Energy Benefits of Reconfigurable Hardware for Use in Underwater Sensor Nets

Bridget Benson, Ali Irturk, Junguk Cho, and Ryan Kastner

Abstract— **Small, dense underwater sensor networks have the potential to greatly improve undersea environmental and structural monitoring. However, few sensor nets exist because commercially available underwater acoustic modems are too costly and energy inefficient to be practical for this applications. Therefore, when designing an acoustic modem for sensor networks, the designer must optimize for low cost and low energy consumption at every level, from the analog electronics, to the signal processing scheme, to the hardware platform. In this paper we focus on the design choice of hardware platform: digital signal processors, microcontrollers, or reconfigurable hardware, to optimize for energy efficiency while keeping costs low. We implement one algorithm used in an acoustic modem design - Matching Pursuits for channel estimation - on all three platforms and perform a design space exploration to compare the timing, power and energy consumption of each implementation. We show that the reconfigurable hardware implementation can provide a maximum of 210X and 52X decrease in energy consumption over the microcontroller and DSP implementations respectively.**

I. Introduction

Small, dense underwater sensor networks (UWSNs) have the potential to greatly improve environmental (pollution, coral reef, seismic, ocean current, etc.) and structural (oil platform, pipeline, undersea tunnel, etc.) monitoring which leads to greater understanding of our earth's bodies of water and the increased safety of mankind. These sensor networks are likely to have on the order of 10s to 100s of nodes spaced a relatively small distance apart (up to a few hundred meters). Few of these networks currently exist because commercial off-the-shelf (COTS) underwater modems [1-3] (devices that actually receive and transmit data underwater) are not well suited for this application. The COTS modems' energy consumption, ranges, and price points are all designed for sparse, long-range, expensive systems rather than small, dense, and cheap sensor-nets [4]. Therefore, a new low-cost (to allow for the deployment of 10s to 100s of nodes), low-energy (to allow for long deployment) underwater acoustic modem must be designed.

There are many design choices that must be considered when designing a low-cost, low-energy underwater acoustic modem including, but not limited to, the choice of signal processing scheme, the choice of underwater transducer and corresponding analog electronics, the choice of interfaces to sensors or higher level networking devices, and the choice of hardware platform for the implementation. Each design choice is a research area in itself, so this paper focuses on the choice of hardware platform for the acoustic modem design. Many research underwater acoustic modems have already been made using a variety of different hardware platforms (including digital signal processors (DSPs) [5-9], microcontrollers [10, 11], and reconfigurable hardware such as field programmable gate arrays (FPGAs) [12-14] but no work categorizes the energy benefits one platform can provide over another.

The goal of this paper is to analyze the energy benefits reconfigurable hardware can provide when used as the hardware platform for an underwater acoustic modem. Reconfigurable hardware platforms strike a balance between solely hardware and solely software solutions, as they have the programmability of software with performance capacity approaching that of a custom hardware implementation. They also present designers with substantially more parallelism allowing for a more efficient application implementation. [15-20] In order to quantify the potential benefits, we focus our discussion on the implementation of the Matching Pursuits for channel estimation algorithm. We select this algorithm because it can be used in any acoustic modem design (as it provides increased noise immunity for improvement in signal detection) and is highly parallelizable (making it an ideal candidate for a hardware solution). We implement this algorithm in a reconfigurable intellectual property (IP) core, provide a design space exploration of this core, and compare its multiple implementations with its implementation on a microcontroller and a DSP.

The major contributions of this paper are:

- A design space exploration of area, timing, throughput, power and energy consumption tradeoffs using different levels of parallelism, bit widths, and FPGA devices for the implementation of the Matching Pursuits algorithm;
- A comparison of execution time, power, and energy consumption of Matching Pursuits algorithm on a microcontroller, DSP, and FPGA
- An energy efficient implementation of the Matching Pursuits algorithm on reconfigurable hardware that provides 210X and 52X energy decrease over the microcontroller and DSP implementations respectively

This paper is organized as follows: Section II presents a high level description of underwater acoustic modem design. Section III presents the Matching Pursuits algorithm for channel estimation and summarizes the design specifications needed for the implementation of the IP core. Section IV presents the design of the IP core for the Matching Pursuits algorithm and discusses the tradeoffs for the design space exploration of the core. Section V compares the results of the reconfigurable hardware, DSP and microcontroller implementations. We conclude in section VI.

II. Acoustic Modem Design

In an underwater sensor network, just as in a terrestrial network, the modem is responsible for implementing the physical layer of the network stack which is shown in Figure 1. That is, the modem is responsible for the actual physical transmission and reception of data across the network. The higher network layers are responsible for MAC protocols (link), routing protocols (network), transport protocols (transport), and data processing (application).

