Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 948957, 7 pages
doi:10.1155/2012/948957

Research Article

Low-Complexity Hardware Interleaver/Deinterleaver for IEEE 802.11a/g/n WLAN

Zhen-dong Zhang,[1] Bin Wu,[1] Yu-mei Zhou,[1] and Xin Zhang[2]

[1] *ASIC and System Department, Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China*
[2] *DSP Software Department, Datang Mobile Communications Equipment Co., Ltd., Beijing 100083, China*

Correspondence should be addressed to Zhen-dong Zhang, zhangzhendong@ime.ac.cn

Received 27 June 2011; Revised 15 December 2011; Accepted 30 December 2011

Academic Editor: Sungjoo Yoo

A high-speed low-complexity hardware interleaver/deinterleaver is presented. It supports all 77 802.11n high-throughput (HT) modulation and coding schemes (MCSs) with short and long guard intervals and the 8 non-HT MCSs defined in 802.11a/g. The paper proposes a design methodology that distributes the three permutations of an interleaver to both write address and read address. The methodology not only reduces the critical path delay but also facilitates the address generation. In addition, the complex mathematical formulas are replaced with optimized hardware structures in which hardware intensive dividers and multipliers are avoided. Using 0.13 um CMOS technology, the cell area of the proposed interleaver/deinterleaver is 0.07 mm^2, and the synthesized maximal working frequency is 400 MHz. Comparison results show that it outperforms the three other similar works with respect to hardware complexity and max frequency while maintaining high flexibility.

1. Introduction

Over the past several years IEEE 802.11a/g wireless local area network (WLAN) [1, 2] has appeared in a great number of electronic devices, from notebooks to personal media players, to mobile phones. Recently, new applications such as simultaneous transmission of multiple HDTV signals, videos, and online games have created a need for higher throughput WLAN. The latest established IEEE 802.11n WLAN [3] employs multiple-input multiple-output (MIMO) orthogonal frequency-division multiplexing (OFDM) transmission technique to enable high-throughput communication for up to 600 Mb/s. However, it also increases the computational and the hardware complexities greatly, compared with the original 802.11a/g standard. In order to obtain economical 802.11n products, optimized implementation of the physical layer of 802.11n has gained significant attention for the researchers [4].

Interleaver is mandatory in the physical layer of 802.11n. It plays a key role in exploiting spatial diversity and frequency diversity [5]. Interleaving algorithms for high-throughput (HT) wireless communication system to reduce the decoding errors have been proposed in [6–9]. Hardware architectures for 802.11a/g interleaver have been reported in several literatures [10–12]. However, few papers [13, 14] focus on the hardware implementation of 802.11n interleaver and deinterleaver. According to IEEE 802.11n standard, the interleaver and deinterleaver are required capable of processing 648 bits within 3.6 μs when short guard interval is used, which means that a minimum operating frequency of 180 MHz is required to be achieved. In addition, four inter-leavers and four deinterleavers are needed in a 4-stream 802.11n system. It can be seen from [15] that if the interleaver/deinterleaver is not efficiently implemented, it occupies silicon area as many as some significant blocks (i.e., Fast Fourier Transform (FFT), Viterbi Decoder, and Phase Tracking.) do. Therefore, designing a high-speed low-complexity interleaver/deinterleaver for the 802.11n WLAN is very important. In this paper, a design methodology, which distributes three permutations of an interleaver to both write address and read address, is proposed to reduce critical path delay and hardware complexity. Besides, the complex

mathematical formulas are replaced with optimized hardware structures. All hardware intensive blocks as dividers and multipliers in the permutation formulas are suppressed.

The remainder of this paper is organized as follows. Section 2 introduces interleaving algorithm for IEEE 802.11a/g/n. In Section 3, the proposed design methodology and hardware architecture are demonstrated in detail. Implementation and comparison results are shown in Section 4. Conclusions are given in Section 5.

