SOLVING MEMORY ACCESS CONFLICTS IN LTE-4G STANDARD

C. Chavet, Senior member IEEE, F. Lozachmeur, T. Barguil,
A. S. Hussein and P. Coussy, Senior member IEEE

Lab-STICC, CNRS, UMR 6285, Université Bretagne Sud

ABSTRACT

In mobile telecommunications domain, LTE-4G is currently the most advanced standard available. A major design issue of LTE-4G based systems resides in solving the memory access conflicts when connecting Rate-Matching (RM) and Error Correction Code (ECC) modules. In this paper, we first describe and analyze the problem before proposing a dedicated memory mapping approach. Results show that our method allows removing any conflicts for any block sizes and any parallelism degrees in the context of LTE-4G.

***Index Terms*—**Hardware design, memory access conflicts, LTE standard, Error Correction Codes, Rate Matching.

1. INTRODUCTION

The works of Shannon [1] have demonstrated it is possible to send digital data through noisy channel with high reliability. The idea roughly consist to first encode a digital message with an Error Correction Code (ECC) at transmitter side, and then to decode it at receiver side in order to retrieve the original message (cf. Figure 1). Several ECC have been introduced since so far to sustain this concept.

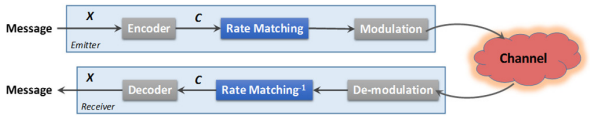

Figure 1. Communication between emitter and receiver

In order to achieve high throughput requirements ECC decoders are based on parallel architectures. In such parallel architectures, several Processing Elements *PEs* are concurrently used to decode the received information [11]. In this context, several memory banks *RAMs* are connected with these *PEs* through a dedicated interconnection network. This network transfers data between *PEs* and *RAMs* according to predefined access orders, i.e. the interleaving rules. Designing efficient parallel hardware architecture i.e. with no memory access conflicts is a very complex and time consuming task (e.g. several man/months in 3GPP-LTE for an expert). A lot of work has been done to explore solutions to avoid interleaving conflicts in ECC architectures ([11] to [17]).

This paper however focuses on LTE-4G standard [9] which relies on QPP algorithm that has been designed to avoid such problems [19]. Unfortunately, conflicting issues still appear when inverse Rate Matching (RM) module and ECC decoder are connected (see Figure 1). Indeed, on the one hand the goal of RM is to match the number of bits of packets sent through the channel, to the number of bits that can be transmitted for a given allocation (required throughput, QoS…). On the other hand, the encoder maps X message digits into C code-word digits (with $C > X$) providing the code rate $r=X/C$ which defines the redundancy introduced by ECC. Unfortunately, the sequence of code-words generated by the inverse rate matching does not match with the order expected by the ECC decoder [19].

This paper is organized as follows. Section 2 describes how LTE-4G RM builds packets for channel transmission and why memory access conflicts appear in parallel LTE decoder. Section 3 proposes a deeper analysis of memory access conflicts between inverse RM module and ECC decoder. Section 4 describes a dedicated memory mapping approach and presents results which show that any conflicts for any block sizes and any parallelism degrees in the context of LTE-4G are removed.

2. LTE-4G OVERVIEW

2.1. Building payload for channel transmission

To transmit a code at an arbitrary rate (depending on available physical resources) a Rate Matching is performed after the turbo encoding (WCDMA [9], which performs a *1/3* rate encoding). For every single input bit, 3 output bits are given: the first bit is the original input bit (called *systematic* bit) ; the remaining two bits are the interleaved versions of the input bit (referred as *parity 1* and *parity 2 bits*, Figure 2). These three streams of *systematic*, *P1* and *P2* are then provided as inputs to the rate matching module.

