Efficient Interleaver Design for MIMO-OFDM Based Communication Systems on FPGA

Zafar Iqbal[†], Saeid Nooshabadi[§], and Heung-No Lee[†]
[†]Dept. of Information and Communications, Gwangju Institute of Science and Technology
Gwangju, 500–712, Republic of Korea, Emails: {zafar, heungno}@gist.ac.kr
[§]Dept. of Electrical and Computer Engineering, Michigan Technological University
Houghton, Michigan 49931-1295, Email: saeid@mtu.edu

***Abstract*—In this paper, we present a memory-efficient and faster interleaver implementation technique for MIMO-OFDM communication systems on FPGA. The IEEE 802.16 standard is used as a reference for simulation, implementation, and analysis. A method for the interleaver design on FPGA and its memory utilization results are presented. Our design utilizes the minimum required on-chip memory for the interleaver implementation. Using the proposed interleaver design method, the data rates for MIMO-OFDM based communication systems are doubled for 2×2 MIMO systems without using the transmit diversity.**

I. Introduction

The IEEE 802.16 defines the standard for broadband wireless access covering the physical layer and medium access specifications for wireless metropolitan area networks (WMAN). The IEEE 802.16 Air Interface Standard is a technology that is playing a key role in fixed broadband wireless MAN [1]. The forward error correction (FEC) mechanism in the standard plays a very important role in its performance. A number of techniques are being used to achieve highly effective error-control coding such as Turbo codes and concatenated codes. However, interleaving also plays a major role in the FEC mechanism. The aim of interleaving is to reorder the incoming data and make the adjacent bits non-adjacent by a factor, to cope with the burst errors occurring during the transmission of data over the channel. Memory utilization and frequent memory accesses time are a crucial part of interleaver design, targeting less memory utilization and reduced memory access in order to reduce the power dissipation of the overall system.

A memory-efficient interleaver design method has been proposed in [2], where the author presents a divided memory bank architecture for the implementation of interleaver for IEEE 802.16e. An efficient memory memory address manipulation technique that can improve the performance interleaver is proposed in [3], but no details are provided. An analysis of the effects of interleaving process on spectral efficiency of IEEE 802.16 for different environments is done in [4]. They also measured the system throughput and interleaver block delay, and proposed solutions for interleaver design. In [5], the authors investigate structural behavior of interleaving parameters and suggest several optimization methods for the convolutional turbo code (CTC) interleaver of IEEE 802.16 standard.

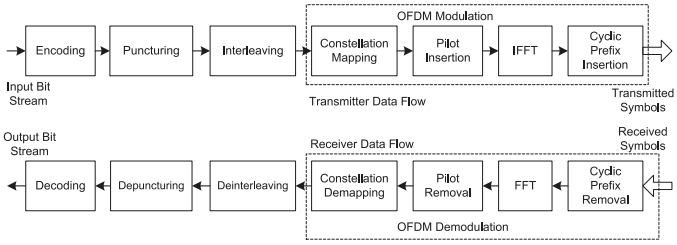

Fig. 1. OFDM Communication System Physical Layer

In this paper, we present an efficient interleaver design for IEEE 802.16 system on FPGA. This paper focuses on the interleaver design of the system implemented in [6]. Our goal is to achieve minimum memory usage, faster interleaving, and increased speed of the overall system. We use the interleaving method defined in the standard and present an efficient FPGA implementation of the proposed design.

II. System Description

The basic OFDM communication system's physical layer is shown in Figure 1. The forward error correction (FEC) blocks include convolutional encoding, puncturing, and interleaving.

A modification of the system described in Figure 1 is to use two separate data streams to enhance the data rate and possibly increase the number of antennas by using spatial as well as transmit diversity. However, In this analysis, only spatial diversity is used by having two parallel data streams that make up a 2×2 MIMO-OFDM communication system. Four different schemes are implemented for analysis, which are categorized as follows. Detailed simulation and implementation results of these schemes are given in [6].

1) Case 1: Cross-antenna (C-A) convolutional coding with per-antenna (P-A) interleaving (CAC-PAI).
2) Case 2: Per-antenna convolutional coding with per-antenna interleaving (PAC-PAI).
3) Case 3: Cross-antenna convolutional coding with cross-antenna interleaving (CAC-CAI).
4) Case 4: Per-antenna convolutional coding with cross-antenna interleaving (PAC-CAI).