2. Interleaver for 802.11a/g/n

The interleaver used in IEEE 802.11a/g/n WLAN is block interleaver with block size corresponding to the number of bits in a single OFDM symbol. For 802.11a/g or 802.11n single-stream mode, the interleaving algorithm is defined by two permutations. If more than one spatial stream exists in the 802.11n physical layer, a third permutation called frequency rotation will be applied to the additional spatial streams. Let k be the index of the coded bits before the first permutation, let i be the index after the first permutation, j be the index after the second permutation, and r be the index after the third permutation prior to modulation mapping. The first permutation which ensures adjacent coded bits mapped onto nonadjacent subcarriers is defined as

$$i = N_{\text{row}}(k \bmod N_{\text{col}}) + \text{floor}\left(\frac{k}{N_{\text{col}}}\right), \tag{1}$$

where $k = 0, 1, \ldots, N_{\text{cbpss}}(i_{\text{ss}}) - 1$. N_{col} is the column number, which can be 13, 16, or 18. $N_{\text{row}} = N_{\text{cbpss}}(i_{\text{ss}})/N_{\text{col}}$ is the number of rows of the interleaving matrix. The modulo operation is denoted by mod, while the function floor$(\cdot)$ stands for the largest integer not exceeding the parameter. The second permutation ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation and, thereby, long runs of low reliability bits are avoided. It is defined as

$$\begin{aligned} j = {} & s(i_{\text{ss}}) * \text{floor}\left(\frac{i}{s(i_{\text{ss}})}\right) \\ & + \left(i + N_{\text{cbpss}}(i_{ss}) - \text{floor}\left(N_{\text{col}} * \frac{i}{N_{\text{cbpss}}(i_{\text{ss}})}\right)\right) \\ & \times \bmod s(i_{\text{ss}}), \end{aligned} \tag{2}$$

where $i = 0, 1, \ldots, N_{\text{cbpss}}(i_{\text{ss}}) - 1$. $N_{\text{bpscs}}(i_{\text{ss}})$ is the number of coded bits per subcarrier. $s(i_{\text{ss}}) = \max(N_{\text{bpscs}}(i_{\text{ss}})/2, 1)$. The third permutation ensures that the consecutive carriers used across spatial streams should not be highly correlated. It is defined as

$$\begin{aligned} r = {} & \left(j - \left(((i_{\text{ss}} - 1) * 2) \bmod 3 + 3 \right.\right. \\ & \left.\left. * \text{floor}\left(\frac{i_{\text{ss}} - 1}{3}\right)\right) * N_{\text{rot}} * N_{\text{bpscs}}(i_{\text{ss}})\right) \\ & \times \bmod N_{\text{cbpss}}(i_{\text{ss}}), \end{aligned} \tag{3}$$

where $j = 0, 1, \ldots, N_{\text{cbpss}}(i_{\text{ss}}) - 1$. N_{rot} is frequency rotation parameter. It is 11 when 20 MHz channel is used and 29 when 40 MHz channel is used.

The deinterleaver, which performs the inverse rotation, is also defined by three permutations. The first permutation reverses the third permutation of the interleaver. It is defined as

$$\begin{aligned} j = {} & \left(r + \left(((i_{\text{ss}} - 1) * 2) \bmod 3 + 3 * \text{floor}\left(\frac{(i_{\text{ss}} - 1)}{3}\right)\right)\right. \\ & \left. * N_{\text{rot}} * N_{\text{bpscs}}(i_{\text{ss}})\right) \bmod N_{\text{cbpss}}(i_{\text{ss}}). \end{aligned} \tag{4}$$

The second permutation reverses the second permutation in the interleaver. It is defined as

$$\begin{aligned} i = {} & s(i_{\text{ss}}) * \text{floor}\left(\frac{j}{s(i_{ss})}\right) \\ & + \left(j + \text{floor}\left(N_{\text{col}} * \frac{i}{N_{\text{cbpss}}(i_{\text{ss}})}\right)\right) \bmod s(i_{\text{ss}}). \end{aligned} \tag{5}$$

The third permutation reverses the first permutation of the interleaver. It is defined as

$$k = N_{\text{col}} * i - \left(N_{\text{cbpss}}(i_{\text{ss}}) - 1\right) * \text{floor}\left(\frac{i}{N_{\text{row}}}\right). \tag{6}$$

