A Scalable Wavelet Transform VLSI Architecture for Real-Time Signal Processing in High-Density Intra-Cortical Implants

Karim G. Oweiss, *Member, IEEE*, Andrew Mason, *Senior Member, IEEE*, Yasir Suhail, *Student Member, IEEE*, Awais M. Kamboh, *Student Member, IEEE*, and Kyle E. Thomson

Abstract—This paper describes an area and power-efficient VLSI approach for implementing the discrete wavelet transform on streaming multielectrode neurophysiological data in real time. The VLSI implementation is based on the lifting scheme for wavelet computation using the symmlet4 basis with quantized coefficients and integer fixed-point data precision to minimize hardware demands. The proposed design is driven by the need to compress neural signals recorded with high-density microelectrode arrays implanted in the cortex prior to data telemetry. Our results indicate that signal integrity is not compromised by quantization down to 5-bit filter coefficient and 10-bit data precision at intermediate stages. Furthermore, results from analog simulation and modeling show that a hardware-minimized computational core executing filter steps sequentially is advantageous over the pipeline approach commonly used in DWT implementations. The design is compared to that of a B-spline approach that minimizes the number of multipliers at the expense of increasing the number of adders. The performance demonstrates that *in vivo* real-time DWT computation is feasible prior to data telemetry, permitting large savings in bandwidth requirements and communication costs given the severe limitations on size, energy consumption and power dissipation of an implantable device.

Index Terms—B-spline, brain machine interface, lifting, micro-electrode arrays, neural signal processing, neuroprosthetic devices, wavelet transform.

I. INTRODUCTION

VLSI implementation of the discrete wavelet transform (DWT) has been widely explored in the literature as a result of the transform efficiency and applicability to a wide range of signals, particularly image and video [1], [2]. These implementations are generally driven by the need to fulfill certain characteristics such as regularity, smoothness and linear

phase of the scaling and wavelet filters, as well as perfect reconstruction of the decomposed signals [3].

In some applications, it is desirable to meet certain design criteria for VLSI implementation to enhance the overall system performance. For example, minimizing area and energy consumption of the DWT chip is highly desirable in wireless sensor network applications where resources are very scarce. In addition to miniaturized size, minimizing power dissipation is strongly sought to minimize tissue heating in some biomedical applications where the chip needs to be implanted subcutaneously.

In this paper, we deal primarily with the design of DWT VLSI architecture for an intracortical implant application. Motivated by recent advances in microfabrication technology, hundreds of microelectrodes can be feasibly implanted in the vicinity of small populations of neurons in the cortex [4], [5], opening new avenues for neuroscience research to unveil many mysteries about the connectivity and functionality of the nervous system at the single cell and population levels. Recent studies have shown that the activity of ensembles of cortical neurons monitored with these devices carry important information that can be used to extract control signals to drive neuroprosthetic limbs, thereby improving the lifestyle of severely paralyzed patients [6]–[8].

One particular challenge with the implant technology is the need to transmit the *ultra-high* bandwidth neural data to the outside world for further analysis. For example, a typical recording experiment with a 100 microelectrode 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 wireless telemetry. Other significant challenges consist of the need to fit circuitry within ~ 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 that may cause neural tissue damage. In previous studies, we have shown that the DWT enables efficient compression of the neural data while maintaining high signal fidelity [9]–[11]. To be implemented in an actual implanted device, chip size, computational complexity and signal fidelity must be balanced to create an optimal application-specific integrated circuit (ASIC) design tailored to this application.

Generally speaking, the case of computing the DWT for high throughput streaming data has not been fully explored [12]. It has been argued that a lifting scheme [13] provides the fewest arithmetic operations and in-place computations, allowing larger savings in power consumption but at the expense of
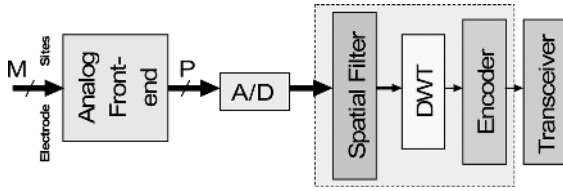
Fig. 1. Block diagram of an implantable neural system illustrating the mixed signal processing proposed.

longer critical path than that of convolution-based ones [13]. Recent work by Huang *et al.* [14] focused on analyzing DWT architectures with respect to tradeoffs between critical path and internal buffer implementations. Such critical path can be shortened using pipelining with additional registers or using a so-called *flipping* structure with fixed number of registers [15]. The *B-spline* approach [16], on the other hand, requires fewer multipliers than lifting, replacing them with adders that may permit a smaller chip area [17]. Nonetheless, most of the reported hardware approaches focus on computational speed and do not adequately address severe power and area constraints. By comparing with other implementations of the DWT in this paper, we demonstrate that the appropriate compromise among power, size and speed of computations is achieved with a sequential implementation of integer arithmetic lifting approach.

The paper is organized as follows. In Section II, the classical single channel one-dimensional (1-D) DWT and lifting DWT are introduced. Section III describes the motivation for integer lifting DWT and approaches to efficiently map the algorithm to hardware for a single channel, single level DWT decomposition. In Section IV, proposed architectures for integer lifting are described and analysed. Section V describes hardware considerations of the proposed architecture for multiple channels and multiple levels of decomposition, and Section VI describes performance comparisons and overall results.

II. THEORY

A typical state-of-the-art implantable neural interface system as depicted in Fig. 1 contains an analog front end consisting of pre-amplification, multiplexing and A/D conversion prior to extra-cutaneous transmission. An analog front end integrated onto a 64-electrode array would occupy 4.3 mm^2 in 3 μm technology and would dissipate 0.8 mW of power [5]. This traditional approach is not well suited for wireless data transmission due to power demands associated with the resulting large data throughput. In the proposed approach, the power and chip area of the analog front end is reduced by using contemporary mixed-signal VLSI design approaches and more modern fabrication processes (e.g., 0.18 μm), allowing advanced signal processing to take place within the implanted system without significant increase in the chip size. Power- and area-efficient implementations of the spatial filter, the DWT, and the encoder blocks would provide on-chip signal processing and data compression, enabling wireless transmission by reducing bandwidth requirements. In this paper, we only discuss VLSI implementation of the DWT block.
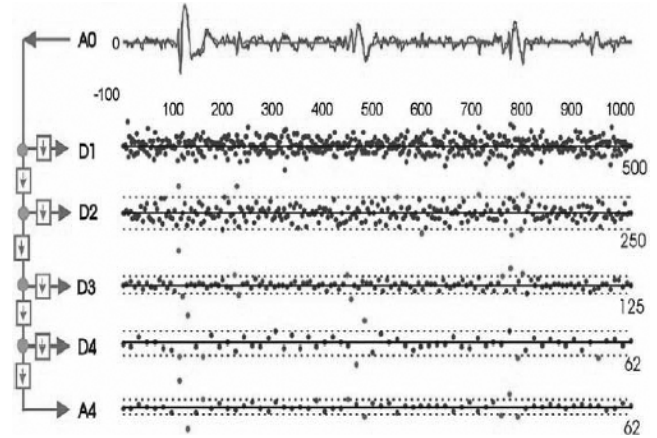

Fig. 2. 4-level DWT of a single channel noisy neural trace (blue) using symmlet4 basis. The original signal labeled A0 is in the top trace. The largest transform coefficients (in red) that survive the denoising threshold are used to approximate the original signal shown in red in the top trace [11]. The original data length is 1024 samples ($\sim$ 40 ms at 25-kHz sampling frequency).

A. Pyramidal Single Channel DWT

The classical, convolution-based, dual-band DWT of a given signal involves recursively convolving the signal through two decomposition filters $L(z)$ and $H(z)$, and decimating the result to obtain the approximation and detail coefficients at every decomposition level j. These filters are derived from a scaling function and a wavelet function that satisfy subspace decomposition completeness constraints [18]. A typical FIR low pass and high pass 3-tap filter is expressed as

$$L(z) = l_0 + l_1 z^{-1} + l_2 z^{-2} \qquad (1)$$
$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2}. \qquad (2)$$

So that the approximation and detail coefficients $\underline{a}^j$ and $\underline{d}^j$, respectively, at the jth level can be computed as

$$a^j(i) = \sum_{k=0}^{K-1} a^{j-1}(2i-k)l_k \qquad (3)$$

$$d^j(i) = \sum_{k=0}^{K-1} a^{j-1}(2i-k)h_k \qquad (4)$$

where K is the number of filter taps. The obtained coefficient vectors $\underline{a}^j$ and $\underline{d}^j$ are $N/2^j$-dimensional, where N is the length of the original input sequence. Equation (3) and (4) describe the original pyramidal algorithm reported by Mallat [18]. Reconstruction of the original sequence from the DWT coefficients is achieved through

$$a^{j-1}(2i-1) = \sum_{k=0}^{(K/2)-1} a^j(i-k)\tilde{l}_k \qquad (5)$$

$$d^{j-1}(2i-1) = \sum_{k=0}^{(K/2)-1} a^j(i-k)\tilde{h}_k \qquad (6)$$

where $\tilde{l}_k$ and $\tilde{h}_k$ are the coefficients of the synthesis filters, respectively. These are related to the analysis filters through the 2-scale equation [18].