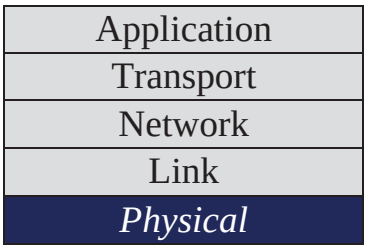

Figure 1. The Underwater Acoustic Modem fits into the physical layer of a typical network stack.

Acoustics are used in underwater communications instead of radio frequency (RF) as in terrestrial networks because it is a well known fact that electromagnetic waves attenuate rapidly underwater making them an insufficient carrier of data through the water. Underwater acoustic modems consist of three main components as shown in Figure 2: **1.** the analog front end (dark gray), **2.** a hardware platform (light gray) and **3.** serial interfaces (black). The analog front end is responsible for converting electrical signals into sound waves and vice versa (transducer) and for generating the appropriate power level for the received and transmitted signals (analog electronics which include an amplifier, pre-amplifier, and transmit/receive switch). The hardware platform is responsible for control and signal processing, namely performing modulation and demodulation using a specific signaling scheme (i.e. frequency shift keying (FSK), direct sequence spread spectrum (DSSS), or orthogonal frequency division multiplexing (OFDM)) and performing error encoding and decoding. The serial interfaces are responsible for communication with underwater sensors and/or higher level network layers.

For a low-cost, low-energy acoustic modem design, the designer must optimize the implementation at every level, from the analog electronics, to the signal processing scheme, to the hardware platform. In this paper we focus on the design choice of hardware platform: digital signal processors, microcontrollers, or reconfigurable hardware, to optimize for energy efficiency.

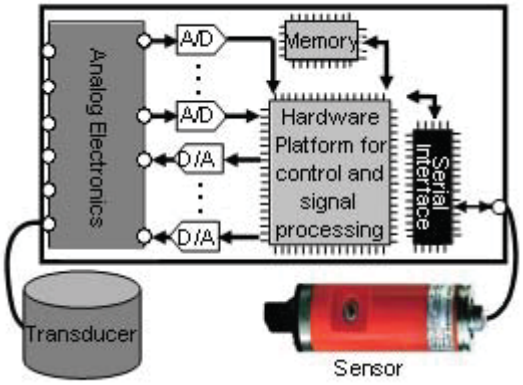

Figure 2. Major components of an underwater acoustic modem: the analog front end (dark gray) the hardware platform (light gray) and serial interface (black).

III. Matching Pursuits Algorithm and Design Specifications

In order to quantify the potential benefits reconfigurable hardware can provide to the underwater acoustic modem, we focus our discussion on the implementation of the Matching Pursuits for channel estimation algorithm. We select this algorithm because it can be used in any acoustic modem design (as it provides increased noise immunity for improvement in signal detection) and is highly parallelizable (making it an ideal candidate for a hardware solution). The MP algorithm for channel estimation is shown in Figure 3. This algorithm is presented in [21] and was redesigned from [22, 23] for speed improvement with zero reduction in channel estimation accuracy. Channel estimation is a common problem to many fields of research and in particular in underwater acoustics where the received signal is prone to strong multipath (bounces off the sea floor, surface, and obstacles such as coral heads/rocks), and dispersion. Channel estimation algorithms are used to calculate delay and attenuation parameters of each transmission path. Given estimates of the channel parameters, signal corruption due to multipath propagation can be easily reversed, and the signals due to multiple paths can be combined coherently for increased noise immunity for improvement in signal detection.

MP takes in four matrices as input: the receive signal vector $r \in C^{2N_s \times 1}$, the signal matrix S, $\in R^{2N_s \times N_s}$ defined in [23], the Hermetian matrix $A = S^H S$, $\in R^{N_s \times N_s}$, and vector a, $\in R^{N_s \times 1}$. The vector a is simply one divided by the diagonal elements of A and is used to eliminate the need for division operations. The S, A, and a matrices are static matrices as the values are known *a priori* and therefore can be pre-computed once and stored in memory. In steps (1-5), MP computes matched filter outputs, $V \in C^{N_s \times 1}$, and initializes the channel coefficients, $F \in C^{N_s \times 1}$, and temporary channel coefficients, $G \in C^{N_s \times 1}$, to zero. In steps (7-15) MP loops over the hypothesized number of paths, N_f, iteratively canceling the strongest detected signal component to estimate the next

