Hardware Implementation of Symbol Synchronization for Underwater FSK

Ying Li, Xing Zhang

Bridget Benson, Ryan Kastner

Abstract— **Symbol synchronization is a critical component in the design of an underwater acoustic modem. Without accurate symbol synchronization, higher bit error rates incur thus reducing the reliability and quality of service of the wireless network. This paper provides a practical description of the design choices and hardware implementation details required to build a compact and efficient symbol synchronizer suitable for a short-range, low-power underwater FSK acoustic modem. Experimental results show the design meets the timing requirements of the underwater modem and provides accurate synchronization while consuming only 0.240W power in a Spartan3 xc3s2000 FPGA.**

I. INTRODUCTION

Symbol synchronization, the ability of the receiver to synchronize to the first symbol of an incoming data stream [1], is a critical component in the design of an underwater acoustic modem. When the modem receiver obtains an input stream, it must be able to find the start of the data sequence to set accurate sampling and decision timing for subsequent demodulation. Symbol synchronization is particularly important in the underwater acoustic channel as the channel can be highly variable and plagued by multipath, variable noise levels, path-loss, Doppler shifts, and long propagation delays [2-3]. Without accurate symbol synchronization, higher bit error rates incur thus reducing the reliability and quality of service of the wireless network.

As the goal of our overall underwater acoustic modem, described in [4], is to provide a low-cost, low power alternative to existing research [5-10] and commercial modems for the application of small, dense, underwater sensor networks, it is imperative that our symbol synchronizer is simple and compact to allow it to fit into a low-cost, low-power device.

A few research papers describe underwater symbol synchronization techniques, but none provide a hardware based implementation suitable for low-cost, low-power, short range acoustic modem. Iltis et. al [6] and Sun et al. [11] focus on describing symbol synchronization techniques to overcome the effects of multipath and Doppler shift in complicated underwater channels respectively. These techniques are computationally complex and over suited for the simpler short-range underwater channel. Jurdak et al. [12] present a simpler symbol synchronization technique known as S4 (short signature synchronization symbol), but focus on a software implementation of their method.

Thus, the purpose of this paper is to describe the design choices, implementation details, and control arrangement required to build a compact, hardware based, real-time symbol synchronizer suitable for an inexpensive and power efficient underwater FSK acoustic modem. The major contributions of this paper include:

- A description of the design considerations necessary to design a symbol synchronization method suitable for an FSK based underwater acoustic receiver
- A discussion of the hardware design and control details required to reduce the area and power of the symbol synchronizer
- A complete hardware implementation of a compact, real-time, accurate symbol synchronizer

The rest of this paper is organized as follows: Section 2 describes our overall acoustic receiver design and its parameters that govern many of design choices for symbol synchronization. Section 3 describes a simple process for symbol synchronization and the design choices we made to make it practical for underwater FSK. Section 4 describes the hardware implementation details to make a compact implementation of our symbol synchronizer. Section 5 describes the synchronizer's control scheme for real-time accurate synchronization and section 6 presents the area, timing, and synchronization results. We conclude in section 7.

II. UNDERWATER FSK RECEIVER DESIGN

Figure 1 illustrates a block diagram of our underwater acoustic receiver design. The input signal *adc_in* is the received analog signal which consists of a modulated wave form (when data is present) and noise. The following analog

to digital converter (ADC) and Digital Down Converter (DDC) recover the signal to the digital 'baseband' according to the FSK modulation scheme and known carrier frequency. A synchronizer is then required to locate the symbol synchronization point. The demodulator block is disabled by not giving signal from a DMUX until it obtains a valid index from the synchronizer. This index corresponds to the synchronization point allows the demodulator to begin demodulation with accurate symbol timing. The demodulator adopts the FSK demodulation scheme described in [4] making use of two bandpass filters (one centered on the mark frequency and one centered on the space frequency) to decode the sequence. The decoded bit stream *data_out* is then sent to the host computer and translated to a readable message.

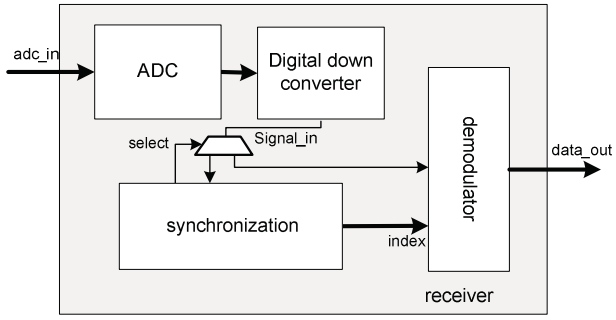

Figure 1. Block diagram of underwater acoustic FSK receiver

The modem makes use of the frequency and rate parameters shown in Table 1. These parameters govern the timing constraints of the symbol synchronization technique described in the following section.

