DSP Implementation of SISO and MIMO OFDM Acoustic Modems

Hai Yan, Shengli Zhou, Zhijie Shi, Jun-Hong Cui, Lei Wan, Jie Huang, and Hao Zhou
The Underwater Sensor Network Lab, University of Connecticut, Storrs, CT 06269, USA

***Abstract*—Significant progress has been made recently on the use of multicarrier modulation in the form of orthogonal frequency division multiplexing (OFDM) for high data rate underwater acoustic communications. In this paper, we present implementation results of OFDM acoustic modems under both single-input single-output and multi-input multi-output (two transmitters and two receivers) settings. The data rates are 3.2 kb/s and 6.4 kb/s for SISO and MIMO systems, respectively, with QPSK modulation, rate-1/2 channel coding, and signal bandwidth of 6 kHz. Real-time operation has been achieved with impressive margins, using a floating point TMS320C6713 DSP development board. Specifically, with convolutional coding, the processing time for each OFDM block of duration 210 ms is about 38 ms and 77 ms for SISO and MIMO settings, respectively, with the DSP running at 225 MHz. With nonbinary low-density parity-check (LDPC) coding, which gains about 2 dB in error performance relative to convolutional coding, the processing time per block increases to 50 ms and 101 ms for SISO and MIMO settings, respectively. We have also implemented the OFDM acoustic modems using a fixed-point TMS320C6416 DSP development board, where the DSP core runs at 1 GHz. The per-block processing time reduces by two thirds with negligible performance degradation.**

I. Introduction

Recently, multicarrier modulation in the form of orthogonal frequency division multiplexing (OFDM) has been actively pursued for high data rate underwater acoustic communications; see e.g., performance results with experimental data in [1]–[6]. The combination with multi-input and multi-output (MIMO) techniques has also been made to drastically increase the data rate through spatial modulation [7]–[9].

In this paper, we investigate the implementation of OFDM modems for underwater acoustic communications. First, we implement both single-input single-output (SISO) and multi-input multi-output (MIMO) OFDM modems on a floating-point TMS320C6713 DSP development board. Through the work load analysis, we optimize the implementation to accelerate the decoding speed. Real-time operation has been achieved with impressive margins. We also examine the fixed-point implementation on a TMS320C6416 DSP development board for both SISO and MIMO systems. Running at a higher clock frequency than TMS320C6713, the fixed-point implementation reduces the processing time by two thirds, with negligible performance degradation.

There are a few acoustic modems available, including commercial products such as [12]–[14], a model widely used in the research community [15], and experimental designs such as [16]–[19]. The designs in [12], [18], [19] are based on non-coherent frequency-shift-keying (FSK), and those in [13], [14], [17] are based on spread spectrum; all of them inherently have low data rate. The reconfigurable acoustic modem (rModem) of [16] has a flexible structure that can facilitate quick prototyping of different algorithms. The Micro-Modem of [15] has two operating modes: 1) a low-power low-rate mode based on non-coherent FSK, and 2) a high-power high-rate mode based on coherent phase-shift-keying (PSK). To the best of our knowledge, there is only one published paper on single-transmitter OFDM modem implementation [10] in addition to our initial effort in [11]. Both the receiver algorithms and the hardware platforms are different comparing [11] and [10]. So far, there is no reported result on MIMO-OFDM implementation.

The rest of this paper is organized as follows. We describe the OFDM transceiver design in Section II. The results are presented in Section III for floating-point implementation and in Section IV for fixed-point implementation. Section V concludes the paper.

II. OFDM Transceiver Design

In this paper, we consider two system setups: (1) a SISO system and (2) a MIMO system with two transmitters and two receivers, as shown in Fig. 1. For MIMO OFDM, two separate data streams are transmitted in parallel [7], while only one data stream is transmitted for SISO OFDM [1]. The data format is illustrated in Fig. 2. The SISO transmission is obtained by turning the second transmitter off.

A. Transmitter Design

The basic parameters for the zero-padded (ZP) OFDM are listed in Table I. The sampling rate is $f_s = 48$ kHz and the signal bandwidth is $B = 6$ kHz. The carrier frequency is $f_c = 12$ kHz, but can be changed easily to fit the transducer characteristics.