$Input\ (r, S, A, a)$
$Output\ (f)$
$// r : received\ signal\ vector, r \in C^{2N_s \times 1}$
$// S = [S_1, \dots, S_i, \dots, S_{N_s}], S \in R^{2N_s \times N_s}$
$and\ column\ S_i \in R^{2N_s \times 1}$
$// A = [A_1, \dots, A_k, \dots, A_{N_s}], A \in R^{N_s \times N_s}$
$and\ column\ A_k \in R^{N_s \times 1}$
$// a = [a_1, \dots, a_k, \dots, a_{N_s}]^T, a \in R^{N_s \times 1}$
$and\ column\ a_k \in R$
$// f : estimated\ channel\ coefficients, f \in C^{N_s \times 1}$
$\boldsymbol{MP}\ (r, S, A, a)$
1. $\boldsymbol{for}\ i = 1, 2, \dots, N_s$
   $// compute\ matched\ filter\ (MF)\ outputs$
2. $V_i^0 \leftarrow S_i^T r$
3. $F_i \leftarrow 0$
4. $G_i \leftarrow 0$
5. $\boldsymbol{endfor}$
6. $q_0 \leftarrow 0$
   $// do\ successive\ interference\ cancellation$
7. $\boldsymbol{for}\ j = 1, 2, \dots, N_f$
   $// update\ MF\ outputs$
8. $V^j \leftarrow V^{j-1} - F_{q_{j-1}} A_{q_j - 1}$
9. $\boldsymbol{for}\ k = 0, 1, \dots, N_s - 1$
10. $G_k \leftarrow V_k^j a_k$
11. $Q_k \leftarrow (V_k^j)^* G_k$
12. $\boldsymbol{endfor}$
13. $q_j \leftarrow \underset{k, k \neq q_1, \dots, q_{j-1}}{argmax} \{Q_k\}$
14. $F_{q_j} \leftarrow G_{q_j}$
15. $\boldsymbol{endfor}$
16. $\boldsymbol{return}(\boldsymbol{F})$

Figure 3. Matching Pursuits Algorithm for Channel Estimation

channel coefficient. Specifically, MP updates the matched filter outputs by canceling the strongest detected signal component (8) and computes decision variables, $Q \in R^{N_s \times 1}$, and temporary channel coefficients, $G \in C^{N_s \times 1}$ (10-11). MP then searches for the next strongest channel coefficient by finding the index, *q*, of the maximum decision variable, *Q*, that is not equal to any index that has already been found (13). MP saves the temporary channel coefficient value at that index, G_q, as the next strongest channel coefficient (14). This coefficient is then used in the next iteration of the *for loop* in (7) for successive cancelation. When the algorithm is complete, it returns the estimated channel coefficients (16).

The algorithm applies to any direct sequence spread spectrum CDMA (DS-CDMA) signal. As previously mentioned, not all underwater acoustic modems use direct sequence spread spectrum (DSSS) signaling as their signaling scheme, but [6, 24, 25] have shown that DSSS waveforms yield significantly lower error rates than Frequency Shift Keying (FSK) (a more common signaling scheme found in existing research modems) due to frequency diversity. These lower error rates allow lower transmit power and lower energy consumption thus contributing to an overall lower energy design.

One research modem that uses direct sequence spread spectrum signaling is the UCSB AquaModem. Because the AquaModem was successfully field tested at the Moorea Coral Reef Long Term Ecological Research Site where it achieved symbol error rates averaging < 1%, [26] we use its design parameters to determine the size and values of the inputs needed for the Matching Pursuits Algorithm for channel estimation.

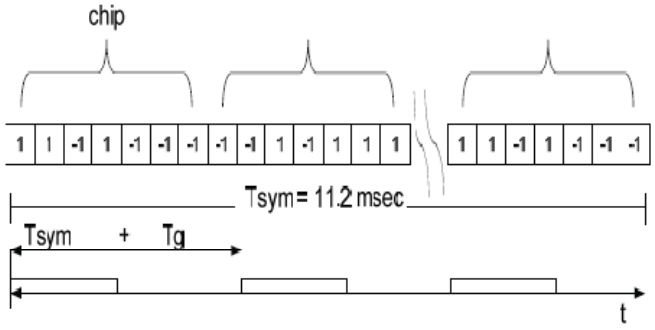

Figure 4. Walsh/m-sequence signals for the AquaModem

The AquaModem uses direct sequence spread spectrum signaling based on eight composite Walsh and m-sequence waveforms [6]. Each waveform is comprised of 8 symbols. Each symbol is orthogonal to every other symbol simplifying symbol detection. Each symbol is multiplied by a 7 'chip' spreading sequence to spread the energy of the symbol across a wider bandwidth making the waveforms instantaneously wideband and providing robustness to frequency selective multipath. The 8 symbol x 7 chip (56 chip) waveform is shown in Figure 4.

This 56 chip waveform must have a time duration greater than 10 milliseconds, the duration of the multipath spread in shallow water [27, 28]. Therefore, the chip duration is given as 0.2 ms, making the waveform duration 0.2 * 56 = 11.2 ms. An 11.2 ms time guard band for channel clearing is added to eliminate the need for equalization. Thus the duration to send one waveform is 11.2 + 11.2 = 22.4 ms. Nyquist sampling requires the sampling interval to be half the chip duration, giving a sampling rate of 0.1 ms and a total of 0.1 (samples/ms) * 22.4 (ms) = 224 samples per waveform.

