Chapter 9
Hardware Design of Parallel Interleaver Architectures: A Survey

Cyrille Chavet, Awais Hussain Sani, and Philippe Coussy

9.1 Motivation

Early developers of digital communication systems assumed that information could be transmitted through noisy channel with high reliability by increasing the signal to noise ratio. This could only be achieved at that time by increasing transmitted signal power enough to ensure that signal can reliably be transmitted. The revolutionary work of Shannon [1] changed this view by proving that it is possible to send digital data to receiver through noisy channel with high reliability by first encoding digital message with error correction code at transmitter and then subsequently decode it at receiver to generate original message.

The function of the encoder is to map X digits message into C digits codeword where $C > X$. The code rate $r = X/C$ defines the redundancy introduced by corresponding error correction code. Encoded message passes through channel which corrupts the message by adding some noise into it. At receiver, error correction decoder uses this added redundancy to determine the original message despite the noise introduced by channel. Typical communication chain is shown in Fig. 9.1.

Different error correction codes are introduced in literature. They can be classified into two broad categories: *block codes* and *convolutional codes*. In block codes, original information sequence is divided into different message blocks and each message is independently encoded to generate codeword bits whereas in convolutional codes, encoder takes information sequence as a continuous stream

C. Chavet (✉) • P. Coussy
Lab-STICC laboratory, Université de Bretagne Sud, 4 Rue Jean Zay, 56100 Lorient, France
e-mail: cyrille.chavet@univ-ubs.fr; philippe.coussy@univ-ubs.fr

A.H. Sani
SATT Ouest Valorisation, 14C rue du Pâtis Tatelin Métropolis 2, CS 80804,
35708 Rennes Cedex 7, France
e-mail: awais-hussain.sani@ouest-valorisation.fr

C. Chavet, P. Coussy (eds.), *Advanced Hardware Design for Error Correcting Codes*, DOI 10.1007/978-3-319-10569-7_9

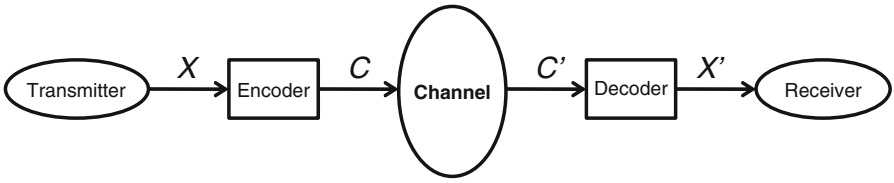

Fig. 9.1 Communication system

and generates a continuous stream of codeword bits. Therefore in block codes, encoder must wait for the entire message block before it starts encoding whereas convolutional encoder can start encoding and transmitting codeword before it obtains the entire message.

Many types of block codes are used in different applications but among the classical ones, Reed–Solomon [2] is the most popular due its widespread use in CD, DVD, and hard disk drives. Other examples of classical block codes are Golay codes [3] and Hamming codes [4]. Low density parity check codes (LDPC) is a class of linear block codes with error correction capabilities very close to the channel capacity. Due to their excellent error correction performance, it has already been included in several wireless communication standards such as DVB-S2 and DVB-T2 [5], WiFi-*IEEE 802.11n* [6], or WiMAX-*IEEE 802.16e* [7].

Convolutional codes, such as Turbo codes [8], perform like a finite state machine which converts continuous stream of X message bits into continuous stream of C coded bits (where $X > C$). Due to their simple structure and efficiently implementable iterative decoding algorithm, convolutional codes are increasingly used in different telecommunication standards. Thanks to their excellent error correction capabilities, Turbo codes are part of current telecommunication standards such as [9, 10] and digital broadcasting [11]. These Turbo codes are constructed through the parallel concatenation of two convolutional codes to achieve good error correction performance. Their outstanding performances are also possible due to the presence of pseudo-random interleaver that scrambles data to break up neighborhood relations.

Meanwhile, the large acceptance of smart-phones, laptop, digital television, and mobile broadband devices leads to the era of high data rate wireless applications. The rapid and huge increase in data traffic strains network capacity and researchers are developing new techniques to cope with this high throughput requirement. As a result of this effort, advanced technologies such as OFDM, MIMO, and advanced error correction techniques are included in different standards to reliably transfer data at high rates on wireless networks.