As shown in Fig. 2, each data burst consists of two parts: the preamble and the data payload. The preamble is used for packet detection and synchronization. The data payload contains one or more ZP-OFDM blocks.

The preamble consists of two identical OFDM symbols with one cyclic prefix (CP) as depicted in Fig. 2. Each OFDM symbol has $K_1 = 512$ subcarriers and the CP length is 60% of the OFDM symbol duration. Hence, the preamble has a duration of $(2 + 0.6)K_1/B = 221.9$ ms.

This work is supported by the ONR grant N00014-07-1-0805 (YIP), and the NSF grants CNS-0709005 (CRI) and CNS-0821597 (MRI).

978-1-4244-5222-4/10/$26.00 ©2010 IEEE

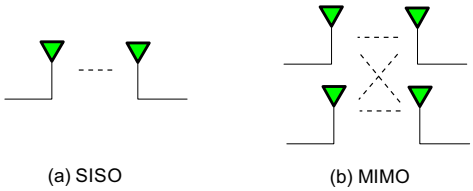


Fig. 1. The SISO and MIMO (2x2) system setups

TABLE I
ZP-OFDM PARAMETERS

Sampling rate	$f_s = 48$ kHz
Center frequency	$f_c = 12$ kHz
Signal bandwidth	$B = 6$ kHz
OFDM block duration	$T = 170.7$ ms
Guard interval	$T_g = 40$ ms
Total block duration	$T' = 210.7$ ms
Subcarrier spacing	$\Delta f = 5.86$ Hz
Number of subcarriers	$K = 1024$
Number of data carriers	$K_d = 672$
Number of pilot carriers	$K_p = 256$
Number of null subcarriers	$K_n = 96$

For the payload, each OFDM symbol has $K = 1024$ subcarriers, and hence the duration of one OFDM block is $T = K/B = 170.7$ ms. The ZP guard interval is of $T_g = 40$ ms, which can be changed depending on the channel delay spread. Among the $K = 1024$ subcarriers, there are $K_n = 96$ null subcarriers and $K_p = 256$ pilot subcarriers, and $K_d = 672$ data subcarriers. For MIMO-OFDM, the pilot subcarriers are split into two non-overlapping sets, one for each data stream [7].

QPSK constellation is used. We consider two channel codes:

- 64-state rate-$1/2$ convolution code, with the code generator as (133,171).
- rate-$1/2$ nonbinary LDPC code over GF(4) [3]. Each symbol over GF(4) is directly mapped to one QPSK constellation point.

Accounting for all the overheads of channel coding, null and pilot subcarriers, and guard intervals, the overall data rate is

$$R = \begin{cases} \frac{1}{2} \cdot \frac{1 \cdot 2 \cdot K_d}{T+T_g} = 3.2 \text{ kb/s}, & \text{SISO-OFDM} \\ \frac{1}{2} \cdot \frac{2 \cdot 2 \cdot K_d}{T+T_g} = 6.4 \text{ kb/s}, & \text{MIMO-OFDM} \end{cases} \quad (1)$$

B. Receiver Overview

Fig. 3 shows the procedure of receiver processing. When started, the device is in the searching state. The codec keeps sampling the audio signal and searches for the high energy section which may indicate the beginning of a transmission. When the energy level of the signal samples reaches a threshold, the device enters the recording state, in which the modem incrementally saves samples for the duration of a transmission. The sampled data are first processed by the synchronization module. If a valid preamble is detected, the receiver will begin data decoding for each OFDM block sequentially until the end of the data burst.

We next provide more details on the receiver synchronization and the block-by-block processing.

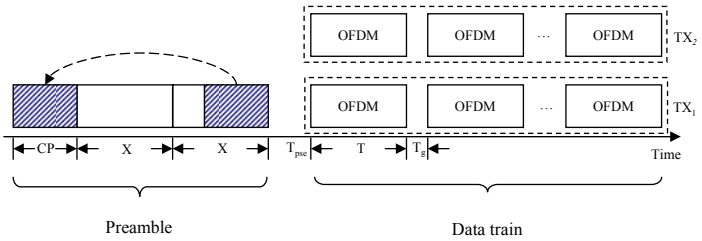


Fig. 2. The data burst of one preamble and multiple OFDM blocks

C. Synchronization

After bandpass filtering, a timing metric *M(d)* is computed through a sliding window correlator [11], [20]:

$$M(d) = \frac{\sum_{m=0}^{N_c-1} \left[y^*(d+m)\, y(d+m+N_c) \right]}{\sqrt{\sum_{m=0}^{N_c-1} |y(d+m)|^2 \cdot \sum_{m=0}^{N_c-1} |y(d+m+N_c)|^2}}. \quad (2)$$