3. Proposed Interleaver/Deinterleaver Design

3.1. Proposed Design Methodology. Generally, the implementing approaches for interleaver and deinterleaver can be classified in two categories, look-up-table- (LUT-) based approach and address-generation-unit- (AGU-) based approach [10]. The former can support most interleaving schemes and has low hardware complexity. However, because the complete permuted sequence of every interleaving mode has to be stored explicitly in the LUT, it is not suitable for multimode design. In IEEE 802.11n standard, there are 77 HT modulation and coding schemes (MCSs), in which 32 interleaving modes are defined. And in the IEEE 802.11a/g standard, there are 8 non-HT MCSs, in which another 4 interleaving modes are defined. Therefore, the latter AGU-based approach, which can achieve a significant reduction of silicon area at the price of designing a smart AGU, is a better candidate for 802.11a/g/n interleaver and deinterleaver.

For the hardware implementation of interleaver with AGU, the main challenge is to simplify address computation for AGU and at the same time meeting the high-speed requirements. In general, interleaving operation is realized by writing the incoming data stream into a memory matrix according to the permuted address, and then simply reading out data with sequential address [14]. As shown in Section 2, the set of equations (1)–(3), which provides permuted address for the 802.11a/g/n interleaver, involves a large number of hardware intensive blocks, for example, dividers, multipliers, and so forth. Therefore, this conventional implementation will lead to long critical path and very high hardware complexity. In order to deal with this issue, a design methodology that uses the permuted sequence from (1) as the write address and the permuted sequence from (4)-(5) as the read address is firstly proposed in our previous work [13]. Here, we introduce a more attractive design methodology

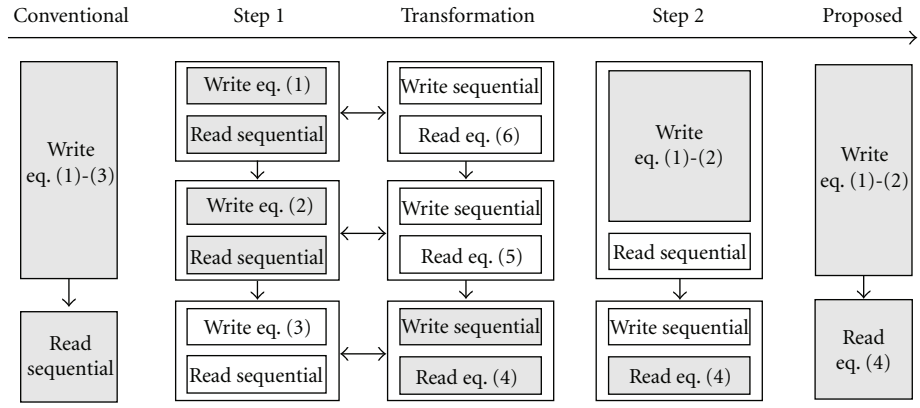

Figure 1: Principle of the proposed design methodology.

which has lower hardware complexity than the original one. The principle of the new design methodology is illustrated in Figure 1. First the three permutations of interleaver are implemented separately. Since (6), (5), and (4) are the inverse of (1), (2), and (3), respectively, the permutations defined by (1)–(3) can also be realized by first writing the incoming data stream into the memory matrix with sequential address, and then reading out the data according to the permuted address defined by (6)–(4). Based on this transformation, the three permutations of intealerver can be implemented using a hybrid method denoted by step 2 in Figure 1. Because the consecutive sequential reading and sequential writing can be omitted, the interleaving operation can be realized by using the permuted sequence from (1)-(2) as the write address and the permuted sequence from (4) as the read address.

3.2. Optimized Hardware Structures. Due to complex mathematical computation in (1)-(2) and (4), the direct arithmetic way to yield the permuted sequence is hardware inefficient. In order to avoid the hardware intensive blocks, for example, dividers, multipliers, and so forth, those complex equations are further replaced with optimized hardware structures.

