Area-Power Efficient VLSI Implementation of Multichannel DWT for Data Compression in Implantable Neuroprosthetics

Awais M. Kamboh, *Student Member, IEEE*, Matthew Raetz, *Student Member, IEEE*, Karim G. Oweiss, *Member, IEEE*, and Andrew Mason, *Senior Member, IEEE*

***Abstract*—Time-frequency domain signal processing of neural recordings, from high-density microelectrode arrays implanted in the cortex, is highly desired to ease the bandwidth bottleneck associated with data transfer to extra-cranial processing units. Because of its energy compactness features, discrete wavelet transform (DWT) has been shown to provide efficient data compression for neural records without compromising the information content. This paper describes an area-power minimized hardware implementation of the lifting scheme for multilevel, multichannel DWT with quantized filter coefficients and integer computation. Performance tradeoffs and key design decisions for implantable neuroprosthetics are presented. A 32-channel 4-level version of the circuit has been custom designed in 0.18-μm CMOS and occupies only 0.22 mm^2 area and consumes 76 μW of power, making it highly suitable for implantable neural interface applications requiring wireless data transfer.**

***Index Terms*—Brain–machine interface (BMI), integer lifting wavelet transform, neural signal compression, VLSI design.**

I. Introduction

NEUROPROSTHETICS devices and brain–machine interfaces (BMIs) are increasingly playing a vital role in helping patients with severe motor disorders achieve a better lifestyle by enabling direct interface to the central nervous system at various levels. Recording the activity of cortical neurons with microelectrode arrays was shown to be essential to quantify their degree of involvement in encoding movement parameters, thereby permitting decoding of the neural signals to take place to control artificial limbs [1]. Such BMIs show great promise in many biomedical applications.

One particular challenge with BMI technology is the need to transmit the high bandwidth neural data from the implanted device to the outside world for further analysis. For example, a typical recording experiment with a 100-electrode array sampled at 25 kHz per channel with 12-bit precision yields an aggregate data rate of 30 Mbps, which is well beyond the reach of state-of-the-art telemetry links for biological applications. Another significant challenge is the need to fit implanted circuitry within $\sim$ 1 cm^2 for the entire signal processing system, and operate the chip at very low power (no more than 8–10 mW) to prevent temperature rise above 1 °C and avoid neural tissue damage [2].

For implanted microelectrode arrays, the power required to wirelessly transmit raw data to extra-cranial processing units is prohibitively large. Likewise, the hardware required to perform full neural data analysis is too complex to be implemented within an implanted system. Data compression before transmission is an attractive alternative, if it can be preformed with minimal hardware resources. Discrete wavelet transform (DWT) is a very effective method for compressing neural data [3], [4]. The resulting nonzero DWT coefficients give a sparse representation of the signal that greatly reduces the power required to upload data.

To utilize DWT-based compression with modern neuroprosthetics devices, a multichannel, multilevel implementation is necessary because microelectrode arrays sample multiple data channels simultaneously and multiple decomposition levels improve signal reproduction accuracy. Thus, area and power efficient hardware that can perform multichannel, multilevel DWT in real time is highly desirable. In contrast to traditional DWT applications, neuroprosthetics can afford long computation intervals, up to 40 μs [4], permitting hardware to prioritize power and area efficiency over speed. From the hardware point of view, integer-lifting and B-spline DWT factorization schemes have very efficient implementations. The lifting approach to the DWT reduces the required arithmetic operations and, as an in-place implementation, requires less memory at the expense of a longer critical path [5], [6]. The B-spline factorization reduces the critical path delay by converting some of the required multiply operations into less computationally intensive shift-and-add operations [6].

With few exceptions, recent efforts to optimize DWT hardware have concentrated on increasing throughput at the expense of area and power [5], [6]. In contrast, our prior work has identified an optimal implementation of the DWT architecture where chip area and power have priority over speed [7]. The approach relies on a hardware-efficient integer lifting factorization scheme and the "symmlet" family of wavelet basis that has been shown to provide a near optimal compression of neural signals [4].

This paper describes the design of a highly area and power efficient VLSI circuit that implements multichannel, multilevel lifting-based DWT in real time. Section II explains the architecture used to calculate the DWT while the design decisions for a computation core (CC) are detailed in Section III. Section IV

Manuscript received March 13, 2007; revised August 20, 2007. This work was supported in part by the National Institute of Health under Grant NS047516 and in part by the National University of Sciences and Technology, Islamabad, Pakistan. This paper was recommended by Associate Editor M. Sawan.

The authors are with the Electrical and Computer Engineering Department, Michigan State University, East Lansing, MI 48824 USA (e-mail: mason@msu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBCAS.2007.907557

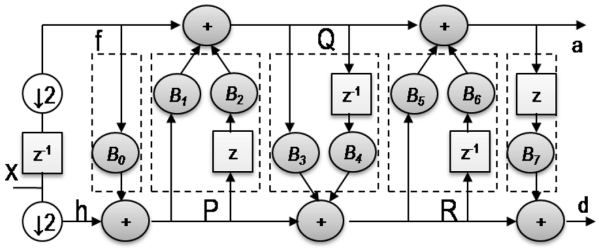


Fig. 1. Data flow diagram for lifting DWT using the symmlet 4 basis.

