A Low Complexity Stopping Criterion for Reducing Power Consumption in Turbo Decoders

Pallavi Reddy[12], Fabien Clermidy[1]
[1]CEA-LETI, MINATEC
Grenoble, France
{pallavi.reddy, fabien.clermidy}@cea.fr

Amer Baghdadi[2], Michel Jezequel[2]
[2]Electronics Department, ENST Bretagne
Brest, France
{amer.baghdadi, michel.jezequel}@enst-bretagne.fr

Abstract— **Turbo codes are proposed in most of the advanced digital communication standards, such as 3GPP-LTE. However, due to its computational complexity, the turbo decoder is one of the most power hungry blocks in digital baseband. To alleviate this issue, one way is to avoid surplus computing phases thanks to the early termination of the iterative decoding process. The use of stopping criteria is one of the most common algorithm level power reduction methods in literature. These methods always come with some hardware overhead. In this paper, a new trellis based stopping criterion is proposed. The novelty of this approach is the lower hardware overhead thanks to the use of trellis states as key parameter to stop the iterative process. Results are showing the importance of this added hardware in terms of method efficiency. Compared to state-of-the-art Log Likelihood Ratio (LLR) based techniques, proposed Low Complexity Trellis Based (LCTB) is demonstrating 23% less power consumption on average, for comparable performance level in terms of Bit Error Rate (BER) and Frame Error Rate (FER).**

Keywords-stopping criteria; turbo-decoder; trellis states; low complexity; power reduction

I. INTRODUCTION

Turbo-decoding is an attractive channel decoding scheme and is widely used in wireless communication. The superior performance of turbo codes [1] comes from the combination of parallel concatenated coding, recursive encoding, pseudorandom interleaving and iterative decoding. However, this performance comes at the cost of an important computational complexity. In the mobile terminals, this complexity is directly related to power consumption. As an example, the decoder of a 3GPP-LTE application in [2] is responsible for 60% of the total baseband power consumption at 50 Mbits/s throughput. Thus, turbo-decoders have to balance the two conflicting requirements of low-energy consumption and high-performance requirements.

A typical turbo-decoder includes two Soft-Input/Soft-Output (SISO) decoders (denoted as DEC1 and DEC2 in Fig. 1) as well as interleaver and de-interleaver units between the decoders and a simple comparator for deducing the hard decision (0 or 1). Using Max-Log-MAP algorithm [4] the SISO decoders makes "soft" information: a soft information is a statement of the probability that each received bit is '1' or '0', also referred as Log Likelihood Ratio (LLR) and it depends on the preceding sequence of bits. The soft information made by DEC1 are interleaved to be included in the input to DEC2. Then, DEC2 makes soft information based on both the received bits and the soft information from DEC1. This soft information are again interleaved and sent as input to DEC1. Consequently, DEC1 has more information to use than it had on the previous iteration and it strives to use the additional information to make a more accurate decision. The iterative decoding process is repeated either a fixed number of times or until some criterion for stopping the iterations is satisfied. Then hard decisions are made on the basis of the soft information.

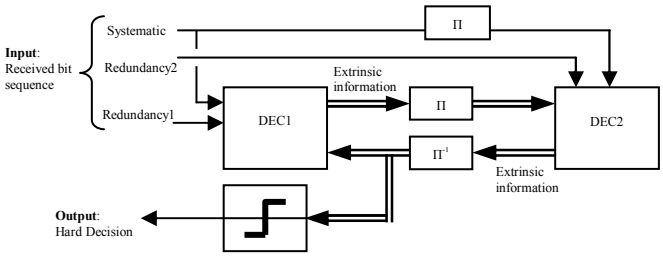

Figure 1. Parameters considered for stopping criteria

Based on this architecture template, solutions to achieve low power implementations can be categorized at two levels, as architecture level and algorithm level.

At the architecture level, some modifications of the architecture are done in order to reduce the energy per operation [17]. Examples of such solutions are low power SISO architecture [18], memory arrangements [16], pipeline units [19] and architectures realized thanks to hardware-software co-design [20]. Considering their respective importance, most of the efforts are done on, the computing unit and the storage. By modifying the computing part, throughput can be increased and hence the energy spent per decoded bit is reduced. By controlling the number and sizes of the memories used for storage, the power consumption due the memories can be controlled.