TABLE I. MODEM TIMING CONSTRAINTS AND PARAMETERS

Properties	**Assignment**
Modulation	FSK
Carrier frequency	35 KHz
Mark frequency	1 KHz
Space frequency	2 KHz
Symbol duration	5 ms
Sampling Clock	800 KHz
Demodulator Clock	16 KHz

III. SYMBOL SYNCHRONIZATION FOR UNDERWATER FSK

The most common and simple symbol synchronization approach relies on the transmission of a predefined sequence of symbols, often referred to as a training, or reference sequence. The transmitter sends a packet that begins with the reference sequence and the receiver compares the received sequence with the known reference sequence in order to locate the start of the packet [1, 13-14]. The comparison is effectively performed by correlating the received sequence with the reference sequence when a new sample is received. When the reference and receiving sequence exactly align with each other, the correlation result reaches a maximum value and the synchronization point is located (Figure 2).

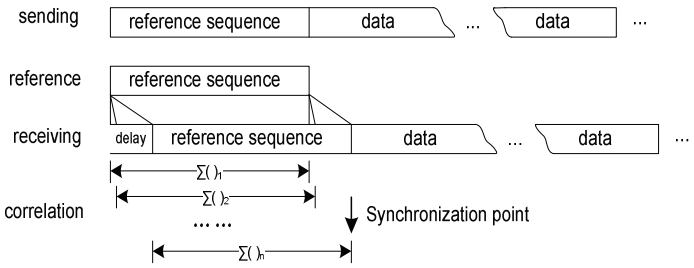

Figure 2: Process of symbol synchronization

Though this synchronization method is straight forward in theory, there are many design choices that must be considered before implementing the algorithm:

A. How can the peak correlation result be detected effectively?
B. What reference code is appropriate for the application?
C. How will the reference code be represented?
D. How long should the reference code be?

A. Peak Detection

Typical methods to detect a correlation peak involve setting a correlation threshold where the correlation result only rises above the threshold when the receiving sequence and reference sequence are perfectly aligned. The threshold must be large enough to minimize the number of false alarms (the system detects the reference code is present when it is not), but small enough to guarantee the peak will still be detected when the signal level is low.

The underwater acoustic channel is highly variable and plagued by variable noise levels, path-loss, and long propagation delays [2-3]. Thus, selecting a static correlation threshold for the underwater receiver is impractical as the received signal strength and noise level are changing in real-time. We therefore need to set a dynamic correlation threshold based on a dynamic estimation of the noise level. As a data packet could be sent at any time, it is hard to find a suitable rate for the receiver to take 'noise' samples periodically to set the dynamic threshold. The receiver must be able to set an appropriate threshold whether data or noise is present. The goal can be achieved through the use of an orthogonal reference sequence. As shown in the equations (1-4), the receive sequence, R(t), contains both the delayed transmitted sequence (the reference sequence followed by data), S(t-τ), and noise n(t) in (1). When we correlate the received sequence with the reference code, r(t), and the orthogonal reference code $r_o(t)$ at the receiver separately, the correlation result of the reference branch C(t) (2) contains both

signal and noise portion, whereas the correlation result of the orthogonal branch Co(t) (3) contains solely the noise correlation result. We can therefore estimate the real-time noise level by calculating the standard deviation of the correlation result of the orthogonal branch T(t) (4) and set the threshold according to this result. Though the threshold calculation is not a true estimate of the noise (as it includes the 'noise' from the imperfection of the orthogonal correlation), it serves as a reasonable threshold for peak detection.

$$R(t) = S(t-\tau) + n(t) \quad (1)$$

$$C(t) = R(t) * r(t) = \sum_j R'(t) r(t+j) \quad (2)$$

$$\begin{aligned} C_o(t) &= R(t) * r_o(t) = \sum_j R'(t) r_o(t+j) \\ &= \sum_j S'(t-\tau) r_o(t+j) + \sum_j n'(t) r_o(t+j) \\ &= \sum_j n'(t) r_o(t+j) \end{aligned} \quad (3)$$

$$T(t) = \sum_j C_o(t)^2 = \sum_j n'(t) r_o(t+j) \quad (4)$$

B. *Reference Code Selection*

In order to make use of the scheme described above, we must select a reference code that has an excellent periodic and aperiodic autocorrelation function, and an excellent cross-correlation function (with values close to zero) with its orthogonal code. We therefore choose two orthogonal Gold codes as Gold codes show reasonable cross-correlation and off-peak autocorrelation values while providing perfect orthogonality in the zero-offset case [14].