An example of the DWT decomposition of a single channel neural trace is illustrated in Fig. 2. The useful information is
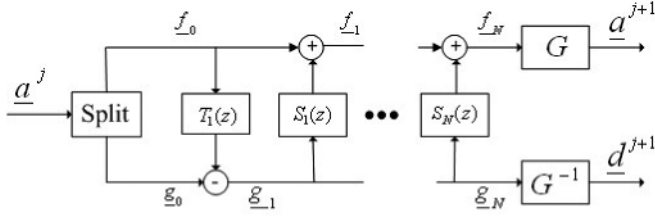
Fig. 3. Lifting-scheme for computing a single level DWT decomposition [13]. The polynomials $T_n(z)$ and $S_n(z)$ are obtained through factorization of the wavelet filters $L(z)$ and $H(z)$, respectively.

mostly contained in the short transients -or spikes- above the noise level that result from the activity of an unknown number of neurons. It can be observed that the sparsity introduced by the DWT compaction property enables very few "large" coefficients to capture most of the spikes' energy, while leaving many "small" coefficients attributed to noise. This property permits the later ones to be thresholded [19], yielding the denoised signal shown.

For near-optimal data compression, a wavelet basis needs to be selected to best approximate the neural signal waveform with the minimal number of data coefficients. A compromise between signal fidelity and ease of hardware implementation has to be made. A near-optimal choice was proposed in [9] from a compression standpoint and demonstrated that the *biorthogonal* and the *symmlet*4 wavelet functions are advantageous over other wavelet basis families for processing neural signals. From a hardware implementation viewpoint, the *symmlet*4 family has much smaller support size for similar number of vanishing moments compared to the biorthogonal basis [20]. In addition, they can be implemented in $O(N)$ operations.

B. Single Channel Lifting-Based Wavelet Transform

The lifting scheme [12] illustrated in Fig. 3 is an alternative approach to computing the DWT. It is based on three steps: First, splitting the data at level j into *even* and *odd* samples $\underline{f}_0$ and $\underline{g}_0$, respectively; Second, predicting the odd samples from the even samples such that the prediction error becomes the high pass coefficients $\underline{g}_1$; and third, updating the even samples $\underline{f}_0$ with $\underline{g}_1$ to obtain the approximation coefficients $\underline{f}_1$. This process is repeated N times. At an arbitrary prediction and update step n, the prediction and update filters $T_n(z)$ and $S_n(z)$, respectively, are obtained by factorizing the wavelet filters $L(z)$ and $H(z)$ into n lifting steps. The data at each step, after applying the new filters are labeled as $\underline{f}_0, \underline{f}_1, \ldots, \underline{f}_N$ and $\underline{g}_0, \underline{g}_1, \ldots, \underline{g}_N$, respectively. The last step is a multiplication by a scaling factor G to obtain the approximation and details $\underline{a}^{j+1}$ and $\underline{d}^{j+1}$ of the next level.

A lifting factorization of the *symmlet*4 wavelet basis amounts to the following filtering steps:

$$\begin{aligned}
\text{step1}: \quad & g_1[i] = g_0[i] + C_1 \times f_0[i] \\
\text{step2}: \quad & f_1[i] = f_0[i] + C_2 \times g_1[i] + C_3 \times g_1[i+1] \\
\text{step3}: \quad & g_2[i] = g_1[i] + C_4 \times f_1[i] + C_5 \times f_1[i-1] \\
\text{step4}: \quad & a[i] = f_1[i] + C_6 \times g_2[i] + C_7 \times g_2[i-1] \\
\text{step5}: \quad & d[i] = g_2[i] + C_8 \times a[i+1] \quad\quad\quad\quad (7)
\end{aligned}$$

TABLE I
SYMMLET-4 DWT LIFTING COEFFICIENTS AND THEIR 6-BIT (5-BIT + SIGN) INTEGER APPROXIMATIONS

	Floating point value	6-bit integer		Floating point value	6-bit integer
C_1	0.39114	6	C_5	0.16203	3
C_2	-0.12439	-2	C_6	0.43128	7
C_3	-0.33924	-5	C_7	0.14598	2
C_4	-1.41951	-23	C_8	-1.04925	-17

TABLE II
SYMMLET-4 DWT B-SPLINE COEFFICIENTS AND THEIR 6-BIT (5-BIT + SIGN) INTEGER APPROXIMATIONS

	Floating point value	6-bit integer		Floating point value	6-bit integer
B_1	-3.6069	-7	B_4	4.3913	9
B_2	1.8668	4	B_5	8.4845	17
B_3	-0.4248	-1	B_6	2.3385	5

where the intermediate values $f_1[i]$, $g_1[i]$, and $g_2[i]$ are discarded after being used, $f_2[i] = a[i]$ is the resulting approximation coefficient, $g_3[i] = d[i]$ is the resulting detail, and C_1 through C_8 are the coefficients of the prediction and update filters listed in Table I.

C. Single Channel B-Spline Based Wavelet Transform

Alternatively, a B-spline approach for DWT computation [16] is based on factorizing the filters as

$$\begin{aligned}
L(z) &= (1 + z^{-1})^{\gamma_l} R(z) \cdot l_o \\
H(z) &= (1 - z^{-1})^{\gamma_h} Q(z) \cdot h_o \quad\quad (8)
\end{aligned}$$

where $Q(z)$ and $R(z)$ are known as the distributed parts, h_o and l_o are normalization factors [17], γ_l and γ_h are the orders of the *B-spline* parts, respectively. For the *symmlet*4, this factorization can be expressed as

$$\begin{aligned}
L(z) &= (1 + z^{-1})^4 (1 + B_1 z^{-1} + B_2 z^{-2} + B_3 z^{-3}) \\
H(z) &= (1 - z^{-1})^4 (1 + B_4 z^{-1} + B_5 z^{-2} + B_6 z^{-3}) \quad (9)
\end{aligned}$$

where the coefficients B_1 through B_6 are listed in Table II. Since the B-spline parts in both filters can be expressed as

$$(1 \pm z^{-1})^4 = (1 + 6z^{-2} + z^{-4}) \pm (4z^{-1} + 4z^{-3}) \quad (10)$$

they can be typically implemented using simple shifting and addition. The polyphase decomposition similar to lifting can therefore be performed on the distributed parts $R(z)$ and $Q(z)$ [16]. This is achieved by splitting the distributed parts into odd and even components $R_e(z)$ and $R_o(z)$, $Q_e(z)$ and $Q_o(z)$, respectively. For example, the low-pass even distributed part can be represented as $R_e(z) = 1 + B_2 z^{-2}$, and likewise for the remaining components. The benefit in the B-spline method is a reduction in the number of floating point multiplications at the expense of more additions [17]. Table III compares the computational requirements of lifting and B-spline DWT implementations along with traditional convolution. In B-spline, four x4 multiplications are replaced by shifts and two x6 multiplications are replaced by shifts and additions (x2+x4). Relative to lifting,

	Lifting	B-spline	Convolution
Multiplications	8	6	14
Additions	8	18	14

B-spline requires two fewer multiplications at the expense of ten more additions for one level of decomposition. Nevertheless, as the detailed low-power/area DWT implementation below will show, any benefit to B-spline is diminished for multilevel multichannel decomposition.

D. Hardware Considerations

Power and area requirements of the DWT hardware are determined largely by the complexity of the computational circuitry and the required memory. To systematically reduce hardware requirements, we have explored different options to reduce computation and memory requirements at the algorithm level and analyzed their impact on signal integrity to determine an optimal approach. We summarize below two key ideas that contribute largely to the reduction of circuit complexity and memory requirements that are discussed in subsequent sections, while more details of this analysis are further provided in Section V.