Even if architecture-level optimizations can lead to important power consumption reductions, algorithm-level ones have more important possible gains [3, 4]. They consist of changing the algorithm itself such as they have direct impact on the computations and storage. Very little modifications of the algorithm could achieve considerable savings in the

978-3-9810801-7-9/DATE11/©2011 EDAA

power/energy. The drawback is to ensure a minimum effect on the performance as well as the hardware overhead. These two points must be analyzed while implementing algorithm level techniques.

For turbo-decoder, one of the well known algorithm level power reduction method is iteration control: indeed impact of the number of iterations is huge on the power consumption, as the computations and storage is directly multiplied by the number of iterations. Such control is called in the literature, as *'stopping criteria'* or *'early termination'*. The principle is to stop the iterative process when the data is proved to be decoded correctly. Hence, in order to decide the stopping point, many parameters have been considered in the state of the art and are discussed in Section II. Section III explains rational of the approach for proposed stopping criteria. Section IV explains the proposed method and its implementation details. Finally, section V presents the methodology and environment for extracting results, followed by the results in terms of BER, FER, number of iterations and power consumption.

II. STATE OF THE ART

As explained in the introduction, turbo decoding is an iterative process. A special criterion, called stopping criterion can be used to determine the convergence of the decoding, and at the iteration i where the decoding process can be terminated. The ideal number of iterations can be found thanks to the *Magic Genie Rule* [5]. This rule uses foreknowledge of the transmitted bit sequence: the decoded information is compared with the original bit sequence allowing the iterative process to stop at the exact minimum number of iterations. Unfortunately, this method cannot be applied at run-time as it needs foreknowledge of transmitted bit sequence.

The approximation of the exact number of iterations allows a large saving of power, especially for medium and high Signal to Noise Ratio (SNR), without degrading the performances of the turbo-decoding. For practical implementations, stopping criteria are based on different parameters and can be classified into two categories (i) hard decision rules and (ii) soft information rules.

A. Hard Decision Rules

In turbo-decoders, the hard decision is the one bit decision made using the soft information; the hard decision output gives either 1 or 0 depending on the soft information value. There are many stopping criteria employed based on hard decision. For example Hard Decision Aided (HDA) [5] rule stops the iterative process if hard decisions of a decoder for two consecutive iterations are matched. The hardware required for computing this criterion is N compares and N storage elements. Here N depends on the turbo decoding standard and varies from 40 to 5114 bits [21]. In order to reduce the storage of HDA, Improved HDA (IHDA) was proposed [6]. IHDA decides on stopping the iterations by matching the hard decisions of the two decoders in the same iteration. As the comparison to check the stopping point is done in the same iteration, there is no need of the storage of the hard decision of the previous iteration. Hence, with this method the storage overhead is eliminated compared to HDA. Therefore, the total hardware overhead is reduced to one 10 bit adder and a single bit comparator.

The drawback of classical HDA is its inefficiency at low SNR channels, so in order to solve this problem, a combined method is proposed in [7]. In this method, if the hard decisions between two iterations are all the same then the iterations are stopped; if not then Hard Decision Decrease (HDD) and Difference of HDD (DHDD) is computed. A threshold on the DHDD value is used to stop the iterations. Here hardware overhead is in terms of one N bit substractor, 1 bit comparator and N storage elements.

Hard decision is single bit information, and when it is considered for stopping criteria, the hardware complexity is low. But when the hard decision rules are modified for improving performance, then hardware overhead is added with improved performance. On the other hand, the soft information based rules offer more control over the tradeoffs between undetected and falsely detected errors [5], by setting the proper threshold for detection. Soft information rules provide better control over the iterative process. Hence low complexity soft information rules are discussed in the following text.

B. Soft Information Rules

1) Basic - Saturation of soft information of the both decoders is observed [11]. Overhead is in terms of N storage elements and N compares**.** To avoid the storage here, the soft information can be used for stopping iterations by comparing it with some high and low thresholds [5]; so the comparison is done on the fly and no need of storage. Overhead for this method is N compares.

One more way of iteration control is by observing mean reliability [11]. When mean reliability do not change in two subsequent iterations, it means decoding is considered to be converged [12, 13] so mean reliability can be used as stopping parameter. Output soft information varies in large range; hence mean value differs considerably for different information blocks. Therefore, rather than comparing the average or minimum soft output value, a priory values are used [14].