The metric $M(d)$ is the cross-correlation between two adjacent windows each with $N_c = f_s/(B/K_1)$ samples. Each item of the metric $M(d)$ is calculated by multiplying two samples from two adjacent windows and at the same position, and then adding all the products up. Both windows then slide to the right by one sample to calculate the next item of $M(d)$ until it reaches the end of the signal.

In the absence of noise, $|M(d)|$ reaches a maximum magnitude within a plateau due to the preamble structure. The synchronization uses the middle of the plateau as the timing estimate to determine the starting position of the data signal. The middle of the plateau is calculated by finding the shoulders of this plateau (e.g. 90% of the maximum magnitude). Note that this procedure does not require any channel knowledge.

D. Block-by-block Processing

After synchronization, the receiver accumulates the received samples. When one ZP-OFDM block is ready, the following tasks are carried out, as depicted in Fig. 3.

1) Downshift and low pass filtering. After downshifting to the baseband and low pass filtering, the received sequence is downsampled to the baseband rate.
2) Doppler shift compensation. The Doppler shift due to the channel variation is modeled as if it were due to carrier frequency offset (CFO) between the transmitter and the receiver. We compensate the CFO on the OFDM block for each tentative CFO ϵ and evaluate the FFT output on the null subcarriers. The energy of the null subcarriers is used as the cost function $J(\epsilon)$. An estimate of ϵ can be found through
$$\hat{\epsilon} = \arg\min_{\epsilon} J(\epsilon). \quad (3)$$
3) Channel estimation. For SISO OFDM, all pilot subcarriers are used for one data stream. For MIMO OFDM, the pilot subcarriers are divided into two non-overlapping sets, with each data stream assigned one set of uniformly distributed pilot subcarriers [7]. The least-squares (LS) channel estimator is used for each transmitter and receiver pair, which amounts to two FFT operations [1].

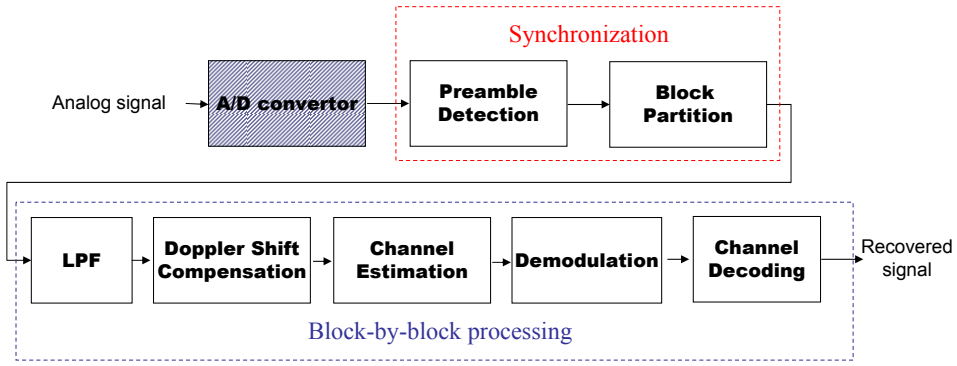


Fig. 3. The procedure of the OFDM receiver

4) Demodulation. For SISO OFDM, no demodulation step is needed. For MIMO OFDM, we use the zero-forcing (ZF) receiver in [7] to separate the two data streams, which amounts to inversion of a 2×2 matrix on each data subcarrier.
5) Decoding. Viterbi algorithm is applied for the decoding of convolutional code. The Min-Sum decoding is used for decoding of the nonbinary LDPC code [21].

To achieve real-time operation in this setting, the receiver needs to decode a received OFDM block within a period of $T + T_g = 210.7$ ms so the current block is processed before the next one arrives.