TABLE I
AQUAMODEM DESIGN PARAMETERS

Walsh Symbol length: N_w	8 symbols
m-sequence length: L_{pn}	7 chips
Chip Duration: T_c	0.2 msec
Sampling Interval: $T_s = T_c/2$	0.1 msec
Symbol duration: $T_{sym} = L_{pn} * N_w * T_c$	11.2 msec
Time guard interval: $T_g = T_{sym}$	11.2 msec
Samples/symbol: $N_s = T_{sym}/T_s$	112 samples
Samples/time guard: $N_t = T_g/T_s$	112 samples
Total receive vector samples: $R_v = N_s + N_t$	224 samples

These design parameters govern the size of the input receive vector, *r*, and signal matrices *S*, *A*, and *a* for the MP algorithm. The receive vector is of size *224 x 1* (as described above), the signal matrix, *S*, is of size *224 x 112*(the 224 rows representing the size of the received vector and the 112 columns representing shifted versions (and hence different paths) of the 112 chip waveform), the Hermetian matrix, *A*, is of size *112 x 112*(size derived from the *S* matrix) , and the vector *a* is of size *112 x 1*(size derived from the *A* matrix). The MP algorithm then takes the receive vector and matches it to each row in the S matrix to find the strongest path and the channel coefficient for that path. That path is then canceled from the received vector and the next strongest path is found until N_f channel coefficients have been determined.

IV. Design Space Exploration

MP is inherently parallel, and an ideal candidate for efficient implementation on modern reconfigurable platforms. In this section we present our IP core for the matching pursuits algorithm and describe the design parameters of levels of parallelism, bit widths, and device selection that we explore to achieve an accurate energy efficient design, subject to the timing constraint of 22.4 ms between received samples.

A. The IP Core

An IP core for channel estimation was implemented in [21]. We look to [21] as a starting point for our implementation of an IP Core. We use high level design tools Simulink and System Generator to implement and test our modified MP design. The block diagram of our MP design is shown in Figure 5. In this design, the "Filter and Cancel Block" (FC block) is replicated 112 times to correspond to the 112 columns of the signal matrices *S*, *A* and *a*. In each block, column *k* of *S*, column *k* of *A* and element *k* of *a* is stored in memory. This replication effectively allows for the unrolling of the *for loops* in steps 1-5 and 9-12 in Figure 2. The design uses duplicate hardware to process the real and imaginary data at the same time. The registers V_{KR}, G_{KR}, and F_{KR} store the real values of the column matched filter output, temporary channel coefficient, and estimated channel coefficient respectively. The registers V_{KI}, G_{KI}, and F_{KI} store the imaginary values of the column matched filter output, temporary channel coefficient, and estimated channel coefficient respectively. The register Q_K stores the decision variable, *Q*, used in steps 11 and 13 of the algorithm. R_r and R_i represent the real and imaginary portions of the receive vector respectively. The outputs of each of the 112 FC blocks (*Q*, G_{KR}, and G_{KI}) are fed into the "q-gen block" to perform steps 13 and 14 of the algorithm. The outputs of the "q-gen block" are then fed back into the each FC block to perform the successive inference cancellation in the *for loop* of steps 7-15. Once the loop in 7-15 has finished, the *F* registers contain the estimated channel coefficients. The control logic is implemented in a Xilinx M-code block and is not shown for sake of simplicity.

B. Levels of Parallelism

More parallelism in the IP Core greatly decreases the execution time of the algorithm, but comes at a cost of more area and higher power consumption. Because we are interested in minimizing energy (power times time), a highly parallelized version of the IP Core may not be the best suited for our application. We therefore investigate the area/timing/ throughput/power/energy tradeoffs of different levels of parallelism of our IP Core design. For example, a 'fully parallel' design would have 112 Filter and Cancel (FC) blocks (as we described in the previous subsection). We can reduce the area of the 'fully parallel' design by almost half and increase the latency by almost 2 by doubling up on all the memory resources per FC block (i.e. two columns of *S*, two rows of *A*, two elements of *a*, and two times all the registers) and changing the control to execute each block twice. We could continue to serialize the design until we have only 1 FC block with the entire contents of the signal matrices *S*, *A*, and *a*, and 112 real and imaginary *V*, *G*, *I* and *Q* registers contained within its memory elements.