C. *Sequence Representation*

Gold codes are binary sequences of length 2^m-1 consisting of '1's and '-1's. As the underwater acoustic receiver uses FSK modulation, we can represent the Gold code by using the mark frequency to represent a '1' and a space frequency to represent a '-1.' However, representing the Gold code as a sequence of sinusoids eclipses its desired auto-correlation and cross-correlation properties. Thus we need to introduce a 'signal shaping' function before computing a correlation to make the receiving sequence look like a sequence of '1s' and '-1s' instead of a sequence of sinusoids.

D. *Sequence Length Selection*

We select an appropriate reference length for the Gold code by simulating the symbol synchronization method in MATLAB with a packet consisting of Gold code of lengths 3, 7, 15, and 31 followed by 80 symbols of data with a value of 15dB SNR (the same setting as [4]). Ten thousand simulations produced correct results and a sharp and distinct peak for the synchronization point for packets with Gold codes of length 15 and 31, but failed to produce accurate results for packets with Gold codes of length 7 and 3. Thus, we select a reference code length of 15 as it is the minimum length that provides accuracy. Figure 3 shows the simulation result for a packet with a reference code of length 15. The dashed line shows the result of the orthogonal correlation, the thick line shows the resulting threshold and the solid line shows the reference correlation result. The result of the correlation and orthogonal correlation is virtually zero when only noise is present. The correlation result first rises above the threshold when the data packet is present and the synchronization point is accurately detected as the highest point above the threshold within the following two reference lengths. A two reference length period is selected as the peak searching interval because it takes two reference lengths for an incoming reference signal to enter and exit the interval of correlation.

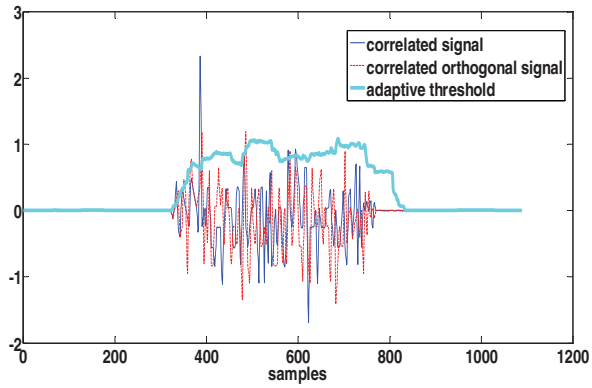

Figure 3. MATLAB simulation of symbol synchronization for underwater FSK

E. *Symbol Synchronizatin for Underwater FSK*

Our symbol synchronization scheme for underwater FSK is therefore given in Figure 4. The input signal, *Sig_in*, is the baseband signal coming from the DDC. The signal first passes through a signal shaping block which shapes the input signal from a sinusoid to a square wave. The signal shaping block consists of the same mark and space filters as those in the demodulator followed by a down sampler, absolute value operation, and subtraction. The effect of the filters is to remove out of band noise and begin to shape the signal as having regions of different amplitudes. The down sampler and absolute value operations further smooth the signal into an all positive square-like wave. The subtraction produces the desired square like wave consisting of positive and negative numbers ready for correlation with the reference code consisting of '1s' and '-1s.'

After the signal passes through the shaping block, it then performs a parallel correlation with the expected reference code and orthogonal reference code. The result of the correlation with the orthogonal branch is squared and summed to compute its standard deviation and serve as a threshold for the reference code correlation operation (as previously described). The 'find max index' logic compares the reference correlation result with the threshold. When the reference correlation result rises above the threshold, the logic looks for the highest peak above the threshold over two reference length periods to ensure it can find the highest peak that denotes the synchronization point. The index of peak position is recorded

and sent to the demodulator for accurate symbol timing (see section 5 for further details on the control flow).

