In-place Memory Mapping Approach for Optimized Parallel Hardware Interleaver Architectures

Saeed Ur Reehman, Cyrille Chavet, Philippe Coussy and Awais Sani
Lab-STICC laboratory / Université de Bretagne Sud
Lorient, France

***Abstract*— Due to their impressive error correction performances, turbo-codes or LDPC architectures are now widely used in communication systems and are one of the most critical parts of decoders. In order to achieve high throughput requirements these decoders are based on parallel architectures, which results in a major problem to be solved: parallel memory access conflicts. To solve these conflicts, different approaches have been proposed in state of the art resulting in a lot of different architectural solutions. In this article, we introduce a new class of memory mapping approach solving the conflicts with an optimized architecture based on in-place memory mapping for any application.**

Keywords—interleaver; memory mapping; architecture; error correction codes

I. INTRODUCTION

The impressive and rapid increase in data traffic in wireless communication domain (smart-phones, DTV, internet-of-things…) begins to overload network capacity and researchers are still exploring new techniques to target high throughput application and devices. In order to reliably transfer data at high rates on networks, several technologies such as OFDM, MIMO and advanced error correction technique are used. Among them Turbo-codes [9] and Low Density Parity Check (LDPC) [7] are the most efficient to achieve very low bit error rates for low Signal-to-Noise Ratio (SNR) applications. These Error Correcting Codes (ECC) are used in various domains for different standards such as wireless accesses (WiMAX, UWB…), telecommunications (HSPA, LTE…), high-speed wired links (10GBASE-T…), digital video broadcasting (ATSC-x, DVB-Sx…) [1]-[5]. ECC are created by a set of Processing Elements PEs (encoders/decoders) connected to a set of memory elements (registers, RAMs…) through an interconnection network (butterfly, barrel shifters…), cf. Fig. 1. The data blocks are exchanged by the PEs according to the interleaving law defined by the target standard. In order to respect the required throughput performances, parallel architectures must be employed to speed up the iterative decoding process. However, such parallel architectures suffer from memory access conflicts when the interleaving law requires simultaneous access to the same memory bank referred as "*memory conflict problem*" [9][21]. Several approaches are proposed in state of the art in order to solve it.

In this paper we propose a new approach to find conflict free solution targeting in-place memory mapping based architecture. This methodology leads to optimized architectures compared to state of the art solutions in terms of hardware cost.

The paper is organized as follows: Section II summarizes different existing approaches whereas Section III explains the memory mapping problem. The proposed approach is presented and illustrated in Section IV. Experiment and results are given in Section V.

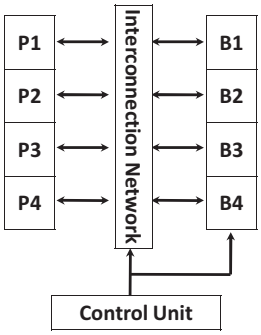

Fig. 1. General architecture

II. RELATED WORKS

As shown in [21], the existing solutions to solve memory access problem for ECC can be categorized into three families. A first solution to avoid the collisions consists of designing dedicated conflict-free interleaver rules. In [6], the authors defined collision-free permutations thanks to a combination of a spatial and a temporal permutation. In [19], the authors described a design rule to obtain such interleavers with an incremental algorithm that generates collision-free interleavers by adding new elements to a small initial permutation in successive steps. The memory conflict problem for LDPC codes is handled by constructing *structured LDPC* codes [17] in which the H matrix must be constructed properly to avoid conflict accesses. Structured codes are part of telecommunication standards such as WiFi [4] or WiMAX [5]. However, they only support one class of LDPC codes. A generic approach to solve mapping problem is required to handle various existing and future classes of LDPC codes, e.g. non-binary LDPC codes. Moreover, the conflict free data access order in the decoder can be different from the data access order coming from the channel, e.g. QPP interleaver: conflict problem is reported on the channel interleaver [21].

In second family of approaches, collision problem is tackled either through the addition of extra memory elements and/or complex interconnection network. In [16], the authors proposed an approach based on Double-Buffer Contention-Free (DBCF) architecture. The *DBCF* architecture is built around the interleaver between the PEs and memory banks. It consists of FIFOs associated with processors, circular buffers, multiplexers and bypass units. However, this architecture is configured by using simulation traces. In [14], a different interconnection network is presented to provide scalability and to allow any permutation to be routed. Memory conflicts are managed by deflecting the conflicting packets appropriately until they reach the target processor rather than blocking or buffering them. However, these flexible networks suffer from large silicon area and cost due to increased buffer control architecture necessary to manage conflicting packets.