In (1)-(2), the parameter $s(i_{ss})$ is 1 for both BPSK and QPSK, 2 for 16QAM, and 3 for 64QAM. Next we consider the three different cases separately. For the BPSK and QPSK case, (2) can be rewritten as $j = i$. Hence, the relation between j and k can be simply represented as

$$j = N_{\text{row}}(k \bmod N_{\text{col}}) + \text{floor}\left(\frac{k}{N_{\text{col}}}\right). \tag{7}$$

To describe the recursion of (7) in a generic way, first the permuted sequence j is represented with constant and parameter N_{row}, for example, the index j_0 is equal to 0 when the input k is 0, the index j_1 is equal to N_{row} when the input k is 1, the index j_2 is equal to $2N_{\text{row}}$ when the input k is 2, and so forth. Then we put the permuted sequence into the interleaving matrix row by row. Finally the behavior of (1), (2) for BPSK and QPSK is obtained as shown in Figure 2. In every row, the value of first index is equal to the row number, and the value of next index is generated by adding the parameter N_{row} to the value of current index.

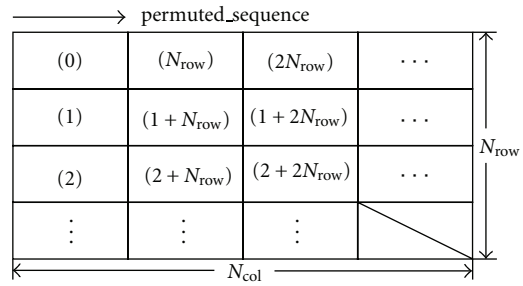

Figure 2: Permuted sequence from (1)-(2) for BPSK and QPSK.

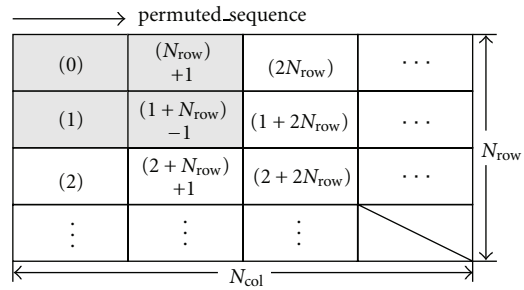

Figure 3: Permuted sequence from (1)-(2) for 16QAM.

For the 16QAM case, the parameter $N_{\text{cbpss}}(i_{ss})$ is integer times of 2; then (2) can be rewritten as

$$j = 2 * \text{floor}\left(\frac{i}{2}\right) + \left(i - \text{floor}\left(\frac{i}{N_{\text{row}}}\right)\right) \bmod 2. \tag{8}$$

First the permuted sequence j from (8) is also represented with constant and parameter N_{row}, for example, the index j_0 is equal to 0 when the input k is 0, the index j_1 is equal to $N_{\text{row}} + 1$ when the input k is 1, the index j_2 is equal to $2N_{\text{row}}$ when the input k is 2, and so forth. Then we also put the permuted sequence j into the interleaving matrix row by row. Finally the behavior of (1)-(2) for 16QAM is obtained as shown in Figure 3.

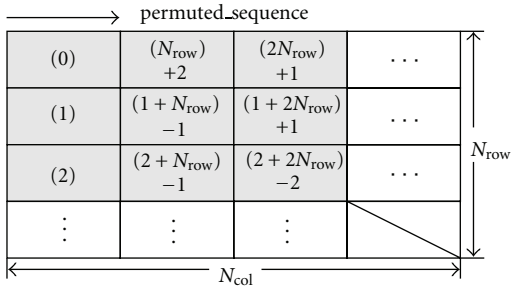

Figure 4: Permuted sequence from (1)-(2) for 64QAM.

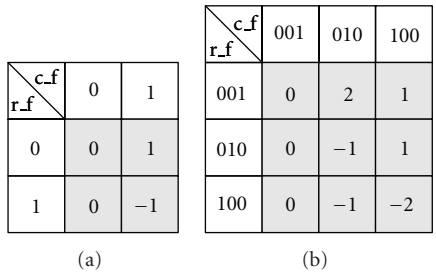

Figure 5: Submatrixes for (a) 16QAM and (b) 64QAM.

