

A DSP Implementation of OFDM Acoustic Modem

Hai Yan, Shengli Zhou, Zhijie Jerry Shi, and Baosheng Li
Underwater Sensor Network Lab
University of Connecticut
371 Fairfield Way, Storrs, CT, 06269, USA
{hai.yan,shengli,zshi,baosheng}@engr.uconn.edu

ABSTRACT

The success of multicarrier modulation in the form of OFDM in radio channels illuminates a path one could take towards high-rate underwater acoustic communications, and recently there are intensive investigations on underwater OFDM. In this paper, we implement the acoustic OFDM transmitter and receiver design of [4, 5] on a TMS320C6713 DSP board. We analyze the workload and identify the most time-consuming operations. Based on the workload analysis, we tune the algorithms and optimize the code to substantially reduce the synchronization time to 0.2 seconds and the processing time of one OFDM block to 1.7 seconds on a DSP processor at 225 MHz. This experimentation provides guidelines on our future work to reduce the per-block processing time to be less than the block duration of 0.23 seconds for real time operations.

Categories and Subject Descriptors

C.3 [Special-purpose and application-based systems]: Micro-processor/microcomputer applications; B.4.1 [Hardware]: Data communications devices

General Terms

Algorithms, Design, Experimentation

Keywords

Multicarrier, OFDM, DSP, Acoustic Modem

1. INTRODUCTION

Although land-based wireless sensor networks have proliferated in many applications, the use of underwater sensor networks has been limited. Nonetheless, there has been a growing interest in building distributed and scalable underwater wireless sensor networks (UWSN) that will bring significant advantages and benefits in a wide spectrum of underwater applications, such as ocean observation for scientific exploration, commercial exploitation, coastline protection and target detection in military events. Improving underwater acoustic communications among distributed sensor nodes is one of the major design challenges for building UWSNs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WUWNet'07, September 14, 2007, Montréal, Québec, Canada.
Copyright 2007 ACM 978-1-59593-736-0/07/0009 ...\$5.00.

Existing coherent underwater communication uses single carrier transmission and relies on linear or non-linear equalization techniques to suppress inter-symbol interference [1]. As the data rate increases, the symbol duration decreases, and thus a channel with the same delay spread contains more channel taps when converted to the baseband discrete-time model. This imposes great challenges for channel equalization, whose complexity will prevent substantial rate improvement with existing single-carrier approaches.

Multicarrier modulation in the form of orthogonal frequency division multiplexing (OFDM) has been quite successful in broadband wireless communication over radio channels, e.g., wireless local area networks (IEEE 802.11a/g/n). Motivated by this fact, researchers have long attempted to apply OFDM in underwater acoustic channels. Recently, we have seen intensive investigations on underwater OFDM, including [2] on a low-complexity adaptive OFDM receiver, and [3, 4] on a pilot-tone based block-by-block receiver. As a senior design project, an undergraduate team at University of Connecticut has demonstrated multicarrier OFDM transmission and reception in air and in a water tank, where the algorithms in [3, 4] are implemented by Matlab programs in two laptops [5].

In this paper, we implement the OFDM transmission and reception algorithms of [3, 4] on a TI TMS320C6713 DSP board with a processor running at 225 MHz. In-air communications are successfully tested. We analyze the workload and identify the most time-consuming operations. Based on the workload analysis, we optimize the algorithms and reduce the synchronization time to 0.2 seconds and the processing time to 1.7 seconds per OFDM block. Further work is needed to reduce the per-block processing time to be less than 0.23 seconds for real-time operations.

2. RELATED WORK

There are quite a few acoustic modems available, including commercial products such as [6–8], a model widely used in the research community [9], and experimental designs such as [10–13]. The designs in [6, 12, 13] are based on non-coherent frequency-shift-keying (FSK), and those in [7, 8, 10] are based on spread spectrum; all of them inherently have low data rate. The reconfigurable acoustic modem (rModem) of [11] has a flexible structure that can facilitate quick prototyping of different algorithms. The Micro-Modem of [9] 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). Implementation of an acoustic modem based on multicarrier OFDM is not available so far, which is the focus of this paper.