For this analysis, a coding rate of 1/2 is used for BPSK modulation, while coding rate of 3/4 is used for QPSK, 16-QAM and 64-QAM. Next step is interleaving which is

978-1-4673-1356-8/12/$31.00 ©2012 IEEE

implemented using a block interleaver, whose size varies according to the modulation scheme used and the system configuration [1]. The receiver performs these functions in reverse order to retrieve the data as shown in Figure 1. We use memoryless AWGN channel, and a an ideal channel gain of 1 for each sub-carrier which eliminates the need for channel estimation and carrier recovery.

The encoded data is interleaved by a block interleaver with a block size corresponding to the number of coded bits per the allocated subchannel per OFDM symbol, N_{cbps}. The interleaver is defined by a two step permutation. The first ensures that adjacent coded bits are mapped onto nonadjacent subcarriers, while the second ensures that adjacent coded bits are mapped alternately onto less or more significant bits of the constellation to avoid long runs of lowly reliable bits.

Let N_{cpc} be the number of coded bits per subcarrier ie 1, 2, 4, or 6 for BPSK, QPSK, 16-QAM, and 64-QAM respectively. Let $s = ceil(N_{\text{cpc}}/2)$. Within a block of N_{cbps} bits at transmission, let k be the index of the coded bit before the first permutation; m_k be the index of that coded bit after the first and before the second permutation; and let j_k be the index after the second permutation, just prior to modulation mapping.

The first permutation is defined as,

$$m_k = (N_{\text{cbps}}/12).k_{mod12} + \lfloor k/12 \rfloor \qquad k = 0, 1, ..., N_{\text{cbps}} - 1 \tag{1}$$

The second permutation is defined as,

$$j_k = s.\lfloor m_k/s \rfloor + (m_k + N_{\text{cbps}} - \lfloor 12.m_k/N_{\text{cbps}} \rfloor)_{mod(s)} \qquad k = 0, 1, ..., N_{\text{cbps}} - 1 \tag{2}$$

The de-interleaver performs the inverse operation and is also defined by two permutations. Within a received block of N_{cbps} bits, let j be the index of a received bit before the first permutation; m_j be the index of that bit after the first and before the second permutation; and let k_j be the index of that bit after the second permutation, just prior to delivering the block to the decoder. The first permutation in the de-interleaver is the inverse of the second permutation in the interleaver and conversely.

The first permutation is defined as,

$$m_j = s.\lfloor j/s \rfloor + (j + \lfloor 12.j/N_{\text{cbps}} \rfloor)_{mod(s)} \qquad j = 0, 1, ..., N_{\text{cbps}} - 1 \tag{3}$$

The second permutation is defined as,

$$k_j = 12.m_j - (N_{\text{cbps}} - 1).\lfloor 12.m_j/N_{\text{cbps}} \rfloor \qquad j = 0, 1, ..., N_{\text{cbps}} - 1 \tag{4}$$

Table I shows the bit interleaver sizes as a function of modulation and coding. The first bit out of the interleaver maps to the MSB in the constellation [1]. We implement the 16-subchannel system and hence the corresponding interleaver block sizes.

TABLE I
BLOCK SIZES OF THE BIT INTERLEAVER

Mod. Scheme	Default 16 subchannels	8 subchannels	4 subchannels	2 subchannels	1 subchannel
	N_{cbps}				
BPSK	192	96	48	24	12
QPSK	384	192	96	48	24
16-QAM	768	384	192	96	48
64-QAM	1152	576	288	144	72

III. SYSTEM IMPLEMENTATION

The IEEE 802.16 (WiMAX) system is implemented on FPGA for design emulation and verification.

Convolutional encoder and puncturing blocks are implemented using shift registers and XOR gates. For QPSK, 16-QAM, and 64-QAM mapping, puncturing reduces the code rate to 3/4 as described in [1]. There is no puncturing used for BPSK mapping as the code rate is always rate 1/2.

The interleaver[1] is implemented using RAM blocks in the FPGA and and employing logic cells (LC) for the state machine of the address generator for read/write operations. Double buffering technique is used to implement the interleaver to eliminate the delay in the interleaving process. After the first block of symbols is stored in the buffer, the address generator starts generating read addresses and starts reading data from the buffer. In the mean time, the second buffer is filled with incoming data and the interleaver will start reading from the second buffer after the first one is read out completely. This techniques has the initial latency equal to the incoming time for one block of symbols but there is no delay in the subsequent interleaving process. Table II shows the buffer sizes for different interleaving schemes used in our system. The number of buffers increase with the increase in modulation symbol size, so that we can write/read data from them at the same time.