2) Sign Base - The sign of soft information is used as a parameter for iteration control. Some of these methods are Sign Change Ratio (SCR)[8], Sign Difference Ratio [9], and Mean Sign Change (MSC) [13]. The sign changes between two iterations or two decoders are considered to stop the iterations. For these methods, the overhead is in terms of N additions, N compares, a counter, N storage elements(only if sign is observe for consecutive iteration).

3) Others - There are some method which combines the soft information and hard decision rules. A criterion proposed in [15], counts the number of absolute values less than a threshold and the number of hard decision one's to stop the iterations. Here, the hardware overhead is in terms of less than N compares, and 2 Counters.

Soft information rules have better control over iterative process while keeping comparable performance. Therefore, in this work, for the exploration of the parameter for stopping the iterations, soft information is considered and explored.

III. RATIONAL OF THE APPROACH

Different parameters are used for stopping the iterative process; the major issues on choosing the right parameter are performance and hardware overhead. In most of the low complexity approaches the soft inputs and outputs of the decoder are used. In order to keep the minimum hardware overhead, while defining a stopping criterion, the size of parameter and storage requirement needs to be taken into account. It is well known fact that the memory is critical issue in turbo decoding when designing power efficient architecture. Motivation behind this work is to keep the storage needs to minimum by choosing a parameter of smallest size while keeping comparable performance.

The parameters explored in state of the art works are the soft input or output of the decoder. To be able to choose proper parameter for stopping criteria, here we go in the details of the Max Log MAP algorithm [4] employed by the decoders (Fig. 1). The (4), is a reminder of Max Log MAP algorithm. This algorithm decides whether information bit u_k=0 or 1 depending on $L(u_k)$ or LLR (or extrinsic value (Fig. 1)). The term α_k (s) is forward state metrics (Fig. 2). It is the probability of arriving at a branch in a particular state s. By summing over all branches from state s' to s, we get a forward recursion for calculating $\alpha_k(s)$ in terms of the values of γ_k(s', s) as in (5). The term $\beta_{k-1}(s')$ is backward state metrics (Fig. 2). It is the probability of exiting a branch via state s'. By summing over all branches exiting from state s' to s, we get a backward recursion for calculating $\beta_{k-1}(s')$ in terms of the values of $\gamma_k(s',s)$ as in (6).

$$L(u_k) = \max_{U1}[\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)] - \max_{U0}[\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)] \quad (4)$$

$$\alpha_k(s) = \max_{s'}(\alpha_{k-1}(s') + \gamma_k(s', s)) \quad (5)$$

$$\beta_{k-1}(s') = \max_{s'}(\beta_k(s) + \gamma_{k+1}(s', s)) \quad (6)$$

Here, k is index of the symbols {1,2,…N} and N is block size.

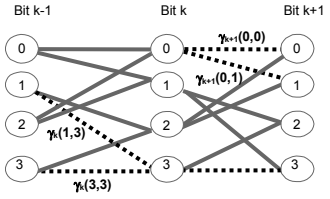

Figure 2. State metrics

To explore further dependencies of α and β, we compute values of α and β for symbols k=0 to k=N. As demonstrated in (7) to (12), with boundary conditions, $\alpha_0(0) = 0$ and $\alpha_0(s)$ = -infinity when s≠0, and $\beta_N(0) = 0$ and , $\beta_N(s)$ = -infinity when s≠0. α and βvalues are depending on only the γ values.

$$\alpha_1(s) = \max_{s'}(0 + \gamma_1(s', s)) \quad (7)$$

$$\alpha_2(s) = \max_{s'}(\gamma_0(s', s) + \gamma_2(s', s)) \quad (8)$$

$$\alpha_N(s) = \max_{s'}(\gamma_1(s', s) + \gamma_2(s', s) + \ldots + \gamma_N(s', s)) \quad (9)$$

$$\beta_{N-1}(s') = \max_{s'}(0 + \gamma_N(s', s)) \quad (10)$$

$$\beta_0(s') = \max_{s'}(\gamma_1(s', s) + \ldots\ldots + \gamma_N(s', s)) \quad (11)$$

Values of γ are predefined as per (12).

$$\gamma(s',s) = \tfrac{1}{2} x_k^s (L_e(u_k) + L_c y_k^s) + (L_c/2) x_k^p y_k^p \quad (12)$$

(12) is derived assuming that the code is transmitted through an Additive White Gaussian Noise (AWGN) channel, with noise variance of σ^2. Here x_k and y_k represent the transmitted codeword associated to this transition and codeword received from the channel. Superscript p & s denote parity and systematic bits. $L_e(u_k)$ is the extrinsic information received from the other decoder and L_c is $2/\sigma^2$. For i^{th} iteration and k^{th} symbol the only parameter changing in (12) is x_k, which depends on the transition and hence depends on the states s' and s. Thus γ values are depending on the trellis states s' and s, and therefore α, β and extrinsic information are depending on the trellis states.