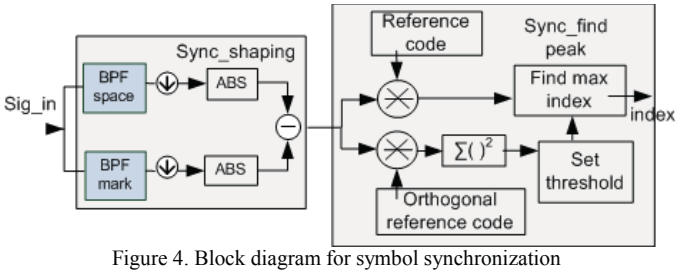

Figure 4. Block diagram for symbol synchronization

IV. HARDWARE IMPLEMENTATION

In order to implement our symbol synchronization design into hardware, we need to consider the design parameters of our acoustic receiver. Recall that the ADC sampling rate is 800KHz and the symbol period is 5ms. These parameters give a total of 4000 samples per symbol. With a reference length of 15 symbols, the entire reference sequence is 60,000 samples. Multiply that by 2 to include the orthogonal reference sequence. Correlating 120,000 samples requires 120,000 multiplications and 119,998 additions. These operations are either too area-consuming or computationally infeasible to complete in one 800KHz sampling clock period (1.25 us). We thus focus our hardware design implementation on reducing the complexity of the correlation.

Since the signal shaping block down samples the incoming signal to shape the signal into a square-like wave. As a square wave only requires one bit to represent the sequence, ideally we could down sample the shaped signal by a factor of 4000, thus reducing the number of samples for correlation to 15 (one sample per symbol). However, in an implementation of a communications system, signals should be oversampled by at least a factor of 4 to account for noise.

Thus we down sample the original signal by 1000 to create a signal sequence consisting of 4 samples per symbol, or 60 samples. We therefore require a correlater that can handle 120 multiplications (including the orthogonal branch) and 118 additions in one 800Hz (800kHz/1000) clock period.

Our correlation architecture (for one correlation branch) is giving in Figure 5. Each 800Hz clock period, the shaped signal is shifted into a 60 element shift register that must be correlated with the reference signal. In order to reduce the path length, we applied a 4KHz local clock to pipeline the storing. As the reference signal is represented by '1's and '-1's, we can simplify the correlation operation by replacing the multiplications with a sign selection method (SS core): when the reference signal is a '1', add the relevant stored shaped signal, when the reference signal is a '-1' subtract the relevant stored shaped signal.

We implement the correlation in a 15-stage pipeline as a fully parallel shift register implementation would require too many resources and a fully serial shift register implementation would require a significantly faster clock. The appropriate 15 elements of the shift register are serially selected into an SS core at a 16KHz clock rate to achieve a partial correlation result. The outputs of four SS cores are then summed in one 800Hz clock period to produce the complete correlation result. Thus we achieve a compact, timing accurate correlation implementation. The symbol synchronization has a delay of eighty 16KHz clock cycles, 1 for clocking data in, 60 for shifting the signal into the correlator, 15 for the pipelined additions and 4 for shifting the result to the output.

The rest of the symbol synchronization design is computationally inexpensive; the filters consist of two small 20-tap FIR filters designed using [15] and the threshold calculator only requires one multiplier and one adder. The 800Hz clock used for synchronization is source synchronous with the 800KHz sampling clock and 16KHz demodulator clock to avoid clock confliction.

