Parallel Interleaver Architecture with New Scheduling Scheme for High Throughput Configurable Turbo Decoder

Guohui Wang*, Aida Vosoughi*, Hao Shen*, Joseph R. Cavallaro*, and Yuanbin Guo[†]

*Department of Electrical and Computer Engineering, Rice University, Houston, Texas 77005

[†]Wireless R&D, US Research Center, Futurewei Technologies, 5340 Legacy Dr., Plano, Texas 75024

Email: {wgh, av26, hs9, cavallar}@rice.edu, yuanbinguo@huawei.com

Abstract—Parallel architecture is required for high throughput turbo decoder to meet the data rate requirements of the emerging wireless communication systems. However, due to the severe memory conflict problem caused by parallel architectures, the interleaver design has become a major challenge that limits the achievable throughput. Moreover, the high complexity of the interleaver algorithm makes the parallel interleaving address generation hardware very difficult to implement. In this paper, we propose a parallel interleaver architecture that can generate multiple interleaving addresses on-the-fly. We devised a novel scheduling scheme with which we can use more efficient buffer structures to eliminate memory contention. The synthesis results show that the proposed architecture with the new scheduling scheme can significantly reduce memory usage and hardware complexity. The proposed architecture also shows great flexibility and scalability compared to prior work.

Index Terms—VLSI, turbo decoder, parallel interleaver, HSPA+, high throughput, contention-free, parallel processing, memory conflict.

I. INTRODUCTION

Recently, rapidly increasing demand for high quality data services is leading to the deployment of Evolved High-Speed Packet Access (HSPA+) and Long Term Evolution (LTE) mobile communication systems. Although LTE can provide higher data rates than HSPA+, the cost of the LTE infrastructure is high; in contrast, existing infrastructures need only a simple upgrade to support the features defined in HSPA+. Therefore, HSPA+ and LTE are expected to co-exist for a long term, especially when HSPA+ keeps evolving. For example, the most recent HSPA+ standard [1] supports up to 337.5 Mbps data rate for downlink, and up to 672 Mbps has been proposed for the future 3GPP extension that uses advanced antenna techniques [2]. Due to the above reason, the configurable multi-standard devices are in high demand. In addition, as the LTE and HSPA+ standards evolve, the higher data rate requirements pose significant challenges to future system design and VLSI implementations.

Turbo decoder is one of the most important blocks in 3G/4G wireless communication systems. Turbo codes are a class of error-correcting codes which are widely used in many 3G/4G communication systems due to outstanding error-correcting performance [3]. To boost the decoding throughput, many parallel turbo decoding algorithms have been investigated in the past several years [4, 5, 6]. As is shown in Fig. 1, P soft-input soft-output (SISO) decoders work in parallel to decode a codeword with block size K , with each SISO decoder processing a sub-block with K/P bits. After being processed, the data are stored in M memory modules. In a parallel decoder architecture, a parallel interleaver is required to generate multiple interleaving addresses per clock cycle, which are then used by parallel data streams. However, if two or more SISO decoders access the same memory module, a memory conflict occurs. Due to the memory conflict problem caused by the randomness property of the interleaving algorithm, parallel interleaver has become the most challenging part in the entire design process of parallel turbo decoders. HSPA+ turbo interleaver suffers severely from memory

Fig. 1. Parallel turbo decoder architecture.

conflicts. For LTE, although the Quadratic Permutation Polynomial (QPP) interleaver is designed to eliminate memory conflicts for parallel architecture, it still has some limitations to the level of parallelism. Moreover, for certain turbo decoder architectures such as Radix-4, even the QPP interleaver is not fully contention-free.

In this paper, we propose to solve the aforementioned problem. This paper is organized as follows. Section II describes the major challenges to design and implement interleavers for a high throughput parallel turbo decoder. In Section III, we propose a new scheduling scheme for the interleaving systems that can significantly reduce the complexity of the buffer structure and control logic while solving the serious memory conflict problem. In addition, a new parallel interleaver architecture with low hardware complexity is proposed to allow efficient data scrambling. Section IV presents the VLSI implementation of the proposed architecture and comparison with related work. Finally, Section V concludes this paper.

II. CHALLENGES OF PARALLEL INTERLEAVER

There are two major challenges to implement a parallel interleaver for a high throughput parallel turbo decoder: (1) reducing memory usage, and (2) solving the memory conflict problem.