However, the excellent performance of error correction codes comes at the expense of computational complexity. Hence, parallel architectures must be employed to speed up the decoding process and support required application throughputs. Moreover, several parameters such as scheduling, parallelism level, memory organization, and network architecture need to be explored to trade

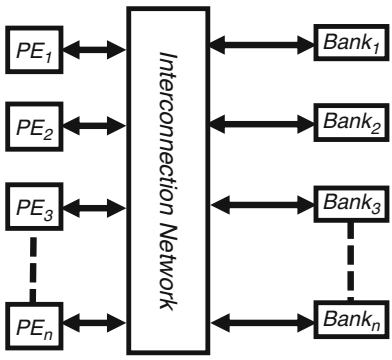

Fig. 9.2 Decoder parallel architecture

off circuit area and performances. This requires the development of dedicated approaches to efficiently implement decoder architecture. In such implementation (cf. Fig. 9.2), several processing elements (PEs) are used to obtain the required throughput. Memory is used to store different messages generated during the decoding process. These messages are written into and read out of the memory according to particular permutation defined by a permutation law. This architecture, however, suffers from memory access collision problem when more than two processing elements want to access the same memory bank. Collision problem becomes a significant issue with the increase of code word length and is discussed in the next section. Moreover, with the growing demand of high data rate applications and shrinking time to market constraint, this problem proves to be one of the most problematic factors in designing efficient decoder architectures.

To manage this problem, conflicts can be resolved either during definition of interleaving law or at run time or at design time. Designing conflict free interleaving law often simplifies the construction of parallel decoder architectures. However, it traditionally only supports particular parallelism used in decoding algorithms (e.g., LTE only supports SISO decoder level parallelism for a subset of block lengths). Managing conflict problems at runtime (e.g. serializing/postponing conflicting accesses through buffers) results in additional hardware cost and delay, and may be less interesting for high data rate and low power applications. In order to resolveconflict problem for any type of parallelism and interleaving law, design time conflict resolution is another solution. Here, conflict-free memory mappings are found off-chip either manually or automatically. In manual approach, the designer finds the conflict-free memory mapping after analyzing the interleaving law and then designs a controller using FSM controllers. However, in automatic approach different dedicated algorithms are developed and run on computer to generate ROM-based controllers. In order to support multiple block lengths and standards on a single chip, automated approaches require to pre-calculate memory mapping for each block length and to store them on-chip which results in large memory footprint. More recently, a new kind of approach has been proposed based on a hybrid strategy that aims to benefit both from runtime and design time approaches through on-chip memory mapping. The idea is to generate on-chip conflict free memory mappings, during the execution of the application.

9.2 Problem Formulation

First, we present a problem formulation based on an example based on access order of turbo decoder. However, it has been already been shown in [12–15]... that the problem is the same for LDPC codes.

In parallel decoder architectures, several processing elements *PEs* are concurrently used to decode the received information. In order to increase memory bandwidth, several memory banks *Bs* are connected with these *PEs* through a dedicated interconnection network (see Fig. 9.2). This network exchanges data between *PEs* and *Bs* according to predefined access orders. These orders are parameterized by block lengths and *PEs* parallelism.

Typical parallel decoder architecture is shown in Fig. 9.2. In our example, natural and interleaved orders are defined as follow:

Natural order = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
Interleaved order = {0, 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 11}

For parallel processing, this codeword is divided into four windows in both natural and interleaved order and arranged in data access matrices of Fig. 9.3. In this figure, each row (or window) is processed by one processing element whereas data in each column (or time instance) need to be accessed concurrently.

To increase memory bandwidth, three memory banks are used so that processing elements can concurrently get data elements in parallel. Data elements must be stored in banks in such a manner that at each time instance in natural order, all the processing elements always access different memory banks as shown in Fig. 9.4a. However by using this memory mapping, all processing elements always access the same memory bank at each time instant in interleaved order as shown in Fig. 9.4b. This results in memory conflict problem [16] and increases latency and thus reduces system throughput and increases system cost.

To solve this memory conflict problem, several approaches can be proposed in order to *manage concurrent parallel access to all the data elements in both read and write accesses with or without any conflict and/or with or without dedicated additional hardware mechanism*.