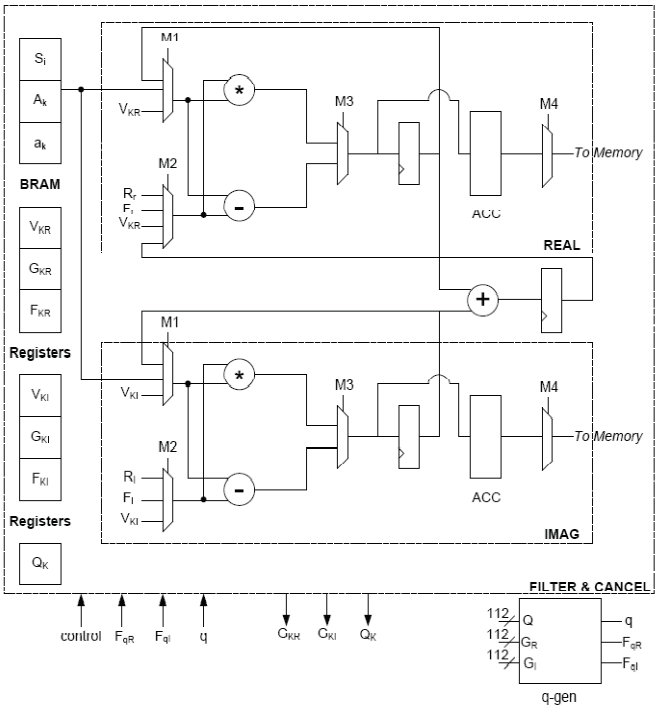

Figure 5. Modified IP Core for Channel Estimation

C. Bit Widths

The *S*, *A*, and *a* signal matrices are large real matrices of size 224 x 112, 112 x 112 and 1 x 112 elements, respectively. If each value is represented by 32 bits, the total number of bits to represent these matrices is 1208Kb. This large amount of data greatly increases the area of the design and requires a large FPGA device just to store all this data in on-chip block RAM (BRAM). Designers can trade off the number of bits with accuracy to obtain a design with desired precision for the lowest possible area. Meng et. all [21] determined 8-10 bits is sufficient for accurate channel estimation with optimal dynamic range scaling for their MP implementation. We provide a means to explore how different bit widths affect the overall area, timing, throughput, power and energy of the design.

D. FPGA Device Selection

Different FPGA devices offer different amount of resources: configurable logic blocks (CLBs), memory units (BRAMs), and embedded multipliers (DSP48s) and are made with different nanometer technologies. Therefore, device selection has a large affect on the power and energy consumption of the design. We have to select a device that has enough resources for the MP design (depending on the bit width and level of parallelism chosen). We can then use the appropriate Xilinx Power Estimator to determine the power usage of the synthesized design for the selected device and can use the power and timing results to determine the design's energy consumption.

V. RESULTS

We generated multiple designs of the IP Core for channel estimation, varying the levels of parallelism, bit widths, and FPGA device. All designs were generated in the Simulink environment using Xilinx blocksets with an estimated number of paths $N_f = 6$ (determined to be a good number for Nf during AquaModem field tests). Every design is synthesized, placed and routed with Xilinx ISE 9.1 and power estimated with the Xilinx Power Estimator. The two devices in this study are the Virtex-4 xc4vsx55 and the Spartan-3 xc3s5000 - the Virtex-4 for its increased speed and the Spartan-3 for its lower power consumption. These devices are the largest devices in their respective device families therefore offering the maximum amount of resources available per family to allow for the most possible parallelism. Table 2 shows the area (slices), timing (microseconds) and throughput (calculated as maximum clock frequency divided by the number of clock cycles) results for our design space exploration. The timing assumes the receive vector is already in memory and therefore does not include the time it would take to obtain the receive vector from actual hardware. The column titled #FC blocks represents the number of Filter and Cancel blocks and hence the level of parallelism in the design. Note that the 'fully parallel' design (with 112 FC blocks) could not be implemented in the Spartan 3 because of its limited number of DSP48 resources and therefore is not shown. The fully parallelized design requires 224 DSP48 resources; our largest Virtex-4 device having 512 and our largest Spartan-3 device having only 104.

The results indicate that if area, timing and throughput were the primary design objective of the IP Core, the Virtex-4 offers the better solution over the Spartan-3 over all bit widths and levels of parallelism. The results also indicate that the timing (even for the serial design of 1 Filter and Cancel Block) is well within the constraint of 22.4ms between received samples.