III. Floating-Point Implementation

We now describe the implementation results on a development board for the TMS320C6713 floating-point DSP. The DSP runs at 225 MHz.

A. Board Description

TMS320C6713 belongs to the TMS320C6000 DSP family, which uses VelociTI, a very-long-instruction-word (VLIW) architecture [22]. It is among the most powerful floating-point DSPs from the TMS320C6000 family. The DSP core has eight independent functional units, including two fixed-point ALUs, four floating-/fixed-point ALUs, and two multipliers. It can execute up to eight instructions simultaneously in one cycle. It has two general-purpose register files, which have 32 32-bit registers in total [23]. It provides 256K bytes on-chip L2 memory which can be configured as unified cache or SRAM.

The board has a plenty of peripherals that facilitate the development. Each DSP board is equipped with one AIC23 codec, one enhanced direct memory access (EDMA) controller, two multichannel buffered serial ports (McBSPs). In our implementation, one serial port McBSP0 is configured as a unidirectional port to control and configure the AIC23 codec. The codec receives serial commands through McBSP0, which set configuration parameters such as volume, sampling rate and data format. Audio data are transferred back from the codec through the other serial port McBSP1, which is configured as a bidirectional serial port. The enhanced DMA controller (EDMA) is configured to take every 16-bit signed audio sample arriving on McBSP1 and store it in a buffer in memory. Once the buffer is full, the EDMA controller generates an interrupt to inform the DSP core that the audio data are ready for processing.

B. Implementation

The majority of the implementation efforts is on optimization, which can be classified into two categories: algorithm optimization and programming optimization. For the algorithm optimization, we adopt proper algorithms for those computationally intensive operations, such as convolution and CFO estimation. For the programming optimization, we apply a few coding strategies, such as loop unrolling and compiler directives, to speed up the processing. Some descriptions are as follows.

- Use the overlap-add and FFT-based algorithm to perform linear convolution. This well-known method can speed up significantly linear filtering operations.
- Use adaptive searching algorithm for CFO estimation. Instead of using a simple linear search, we use adaptive search to reduce the total number of evaluations of the cost function. The basic idea of the adaptive search is as follows. First, a coarse searching (with large interval) is performed to construct a subset of the candidate CFOs in which the target CFO will reside with high probability. Then a bi-sectional search is applied for quick convergence.
- Recursively compute the auto-correlation. Instead of involving all samples inside the sliding window as in (2), the recursive method calculates all correlation items except for the first one using only one sample at the beginning and one entry at the end, as described in [20].
- Efficiently use the hardware resources on the DSP chip such as VLIW architecture and on-chip memory.

C. Processing Time

We use the 32-bit timer on the DSP board to measure the execution time. The per-block processing time of one OFDM block of duration $T + T_g = 210$ ms is shown in Table II for SISO, and in Table III for MIMO settings. It takes about twice the time to decode in the MIMO case as that in the SISO case as two channels are processed to recover two data streams simultaneously.

TABLE II
PROCESSING TIME OF ONE SISO OFDM BLOCK OF 210 MS

	Conv. code, 64-state	LDPC, GF(4)
Downshifting, LPF	12.61 ms	
CFO estimation	12.20 ms	
Channel estimation	0.84 ms	
Other	2.52 ms	
Decoding	9.80 ms	21.55 ms
Total per-block time	37.97 ms	49.72 ms

TABLE III
PROCESSING TIME OF ONE MIMO OFDM BLOCK OF 210 MS

	Conv. code, 64-state	LDPC, GF(4)
Downshifting, LPF	25.14 ms	
CFO estimation	24.34 ms	
Channel estimation	1.42 ms	
ZF equalization	2.4 ms	
Other	3.79 ms	
Decoding	19.90 ms	43.56 ms
Total per-block time	76.99 ms	100.65 ms

With either convolutional or nonbinary LDPC codes, real-time decoding is achieved with impressive margins. For channel decoding alone, it takes more than twice the time for the nonbinary LDPC code than the convolutional code. Using LDPC instead of convolutional coding, the overall processing time per OFDM block increases by 30%, however, the performance is considerably enhanced as shown next.