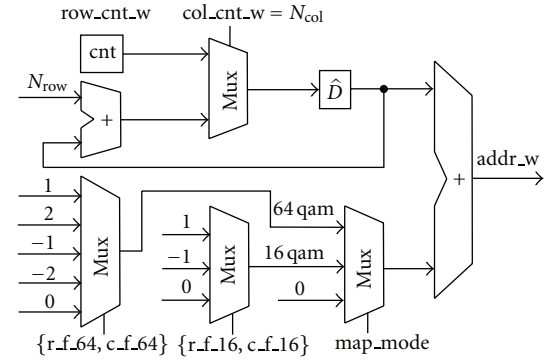

Figure 6: Proposed hardware structure for (1)-(2).

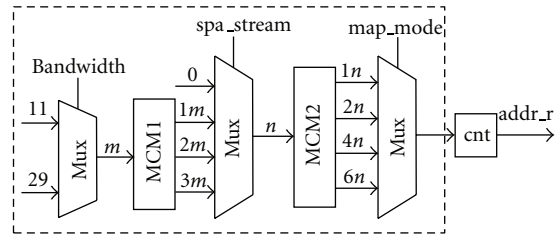

Figure 7: Proposed hardware structure for (4).

For the 64QAM case, the parameter $N_{cbpss}(i_{ss})$ is integer times of 3; then (2) can be rewritten as

$$j = 3 * \text{floor}\left(\frac{i}{3}\right) + \left(i - \text{floor}\left(\frac{i}{N_{row}}\right)\right) \bmod 3. \quad (9)$$

We analyze (9) using the similar manner as described for (8). The behavior of (1)-(2) for 64QAM is obtained as shown in Figure 4.

It can be seen that the sequences inside brackets are identical for all the three cases. Hence, the permuted sequence from (1)-(2) can be represented as the summation of identical sequence with offset sequence. For the BPSK and QPSK case, the offset sequence is all-zero sequence. For the 16QAM case, only 0, +1, and −1 are consisted in the offset sequence. And for the 64QAM case, the offset sequence is formed with 0, +1, −1, +2, and −2. To efficiently generate the offset sequence in hardware, a novel solution that divides the interleaving matrix into several small submatrices is proposed. Figure 5 shows the submatrices for 16QAM and 64QAM, where the row flag (r_f) and column flag (c_f) are used to determine the location of each index, thereby, the offset sequence can be generated using a few multiplexers and cyclic shift registers. Eventually, (1)-(2) can be implemented using an optimized hardware structure as shown in Figure 6. The upside circuits are used to generate the identical sequence, whereas the three underside multiplexers are used to generate the offset sequence.

In (4), the parameter i_{ss} is the index of spatial stream. If i_{ss} is equal to 1, (4) can be rewritten as $j = r$, and thus the permuted sequence j is sequential. It does not perform any frequency rotation. If i_{ss} is equal to 2, 3, or 4, the permuted sequence for any modulation scheme is also sequential but with a nonzero starting value. The starting values for different modulation schemes are shown in Table 1. An optimized hardware structure matches that the behavior of (4) is proposed as shown in Figure 7. The circuits in the dashed box are used to generate the starting values for the followed counter. The two multiple constant multiplication (MCM) units are implemented using hardwired shifts and adders, for example, $2m = m \ll 1$, $3m = 2m + m$, and so forth.

3.3. Complete Hardware Architecture. The proposed hardware architecture for the IEEE 802.11a/g/n interleaver/deinterleaver is shown in Figure 8. It consists of six functional blocks. The write address generation unit (WAGU) is used to generate the permuted sequence defined by (1)-(2), and the read address generation unit (RAGU) is used to generate the permuted sequence defined by (4). Figures 6 and 7 show the optimized hardware structures for WAGU and RAGU, respectively. The two single-port RAMs denoted by SPRAM1 and SPRAM2 are act as a ping-pang double buffer to support consecutive OFDM symbols. Because only one set of address bus exists in one single-port RAM, one multiplexer is used for each to select the input address. The look-up table (LUT) is used for storing configuring parameters as N_{cbpss} and N_{row}, thereby, the supported interleaving and deinterleaving modes can be changed by updating the small LUT without modifying any other computational or control unit.

