Turbo Decoder VLSI Architecture with Non-Recursive max* Operator for 3GPP LTE Standard

Ashfaq Ahmed, Maurizio Martina, Guido Masera
Department of Electronics & Telecommunication
Politecnico di Torino
Torino, Italy
Email: {ashfaq.ahmed, maurizio.martina, guido.masera}@polito.it

***Abstract*—This paper presents a highly parallel turbo decoder architecture for 3GPP LTE standard with a new non-recursive max* operator. High parallelism is introduced at several levels to achieve high throughput, to meet LTE requirements. The decoder supports all codes specified by LTE and features low complexity, obtained by using the new non-recursive max* operator. The decoder achieves a maximum throughput of 376.4 Mbps at 250 MHz, occupying an area of 1.62 mm^2 on 90-nm Standard Cell ASIC technology . The decoder shows better decoding efficiency (Bits/Cycle/Iterations) and throughput to area ratio (Throughput/mm^2) than many of the previously implemented decoders.**

***Keywords*—3GPP LTE; iterative decoding; parallel turbo decoder; VLSI architecture**

I. INTRODUCTION

Turbo codes [1] have gained huge attention in the last twenty years. Indeed Turbo codes achieve Near Shannon limit capacity, excellent error correction performance, intrinsic parallelism and high coding gain which made them an eligible candidate for a large number of wireless communication standards e.g., WiMax [2], 3GPP LTE [3], UMTS-LTE [4], CCSDS [5]. LTE (Long Term Evolution) is the next step forward in cellular 3G and 4G services. It is designed to meet carrier needs for high speed data and media transport as well as high capacity voice support. The standard specifies data rates up to 100 Mbps in the down-link (two MIMO channels), 50 Mbps in the up-link (single channel) and 20 MHz bandwidth channel.

VLSI implementation of turbo decoders is a challenging task because of high throughput constraints of the standards. A lot of research has been carried out to implement efficient decoders satisfying area, speed and power constraints [6][7][8]. Typically two SISO (Soft Input Soft Output) processors are concatenated in parallel in order to achieve high throughput. Parallelism can be introduced in the decoding process at several levels to improve decoding speed [9] e.g., using multiple processors, exploiting the trellis representation parallelism to decode a set of bits, internal parallelization by using multiple windows in each processing element [10], which eventually reduces the latency. However, highly parallel architectures introduce high complexity in terms of hardware and logic. In the proposed decoder architecture, several of previously mentioned methods have been adopted to improve throughput, along with a novel implementation of non-recursive max* operator [11]. In the non-recursive max* implementation the complexity and overhead of the operator is reduced by computing the *n*-input max* as a whole instead of recursively applying the 2-input max* operation. In the low complexity max* operator, the two maximums are calculated and using the two outputs, the final maximum is found.

The remainder of the paper is organized as follows. Section II describes the code adopted in LTE. In this section the explanation of BCJR algorithm, that is the core of turbo decoding, is given. Section III details the proposed architecture including the overview of the main modules, focusing on the novelty in the architecture i.e., the non-recursive max*. The implementation results along with the comparisons are shown in Section IV. Finally, the whole architecture is concluded in Section V.

II. LTE TURBO CODES AND DECODING ALGORITHM

The 3GPP LTE turbo codes are based on the parallel concatenation of two 8-state Convolutional Codes (CCs) and one Quadratic Polynomial Permutation (QPP) interleaver [12]. The constituent code used in 3GPP LTE is a single binary systematic CC. The generated input frame is fed into the convolutional encoder of rate 1/3. An information sequence $\underline{u}$ is encoded by a convolutional encoder, and an interleaved version of u is encoded by a second convolutional encoder. The initial values of the shift registers of the 8-state constituent encoders are all zeros. After encoding of the current frame, termination is performed: during the last three cycles of the frame, values are input into the encoder to force it to the zero state. The total number of output bits generated by the encoder is $L = 3 \cdot K + 12$, where $40 \leq K \leq 6144$ is the frame length. The last 12 bits are the trellis termination bits.

