

A Scalable Architecture for Streaming Neural Information from Implantable Multichannel Neuroprosthetic Devices

Kyle E. Thomson
ECE Department
Michigan State University
East Lansing, USA

Yasir Suhail
ECE Department
Michigan State University
East Lansing, USA

Karim G. Oweiss
ECE Department
Michigan State University
East Lansing, MI, USA

Abstract— Two hardware architectures for implementing lifting-based discrete wavelet transform (DWT) suitable for implantable, real-time operation of high-density sensor array neuroprosthetic devices. A core computational node (CN) is designed for use in both architectures to yield maximum processor usage. The first uses multiple pipelined replicas of the CN, requiring fewer clock cycles. The second architecture reuses a single CN, thus requires less chip area but longer time delay. By utilizing the difference between the data sampling rate and available computation bandwidth, the novelty of both designs lies in the scalability to an arbitrary number of channels by interleaving the DWT computation without affecting the real-time operability. Performance comparison and overall considerations of both designs are presented in details.

I. INTRODUCTION

Rapid advances in microfabrication technology enabled the integration of high-density microelectrode arrays on a single device for implantable neuroprosthetic applications [1]. Consequently, the need to transmit recorded neural data from an implantable neuroprosthetic device in real-time becomes increasingly demanding. Current signal processing and communication technology is not capable of transmitting the data within the available bandwidth [3]. Because of the sparse nature of neural signals on the time base, an alternative to existing techniques for raw data transmission is to extract the useful information prior to extra-cutaneous transmission. However, this strategy imposes severe limitations on the computational capabilities of the associated signal processing due to chip size and power constraints for implantability purposes.

Recently, new methods for processing neural data have shown that bandwidth restrictions can be overcome by using a discrete wavelet transform (DWT) representation due to its high compression capabilities [2], while preserving the temporal information of the neural signals. The latter is thought to carry all the information needed to understand the

neural coding mechanism in the nervous system. Equally important, hardware implementation of the DWT using the *lifting* approach permits significant reduction in the internal memory requirements [2][4]. Because external memory consumes the largest amount of chip area and power, pipelining is a desirable feature to reduce intermediate data storage [5]. Accordingly, an appropriate compromise among power, required bandwidth, computational load, memory, and delay time is needed to yield the most reliable design with highest signal fidelity. These parameters are linked in a very complex way. Nevertheless, some considerations can relax some of the parameter constraints to ease the design approach:

- A neural prosthesis system can tolerate delays of *several* milliseconds. Therefore, the DWT can be computed for multiple data channels and multiple decomposition levels simultaneously.
- 32 data channels from a closely-spaced device are a manageable number of interconnects and signal conditioning components in the volume over the tissue being considered.
- A 14mm burr hole used by neurosurgeons is sufficient for a package of several hundred thousand transistors.
- 25 kHz samples/sec quantized to 8 bits is a usual and logical sampling scheme if scaling is controlled [3].

The work we present herein constitutes our application-driven approach to design efficient architecture for DWT computation in implantable neuroprosthetic device applications taking into account the above considerations.

II. LIFTING-BASED WAVELET TRANSFORM

The traditional Mallat's algorithm for evaluating the wavelet transform of a given signal involves recursively convolving the signal through two decomposition filters l and b , and decimating the result to obtain the approximation and detail coefficients at every decomposition level [2]. The

lifting method, on the other hand, is based on factorizing the wavelet filters l and b into N lifting steps to obtain filters S_n and T_n as

$$P(z) = \begin{bmatrix} \frac{l(\sqrt{z})+l(-\sqrt{z})}{2} & \frac{b(\sqrt{z})+b(-\sqrt{z})}{2} \\ \frac{l(\sqrt{z})-l(-\sqrt{z})}{2} & \frac{b(\sqrt{z})-b(-\sqrt{z})}{2} \end{bmatrix} \quad (1)$$

$$= \begin{bmatrix} K & 0 \\ 0 & K^{-1} \end{bmatrix} \begin{bmatrix} 1 & S_N(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & S_1(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Mathematically, this is implemented by splitting the data into *even* and *odd* samples and applying the S_n and T_n filters consecutively as shown in fig 1. The data at each step, after applying the filters are labeled as f_0, f_1, \dots, f_N , and h_0, h_1, \dots, h_N , respectively. The last step is a multiplication by a scaling factor K . The outcome at an arbitrary decomposition level j is obtained as the approximation a_j and detail d_j . The former is fed back for the next level of DWT decomposition, whereas the latter is stored.