The state transition diagram for the control finite state machine (FSM) is shown in Figure 9. Initially the FSM is in the idle state S0. When the input valid signal din_valid is activated, the FSM enters the state S1, in which the input data stream is written into SPRAM1. After N_{cbpss} bits have been written into the SPRAM1, the FSM enters the state

Table 1: Starting values for different modulation schemes.

Channel bandwidth	Spatial Stream	BPSK	QPSK	16QAM	64QAM
20 MHz	1	0	0	0	0
	2	11×2	$11 \times 2 \times 2$	$11 \times 2 \times 4$	$11 \times 2 \times 6$
	3	11×1	$11 \times 1 \times 2$	$11 \times 1 \times 4$	$11 \times 1 \times 6$
	4	11×3	$11 \times 3 \times 2$	$11 \times 3 \times 4$	$11 \times 3 \times 6$
40 MHz	1	0	0	0	0
	2	29×2	$29 \times 2 \times 2$	$29 \times 2 \times 4$	$29 \times 2 \times 6$
	3	29×1	$29 \times 1 \times 2$	$29 \times 1 \times 4$	$29 \times 1 \times 6$
	4	29×3	$29 \times 3 \times 2$	$29 \times 3 \times 4$	$29 \times 3 \times 6$

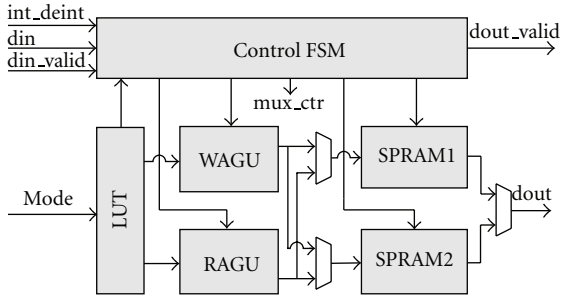

Figure 8: Hardware architecture for interleaver/deinterleaver.

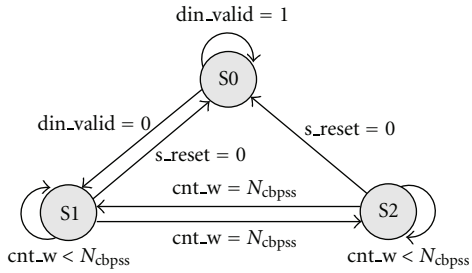

Figure 9: State diagram for the control FSM.

S2, in which the input data stream is written into SPRAM2 and the output data stream is read from SPRAM1. Note that the FSM is changed to the idle state S0 from whatever state the FSM is in when the soft reset signal (s_reset) of the interleaver/deinterleaver is activated.

Since the deinterleaving operation is inverse of interleaving operation, the deinterleaver can be realized by alternating write address and read address of interleaver. Instead, we implement the deinterleaver via alternating the request signals of write address and read address in the control FSM. Whether the proposed design acts as interleaver or deinterlever is controlled by the input signal int_deint. Because IEEE 802.11a/g/n WLAN is time division duplex system, the same interleaver/deinterleaver hardware block can be shared by transmitter and receiver.

4. Implementation and Comparison

At first, the proposed interleaver/deinterleaver is modeled in Verilog. The functional verification is done by comparing the results from ModelSim simulator with the results from original equations. After functional verification, the proposed interleaver/deinterleaver is synthesized into a 0.13 μm one-poly six-metal layer (1P6M) CMOS library from semiconductor manufacturing international corporation (SMIC). Since only a few discrete-size memory blocks can be generated by the memory compiler tool Artisan, two 672×6-bit single-port RAMs are adopted for the IEEE 802.11a/g/n interleaver/deinterleaver to support the maximal block size of 648 and 6 soft bits processing in the decoder.