D. BLER Performance

To compare the performance of convolutional and LDPC codes, we recorded 10 data bursts from the experiment setup in a water tank as depicted in Fig. 4, where each data burst has 10 ZP-OFDM blocks as shown in Fig. 5. In the MIMO setup with two transmitters and two receivers, there exist four channels. The channel impulse responses measured on one receiver are plotted in Fig. 6, corresponding to two transmitters.

Let z_m denote the FFT output on the mth subcarrier. We add artificially generated noise samples (say v_m at the mth subcarrier) directly to z_m to obtain $z'_m = z_m + v_m$, based on which channel estimation and data detection are carried out. The signal-to-noise ration (SNR) measured at the pilot subcarriers is:

$$\text{Pilot SNR} = \frac{\mathrm{E}_{m\in P}\left\{|z'_m|^2\right\} - \mathrm{E}_{m\in N}\left\{|z'_m|^2\right\}}{\mathrm{E}_{m\in N}\left\{|z'_m|^2\right\}}, \quad (4)$$

where $_P$ and $_N$ are the sets of pilot and null subcarriers, respectively. Figs. 7 and 8 show the BLER performance of SISO and MIMO settings. For each point plotted, at least a total of 1000 blocks are decoded, with at least 10 different noise realizations added to those 100 recorded blocks. The two data streams in Fig. 8 have different performance due to different channel strengths as shown in Fig. 6. At BLER of 10^{-2}, nonbinary LDPC codes outperform convolutional codes by about 2 dB, which agrees with the simulation results in [3]. Such a performance improvement justifies the use of nonbinary LDPC codes despite the increase of the decoding time.

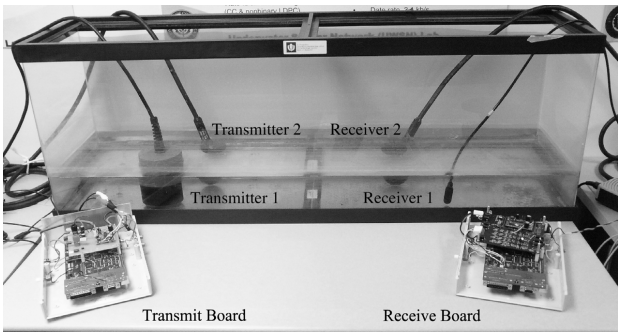


Fig. 4. The experimental setup in a water tank

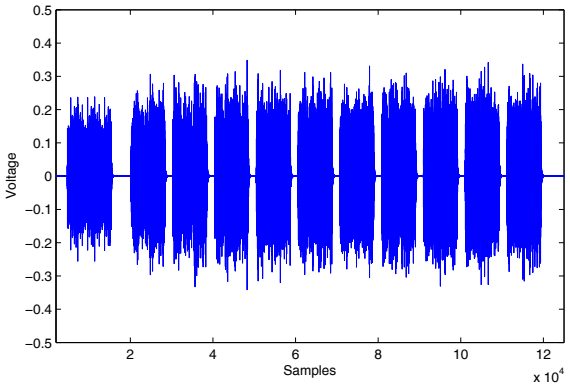


Fig. 5. One recorded data burst of one preamble followed by 10 data blocks

IV. FIXED-POINT IMPLEMENTATION

The DSP for fixed-point implementation is TMS320C6416. Like TMS320C6713, it belongs to the TMS320C6000 family and has a VLIW architecture [22]. TMS320C6414 can run at much higher clock rates than floating-point DSPs so it can reduce the execution time of many applications. In addition, the fixed-point implementation can facilitate future FPGA or ASIC implementations where fixed-point operations are adopted for smaller area and lower energy consumption. On the other hand, compared to the floating-point implementation, the fixed-point implementation may suffer from some loss on the error-rate performance, which needs to be investigated.

Most features on TMS320C6416 are similar to those on TMS320C6713. The improvements include higher clock rate (1 GHz), larger on-chip L2 memory (1 Mbytes), and more registers (64 32-bit registers in total).

A. Implementation

The major difference between the fixed- and floating-point implementations is the representation of values. Most optimizations, especially those at the algorithm level, are the same in both implementations.