With above demonstration, this work begins with the assumption that the values of states arrive to saturation when data is decoded correctly. Then detecting this saturation would enable the stopping of the iterations. In order to explore this possibility, saturation of the state values and extrinsic information are observed in order to compare the saturation point. In order to observe saturation of states and extrinsic values, software model for turbo decoder defined in C++ is used. Simulations for the single binary turbo decoding at ½ code rate, block size 1440 and with AWGN channel, have been done. Values of states and extrinsic information have been observed for the 20^{th} incoming frame at 0.9db. As shown in Fig. 3(a) for the first 10 symbols of the 20^{th} frame, the values of states saturate after 4^{th} iteration. This saturation is stable and with no variation after saturation. The same is observed for the extrinsic information in Fig. 3(b). Thus, observing the trellis states also allows stopping the iterative process at right point. In this case the saturation is detected at 5^{th} iteration and the iterative process is stopped.

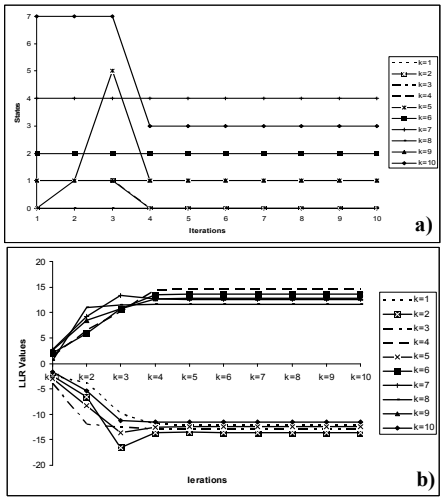

Figure 3. Saturation detection using a) Trellis States, and b) LLR

After having rational reasoning for precision for the use of states in stopping criteria, now let's consider the benefits of

using states as a parameter for iteration control. As per (4), when the soft input/output information is used, the size of the information is 10 bits, which may vary from design to design and as per quantization used. While the states are defined either with 2 bits or 3 bits, it depends on the number of states in the decoding standard. We consider here the 8 state single binary turbo decoding, so the eight states are represented with 3 bits. When the trellis states are used as a parameter to stop the iterative process, then the size of parameter reduces to 3 bits when compared with the extrinsic information. The new trellis states based stopping rule is explained theoretically followed with its implementation details in next sections.

IV. TRELLIS-BASED METHODS

The extrinsic information, as defined in (4), is function of the α, β and γ parameters. As presented in section III, all these values are depending on states s' and s. For each symbol k, there is a pair of s' and s which obtains "$\max_{U1}[\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)]$" for iteration i. Let's call these states $(s_{U1'max})_k^{(i)}$ and $(s_{U1max})_k^{(i)}$