1) Integer Approximation: Fixed-point integer approximation limits the range and precision of data values but greatly reduces the computational demand and memory requirements for processing and storage. To explore the potential of utilizing integer approximation in the proposed system, we observed that neural signal data will be entering the system through an A/D converter and will thus inherently be integer valued within a prescribed range. The data is first scaled to obtain data samples within a 10-bit integer precision. The integer approximation is then computed for the scaled data. The integer-to-integer transformation [22] involves rounding-off the result of the lifting filters T and S that are used to filter odd and even data samples, respectively. The last step that requires scaling by G and G^{-1} is omitted. Hence, the dynamic range of the transform at each level will now change by G. As our results will demonstrate (Section V), the minimized circuit complexity associated with integer representation should be well suited to this application provided that data precision is sufficient to maintain signal integrity.

2) Quantization of the Filter Coefficients: Rounding-off wavelet filter coefficient values to yield a fixed point integer precision format can further reduce the computation and memory requirements. Implementing lifting-based wavelet transform with only integer computational hardware requires the filter coefficients be represented as integers along with the sampled data. Tables I and II show the scaled filter coefficients C_1-C_8($\mathbf{x}16$) and B_1-B_6($\mathbf{x}2$) for the *symmlet*4 basis. These coefficients are further quantized into integer values. The level of quantization has a significant impact on the complexity of computational hardware. We quantified the effect of the round off and quantization errors on the signal fidelity as a function of multiplier complexity [21]. Our results (Section V) demonstrate that 6 bits (5 bits + 1 sign bit) coefficient quantization can adequately preserve signal integrity.

III. SINGLE-CHANNEL SINGLE-LEVEL HARDWARE DESIGN

In a first-order analysis, the area of a CMOS integrated circuit is proportional to the number of transistors required, and power consumption is proportional to the product of the number of transistors and the clocking frequency. Through transistor-level custom circuit design, circuit area and power consumption can be further reduced, with significant improvement in efficiency over field-programmable gate arrays (FPGA) or standard cell ASIC implementations.

Parallel execution of the DWT filter steps using a pipelined implementation is known to provide efficient hardware utilization and fast computation. In fact, a vast majority of the reported hardware implementations for lifting-based DWT rely on pipeline structures [20], [23], [24]. However, these circuits target image and video applications where speed has highest priority and the wavelet basis is chosen to optimize signal representation. A different approach is required to meet the power and area constraints imposed by implantability requirements, the low bandwidth of neural signals, and the type of signals observed. Two promising integer lifting DWT implementations, a pipeline approach and a sequential scheme, have been optimized and compared for the *symmlet4* factorization and data/coefficient quantization described above. Furthermore, the hardware requirements for lifting DWT have been compared to a B-spline implementation to verify the advantage of lifting in the application at hand.

A. Computation Core Design

To begin, notice that the arithmetic operations in the lifting scheme in (7) have a noticeable regularity that permits any arbitrary step to be defined as

$$O_{ij} = X + a_i Y + b_j Z \tag{11}$$

where O, X, Y, and Z take the values of g and f in (7), and a_i and b_j are the quantized filter coefficients given in Table I. The regularity of this repeated operation indicates that an optimized integer DWT implementation would include a hardware unit specifically designed to evaluate (11). By tailoring this circuit to the near-optimal data and coefficient bit width described above, a single computation core (CC) suitable for all lifting filter steps in (7) can be obtained.

Fig. 4 describes a CC block that was custom designed to minimize transistor count and power consumption while supporting up to 10-bit data and 6-bit filter coefficients, both in signed integer formats. The CC employs a simple hardwired shifting operation to remove the x16 scaling factor from the quantized coefficients. It generates a 10-bit output and an overflow error bit, though the lifting scheme should inherently maintain results within 10-bit magnitude. Several multiplier topologies were experimentally compared to define the most efficient option for 6×10-bit operations. A Wallace tree multiplier with modified Booth recoding was implemented along with a custom 3-term adder optimized for power rather than speed. The fixed x16 scaled integer coefficients were modified for Booth recording before being stored in on-chip ROM to eliminate the need for
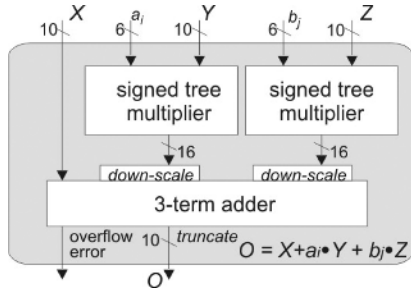
Fig. 4. Customized computation core for integer-lifting wavelet transform using binary scaled filter coefficients.

an on-chip encoder. The resulting circuit very efficiently implements steps 2-4 of (7) and can also compute steps 1 and 5 using a control signal that shuts off the unused multiplier to eliminate unnecessary power consumption.

B. Real-Time Integer DWT Processing Architectures

To identify the most efficient architecture for executing the entire set of lifting equations in real time on a continuous flow of input data samples, let us first re-define the filter equations in (7) with a more hardware-friendly notation. Building on the concept of a fixed three-term computation core described above, the notation in (11) can be used to rewrite (7) at a specific cycle, $i = 1$, as

$$
\begin{aligned}
\text{step1}: \quad & O1^1 = g^1 + a_1 \cdot f^1 \\
\text{step2}: \quad & O2^1 = f^1 + a_2 \cdot O1^2 + b_2 \cdot O1^1 \\
\text{step3}: \quad & O3^1 = O1^1 + a_3 \cdot O2^1 + b_3 \cdot O2^0 \\
\text{step4}: \quad & O4^1 = O2^1 + a_4 \cdot O3^1 + b_4 \cdot O3^0 \\
\text{step5}: \quad & O5^1 = O3^1 + a_5 \cdot O4^2
\end{aligned}
\tag{12}
$$

where g and f are the input data pair of samples, the outputs of steps 1–5 are $O1$-$O5$, coefficients C_1-C_8 have been replaced by a_1-a_5 and b_2-b_4 to indicate the CC input to which they will be applied, and the superscripts represent the computation cycle in which the data value was generated. The 2nd and 3rd terms in step 2 have been swapped to maintain a regular data flow described further below. Steps 2 and 5 require data from future computation cycles. Thus, in order to compute the five filter steps in real time, where all inputs must be available from prior computations, execution must span *three* computation cycles. During cycle $i = 3$ the following five steps can be executed in real time:

$$
\begin{aligned}
1: \quad & O1^3 = h^3 + a_1 \cdot f^3 \\
2: \quad & O2^2 = \boldsymbol{f^2} + a_2 \cdot O1^3 + b_2 \cdot \boldsymbol{O1^2} \\
3: \quad & O3^2 = \boldsymbol{O1^2} + a_3 \cdot O2^2 + b_3 \cdot \boldsymbol{O2^1} \\
4: \quad & O4^2 = O2^2 + a_4 \cdot O3^2 + b_4 \cdot \boldsymbol{O3^1} \\
5: \quad & O5^1 = \boldsymbol{O3^1} + a_5 \cdot O4^2.
\end{aligned}
\tag{13}
$$

Notice that each step in (13) relies only on previously calculated data, provided these steps are performed *sequentially*. Having rearranged the terms in step 2 of (7), the output of each step in (13) becomes the 2nd term input to the subsequent step, which is useful for efficient hardware implementation. Notice also that most of the data values needed are generated within the same

cycle; only the four values in (13) with boldface type (two are repeated twice) are generated in a previous cycle. Thus, if the filter steps are implemented sequentially, only *four* storage/delay registers are required.

Although (13) does allow real time computation of the filter steps in sequence, dependencies within the steps in (13) preclude parallel execution necessary for a pipeline implementation. To make each filter step dependent only on data from prior cycles, execution must span *seven* data samples. During cycle $i = 7$ the following sequence could be computed without any dependency on current or future cycle results:

$$
\begin{aligned}
O1^7 &= h^7 + a_1 \cdot f^7 \\
O2^5 &= f^5 + a_2 \cdot O1^6 + b_2 \cdot O1^5 \\
O3^4 &= O1^4 + a_3 \cdot O2^4 + b_3 \cdot O2^3 \\
O4^3 &= O2^3 + a_4 \cdot O3^3 + b_4 \cdot O3^2 \\
O5^1 &= O3^1 + a_5 \cdot O4^2.
\end{aligned}
\tag{14}
$$

Here, the second term of each computation relies on the output from the preceding step during the previous computation cycle. In a pipeline, these four second-term data inputs could be held in a memory with one-cycle delay. The first and third terms require seven additional data values from prior cycles, one of which is needed twice, resulting in six independent values. One of the values (f^5 in step 2) needs a two-cycle delay, requiring an extra delay register. Thus, a total of 11 storage/delay registers would be required to hold all of the necessary values from prior cycles for a pipeline implementation.