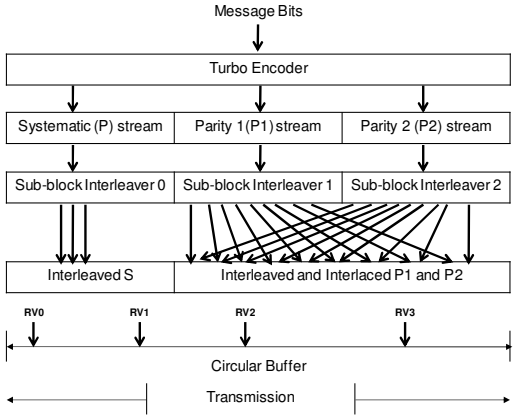

Figure 2. LTE-4G Rate Matching algorithm

RM algorithm punctures the bits of a mother code-word produced by the encoder. First, from the incoming message bits $D = \{d_0, d_1, \ldots, d_{M-1}\}$, with $M<6145$, the turbo-encoder generates three encoded sequences $d_k^{(i)}$, $i=0,1,2$ which are referred as *Systematic*, *Parity 1* and *Parity 2* respectively (Figure 2).

If M is not a regular payload size defined in [9], filler bits are added in order to reach the next upper regular one. For example, if M = 84, 4 filler bits are added in order to reach the nearest bigger payload size $K_7 = 88$. It should be noticed that a filler bit is a <*NULL*> value and that the generated *Parity 2* sequence is not affected by filler bits. The turbo-encoder also computes 4 tail bits added in each generated code-word as RSC. At this step, the number of encoded message bits D is $D=88+4=92$.

Then each sequence is considered as a matrix of 32 columns ($C_{subblock}^{TC}$=32). The bits sent to the sub-block interleavers are denoted as $d_0^{(i)}$, $d_1^{(i)}$, $d_2^{(i)}$,… $d_{D-1}^{(i)}$, where D is the number of bits. The output sequences of bits of the sub-block interleavers are derived as follows:

1- Let $C_{subblock}^{TC} = 32$ be the number of columns of the matrix. The columns of the matrix are numbered *0,1…* $C_{subblock}^{TC} - 1$.
2- Determine the number of rows $R_{subblock}^{TC}$ of the matrix by finding minimum integer such that:

$$D \leq R_{subblock}^{TC} \times 32$$

However, if the resulting matrices are not entirely filled, dummy bits (i.e. <*NULL*> values) are padded at the beginning of the matrices. Now the matrices are ready to perform inter-column permutation, as described in [9], and the resulting interleaved sequences (i.e., $v_k^{(0)}$, $v_k^{(1)}$, $v_k^{(2)}$) are provided to the bit collection step and next interlaced in a circular buffer. Finally, the payload sequence of encoded bits e_k for transmission is generated as defined in the standard:

1- E bits are extracted from the circular buffer starting from an offset k_0 and all dummy and filler bits are discarded.
2- If the algorithm reaches the end of the circular buffer, then it moves to its first element.

In the LTE-4G standard, four redundancy versions (*rv,* a.k.a. *code rate*) are defined: they respectively start to read data at the top of the columns *2*, *26*, *41* or *53* (resp. rv_0, rv_1, rv_2, rv_3). In our pedagogical example with $E=88$ and rv_0, the generated sequence of output bits transferred over the channel, is presented in Figure 3.

4	36	68	20	52	84	32	64	16	48	80	8	40	72	24	56	T	30	62	14	46	78	6	38	70	22	54	86	34	66	18	50	82	10
42	74	26	58	T	29	61	13	45	77	5	37	69	21	53	85	33	65	17	49	81	9	41	73	25	57	T	31	63	15	47	79	7	39
71	23	55	87	35	67	19	51	83	11	43	75	27	59	T	28	29	60	61	12	13	44	45	76	77	4	5	36	37	68	69	20	21	52

Figure 3. Output bit sequence with redundant bits from parity 1 and 2 (resp. dark and light grey)

2.2. Channel de-interleaving and resulting memory conflicts

From the receiver point of view (cf. Figure 4), the input order corresponds to the output sequence e_k described in the previous section. Each input is one *LLR* (Log-Likelihood Ratio) and is quantized on q_w bits. The channel de-interleaver module performs different functions on the incoming quantized bit-LLR:

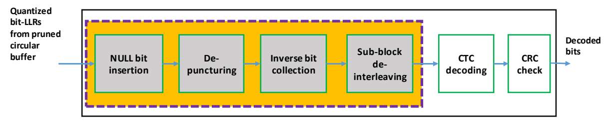

Figure 4. Functional coverage in LTE decoder chain

The first step consists in retrieving turbo-encoded bit sequence and provide it to the parallel turbo decoder (CTC decoding). The turbo decoder receives a vector of $\boldsymbol{K}$ payload bits and produces $\boldsymbol{3k_i + 12}$ coded bits, including filler bits and dummy bits where $\boldsymbol{k_i}$ is one of the 188 block sizes defined in the standard (see [9]), ranging from 40 to 6144 payload bits. The constant value ***12*** refers to termination/tail bits added in the code-word as Recursive Systematic Code (RSC, [21]).