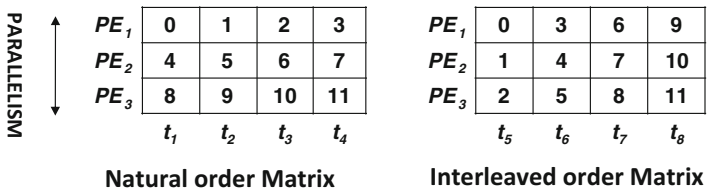

Fig. 9.3 Data access matrices

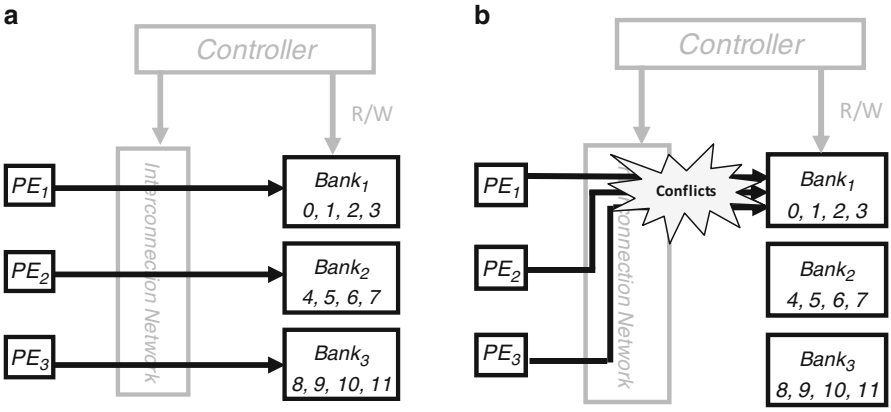

Fig. 9.4 Memory conflict problem in parallel turbo decoder. (**a**) Conflict-free natural order access. (**b**) Conflict-full interleaved order access

9.3 An Overview of Memory Access Conflict Solving Approaches

In recent standards, different conflict free interleaving laws have been defined. For example, 3GPP-LTE uses quadratic permutation polynomial (QPP) interleaver [17] whereas WiMAX [7] uses ARP [18] interleaver to permute the data. These interleavers often simplify the parallel decoder architecture. However, they are conflict-free only for particular types (e.g., [19] or [20]) or degrees of parallelism used in turbo decoding or for a subset of block lengths. Hence, several solutions are proposed in literature to solve such conflicts at runtime or at design time. At runtime, architectures use routing and/or buffering techniques in the interconnection network to serialize conflicting accesses. Design time approaches are able to generate in-place memory mapping (i.e., a given data is stored in one and only one memory bank, and one memory address) in order to reduce the cost of the controller. Some other approaches try to strongly optimize the controllers by moving a data from one memory place to another between each access to this data.

9.3.1 Conflict Solving During the Definition of the Interleaving Law

A first class of approach consists in defining conflict free interleaving law. An example of such solution is proposed in [21] based on Turbo-codes. In this approach spatial and temporal permutations of data are introduced to construct conflict-free interleaver with random interleaver like properties. Consider a block length of n data arranged row by row into a matrix M. Interleaver function is the sum of both

temporal and spatial permutations of the lines and columns of M. The benefit of this approach is that one can use barrel shifter interconnection network to realize turbo decoder for this interleaving law in practical applications. However, the approach is dedicated, i.e., not standard compliant.

More recently, [17] has proposed the QPP interleaver architecture. The authors show that QPP interleaver is maximum contention-free, i.e., for every window size W, which is a factor of the interleaver length N, the interleaver is contention free. QPP interleaver is defined by the following equation:

$$\Pi(\mathrm{x}) = \left(f_1 x^2 + f_2 x\right) \mod N$$

where x and $\Pi(\mathrm{x})$ represent the original and interleaved address respectively, and integers f_1, f_2 are different for different block lengths and can be found in the standard.

This kind of approach has been recently used in standard like LTE. However, QPP contention-free property is true for SISO decoder level parallelism only. For higher data rate applications when trellis and recursive units parallelism are also included in each SISO, QPP interleaver is no more contention-free and requires additional router and buffer mechanisms to manage problems. Moreover, having no conflict for QPP interleaver allows to design conflict-free architecture in 3GPP LTE decoder only: it indeed results in designing a channel interleaver that has to manage memory conflicts in order to present data to QPP in the required organization.

From LDPC point of view, these codes are completely specified by their parity matrices. These matrices represent how data (named *variable nodes* in LDPC) must be processed by the processing elements (named *check nodes* in LDPC) in order to achieve good error correction performances. Hence, proper construction of such matrices is necessary to obtain excellent error correction capabilities of LDPC. Different constraints can be added during the construction of parity matrices either to achieve significant coding gains or to simplify the decoder architecture. The matrices can also be constructed is such a way that data transfer between check nodes (CNs) and variable nodes (VNs) is made without any conflict for partially parallel architecture [22, 23]. The codes obtained during such construction procedure are called *structured codes*.

Structured codes remove memory conflict problem because transfer of messages between (CNs) and (VNs) is carried out through simple rules (like indices permutation). Also, structured codes simplified the decoder architecture since interconnection network can be implemented through simple network (like barrel shifter) by exploiting the regularity introduced during code construction. Due to simplicity in construction, structured codes are part of current telecommunication standards; e.g. [6] or [7].