C. Pipeline Design

The integer DWT filter equations in (14) can be implemented simultaneously in a pipeline structure that permits real time, continuous signal processing to take place. Fig. 5(a) illustrates a pipeline structure designed around the customized three-term computation core from Fig. 4. The output of each of the five filter stages is held by a darkly shaded pipeline register, and other registers provide the necessary delays. By clocking all of the registers out of phase from the CC blocks, continuous operation is provided. The computation latency is seven cycles, due to the five pipeline stages and the two delay cycles built into (14). The temporal latency for detail ($O5$) and approximation ($O4$) results is 14 samples because each computation cycle operates on a pair of data samples. The overall pipelined *computational node* consists of five CC blocks, 15 10-bit registers, and an $8 \times 6b$ coefficient ROM. An additional delay phase could be added at the $O4$ output to synchronize the latency of the detail and approximation outputs.

D. Sequential Design

Although the pipeline structure achieves fast integer DWT processing via a large hardware overhead, it is very resource-efficient and thus well suited for low-power, single channel, neural signal processing. However, as discussed below, scaling the pipeline for multiple data channels and/or multiple decomposition levels begins to break down the efficiency of the pipeline structure. An alternative approach is to process each of the filter steps (or pipeline stages) sequentially using a single CC
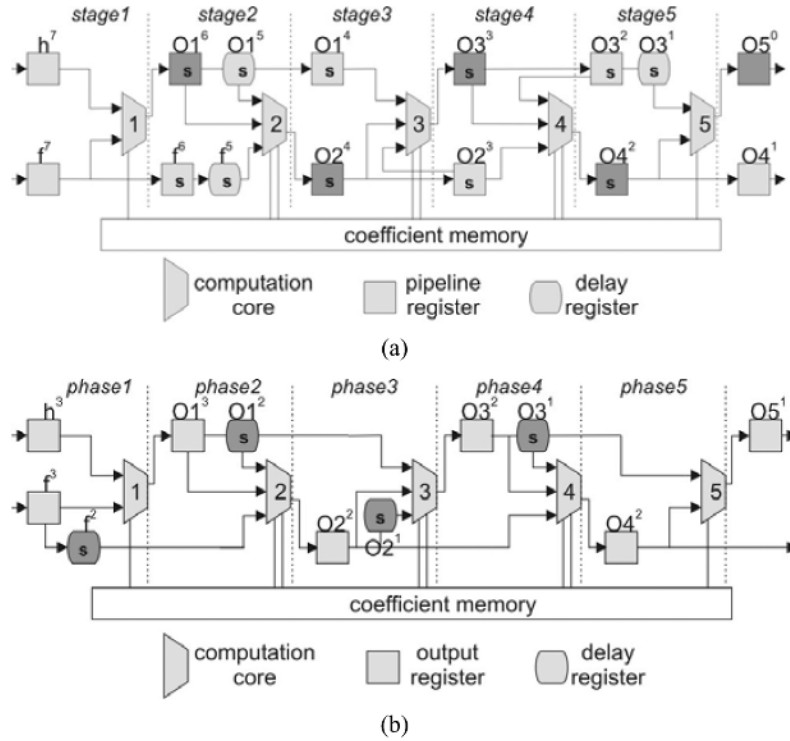
Fig. 5. (a) Pipeline structure for integer-lifting wavelet transform with data notations to match filter equations in (11) at a single point in time. (b) Sequential structure over five operation phases for comparison to the pipeline structure.

block and a fraction of the registers required by the pipeline. This approach takes advantage of the low bandwidth of neural signals that permits the CC to be clocked much faster than the input data sampling frequency (typically in the range of 25–40 kHz).

Sequential processing of the integer DWT filter steps can be achieved using (13), where each stage depends only on data from previous cycles or from same-cycle outputs generated in a preceding step. The simplicity of data dependencies relative to the pipeline structure can be observed from Fig. 5(b), which illustrates the sequential structure in a format comparable to the pipeline. Here, each section of the circuit represents a temporal phase rather than a physical stage. An important observation is that significantly fewer registers are needed because the inputs of subsequent phases rely largely on preceding outputs from the same computation cycle. Therefore, it can be shown that the overall sequential DWT circuit can be efficiently implemented with six 10-bit registers to manage data flow between computation cycles, a single CC block, an 8×6b coefficient ROM, and a simple control block to direct data from memory to the appropriate CC input during each phase of operation. Sequential execution has a computation latency of *two* cycles, and the temporal latency for detail and approximation results is *four* samples.

E. Analysis and Comparison

As stated above, the sequential approach requires only one CC unit and six 10-bit memory registers compared to five CC units and 15 registers for the pipeline circuit. The sequential design does, however, require additional multiplexers and control logic to redirect data and coefficients to CC inputs, which are not necessary in the inherently hardware-efficient pipeline design. This added circuitry will make the critical path of the sequential circuit longer than that of the pipeline structure. Furthermore, to maintain the same throughput, the sequential design must be operated at five times the clock rate of the pipeline. Because data is processed in a real-time streaming mode, neither approach requires a large input data buffer.

Both architectures have been thoroughly analyzed to determine which approach is best suited to the power and area requirements of an implantable neural signal processor. To first validate that both approaches can achieve the application speed requirements, a custom computation core has been implemented in CMOS, and analog simulations show the critical path delay is 6.5 ns in 0.5-μm technology. Thus, approximately 6000 computation cycles could be preformed within a nominal 25-kHz sampling frequency for neural signals. This indicates that speed is not a critical design constraint and that circuit optimization can focus on chip area and power consumption.

Using custom design techniques, the chip area, A, required to implement both approaches will be roughly proportional to the number of transistors in the circuit

$$A = \sum_i T_i \cdot N_i \tag{15}$$

where T is the area per transistor and N is the number of transistors in the ith circuit block. Empirical observations of several custom circuit layouts shows that a single value for T reasonably approximates all of the integer DWT blocks, especially for comparing two similar circuits. Conservative values of 80 μm^2 per transistor for 0.5-μm technology and 5 μm^2 per transistor for 0.13-μm technology have been selected to estimate the required chip real estate.

TABLE IV
CHARACTERISTICS OF SINGLE-LEVEL, SINGLE-CHANNEL INTEGER DWT
HARDWARE FOR PIPELINE AND SEQUENTIAL CONFIGURATIONS AT TWO
TECHNOLOGY NODES

	# transistors	0.5μm CMOS		0.13μm CMOS	
		area [mm²]	power [μW]	area [mm²]	power [μW]
Pipeline	23,336	1.867	15.10	0.117	0.71
Sequential	7,931	0.634	18.55	0.040	0.87

Although absolute power consumption is inherently difficult to estimate, for the purpose of comparing the two design alternatives, dynamic power can be determined as

$$P = k \cdot VDD^2 \cdot f_s \cdot \sum_i N_i \cdot s_i \qquad (16)$$

where VDD is the supply voltage and f_s is the data sampling frequency (nominally 25 kHz). The parameter k accounts for the average output load capacitance, the average number of transistors per output transition, and the average output transitions per clock cycle. This parameter is a function of both fabrication process and circuit topology and has been derived empirically as 3 and 0.75 fF for 0.5-μm and 0.13-μm technology, respectively. The variable s is the clock rate scaling factor relative to f_s for each block i such that the clocking frequency of each circuit block is $s_i \cdot f_s$. For example, in the pipeline configuration, the computation core will be clocked only every other cycle, i.e., $1/2f_s$, so that the first of the pair of samples to be processed can be acquired in the idle cycle. Correspondingly, because the sequential configuration must be clocked at five times the rate of the pipeline, it will have an average clocking rate of $2.5f_s$. In the pipeline approach, all of the blocks are clocked at the same frequency, except the coefficient memory that is static in both designs. In the sequential implementation, one of the multipliers is idle during two of the five stages, so we estimate the sequential CC clock scaling factor to be 2. Similarly, in the sequential controller, most of the circuits are clocked at $2.5f_s$ while others are clocked at $1/2f_s$, so we estimate its clock scaling factor to be 2 as well.