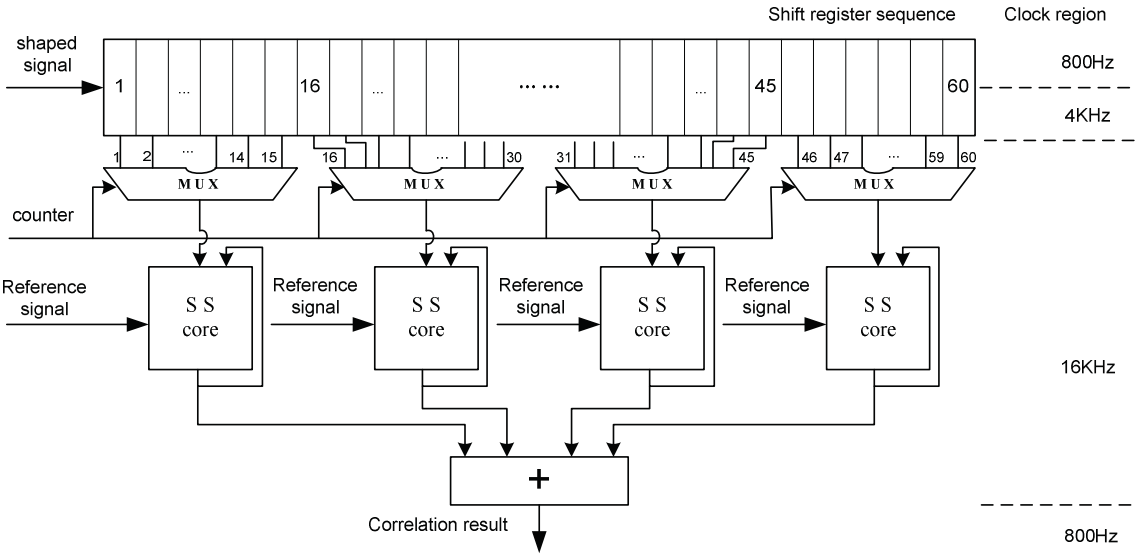

Figure 5. Hardware units of correlation implementation in underwater FSK

V. SYMBOL SYNCRHONIZATION CONTROL

In order to ensure correct operation of our symbol synchronizer, we need to consider the control scheme of the algorithm: When should synchronization begin? How can we use the synchronization point to set correct symbol timing for demodulation? This section describes the symbol synchronization control scheme and its implementation.

A. Initialization

Upon startup the modem must initialize the known reference code and orthogonal reference code to memory at the clock rate 800KHz clock and performing an initial correlation over one reference length (60, 800Hz clock cycles) to ensure a valid threshold value, because the threshold calculation involves summing the correlation results from orthogonal branch over one reference length. Therefore, the total initialization occupies 75ms.

B. Achieving Accurate Symbol Timing

After initialization, the synchronizer executes as previously described: shaping the incoming signal, correlating it with the reference code and orthogonal reference code and searching for a correlation result above the threshold. When the reference correlation result rises above the threshold, the block looks for the highest peak above the threshold over two reference length periods to ensure it can find the highest peak that denotes the synchronization point. But how can the synchronization point be used for accurate symbol timing? Figure 6 and the following description provide our solution.

When the correlation result first rises above the threshold, we assume that this point is the synchronization point and thus need to start the symbol clock. Recall that the down-sampled clock used for synchronization is 800Hz and the symbol clock is 200Hz; therefore a symbol clock can be formed by 4 cycles of the down-sampled clock. Thus, at the time a peak occurs (1st peak), a reset signal is asserted (*Rst_symbol_cnt*) to reset a 2 bit counter (*Symbol_clk_cnt*) which is used to generate the symbol clock every four 800Hz cycles. In the meantime, the index of the peak location, A, is stored into a real-time index register (*index_reg*) and the demodulator is enabled (*en_dem*). The real-time demodulated values are then stored in a temporary buffer and the number of demodulated symbols is counted by another counter (*symbol_num_count*).

If another, larger peak, occurs within two reference lengths (2nd peak), *rst_symbol_cnt* is asserted to resynchronize the symbol clock at this higher peak point. The value of *index_reg* is also replaced to B (the number of samples offset from the initial peak A). The counter, *symbol_num_count,* continues to count the number of demodulated symbols and the demodulated results are stored in a temporary buffer. The values already demodulated between A and B will be ignored as the peak at A was not the maximum.

When an even larger, real peak occurs (3rd), *rst_symbol_cnt* is again re-asserted to resynchronize the symbol clock at this higher peak. The value of *index_reg* is set to C (the number of samples offset from the initial peak A). Since B was not the maximum either, the values already demodulated between time A and C will have to be ignored.

At the end of two reference lengths, the location of the maximum peak, C, can be found in *index_reg* and the number of symbols demodulated since the first peak can be found in *symbol_num_count.* We can then calculate the address of the first valid piece of data in the temporary buffer with the following equation:

$$\text{addr} = symbol\_num\_cnt - \text{round}\left[\frac{NO\_OF\_SAMPLES - 1 - index\_reg}{DS\_FACTOR}\right]$$

where $NO\_OF\_SAMPLES$ is the number of samples in two reference lengths (120), DS_FACTOR is the down sampling factor between the synchronization clock (800Hz) and the symbol clock (200Hz) and round[(*NO_OF_SAMPLES* -1 -*index_reg*) / *DS_FACTOR*] gives the number of symbols demodulated since the synchronization point (C) found in *index_reg*.