Although it is proved in [22] that performances of structured codes are very close to random codes, adding constraints to construct structured codes may degrade the code's decoding performance. Therefore, special attention should be taken while selecting constraints to develop structured codes to keep remarkable error correction capabilities of LDPC. Also, structured codes only support one class of LDPC codes

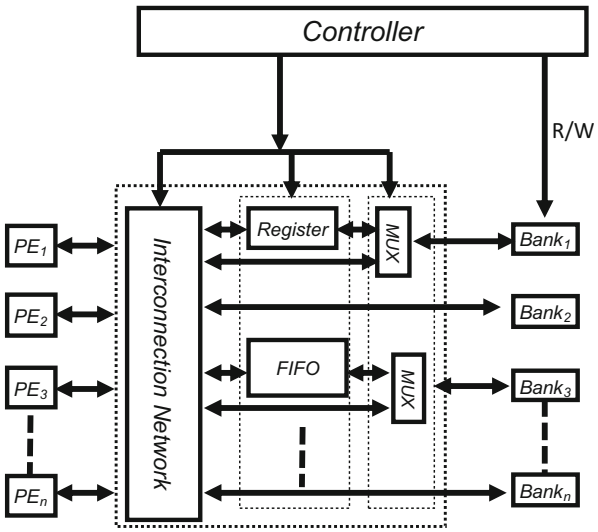

Fig. 9.5 Conflict buffering mechanism added to the network architecture

and to handle diverse existing and future classes of LDPC codes (such as no-binary LDPC codes), a general approach to handle memory mapping problem is required. Finally, it must be noticed that if the data access order can be defined to be conflict-free in the decoder part of the architecture; this supposes that this decoder can be fed with data in a proper order, which can differ from the order of data coming from the channel. Hence, in this case (like for QPP interleavers) the problem is only moved from inside the decoder up to its interface, i.e., its environment.

9.3.2 Conflict Solving Through Dedicated Runtime Approaches

A second class of solution to deal with memory access conflict problem is to simply store data elements in different memory banks without considering conflicting accesses and then use additional buffers in the interconnection network to manage conflicts at runtime [24] see Fig. 9.5.

In [25] the authors propose to add in the interconnection network a dedicated Double-Buffer based Contention-free (DBCF) to design a HSPA+ decoder. This architecture is configured, thanks to the statistical property of the memory conflict based on simulation results analysis. As soon as DBCFs detect conflicts, the conflicting accesses are routed into a dedicated circular buffer (see Fig. 9.6).

Another example of conflict-solving oriented architecture can be found in [26]. In this approach, the interconnection network is based on a Network-on-Chip that can be configured on-the-fly to emulate any "classical" interconnection network such

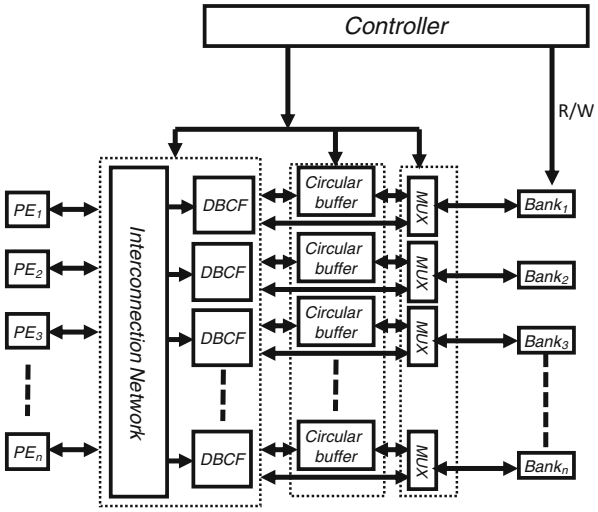

Fig. 9.6 Architecture based on DBCF

like Butterfly, Benes, De Bruijn, cross-bar… Then, if a memory conflict access is detected by the routers, then only one access will be performed to the memory banks, and the other conflicting ones will be re-routed into the network. Then, these conflicting "packets" could re-try to access the memory banks later on (see Fig. 9.7).

Runtime approaches generally increase the cost and latency of the system due to presence of interconnection network and buffer management mechanism to manage conflicts. The total latency of the system is also increased since each conflicting data access must travel buffers before being stored in the memory banks. Hence, such approaches are also often referred as *time relaxation* since additional cycles can be used to solve memory access conflicts.

9.3.3 Conflict Solving Through Dedicated Memory Mapping Approaches

A third class of solution to deal with memory access conflict problem is to store data elements (mapping process) in different memory banks so that all the Processing Elements (PEs) can access the data without any conflict. Dedicated mapping algorithms are used to perform some pre-processing steps to determine the memory locations for each data element used.