TABLE 2
AREA, TIMING, AND THROUGHPUT RESULTS OF DESIGN SPACE EXPLORATION OF IP CORE

Bit Width	#FC blocks	Device	Area (slices)	Timing (us)	Throughput (s^{-1})
	112	Virtex-4	11508	3.95	0.253
	14	Virtex-4	1439	31.63	0.032
	14	Spartan-3	1897	48.94	0.020
	1	Virtex-4	103	442.80	0.002
8 bits	1	Spartan-3	136	685.17	0.001
	112	Virtex-4	16884	4.10	0.244
	14	Virtex-4	2111	32.83	0.030
	14	Spartan-3	2783	49.85	0.020
	1	Virtex-4	151	459.65	0.002
12 bits	1	Spartan-3	199	697.83	0.001
	112	Virtex-4	22260	4.32	0.231
	14	Virtex-4	2783	34.59	0.029
	14	Spartan-3	3665	52.65	0.019
	1	Virtex-4	199	484.24	0.002
16 bits	1	Spartan-3	262	737.07	0.001

Figure 6 shows the total power (W) and energy (micro Joules) consumption of each design presented in Table 2. These values are based on just one run of the MP algorithm

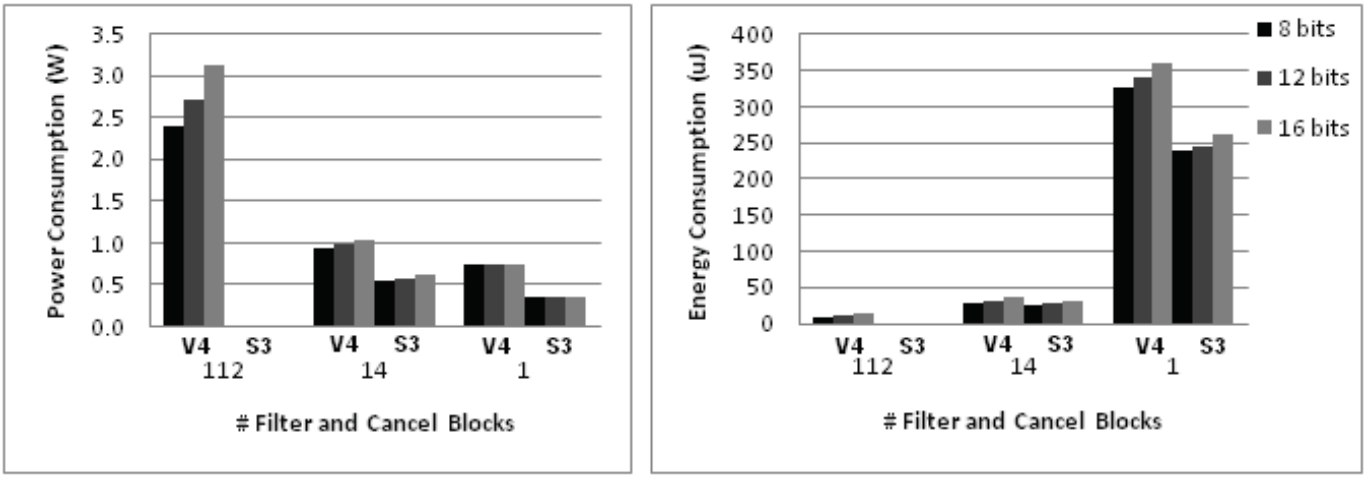

Figure 6. Power and Energy Consumption Results of Design Space Exploration of IP Core. There was no fully parallel design (112 FC blocks) for the Spartan 3 device because of its limited number of DSP48 resources. The legend applies to both the power and energy graphs.

and assume the processor enters an idle mode after processing to save power. This assumption requires the processor indeed has a power down mode and also does not consider the cost of reconfiguration on power up. Note that the Virtex-4 has a quiescent power of 0.723W and the Spartan-3 has a quiescent power of 0.335W. As expected, the Virtex-4 consumes more power than the Spartan-3 over all bit widths and levels of parallelism as the Virtex-4 is a larger device. We also observe that the power consumption increases as the design become more parallelized, which is a result of the increased area (and hence resource usage) of the more parallelized designs. We note that the power consumption of the most serial design (1 FC block) shows little dependence on bit width and is close to that of the quiescent power of the FPGA device. Although power consumption increases with increasing parallelism, we observe the reverse for energy consumption. Energy consumption is computed by multiplying the power consumption by the time, so high power consumption and/or high latency can lead to high energy consumption. Because the energy consumption reduces with increased parallelism, the increased speed of the parallel design makes up for the increased power. We also observe that the Spartan 3 consumes less energy than the Virtex 4 for the design with 1 FC block and consumes almost the same amount of energy as the Virtex 4 for the design with 14 FC blocks. We cannot compare the Virtex 4 and the Spartan 3 for the design with 112 FC blocks because the Spartan 3 device did not have enough resources for this design.