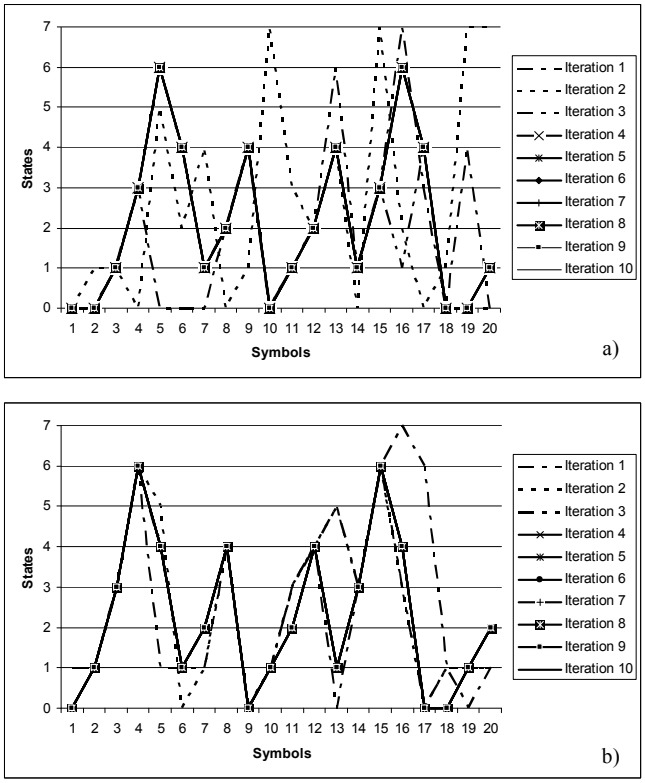

Figure 4. State $s_{U1'max}$ (a) & s_{U1max} (b) values on different iterations @ 0.9 dB

Fig. 4 shows a simulated example for the values of the two states for 10 successive iterations of the turbo-decoder. After the 4th iteration, the successive values of both s_{U1max} and $s_{U1'max}$ are similar, which is highlighted by the continuous line. As a conclusion, Fig. 4 indicates that at 0.9db it takes 4 iterations to decode the data. Based on this observation of the values of states, the Trellis Based (TB) stopping criterion is defined in (13).

$$(s_{U1'max})_k^{(i)} == (s_{U1'max})_k^{(i-1)} \quad for\ k=\{1,2,....,N\} \qquad (13)$$

As shown in (13), TB compares the state $s_{U1'max}$ at iteration i for symbol k with the $s_{U1'max}$ for the same symbol k at the previous iteration (i-1). If, for all the symbols, the values of $s_{U1'max}$ are matched for two consecutive iterations, then the iterative process is stopped. In order to observe the values of states at each iteration and count its repetition, the hardware needed is listed in Table I. Here the hardware overhead is in terms of memory, registers, comparators and counters. Three times Nbits memories are needed for the storage of all the states during one iteration.

As a result, the TB approach is having a potential large overhead of memory depending on the final applications. Further efforts are necessary to limit this overhead. Figure 4 gives another solution to simplify the method: as the decoding converges, the state $s_{U1'max}$ of the current symbol repeats its value in state s_{U1max} of the previous symbol for the current iteration i. Using this observation, the Low-Complexity TB (LCTB) method is defined with (14).

$$(s_{U1'max})_k^{(i)} == (s_{U1max})_{k-1}^{(i)} \quad for\ k=\{2,....,N-1\} \qquad (14)$$

In LCTB, iterative process is stopped when (14) stands true for all the symbols except the first and last ones, where the comparisons with previous state or next state cannot be done.

The advantage of LCTB is the use of states $s_{U1'max}$ and s_{U1max} for the same iteration. Therefore the comparison can be done on the fly and thus there is no need of storage and the memory is eliminated.

TABLE I. COMPLEXITY OF STOPPING CRITERIA

	LLRTC[13]	**LLRC[5]**	**TB**	**LCTB**
Memory (bits)	-	(Nx10)	(Nx3)	-
Registers (bits)	(Nx10)x2	(Nx10)	(Nx3)	(Nx3)
Comparator (bits)	10	10	3	3
Counter (bits)	logN/0.3	logN/0.3	Log(N)/0.3	Log(N)/0.3

N varies from 40 to 5114 bits [21]

V. IMPLEMENTATION AND RESULTS

After defining theoretically the TB and LCTB stopping criteria, the next step is to implement these methods and obtain results for performance and power. Here performance is in terms of number of iterations saved, BER and FER.

A. Software Model

In order to evaluate the performance, the stopping criteria have been implemented in C++. This software model is simulating a single binary turbo decoder at ½ code rate with 8 states and with AWGN channel. For comparison, the extrinsic information based stopping criteria, fixed number of iterations and magic genie rule are also simulated. LLRC (log likelihood ratio compare)[5] and LLRTC(LLR threshold compare)[13] are the two extrinsic information based methods under consideration for comparison. These two methods are chosen for comparison with TB and LCTB, in order to prove what we

have stated earlier about the dependency of extrinsic information over the trellis states and the savings in terms of hardware overhead. The hardware needs for all the methods under consideration are listed in table 1.