A. Decoder

The main processing elements inside each decoder are the Soft-In–Soft-Out (SISO) modules [13] which executes the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [14], usually in its logarithmic form [15]. The SISO module processes the intrinsic log-likelihood ratios (LLRs), received from the channel, of a coded symbol c and calculates the extrinsic information for the information bits u. The LLRs are exchanged by the SISO modules through the interleaving/deinterleaving permutation laws Π /Π^{-1}. The output extrinsic LLR of symbol u at the kth step is computed as

 ISBN: 978-1-61208-264-6

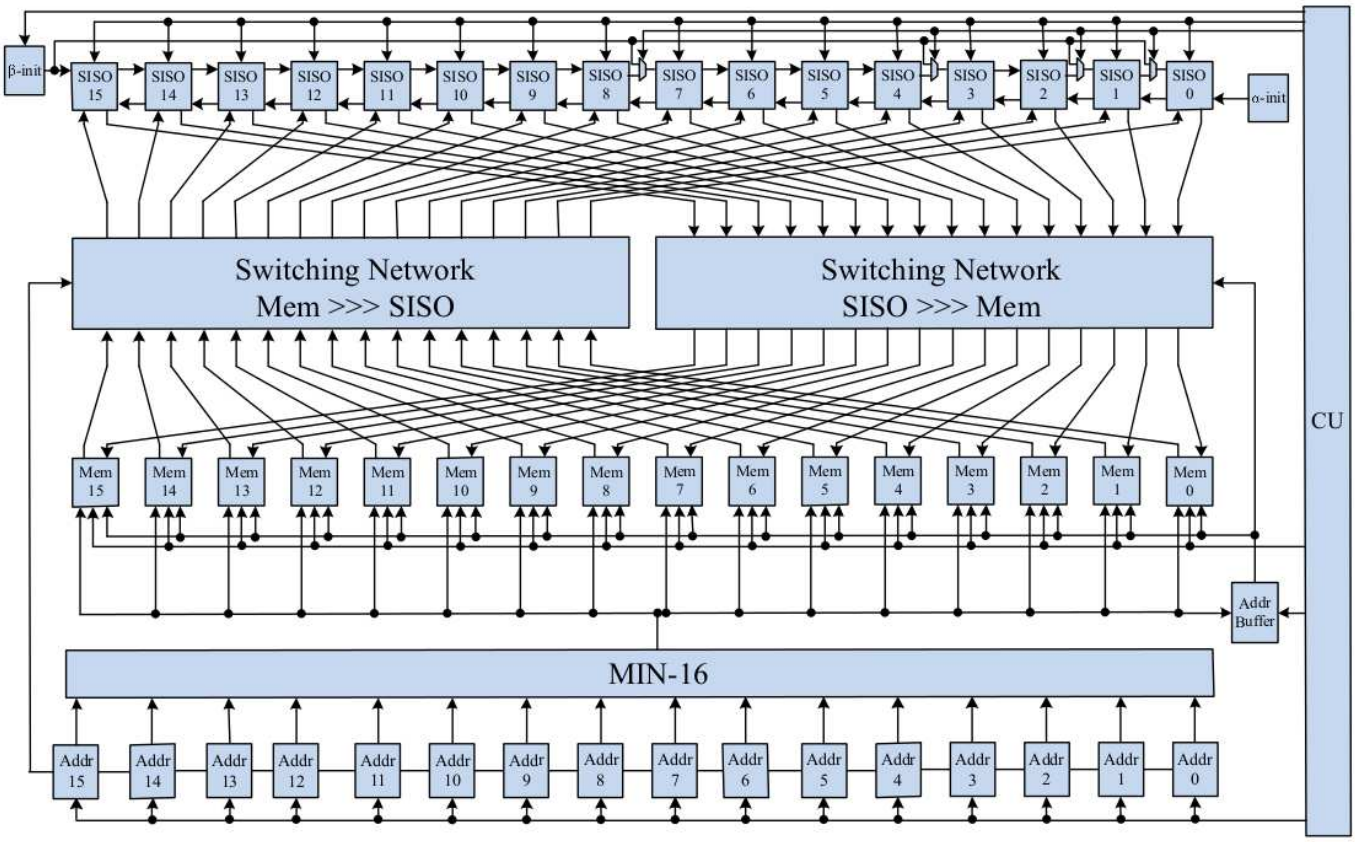


Figure 1. Complete Turbo Decoder

$$\lambda_k^{ext} = \delta \left(\max_{R_1}^{*} \{b\} - \max_{R_0}^{*} \{b\} - \lambda_k^{apr} \right) \quad (1)$$