These algorithms can be categorized as (1) *unconstrained*, i.e., the targeted interconnection network supports any permutations (e.g., Benes networks, cross-bar…), (2) *hard-constrained*, i.e., the designer wants to use a cheaper (from architecture point of view, e.g., a barrel-shifter) interconnection network, but at

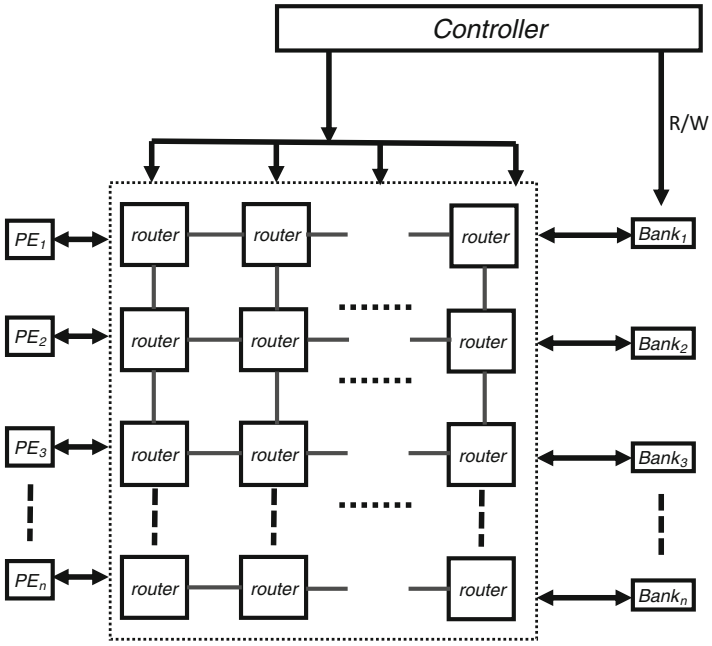

Fig. 9.7 NoC-based architecture

the expense of a strong limitation of the set of possible permutations, and (3) *soft-constrained*, i.e., the architecture can be modified during the memory mapping step in order to reduce its final complexity.

9.3.3.1 Unconstrained Memory Mapping Approaches

Several unconstrained mapping approaches are proposed in state of the art to find a memory mapping that will be natively conflict free; i.e., each processing elements can access all its data without any conflict at each time instance [13–15, 27]...

Simulated annealing approaches like [13] (in-place memory mapping only) or [14, 15] (in-place memory mapping and multi-read/multi-write memory mapping) are able to find a conflict-free memory mapping for both Turbo-codes and LDPC. In order to apply the proposed approach on LDPC, the authors of [13] proposed to modify memory access schedule by using a static single assignment (SSA) form which results in oversized memory architecture to store data since each data access is stored in a dedicated memory address. However, in [14, 15] the authors removed this limitation by using several memory locations for each data and by sharing this memory location among multiple data.

In [27] authors take leverages of Bipartite Edge Coloring techniques to solve the mapping problem for both turbo and LDPC codes in polynomial time. Hence,

they first transform data access matrices for Turbo Codes and mapping matrices for LDPC into bipartite graphs. Afterwards, bipartite edge coloring algorithm is applied on these graphs to solve mapping problem. Since edge coloring algorithm is always able to color the edges of bipartite graph with minimum colors, therefore it is always possible to solve memory mapping problems for both turbo and LDPC codes in polynomial time.

However, these approaches store data "randomly" in memory banks. Parallel interleaver architectures could thus be significantly optimized in terms of interconnection network and memory controller costs. Indeed no regularity can be easily extracted from the control words generated by the memory controller, and as a consequence, no optimization can be efficiently performed. Even if the designer would prefer to use an optimized network (e.g., Butterfly, Barrel-shifter...), if no dedicated memory mapping approach is proposed then the resulting interconnection network must be a full crossbar or a Benes network. Hopefully, some approaches are able to find conflict free memory mapping that is fully compatible with a user-defined interconnection network.

9.3.3.2 Hard-Constrained Memory Mapping Approaches

In [14, 15], an approach called Static Address Generation Easing—SAGE—is presented that considers a target interconnection network to find memory mapping. In this approach, two empty matrices called SAGE Mapping Matrices are used to store banks information during algorithm execution. To find architecture-oriented memory mapping, two constraints are defined to be respected during algorithm execution. First, each column of the mapping matrices (see Sect. 9.2) should contain different memory banks and second, if the interleaving law allows, each column should respect the rules of the steering network component. This approach guaranties if the target interconnection network is compatible with the interleaving law, then the final memory mapping will respect it.

In [28], an approach based on transportation problem modeling which finds conflict-free memory mapping for every type of turbo codes and which optimizes the resulting interleaving architecture has been proposed. The mapping problem for turbo codes is transformed as transportation problem by considering all the data nodes as producers and all the time nodes as consumers. The main interest of this approach is that the designer is able to obtain conflict-free memory mapping in polynomial time and this mapping respects the targeted interconnection network, if it is compatible with the interleaving law.

The main weakness of this kind of approaches is that they are limited to one targeted interconnection network defined at design time. Moreover, they cannot be applied to any data permutation approaches since they are limited to Turbo-codes based systems.