Since the size of registers on TMS320C6416 is 32 bits, we use 32-bit fixed-point numbers. To make the data compatible with the C64x DSP library, we adopt the Q.31 format, where the most significant bit is the sign bit, and the remaining 31 bits represent the fraction part of a number. The Q.31 format can only represent numbers between $[-1, 1)$. Consequently, proper scaling is required during the processing. For example,

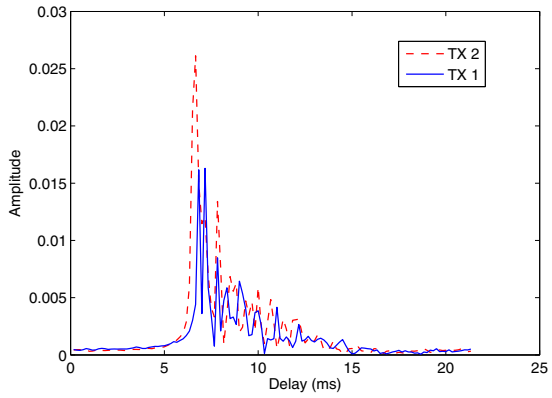


Fig. 6. Channel impulse responses measured on one receiver

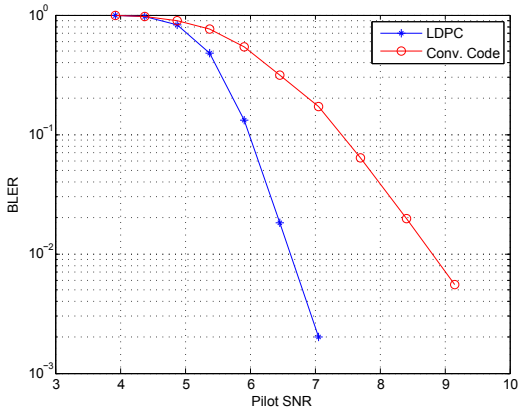


Fig. 7. BLER as a function of pilot SNR, SISO-OFDM

the sampled input signals are naturally normalized to $(-1, 1)$. Scaling schemes are carefully designed for FFT and IFFT to avoid overflow. A proper scaling factor is dynamically determined before Viterbi decoding. For the multiplication intensive steps like the log-likelihood ratio computation, scaling factors are carefully chosen to prevent significant precision loss.

B. Processing Time

The processing time is recorded in Table IV. We can see that the decoding time of fixed point implementation is about 1/3 of that of floating point implementation. This reduction is impressive. However, the fixed-point DSP is running at 1 GHz, compared to 225 MHz for the floating point DSP.

TABLE IV
PROCESSING TIME OF FLOATING- AND FIXED-POINT IMPLEMENTATION

	SISO OFDM	MIMO OFDM
Floating-point	38.06 ms	76.65 ms
Fixed-point	12.69 ms	25.14 ms

C. BLER Performance

To evaluate the performance loss due to fixed-point implementation, we add noise directly to the recorded passband data. All modules in the block-by-block processing in Fig. 3 are involved in the fixed-point implementation. Here we use the convolutional code for illustration.

Fig. 9 shows the pilot SNR and block error rate (BLER) performance of SISO OFDM at different input SNR levels, and Fig. 10 shows the counterparts for MIMO-OFDM. The differences between fixed- and floating-point implementations are negligible. Since the input data are sampled with 16 bits, the 32-bit fixed-point processing has enough room to avoid overflow without dropping meaningful bits. Certainly, programming effort is much increased for fixed-point implementation as compared to the floating-point alternative.

V. CONCLUSIONS

In this paper, we reported implementation results of SISO and MIMO OFDM acoustic modems using both floating- and fixed-point DSP platforms. Real-time decoding was achieved with impressive margins in our settings. Nonbinary LDPC coding improves the performance relative to convolutional coding, at the cost of increased decoding time. Fixed point implementation drastically reduces the processing time compared with the floating-point counterpart, running at a higher clock frequency. The implementation effort in this paper provides some useful insights towards realizing a practical OFDM modem for high-data-rate and reliable underwater acoustic communications.