In the third kind of approach, different algorithms are proposed to provide conflict free concurrent accesses to all processing elements. The idea is to pre-process data blocks at design time in order to determine the memory locations for each data element avoiding the conflict accesses. In [12] the authors proposed a heuristic to find conflict free memory mapping for turbo decoders. They proved that conflict free memory mapping always exists to tackle collision problem for any code. However, it is based on a simulated-annealing algorithm in which one cannot predict when the algorithm will end and the final architecture is not optimized. In [10], a simplified approach called Static Address Generation Easing (*SAGE)* is presented. This approach is able to find an in-place conflict free memory mapping with respect to a targeted interconnection network, but only for turbo-codes. A more generic approach is proposed in [18] [13] for LDPC codes but they are based on a multiple read multiple write (MRMW) mapping architecture. In [15], a polynomial time memory mapping algorithm is proposed which is based on bipartite edge coloring algorithm for turbo codes and for LDPC codes. However, this approach is able to find an in-place mapping only for Turbo-Codes.

III. IN-PLACE MEMORY MAPPING PROBLEM

Let us consider a parallel turbo decoder architecture composed of P processing elements $PE = \{PE_1,...PE_p\}$ and P memory banks $B = \{b_1,...b_p\}$ to store $L = \{e_1,...e_L\}$ data elements. Each bank has to store $M=L/P$ data. Fig. 2 models memory accesses schedule as a table for a parallelism $P=4$. Lines represent the set of data used by each processing elements (e.g. PE_1 accesses data *0, 1, 2, 6 and 4*). Columns represent computation instants needed to process the $L=12$ data, e.g. at time instance t_1 data *0, 3, 6 and 9* will be accessed concurrently by PE_1, PE_2, PE_3 *and* PE_4 respectively.

In-place memory mapping finds a conflict-free memory mapping for a given memory access schedule (i.e. interleaving law and parallelism) and ensures that assignment of a data element in a memory element remains the same during all the processing steps of the data block. However, many memory conflict problems (e.g. shuffled turbo, LDPC) still require *MRMW* mapping architecture [15] because no existing method is able to find in-place memory mapping solution. If in-place memory mapping architecture could be used to solve these problems then the final architecture would be strongly optimized.

	t_1	t_2	t_3	t_4	t_5	t_6
PE_1	0	1	2	0	6	4
PE_2	3	4	5	8	5	7
PE_3	6	7	8	2	10	1
PE_4	9	10	11	11	3	9

Time

Fig. 2. Memory access schedule

IV. PROPOSED APPROACH

In [15], a polynomial time approach based on bipartite edge coloring algorithm is presented. In this approach, in-place memory mapping architecture is used to solve turbo-like problems and *MRMW* mapping architecture is used to solve LDPC-like problems. Moreover, it has been shown in [13] that if the number of memory is limited to $P=B$, a MWMR memory mapping is mandatory for LDPC-like problems. Hence, in order to obtain an in-place memory mapping for all mapping problem, the solution is to find a memory mapping with more than P memory banks. On the other hand, the Vizing theorem [11] proves that it could be possible to find a coloring with $P+1$ memory banks, *iff* the graph is simple. A *simple graph* contains no parallel edge between any two nodes. Then this theorem can be used to find in-place conflict free memory mapping for simple graph by adding one additional memory bank in the architecture.

Our approach first models memory accesses as a conflict graph and then uses a coloring algorithm based on the Vizing theorem to find an in-place conflict free memory mapping. Our flow (see Fig. 3) is based on two main steps. The first step transforms the memory access matrix (see Fig. 4) into a graph G (see Fig. 5). If this conflict graph is *simple*, our mapping algorithm is applied for edge coloring the graph. It selects a given uncolored edge in the graph G and colors this edge if it is possible. An edge e_i between nodes n_k, n_l can be colored with color c_j iff none of the other edges connected to n_k, n_l is already colored with c_j.

If no simple coloring solution can be found for an edge, then a dedicated procedure must be applied: first, a mapping fan is generated as a sequence of nodes. This fan summarizes all the coloring information of the nodes connected to the edge e_s. Then, this fan can be explored to check all the possible solutions according to different scenarios in the graph for coloring the edge e_s. We have represented each node as $n_i(c_i)$ in which n_i is the time node and c_i is the missing color available for coloring at that code. A Vizing fan is an ordered sequence in which edges at a node n_i is connected to other node n_x with a missing color c_x which is the color of the next edge in the sequence of connected nodes with n_i.