where $\delta = 0.75$ is a scaling factor [16]. The trellis transition for input equal to '0' is represented by R_o, whereas the trellis transition due to input equal to '1' is represented by R_1. The $\max_{R_x}^{*}$ finds transitions with maximum probability w.r.t. input equal to x, where $x \in \{1,0\}$. λ_k^{apr} is the a-priori information received from the other SISO module, whereas λ_k^{ext} is the extrinsic information generated by the SISO at trellis step k and it is transferred to the other SISO module by means of the interleaver memory. b is defined as,

$$b(k) = \alpha_{k-1}(s) + \gamma_k \left(s, s^{'} \right) + \beta_{k-1} \left(s^{'} \right) \quad (2)$$

$$\alpha_k \left(s^{'} \right) = \max_{s}^{*} \left(\alpha_{k-1}(s) + \gamma_k \left(s, s^{'} \right) \right) \quad (3)$$

$$\beta_{k-1}(s) = \max_{s}^{*} \left(\beta_k \left(s^{'} \right) + \gamma_k \left(s, s^{'} \right) \right) \quad (4)$$

$$\gamma_k \left(s, s' \right) = y_k^s + \lambda_k^{apr} + y_k^p \quad (5)$$

where α and β are known as forward and backward state metrics and are calculated during forward and backward recursions, while γ is the branch metric and is calculated for each transition. SISO takes the following steps to calculate extrinsics and the output bits:

1) It calculates γ for the trellis section k using the systematic and parity channel LLRs and the a-priori information coming from the other SISO, as in (5).
2) From the calculated γ, it computes α by adding the previous α to the corresponding γ. So (3) can be expanded as,

$$\alpha_k = \max^{*} \left(\alpha_{k-1}^1 + \gamma_1, \alpha_{k-1}^2 + \gamma_2 \right) \quad (6)$$

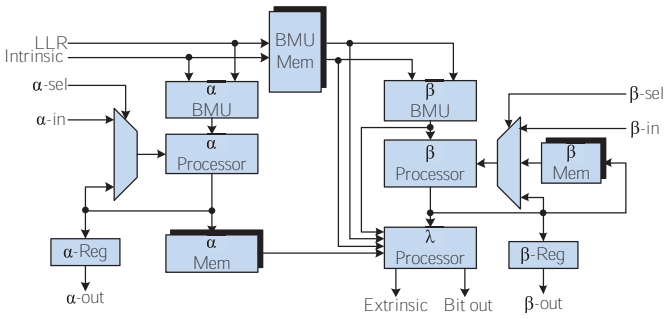
3) β is calculated in the same way like α, but in this case the recursion is in reverse direction (see (4)).
4) After calculating α, β and γ, the final step is to calculate the extrinsic information and the decoded bit value, as in (1) and (2). The calculated extrinsics are passed to the other SISO by means of interleaver memory, while the decoded bit is stored in the decoded bits memory after a certain number of iterations.

III. Proposed Architecture

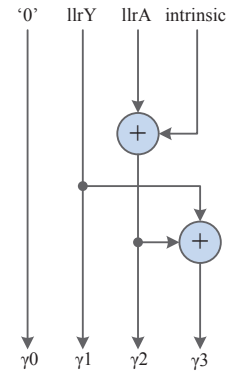
The current 3GPP LTE standard features native code rate of 1/3, while higher code rates can also be achieved through external rate matching. All the block sizes are even numbers and are divisible by both 4 and 8 , besides all the block sizes greater than 512, 1024 and 2048 are also divisible by 16, 32 and 64 respectively [17]. The complete architecture is shown in Fig. 1.