A. Interleaver for BPSK Mapping

For BPSK mapping, the interleaver is implemented using a single memory block of double the required size. For example, an interleaver of size 192 is implemented using a buffer of 384 bits for per-antenna interleaving and an interleaver of size 384 is implemented using a buffer of 768 bits for cross-antenna interleaving as shown in Table II. Incoming bits are first stored in the RAM until 192 bits are filled and then the read-out is enabled. A state machine generates write addresses for the RAM to write data to it. After 192 writes to the RAM, it asserts the read enable signal and starts generating read addresses for the RAM while continuing with the write process. This way,

[1]**Note:** Here, we will explain only the interleaver. Deinterleaver is implemented using the same technique but only reversing the data read and write orders.

TABLE II
BUFFER SIZES FOR DIFFERENT INTERLEAVERS

Mapping Type	BPSK		QPSK		16-QAM		64-QAM	
Interleaver Type	P-A Interleaving	C-A Interleaving	P-A Interleaving	C-A Interleaving	P-A Interleaving	C-A Interleaving	P-A Interleaving	C-A Interleaving
Buffer Size	384	768	384	768	384	768	384	768
No. of Buffers	1	1	2	2	4	4	6	6

the next part of the buffer is filled when the interleaver finishes reading the first 192 bits and then it starts reading the next 192 bits.

B. Interleaver for QPSK Mapping

For QPSK mapping, the interleaver is implemented using two memory blocks of double the required size. A state machine generates a single address for the two RAMs to write data to them simultaneously. The pattern thus formed can be read out column-wise from RAM1 and RAM2 alternatively to implement the interleaver function. After 192 writes to each RAM, it asserts the read enable signals and starts generating read addresses for each RAM while continuing with the write process. The address generator generates two read addresses successively with an increment of 6, in order to read 2 locations from one RAM during one read operation. The first column of RAM1 is read first, followed by the first column of RAM2 and this process continues until the last location is read. This technique enables us to write 2 bits to the buffer at the same time and read 2 bits successively to generate a 2-bit symbol for QPSK mapping.

C. Interleaver for 16-QAM Mapping

For 16-QAM mapping, the interleaver is implemented using four memory blocks of double the required size. The method of read and write address generation is the same as explained before except the fact that now we have four separate RAMs that are first filled simultaneously and then the data is read out column-wise from RAM1, RAM2, RAM3, and RAM4 alternatively to implement the interleaver function. The address generator generates four read addresses successively with an increment of 3, in order to read 4 locations from one RAM during one read operation. This technique enables us to write 4 bits to the buffer at the same time and read 4 bits successively to generate a 4-bit symbol for 16-QAM mapping.

D. Interleaver for 64-QAM Mapping

For 64-QAM mapping, the interleaver is implemented using six memory blocks of double the required size. The method of read and write address generation is the same as explained before except the fact that now we have six separate RAMs that are first filled simultaneously and then the data is read out column-wise from RAM1, RAM2, RAM3, RAM4, RAM5, and RAM6 alternatively to implement the interleaver function. The address generator generates six read addresses successively with an increment of 2, in order to read 6 locations from one RAM during one read operation. This technique enables us to write 6 bits to the buffer at the same time and read 6 bits successively to generate a 6-bit symbol for 64-QAM mapping. The interleaver for 64-QAM consists of six RAM blocks and an address generator as shown in Figure 2. These figures are not shown for the other mapping schemes as it should be easily understood by reducing the number of RAM blocks and following the explanation given earlier.

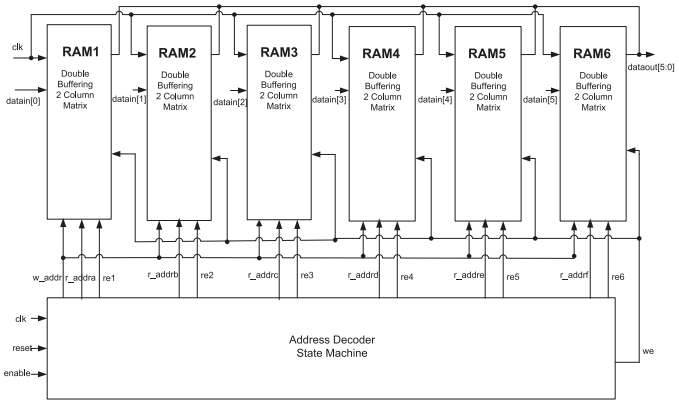

Fig. 2. Interleaver Schematic for 64-QAM Mapping

Constellation mapping for each scheme is implemented using a ROM which stores the pre-calculated I and Q output values for each input symbol. Two ROMs, one for each I (real) and Q (imaginary) values are used, having a 16-bit output with 14 fractional bits, 1 bit for magnitude, and 1 sign bit. The constellation mapping block for each scheme, implements the mapping technique as explained in [1], and generates the output I and Q data which is then fed to the IFFT module to compute the IFFT of the symbol.