A. Reducing Memory Usage for Interleaving Address Generation

The traditional interleaver stores all the interleaving patterns in a look-up table (LUT) in the memory. As the demand for multi-standard and configurable turbo decoders grows, the LUT-based interleaving address generation (IAG) methods have become impractical for VLSI implementations due to the huge amount of memory usage. For instance, the valid block size of turbo codes in the HSPA+ standard is 40 ~ 5114. To provide interleaving address patterns for all block sizes, several hundred Mbits of memory is required to implement the LUT. Even with compression techniques applied, the required memory size is still too large for a SOC architecture. Not to mention that many modern modems require the ability to support multiple standards with different interleavers.

On-the-fly IAG uses computation units to compute the interleaving patterns instead of storing them in the memory. Many research papers proposed on-the-fly IAG hardware implementations [6, 7, 8, 9, 10, 11]. However, most of them focused on non-parallel interleaver architectures. Moreover, they did not provide efficient hardware

implementations which take advantage of hardware sharing for the highly parallel architecture. In this paper, we propose an efficient parallel interleaver architecture to enable high throughput and low-cost hardware supporting both interleaving and deinterleaving.

B. Solving Memory Conflict Problem

In a parallel turbo decoder depicted in Fig. 1, if more than one SISO decoder tries to access the same memory module, a memory conflict occurs. The ratio of memory conflicts increases exponentially as the parallelism goes higher [12, 13]. To solve the memory conflict problem, extra clock cycles need to be spent, which prevent the turbo decoder from achieving high data rate.

Numerous methods and system architectures have been proposed to solve the memory conflict problem in a parallel turbo decoder [5, 14, 12, 15, 16, 17]. However, there are several limitations of these approaches. First, most of them can only support parallel turbo decoders with low parallelism degrees, such as 2 or 4. Second, to reduce memory conflicts they introduced complex buffer structures, which result in large chip area and complicated control logic. Furthermore, most of the previous works only focused on eliminating or reducing conflicts for memory writing, but few have investigated how to solve the memory reading conflict problem, which is even harder to solve and has worse effect on system throughput than memory writing conflicts. In addition, researchers in [12] mentioned that memory reordering may be used to solve the memory reading conflict problem, but the reordering logic will add too much overhead to the memory control blocks. To solve the aforementioned problems, we propose a new scheduling scheme which allows us to utilize an efficient buffer architecture proposed in our previous work [13].

III. PROPOSED ARCHITECTURE WITH NEW SCHEDULING SCHEME

In this section, we present two major contributions of this paper: a new scheduling scheme for parallel turbo decoder and a parallel interleaver architecture.

A. New Scheduling Scheme

Since there is data dependency during trellis traversal in a turbo decoder, SISO decoders cannot proceed until the currently requested datum is ready. Therefore, frequent stalling of SISO decoders will significantly decrease the decoding throughput. In order to achieve high throughput, we should keep all the SISO decoders running at full speed without stalling. Fig. 2 demonstrates two cases showing how the memory conflict problem is solved using general memory conflict solvers, when three SISO decoders are trying to access (write or read) the same memory module (MEM1 in the figure).

For a memory writing conflict (Fig. 2-(a)), we can utilize buffer structures such as double-buffer contention-free (DBCF) architecture as a “write conflict solver” to cache the data accesses and to balance the memory access workloads among memory modules [13]. All the SISO decoders can still keep full-speed processing so that the overall high throughput can be guaranteed. In contrast, if three SISO decoders request data from memory in the same clock cycle (Fig. 2-(b)), only SISO decoder-1 can get requested data1 immediately. But SISO decoder-2 must wait for a clock cycle to get data2 and SISO decoder-3 must wait for two clock cycles to get data3. Although buffering and data prefetching may help reduce the memory reading conflicts, it is difficult to completely eliminate memory conflicts and the hardware complexity of the complicated buffer controller will be very high. In addition, the read conflict solver and the writing conflict solver are so different that implementing both of them utilizes even more hardware resources and requires extra multiplexing logic.

Fig. 2. (a) Memory conflict solution when SISO decoders write data to the same memory MEM1; (b) Memory conflict solution when SISO decoders read data from the same memory MEM1. Multiple memory modules are used to store SISO decoders' output data. Here, only MEM1 is drawn for simplicity since other memory modules are not accessed in the cases shown above.

Fig. 3. (a) Original turbo decoder scheduling; (b) Proposed new scheduling scheme for parallel turbo decoder. (The numbers in the circles indicate the order of scheduling steps.)