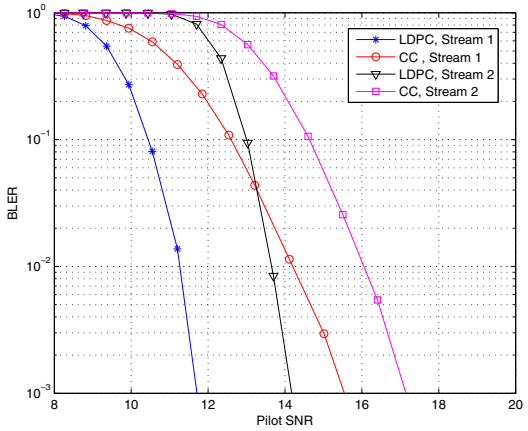


Fig. 8. BLER as a function of pilot SNR, MIMO-OFDM

ACKNOWLEDGMENT

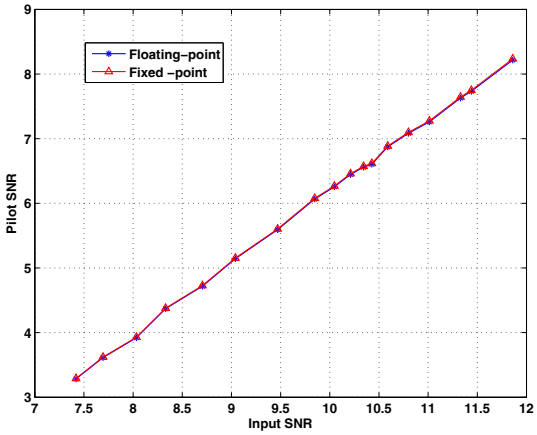
We thank Ms. Janny Liao for her help on various implementation issues during the modem prototype development.

REFERENCES

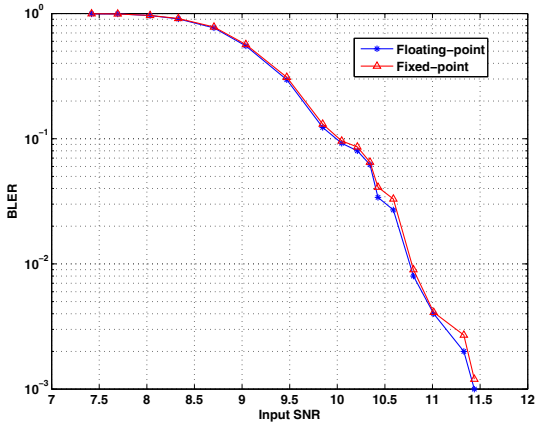
[1] B. Li, S. Zhou, M. Stojanovic, L. Freitag, and P. Willett, "Multicarrier communication over underwater acoustic channels with nonuniform Doppler shifts," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 2, pp. 198–209, Apr. 2008.

[2] M. Stojanovic, "Low complexity OFDM detector for underwater channels," in *Proc. of OCEANS Conference*, Boston, MA, Sept. 18-21, 2006.

[3] J. Huang, S. Zhou, and P. Willett, "Nonbinary LDPC coding for multicarrier underwater acoustic communication," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 9, pp. 1684–1696, Dec. 2008.

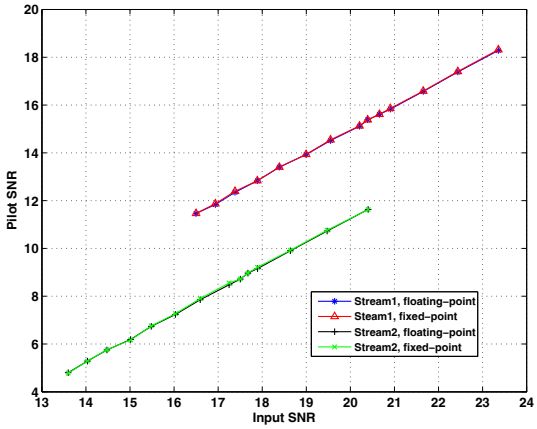


(a) Pilot SNR vs input SNR

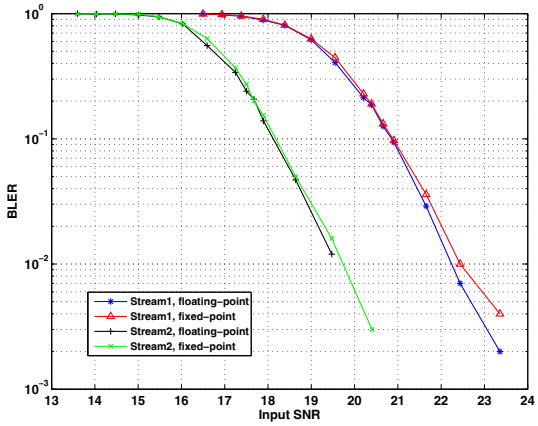


(b) BLER vs input SNR

Fig. 9. The comparison of fixed- and floating-point implementation of SISO-OFDM



(a) Pilot SNR vs input SNR



(b) BLER vs input SNR

Fig. 10. The comparison of fixed- and floating-point implementation of MIMO-OFDM, 2 transmitters and 2 receivers