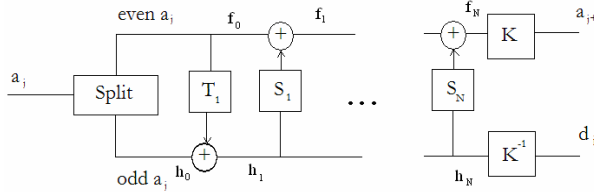


Fig 1. Lifting scheme for DWT computation

It has been shown that fixed-point integer wavelet transform computation using the *symmlet4* wavelet with filter coefficients truncated to 4 bits gives performance comparable to the floating-point case [6]. When using integer wavelet transform, the last scaling steps, which require multiplication by K and $1/K$, can be eliminated. For a factorization of the *symmlet4* wavelet, the filtering steps corresponding to fig. 1 can be written as

$$\begin{aligned} \text{step1: } h_1(t) &= h_0(t) + C_1 \times f_0(t) \\ \text{step2: } f_1(t) &= f_0(t) + C_2 \times h_1(t) + C_3 \times h_1(t+1) \\ \text{step3: } h_2(t) &= h_1(t) + C_4 \times f_1(t) + C_5 \times f_1(t-1) \\ \text{step4: } a(t) &= f_1(t) + C_6 \times h_2(t) + C_7 \times h_2(t-1) \\ \text{step5: } d(t) &= h_2(t) + C_8 \times a(t+1) \end{aligned} \quad (2)$$

where f_0 and h_0 are the even and odd data samples; f_1 , h_1 , and h_2 are intermediate values which are discarded after being used, due to the in-place nature of the lifting algorithm; $f_2 = a(t)$ is the resulting approximation coefficient; $h_3 = d(t)$ is the resulting detail; C_1 through C_8 are the coefficients of the prediction and update filters given in Table I along with their scaled to 4 bit integer values.

TABLE I. SYMMLET-4 DWT FILTER COEFFICIENTS

Coef.	Value	Scaled	Coef.	Value	Scaled
C_1	0.39114	3	C_5	0.16203	2
C_2	-0.12439	-1	C_6	0.43128	3
C_3	-0.33924	-5	C_7	0.14598	1
C_4	-1.41951	-11	C_8	-1.04925	8

III. HARDWARE DESIGN

Our goal is to design an architecture to perform the 1-D DWT for multichannel data under the constraints of implantability and real-time operation. Unlike the interleaved computation proposed in [5], which assumes a whole data frame to be available, an interleaved computation of each level for a given number of channels can efficiently take place by making use of the difference between the biologically relevant sampling rate and the available clock speed. For that purpose, the DWT computation has to actively take place while an entire data frame is being acquired across multiple channels. The arithmetic operations in (2) have a noticeable regularity that permits any arbitrary step to be defined as

$$W_{ij} = X + C_i Y + C_j Z \quad (3)$$

where the variables W , X , Y and Z take the values of f_p and h_q , C_i and C_j are the constant coefficients given in Table 1. We will refer to the hardware block that computes equation (3) as a computational node (CN).

B. Simultaneous Execution

Each of the five steps in (2) can be separated such that the associated constants can be directly *hardwired* into its own CN. Each CN belongs to a separate pipeline stage that can be executed simultaneously with all other pipeline stages. Since the steps are interdependent, they must be operating on multiple time instants. Assuming that an arbitrary sample pair at time t_n is available, then the sample pair being computed at the G^{th} pipeline stage will correspond to t_s where

$$t_s = t_n - (G - 1) \times 2 \quad (4)$$

We'll denote by E_0 and O_0 the values held by the input registers for the *even* and *odd* incoming data sample pair, respectively. Also, we'll denote by $\{P_0, P_1, P_2\}$, $\{Q_0, Q_1, Q_2\}$, $\{R_0, R_1, R_2\}$, $\{A_0, A_1\}$, and $\{O_1, O_2\}$ the values at registers for internal use by the DWT computation block. Thus, A_2 and D_0 are *output* registers holding the approximation and detail values at any given level. We will also define a *delay node* (DN) as a pipeline stage in which no computation takes place. Due to the noncausal nature of the DWT, delay nodes are necessary to deal with data interdependencies. After a single simultaneous execution of

(2), the newly computed values are stored in all zero indexed registers. Thus, equation (2) can be redefined as

$$\begin{aligned}
& \text{CN 1 : } P_0 = E_0 + C_1 \times O_0 \\
& \text{DN 2} \\
& \text{CN 3 : } Q_0 = O_2 + C_2 \times P_2 + C_3 \times P_1 \\
& \text{CN 4 : } R_0 = P_2 + C_4 \times Q_1 + C_5 \times Q_2 \\
& \text{CN 5 : } A_0 = Q_2 + C_6 \times R_1 + C_7 \times R_2 \\
& \text{DN 6} \\
& \text{CN 7 : } D_0 = R_2 + C_8 \times A_1
\end{aligned} \tag{5}$$

To enable simultaneous execution of all steps, the zero indexed registers where the current data is being written are not read during the execution. In order to fill the higher indexed registers, a two-phase update step is necessary. The update step operates after a single execution of all steps as illustrated in fig.2. From (5), the pipeline length is seven, and it operates on a pair of data samples, so the resulting detail and approximation temporal latency is 14 samples.