After the modem has demodulated the entire packet, it desserts the *en_dem* signal and the synchronizer returns to searching for a correlation result above the threshold to synchronize on the next incoming packet.

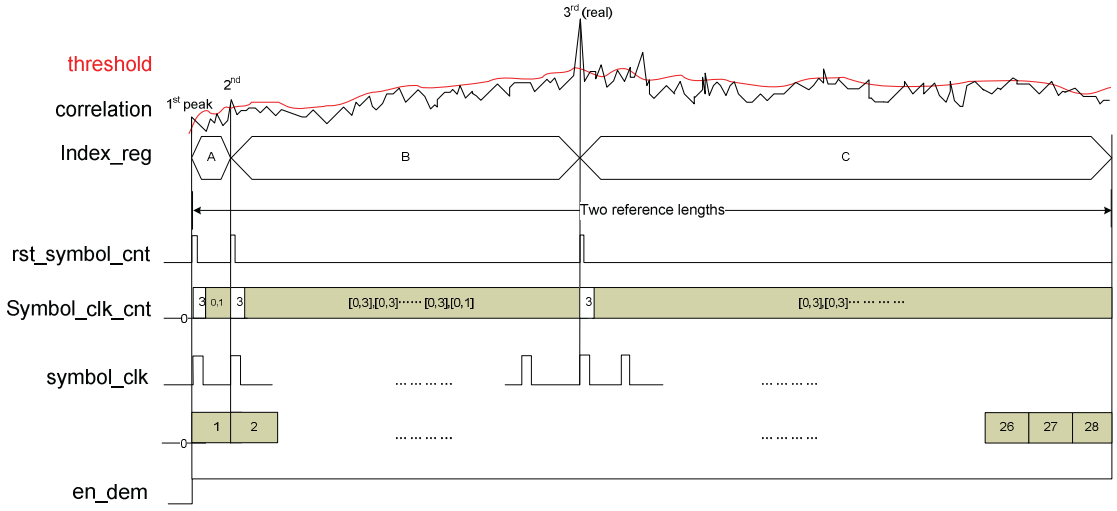

Figure 6. Control of symbol synchronization

VI. RESULTS

In order to test the accuracy of our symbol synchronizer, we collected noise followed by a packet of data followed by noise followed by another packet of data in our underwater lab bench setup (an 18 inch bucket of water with a send and receive transducer). We used the collected data as input to our receiver design and monitored signals in ModelSim[16] to verify the synchronizer's operation. The packet of data collected consisted of the Gold Code of '011001010111101' followed by 80 bits of data with 12dB SNR. Figure 7 shows the hardware simulation result for our design described in Verilog HDL. The four signals in the figure are: the input signal to the synchronizer (Signal in), the output of the signal shaping block (Shaped Signal), the threshold set by orthogonal correlation (threshold), and the output of the reference code correlation (correlation). The horizontal bars illustrate the actual synchronization point (actual) and the one reported (index) from the synchronizer. As expected, the index is exactly 80 cycles behind the actual synchronization point illustrating accurate symbol timing is achieved. We repeated the test altering the SNR level in the bucket of water, to 8, 4, and 2 dB and obtained accurate synchronization for all tests (although demodulation was not accurate for the 2dB test).

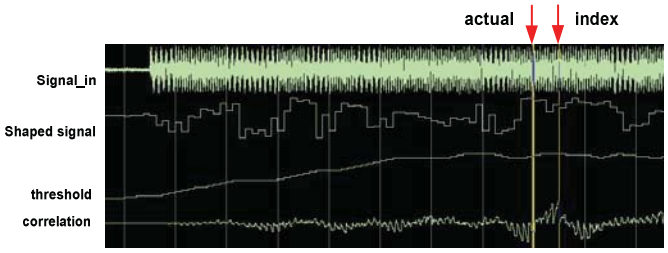

Figure 7. Experimental result

Table 2 shows the FPGA hardware resources required for each component of the acoustic receiver design with standard optimization. These resources are reported for a Xilinx Spartan 3 xc3s2000 FPGA (a low-cost and low power device) with the percent of total resources shown in parentheses. It is difficult to compare the resource usage to other underwater synchronization hardware implementations as these values are not reported in the literature. The power consumption of the receiver (the DDC, demodulator, and synchronizer), determined by XPower Estimator 9.1.03 [18], is 0.240W which is significantly lower than the 0.8W receive power reported by some commercial acoustic modems [19]. Although the synchronizer is the most area consuming component of the acoustic receiver design, it offers a compact implementation capable of fitting into a device that costs less than $60.