In the next subsections, the way packets are built at the transmitter side is not formally detailed. Rather, a pedagogical example, built with respect to the standard process from [9], is introduced.

Since the sub-block de-interleaver shall support P samples per clock cycles (w.r.t. QPP interleaver [19]), data should be provided respecting a given order described in Figure 5 where each column represents a timing step.

1	**2**	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
23	**24**	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
45	**46**	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
67	**68**	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}

Figure 5. Input order for QPP interleaver (*P=4, 22 cycles*)

To support this scenario in which four data are accessed in parallel, each row of the QPP matrix must be mapped in a dedicated memory bank. However the inverse RM module outputs the data in a different and incompatible order (see Figure 6): four data are still provided in parallel and so each row of the RM matrix must be mapped in a dedicated memory bank. Unfortunately, the two sequences (and their associated memory mapping) do not fit, resulting in memory conflict accesses. For example, data <*2, 24 and 68*> are stored in the same memory: "bank 3"; which is not compatible with QPP interleaver input constraint since data <*2, 24, 46 and 68*> are supposed to be accessed in parallel at time t_2 (Figure 5).

4	52	16	40	30	78	22	66	10	58	45	69	33	81	25	15	39	87	51	75	60	0
36	84	48	72	62	6	54	18	42	29	77	21	65	9	57	47	71	35	83	27	12	1
68	32	80	**24**	14	38	86	50	74	61	5	53	17	41	31	79	23	67	11	59	44	**2**
20	64	8	56	**46**	70	34	82	26	13	37	85	49	73	63	7	55	19	43	28	76	3
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}

Figure 6. Input sequence order without tailing bits (*Systematic*)

3. DEEPER ANALYSIS OF MEMORY CONFLICTS

3.1. Simple memory mapping approach

In order to analyze the memory mapping problem previously introduced, we explored all the bit sequences that could be provided by the IR matching to the QPP interleaver (see Figure

1), as described in the LTE-4G standard. For each, we computed the number of memory access conflicts generated with a simple memory mapping approach. Figure 7 shows the number of conflicts generated by the RM algorithm in the receiver for a parallelism of 8, with the simple and natural memory mapping approach mentioned in the previous section.

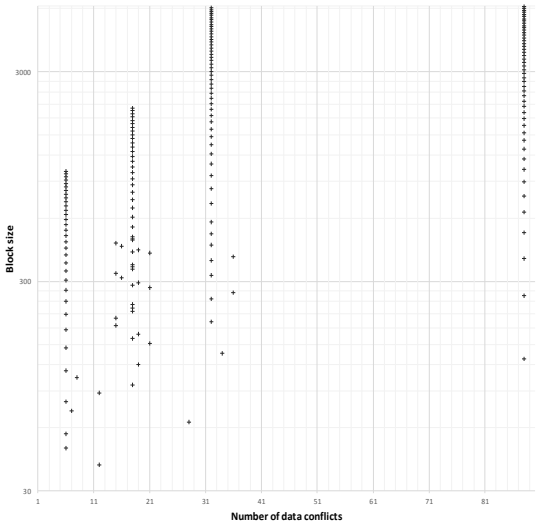

Figure 7. Number of memory conflicts for each block size with a parallelism $P=8$ (log. scale)

It can be observed that if the number of conflicts grows with the number of data, even for the smallest block size (i.e. 40) the number of conflicts to be handled is important (12 in this case). Traditional solution to solve such conflicts consists in adding FIFOs before the memory banks, to temporary store all the conflicting data. However, this widely impacts area, power consumption and timing performances of the final architecture.

3.2. Smarter mapping approaches

A careful analysis of Figure 7 highlights a regularity in the number of memory conflicts depending on the block sizes. The same kind of observations have been made for parallelism 2, 4, 16, 32, 64 and 128. Obviously, these observations confirm that the different block sizes of the standard have been defined with a very smart mathematical approach (see [9][18]). As a consequence, a smart systematic sub-block de-interleaving (Figure 4) approach might be proposed in order to avoid any memory conflicts.