3. TRANSMITTER DESIGN

The overall data flow is depicted in Fig. 1, where the synchronization preamble is followed by data transmission. The signal

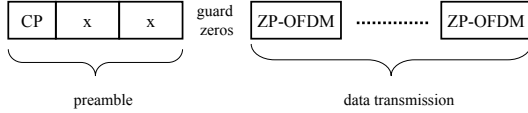


Figure 1: The data flow of preamble and data transmission

bandwidth is $B = 5$ kHz, centered around the carrier frequency $f_c = 12.5$ kHz. The sampling rate is $f_s = 44.1$ kHz.

We use the synchronization preamble adopted in IEEE802.11 standards, where two identical OFDM symbols with one cyclic prefix (CP) are transmitted as depicted in Fig. 1. 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 $(2 + 0.6)/(B/K_1) = 0.266$ s.

For data transmission, we use zero-padded (ZP) OFDM [3, 4]. Each OFDM symbol has $K = 1024$ subcarriers, and hence the OFDM duration is of $T = 1/(B/K) = 0.2048$ s. The ZP guard interval is of $T_g = 25$ ms. Out of $K = 1024$ subcarriers, there are $K_n = 56$ null subcarriers and $K/4 = 256$ pilot subcarriers. QPSK is used on each data subcarrier. Over a bandwidth of 5 kHz, the raw data rate (without channel coding) is

$$R = \frac{2(K - K/4 - K_n)}{T + T_g} = 6.2 \text{ kbps} \quad (1)$$

where the rate loss due of the ZP guard is accounted for. We include a 16-state rate 1/2 convolutional code, that leads to a coded data rate of 3.1 kbps. Puncturing the convolutional code can increase the coded data rate, if desired.

For real-time operation, the receiver has to decode an OFDM block within one OFDM symbol duration, which is $T + T_g = 0.2298$ s in our case. In this experimentation, we test one data burst of one preamble followed by one OFDM data block.

4. RECEIVER ALGORITHMS

4.1 Synchronization

As shown in Fig. 1, the last portion of the first OFDM symbol in the preamble will serve as CP for the second OFDM symbol. Hence, channel propagation will have the same impact on each of these two OFDM symbols. Let $y(n)$ denote the received samples in the baseband discrete-time representation. The receiver calculates the following timing metric via a sliding window correlator:

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

The metric $M(d)$ is the cross-correlation between two adjacent windows each with $N_c = f_s/(B/K_1)$ samples. In the absence of noise, $|M(d)|$ reaches a maximum magnitude within a plateau due to the CP structure. The middle of the plateau is determined by finding the shoulders of this plateau and using the middle as the timing estimate. Note that this timing algorithm does not require any channel knowledge.

4.2 CFO estimation

We will use the block-by-block receiver structure of [3, 4]. The channel time variation within each OFDM symbol is modeled as a multiplicative process $e^{j2\pi\epsilon t}$ on the received signal after the multipath propagation, where ϵ is termed as carrier frequency offset (CFO). We use null subcarriers to facilitate the finding of the CFO. The idea is as follows. We collect the baseband samples for each

OFDM block. For each tentative ϵ , we compensate the CFO on the OFDM block and evaluate the FFT output on the null subcarriers. The energy of the null subcarriers is used as the cost function $J(\epsilon)$. If the receiver compensates the data samples with the correct CFO, the null subcarriers will not see the intercarrier interference (ICI) from neighboring data subcarriers. Hence, an estimate of ϵ can be found through

$$\hat{\epsilon} = \arg \min_{\epsilon} J(\epsilon), \quad (2)$$

which can be solved via one-dimensional search for ϵ .

4.3 Channel estimation

After CFO compensation, the ICI is greatly reduced. To estimate the channel frequency response, we use $N_p = K/4$ pilot tones. The N_p pilot symbols are equally spaced within K subcarriers, and they are PSK signals with unit amplitude [3, 4]. Then, the least square (LS) channel estimation solution does not involve matrix inversion, and can be implemented by a $K/4$ -point IFFT [3, 4].