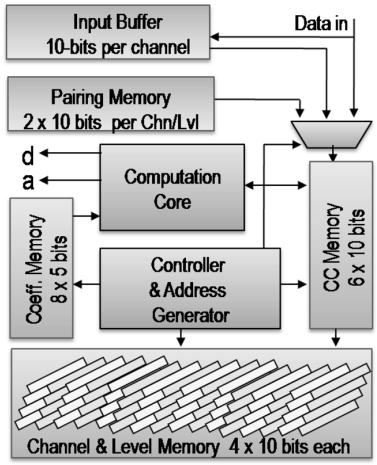


Fig. 2. Complete system diagram for sequential calculation of DWT.

describes optimizations in the memory modules, and Section V discusses controller operations. Results for a 32-channel 4-level DWT realization are summarized and performance of the system is analyzed in Section VI, followed by conclusions in Section VII.

II. Architecture

Fig. 1 shows the data flow diagram of lifting-based DWT using the symmlet 4 bases. If h and f are two sequential input samples from the same channel, a and d are approximation and detail results, and P, Q, and R are intermediate results, the five filter steps in this flow can be expressed by

$$\begin{aligned} P_0 &= h_0 + B_0 f_0 \\ Q_{-1} &= f_{-1} + B_1 P_{-1} + B_2 P_{-0} \\ R_{-1} &= P_{-1} + B_3 Q_{-1} + B_4 Q_{-2} \\ a_{-1} &= Q_{-1} + B_5 R_{-1} + B_6 R_{-2} \\ d_{-2} &= R_{-2} + B_7 a_{-1} \end{aligned} \tag{1}$$

where $B_0 - B_7$ are the eight constant filter coefficients and subscripts of the other variables represent the time sample, 0 being the current sample, -1 the previous sample, and so on.

To ease hardware requirements, it has been shown that integer lifting DWT with quantized data and filter coefficient values maintains a high signal to noise ratio. Based on this analysis, we have chosen 10-bit data and 5-bit coefficients (including sign bit) for our hardware implementation. Our previous work evaluated two structurally different hardware approaches, namely pipeline and sequential, for suitability in implantable devices based on their resource demands [2]. Derived from this system-level analysis, we have found that for the single channel case, the pipeline architecture consumes smaller power at the expense of considerably larger area as compared to the sequential architecture, resulting in a higher area-power product. The difference in area-power product increases with the increasing number of channels and levels, making the sequential architecture the better choice of the two especially for higher number of channels.

Fig. 2 describes the DWT architecture resulting from our circuit-level design efforts. It is composed of a customized CC, a digital controller, and five memory modules for incoming data, filter coefficients, intermediate CC products, and intermediate values necessary to sequentially process multiple levels and channels. Sequential data samples (inputs h and f) from a given channel are processed in pairs to generate the approximate and detail DWT coefficients (outputs a and d). To achieve system-level goals of power and area efficiency, each of these architectural blocks has been customized at the circuit level to minimize transistor count and eliminate unnecessary output transitions.

III. Computation Core

Analysis of the "symmlet4" lifting factorization and resulting equations in (1) show a noticeable computation regularity; all arithmetic operations can be expressed in the general form of $W = X + B_i Y + B_j Z$. This regularity can be exploited to minimize hardware by implementing a single CC that sequentially executes all computational steps in (1). Sequential reuse of the same hardware significantly reduces chip real estate requirements without impacting performance in this low bandwidth application.

The merits of using two's complement arithmetic versus sign-magnitude arithmetic were considered. The 10-bit input data is received in sign magnitude form, as are the constant 5-bit filter coefficients. The CC multipliers are required to perform 10×5 operations. Methods for multiplying sign-magnitude numbers do not work with two's complement numbers without adaptation. We have implemented several combinations of adders and multipliers using sign-magnitude and two's complement representation and found that it is most efficient to handle multiplication in sign-magnitude form while additions are performed in two's complement. As shown in Fig. 3, the CC can be implemented using two multipliers and a three-term adder formed by cascading two two-term adders. Using this CC to sequentially execute the steps in (1) requires five cycles to compute results for one sample pair. The critical path for this CC is $D_m + 2D_a$, where D_m and D_a are the delays of the multiplier and the adder, respectively. Overflows that can occur during these computations are handled by replacing the results with appropriate positive or negative saturation values.

A. Adders

Ripple carry–, carry save, and carry–look-ahead adders were analyzed for this DWT implementation. Because delay does not pose a bottleneck and area and power are much more important, the lower transistor count required by ripple carry adder was favored. Ripple carry adder performance depends largely upon structure of its individual full-adder cells, whose properties can

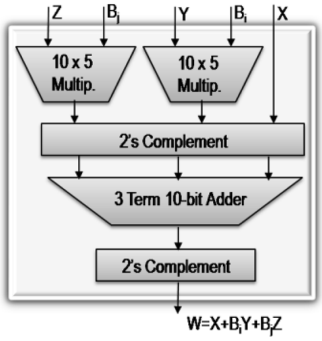


Fig. 3. Computation core architecture for integer lifting DWT used to compute (1).