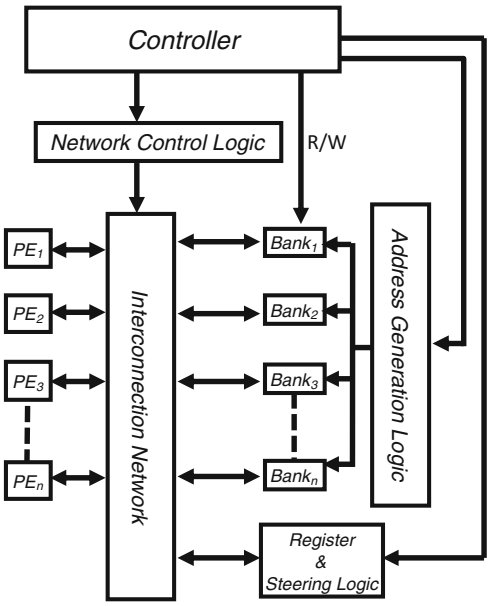

Fig. 9.8 Architecture generated by memory relaxation approach

9.3.3.3 Soft-Constrained Memory Mapping Approaches

The last class of approaches tries to take advantage of both runtime approach mechanisms and design time approach efficiency. In previous mapping approaches, the generated memory mappings induce sets of control words in order to control the memories and the network of the decoder architecture. If no regularity can be extracted from these control words no optimization can be performed. On the contrary, if such regularity can be extracted, the addressing sequence, and the associated controller cost can be greatly reduced.

Such regularity can be obtained by applying the approach described in [12]: in this solution, additional registers are used to store conflicting data but also some non-conflicting data if and only if this enables to simplify the memory controller architecture (see Fig. 9.8). This approach, referred as *Memory relaxation* (i.e., additional memory elements—registers or FIFOs—can be allocated to remove conflicts or to enable strong optimization of the controllers) is also able to generate a conflict-free memory mapping with respect to a target interconnection network (Barrel-Shifter, Butterfly...). However the final architecture suffers from hardware overhead due to additional registers and their dedicated additional steering logic. This overhead depends on the compatibility between interleaving law and permutation characteristics on the targeted interconnection network. For higher incompatibilities, the approach results in higher hardware overhead and latency.

Since interconnection network also impacts the cost of the architecture [28], then a smart memory mapping approach could also focuses on optimizing this network in order to adapt its structure to the interleaving law.

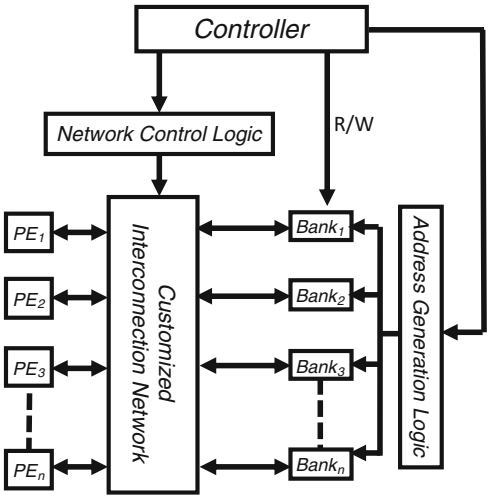

Fig. 9.9 Architecture generated by network relaxation approach

Ur Rehman et al. [29] presents a mapping approach that considers the customization of the interconnection network and reduces the cost of the controller architecture. This approach, referred as *Network relaxation*, modifies the original network by adding additional multiplexers/switches (see Fig. 9.9). The mapping step aims to fully explore the memory mapping solution space by checking all the permutations of the selected network. If no memory mapping solution exists for this network, then the set of permutations will be extended by adding a steering component which results into a customized network architecture with enriched set of permutations.

This approach proved to be the most efficient (compared to state of the art) in terms of hardware cost and latency. However, this approach, like the other ones from the literature, generates a static architecture that cannot be modified on-the-fly depending on system requirements (QoS, application switching, changing application parameters...).

9.3.4 Hybrid Approach: On-chip Memory Mapping Approach

In parallel decoder architectures, design time approaches require storing into ROM the control words to drive the network and to address data memories for particular block length or/and interleaving law. So, in order to design flexible decoder architectures that support multiple block lengths and multiple interleaving laws, several ROMs are needed (see Fig. 9.10) which results in an important hardware overhead.

In order to be flexible at a reasonable hardware cost (i.e., reduced memory footprint), a solution where the mapping algorithm is run on chip has been proposed.