To solve the above-mentioned problem, we propose a new scheduling scheme for turbo decoding. Fig. 3-(a) shows the original scheduling method in a traditional turbo decoder [3]. In 1st half iteration, the SISO decoder reads/writes contiguous data from/to the memory in a natural order. In the 2nd half iteration, the SISO decoder first reads data from the memory in an interleaved way; after computation, the SISO decoder writes the output data into the memory in a deinterleaved way. Due to the randomness of the interleaving/deinterleaving algorithm, both memory reading and writing suffer from memory contention in a parallel turbo decoder.

Fig. 3-(b) shows our new scheduling scheme in which we write the data in the interleaved order in the 1st half iteration so that in the 2nd half iteration we can read all the data in a natural order. In the new scheduling scheme, all the memory reading are in-order access so there is no memory conflict at all. The benefits of this new scheduling scheme are three-fold. First of all, the memory writing conflict is easier to solve than the memory reading conflict. We can use efficient buffer structures such as DBCF to solve the memory writing conflict in both half iterations while maintaining high throughput [13]. Second, without memory reading conflict, we can remove the read conflict solver hardware from the interleaver. This simplifies the hardware design and reduces the hardware resource requirement. Third, the new scheduling method generates a more balanced workload between two half iterations. Since both the read and write conflict solvers introduce extra penalty clock cycles, the original scheduling has more penalty clock cycles in the 2nd half iteration. In the new scheduling scheme, the symmetric structure of

Fig. 4. Architecture of parallel IAG. (Please note: “deintl” is short for “deinterleaver”.)

both half iterations results in more balanced workload, which makes the control logic easier to design.

It is worth mentioning that the new scheduling scheme is standard-independent, which means that multi-standard turbo decoders supporting HSPA+, LTE and other standards can benefit from this new scheduling scheme.

B. Proposed Parallel Interleaver Architecture

The on-the-fly parallel interleaving address generator (IAG) is one of the most challenging parts in a parallel turbo decoder. The new scheduling method requires the IAG hardware to compute interleaved addresses and deinterleaved addresses in two half iterations, respectively. We propose a unified parallel IAG architecture to support both interleaver and deinterleaver modes with low hardware overhead.

Fig. 4 shows our proposed parallel IAG architecture, which consists of the following key blocks: (1) preset parameter storage, (2) preprocessing unit, (3) run-time parameter storage, and (4) run-time IAG block. The preset parameter storage contains a few ROMs, which store parameters used in the preprocessing unit such as prime number table, permutation tables, and so on. The preprocessing unit computes the parameters and permutation patterns used in the run-time IAG block. To support the deinterleaver algorithm, extra preprocessing is needed except for the preprocessing logic for the interleaver algorithm, so the preprocessing unit includes two major parts: shared preprocessing logic which is used in both interleaver and deinterleaver modes, and deinterleaver preprocessing logic which is only needed for the deinterleaver. The preprocessing unit generates several run-time parameters based on the block size K (such as the number of rows/columns of the permutation matrix), and then stores them in a register file. The preprocessing unit also produces permutation patterns such as inter-row and intra-row permutations, which are stored in the run-time parameter storage (several RAMs). Finally, the run-time IAG block contains computation and control logic to convert the input addresses to the interleaved/deinterleaved addresses. Based on the input block size K and mode selection, run-time IAG can work in the corresponding mode.

It is worth noting that we only need one single copy of the preprocessing hardware. This can help us greatly reduce the hardware resources, since the preprocessing logic is of high complexity. The new scheduling scheme proposed in Section III-A makes it possible that the parallel interleaver can function in a time-division multiplexing manner. Therefore, hardware resources for two modes can be shared between two half iterations, which include the deinterleaver preprocessing unit, part of the control logic inside the run-time IAG, and a few specific ROMs and RAMs.

Fig. 5 shows the architecture of a complete parallel interleaver solution, combining the proposed parallel IAG and an efficient memory

Fig. 5. Overall architecture of parallel interleaver for parallel turbo decoding.

writing conflict solver called DBCF buffer [13]. This architecture can efficiently generate interleaving addresses on-the-fly and effectively solve the memory conflict problem using low complexity hardware for highly parallel turbo decoders. This architecture is independent of the type of SISO decoders, parallelism degrees and interleaver algorithms, so it has great flexibility and scalability.

IV. VLSI IMPLEMENTATION

We implemented an RTL model of the proposed parallel interleaver architecture using Verilog HDL. As a case study, the HSPA+ interleaver algorithm is implemented for the IAG [1]. Block size K is configurable. The interleaver can work at both interleaver and deinterleaver modes based on the mode selection input. The preprocessing unit uses 284 clock cycles on average to finish the preprocessing. However, the preprocessing is performed only once during initialization when a new block size is set, so the overhead for preprocessing can be neglected. As is shown in Fig. 3-(b), the IAG hardware works in interleaver mode in the 1st half iteration, and in deinterleaver mode in the 2nd half iteration.