The OFDM modulator block is implemented using double buffers for both I and Q inputs to the IFFT module. A buffer of size 384×16-bit is used for each I and Q input. When the first 192 locations of the buffer are filled, this block of data is input to the IFFT module along with the insertion of pilot symbols, DC, and null subcarriers. By the time the next 192 data symbols are written to the buffer, the IFFT module is ready for the next block of data and the process is repeated. The IFFT module is implemented using pipelined fast Fourier transform architecture. The OFDM modulation block is where pilot symbols, DC, and null symbols are inserted, IFFT is computed, and cyclic prefix is inserted to produce a 320 point OFDM output symbol.

IV. HARDWARE RESOURCE UTILIZATION

A. Interleaver Memory Utilization

Table III shows the RAM resource utilization for the different types of interleavers. Each look-up table (LUT) is 32 bit

TABLE III
RAM RESOURCE UTILIZATION BY INTERLEAVERS

Mod. Scheme	System Type	Required RAM Size (bits)	RAM Type Instantiated		RAM Size Instantiated (bits)	
			Automatic	Distributed	Automatic	Distributed
BPSK	Case 1/2	768	24 LUT	24 LUT	768	768
	Case 3/4	768	1 BRAM	24 LUT	18 K	768
QPSK	Case 1/2	1536	48 LUT	48 LUT	1536	1536
	Case 3/4	1536	1 BRAM	48 LUT	36 K	1536
16-QAM	Case 1/2	3072	96 LUT	96 LUT	3072	3072
	Case 3/4	3072	2 BRAM	96 LUT	72 K	3072
64-QAM	Case 1/2	4608	144 LUT	144 LUT	4608	4608
	Case 3/4	4608	3 BRAM	144 LUT	108 K	4608

TABLE IV
INTERLEAVERS RESOURCE UTILIZATION

Mod. Block	Interleaver Size (bits) – RAM Type	Maximum Frequency (MHz)	RAM Utilization		Slice Logic Utilization	
			No. of BRAMs out of 132	No. of LUTs out of 12480	No. of Slice Regs out of 32640	No. of Slice LUTs out of 32640
BPSK	192–Auto	181.48	0	12	77	170
	192–Dist.	181.48	0	12	77	170
	384–Auto	175.53	1 (18 Kb)	0	83	161
	384–Dist.	181.02	0	24	84	190
QPSK	384–Auto	172.01	0	24	95	217
	384–Dist.	172.01	0	24	95	217
	768–Auto	177.86	1 (36 Kb)	0	102	176
	768–Dist.	158.37	0	48	102	236
16-QAM	768–Auto	160.5	0	48	104	216
	768–Dist.	160.5	0	48	104	216
	1536–Auto	169.73	2 (36 Kb)	0	111	190
	1536–Dist.	169.73	0	96	111	303
64-QAM	1152–Auto	169.19	0	72	110	255
	1152–Dist.	169.19	0	72	110	255
	2304–Auto	154.58	3 (36 Kb)	0	117	188
	2304–Dist.	152.76	0	144	117	355

and BRAM is 36 Kb, which can also be used in separate blocks of 18 Kb. As we can see, the implementation is very efficient in terms of RAM resource utilization if Distributed RAM extraction method is used during the synthesis of our design. However, if Auto RAM extraction is used, the synthesizer uses Block RAM resources to implement some of the memory for interleavers in order to save distributed RAM resources which results in an increase in the operating frequency of the overall system. In this implementation, most of the block RAM resources are wasted as the modulation size increases to 64-QAM.

B. Interleaver Resource Utilization

Table IV shows the overall resource utilization, for both types of instantiation $i.e.$, block RAM and distributed RAM, by each type of interleaver used in our system. As we can see the larger size interleavers try to instantiate block RAM instead of distributed RAM in order to save resources.

Table V shows a comparison between our implementation and the one in [2] which targets an ASIC. Our method provides a simple write and read logic with no overhead of extra memory usage and complex circuitry.