The implementation details and comparison results are shown in Table 2. The area and max working frequency are reported by Design Compiler (DC) tool. We compare this implementation with three other similar works. Among the four designs, this work is the only one that supports three standards. It is to be noted that because of the differences in target technology, parallelism, and ping-pang buffer, it is hard to make an absolutely fair comparison. In practice, although memory dominates the total silicon area of an interleaver/deinterleaver, the hardware complexity mainly depends on the implementation of AGU since features as parallelism and ping-pang buffer can be simply changed by updating control logic and memory configuration. Hence, it is appropriate to make a relatively fair hardware complexity comparison between two works by comparing their normalized complexity (NC). The NC represents the normalized hardware requirements for realizing the aforementioned three permutations; it is defined by.

$$\text{NC} = \frac{\text{AGU \& Logic\_Area}}{(\text{MFS})^2 * \text{AGU\_Number}} * 10^{-6}. \quad (10)$$

The parameter MFS is the minimum feature size of target technology. In [16], an interleaver complaint to WWiSE proposal is implemented using parallel-bit architecture. The parallel-bit architecture can achieve small interleaving latency at the price of high hardware complexity. Unfortunately, the interleaving algorithm defined in the final 802.11n standard is different from the WWiSE proposal, therefore it cannot be directly used to implement the 802.11n interleaver/deinterleaver. In [13], we have presented a low-cost 802.11n interleaver/deinterleaver. It can be seen that this new implementation offers a reduction of 37.5% NC over the earlier one. The reason is that initial read address and initial register values of each mode no longer need to be stored in LUT for the present proposed design methodology.

Table 2: Implementation details and comparison results.

	IWCMC'05 [16]	ASICON'09 [13]	ISCAS'09 [14]	This work
Function	Int.	Int. & Deint.	Int. & Deint.	Int. & Deint.
Standard	WWiSE Proposal to802.11n	802.11n	802.11n	802.11a/g/n
Parallel streams	1	1	4	1
Parallel bits	24	3	6	6
Technology	0.18 μm	0.13 μm	65 nm	0.13 μm
Ping-pang buffer	Yes	Yes	No	Yes
Memory area (μm^2)	564587	53630	25136	63273
AGU and logic area (μm^2)	168658	11270	9690	7049
AGU number	1	1	3	1
Total area (μm^2)	733245	64900	34824	70322
Max frequency (MHz)	200	350	225	400
Normalized complexity	5.205	0.667	0.765	0.417

In [14], a 4-stream interleaver is proposed for 802.11n WLAN. Parallel-stream architecture can use 3 sets of AGU to provide address for 4 spatial streams since the maximum types of spatial streams defined in 802.11n are 3. However, the parallel-stream implementation will degrade flexibility from point of view of system integration compared to single-stream architecture; that is, the single-stream implementation can be integrated into 1-, 2-, 3-, or 4-stream 802.11n WLAN chip without any modification. The comparison shows that our implementation offers a reduction of 45.4% NC over that of reference [14]. Moreover, ping-pang buffer is necessary for interleaver/deinterleaver in practical 802.11n WLAN system, thereby control logic and memory configuration of reference [14] need to be updated to support the processing of consecutive OFDM symbols. To sum up, the proposed implementation outperforms the three other works with respect to hardware complexity and max frequency while maintaining high flexibility. On the one hand, this can be attributed to the fact that the three permutations of interleaver/deinterleaver are properly distributed to both write address and read address. On the other hand, by using submatrixes instead of full interleaving matrix in the first two permutation steps, only a few adders and multiplexers are required in the proposed AGU.

5. Conclusion

This paper presents a high-speed low-complexity interleaver/deinterleaver for IEEE 802.11a/g/n WLAN. Currently, it has been successfully integrated into a 2-stream 802.11a/g/n transceiver chip fabricated by SMIC. The proposed design methodology and hardware architecture can also be used to implement other block interleaver/deinterleaver. The interleaver/deinterleaver complaint to IEEE 802.16d/e standard or HiperLAN/2 standard can be obtained just by updating the parameter LUT in the proposed design. The comparison results show that the proposed interleaver/deinterleaver has lower hardware complexity and can run at higher working frequency compared with three other similar works, which makes it suitable for the IEEE 802.11a/g/n WLAN.