4.4 Data demodulation and decoding

OFDM data demodulation amounts to scalar inversion on each data subcarrier. Viterbi algorithm is applied for decoding.

5. DSP IMPLEMENTATION

In this section, we describe the implementation of the OFDM acoustic modem using an off-the-shelf DSP development board. Figure 2 shows the lab setting we used to test the acoustic modem.

5.1 TMS320C6713 DSP board overview

We used the TMS320C6713 digital signal processor (DSP) board. C6713 is a recent version of the TMS320 family of DSPs provided by Texas Instrument. It uses VelociTI, a high-performance very-long-instruction-word (VLIW) architecture. The C6713 core has eight independent functional units: 2 fixed-point ALUs, 4 floating-/fixed-point ALUs, and 2 multipliers. So it can execute up to eight 32-bit instructions per cycle. There are two general-purpose register files, A and B, which have 32 32-bit registers in total [14].

C6713 has a cache-based memory architecture. At level-one (L1), program (or instruction) and data caches are separated, each 4KB. They are not included in the memory map and are enabled at all times. The level-2 (L2) cache is configurable. There is a total of 256KB L2 memory and up to 64KB can be configured as a unified L2 cache. The rest of L2 memory is managed by program. For example, when the L2 cache is disabled, program can manage all of the 256KB L2 RAM. In our implementation, the L2 cache is configured to be 64KB. Therefore, the internal memory managed by program is 192KB.

The TMS320C6713 board also provides plenty of peripherals, including an enhanced direct memory access (EDMA) controller, two multichannel buffered serial ports (McBSPs), two 32-bit timers, and a power-down logic.

5.2 A/D interface

One of the key components of the acoustic modem is the audio signal input/output module. The C6713 development board we used has a built-in module for sampling and generating audio signals. The tasks are handled by the integrated TLV320AIC23 codec, a high-performance stereo audio codec with highly integrated analog functionality [15]. Data-transfer word lengths of 16, 20, 24, and 32 bits, with sampling rates from 8 kHz to 96 kHz, are supported [15].

The operation mode of AIC23 can be programmed with a set of



Figure 2: The DSP based prototype for acoustic OFDM modem

control registers. In our implementation, we select the microphone as the audio signal source and set the sampling rate to 44.1 kHz.

To sample audio signals, one of the multichannel buffered serial ports (McBSPs) is configured to connect the AIC23 codec. The audio data is transferred between the codec and the internal L2 memory through the enhanced direct memory access (EDMA) channel. To save the raw audio data from the AIC23 codec continuously, the commonly-known double-buffering method is used. When one of the two buffers is filled, a DMA interrupt is initiated and the data is passed to the interrupt service routine (ISR) and then is processed. At the same time, the codec keeps sampling and saves data into the other buffer. So data sampling and processing can be done simultaneously and no incoming signals are missed even if the DSP is processing previously received data.

5.3 Implementation issues

The state diagram of the OFDM acoustic modem is shown in Figure 3. When started, the device is in the searching state. The AIC23 codec keeps sampling the audio signal and searching 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. Then the sampled data are processed in the auto-correlation state to find the synchronization signal. If the synchronization is successful, the sampled data will be further processed by the CFO/channel estimation and data demodulation modules.

C6713 has limited memory resources. The 192KB internal L2 memory cannot hold all the data. In our implementation, we put initialization parameters and constants to external flash memory. We also manage the internal L2 memory carefully. If some data are accessed repeatedly, we load them into L2 memory first.

The total code size including the library routines is around 87 KB. The total memory used by the program is about 1.83 MB.

6. CODE OPTIMIZATION

The OFDM modem prototype works well in the air testings. The in-air testing results are similar to those based on Matlab implementations in [5]. We hence expect similar in-water testing results as those in [5] since the same algorithms are used.