TABLE 3
COMPARISON OF THE DSP MP IMPLEMENTATION WITH THE LEAST AND MOST ENERGY CONSUMING VIRTEX4 AND SPARTAN 3 IP CORE IMPLEMENTATIONS

	Time (us)	Power (W)	Energy (uJ)	Energy Decrease (MicroBlaze)	Energy Decrease (DSP)
MicroBlaze 32 bit	6341.84	0.38	2000.40	1.0X	0.25X
DSP 32 bit	468	1.07	500.76	3.99X	1.0X
Virtex-4 1 FC block 16 bit	484.24	0.74	360.52	5.55X	1.39X
Spartan-3 1 FC block 16 bit	737.07	0.35	260.92	7.67X	1.92X
Virtex-4 112FC block 8 bit	3.95	2.40	9.50	210.57X	52.71X
Spartan-3 14 FC block 8 bit	48.94	0.53	25.82	77.47X	19.39X

Table 3 compares the energy consumption of the least and most energy consuming Spartan-3 and Virtex-4 IP Core designs with the MP design implemented in a TIC6713 DSP and with the MP design implemented on MicroBlaze (a 32-bit soft core microprocessor from Xilinx). The last column two columns showing energy decrease were computed by dividing the energy used by the microcontroller by the energy used for the specified design (4^{th} column) and by dividing the energy used by the DSP by the energy used for the specified design (last column). Note that the DSP and MicroBlaze implementations use 32-bit floating point representation of numbers whereas all of our FPGA designs use varying bit width fixed point numbers making direct comparison difficult. Thus, if, the DSP and Microblaze implementations could use lower precision, then perhaps they could offer further energy savings. The total computational time for the DSP was estimated by measuring the time to compute one coefficient (about 78 us) and multiplying this number by the size of N_f (6 coefficients). The power for the DSP design was estimated using TI's Spreadsheet Power Estimator. The total computational time for the MicroBlaze implementation was calculated using an embedded timer. The MicroBlaze design was synthesized, placed and routed with Xilinx SDK 9.1. The output of the synthesized design provided the maximum clock frequency per design and the map report file (.mrp) for import into the Xilinx Power Estimator. Our results show that the most serial reconfigurable hardware designs for the Virtex 4 (row 3) and the Spartan 3 (row 4) are fairly comparable to the DSP implementation offering only 1.39X and 1.92X energy decrease over the DSP implementation respectively; the Virtex 4 offering similar timing and power to the DSP implementation and the Spartan 3 offering an increase in timing with a decrease in power over the DSP implementation. Though the Microblaze solution offers comparable power to the serial Spartan 3 implementation and less power than the serial Virtex 4 implementation, its energy consumption is considerably larger than either of these implementations because of its extremely high latency. This high latency is likely a result of the lack of specialized hardware for DSP applications in the soft core processor. The more parallel reconfigurable hardware designs for the Virtex 4 (row 5) and the Spartan 3 (row 6) offer substantial energy improvement over both the DSP and MicroBlaze implementations. Though the most parallel Virtex 4 implementation (row 5) has the highest power consumption of all the designs, it has an extremely small computation time allowing for a low-energy design and an astonishing energy decrease of 210X and 52X over the Microblaze and DSP implementations respectively. The more parallel Spartan-3 implementation also offers a rather small computation time for moderate power consumption also providing 77X and 19X decrease over the Microblaze and DSP implementations respectively.

VI. CONCLUSION

The results clearly indicate that no matter what design space parameters are chosen, the reconfigurable hardware implementation offers energy savings over the DSP and microcontroller implementations, with the more parallel implementations offering increased energy savings. However, the reconfigurable hardware implementation came with a cost of much higher design time, as simple C code cannot be used for such an implementation (as it can for the DSP and microcontroller). But, the fact that the energy consumption for the fully parallel IP Core implementation can offer 210X and 52X decrease in energy consumption over the microcontroller and DSP implementations respectively provides evidence that reconfigurable hardware can provide some energy benefit for to the overall acoustic modem design.

One might consider using an application specific integrated circuit (ASIC) as the hardware platform because ASICs, like

reconfigurable hardware allow for a custom, highly parallel implementation that can also optimize for energy efficiency. However, unlike FPGAs, microcontrollers, and DSPs, ASICs are not reconfigurable and are not commodity off the shelf parts, making them an expensive option for a low-cost modem.

Our future work includes performing energy optimizations (while keeping costs down) for the other parts of the underwater acoustic modem design including the analog front end, choice of signal processing scheme and interface to higher network levels. By optimizing energy at every level, we can eventually achieve a low-cost, low-energy acoustic modem to make the proliferation of underwater sensor networks a reality.

ACKNOWLEDGMENT

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Our thanks to Daniel Doonan, the UCSB AquaModem's hardware engineer for providing us with measurements for the DSP implementation.

REFERENCES

[1] Benthos, Inc. Fast and reliable access to undersea data. http://www.benthos.com/pdf/Modems/ModemBrochure.pdf.

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

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

[4] 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*. April 2006.

[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," *Proceeding of OCEANS Conference*, 2005.