Results of the software model in terms of BER & FER are shown in Fig. 5. BER stays same for all the methods but FER gets deflected after 1db for LLRTC, TB and LCTB. The savings in number of iterations is shown in Fig. 6. Here, as obviously, the maximum iterations are taken by the fixed number of iteration (fixed limit to 10), while the least number of iterations are taken by Magic Genie (MG) rule. LCTB is closer to magic genie, while TB is taking slightly more number of iteration compared to LCTB. LLRTC is taking 7% more iterations than TB. While LLRC is saving 38% iterations compared to fixed number of iterations, even though it stands the worst solution in considered methods

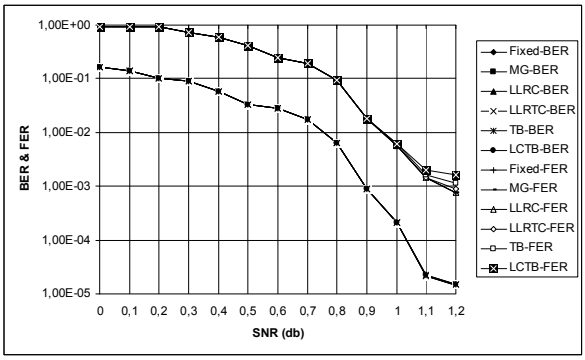

Figure 5. BER and FER for different stopping criteria (Fixed number of iteration (Fixed), Magic Genie (MG), LLR compare (LLRC), LLR Threshold Compare (LLTC), Trellis Based (TB), Low Complexity TB (LCTB))

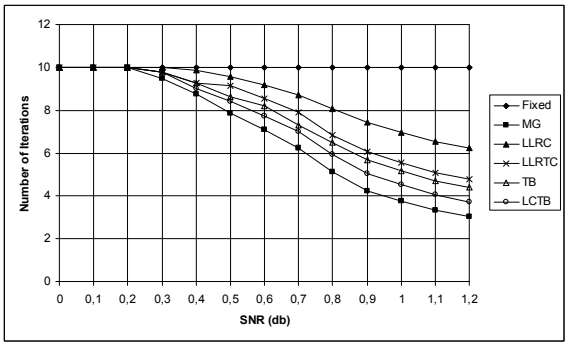

Figure 6. Number of iteratios for different stopping criteria (Fixed number of iteration (Fixed), Magic Genie (MG), LLR compare (LLRC), LLR Threshold Compare (LLTC), Trellis Based (TB), Low Complexity TB (LCTB))

B. Power Consumption Analysis

In order to evaluate the impact on power due to the hardware overhead, power consumption analysis of the each of the stopping criteria has been done. Hardware module simulating the stopping criteria for worst case (N=5114) has been developed in VHDL.

Block diagram, for the all considered stop criteria, is shown in Fig. 7(a). Here the all blocks are defining the hardware blocks for LLRC and TB. While the 'hashed' parts are representing the hardware blocks for LLRTC and LCTB, In case of LLRTC and LCTB the input for Reg 1 and Reg 2 are coming directly from decoder, as memory does not exist for these methods. Combinational Logic is running the common routine for checking the counter value and making decision for stopping the iteration, which is represented here in the form of a flowchart.

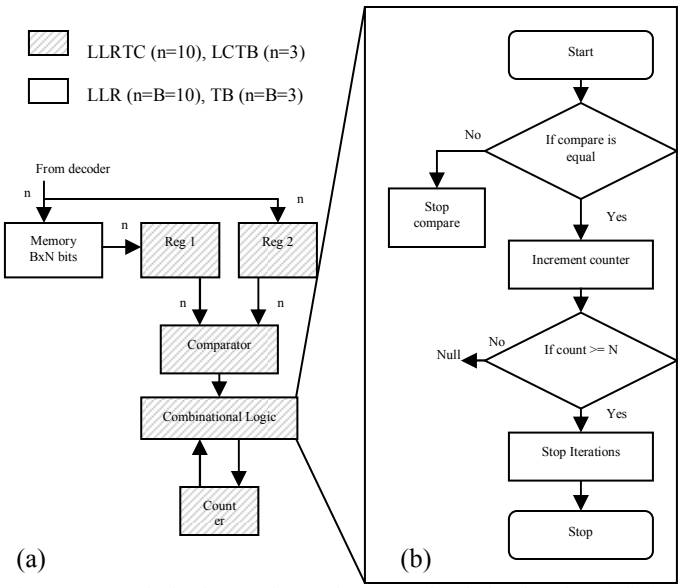

Figure 7. (a) Block Diagram for Hardware Overhead, (b) Flowchart defining Combinational Logic