The major modules of the decoder are: (i) the 16 SISO processors, which execute the BCJR algorithm, (ii) the 16 memories storing the extrinsic values, (iii) two 16x16 switching

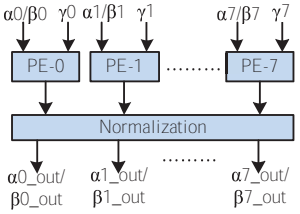
 ISBN: 978-1-61208-264-6



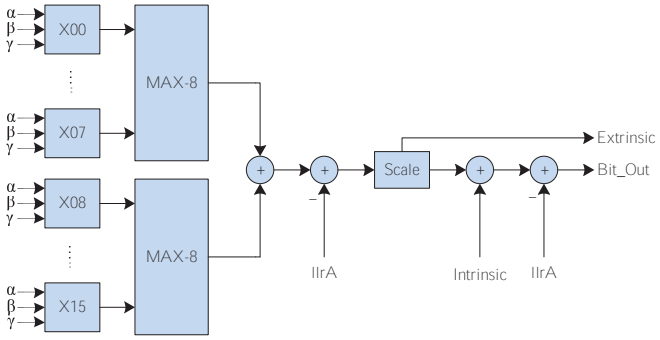
(a) SISO Processor



(b) α or β Branch Metric Unit (BMU)



(c) α and β State Metric Unit



(d) λ-Processor

Figure 2. SISO Processors and Its Major Modules

networks for data transfer from processors to memories and from memories to processors, (iv) the address generator which generates addresses using (7) and (v) the control unit. 3GPP LTE standard uses QPP (Quadratic Permutation Polynomial) [12], which has been proved to be contention free. The QPP interleaver of size N can be expressed as,

$$f(x_o) = \left(f_1 \cdot x_o + f_2 \cdot x_o^2\right) \bmod N \qquad (7)$$

The details of the interleaver are shown in [18], where all the addresses are generated on-the fly for any parallelism factor and f_1 and f_2 are given in the standard and their value depends on the block size N [17]. The decoder is designed to support the maximum code length i.e., N=6144. The maximum depth of the extrinsic memories is N/P, where N=6144 and P=16. On each clock cycle the address generators output 16

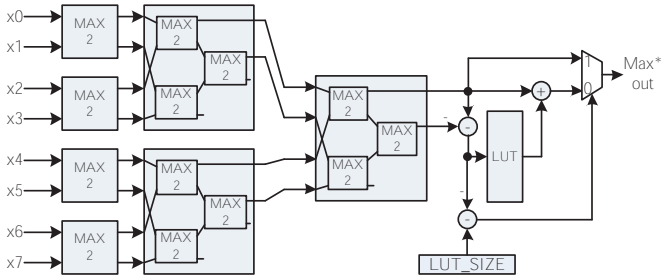


Figure 3. Non-recursive max* module with 8 inputs

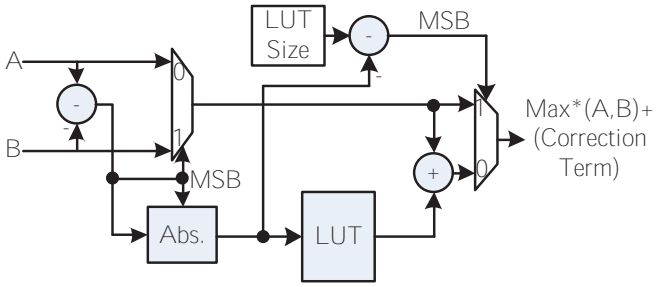


Figure 4. Non-recursive max* module with 2 inputs

addresses and the minimum of these 16 address is calculated through MIN-16 block, which is the effective address for each extrinsic memory. During the forward recursion, all generated addresses are also stored in the *AddrBuffer*, which is a LIFO of W height, where W is the maximum window length. After forward recursion through one complete window, the backward recursion is started and also on the same time the SISO produces the extrinsics, which are passed to the memories through the switching matrix. The address for the write operation (in the extrinsic memories) is always taken from the *AddrBuffer.*

α-init and β-init modules calculate and hold the initial values of α and β. The encoder always starts from all-zero state, which implies that α-init is a pre-computed value, while for β, three tail bits are used with a backward recursion to calculate the value of β at $N-1$ state.