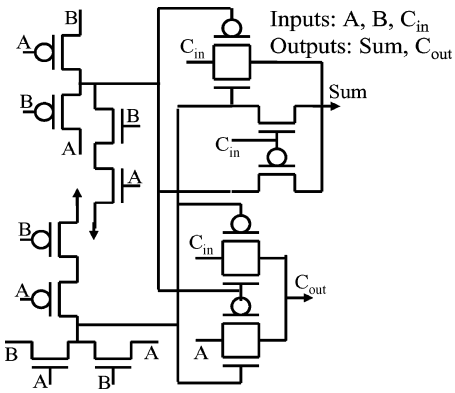


Fig. 4. Single bit full adder cell used in the CC adders based on pass transistor logic [8].

vary widely to match different applications. Comparative analysis of several full adder structures was performed at the circuit level to take into account transistor count, activity factor, and dynamic power due to parasitics. Based on these results, the pass gate logic 1-bit full adder presented in [8] and shown in Fig. 4 was chosen to best suit our DWT application. Unlike some adder cells with slightly lower transistor counts, this cell has no internal direct path between power and ground, eliminating short circuit current during transistor switching. Furthermore, the use of transmission gates limits the output voltage swing and reduces dynamic power consumption without compromising overall speed or reliability. In 0.18-μm CMOS, this 16 transistor cell was designed to dissipate 0.4 μW power and occupy 41.5 μm^2. It was utilized to construct 10-bit adders for the three-term adder block and 13-bit adders for the 10 $\times$ 5 multipliers.

B. Multipliers

The CC requires two multipliers to perform 10 $\times$ 5 operations. Booth multiplication algorithm is especially suited for two's complemented multiplication; however, it works well for block multiplications. Both Booth and array multiplier structures were optimized to the specific bit resolutions of our DWT application and compared for area and power. The Booth multiplier consists of recoders, partial product generators, 3:2 and 2:1 compressors and a final adder. The recoders and partial product generators were optimized for low power [9]. Two different recoders and partial product generators were considered, both requiring 18 transistors each. The two partial product generator architectures, NPR3a and NPR3b, presented in [9], were tailored to this application and it was determined that, while power consumption is similar, NPR3b needs slightly fewer transistors (14 versus 16) and is preferred. The 3:2 and 2:1 compressors are full and half adders respectively. In contrast, the array multiplier ANDs every bit of the multiplier with every bit of the multiplicand to create partial products. A Wallace tree structure was implemented to compress the partial products. Both Booth and array multipliers require an adder at the final stage. In comparison, a CC design using a Booth multiplier was shown to occupy 16% more area than a CC design with an array multiplier. Table I gives area, average power and delay measurements of the custom designed adder and array multiplier sub-modules selected for use in our DWT CC. It also includes the average power, area and delay measurements for the complete computational core including two's complement circuits. The average power includes static and dynamic power dissipation at a clock frequency of 6.4 MHz.

TABLE I
CC TRANSISTOR COUNTS AND WORST CASE VALUES

Sub-module	# of TX	Area (μm^2)	Power (μW)	Delay (ns)
10-bit Adder	160	418	2.07	0.41
10x5 Multiplier	733	2858	9.04	2.13
Comp. Core	1854	9828	27.36	6.07

IV. MEMORY MODULES

A. Multichannel/Level Memory Module

Neuroprosthetics applications generally depend upon multiple data streams taken from an array of microelectrodes. Thus, our DWT system has been designed to compress data from multiple channels simultaneously using multilevel decomposition in real time. Analysis of our DWT implementation indicates that there is sufficient bandwidth within a single CC to process well over 100 data channels sequentially. Sequential processing significantly reduces computational hardware demand; however, the intermediate values, or "state," of the current channel needs to be stored in memory so they are available when the CC later processes the next sample from this channel.

Similarly, memory is needed to store intermediate values while switching between different levels. The memory block required to hold these intermediate values is called the channel/level memory. The values that need to be stored and made available to process future samples are defined by (1). For each level and channel (beyond one), the lifting architecture requires five 10-bit values—a_0, f_0, P_0, Q_0, and R_0—to be saved, four of which are stored in channel/level memory while one is stored in the pairing memory. Every level and every channel requires a corresponding 40-bit memory register. In a 32-channel 4-level design, this amounts to 128 registers. The channel/level memory was implemented in both an SRAM using standard six transistor cells and a DRAM. For a large number of channels and levels, the channel/level memory module was found to dominate the power and area of the overall DWT system.

TABLE II
CC TRANSISTOR COUNTS AND WORST CASE VALUES

32 Chn/ 4 Lvl	No. of Tx	Total Area (μm^2)	Bit Density (μm^2/bit)
SRAM	34814	129830	25.36
DRAM	13414	46433	9.07

For the prototype DWT chip, the channel/level memory was implemented in SRAM to maximize reliability and ease system-level testing. However a DRAM implementation is particularly efficient in this application; due to the sequential nature of the DWT, the entire memory block is periodically overwritten, eliminating the need for refresh circuitry during normal operation. For example, at a sampling rate of 25 kHz, the least frequently accessed level 4 data is overwritten every 0.64 ms, which is much faster than the hold time easily achievable by a DRAM cell implemented in a standard CMOS process. This feature permits significant area savings over the SRAM implementation, as shown in Table II which compares relative performance criteria for a 32-channel 4-level memory block.

In a standard 0.18-μm CMOS process, use of a discrete DRAM storage capacitor is prohibitively area expensive. Alternatively, the gate capacitance of a large MOSFET has been used to store charge. Assuming that the sense amplifiers need a stored value of at least 1 V to reliably read the correct value, a MOSFET with at least 0.2-μm^2 gate area is required to realize a dependable hold time. Constructed in this fashion, comparison of two custom designed CMOS memory blocks shows that a DRAM block would permit a 64% area savings over an SRAM block for a 32-channel 4-level implementation.