TABLE II. POWER AND AREA OVERHEAD FOR STOPPING CRITERIA

		LLRC	LLRTC	TB	LCTB
Power (mW)	**Memory + Address Generator + Buffer**	7.02	-	2.34	-
	Registers + Comparator + Counter	0.04	0.04	0.02	0.02
	Total Power Overhead (per iteration)	7.06	0.04	2.36	0.02
Area (mm^2)	**Memory + Address Generator + Buffer**	0.0676	-	0.0309	-
	Registers + Comparator + Counter	0.0037	0.0037	0.0037	0.0037
	Total Area Overhead	0.0713	0.0037	0.0346	0.0037

The RTL model for stopping criteria has been *Synopsys* topographic design compiler, using low power technologies (65nm, 1V,25°C). The synthesis has been done at 500MHz clock. Power analysis of the hardware overhead has been one following the same power analysis flow as demonstrated in [16]. Post-synthesis power and area results are listed in table 2. As obvious, the major overhead is due to memory. Here interesting point to observe is the real power savings after use of the stopping criteria, i.e. total power savings considering the hardware overhead & iteration saved for each stopping criterion. In order to get the total power, the power consumption of an ASIP [16] is considered as the basic power needed for the turbo decoding per iteration. Then, the total power is computed as per (15).

$$P_{tot} = (P_{ASIP} + P_O) \times I \quad (15)$$

Here, P_{tot} – Total power,

P_{ASIP} – Power consumption by ASIP [16] per iteration

P_O – Power due to hardware overhead

I – Average number of iterations for stopping criteria

In Fig. 8, the total power, for all stopping criteria considered, is shown. With this figure, it is clear that, only observing number of iterations saved by stopping criterion is not enough to decide its efficiency. Power savings is the basic goal of the stopping criteria, so it is important to see the effect of stopping criteria in terms of total power savings.

In LLRC and TB, the memory overhead is causing significant power consumption, and this is visible in Fig. 8. For LLRC the total power till 0.9db is more than in fixed number of iterations. While for TB the total power is more than fixed iteration till 0.4db. On the contrary, the methods including least hardware like LLRTC and LCTB are consumption always below the fixed number of iterations, which is the basic goal of stopping criteria.

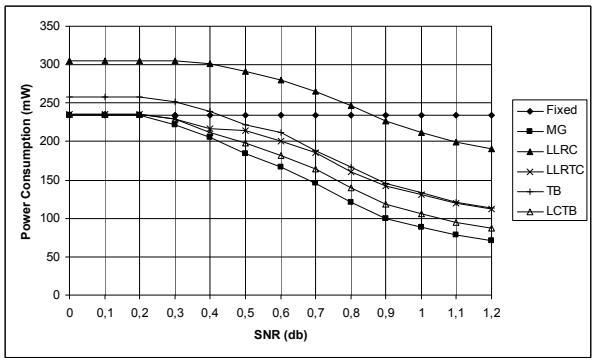

Figure 8. Power consumption of stopping criteria

VI. CONCLUSION

Considering the issue of large power consumption in turbo decoding, two new stopping criteria, TB & LCTB, have been proposed. The advantage of this approach is the lower hardware overhead thanks to the use of trellis states as key parameter to stop the iterative process. The results are showing a negligible impact on performance compared to the ideal *magic genie* rule. Considering 1db SNR. Compared to state-of-the-art LLR based techniques, the proposed LCTB is demonstrating 23% less power consumption on average.

This paper also proves that we must consider the hardware overhead due to the stopping criterion implementation: the power saved by the stopping criterion can vary considerably when taking into account this overhead. Consequently, the usage of large memories for saving intermediate values must be avoided when deciding of a stopping criterion.

ACKNOWLEDGMENT

This work was supported in part by the UDEC project of the French Research Agency (ANR).

REFERENCES

[1] Berrou, C.; Glavieux, A.; Thitimajshima, P.; 'Near Shannon limit error-correcting coding and decoding,' Technical Program, Conference Record, IEEE International Conference on Communications, p 1064 - 1070 vol.2, May 1993.