Our initial implementation is to translate the Matlab programs in [5] to C programs that can run on the DSP board. This initial version takes on the order of minutes for the receiver processing. To identify the time-consuming components, we profiled the execution of the program. The results are shown in Table 1. The results were generated by the standalone simulator load6x with the `-g` option, which sets the simulator in the profiling mode. In the

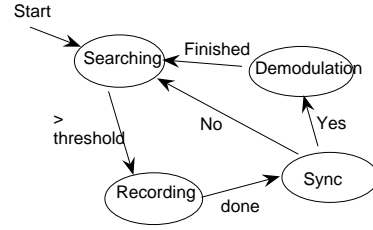


Figure 3: State diagram of acoustic modem

table, the count column lists the number of times a function was called. The instruction column lists the total number of instructions spent inside a function, including calls to other functions. Synchronization mainly depends on three functions, `array_conv()`, `array_mul_sum()`, and `array_sum_sq()`, which contribute most of the execution instructions. The per-OFDM-block processing includes functions such as `complex_conv`, `complex_exp`, `CFO_estimation`, and `Viterbi`.

Based on the workload analysis, we optimize the algorithms and the DSP code. We achieved a speedup of 45 times in terms of the total number of instructions executed. The new performance results are shown in Table 2.

The main steps in our code optimization are the following.

- Use the classical overlap-add FFT-based method to replace the straightforward linear convolution in bandpass and low-pass filtering operations. This step leads to a speedup of 40 times for the function `array_conv()`.
- Compute summation of entries within a sliding window *recursively* in `array_mul_sum()` and `array_sum_sq()` functions. Instead of involving all entries inside the sliding window, the recursive method only involves one entry at the beginning and one entry at the end for computation.
- The receiver initially performs a CFO line search with a step size 0.2 Hz within the $[-20, 20]$ Hz region. We now use a coarse line search with a step size 4 Hz followed by a bisectional method that reduces the search interval by half for each iteration. The new method reduces the total number of FFTs from 200 times to around 20 times, while reaching the same CFO resolution. We have empirically observed that there are only two to four local minimas within the CFO range of $[-20, 20]$ Hz, and the initial coarse search can effectively avoid local minimas.
- It turns out that preparing a vector of complex exponentials in the function `complex_exp` is very slow when calling standard routines such as `sin()` and `cos()`. We now recursively compute $e^{j\alpha n}$ by multiplying the previous value of $e^{j\alpha(n-1)}$ with an increment coefficient $e^{j\alpha}$, whose value is stored and re-used multiple times. This step avoids calling `sin()` and `cos()` routines, and eliminates the need for `complex_exp`, which is computationally expensive.
- In addition to algorithm improvements, we manually tune the code to fully utilize the hardware resources in the C6713 DSP board. Several core inner loops have been unrolled to match the pipeline, and the patterns of memory accesses have been changed to reduce the overhead.

At the DSP frequency of 225 MHz, the total execution time corresponding to Table 2 is around 1.9 seconds to process the data burst with one block of data, where the synchronization takes 0.2

Table 1: Program profiling before optimization

Function	Count	Instructions
Synchronization	1	839682258
array_conv	2	219315479
array_mul_sum	8000	229744000
array_sum_sq	16000	459488000
Per-block processing	1	1284555370
complex_conv	1	22255590
CFO_estimation	1	1179178995
complex_exp	204	1049251031
FFT	204	8362616
Viterbi	1	1264049
Total		2125522768

Table 2: Program profiling after optimization

Function	Count	Instructions
Synchronization	1	3556475
array_conv	2	5351551
array_mul_sum	1	7214
array_sum_sq	2	23190
Per-block processing	1	43282059
complex_conv	1	3703686
CFO_estimation	1	33070013
FFT	23	884811
Viterbi	1	1264049
Total		46858673

seconds and processing one OFDM block takes 1.7 seconds. The division of time on different receiver modules is shown in Figure 4. Note that preprocessing of the OFDM data block (including band-pass filtering, carrier downshifting, and low pass filtering) and CFO estimation occupy about 73% of the total time, and decoding is only around 14% of the total time.