A. Memory Usage

One of the major advantages of the proposed architecture is the reduced memory usage. A 32-byte register file is used to store parameters such as number of rows/columns of permutation matrix, prime numbers and primitive root values. Table I shows the breakdown of memory usage. Table II shows the trend of memory size as parallelism P increases (P is defined such that the parallel turbo decoder generates P soft output values in every clock cycle).

Type	Shared ROM	Deintl ROM	Shared RAM	Deintl RAM
Size (Bytes)	664	32	256	276

Parallelism P	1	2	4	8	16	32
Memory size (KBytes)	1.23	2.43	4.83	9.63	19.22	38.41

Fig. 6. Memory usage comparison between LUT-based architecture and proposed on-the-fly architecture.

Fig. 6 demonstrates that the proposed architecture uses much less memory than the LUT-based architecture for parallel turbo decoder. Please note that the proposed architecture supports all the valid block sizes (40 ~ 5114), while the LUT-based solution only supports one block size ($K = 5114$). To support all the valid block sizes, the LUT-based solutions will be infeasible for a hardware implementation due to the huge amount of memory usage.

B. Synthesis Results and Discussion

We synthesized the implemented RTL model using a TSMC 65nm technology library. The chip area of preprocessing unit is $5108 \mu m^2$ and the chip area of run-time IAG is $6679 \mu m^2$ when targeting at 700MHz clock frequency. Table III shows the synthesis results and estimated maximum throughput of the parallel IAG using the proposed architecture. The same Radix-4 XMAP SISO decoder as in [5, 13] is assumed to form a parallel turbo decoder. The throughput of this kind of turbo decoder can be computed using the following equation: $Throughput = \frac{K \times f_{max}}{(K/(2 \times P) + T_{SISO} + T_{intl}) \times 2 \times N_{iter}}$, in which K is block size, P is parallelism, f_{max} is the max frequency, N_{iter} is the number of iterations for turbo decoding, T_{SISO} is the latency of turbo decoder, and T_{intl} is the latency of the interleaver.

TABLE III
SYNTHESIS RESULTS OF PROPOSED PARALLEL IAG.

Parallelism P	Frequency f_{max}	Area [mm^2]	Throughput @ $N_{iter}=6$
4	700 MHz	0.040	116 Mbps
8	700 Mhz	0.076	230 Mbps
16	700 MHz	0.146	455 Mbps
32	700 MHz	0.287	889 Mbps

With $P = 32$, the maximum achievable throughput satisfies the throughput requirements of the potential future extension of HSPA+ standards (672 Mbps). Therefore, with our architecture, the bottleneck of the whole parallel turbo decoder will not be the parallel interleaver. It is worth noting that the proposed architecture is able to support even higher parallelism (> 32) if a higher throughput is required.

C. Comparison

In Table IV, our proposed architecture is compared with related work. We compare full interleaver solutions which contain IAG, buffer structures and control logic. Our work is a complete interleaver solution as is shown in Fig. 5, which combines the proposed parallel IAG architecture with the new scheduling scheme from this paper, and DBCF buffer architecture from our previous work [13].

TABLE IV
COMPARISON WITH RELATED WORK ON UMTS/HSPA+ INTERLEAVER.

Work	P	Support diff K	Dual-mode ^a	Tech	Area [mm^2]	Norm. Area ^b [mm^2]	Frequency f_{max}
[7]	1	Yes	No	250 nm	2.68	0.139	N/A
[4]	16	No	No	180 nm	8.19	0.067	200 MHz
[8]	1	Yes	No	180 nm	0.24	0.031	130 MHz
[11]	1	Yes	No	130 nm	0.4	0.102	246 MHz
[9]	1	Yes	No	FPGA	0.24 ^c	0.13	N/A
[10]	2	Yes	No	65 nm	0.014	0.007	150 MHz
[5]	16	No	Yes	40 nm	0.14	0.020	350 MHz
Ours	32	Yes	Yes	65 nm	0.491	0.015	550 MHz

^a This column indicates whether an implementation supports interleaver/deinterleaver dual-mode.

^b Chip area is scaled to 65nm tech and is normalized to equivalent area for one generated address.

^c The chip area is calculated using gate count for FPGA (90nm) reported in the paper.