B. Input, Coefficient, Pairing, and CC Memory Modules

Because DWT operates on data pairs, there is a hold cycle between each computation of input pairs wherein the first sample of the pair is acquired. Thus, an input data buffer of size equal to the number of channels is required to store input samples during the hold cycles. For the 32-channel implementation, the input buffer was constructed using FIFO shift registers. Note that in this architecture, the hold cycle is used to compute all results for higher (beyond one) levels of decomposition, where the necessary data is available from previous computations.

A separate coefficient memory block is required to store the eight 5-bit filter coefficients. Because this data is static, it is most efficiently implemented using hardwired connections, effectively operating as a ROM structure with only the electronics necessary to switch the proper coefficient values into the CC at the appropriate computation phase.

To accommodate sequential reuse of CC hardware, at the beginning of each computation cycle, four intermediate results from the previous computation cycle for the same channel must be loaded into the CC. These values, loaded from the channel/level memory module, are stored in a block defined as the CC memory. This CC memory block must be capable of loading all four bytes in parallel and moving data to the appropriate CC input as it cycles through the filter steps in (1). As described below, this requires six 10-bit registers that were implemented using flip-flops for parallel load/store and byte-wise shift operations.

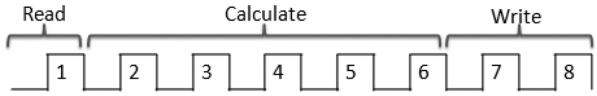


Fig. 5. DWT circuit operation phases for a single computation cycle.

When performing calculations for levels 2 and beyond, the CC input data (f and h in Fig. 2) does not come from inputs to the DWT circuit. Rather, inputs must be pulled from results calculated previously for a lower level. These values are stored in a pairing memory that must contain two 10-bit values for every channel and level except the highest level. All computation cycles except those of the highest level will generate one 10-bit byte to be written to this pairing memory. During all computation cycles for level 2 and beyond, two 10-bit values will be read from this block. Due to this unique read/write structure, this block was implemented independent of the channel/level memory using an SRAM with address decoding circuitry that allows us to enable each 10-bit byte block independently.

C. Power Saving Strategies for Memory

To reduce power consumption in the SRAM blocks, a divided bit line structure [10] was adopted, reducing the overall bit line capacitance and eliminating unnecessary dynamic power consumption during read and write cycles. Each of the sub bit lines were connected to only eight SRAM cells, and extra logic within the decoder controls access to these sub bit lines. With reduced bit line capacitance, the SRAM could be implemented without a sense amplifier, eliminating the need to access both sides of the SRAM cell and reducing bit line currents during read operations.

V. CONTROLLER

The main functions of the controller are to direct overall system operation and route data between DWT circuit blocks at the proper time [11]. The sequence of actions needed to complete a single DWT computation cycle consists of three different phases over a total of eight clock cycles, as shown in Fig. 5. As managed by the DWT controller, one read cycle is followed by five calculation cycles and then two write cycles. During the read cycle, stored values from prior calculations are loaded into the CC memory. During the calculation cycles, the five filter steps in (1) are executed. During the write cycles, results are stored onto channel/level and pairing memory blocks. This sequence must be repeated for each channel within the input sampling period. For example, with 32 data channels and a typical neural signal sampling rate of 25 kHz, the eight operation cycles must be clocked at 6.4 MHz. For fewer channels, the clock rate can be reduced to ensure power is only consumed when computations are required.

Note that the number of levels has no effect on the clock frequency because higher level results will be computed during intermediate hold cycles (see Section IV-B) while data input pairs are being stored. This is illustrated in Fig. 6, which describes the sequence of computation cycles managed by the controller for one channel and four levels. Here, each column represents a computation cycle composed of eight clock cycles. The number of computation cycles necessary for a complete, repeatable, DWT operation sequence is 2^L, where L is the number of levels. As shown in Fig. 6, four levels requires 16 computation

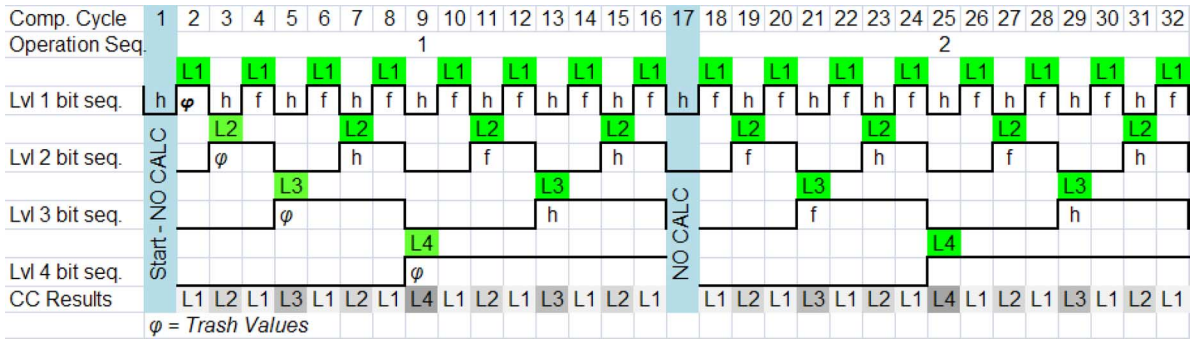


Fig. 6. Per channel activity of the CC at different levels of decomposition for a 4-level implementation. The top line shows the computation cycle count, and the next line counts the cycles within each operation sequence (two shown). A level 1 result is computed when a pair of data samples is received, a level 2 result is computed when two level 1 results are available, and so on. The first results calculated at each level are trash values because the memory initially contains meaningless values. After each 2^L computation cycles (where L is the number of levels), there is an idle (no calc) cycle.