The batcher sorters [19] are used to manage the switching data from SISO to memory and from memory to SISO [20]. The switching matrix takes the data, coming from a memory bank, and the address, generated by the corresponding *Addr* generator, and it sorts the data w.r.t. the addresses. Two switching matrix are used in order to have parallel read and write operations.

There are 16 SISO processors, where each processor passes the α and β to the neighboring SISO, as suggested in [6]. The β-in for the first, second, fourth and eights is multiplexed with either the β-out of the neighboring SISO or from the β-init module. The SISO processors will be discussed in detail in the next section.

The control unit enables the decoder to work for different code lengths. Each code length is associated to a parallelism degree that can be 1,2,4,8 and 16. For example, it enables only 1 SISO processor and only one address generator when the smallest block size is selected. Similarly, the select signals for

 ISBN: 978-1-61208-264-6

Table I. QUANTIZATION

	No. of bits
Channel LLRs λ^s, λ^p	6
Ext./Intrinsics $\lambda^{int.}$	8
Branch metrics γ	12
State Metrics α, β	12

the multiplexers at the inputs of α and β processors are sent by the control signal, in order to pass the correct $\alpha's$ and $\beta's$ to the processors at the end of one N/P bits and $N/(P/NWP)$ bits respectively. Similarly, the control unit is adaptive enough to generate control signal for the data path in order to have correct operation scheduling.

A. SISO processor

The SISO, shown in Fig. 2, is the main processing unit in turbo decoding. In this work, 16 parallel SISO processors are employed, each of which implements (1) to (5) to calculate the extrinsic values and the output bits. The complete architecture of the SISO unit is shown in Fig. 2. Each SISO gets channel LLRs i.e., systematic and parity, the apriori information from memories and the starting state metrics for α and β recursions from other SISOs. The incoming frame is divided in *P* input buffers, where *P* is the number of SISOs. Each SISO decodes the corresponding input bits. So each SISO works on $\frac{N}{P}$ bits, where *N* is the block size. According to Fig. 2, the SISO performs the following decoding operations:

1) The α-BMU (Branch Metric Unit) calculates the γs' (branch metrics) combining the LLRs and the apriori information using (5). At the same time, the BMU-MEM stores all the incoming LLRs and the intrinsic information. BMU-MEM is a LIFO memory. The architecture of the BMU is shown in Fig. 2b.
2) After each calculation of α-BMU, the α-processor calculates the 8 new αs' using previous α and the γ coming from BMU. Eight processing elements (PE-0 ... PE-7) execute in parallel following (3). The new αs' are normalized subtracting the maximum from each of them. Finally the calculated αs' are stored in the α-MEM, which is a LIFO memory as well. The architecture of α-processor is shown in Fig. 2c.
3) When the αs' for a complete window are calculated, the β-BMU and the β-processor start calculating the the $\gamma s'$ and βs' respectively, in the backward direction using (5) and (4).
4) The λ-processor executes in parallel with the β-processor. So after each calculation of a β, the λ-processor calculates the extrinsic, using (1), and also estimates the output bit. The architecture is shown in Fig. 2d.

Each SISO uses three different memories i.e., α-MEM, BMU-MEM and the β-MEM. The α-MEM stores the αs' while traversing the trellis in the forward direction and the BMU-MEM stores the incoming intrinsic and the a-priori LLRs. In Table I, the number of bits required to represent each data managed by the decoder is shown. Table II shows the total memory utilization in the architecture.