To make the comparison fair, we show the normalized chip area for each implementation; furthermore, for the implementation supporting parallel interleaving address generation, the chip area is further divided by parallelism P to get the equivalent chip area for each generated interleaving address. The results in [7, 8, 9, 11] support on-the-fly interleaving address generation for HSPA+, but they do not support parallel decoding due to the lack of a memory conflict solver. The interleavers in [4, 5] are not reconfigurable because they support only one block size. Only [5] and our work support both interleaver and deinterleaver dual-mode. The architecture in [10] can support a parallel turbo decoder with different block sizes, but the architecture is limited to turbo decoders with low parallelism. This comparison shows that the proposed architecture has good hardware efficiency, great flexibility and good configurability.

V. CONCLUSION

In this paper, we present a parallel interleaver architecture with a new scheduling scheme for highly parallel turbo decoders. The proposed architecture can efficiently compute interleaving addresses on-the-fly and effectively solve the memory conflict problem using low complexity hardware for highly parallel turbo decoders. The synthesis results demonstrate the efficiency of the proposed parallel architecture. The implementation results also show great flexibility and configurability of the proposed architecture.

ACKNOWLEDGMENTS

This work was supported in part by Huawei and by the US National Science Foundation under grants EECs-1232274, EECs-0925942 and CNS-0923479.

REFERENCES

- [1] 3rd Generation Partnership Project, "Technical specification group radio access network; multiplexing and channel coding (FDD), Tech. Spec. 25.212 Release-11," 2012.
- [2] Nokia Siemens Networks, "Long term HSPA evolution. mobile broadband evolution beyond 3GPP release 10," 2010. [Online]. Available: http://www.nokiasiemensnetworks.com/sites/default/files/document/HSPA_evolution_white_paper_low_res_141220.pdf
- [3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 2, May 1993, pp. 1064–1070 vol.2.
- [4] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn, "A scalable system architecture for high-throughput turbo-decoders," *The Journal of VLSI Signal Processing*, vol. 39, pp. 63–77.
- [5] T. Ilseher, M. May, and N. Wehn, "A multi-mode 3GPP-LTE/HSDPA turbo decoder," in *IEEE International Conference on Communication Systems (ICCS)*, nov. 2010, pp. 336–340.
- [6] R. Asghar, D. Wu, A. Saeed, Y. Huang, and D. Liu, "Implementation of a Radix-4, parallel turbo decoder and enabling the multi-standard support," *J. Signal Process. Syst.*, vol. 66, no. 1, pp. 25–41, Jan. 2012.
- [7] M.-C. Shin and I.-C. Park, "Processor-based turbo interleaver for multiple third-generation wireless standards," *IEEE Communications Letters*, vol. 7, no. 5, pp. 210–212, may 2003.
- [8] Z. Wang and Q. Li, "Very low-complexity hardware interleaver for turbo decoding," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 7, pp. 636–640, July 2007.
- [9] H. Borrayo-Sandoval, R. Parra-Michel, L. Gonzalez-Perez, F. Printzen, and C. Feregrino-Urbe, "Design and implementation of a configurable interleaver/deinterleaver for turbo codes in 3GPP standard," in *International Conference on Reconfigurable Computing and FPGAs*, Dec. 2009.
- [10] R. Asghar, D. Wu, J. Eilert, and D. Liu, "Memory conflict analysis and interleaver design for parallel turbo decoding supporting HSPA evolution," in *Euromicro Conference on Digital System Design, Architectures, Methods and Tools.*, Aug. 2009, pp. 699–706.
- [11] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, jan. 2009.
- [12] F. Speziali and J. Zory, "Scalable and area efficient concurrent interleaver for high throughput turbo-decoders," in *Proc. Euromicro Symposium on Digital System Design (DSD)*, 2004, pp. 334–341.
- [13] G. Wang, Y. Sun, J. R. Cavallaro, and Y. Guo, "Concurrent interleaver architecture for high throughput multi-standard parallel turbo decoder," in *IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, September 2011, pp. 113–121.
- [14] A. Giulietti, L. van der Perre, and A. Strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements," *Electronics Letters*, vol. 38, no. 5, pp. 232–234, Feb. 2002.
- [15] C. Neeb, M. Thul, and N. Wehn, "Network-on-chip-centric approach to interleaving in high throughput channel decoders," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2005, pp. 1766–1769 Vol. 2.
- [16] C. Chavet and P. Coussy, "A memory mapping approach for parallel interleaver design with multiples read and write accesses," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010.
- [17] M. M. K. Eid M. Abdel-Hamid, Hossam A. H. Fahmy, "Memory conflict analysis for a multi-standard, reconfigurable turbo decoder," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2011.