Different memory mapping algorithms have been used ([11] to [17]). In most of the cases, the solutions were not able to achieve no conflicts for all the block sizes and all the considered parallelism degree. This is the reason why we decided to address this problem from a different perspective i.e. reducing the constraint from QPP interleaver.

Let's consider a LTE-4G decoder architecture with a parallelism $P_s=2$, a payload size $K=64$ and an encoding ratio $r=1/2$. Of course, this is a very simple and pedagogical example to illustrate our approach.

Thanks to the payload generation procedure described in the previous section, the output bit sequence illustrated in Figure 8 is generated.

12	44	28	60	8	40	24	56	16	48	0	32	64	6	38	22	54	14	46	30	62	10	42	26	58	18	50	2	34	66	5	37	21	53
13	45	29	61	9	41	25	57	17	49	1	33	65	7	39	23	55	15	47	31	63	11	43	27	59	19	51	3	35	67	67	4	36	4
36	20	52	20	52	12	44	12	44	28	60	28	60	8	40	8	40	24	56	24	56	16	48	16	0	48	32	0	64	32	64	6	38	6
38	22	54	22	54	14	46	14	48	30	62	30	62	10	42	10	42	26	58	26	58	18	50	18	2	50								

Figure 8. Output bit sequence ($K=64$, $r=1/2$)

Once tail bits (i.e. bits *64* to *67*) have been removed, the resulting de-interleaver bit sequence is:

12	44	28	60	8	40	24	56	16	48	0	32	6	38	22	54	14	46	30	62	10	42	26	58	18	50	2	34	5	37	21	53	13	45
29	61	9	41	25	57	17	49	1	33	7	39	23	55	15	47	31	63	11	43	27	59	19	51	3	35	4	36	20	52				

Figure 9. De-interleaver input bit sequence

Since $P=2$ and $K=64$, each memory (*bank 0* and *bank 1*) has to store $K/P=32$ data. Thus, applying a simple memory mapping approach generates the mapping presented in Figure 10.

12	44	28	60	8	40	24	56	16	48	0	32	6	38	22	54	14	46	30	60	10	42	26	58	18	50	2	34	5	37	21	53		
13	45	29	61	9	41	25	57	17	49	**1**	**33**	7	39	23	55	15	47	31	63	11	43	27	59	19	51	3	35	4	36	20	52		

Figure 10. Simple memory mapping results (K=64 and P=2)

In this pedagogical example, the QPP interleaver processes the data in the order defined in Figure 11.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64		
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}	t_{23}	t_{24}	t_{25}	t_{26}	t_{27}	t_{28}	t_{29}	t_{30}	t_{31}	t_{32}		

Figure 11. QPP interleaver input order (K=64 and P=2)

Once again, it can be observed that the proposed memory mapping (Figure 10) generates conflicts in the decoder: e.g., at time t_2 data *1* and *33* have to be accessed at the same time (Figure 11), -*i.e. they must be stored in different memory banks*- but it is not possible since they are both stored in *bank 1*. A dedicated memory mapping approach is thus required.

4. PROPOSED MEMORY MAPPING APPROACH

We propose to replace the original QPP input order constraint by a custom "*Temporary mapping* constraint" T_{Map}. The idea is to find a mapping constraint that is a simple permutation π of the original QPP data input order, but easier to solve. Hence, these data are read from memories and written to the QPP module, through a simple rerouting network π^{-1}. From the pedagogical example introduced in the previous section, a hypothesis is made:

for a given set of K data, a parallelism P, and a set of $B=P$ memory banks,

(1) each memory banks stores K/P data;
(2) for a given memory address α, the difference between any 2 data in each memory bank at address α is lower than P, and they can be ordered incrementally.

In few words, this means that if in the QPP interleaver any two data are accessed at the same cycle, they cannot be stored in the same memory bank in the memory mapping generated with T_{Map} constraint.

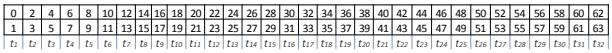

Figure 12. Intermediate T_{Map} constraint (K=64 and P=2)

The obtained intermediate T_{Map} constraint supposes that the QPP interleaver is able to access data as shown in Figure 12.