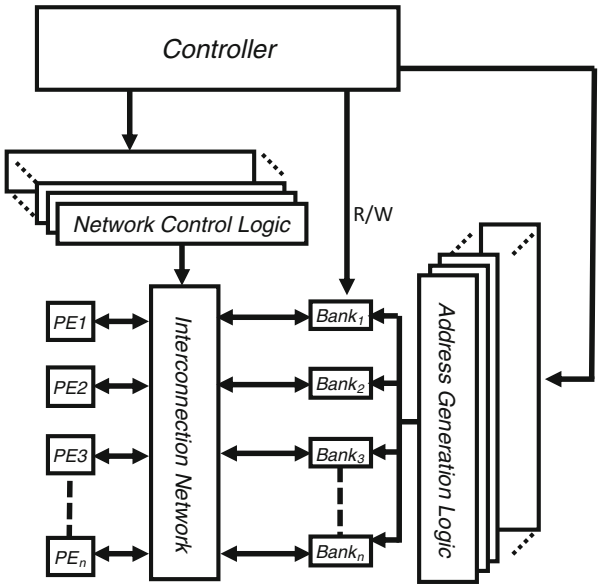

Fig. 9.10 Parallel decoder architecture supporting multiple block lengths

The approach starts by computing new mapping information on the fly as soon as a new block length needs to be decoded and updates these new generated control information in the memory.

Since this approach requires a fast on-chip mapping algorithm, then a novel polynomial time algorithm [29] derived from [30] is used for on-chip implementation. Low computational cost of this algorithm enables memory mapping approaches to implement this algorithm on chip using embedded processor, an ASIP or a dedicated hardware accelerator and to generate network and addressing control bits on-chip. This approach supports multiple standards and block lengths on single chip with reduced memory footprint.

The hardware architecture for embedded memory mapping is shown in Fig. 9.11. Control unit includes a dedicated processing element (General Purpose Processor GPP, Application Specific Instruction set Processor ASIP or Application Specific Integrated Circuit ASIC) to execute the mapping algorithm. The multiple networks and addressing ROMs are replaced by two RAMs, i.e., *Network RAM* and *Addressing RAM*. Control Unit executes the mapping algorithm and updates RAMs if required as soon as the decoder requires decoding a new block length or a new application (e.g., LTE, HSPA+, Wifi).

The execution flow is shown in Fig. 9.12. In the first step, the data access order is generated based on the particular interleaving law along with other required input parameters like block length, parallelism, and scheduling. This first step can be avoided if the designer wants to feed the system with pre-computed data consumption orders. This data access order is simply a scheduling of data accesses.

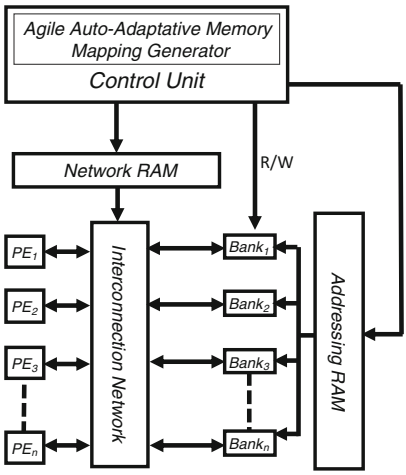

Fig. 9.11 Parallel decoder architecture to embed memory mapping algorithms on chip

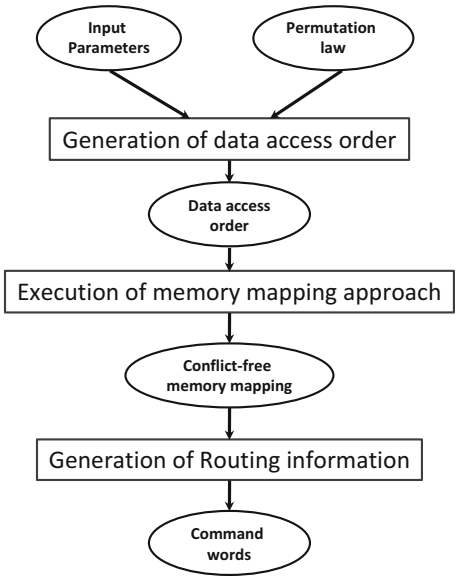

Fig. 9.12 Execution flow

However, the conflict-free memory mapping still need to be generated which is the goal of the second step of the design flow. Finally, the last step generates the routing and control information for the interconnection network and the memory banks.

Low computational cost of the mapping algorithm enables to implement this algorithm on chip using embedded processor, an ASIP or a dedicated hardware accelerator and to generate network and addressing control bits online. This approach will enable designers to support multiple standards and block lengths on single chip with reduced memory footprints.

The significant reduction in execution time and area obtained using this approach encourages embedding memory mapping and routing algorithm in future telecommunication devices.

Conclusion
In this chapter, we have presented a survey of existing solutions to deal with memory conflict accesses in parallel hardware decoder architectures of Turbo-codes and LDPC. The first type of approaches is based on definition "natively conflict-free" interleaving laws. A second family of approaches proposes to use dedicated hardware mechanisms to deal with conflicts at runtime, but with the expense of additional hardware components and latency. The third family of solutions proposes to find conflict-free memory mappings at design time, but at the expense of important memory footprint and hardware overhead when designing flexible decoders. Finally, a new class of approach that runs mapping algorithm on chip has been presented. In this context, a polynomial time memory mapping approach and a dedicated routing algorithm are embedded in the decoder architecture.