Table IV lists the total number of transistors in each approach along with the area and power estimated from (15) and (16) for both 0.5 μm and 0.13-μm technology. As expected, the pipeline computation unit requires nearly three times the area of the sequential approach and would occupy about 21% of the chip area on a 3×3 mm chip in 0.5 μm technology or 5% of a 1.5×1.5 mm chip in a 0.13-μm process. The power model predicts that the sequential approach will consume only 23% more power than the pipeline. The larger power consumption of the sequential approach can be attributed to its requirement for a more complex controller and the need to move more data around within the single computation core. Overall, these results show a tradeoff between area and power consumption between the two approaches.

F. Lifting Versus B-Spline

As an alternative to lifting, the B-spline method was investigated because it permits a reduction in the number of floating point multiplications at the expense of more additions. However, as demonstrated above, for implantable applications, integer processing is preferred. Table III shows that B-spline saves two multiplications at the cost of 10 additions per cycle compared to lifting. Designs using Verilog synthesized to a custom library have shown that, for a pipeline implementation, B-spline requires significantly less 24-bit floating point hardware, but for integer processing (with 10-bit data and 6-bit coefficients) B-spline saves only 6% compared to lifting [25]. Furthermore, B-spline can not be as efficiently implemented in a sequential structure, where lifting has been shown to require only 53% of the B-spline hardware resources for integer DWT. While B-spline implementations do have slightly less delay, speed is not a design constraint. Relative memory requirements are a more important issue in multichannel implementations as we show next.

IV. MULTILEVEL AND MULTICHANNEL INTEGER DWT IMPLEMENTATION

A. Hardware Design

In implantable neuroprosthetic applications where a typical microelectrode array has many electrodes integrated on a single device, there is a strong need to support integer DWT computations with multiple levels of decomposition for multiple signal channels pseudo-simultaneously (i.e., within one sampling period). The lifting scheme and the two integer DWT implementations described above have been chosen because of their ability to scale to an arbitrary number of channels and levels. Considering that both of the single channel, single level, integer DWT approaches discussed above require a substantial portion of a small chip, it is unreasonable to pursue a hardware intensive solution that utilizes a "copy" of the circuit for each channel and level. This would dramatically increase circuit area beyond limitations for implantable systems. Given the available computation bandwidth of the CC block, the more appropriate solution is to scale the clocking frequency as needed to sequentially compute filter equations for multiple channels and/or levels. Although clock scaling will still cause power to increase with channel and level, the circuit area required will be minimized and the power density can be held within the acceptable application limits.

Both the pipeline and sequential architectures can be scaled to multiple channels and/or levels by reusing the computational node hardware and increasing the clocking frequency to complete all computations within the input sample period. In both approaches, registers within the computational node hold data necessary for the next cycle's calculation. To sequentially reuse the computational node, some register values for a specific channel/level must be saved so they will be available when that channel/level is next processed in a future cycle. Fig. 6 shows the multichannel, multilevel, implementations of the pipeline and sequential configurations.

1) Multichannel Considerations: In scaling the system to multiple data channels, the computation clock rate is scaled by the number of channels and a new memory block is added to save critical register data for each channel. For the pipeline, the 11 registers must be stored, while for the sequential circuit only
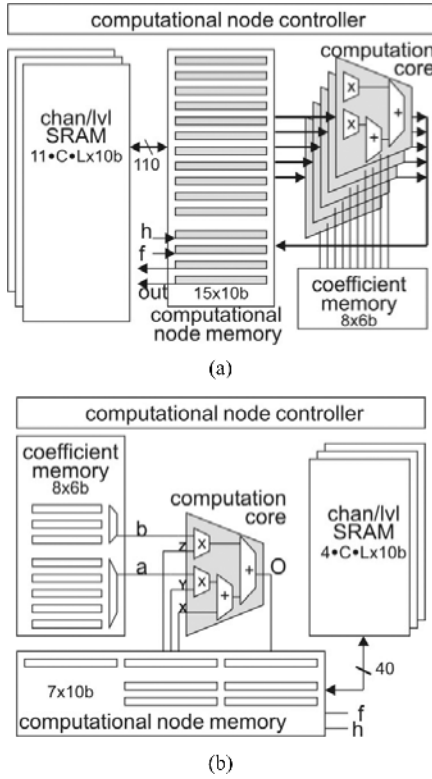
(a)

(b)

Fig. 6. Multilevel, multichannel implementations of (a) pipeline structure and (b) sequential structure.
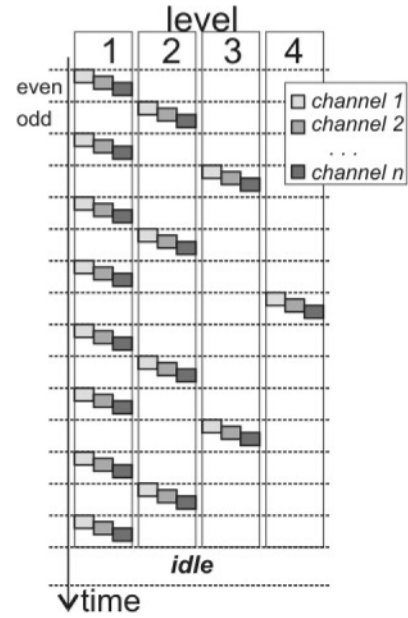

Fig. 7. Sequential processing scheme for multilevel, multichannel computation. At the top of this sequence, one DWT result is available at each decomposition level. With the four levels shown, one idle computation cycle will occur every 16 cycles.

four registers need to be saved. These registers are marked with an "s" in Fig. 4. An on-chip SRAM can be interfaced to the computational node to store register values, and the size of the SRAM will grow linearly with the number of channels. Note for comparison that a sequential B-spline implementation requires eight register values to be stored.

2) Multilevel Considerations: When expanding the DWT to multiple levels, notice that each level of dyadic DWT decomposition introduces only half the number of computations as the previous level. More explicitly, the number of results, R, per number of samples, n, for an arbitrary level j can be expressed as

$$\frac{R}{n} = \sum_{j=1}^{\infty} \left(\frac{1}{2}\right)^j \qquad (17)$$

which is always less than twice the number of samples. Consider also that, to process multichannel input pairs, before each computation cycle the system must implement one idle cycle, wherein the first input of the pair is stored for each channel. Thus, if the level-one computations are executed in, say, the even cycles, the higher level computations can be executed in the odd cycles [26] while input samples (one of the pair) are being stored for the next level-one computation. This is illustrated in Fig. 7.

If we define the *usage rate*, $U(L)$, as the average number of cycles for a single computation to occur, then for the first decomposition level the usage rate is one half, i.e., $U(1) = 0.5$, and the computational hardware is idle during the other half of the cycles. Moreover, $U(L)$ approaches 1.0 as the number of levels increase, i.e.,

$$U(L) = 1 - 2^{-L}.$$

As the number of levels increases, the usage rate will increase toward maximum utilization without increasing computation frequency. For each level of decomposition beyond the first, one memory block per channel is required to store values held in the computational node registers. The registers to be stored are the same as those described in the multichannel case above.

B. Area and Power Modeling

For multiple channels/levels, the need to copy the entire set of pipeline registers to memory effectively negates one of the primary advantages of the pipeline over the sequential approach. On the other hand, the sequential processing circuit is inherently designed to swap new data in/out each clock cycle. To quantitatively compare these two approaches, circuit models have been developed to describe the power and area for each option as a function of the number of channels and the number of decomposition levels. The following models assume the hardware (including control logic) has been scaled to manage multiple channels and levels, though they are still valid for single channel, single level implementations.

A general expression for calculating the area of both the pipeline and the sequential approaches as a function of channels and levels is:

$$A = A_t \sum_i \left(No_i + (C - 1) \cdot L \cdot Nc_i + (L - 1) \cdot C \cdot Nl_i\right) \qquad (19)$$

where A_t is the technology-dependent, empirically-derived average area per transistor, No_i is the number of transistors that remain constant with level and channel in the ith circuit block, Nc and Nl are the number of transistors that scale with channel and level, respectively, C is the number of channels, and L is the number decomposition levels. Although this equation only roughly estimates routing area, it is very useful for comparative

analysis since both approaches consist of similar arithmetic and memory blocks.

