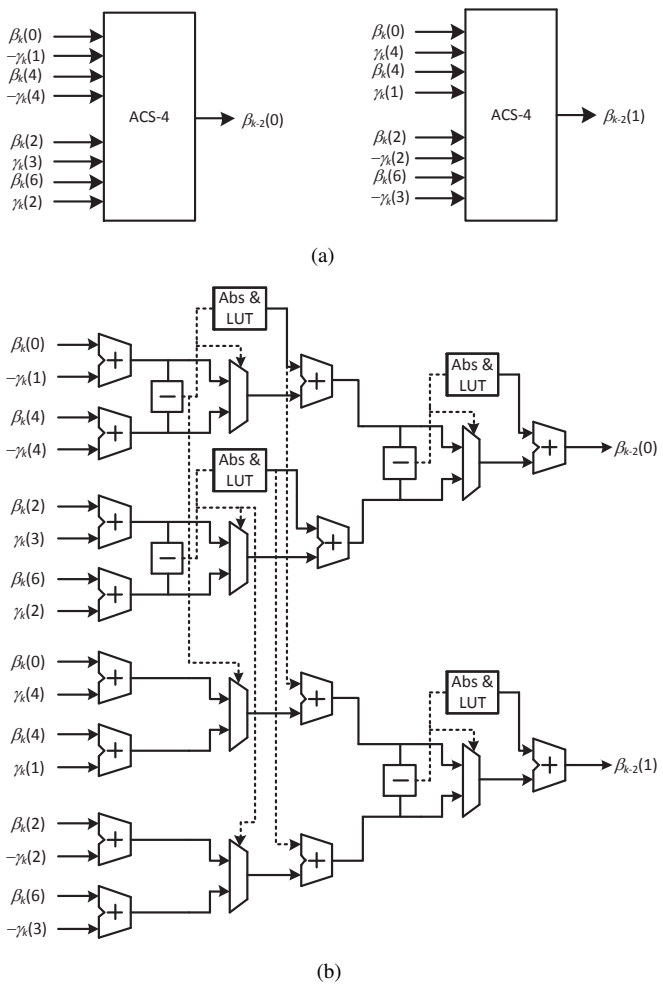


Fig. 3. (a) Conventional radix-4 ACS Architecture for two concurrent metrics computation, (b) The proposed radix-4 ACS architecture based on conventional architecture for two concurrent metrics computation.

of the recursive computation, which highly restricts the clock frequency, achieving a high throughput is by far a more challenging task in a radix-4 framework. Although, several designs have been proposed so far to shorten the latency of the radix-4 architectures, a large silicon area overhead is their unwanted byproduct [10]. Therefore, the goal of this paper is to alleviate the area overhead of radix-4 architectures.

V. Proposed Scheme

According to (16) and (17), the following expressions are obtained:

$$\beta_{k-2}(0) = \max*\{A, B\}, \tag{24}$$

$$\beta_{k-2}(1) = \max*\{C, D\}, \tag{25}$$

where A, B, C and D can be written as follows:

$$A = \max*\{\beta_k(0) - \gamma_k(1), \beta_k(4) - \gamma_k(4)\}, \tag{26}$$

$$B = \max*\{\beta_k(2) + \gamma_k(3), \beta_k(6) + \gamma_k(2)\}, \tag{27}$$

$$C = \max*\{\beta_k(0) + \gamma_k(4), \beta_k(4) + \gamma_k(1)\}, \tag{28}$$

$$D = \max*\{\beta_k(2) - \gamma_k(2), \beta_k(6) - \gamma_k(3)\}. \tag{29}$$

To realize the $\beta_{k-2}(0)$ value, each of A and B values are proposed to be implemented by a radix-2 architecture and finally a third radix-2 architecture is used to achieve (24). The proposed scheme is shown in Fig. 2(b). The value of $\beta_{k-2}(1)$ can also be similarly implemented as depicted in Fig. 3(a). A radix-2 architecture employs a comparator and LUT dealing with distances between two input values to select the maximum value, which then adds the selected amount to the maximum value.