References

1. Shannon C (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423, 623–656
2. AHA. Reed-Solomon error correction codes (ECC). ANRS01-0404
3. Golay MJ (1961) Complementary series. IRE Trans Inf Theory IT-7:82–87
4. Hamming RW (1950) Error detecting and error correcting codes. Bell Syst Tech J XXVI(2):147–160
5. DVB (2008) Frame structure channel coding and modulation for the second generation digital terrestrial television broadcasting system (DVB-T2). DVB Document A122
6. WIFI (2008) Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: enhancements for higher throughput. IEEE P802.11n/D5.02, Part 11
7. WiMAX (2006) 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
8. Berrou C, Glavieux A, Thitimajshima P (1993) Near Shannon limit error-correcting coding and decoding: turbo-codes. In: Proceedings of the IEEE international conference on communications (ICC 93), Geneva, pp 1064–1070
9. HSPA (2004) Technical specification group radio access network; multiplexing and channel coding (FDD). 3GPP, 25.212 V5.9.0
10. LTE (2008) Technical specification group radio access network; evolved universal terrestrial radio access; multiplexing and channel coding (release 8). 3GPP Std. TS 36.212
11. DVB-SH (2008) 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
12. Briki A, Chavet C, Coussy P (2013) A conflict-free memory mapping approach to design parallel hardware interleaver architectures with optimized network and controller. In: IEEE workshop on signal processing systems, Tapei

13. Tarable A, Benedetto S, Montorsi G (2004) Mapping interleaving laws to parallel turbo and LDPC decoder architectures. IEEE Trans Inf Theory 50(9):2002–2009
14. Chavet C, Coussy P (2010) A memory mapping approach for parallel interleaver design with multiples read and write accesses. In: IEEE international symposium on circuits and systems, Paris, pp 3168–3171
15. Chavet C, Coussy P (2010) Static Address Generation Easing: a design methodology for parallel interleaver architecture. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 1594–1597
16. Giulietti A, van der Perre L, Strum M (2002) Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements. Electron Lett 38(5):232–234
17. Takeshita OY (2006) On maximum contention-free interleavers and permutation polynomials over integer rings. IEEE Trans Inf Theory 52(3):1249–1253
18. Berrou C, Saouter Y, Douillard C, Kerouédan S, Jézéquel M (2004) Designing good permutations for turbo codes: towards a single model. In: Proceedings of the IEEE international conference on communications (ICC 2004), pp 341–345
19. Woodard JP, Hanzo L (2000) Comparative study of turbo decoding techniques: an overview. IEEE Trans Veh Technol 49:2208–2233
20. Blankenship BC (2005) High-throughput turbo de-coding techniques for 4G. In: International conference on 3G wireless and beyond, pp 137–142
21. Gnaedig D, Boutillon E, Jezequel M, Gaudet VC, Gulak PG (2003) On multiple slice turbo codes. In: Proceedings of 3rd international symposium on turbo codes, pp 343–346
22. Mansour M, Shanbhag N (2003) High-throughput LDPC decoders. IEEE Trans VLSI Syst 11(6):976–996
23. Zhang T, Parhi KK (2001) Joint code and decoder design for implementation-oriented (3;k)-regular LDPC codes. Asilomar Conf Signals Syst Comput 2:1232–1236
24. Thul MJ, Gilbert F (2002) Optimized concurrent interleaving architecture for high-throughput turbo-decoding. In: Ninth international conference on electronics, circuits and systems, vol 3, pp 1099–1102
25. Wang G, Sun Y, Cavallaro JR, Guo Y (2011) High-throughput contention-free concurrent interleaver architecture for multi-standard turbo decoder. In: IEEE international conference on application-specific system, architectures and processors (ASAP) pp 113–121
26. Moussa H, Muller O (2007) Butterfly and Benes-based on-chip communication networks for multiprocessor turbo decoding. In: Design, automation and test in Europe, pp 654–659
27. Sani AH, Coussy P, Chavet C, Martin E (2011) An approach based on edge coloring of tripartite graph for designing parallel LDPC interleaver architecture. In: IEEE international symposium on circuits and systems
28. Sani AH, Coussy P, Chavet C, Martin E (2011) A methodology based on transportation problem modeling for designing parallel interleaver architectures. In: Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing, Prague, May 22-27, 2011
29. Ur Rehman S, Sani A, Chavet C, Coussy P (2014) Embedding polynomial time memory mapping and routing algorithms on-chip to design configurable decoder architecture. In: Ninth IEEE international conference on acoustics, speech and signal processing
30. Sani AH, Chavet C (2013) A first step toward on-chip memory mapping for parallel turbo and LDPC decoders: a polynomial time mapping algorithm. IEEE Trans Signal Process 61(16):4120–4140