TABLE II. FPGA RESOURCE USAGE

	Occupied Slices	LUTs	BRAMs
DDC	284	541	9
Demodulator	1025	1980	1
Synchronizer	12000	22101	2

VII. CONCLUSION

Symbol synchronization is a critical component in the design of an underwater acoustic modem. When the modem receiver obtains an input stream, it should make sure to find the correct start of the data sequence to set accurate sampling and decision timing for subsequent demodulation.

This paper describes a practical description of the hardware design choices and implementation details required to build a compact hardware symbol synchronizer suitable for our low-cost, low-power underwater FSK acoustic modem. Experimental results show the design provides accurate synchronization while occupying only 58% of occupied slicess and 54% of LUTs in the low-cost, low-power Spartan 3 xc3s2000 FPGA.

Future work will be to perform open ocean experiments to test the operation of the synchronizer and the entire modem design in a real underwater environment.

ACKNOWLEDGMENT

This material is based upon work partially supported by the China Scholarship Council and partially supported under National Science Foundation Grant #0816419 and a National Science Foundation Graduate Research Fellowship.

REFERENCES

[1] S. Bregni, Synchronization of Digital Telecommunications Networks, *WILEY*, 2002.

[2] E. M. Sozer and M. Stojanovic, "Underwater Acoustic Networks", *IEEE Journal of Oceanic engineering*, Vol. 25, No. 1, January 2000.

[3] I. F. Akyildiz, D. Pompili and T. Melodia, "Challenges for Efficient Communication in Underwater Acoustic Sensor Networks", *ACM Sigbed Review*, Vol. 1, No. 2, July 2004.

[4] Y. Li, B. Benson, R. Kastner, X. Zh, "Bit Error Rate, Power and Area Analysis of Multiple Implementations of Underwater FSK", *Proceedings of ERSA*, 2009.

[5] 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," *Proceedings of MTS/IEEE Oceans*, 2005.

[6] R. Iltis, H. Lee, R. Kastner, D. Doonan, T. Fu, R. Moore and M. Chin, "An Underwater Acoustic Telemetry Modem for Eco-Sensing," *Proceedings of MTS/IEEE Oceans*, 2005

[7] J. Wills, W. Ye, and J. Heidemann, "Low-power acoustic modem for dense underwater sensor networks," *Proc. of WUWNet Conference*, 2006.

[8] C.V. Jurdak, Lopes, and P. Baldi. "Software Acoustic Modems for Short Range Mote-based Underwater Sensor Networks," *Proceedings of MTS/IEEE Oceans Asia Pacific*. 2006.

[9] B. Benson, G. Chang, D. Manov, B. Graham, and R. Kastner, "Design of a low-cost acoustic modem for moored oceanographic applications," *Proceedings of WUWNet Conference*, 2006.

[10] R. Sözer and M. Stojanovic, "Reconfigurable acoustic modem for underwater sensor networks", *Proc. of WUWNet Conference,* 2006.

[11] H. Sun, R. Xu, F. Xu, "A New Accurate Symbol synchronization Scheme for Underwater Acoustic Communication System", *Proceedings*

of IEEE International Workshop on Anti-counterfeitiing, Security, Identification, 2007.

[12] R. Jurdak , A. Ruzzelli, G. O'Hare, and C. Lopes, "Reliable Symbol Synchronization in Software-Drive Acoustic Sensor Networks", *Proceedings of IEEE Globecom*, 2006.

[13] S. Bregni, Synchronization of Digital Telecommunications Networks, *WILEY*, 2002. J. Heidemann, Y. Li, A. Syed, J. Wills, and W. Ye, "Research Challenges and Applications for Underwater Sensor Networking", *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2006.

[14] C. Chien, "Digital radio system on a chip: a system approach," *Springer,* 2001.

[15] Spiral. http://spiral.net/hardware/filter.html

[16] ModelSim SE 6.4a. http://model.com/content/modelsim-downloads

[17] Kyovtorov "FPGA Implementation of Low-Frequency GPR signal…"

[18] Xilinx XPower Estimator 9.1.03. http://www.xilinx.com/products/design_resources/power_central/

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