In this work, we only use an off-the-shelf DSP board to evaluate the performance and complexity of OFDM algorithms. Currently processing an OFDM block of duration 0.23s needs about 1.7 seconds. Hence, one order of magnitude speedup is needed to meet real-time operation requirements. In our future work, we will pursue the following options.

- We now use double-precision 64-bit floating point operations. Single-precision floating point operations may speed up the process considerably with only minimal performance degradation.
- We can pursue a hybrid DSP/FPGA-based solution, which moves time-consuming tasks into FPGA.

7. CONCLUSION

In this paper, we implemented the coherent OFDM algorithm on a TMS320C6713 DSP board for acoustic communications. Based on the program profiling, we identified the time-consuming operations of the OFDM algorithms. We optimized the code and achieved significant speedups. We reduced the processing time per OFDM block to about 1.7 seconds. Since the duration of an OFDM block is 0.23 seconds, the current implementation does not meet the real-time operation requirements yet. In the future, we are motivated to pursue a hybrid DSP/FPGA-based solution to construct a real-time OFDM modem.

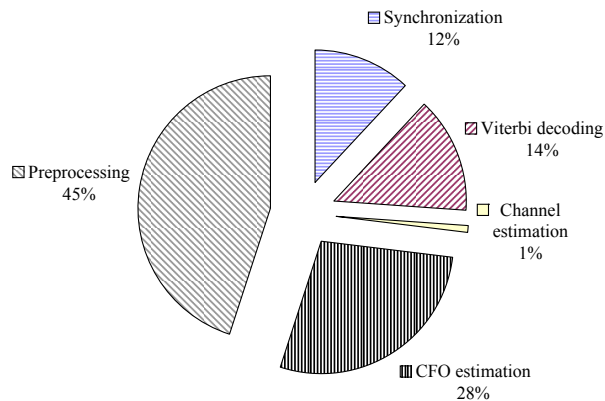


Figure 4: Pie chart of the total execution time

8. ACKNOWLEDGEMENT

This work was supported by ONR grant N00014-07-1-0805.

9. REFERENCES

- [1] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, Jan. 2000.
- [2] M. Stojanovic, "Low complexity OFDM detector for underwater channels," in *Proc. of OCEANS*, Sept. 2006.
- [3] B. Li, S. Zhou, M. Stojanovic, and L. Freitag, "Pilot-tone based ZP-OFDM demodulation for an underwater acoustic channel," in *Proc. of OCEANS*, Sept. 2006.
- [4] B. Li, S. Zhou, M. Stojanovic, L. Freitag, and P. Willett, "Non-uniform Doppler compensation for zero-padded OFDM over fast-varying underwater acoustic channels," in *Proc. of OCEANS*, June 2007.
- [5] S. Mason, R. Anstett, N. Anicette, and S. Zhou, "A broadband underwater acoustic modem implementation using coherent OFDM," in *Proc. of National Conference for Undergraduate Research*, April 2007.
- [6] Benthos, Inc. *Fast and reliable access to undersea data*. <http://www.benthos.com/pdf/Modems/ModemBrochure.pdf>.
- [7] LinkQuest, Inc. *Underwater acoustic modems*. http://www.link-quest.com/html/uwm_hr.pdf
- [8] DSPCOMM, *Underwater wireless modem*. <http://www.dspcomm.com>.
- [9] 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.
- [10] 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.
- [11] E. Sozer and M. Stojanovic, "Reconfigurable acoustic modem for underwater sensor networks," in *Proc. of WUWNet*, Sept. 2006.
- [12] J. Wills, W. Ye, and J. Heidemann, "Low-power acoustic modem for dense underwater sensor networks," in *Proc. of WUWNet*, Sept. 2006.
- [13] 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*, Sept. 2006.
- [14] Texas Instruments, "TMS320C6000 DSP peripherals overview reference guide," Texas Instruments, 2003.
- [15] Texas Instruments, "TLV320AIC23 stereo audio CODEC, 8- to 96-kHz, with integrated headphone amplifier: Data manual," July 2001.