[4] G. Leus and P. van Walree, "Multiband OFDM for covert acoustic communications," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 9, pp. 1662–1673, Dec. 2008.

[5] T. Kang and R. Iltis, "Iterative carrier frequency offset and channel estimation for underwater acoustic OFDM systems," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 9, Dec. 2008.

[6] K. Tu, D. Fertonani, T. Duman, and P. Hursky, "Mitigation of intercarrier interference in OFDM systems over underwater acoustic channels," in *Acoustics'08 Conference*, Paris, France, July 2008.

[7] B. Li, J. Huang, S. Zhou, K. Ball, M. Stojanovic, L. Freitag, and P. Willett, "MIMO-OFDM for high rate underwater acoustic communications," *IEEE Journal of Oceanic Engineering*, vol. 34, no. 4, Oct. 2009.

[8] P. Carrascosa and M. Stojanovic, "Adaptive MIMO detection of OFDM signals in an underwater acoustic channel," in *Proc. of MTS/IEEE OCEANS Conference*, Quebec City, Canada, Sept. 15-18, 2008.

[9] Y. Emre, V. Kandasamy, T. M. Duman, P. Hursky, and S. Roy, "Multi-input multi-output OFDM for shallow-water UWA communications," in *Acoustics'08 Conference*, Paris, France, July 2008.

[10] Z. Yan, J. Huang, and C. He, "Implementation of an OFDM underwater acoustic communication system on an underwater vehicle with multiprocessor structure," *Front. Electr. Electron. Eng. China*, 2007.

[11] H. Yan, S. Zhou, Z. Shi, and B. Li, "A DSP implementation of OFDM acoustic modem," in *Proc. of the ACM International Workshop on UnderWater Networks (WUWNet)*, September 14, 2007.

[12] Benthos, Inc., "Fast and reliable access to undersea data," http://www.benthos.com.

[13] LinkQuest, Inc., "Underwater acoustic modems," http://www.link-quest.com/html/uwm_hr.pdf.

[14] DSPCOMM, "Underwater wireless modem," http://www.dspcomm.com.

[15] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: An acoustic communications and navigation system for multiple platforms," in *Proceeding of OCEANS*, 2005.

[16] E. Sozer and M. Stojanovic, "Reconfigurable acoustic modem for underwater sensor networks," in *Proc. First ACM International Workshop on Underwater Networks*, Los Angeles, CA, Sept. 2006.

[17] T. Fu, D. Doonan, C. Utley, R. Iltis, R. Kastner, and H. Lee, "Design and development of a software-defined underwater acoustic modem for sensor networks for environmental and ecological research," in *Proc. of OCEANS*, Sept. 2006.

[18] J. Wills, W. Ye, and J. Heidemann, "Low-power acoustic modem for dense underwater sensor networks," in *Proc. of WUWNet*, Los Angeles, CA, Sept. 2006.

[19] B. Benson, G. Chang, D. Manov, B. Graham, and R. Kastner, "Design of a low-cost acoustic modem for moored oceanographic applications," in *Proc. of WUWNet*, Los Angeles, CA, Sept. 2006.

[20] S. Mason, C. R. Berger, S. Zhou, and P. Willett, "Detection, synchronization, and Doppler scale estimation with multicarrier waveforms in underwater acoustic communication," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 9, pp. 1638–1649, Dec. 2008.

[21] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. of International Conference on Communications*, Paris, France, June 2004, pp. 772–776.

[22] *TMS320C6000 CPU and Instruction Set reference guide*, Texas Instruments, 2006.

[23] *TMS320C6000 DSP peripherals overview reference guide*, Texas Instruments, 2003.