[6] Ronald A. Iltis, Hua Lee, Ryan Kastner, Daniel Doonan, Tricia Fu, Rachael Moore and Maurice Chin, "An Underwater Acoustic Telemetry Modem for Eco-Sensing," *Proceedings of MTS/IEEE Oceans*, September 2005

[7] 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*, Sept. 2006.

[8] DSP Implementation of OFDM: H. Yan, S. Zhou, Z. Shi, and B. Li, "A DSP implementation of OFDM acoustic modem," in *Proc. of the ACM International Workshop onUnderWater Networks (WUWNet)*, Montreal, Quebec, Canada, September 14, 2007

[9] Taehyuk Kang and Ronald Iltis, "Matching Pursuits Channel Estimation for an Underwater Acoustic OFDM Modem" *in Proc of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing Special Session on Physcial Layer Challenges in Underwater Acoustic Communications*, Las Vegas, Nevada, April 2008

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

[11] Jurdak, C.V. Lopes, and P. Baldi. "Software Acoustic Modems for Short Range Mote-based Underwater Sensor Networks," In Proc. of IEEE Oceans Asia Pacific. Singapore. May, 2006.

[12] Rethem Mutlu Sözer, Milica Stojanovic, Reconfigurable acoustic modem for underwater sensor networks, Proceedings of the 1st ACM international workshop on Underwater networks, September 25-25, 2006, Los Angeles, CA, USA

[13] Kun, Zhong; Sen, Quek Swee; Aik, Koh Tiong; Aik, Tan Bien, "A Real-Time Coded OFDM Acoustic Modem in Very Shallow Underwater Communications," *OCEANS 2006 - Asia Pacific* , vol., no., pp.1-5, 16-19 May 2007 (10kbps, 1700m)

[14] Nasri, N.; Ben Hnia, H.; Kachouri, A.; Abdellaoui, M.; Samet, M., "Modulation/demodulation techniques with FPGA's architecture to improve OFDM wireless underwater communication transceiver," *Design and Test of Integrated Systems in Nanoscale Technology, 2006. DTIS 2006. International Conference on* , vol., no., pp. 400-403, Sept. 5-7, 2006

[15] R. Kastner, A. Kaplan, and M. Sarrafzadeh, *Synthesis Techniques and Optimizations for Reconfigurable Systems*. Boston: Kluwer Academic, 2004.

[16] William H. Mangione-Smith , Brad Hutchings , David Andrews, André DeHon , Carl Ebeling , Reiner Hartenstein , Oskar Mencer , John Morris, Krishna Palem , Viktor K. Prasanna , Henk A. E. Spaanenburg, Seeking Solutions in Configurable Computing, *Computer*, v.30 n.12, p.38-43, December 1997

[17] A. DeHon and J. Wawrzynek, "Reconfigurable computing: what, why, and implications for design automation," *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361). IEEE. 1999*, pp. 610-15.

[18] K. Bondalapati and V. K. Prasanna, "Reconfigurable computing systems," *Proceedings of the IEEE*, vol. 90, pp. 1201-17, 2002.

[19] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, pp. 171-210, 2002.

[20] P. Schaumont, I. Verbauwhede, K. Keutzer, and M. Sarrafzadeh, "A quick safari through the reconfiguration jungle," *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232). ACM. 2001*, pp. 172-7.

[21] Y. Meng, A.P. Brown, R.A. T.Sherwood, H.Lee, and R.Kastner, "IP Core: Algorithm and Design Techniques for Efficient Channel Estimation in Wireless Applications." *Design Automation Conference (DAC) 2005*

[22] S. F. Cotter and B.D. Rao. Sparse channel estimation via matching pursuit with application to equalization. *IEEE Trans. Communications*, 50:374-377, Mar. 2002

[23] S. Kim and R. A. Iltis. A matching pursuit/GSIC-based algorithm for DS-CDMA sparse channel estimation. *IEEE Signal Processing Letters*, 11:12-15, Jan. 2004

[24] J. Proakis, *Digital Communications*. New York, NY: McGraw-Hill, 1995

[25] L. Freitag, M. Stojanovic, S. Singh, and M. Johnson, "Analysis of channel effects on direct-sequence and frequency-hopped spread-spectrum acoustic communication," *IEEE Journal of Oceanic Engineering*, vol. 26, pp. 586-593, 2001

[26] Tricia Fu, Daniel Doonan, Chris Utley, Bridget Benson, Ryan Kastner, Ronald A. Iltis, and Hua Lee. Work In Progress Poster: "AquaModem Field Tests in Moorea" *International Workshop on Underwater Networks (WUWNet)*, September 2007

[27] M. Stojanovic and L. Freitag, "Acquisition of direct-sequence spread-spectrum acoustic communication signals," in *Proceedings of MTS/IEEE Oceans Conference 2003*, pp. 279–286, 2003

D. B. Kilfoyle and A. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 25, pp. 4–27, January 2000