Using (16), a general expression for power consumption as a function of channels and levels, which is valid for both approaches being considered, is given by

$$P = 2 \cdot k \cdot VDD^2 \cdot fs \cdot \sum_i \Bigl(No_i + (C-1) \cdot L \cdot Nc_i$$

$$+ (L-1) \cdot C \cdot Nl_i (Cf_i \cdot Lu_i \cdot s_i) \Bigr) \quad (20)$$

where Cf_i is the channel clock frequency scaling factors, Lu_i is a level usage factor, and all other variables are as previously defined. Recall that the clock scaling factor s_i was chosen to accommodate the fact that, in single level designs, every other cycle was idle while the data pair was being collected. To maintain a consistent definition of variables in multilevel implementations, which utilize the idle cycles to process all higher levels, the factor of 2 is introduced at the beginning of (20).

Both the pipeline and sequential architectures have been developed to define the model parameters given in Table V, which are valid for $L > 1$ and $C > 1$. The computational node circuitry, including control logic, has been scaled up to manage an arbitrary number of levels and channels, with negligible per channel/level increase in complexity. Thus, only data memory increases with the number of channels. Clocking frequency of the computational node circuits must scale with channel, while each memory block is only accessed once per cycle regardless of the number of channels. The controller frequency scales linearly with channel but is assumed to remain constant with level. For all other circuit blocks, the usage rate $U(L)$ accounts for inactive computation cycles.

V. RESULTS AND DISCUSSION

A. Signal Integrity

We have assessed the effects of data and filter coefficient approximations on the quality of the signals obtained after reconstruction. We quantified the performance in terms of the complexity of hardware required to implement (7) and illustrated the results in Fig. 8. The wavelet filter coefficients were quantized to different resolutions ranging from 4 to 12 bits, with the 6-bit values given in Table I. The data was also quantized in the same range. The *effective* signal-to-noise ratio (eSNR), defined as the log ratio in dB of the peak spike power to the background noise power is illustrated in Fig. 8(a) versus multiplier complexity in equivalent bit addition/sample for an average input SNR of 6 dB. These results demonstrate that, with sufficient precision, the use of integer computations does not result in significant signal degradation as quantified by the observed output SNR. Specifically, with quantization of filter coefficients to 6 bits and data to 10 bits, the output SNR is within 1% of its average input value.

In Fig. 8(b), the spectrum of the residual quantization and round-off noise is also illustrated to demonstrate the loss in the signal power-spectral density in different cases. In the case of 4-bit quantization of the filter coefficients, the residual noise frequency content is closest to that of the original signal in the low frequency range (subband 0–1 kHz), indicating that some signal loss may have occurred in that band. On the other hand,
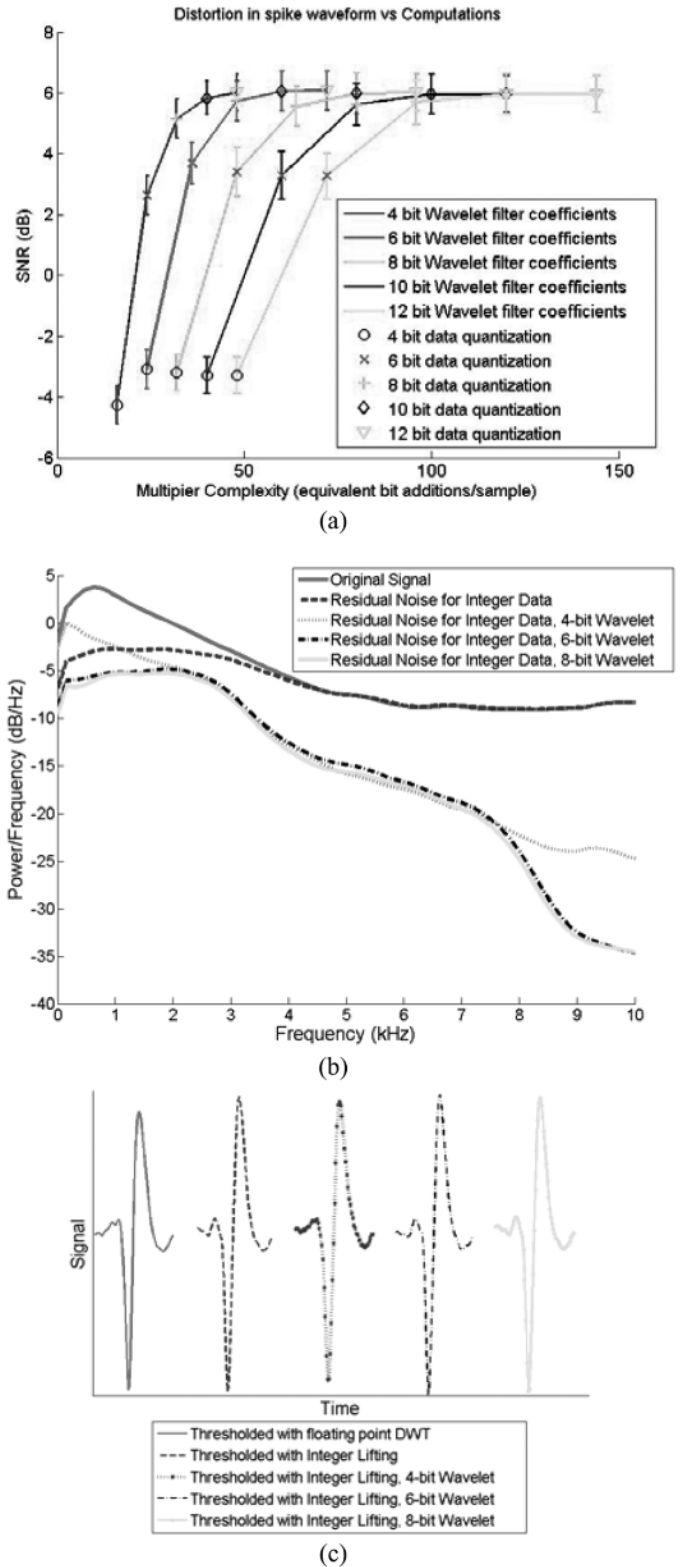

Fig. 8. (a) Effect of round off and quantization errors on the signal fidelity as a function of multiplier complexity. (b) Power-spectral density of the original data and the residual noise for integer approximated data and quantized wavelet filter coefficients for various bit widths. (c) Example spike waveforms obtained in each case.

filter quantization of 6 bit or higher results in residual noise that consists of high frequency components above 8 kHz, which is outside the frequency range of neural spike trains and local field

TABLE V
MODEL PARAMETERS FOR AREA AND POWER CALCULATIONS

	No	Nc	Nl	Cf	Lu	s
Pipeline						
Controller	524	0	0	C	1	0.5
Comp. Block	21848	0	0	C	U(L)	0.5
Coeff. Mem.	964	0	0	C	U(L)	0
Data Mem.	0	1562	1562	1	U(L)	0.5
Sequential						
Controller	744	0	0	C	1	2*
Comp. Block	6125	0	0	C	U(L)	2*
Coeff. Mem.	1062	0	0	C	U(L)	0
Data Mem.	0	572	572	1	U(L)	0.5

* accounts for average usage at 2.5 clock scale

Fig. 10. Power-area product versus level and channel for pipeline and sequential approaches.
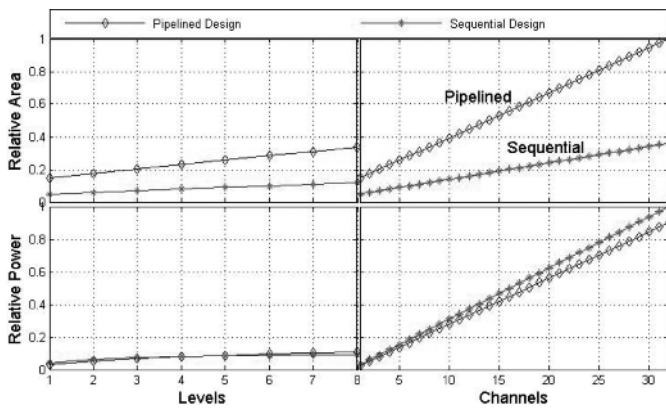

Fig. 9. Comparison of multichannel/multilevel pipeline and sequential integer DWT approaches: relative chip area and relative power consumption versus number of levels and channels.

potentials (LFPs) [27]. A representative example of spike waveforms in each case is illustrated in Fig. 8(c) to demonstrate the very negligible effect of this process on the quality of the average spike waveform. Taking these results all together, it is clear that the choice of 6/10-bit coefficient/data quantization offers the best compromise among multiplier complexity and signal fidelity as concluded earlier.