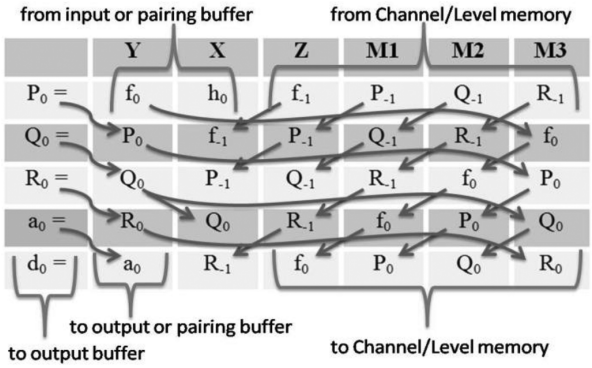


Fig. 7. Data movement in CC memory during the five calculation cycles within a computation cycle (one per sample per level). Top line shows the six register names. Subsequent lines define how data must be moved as the steps in (1) are processed sequentially.

cycles for a complete operation sequence. Within each operation sequence (2^L computation cycles) one cycle will be an idle cycle where no computations are necessary, as indicated by *NO CALC* in Fig. 6. Notice that, in each odd numbered computation cycle, no level one results are being calculated. These odd cycles correspond to input hold cycles discussed above. While input data pairs are being stored in these cycles (to be processed in the following cycle), the CC hardware can be utilized to process results for all levels greater than one in the sequence defined by Fig. 6. Thus, an infinite number of levels could be processed without additional hardware or increasing frequency. The number of channels has no impact on this sequence; each channel is processed sequentially within a single computation cycle, which can be achieved by increasing clocking frequency by a factor equal to the number of channels. Thus, the only major impact of increasing channels or levels is an increase in dynamic power consumption and the addition of channel/level memory.

To control the sequence and timing of operations within the DWT circuit, both an instruction based microprocessor and a state machine based controller were analyzed. Because of the relatively straightforward and repetitive nature of operations, the efficiency of a state machine design was found to be much more compatible with the power and area goals of this DWT circuit. The inherent tradeoff for this efficiency is limited flexibility, and the maximum number of levels and channels had to be set before implementing the state machine controller. Here we will describe a state machine for 32 channels and 4 levels. This design assumes multichannel input data is multiplexed into a single 10-bit input bus, and it will output results sequentially onto a single output bus.

For a 32-channel 4-level DWT system, the state machine controller utilizes a 12-bit counter to keep track of the current state, channel, and level. The three least significant bits of this counter determine the eight clock cycles within a computation cycle (Fig. 5) defining the operation phase for the CC and memory blocks, regardless of the channel or level being processed. The next five bits of the counter specify channel being processed. The remaining four bits represent the level being processed, where each bit represents an individual level (Fig. 6). The level 1 bit also determines if the input stream should go to the input buffer (hold cycle) or the CC (compute level 1 results). In this fashion, a level 1 result is computed once a pair of data samples is received; a level 2 result is computed when two level one results are available; and so on. By increasing the width of the counter, more states could be added to support additional channels or levels.

Fig. 7 describes operations within the CC memory controlled by the three least significant bits of the state machine counter. Here, complex data movement within the CC memory and its interaction with other memory blocks is controlled. Fig. 7 defines CC memory register names, where X, Y, and Z are registers connected to respective inputs of the CC while M1, M2, and M3 store intermediate computation results. Values are read into the CC memory at the beginning of the computation cycle. This is followed by five cycles of calculations and register data shifts. The last calculation cycle produces detail and approximation DWT results. The detail DWT coefficient is sent to the output bus. The approximation DWT coefficient is sent to the output bus and, depending on the level being processed, stored in pairing memory. After the last calculation cycle, four of the values in CC memory, f_0, P_0, Q_0, and R_0, are stored to the channel/level memory for use in subsequent calculations of the current channel/level results.

VI. Results and Analysis

A. Final Hardware Resources

All of the DWT circuit blocks were custom designed and laid out in 0.18-μm CMOS. Table III lists number of transistors required for each module. The corresponding chip area re-

TABLE III
TRANSISTOR COUNT AND AREA FOR HARDWARE MODULES

Module	No. of Tx	Area (μm^2)
Controller	987	6319
Computation Core	1854	9828
Comp. Core Memory (Flip-Flops)	2803	10453
Coefficient Memory (ROM)	122	714
*Pairing Mem. per Chn/Lvl (SRAM)	120*C*(L-1)	Varies
*Input Buffer per Chn (SRAM)	60*C	Varies
*Per Chn or Lvl Mem (SRAM)	240*C*L	Varies
Complete DWT system: 8 Chn, 4 Lvl	17910	74253
Complete DWT system: 32 Chn, 4 Lvl	54339	221643
Complete DWT system: 100 Chn, 4 Lvl	157562	641344
Complete DWT system: 250 Chn, 4 Lvl	385256	1583832