Then, this fan has two possible properties: 1) no repetition of missing colors on the edges in a fan, 2) with repetition of missing colors at the nodes of the fan. In the first case the colors are exchanged in the fan to color the conflicting edge. In the second case, first path is traced and then existence of the loop in the path is detected. Finally, the colors are exchanged and the final coloring is assigned to the graph.

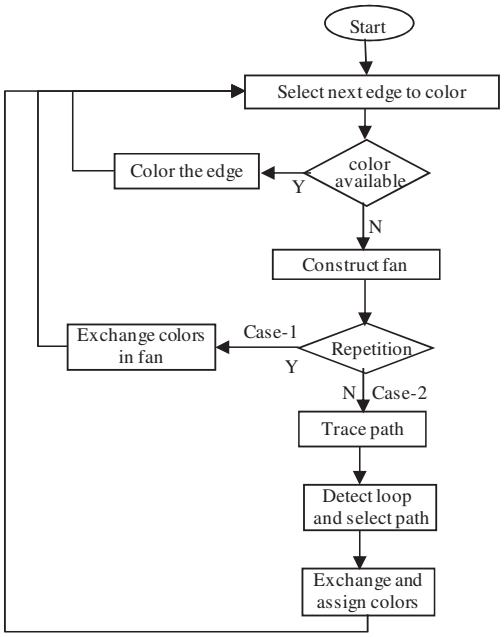

Fig. 3. Proposed design flow

PE_1	0	2	1	18	16	13	4	5	6	21	31	31	12	10	9	23	3
PE_2	1	4	14	19	17	14	11	7	7	22	33	25	26	32	20	24	29
PE_3	2	5	17	20	19	15	12	10	8	8	22	15	27	28	30	25	28
PE_4	3	6	18	21	33	16	13	11	9	23	24	26	32	30	29	0	27
	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}

Fig. 4. Example with two accesses to each data

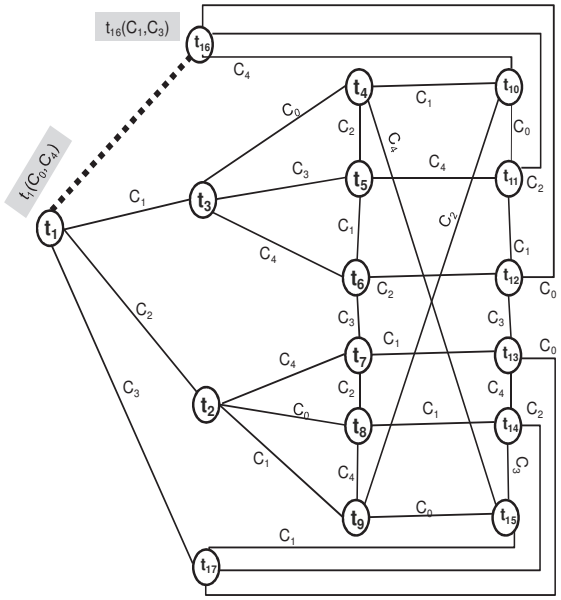

Fig. 5. Graph for edge coloring

Let us consider the example in Fig. 6. The graph G for this example is shown in Fig. 5. In this graph, each edge represents a data element and the nodes connected to this edge represent the two time instances of this data element. The graph G can be colored by using at most $P+1=5$ colors (c_0, c_1, c_2, c_3, and c_4). A first simple coloring step generates a partially colored graph. There is no solution available for the conflict edge e_i between the nodes t_1 and t_{16} as shown by the dotted line in the figure. Therefore, we need to construct a fan in order to apply the whole process as describe below. This fan is shown in Fig. 6 in which the color missing at t_{16} is c_1 (consider only one) so we choose the next edge of the fan colored with c_1. The other time instance connected to edge with color c_1 is t_3 (in which c_2 is missing) so the next edge of the fan should be of color c_2 and so on. After the fan construction, it is analyzed in order to apply our algorithm. In Fig. 9(a) there is no repetition in the missing colors at all the nodes of the fan (excluding starting node t_1). In this case, there must be a color missing at last node of the fan (at t_{17}) which is also missing at starting node (at t_1). Hence, the conflict edge can be colored as the missing color at t_1 is also missing at t_{17}. But we need to shift the conflict edge in the fan in order to be able to color the edge between t_1 and t_{17} with c_4 as it is missing at both t_1 and t_{17} as shown Fig. 6(d). Fig.10 shows the resulting memory mapping.