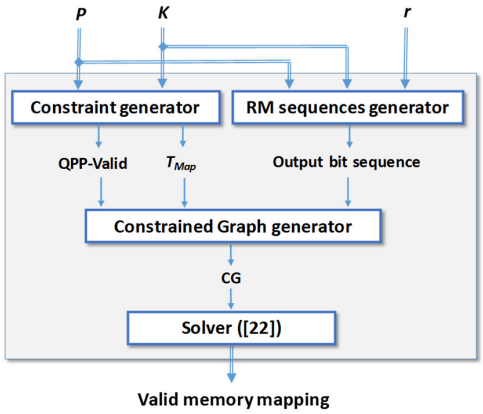

Figure 13. Conflict free memory mapping approach with T_{Map} constraint

Then, a Constraint Graph *CG* taking into account all these constraints is defined and a constraint solver (from [22]) is applied (see Figure 13).

a. Output bit sequence from inverse rate matching

b. Input bit sequence for QPP interleaver

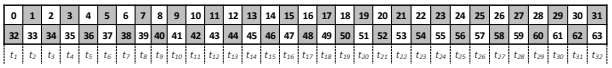

Figure 14. Resulting memory mapping
(*Bank 0 in white, Bank 1 in grey)*

If we apply this memory mapping approach the resulting memory mapping is:

- Bank 0 = *{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63}*
- Bank 1 = *{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62}*

This approach is able to generate conflict free memory mapping with respect to the output bit sequence of the inverse RM module and the QPP interleaver input order (Figure 11). The resulting architecture does not need any additional memories (see Figure 1), since we are able to find no-conflict memory mapping in any case. Only few MUXs are added to route the data from a memory bank to the right decoder input.

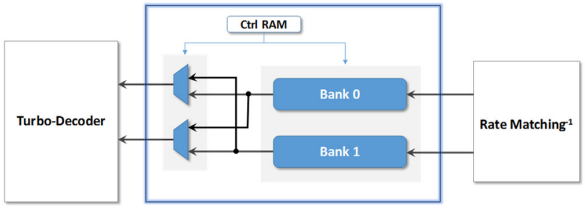

Figure 15. Final architecture for our example (Figure 14)

5. EXPERIMENTS

In our experiments the proposed memory mapping approach has been applied for all combination of *K* data, *r* redundancy value and parallelism *P* (with all the *K* and *r* values extracted from [9], and $P \in \{2, 4, 8, 16, 32\}$). The solver runs on a laptop with an *Intel core* i7-6700HQ (2,6 GHz) and 16Go RAM. The area results are presented in NAND equivalent to be agnostic about the final target (FPGA, ASIC…).

Parallelism P=8	Designed by hand (No conflict)	Simple mapping solution (FIFOs)		Proposed approach	
		Min. conflicts	Max. conflicts	Min. conflicts	Max. conflicts
Number of conflicts	0	6	88	6	88
Impact factor on throughput	no	- 2,2x	- 1,12x	no	
Additional area (NAND eq.)	0	30	440	40	40
Design time (sec)	several mounth	< 1,5		< 1,02	

Figure 16. Cost comparisons for P=8 (see figure 7)

Figure 16 shows a comparison between a handcrafted design (considered as the reference), a FIFO-based architecture obtained with a simple memory mapping approach and an architecture using the memory mapping approach we have presented in the previous section. Hence, contrary to the FIFO-based solution, our approach has no impact on the throughput since, thanks to the conflict-free memory mapping, no memory access is postponed. The area overhead is at worst comparable with the FIFO-based approach and at best 11 times smaller. This additional area is constant because a Benes network is used to support all the permutations while FIFO-based architectures uses dedicated set of MUXs to dispatch the data.

6. CONCLUSION

A major design issue of LTE-4G based systems resides in solving the memory access conflicts when connecting Rate-Matching (RM) and Error Correction Code (ECC) modules. In this paper, we have described and analyzed the problem before proposing a dedicated memory mapping approach to remove all the memory access conflicts. This solution, based on temporary mapping constraints and a mapping solver, has been validated for any of the 188 block sizes defined in the standard and for parallelism from 2 to 32.

Acknowledgements
This research has been founded by the FEDER project FlexDEC-5G.

6. RELATED WORK

[1] C.E. Shannon, “A mathematical theory of communication”, *Bell System Tech. Jour., vol.27, pp. 379–423 and 623–656*, 1948.

[2] I.S. Reed and G. Solomon, “Polynomial Codes Over Certain Finite Fields”, *J. Soc. Ind. Appl. Math., Vol. 8, pp. 300-304, and Math. Rev. Vol. 23B, pp. 510*, 1960.