∗Counts exclude transistors for address decoding circuitry

TABLE IV
AVERAGE ACTIVITY PER CYCLE FOR HARDWARE MODULES

Module	Average Activity per cycle
Controller	8/8 = 1
Comp Core	5/8
Comp Core Memory (Flip Flops)	8/8 = 1
Coefficient Memory (ROM)	5/8
Pairing Memory per channel/level	2/8 = ¼
Input Buffer per channel (SRAM)	1/8
Per Channel or Level Memory	2/8 = ¼

quired by each module, including routing, is also shown. The dominance of channel/level memory over other modules is evident, where increasing the number of channels by a factor of four (from 8 to 32) results in roughly a 300% increase in both the number of transistors and the area consumption. Our model 32-channel, 4-level DWT implementation, requires ∼54 k transistors and occupies roughly 470 μm × 470 μm. These values were obtained with the major memory blocks implemented with SRAM for reliability and testing purposes. If the input buffer, pairing and channel/level memory were replaced by DRAM blocks, the estimated transistor count for the 32-channel, 4-level circuit would drop to around 25 000 and the total DWT circuit area would reduce to a little more than 100 000 μm^2, which is less than 50% of the overall circuit using SRAM. Table III also gives the area and transistor count precise estimates for 100- and 250-channel designs.

A 32-channel system, with a sampling frequency of 25 KHz, operating at 4 levels of DWT decomposition requires a clock frequency of 6.4 MHz. The power is managed separately for each block, thus the hardware portions not working at a given time do not consume any power. As described in Section V, operation of the DWT block can be modeled as a state machine which requires 8 cycles to complete its operation, as shown in Fig. 5. Table IV gives the average activity per cycle of each hardware module.

If the maximum delay of 10 ns is considered to allow for effects of fabrication non idealities, it allows us time to process 500 channels, at 25 KHz sampling frequency, in a time-multiplexed fashion. However handling of 500 channels does not seem feasible especially at the analog-to-digital converters and the neural signal amplifiers.

The overall 32-channel 4-level system at 0.18-μm process technology with 1.3 V $V_{\rm DD}$ and at 6.4 MHz of operating frequency consumes an average power of 76 μW, including static and dynamic power dissipation, with highest power being consumed by CC memory, the CC and the controller respectively. Since these three modules are operational most of the time, they account for about 80% of the total power.

B. Data Compression Versus MSE

The DWT circuit outputs interleaved approximate and detail transform coefficients that give a sparse representation of the original neural signal. Coefficient values below a specific threshold can be set to zero to compress the results into a smaller number of bytes. The nonzero coefficients can then be encoded using a lossless encoding scheme and transmitted to the extracranial or extra-cutaneous processing units [4], [12]. Choosing the value of the zeroing threshold provides a tradeoff between signal integrity and compression ratio.

To test our DWT circuit implementation on real neural data, a linear-step analog to digital converter was used to convert experimentally obtained neural data into a stream of 10-bit digital values. The data was then processed through the DWT system and results were stored for analysis. In one test, the stored transform coefficients were used to reconstruct the neural signal, and this result was compared to the original signal to measure the quality of reconstruction. This analysis was performed for several different zeroing threshold values to evaluate signal quality verses the amount of compression obtained. The final performance metrics can be defined in terms of three quantities, the root mean squared error (RMS), Shannon's entropy [13], and the assigned threshold.

RMS error is a measure of the average difference between the original and the reconstructed signal. Here the difference between the original signal and the reconstructed signal is comprised of two components: the quantization noise due to finite-word length, and the lossy-compression noise resulting from the thresholding operation. The mathematical representation of RMS error is given by

$$\text{RMS\_error} = \sqrt{\frac{1}{N}\sum_{n}^{N}(x_n - \hat{x}_n)^2} \tag{2}$$

where x_n is the original signal, $\hat{x}_n$ is the reconstructed signal, and N is the length of the signal.

Shannon's entropy is a measure of uncertainty associated with a random signal and can be interpreted as the average minimum message length, represented in bits, which must be transmitted to communicate the exact value of the signal.

Entropy gives the theoretical limit to the best possible lossless data compression for any communication. Entropy can be mathematically represented as

$$H(X) = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i) \tag{3}$$

where N is the total number of possible values (also called *symbols* in information theory literature) and $p(x_i)$ is the probability of occurrence of the ith value.

Increasing the zeroing threshold improves the amount of compression achieved (decreasing entropy) but degrades MSE. For neural data processed through our DWT circuit, Fig. 8 plots the MSE and entropy as the zeroing threshold increases and confirms the anticipated tradeoff. Because of this direct

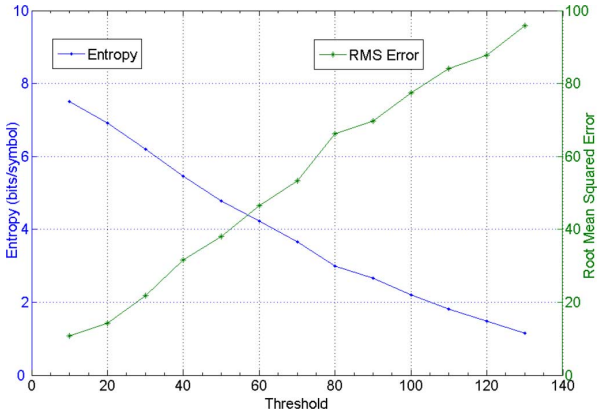


Fig. 8. RMS error and entropy in bits per symbol as a function of threshold value for the neural data set used in our experiments.