Acknowledgments

This work was supported by the Major National Science and Technology Program of China under Grant no. 2010ZX03005-001 and the National Natural Science Foundation of China under Grant no. 60976022.

References

[1] IEEE Std., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: high-speed physical layer in the 5 GHz band," IEEE Std. 802.11a-1999, The IEEE Standards Association, New York, NY, USA, September 1999.

[2] IEEE Std., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: further higher data rate extension in the 2.4 GHz band," IEEE Std. 802.11g-2003, The IEEE Standards Association, New York, NY, USA, June 2003.

[3] IEEE Std., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: enhancements for higher throughput," IEEE Std. 802.11n-2009, The IEEE Standards Association, New York, NY, USA, October 2009.

[4] T. Paul and T. Ogunfunmi, "Evolution, insights and challenges of the PHY layer for the emerging ieee 802.11n amendment," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 4, pp. 131–150, 2009.

[5] J. Terry and J. Heiskala, *OFDM Wireless LANs: A Theoretical and Practical Guide*, chapter 3, SAMS, Indianapolis, Ind, USA, 2002.

[6] V. D. Nguyen and H. Kuchenbecker, "Block interleaving for soft decision viterbi decoding in OFDM systems," in *Proceedings of the IEEE 54th Vehicular Technology Conference (VTC '01)*, pp. 470–474, Atlantic City, NJ, USA, October 2001.

[7] S. Ramseier, "Shuffling bits in time and frequency—an optimum interleaver for OFDM," in *Proceedings of the International Conference on Communications (ICC'03)*, pp. 3418–3422, Anchorage, Alaska, USA, May 2003.

[8] X. F. Wang, Y. R. Shayan, and M. Zeng, "On the code and interleaver design of broadband OFDM Systems," *IEEE Communications Letters*, vol. 8, no. 11, pp. 653–655, November 2004.

[9] N. Huaning, O. Xuemei, and N. Chiu, "Interleaver design for MIMOOFDM based wireless LAN," in *Proceedings of the IEEE Wireless Communication and Networking Conference (WCNC '06)*, pp. 1825–1829, Las Vegas, Nev, USA, April 2006.

[10] T. Eric and L. Dake, "A hardware architecture for a multi-mode block interleaver," in *Proceedings of the IEEE International Conference on Circuits and Systems for Communications (ICCSC '04)*, Moscow, Russia, June 2004.

[11] C. Yu, M. H. Yen, P. A. Hsiung, and S. J. Chen, "Design of a high-speed block interleaving/deinterleaving architecture for wireless communication applications," in *Proceedings of the International Conference on Consumer Electronics (ICCE '09)*, Las Vegas, Nev, USA, January 2009.

[12] B. K. Upadhyaya and S. K. Sanyal, "Design of a novel FSM based reconfigurable multimode interleaver for WLAN application," in *Proceedings of the International Conference on Devices and Communications (ICDeCom '11)*, Mesra, India, February 2011.

[13] Z.-D. Zhang, B. Wu, Y.-X. Zhu, and Y.-M. Zhou, "Design and implementation of a multi-mode interleaver/deinterleaver for MIMO OFDM systems," in *Proceedings of the 8th IEEE International Conference on ASIC (ASICON '09)*, pp. 513–516, Changsha, China, October 2009.

[14] A. Rizwan and L. Dake, "Low complexity hardware interleaver for MIMOOFDM based wireless LAN," in *Proceedings of the IEEE International Symposiumon Circuits and Systems (ISCAS '09)*, vol. 2, pp. 1747–1750, Taipei, Taiwan, May 2009.

[15] J. Son, I.-G. Lee, and S.-K. Lee, "ASIC implementation and verification of MIMO-OFDM transceiver for wireless lan," in *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)*, pp. 1–5, Athens, Greece, September 2007.

[16] Y.-W. Wu, P. Ting, and H.-P. Ma, "A high speed interleaver for emerging wireless communications," in *Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing (IWCMC '05)*, pp. 1192–1197, Maui, Hawaii, USA, June 2005.