It is worth noting that the distance between two input values of (26) is $\beta_k(0) - \beta_k(4) + \gamma_k(4) - \gamma_k(1)$, which is equal to the distance between two input values of (28). The distances between each two input values of (27) and (29) are also equal. Therefore, the compare and LUT unit for the computation of (28) and (29) are omitted leading to a novel architecture, shown in Fig. 3(b). Hereafter, this proposed architecture is referred to as the Maximum Shared Resource (MSR) architecture. This property is true for each pair of $\{(16), (17)\}$, $\{(18), (19)\}$, $\{(20), (21)\}$ and $\{(22), (23)\}$ for the backward recursion metrics and also for each pair of $\{\alpha(0), \alpha(4)\}$, $\{\alpha(1), \alpha(5)\}$, $\{\alpha(2), \alpha(6)\}$ and $\{\alpha(3), \alpha(7)\}$ for forward recursion metrics. In fact, using the proposed MSR architecture, the redundant computation is avoided, alleviating the area overhead in conventional schemes.

VI. Implementation Results

In fact, the proposed MSR is introduced to efficiently reduce the silicon area overhead caused by using the radix-4 implementation platform. The proposed MSR technique can be easily applied to any other architecture existing to-date for the LTE turbo code. In order to observe the impact of the MSR method, this scheme is applied to recent architectures. The synthesis results of both the original architectures and the applied MSR version to previous ACS architectures presented in literature are shown in Table II. For a fair comparison, all architectures were synthesized using the Synopsys Design Compiler in a 0.13 μm CMOS technology and the results are given in terms of the equivalent gate count. Furthermore, the results in Table II are for computing all forward and backward metrics that must be computed for the LTE turbo decoder. The proposed MSR technique can provide 15% reduction in the area when it is applied to the conventional architecture (Fig. 2(b)), which implements the precise log-MAP algorithm. This reduction is a result of the omission of two ALUT units and two subtraction units for each two metrics out of eight possible metrics. Moreover, the MSR technique can also be applied to few recent architectures that are devised to reduce the critical path of the conventional ACS architecture such as the designs in [9], [10], [12]. By applying the MSR method to these schemes, up to 18.1% reduction in the hardware complexity is achieved (Table II). Needless to say that the architecture presented in [9] is used for the max-log MAP turbo decoder. In order to alleviate the performance loss of using max-log MAP algorithm, the scaled max-log MAP is normally used, which reduces the performance gap between the max log-MAP algorithm and log-MAP algorithm to 0.1 dB while near-optimal performance to log-MAP algorithm provides some

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2407232, IEEE Transactions on Circuits and Systems II: Express Briefs

TABLE II
EQUIVALENT GATE COUNTS (KG) OF THE APPLIED MSR TO APPROXIMATION OF LOG-MAP ACS ARCHITECTURES (0.13 μM CMOS)

	Without MSR version	With MSR Version
Conventional	12.160	10.328 (15%)
[9]	6.192	5.832 (5.8%)
[10]	10.472	8.576 (18.1%)
[12]	7.528	7.096 (5.7%)
r^a = 3 [13]	10.360	9.410 (9.2%)
r^a = 4 [13]	13.144	12.288 (6.5%)
[14]	14.368	11.816 (17.8%)
[15]	14.764	12.084 (18.1%)
[16]	15.508	13.155 (15.2%)
[17]	9.244	8.384 (9.3%)
[18]	9.472	7.880 (16.8%)

[a] The r term denotes the precision of the log-MAP approximation [13].

advantages over scaled max log-MAP detailed in [13]. To this point, many architectures based on precise approximation of log-MAP algorithm are presented in literature. The proposed MSR method also can be applied to these architectures [13]–[18], which proves that their MSR version result in up to 18.1% lower area without any performance degradation. It is worth mentioning that each architecture shown in Table II results in different bit error rate (BER) performances (see [13]).

Moreover, in order to increase the throughput, a higher radix architecture can be used. In [19], a method for computing a higher radix computation is proposed as a two-stage technique, where it is proposed to use two serial radix-$2^{P_T/2}$ stage ACSs instead of one radix-2^{P_T} recursive computation, presenting 55% reduction for $P_T = 4$ in the final silicon area. The MSR method, proposed in this paper, can also be used on top of this two-stage technique providing an additional 15% saving in the silicon area, when the conventional ACS architecture is used. In fact, since the proposed MSR technique is devised for radix-4 ACS architectures, in order to implement a higher radix ACS architecture, it can be divided into few radix-4 ACS blocks. For instance, a radix-16 ACS architecture, i.e., P_T = 4, may be implemented using two radix-4 architectures. As for the other values of P_T, e.g., $P_T = 3$, the radix-$2^2 \times 2^1$ architecture may be used where the MSR is applied to the radix-2^2 stage for area reduction purposes. In [20], the higher radix ACS architecture is constructed based on basic architecture referred to as maximum-value generator-2 (MVG-2). The MSR method can be also applied to MVG-2 architecture resulting in lower silicon area compared to its original architecture.