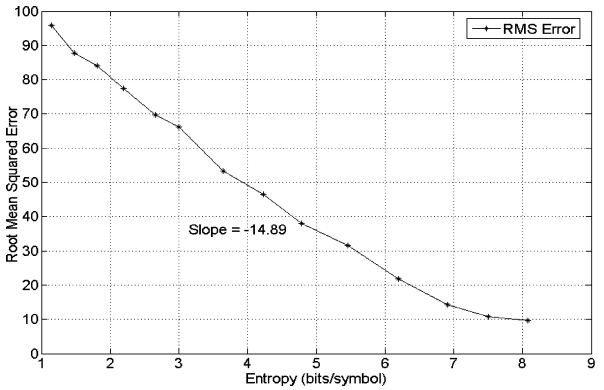


Fig. 9. Relation between RMS error and entropy, where the independent variable is the threshold.

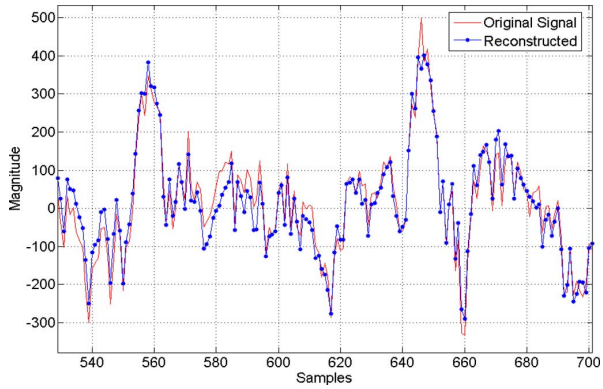


Fig. 10. Original and reconstructed signals for Threshold = 50, RMS error = 37.95 and Entropy = 4.78 bits per symbol at 4 levels of decomposition with 10-bits data and 5-bits coefficients.

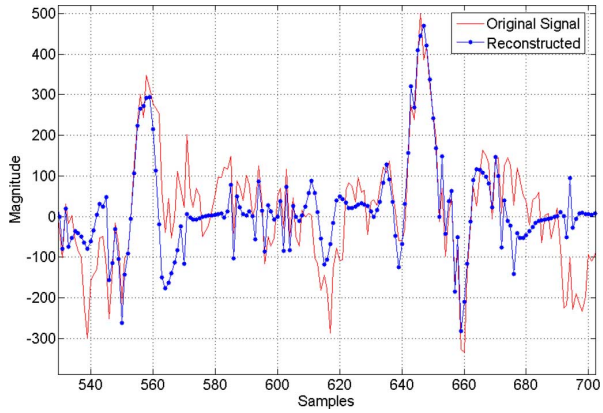


Fig. 11. Original and reconstructed signals for Threshold = 120, RMS error = 87.69 and Entropy = 1.48 bits per symbol at 4 levels of decomposition with 10 bits data and 5 bits coefficients.

tradeoff between MSE and compression ratio, the zeroing threshold must be chosen to match application requirements; if bandwidth limitations are not severe, a low threshold could maximize signal reconstruction, or if near perfect reconstruction is not needed, a high threshold could reduce power for data transmission. Fig. 9 shows the relation between the RMS error and the entropy both of which are dependent on the threshold value. Figs. 10 and 11 show the original and reconstructed signals for threshold values of 50 and 120, respectively, as obtained in our experiments with real neural data. At a threshold of 50, entropy of 4.78 bits per symbol and an RMS error of 37.95 were obtained. For a threshold value of 120, entropy of 1.48 bits per symbol and an RMS error of 87.69 were measured.

It can be seen that a low threshold value, as in Fig. 10 results in a good signal reconstruction so it is suitable for applications that require high quality signal reconstruction. A higher threshold value, as in Fig. 11, results in a relatively worse reconstruction; however, it still captures the spikes in the neural signal very effectively and preserves their shapes and sizes. Since most of the neural signal analysis and applications are based on the information embedded in the spikes only, even this level of reconstruction is highly accurate and useful. Thus, depending on the specific application requirements, the threshold value can be chosen to optimally exploit available bandwidth, available power, and desired reconstruction quality.

VII. Conclusion

A custom VLSI hardware implementation of lifting-based, multichannel, multilevel DWT using "symmlet 4" wavelet filters for neural signal processing was presented. This circuit minimizes chip area and power consumption by sequentially evaluating integer filter equations with quantized coefficients using a highly efficient CC. For this implementation, ripple-carry adders structures with transmission gate-based full adder cells, and a Wallace tree 10×5 array multipliers were found to be the most efficient. Four different memory blocks were identified and implemented using a combination of SRAM, ROM, and flip-flop registers to maximize power and area efficiency. A repeatable sequence for multichannel, multilevel DWT evaluation was described to define system operation and interactions between circuit blocks. A state machine based controller was presented to manage the complex data flow with significantly less hardware than an instruction-base controller. The resulting DWT circuit can pseudo-simultaneously process multiple channels of incoming neural signals in real time, and its ability to perform multi level DWT enables very high data compression while maintaining signal fidelity. A 54 k transistor model implementation in 0.18-μm CMOS requires only 0.22 mm^2 of chip area and 75 μW of power to process 32 channels of data at 4 levels of DWT. Finally, the DWT circuit was used to process real neural data, and the MSE versus

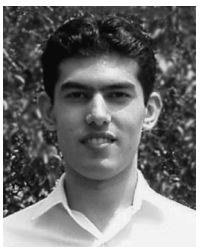
compression ratio tradeoff was analyzed over many values of zeroing threshold. The small size and low power consumption of this DWT VLSI circuit makes it highly suitable for font-end data compression in implantable applications.