[2] Clermidy, F. Bernard, C. Lemaire, R. Martin, J. Miro-Panades, I. Thonnart, Y. Vivet, P. Wehn, N.; 'A 477mW NoC-Based Digital Baseband for MIMO 4G SDR', IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC),p 278, Feb. 2010.

[3] Kaza, J. Chakrabarti, C. 'Design and Implementation of Low-Energy Turbo Decoders', IEEE Transactions on VLSI Systems, Vol 12, No 9, 2004.

[4] Hye-Mi Choi Ji-Hoon Kim In-Cheol Park, 'Low Power Hybrid Turbo Decoding Based in Reverse Calculation' IEEE International Symposium on Circuits and Systems Proceedings. Pp. 4, 2006.

[5] A. Matache, S Dollinar, and F. Pollara; 'Stopping Rules for Turbo Decoders', TMO preogress report, August 2000

[6] Wu, Y.; Woerner, B.D.; Ebel, W.J.; 'A Simple Stopping Criterion for Turbo Decoding', IEEE Communications Letters, VOL. 35, p 701-702, Oct 2001

[7] Wen-Ta Lee; San-Ho Lin; Chia-Chun Tsai; Trong-Yen Lee; Yuh-Shyan Hwang; 'A New Low Power Turbo Decoder Using HDA-DHDD Stopping Iteration', IEEE International Symposium on Circuits and Systems, 1040 - 1043 Vol. 2, 2005

[8] Shao, R.Y.; Shu Lin; Fossorier, M.P.C.; 'Two Simple Stopping Criteria for Turbo Decoding', IEEE Transactions on communications, vol. 47, No. 8, Aug 1999.

[9] Wu, Y.; Woerner, B.D.; Ebel, W.J.; 'A Simple Stopping Criterion for Turbo Decoding', IEEE communications letters, vol. 4, no. 8, Aug 2000.

[10] Rovini, M.; Martinez, A.; 'Efficient Stopping rule for Turbo Decoders', Electronics Letters,Vol 42, No.4, Feb 2006

[11] Zhai, F.; Fair, I.J.; 'New Error Detection Techniques and Stopping Criteria for Turbo Decoding', Canadian Conference on Electrical and Computer Engineering,p 58 - 62 vol.1, 2000

[12] Land, I.; Hoeher, P.A.; 'Using the mean reliability as a design and stopping criterion for turbo codes', Information Theory Workshop, 2001. Proceedings. 2001 IEEE, p 27 – 29, Sept 2001

[13] Fengqin Zhai; Fair, I.J.;'Techniques for Early Stopping and Error Detection in Turbo Decoding', IEEE Transactions on communications, vol. 51, No. 10, October 2003.

[14] Bokolamulla, D.; Aulin, T.; 'A New Stopping Criterion for Iterative Decoding', IEEE International Conference on Communications,Vol 1, 2004

[15] Dong-Soo LEE, In-Cheol PARK; 'A Low Complexity Stopping Criteria for Iterative Turbo Decoding', IEICE Trans Commun. Vol E88-B, No. 1, Jan 2005

[16] Reddy, P.; Clermidy, F.; Al Khayat, R.; Baghdadi, A.; Jezequel, M.; 'Power consumption analysis and energy efficient optimization for turbo decoder implementation', International Symposium on System on Chip (SoC), p 12-17, 2010

[17] Masera, G.; Mazza, M.; Piccinini, G.; Viglione, F.; Zamboni, M.; 'Architectural strategies for low-power VLSI turbo decoders', IEEE Trans. on VLSI Syst., vol. 10, pp. 279 - 285, 2002

[18] Seok-Jun Lee; Shanbhag, N.R.; Singer, A.C.; 'A low-power VLSI architecture for Turbo decoding', IEEE Proceedings of ISLPED '03, p 366 – 371, 2003.

[19] Vogt, T.; Wehn, N.; 'A Reconfigurable Application Specific Instruction Set Processor for Convolutional and Turbo Decoding in a SDR Environment', In Proc. IEEE DATE '08, p 38-43, March, 2008.

[20] Glokler, T.; Meyr, H.; 'Power reduction for ASIPs: A case study', Proc. Workshop Signal Process. Systems (SIPS), pp. 235 - 246, 2001.

[21] European Telecommunication Standards Institute, Universal Mobile Telecom- munications System (UMTS): Multiplexing and channel coding (TDD),3GPP TS 25.222 version 7.3.0 Release 7, p. 18 - 23, May 2005.