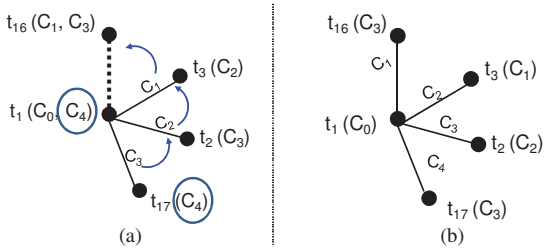

Fig. 6. Vizing Fan coloring – Color repetition in a fan

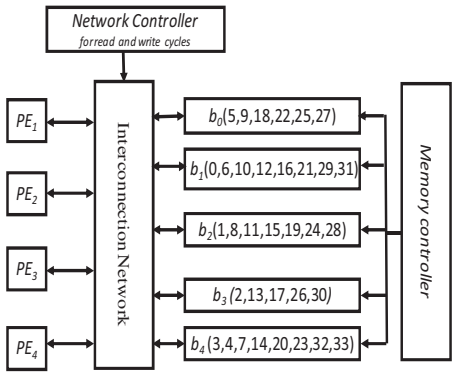

Fig. 7. Resultant in-place architecture

V. Results

Our approach has been applied to solve memory conflict problem for a NB-LDPC code [8] and the results are compared with state of the art approaches. All the results are compared in terms of area. Since the final area of the architecture mainly is mainly due to the control units and the memories (about 95%, see [18]), the PEs and the networks itself are not taken into account in the results. Areas are given in NAND gate equivalent 90nm technology from STMicroelectronics. We performed experiments for different block lengths (from 48 to 788 data) and parallelisms (P=4, 6 and 16). It can be observed that our proposed approach optimizes the final architecture. Compared to [18] our approach reduces the area up to 15%, whereas compared to [13] our in-place memory mapping optimizes the area up to 40%.

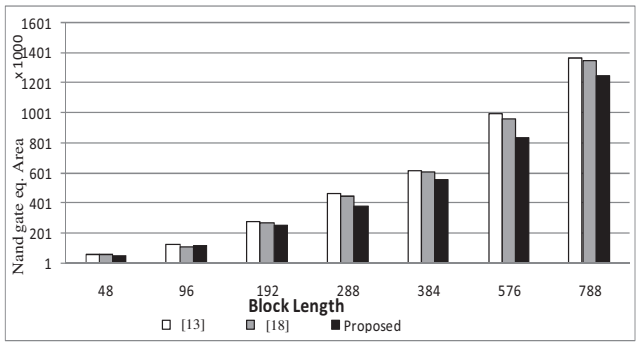

Fig. 8.Area comparison P = 4

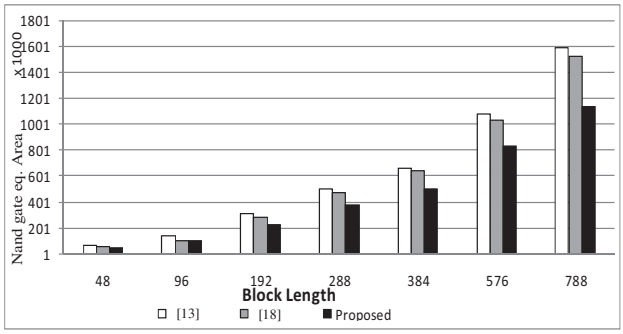

Fig. 9.Area comparison P = 6

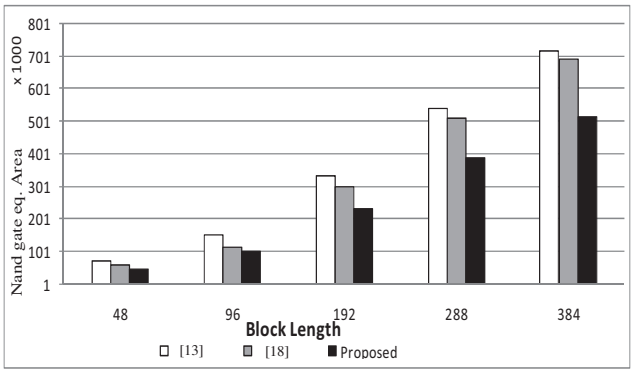

Fig. 10. Area comparison for P = 16

VI. Conclusion

In this paper, we proposed a new approach based on Vizing theorem to produce optimize hardware architectures for memory mapping problems in polynomial time. The proposed approach is based on in-place memory architectures in any case. The result showed that our approach strongly optimize interleaver architectures. However, the proposed method is valid for simple graphs: edge coloring approaches for multi-graphs which can generate in-place memory mapping must be found.