We should emphasize that perfect reconstruction of signals off chip may not be always needed. Typically, neural signals contain the activity of multiple neurons that need to be *sorted out*, and this information remains in the compressed data at the output of the DWT block. We have shown elsewhere that sorting the multi source neuronal signals can be performed directly on the wavelet transformed data [10], [28], and this topic is outside the scope of this paper.

B. Multichannel/Level Implementations

Using (19) and Table V, the relative area for pipeline and sequential architectures as a function of levels and channels is shown in Fig. 9. These results demonstrate that the pipeline requires significantly more chip area than the sequential approach and its area needs grow faster with larger number of channels and levels. This is due primarily to the relatively large number of registers that must be stored per channel or level (11 for pipeline compared to 4 for sequential). Fig. 9 also shows the relative
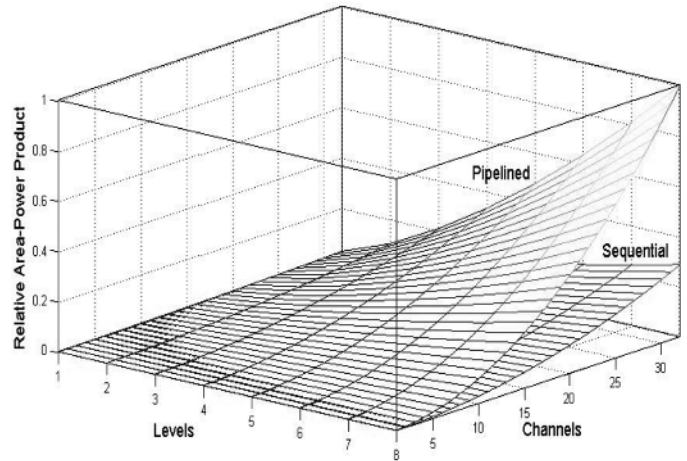
power consumption for the two approaches based on (20). The linear increase in power per channel is slightly higher with the sequential design than the pipeline. Although there is a sharp jump in power from $L = 1$ to $L = 2$, further increases in levels require less and less additional power as the usage rate approaches one. The most important observation from Fig. 9 is that the power consumption of the two implementations is almost similar but the sequential design requires significantly less chip area.

Due to size and power constraints in implantable systems, an important figure of merit is the relative area-power product, which is plotted in Fig. 10 versus both level and channel. Fig. 10 illustrates that the sequential approach is increasingly preferable as the number of channels or the number of decomposition levels increases. The only significant benefits of the pipeline within the enforced design constraints are that it can be clocked at a higher rate and that it takes fewer clock cycles to complete a computation. Both of these factors result in the pipeline having a *higher threshold* on the maximum number of channels that can be simultaneously processed. However, based on the parameters defined above, the sequential execution architecture has an estimated maximum of around 500 data channels (at $L = 1$). Given the chip area limitations, the area-efficient sequential approach is best suited for this application. In an example implementation with 32 channels and 4 levels of decomposition, the models predict that the sequential approach will require 0.692 mm^2 and 50.1 μW in 0.13-μm CMOS, indicating the feasibility of performing front-end signal processing within the constraints of an implanted device.

Another interesting result of this study is the comparison of the area required by the computational node circuitry versus the area required by the memory that holds register values required for multichannel/multilevel operation. Fig. 11 illustrates this result for both sequential and pipeline configurations as a function of channels at $L = 4$. Notice with the pipeline that memory dominates the area when the number of channels is greater than four. For the sequential design, memory dominates when the number of channels is greater than ten. With 10-bit data resolution, at $L = 4$ and $C = 32$, the pipeline requires over 14 000 bits
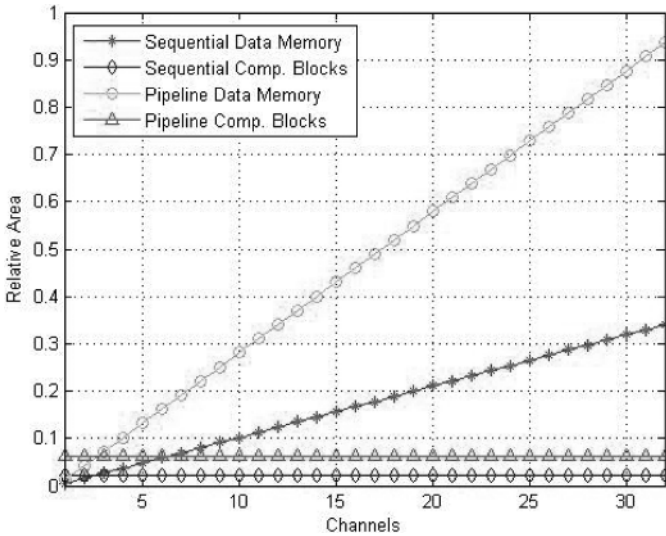
Fig. 11. Relative area versus channels of data memory compared to all other blocks for sequential and pipeline designs, at $L = 4$.

of SRAM, while the sequential circuit requires only about 5000 bits. Reducing memory requirements becomes increasingly important in multichannel applications, again highlighting the advantage of the sequential approach.

C. Lifting versus B-Spline

As illustrated in Fig. 11, the memory required to store intermediate calculation values will dominate circuit area in multichannel implementations. Careful analysis of an optimized sequential B-spline implementation [25] has shown that eight memory registers are required per channel/level, compared to four for sequential lifting and 11 for pipeline lifting. Based on this information and the comparisons above, B-spline has a slight advantage over pipeline lifting but incurs a significant penalty relative to sequential lifting in terms of area. Furthermore, the sequential lifting implementation requires only about 25% of the dynamic power of sequential B-spline, primarily because B-spline takes 18 cycles to execute sequentially compared to 5 cycles for lifting [25]. The advantage of sequential lifting becomes even more profound when static power is considered, especially in deep submicron technologies. Fig. 12 provides an additional comparison, where the number of required gates, synthesized from Verilog descriptions of lifting and B-spline circuits, are plotted. These results illustrate that lifting is increasingly preferable over B-spline as the number of channels and levels increase.

D. Multiplication-Free Lifting

The CC unit proposed in this paper uses one multiplier so that the calculations required per sample are 8 multiplications and 8 additions that can be completed in 5 cycles as listed in Table III. It is noteworthy that a general purpose lifting approach based on only shifts and additions was proposed in [3]. For the sake of completeness, we compared the demands of a CC unit with a multiplier (proposed in this paper) to a CC unit without a multiplier, i.e., composed of only a shifter and an adder. The later

approach resulted in 12 shift operations and 21 add operations, and required 21 cycles per sample. This is because the equations required to compute multiplication-free lifting DWT did not show any regular structure such as the ones in (7). Therefore, substituting another adder and shifter in the data path did not help in reducing the number of cycles required to complete the computation. With respect to area demands, we found that for one sample pair, a CC unit without a multiplier requires 52% less area compared to a CC with multiplier. This obviously translates into large savings in chip area. However, these savings were not substantial when the system is scaled up. For example, a 32-channel/4-level DWT system using a CC with multiplier would occupy 6.5% of the total chip area as opposed to 3.3% using a CC without multiplier. So the overall savings in chip area are only 3.2%. In contrast, the CC without multiplier requires 13.3% more power than a CC with multiplier for this specification. We therefore concluded that the reduction in area using a shift and add strategy in the lifting approach is overshadowed by the increase in power dissipation when multichannel/multilevel decomposition is sought.

VI. CONCLUSION

VLSI architectures to compute a 1-D DWT for real-time multichannel streaming data under stringent area and power constraints have been developed. The implementations are based on the lifting-scheme for wavelet computation and integer fixed-point precision arithmetic, which minimize computational load and memory requirements. A computational node has been custom designed for the quantized integer lifting DWT and characterized to estimate the maximum achievable computation frequency. Negligible degradation in the signal fidelity as a result of these approximations has been demonstrated.