[3] *DVB, Frame structure channel coding and modulation for the second generation digital terrestrial television broadcasting system (DVB-T2)*, DVB Document A122, 2008.

[4] *WIFI, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput*, IEEE P802.11n/D5.02, Part 11, 2008.

[5] *WiMAX, 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,* IEEE P802.16e, Part 16, 2006.

[6] E. Berrou and A. Glavieux, “Near Shannon limit error-correcting coding and decoding: turbo-codes”, *Proc. IEEE Int. Conf. on Communications (ICC 93), pp.1064–1070*, Geneva, Switzerland, 1993.

[7] R. Pyndiah, “Near-Optimum Decoding of Product Codes: Block Turbo Codes”, *IEEE Trans. Comm., vol. 46, pp.1003-1010*, 1998.

[8] *HSPA, Technical specification group radio access network;multiplexing and channel coding (FDD)*, 3GPP, 25.212 V5.9.0, 2004.

[9] *LTE, Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access; Multiplexing and Channel Coding (Release 8),* 3GPP Std, TS 36.212, 2008.

[10] *DVB-SH, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for satellite services to handheld devices (SH) below 3 GHz*, ETSI EN 302-583 V1.1.1, 2008.

[11] A. H. Sani *et al.*, “A First Step Toward On-Chip Memory Mapping for Parallel Turbo and LDPC Decoders: A Polynomial Time Mapping Algorithm”, in *IEEE Trans. on Sig. Proc., vol. 61, no.16, pp.4127-4140*, 2013. DOI: 10.1109/TSP.2013.2264057

[12] S. Ur Rehman *et al.*, “In-place memory mapping approach for optimized parallel hardware interleaver architectures”, *in DATE, Grenoble, France, pp.896 – 899*, 2015.

[13] C. Chavet *et al.*, “Hardware Design of Parallel Interleaver Architectures: A Survey”, in *Advanced Hardware Design for Error Correcting Codes*, pp.177-192, Springer, 2015. ISBN: 978-3-319-10568-0 (Print) / 978-3-319-10569-7 (Online).

[14] M.J. Thul, “Optimized concurrent interleaving architecture for high-throughput turbo-decoding”, *in 9th Int. Conf. Electron. on Circuits and Syst., vol. 3, pp.1099–1102*, 2002.

[15] A.Tarable *et al.*, “Mapping interleaving laws to parallel turbo and LDPC decoder architectures”, in *IEEE Trans. on Inf. The.*, *vol.50, no.9, pp. 2002-2009*, 2004. DOI: 10.1109/TIT.2004.833353

[16] M. Karkooti *et al.*, “Configurable LDPC Decoder Architecture for Regular and Irregular Codes”, *Springer Journal of VLSI Sig. Proc. Sys. for Signal, Image and Video Technology, vol. 53, no. 1-2*, 2008. DOI: 10.1007/s11265-008-0221-7

[17] A. Giulietti *et al.*, “Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements”, in *Electronics Letters, vol.38, no.5, pp. 232-234*, 2002. DOI: 10.1049/el:20020148

[18] *3GPP, Evolved Universal Terrestrial Radio Access (E-UTRA): Multiplexing and channel coding*, TS 36.212, 2009

[19] C.C. Wong and H.C. Chang, “Reconfigurable Turbo Decoder With Parallel Architecture for 3GPP LTE System”, *in IEEE Trans. on Cir. and Sys. II: Express Briefs, vol.57, no.7, pp.566-570*, 2010. DOI: 10.1109/TCSII.2010.2048481

[20] P. Frenger *et al.*, “Performance comparison of HARQ with Chase combining and incremental redundancy for HSDPA”, in *IEEE 54th Vehicular Technology Conf., (Cat. No.01CH37211),* Atlantic City, NJ, 2001, vol.3, pp. 1829-1833.

[21] C. Berrou et al., “Near Shannon limit error-correcting coding and decoding: Turbo-codes”, Proceedings of ICC '93 - IEEE International Conference on Communications, Geneva, Switzerland, 23-26 May 1993.

[22] S. Ur Rehman, C. Chavet and P. Coussy, "In-place memory mapping approach for optimized parallel hardware interleaver architectures", In Proceedings of DATE 2015, Grenoble, France, march 2015