Table II. MEMORIES USED IN THE ARCHITECTURE

Unit	Size (kbit)
α−MEM	36.864
BMU-MEM	9.216
β−local-MEM	24.576
Addr. LIFO	0.216
Addr. Init	114.56
Input Buff.	110.592
Output Buff.	6.144
Extrinsic Mem.	49.152
Total	351.32

B. Non-recursive max* operator

The λ-processor executes (1) and (2). In (1), the non-recursive max* operator [11] is used to find the maximum from all the inputs. In general the max* operator is recursive in nature, where on each recursion it finds the maximum from 2 inputs and then the result is compared with the third one and so on. In recursive max* for *n* input values the max* operator is applied recursively *n*–1 times, From (8) it is evident that the max* operator having *n* input values can be computed non-recursively as it requires only knowledge of the maximum among *n* values and an additive correction term depending on the second maximum value among *n* values. The non-recursive max* is implemented to find the two maximum from the 8 inputs using a simple maximum finding tree and a small look-up table (LUT), as shown in Fig. 3. Fig. 4 shows the max* implementation with 2 inputs, which is used in calculation of (3) and (4). The max* operator can be given as

$$\overset{*}{\max_{i=1:n}}(x_i) \simeq \max_{i=1:n}(x_i) + \log\{[1+\exp(-\delta)]\} \tag{8}$$

The $\log\{[1+\exp(-\delta)]\}$ in (8) is the correction term, which is pre-computed and is stored in a small LUT. The difference between the first maximum and the second maximum , namely δ, becomes the address for the LUT. The data from the LUT is then added to the first maximum to find the max* output.

IV. RESULTS

To decrease the amount of memory and to increase the throughput, large number of windows are used for large code lengths. Each SISO processor works with up to 16 windows. In this case, each SISO is decoding 384 bits. The window is 24 bit long. So at each clock cycle, the address generators produce 16 addresses. The data from the memory and the corresponding addresses from the address generators are fed into the permutation network and finally based on the sorting of addresses, the data reaches its destinations [18].

The throughput of the decoder is calculated as in [23]:

$$Throughput = \frac{N \times F}{2.I\left(\frac{N}{P}+W\right)} \tag{9}$$

where *N* is the number of decoded bits, *F* is the clock frequency at which the decoder is synthesized, *I* is the number of iterations, *P* is the number of SISO processors, and *W* is the window length. The BER and FER results, shown in Fig. 5 and Fig. 6 respectively, are obtained with 5 and 9 iterations for code length 6144. Stopping the decoding after 5 iterations

 ISBN: 978-1-61208-264-6

Table III. RESULTS COMPARISONS: Radix/Processor (Rdx/Proc.), CMOS Technology Process (Tech.), Iterations (Iter.), Clock Frequency (Freq.), Memory (Mem), Active Area (Area), Normalized Area (N.A.), Throughput (TP.), Throughput to Area Ratio (TAR = Mbps/mm), Decoding Efficiency (DE = Bits/Cycle/Iterations)

Design	**Proposed**	**[6]**	[7]	[8]	[17]	**[20]**	[18]	[21]	[22]
Standard	LTE	LTE,WiMAX	LTE	LTE	LTE	LTE	LTE	LTE	LTE,WiMAX, DVB-RCS
Rdx./Proc.	2/16	4/8	2/8	2/8	2/64	4/8	2/16	2/32	4/2
Tech (nm)	90	130	90	90	65	130	90	90	65
Iter.	5	8	8	8	6	5.5	5	5.5	6
Freq.	250	250	275	275	400	300	200	486	520
Mem (Kb)	351.32	N/A	N/A	N/A	N/A	129	413.35	N/A	N/A
Area (mm^2)	1.62	10.7	2.10	2.10	8.3	3.57	2.10	13.82	0.644
N.A. (mm^2)	1.62	5.12	2.10	2.10	15.91	1.71	2.10	13.82	1.234
TP. (Mbps)	376.47	187.5	130	129	1310.72	390.6	284.4	1138	170
TAR	232.39	36.62	61.90	61.42	82.83	228.42	135.43	82.34	137.76
DE	7.53	6	3.78	3.75	19.66	7.16	7.11	12.88	1.96