VII. CONCLUSION

In this paper, by investigating the relation between the recursion computations, a novel method was proposed, which is called MSR. By applying the proposed method to the previous ACS architectures, an area-efficient architecture for recursive computations was achieved. The proposed architectures achieve at most 18.1% reduction in complexity according to the implementation results, which significantly reduces the complexity of the whole MAP core of the turbo decoder. Furthermore, the proposed method can also be used for higher radix designs to reduce complexity.

REFERENCES

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.
[2] S. Belfanti, C. Roth, M. Gautschi, C. Benkeser, and Q. Huang, "A 1Gbps LTE-advanced turbo-decoder ASIC in 65nm CMOS," in *VLSI Circuits (VLSIC), 2013 Symposium on*, June 2013, pp. C284–C285.
[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *Information Theory, IEEE Transactions on*, vol. 20, no. 2, pp. 284–287, Mar 1974.
[4] V. Franz and J. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 2, pp. 186–195, Feb 1998.
[5] J. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," *Vehicular Technology, IEEE Transactions on*, vol. 49, no. 6, pp. 2208–2233, Nov 2000.
[6] C. Studer, "Iterative MIMO Decoding: Algorithms and VLSI Implementation Aspects," Ph.D. dissertation, ETH Zurich, Switzerland Ph.D. dissertation, June 2009.
[7] L. Li, R. Maunder, B. Al-Hashimi, and L. Hanzo, "A Low-Complexity Turbo Decoder Architecture for Energy-Efficient Wireless Sensor Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 14–22, 2013.
[8] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: breaking the ACS-bottleneck," *Communications, IEEE Transactions on*, vol. 37, no. 8, pp. 785–790, Aug 1989.
[9] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 8–17, 2011.
[10] Z. Wang, "High-Speed Recursion Architectures for MAP-Based Turbo Decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 470–474, 2007.
[11] C. Studer, S. Fateh, C. Benkeser, and Q. Huang, "Implementation Trade-Offs of Soft-Input Soft-Output MAP Decoders for Convolutional Codes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 11, pp. 2774–2783, Nov 2012.
[12] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, "A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless," in *2003. Digest of Technical Papers. ISSCC. 2003 IEEE International Solid-State Circuits Conference*, 2003, pp. 150–484 vol.1.
[13] S. Papaharalabos, P. Mathiopoulos, G. Masera, and M. Martina, "On optimal and near-optimal turbo decoding using generalized max operator," *IEEE Communications Letters*, vol. 13, no. 7, pp. 522–524, July 2009.
[14] J.-F. Cheng and T. Ottosson, "Linearly approximated log-MAP algorithms for turbo decoding," in *in Proc. IEEE Vehic. Techn. Conf. (VTC), Spring, Tokyo, Japan*, vol. 3, May 2000, pp. 2252–2256 vol.3.
[15] S. Talakoub, L. Sabeti, B. Shahrrava, and M. Ahmadi, "An Improved Max-Log-MAP Algorithm for Turbo Decoding and Turbo Equalization," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 3, pp. 1058–1063, June 2007.
[16] B. Classon, K. Blankenship, and V. Desai, "Channel codng for 4G systems with adaptive modulation and codng," *IEEE Wireless Communications*, vol. 9, no. 2, pp. 8–13, April 2002.
[17] W. Gross and P. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electronics Letters*, vol. 34, no. 16, pp. 1577–1578, Aug 1998.
[18] H. Wang, H. Yang, and D. Yang, "Improved Log-MAP decoding algorithm for turbo-like codes," *IEEE Communications Letters*, vol. 10, no. 3, pp. 186–188, Mar. 2006.
[19] C.-C. Wong, M.-W. Lai, C.-C. Lin, H.-C. Chang, and C.-Y. Lee, "Turbo Decoder Using Contention-Free Interleaver and Parallel Architecture," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 422–432, 2010.
[20] M. Martina, S. Papaharalabos, P. Mathiopoulos, and G. Masera, "Simplified Log-MAP Algorithm for Very Low-Complexity Turbo Decoder Hardware Architectures," *Instrumentation and Measurement, IEEE Transactions on*, vol. 63, no. 3, pp. 531–537, March 2014.