Detailed comparison between the lifting and the B-spline schemes was presented. It was shown that the lifting approach is more suited when floating point operations are eliminated, thereby superseding the gain achieved by the B-spline approach where adders replace multipliers. Two power and size efficient hardware alternatives for computing the single-level, single-channel wavelet transform have been described and analyzed. The memory management efficiency of the pipeline design results in slightly less power dissipation, while the sequential execution design requires significantly less chip area. Design considerations for scaling these architectures to multichannel and multilevel processing have been discussed. Area and power consumption models with detailed transistor count and switching frequency parameters have been described and used to compare the performance of the two design alternatives in multichannel and multilevel implementations. The results show many interesting characteristics of each design when it scales to an arbitrary number of levels and channels. When the number of channels is two or more, the sequential execution architecture was shown to be more efficient than the pipeline approach in terms of both power and chip area. Furthermore, results indicate that, using this architecture, multilevel processing of many channels simultaneously is
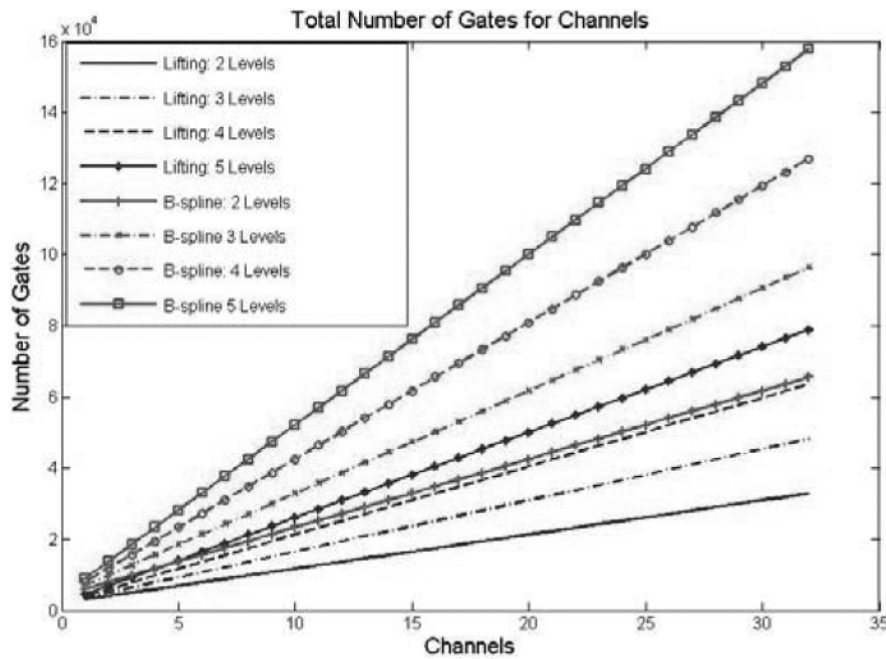
Fig. 12.   Total number of gates as a function of the number of channels and the number of levels for the lifting and B-spline implementation.

feasible within the constraints of a high-density intracortical implant. This work demonstrates that on-chip real-time wavelet computation is feasible prior to data transmission, permitting large savings in bandwidth requirements and communication costs. This can substantially improve the overall performance of next generation implantable neuroprosthetic devices and brain-machine interfaces.

ACKNOWLEDGMENT

REFERENCES

[1] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 6, pp. 191–202, Jun. 1993.

[2] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: A survey," *J. VLSI Signal Process.*, vol. 14, pp. 171–192, 1996.

[3] H. Olkkonen, J. T. Olkkonen, and P. Pesola, "Effcient lifting wavelet transform for micorprocessor and VLSI applications," *IEEE Signal Process. Lett.*, vol. 12, pp. 120–122, 2005.

[4] P. K. Campbell, K. E. Jones, R. J. Huber, K. W. Horch, and R. A. Normann, "A silicon-based, three-dimensional neural interface: Manufacturing processes for an intracortical electrode array," *IEEE Trans. Biomed. Eng.*, vol. 38, no. 8, pp. 758–768, Aug. 1991.

[5] K. D. Wise, D. J. Anderson, J. F. Hetke, D. R. Kipke, and K. Najafi, "Wireless implantable microsystems: High-density electronic interfaces to the nervous system," *Proc. IEEE*, vol. 92, no. 1, pp. 76–97, Jan. 2004.

[6] D. M. Taylor, S. I. Tillery, and A. B. Schwartz, "Direct control of 3-D neuroprosthetic devices," *Science*, vol. 296, pp. 1829–1832, 2002.

[7] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. L. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, pp. 361–365, 2000.

[8] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Instant neural control of a movement signal," *Nature*, vol. 416, pp. 141–142, 2002.

[9] 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.

[10] K. G. Oweiss, "Multiresolution analysis of multichannel neural recordings in the context of signal detection, estimation, classification and noise suppression," Ph.D.dissertation, Univ. Michigan, Ann Arbor, 2002.

[11] K. G. Oweiss, D. J. Anderson, and M. M. Papaefthymiou, "Optimizing signal coding in neural interface system-on-a-Chip modules," in *Proc. 25th IEEE Int. Conf. Eng. Med. Biol*, Sep. 2003, pp. 2016–2019.

[12] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 245–267, 1998.

[13] K. A. Kotteri, S. Barua, A. E. Bell, and J. E. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: Convolution versus lifting," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 5, pp. 256–260, May 2005.

[14] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1575–1586, Apr. 2005.

[15] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping structure: An efficient VLSI architecture for lifitng based discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.

[16] M. Unser and T. Blu, "Wavelet theory demystified," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 470–483, Feb. 2003.

[17] 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.*, vol. 40, pp. 343–353, 2005.

[18] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed.   New York: Academic, 1999.

[19] D. Donoho, "Denoising by soft thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 613–627, May 1995.

[20] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 966–977, Apr. 2002.

[21] Y. Suhail and K. G. Oweiss, "A reduced complexity integer lifting wavelet based module for real-time processing in implantable neural interface devices," in *Proc. 26th IEEE Int. Conf. Eng. Med. Biol.*, Sep. 2004, pp. 4552–4555.

[22] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 332–369, 1998.

[23] B. F. Wu and C. F. Lin, "A rescheduling and fast pipeline VLSI architecture for lifting-based discrete wavelet transforms," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2003, vol. 2, pp. 732–735.

[24] 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.

[25] 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.

[26] P. Y. Chen, "VLSI implementation for one-dimensional multilevel lifting-based wavelet transform," *IEEE Trans. Comput.*, vol. 53, no. 4, pp. 386–398, Apr. 2004.

[27] F. Rieke, D. Warland, R. R. van Steveninck, and W. Bialek, *Spikes: Exploring the neural code*. Cambridge, MA: MIT press, 1997.

[28] K. Oweiss, "Compressed sensing of large-scale ensemble neural activity with resource-constrained cortical implants," *Soc. Neurosci. Abstr.*, vol. 13.11, Oct. 2006.

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. He received the Michigan State University Teacher-Scholar Award in 2006.

Karim G. Oweiss (S'95–M'02) received the B.S. degree and M.S. degree with honors 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 was a Post-Doctoral Researcher in the Biomedical Engineering Department, University of Michigan, in the summer of 2002. In August 2002, he joined the Department of Electrical and Computer Engineering and the Neuroscience program, Michigan State University, East Lansing, 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 in the nervous system, neural integration and coordination in sensorimotor systems, and computational neuroscience.

Prof. Oweiss is a member of the 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.

Yasir Suhail received the B.Tech. degree from the Indian Institute of Technology, Delhi, India, and the M.S. degree from Michigan State University, East Lansing, both in electrical engineering. He is working toward the Ph.D. degree in the Department of Biomedical Engineering at the Johns Hopkins University, Baltimore, MD.

His research interests include applications of signal processing, statistics, and machine learning techniques to biomedical problems.

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

His research interests include signal processing, multimedia communications, VLSI and systems-on-chip design

Mr. Kamboh has held various academic scholarships throughout his academic career.

Andrew Mason (S'90–M'99–SM'06) received the B.S. degree in physics with highest distinction from Western Kentucky University, Bowling Green, in 1991, the B.S.E.E. degree with honors from the Georgia Institute of Technology, Atlanta, Georgia, 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, Lexington. In 2001, he joined the Department of Electrical and Computer Engineering at Michigan State University, East Lansing, where he is currently an Assistant Professor. His research addresses many areas

Kyle E. Thomson was born in Downer's Grove, IL, in 1980. He received the B.S. degree in computer and electrical engineering and the Master's Degree in electrical engineering (focusing on neural signal processing) from Michigan State University, East Lansing, in 2004 and 2006, respectively.

He is currently employed at Ripple, LLC, a startup based in Salt Lake City, UT, focused on neurophysiology instrumentation and neuroprosthetic systems. The company is focused on providing next generation instrumentation for both research and clinical applications. He has held various academic scholarships throughout his academic career. His research interests include signal processing, multimedia communications, VLSI and system-on-chip design.