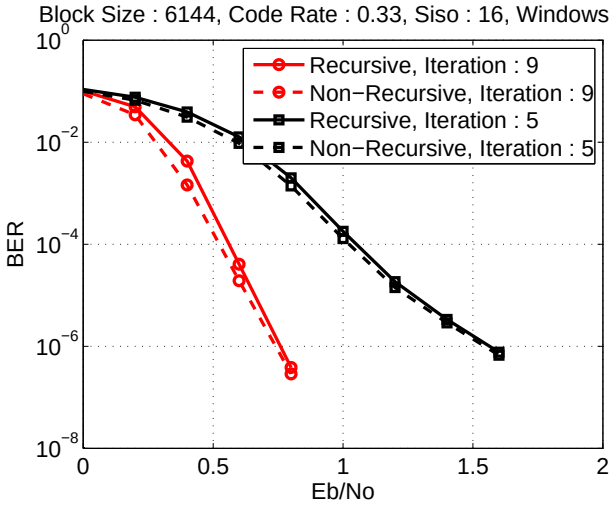


Figure 5. Bit Error Rate (BER), code rate 1/3

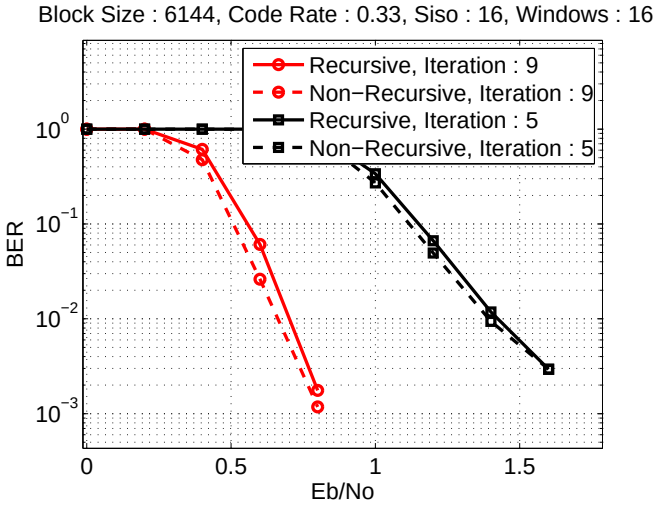


Figure 6. Frame Error Rate (BER), code rate 1/3

instead of 9 introduces a penalty of about 0.5 dB at BER level 10^{-4}, but offers increased throughput and reduced energy dissipation.. It is also shown in Fig. 5 and 6 that the non-recursive max* achieves slight gain over the recursive one.

The synthesis is carried out on Synopsys Design Vision with 90 nm standard cell technology at a clock frequency of 250 MHz. The proposed architecture achieves a maximum throughput of 376.4 Mbps, occupying an area of 1.62 mm^2.

Table III shows the comparison of the proposed architecture with some already published decoders. The comparison is carried out in terms of implementation technology, supported codes and decoded modes, maximum achieved throughput, internal parallelism and occupied area. The area of each implementation is scaled up to 90 nm process technology for fair comparison. The scaling factors of $(\frac{90}{130})^2$ and $(\frac{90}{65})^2$ are used for 130 nm and 65 nm technologies, respectively. A parameter called Throughput to Area Ratio (TAR) [24] is used to evaluate the area efficiency of the designed decoder. Another metric used for comparison purposes is Decoding Efficiency (DE), defined as the number of decoded bits per clock cycle per iteration. DE evaluates the degree of parallelism actually offered by the decoder architecture. As shown in Table III, the proposed architecture achieves better TAR than the other implementations. The DE is better than [6], [7], [8], [20], [18] and [22], whereas [17] and [21] achieve better DE, but they are very expensive architectures, with huge area occupation. So the proposed architecture achieves more than sufficient throughput for 3GPP LTE standard at the cost of lower occupied area than required for most of the alternate solutions.

V. Conclusion

In this paper, an already developed parallel turbo decoder for 3GPP LTE standard is modified with a new architecture of non-recursive max* operator. The analysis shows that with the new max* operator, significant area can be saved and throughput enough to meet the standard requirement can be achieved. The new max* operator is implemented with a very simple logic i.e., by finding just the two maximums and then adding them with a correction term, taken from a LUT.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. 1". In *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, page 1064, May 1993.

 ISBN: 978-1-61208-264-6

[2] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1". *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004)*, 2006.