References

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

[2] 3GPP, "Technical specification group radio access network; multiplexing and channel coding (FDD)" June 2004.

[3] *DVB DocumentA122.* "Frame structure channel coding and modulation for the second generation digital terrestrial television broadcasting system (DVB-T2)," 2008.

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

[5] *IEEE P802.16e, Part 16.* "Air Interface for Fixed and Mobile Broadband Wireless Access Systems," Amendment 2: Physical and MAC Layers for Combined Fixed and Mobile Operation in Licensed Bands, and Corrigendum 1, Feb. 2006.

[6] E. Boutillon, D. Gnaedig, V. Gaudet, P. G. Gulak, and M. Jezequel, "On multiple slice turbo codes," 3rd International Symposium On Turbo Codes and Related Topics", 2003.

[7] J.C. MacKay David and R.M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics letters*, July 1996.

[8] I. Gutierrez, A. Mourad, J. Bas, S. Pfletschinger, G. Bacci, A. Bourdoux, H. Gierszal, "DAVINCI Non-Binary LDPC codes: Performance and Complexity Assessment", *proc of Future Network & Mobile Summit, Italy*, June 2010.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near-Shannon limit error-correcting coding and decoding: Turbo codes", *Proc. IEEE Int. Conf. Commun., vol.2*, 1993.

[10] C. Chavet, P. Coussy, E. Martin and P. Urard, "Static Address Generation Easing: a Design Methodology for Parallel Interleaver Architectures", *proc of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, March 2010.

[11] V. G. Vizing;On an estimate of the chromatic class of a p-graph(In Russian), Diskret.Analiz. 3: 25-30, 1964

[12] A.Tarable, S.Benedetto, and G.Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures", *IEEE Trans. Inf. Theory, vol.50, no.9*, Sept. 2004.

[13] A. Sani, P. Coussy, C. Chavet, and E. Martin, "An approach based on edge coloring of tripartite graph for designing parallel ldpc interleaver architecture," in Circuits and Systems (ISCAS), 2011 IEEE International Symposium on, may 2011.

[14] H. Moussa, A. Baghdadi, M. Jezequel, "Binary de Bruijn on-chip network for a flexible multiprocessor LDPC decoder", *45th DAC : design automation conference*, 2008.

[15] A.H. Sani, C. Chavet and P. Coussy, "A First Step Toward On-Chip Memory Mapping for Parallel Turbo and LDPC Decoders: A Polynomial Time Mapping Algorithm*", IEEE Transactions on Signal Processing, vol. 61, issue: 16, p.4127 - 4140,* 2013.

[16] G Wang, Y Sun, JR Cavallaro, Y Guo, "High-throughput contention-free concurrent interleaver architecture for multi-standard turbo decoder" Application-Specific Systems, Architectures and Processors (ASAP), 2011

[17] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," IEEE Trans. VLSI Syst., vol. 11, no. 6, Dec 2003.

[18] A. Briki, C. Chavet, P. Coussy and E. Martin, "A Design Approach Dedicated to Network-Based and Conflict-Free Parallel Interleavers", In *Proceedings of the 22th ACM Great Lakes Symposium on VLSI (GLSVLSI) 2012*, Salt lake City, USA, may 2012

[19] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," IEEE Trans. Inf. Theory, vol. 52, no. 3, pp. 1249–1253, Mar. 2006.

[20] O. Sanchez, S. ur Rehman, A. Sani, C. Jego, C. Chavet, P. Coussy, and M. Jezequel, "A dedicated approach to explore design space for hardware architecture of turbo decoders", Proc of the IEEE SiPS, Quebec, Canada, 2012

[21] C. Chavet and P. Coussy (Eds), "Advanced Hardware Design for Error Correcting Codes", Springer, ISBN 978-3-319-10569-7, 2014.