TABLE V
IMPLEMENTATION COMPARISON WITH [2]

Method	Write Operation	Read Operation	Mem. Locations	R/W Circuitry
[2]	Needs transposer	Needs LUT	Irregular	Complex
Ours	No transposer needed	No LUT needed	Regular	Simple increment counter

C. Initial Latency and Data Rates

Table VI shows the initial latency of interleaver and the whole transmitter system as well as the output raw data rates that can be achieved with our implementation. Here T_b is the useful symbol time and T_g is the cyclic prefix of the OFDM Symbol. The initial latency for each system remains around 54.5 μs and the resulting OFDM symbol period is 13.82

TABLE VI
INTERLEAVER AND TRANSMITTER OUTPUT LATENCY AND DATA RATES

Mod. Scheme	System Type	Initial Latency				OFDM Symbol Period		Raw Data Rate (Mbps)
		Interleaver		Transmitter		T_g + T_b	Symbol Period, T_s	
		Clock Cycles	Time (μs)	Clock Cycles	Time (μs)			
BPSK	Case 1	192	1.28	1516	54.57	2.76 + 11.06	13.82	13.88
	Case 2	197	1.31	1526	54.92	2.76 + 11.06	13.82	13.88
	Case 3	390	2.60	1522	54.79	2.76 + 11.06	13.82	13.88
	Case 4	395	2.63	1529	55.04	2.76 + 11.06	13.82	13.88
QPSK	Case 1	581	3.87	4544	54.53	2.76 + 11.06	13.82	41.66
	Case 2	588	3.92	4558	54.71	2.76 + 11.06	13.82	41.66
	Case 3	1167	7.78	4548	54.57	2.76 + 11.06	13.82	41.66
	Case 4	1177	7.84	4562	54.75	2.76 + 11.06	13.82	41.66
16-QAM	Case 1	1160	7.73	9081	54.49	2.76 + 11.06	13.82	83.33
	Case 2	1170	7.80	9103	54.62	2.76 + 11.06	13.82	83.33
	Case 3	2331	15.53	9088	54.53	2.76 + 11.06	13.82	83.33
	Case 4	2341	15.60	9103	54.62	2.76 + 11.06	13.82	83.33
64-QAM	Case 1	1739	11.59	13622	54.49	2.76 + 11.06	13.82	125
	Case 2	1752	11.67	13643	54.57	2.76 + 11.06	13.82	125
	Case 3	3495	23.29	13622	54.49	2.76 + 11.06	13.82	125
	Case 4	3505	23.36	13643	54.57	2.76 + 11.06	13.82	125

μs as stated in the IEEE 802.16 standard [1]. The system described in [1] is a single data stream MIMO system with Alamouti transmit diversity but our system is a double data stream MIMO system, so the resulting data rate is twice as higher as the one in IEEE 802.16 standard. This gives us a significant advantage of increasing the data rate while still using a 2×2 MIMO system or even making it a 4×4 MIMO system by using transmit diversity. The initial latency is very low as compared to the 2.3 ms block interleaver latency system described in [4]. Due to the double buffering technique used in interleaver design and IFFT computation, there is no latency after the system outputs the first OFDM symbol.

V. CONCLUSION

In this paper we have provided an efficient way to design the IEEE 802.16 transmitter for FPGA. A special design method is used to implement the interleaver with minimum memory requirement and initial latency. We are able to double the data rate of the standard with efficient design methodologies and optimization. This approach can also be used to design other high-speed communication systems or to improve their speed.

VI. ACKNOWLEDGMENT

This work was supported by the National IT Industry Promotion Agency of Korea, National Research Foundation of Korea (NRF) grant, and by the Korean government Ministry of Education, Science, and Technology (MEST) (Do-Yak Research Program, No. 2011-0016496 and Haek Sim Research Program, No. 2011-0027682).

REFERENCES

[1] *IEEE Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Std. 802.16-2009, May 2009.

[2] Y.-N. Chang, "A low-cost dual-mode deinterleaver design," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 326–332, May. 2008.

[3] X. Yin and J. Liu, "Design and implementation of an improved 3G turbo codes interleaver for 3GPP system," in *9th International Conference on Electronic Measurements and Instruments (ICEMI)*, Beijing, China, Aug. 2009, pp. 3–319 – 3–321.

[4] N. Crisan, L. C. Cremene, E. Puschita, and T. Palade, "Spectral Efficiency Improvement for the under-11 GHz Broadband Wireless Access," in *International Conference on Telecommunications, (ICT)*, Jun. 2008, pp. 1–6.

[5] S.-J. Park and J.-H. Jeon, "Interleaver optimization of convolutional turbo code for 802.16 systems," *IEEE Communication Letters*, vol. 13, no. 5, pp. 339–341, May. 2009.

[6] Z. Iqbal and S. Nooshabadi, "Effects of Channel Coding and Interleaving in MIMO-OFDM Systems," in *54th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Seoul, Korea, Aug. 2011, pp. 1–4.