[3] "3GPP TS 36.212 v8.0.0: Multiplexing and Channel Coding". *3rd Generation Partnership Project*, 2007-2009.

[4] "UMTS : Universal Mobile Telecommunications System". http://www.3gpp.org/Technologies/Keywords-Acronyms/article/umts.

[5] "Consultive Committee for Space Data Systems Recommendation for Space Data System Standards TM Synchronization and Channel Coding CCSDS 121.0-B-2", Sep. 2003.

[6] J. H. Kim and I. C. Park. "A Unified Parallel Radix-4 Turbo Decoder for Mobile WiMAX and 3GPP-LTE". In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, page 487, Sept. 2009.

[7] C. C. Wong and H. C. Chang. "Reconfigurable Turbo Decoder With Parallel Architecture for 3GPP LTE System". *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 57(7):566, July 2010.

[8] C. C. Wong, Y. Y. Lee, and H. C. Chang. "A 188-size 2.1mm^2 Reconfigurable Turbo Decoder Chip with Parallel Architecture for 3GPP LTE System". In *VLSI Circuits, 2009 Symposium on*, page 288, June 2009.

[9] G. Masera. "VLSI for Turbo Codes". In Keattisak Sripimanwat, editor, *Turbo Code Applications*, page 347. Springer Netherlands, 2005.

[10] O. Muller, A. Baghdadi, and M. Jezequel. "Exploring Parallel Processing Levels for Convolutional Turbo Decoding". In *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, volume 2, page 2353, April 2006.

[11] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, and M. Maurizio. "Non-Recursive max* Operator with Reduced Implementation Complexity for Turbo Decoding". *Communications, IET*, 6(7):702, May 2012.

[12] "Multiplexing and Channel Coding, 3GPP TS 36.212 Version 8.4.0, September 2008".

[13] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara. "Soft-Input Soft-Output Modules for the Construction and Distributed Iterative Decoding of Code Networks". *European Transactions on Telecommunications*, 9(2):155, Sep. 1998.

[14] 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*, 20(2):284, Mar. 1974.

[15] P. Robertson, E. Villebrun, and P. Hoeher. "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain". In *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, volume 2, page 1009, Jun 1995.

[16] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, and M. Maurizio. "On Optimal and Near-Optimal Turbo Decoding using Generalized max Operator". *Communications Letters, IEEE*, 13(7):522, July 2009.

[17] Y. Sun and J. R. Cavallaro. "Efficient Hardware Implementation of a Highly-Parallel 3GPP LTE/LTE-Advance Turbo Decoder". *Integration, the VLSI Journal*, 44(4):305, July 2010.

[18] A. Ahmed, M. Awais, A. ur Rehman, M. Maurizio, and G. Masera. "A High Throughput Turbo Decoder VLSI Architecture for 3GPP LTE Standard". In *Multitopic Conference (INMIC), 2011 IEEE 14th International*, page 340, 2011.

[19] K. E. Batcher. "Sorting Networks and their Applications". In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), page 307, New York, NY, USA, May 1968. ACM.

[20] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang. "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE". *Solid-State Circuits, IEEE Journal of*, 46(1):8, Jan. 2011.

[21] S. M. Karim and I. Chakrabarti. "High-Throughput Turbo Decoder using Pipelined Parallel Architecture and Collision-Free Interleaver". *Communications, IET*, 6(11):1416, July 2012.

[22] R. Al-Khayat, A. Baghdadi, and M. Jezequel. "Architecture Efficiency of Application-Specific Processors: A 170Mbit/s 0.644mm^2 Multi-Standard Turbo Decoder". In *System on Chip (SoC), 2012 International Symposium on*, page 1, Oct. 2012.

[23] M. Maurizio, M. Nicola, and G. Masera. "VLSI Implementation of Wimax Convolutional Turbo Code Encoder and Decoder". *Journal of Circuits, Systems and Computers*, 18(03):535, Sep. 2009.

[24] G. Masera, F. Quaglio, and F. Vacca. "Implementation of a Flexible LDPC Decoder". *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(6):542, June 2007.

 ISBN: 978-1-61208-264-6