ACKNOWLEDGMENT

The authors would like to thank T. Canfield for his valuable contributions towards this research.

REFERENCES

[1] M. Nicolelis, "Actions from thoughts," *Nature*, vol. 409, 2001, pp. 403–107.
[2] K. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh, and K. Thomson, "A scalable wavelet transform VLSI architecture for real-time signal processing in multichannel cortical implants," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 54, no. 6, pp. 1266–1278, Jun. 2007.
[3] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1998.
[4] K. G. Oweiss, "A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 7, pp. 1364–1377, Jul. 2006.
[5] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient architectures for 1-D and 2-D lifting-based wavelet transforms," *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1315–1326, May 2004.
[6] C. T. Huang, P. C. Tseng, and L. G. Chen, "VLSI architecture for forward discrete wavelet transform based on B-spline factorization," *J. VLSI Signal Process.*, pp. 343–353, 2005.
[7] A. M. Kamboh, A. Mason, and K. G. Oweiss, "Comparison of lifting and B-spline DWT implementations for implantable neuroprosthetics," *J. VLSI Signal Process. Syst.*, to be published.
[8] A. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 1, pp. 20–29, Feb. 2002.
[9] Z. Huang, "High-level optimization techniques for low-power multiplier design," Ph.D. dissertation, Comp. Sci. Dept., Univ. California, Los Angeles, 2003.
[10] B.-D. Yang and L.-S. Kim, "A low power SRAM using hierarchical bit line and local sense amplifiers," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1366–1376, Jun. 2005.
[11] A. M. Kamboh, M. Raetz, A. Mason, and K. Oweiss, "Area-power efficient lifting-based DWT hardware for implantable neuroprosthetics," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 2371–2374.
[12] K. G. Oweiss, Y. Sohail, K. Thomson, J. Li, and A. Mason, "Augmenting real-time DSP in implantable high-density neuroprosthetics devices," in *Proc. IEEE/EMBS Special Topic Conf. Microtechnol. Med. Biol.*, 2005, pp. 108–111.
[13] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.



Awais M. Kamboh (S'07) received the B.S. degree (hons.) in electrical engineering from National University of Sciences and Technology, Islamabad, Pakistan, in 2003, and the M.S. degree in electrical engineering systems from the University of Michigan, Ann Arbor, in 2006. He is currently working towards the Ph.D. degree at Michigan State University, East Lansing.

His research interests include signal processing, multimedia communications, VLSI and SoC design.

Mr. Kamboh is a student member of the IEEE Communications Society, IEEE Circuit and Systems Society, and the IEEE Engineering in Medicine and Biology Society.

Matthew Raetz (S'04) is currently working toward the B.S. degree in electrical engineering from Michigan State University, East Lansing, where he has been actively engaged in research with the Advanced MicroSystem and Circuits (AMSaC) research group.

He has worked on custom mixed-signal circuits for neural signal processing and low power digital control circuits for sensor-based microsystems. His research interests include system design and low power integrated circuits.

Karim G. Oweiss (S'95–M'02) received the B.S. and M.S. degrees (hons.) in electrical engineering from the University of Alexandria, Alexandria, Egypt, in 1993 and 1996, respectively, and the Ph.D. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, in 2002.

He completed postdoctoral training with the Biomedical Engineering Department, University of Michigan, Ann Arbor, in the summer of 2002. In August 2002, he joined the Department of Electrical and Computer Engineering and the Neuroscience program at Michigan State University, where he is currently an Assistant Professor and Director of the Neural Systems Engineering Laboratory. His research interests span diverse areas that include statistical and multiscale signal processing, information theory, machine learning as well as modeling the dynamics of the nervous system, neural integration and coordination in sensorimotor systems, and computational neuroscience.

Dr. Oweiss is a member of the IEEE Society for Neuroscience. He is also a member of the Board of Directors of the IEEE Signal Processing Society on Brain Machine Interfaces, the Technical Committees of the IEEE Biomedical Circuits and Systems, the IEEE Life Sciences, and the IEEE Engineering in Medicine and Biology Society. He was awarded the excellence in Neural Engineering award from the National Science Foundation in 2001.

Andrew Mason (S'90–M'99–SM'06) received the B.S. degree in physics (highest distinction) from Western Kentucky University, Bowling Green, in 1991, the B.S.E.E. degree (hons.) from the Georgia Institute of Technology, Atlanta, in 1992, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, in 1994 and 2000, respectively.

From 1997 to 1999, he was an Electronic Systems Engineer at a small aerospace company, and from 1999 to 2001, he was an Assistant Professor at the University of Kentucky. In 2001, he joined the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, where he is currently an Associate Professor. His research addresses many areas of mixed-signal circuit design and the fabrication of integrated microsystems. Current projects include adaptive sensor interface circuits, bioelectrochemical interrogation circuits, post-CMOS fabrication of electrochemical sensors, and integrated circuits for neural signal processing.

Dr. Mason serves on the Sensory Systems and Biomedical Circuits and Systems Technical Committees of the IEEE Circuits and Systems Society and the on the Technical Program Committee for IEEE International Conference on Sensors. In 2006, he received the Michigan State University Teacher-Scholar Award.