Phase 1	Phase 2
$O_2 \leftarrow O_1$	$O_1 \leftarrow O_0$
$P_2 \leftarrow P_1$	$P_1 \leftarrow P_0$
$R_2 \leftarrow R_1$	$R_1 \leftarrow R_0$
$Q_2 \leftarrow Q_1$	$Q_1 \leftarrow Q_0$
$A_2 \leftarrow A_1$	$A_1 \leftarrow A_0$

Fig 2. Update step for the simultaneous architecture

B. Sequential Execution

The architecture described above allows much higher throughput but is not space efficient. Another approach to minimize the hardware size can be exploited if the steps are executed sequentially using a single CN to reduce the number of registers. A single usage of the CN, denoted below by a *computational step* (CS), allows the architecture to take the following form

$$\begin{aligned}
& \text{CS1 : } P_0 = E_0 + C_1 \times O_0 \\
& \text{DN1} \\
& \text{CS2 : } Q_0 = O_1 + C_2 \times P_1 + C_3 \times P_0 \\
& \text{CS3 : } R_0 = P_1 + C_4 \times Q_0 + C_5 \times Q_1 \\
& \text{CS4 : } A_0 = Q_0 + C_6 \times R_0 + C_7 \times R_1 \\
& \text{DN2} \\
& \text{CS5 : } D_0 = D_1 + C_8 \times A_0
\end{aligned} \tag{6}$$

Because of the presence of the two DNs, the pipeline length is two. The overall result is a temporal latency of four data samples, as opposed to fourteen in the first design. We note that the indexing on each of the intermediate registers has decreased, which implies that fewer values need to be stored external to the DWT module, as well as decreasing the

number of registers internal to the DWT module. However, this increases the critical path of a single execution. The second phase of the update step in fig 2 is the only one needed, and that results in a shorter pipeline.

C. Multilevel and Multichannel Management

It is desirable to use a single DWT module for all level and channel computations. The current level-channel data is moved to memory external to the CN, and a new level-channel replaces it. Since the CNs new computed values are stored in the zero indexed registers, then the only values that must be stored externally are the ‘one’ and ‘two’ indexed registers.

To minimize external memory access, we use a “bus” style register architecture using multitudes of registers to avoid the latency of memory reading and writing. Unlike the architecture in [5], we use an intelligent memory addressing where each address is subdivided in three parts: The most significant bits are used for channel addressing, the middle bits indicate the decomposition level, and the least significant bits indicate the particular register needed for computation.

IV. PERFORMANCE

For design constraints imposed by the required real-time operation, the total number of available clock cycles N_c between two consecutive data samples is given by

$$N_c = \frac{f_c}{f_s/2} \tag{7}$$

where f_c is the chip clock rate, f_s is the data sampling rate. To evaluate the performance of both designs, the total available latency must be used to determine the maximum number of channels C_{pt} that can be simultaneously processed based on the time constraint and is given by

$$C_{pt} = \frac{N_c}{(W_e + S_o) \times N_d} \tag{8}$$

where N_d is the minimum number of decomposition levels that must be executed between sample pairs to prevent data overlap. This value is independent of the actual number of decomposition levels being processed. W_e is the number of clock cycles needed to execute a single decomposition level, and S_o is the switching overhead required to change decomposition level and channel. For a given fixed chip area, the maximum number of data channels to process C_{pa} is thus given by

$$C_{pa} = \frac{C_a - A_{cl} - N_m \times A_m}{A_r \times N_l \times N_r} \tag{9}$$

where C_a is the chip area dedicated to the DWT block, A_{cl} is the control logic area, N_m is the number of CNs per DWT

block, A_m is the area of a CN, A_r is the size of a single memory module, N_r is the number of memory modules required for a single decomposition, and N_l is the number of decomposition levels required.

V. RESULTS

The design proposed in this paper were implemented in VHDL, and compared to MATLAB-based output. Table 2 summarizes the critical path for each design architecture.

In Figs. 3-4, the performance of both architecture is quantified in terms of the number of data channels for a given chip speed and memory requirements. The results suggest that for a small number of channels and decomposition levels, the parallel architecture is favored over the sequential since the difference in size is compensated for by a large margin in speed. The rate of increase in the number of channels is clearly much larger than memory requirement which implies that with a faster chip, the number of channels that can be accommodated grows substantially without a proportional increase in memory requirements.

TABLE 2: CRITICAL PATH RESULTS FOR BOTH ARCHITECTURES. T_m AND T_a DEFINE THE NUMBER OF CLOCK CYCLES FOR A SINGLE MULTIPLIER AND A SINGLE ADDER, RESPECTIVELY

	Clock cycles W_e	Memory modules N_r	# of CNs N_m	Pipeline Length
Parallel	$2T_m + 3T_a$	10	5	7
Sequential	$10T_m + 15T_a$	5	1	2

VI. CONCLUSION

Two design architectures for computing the DWT as a front-end signal processing stage in neuroprosthetic applications have been proposed. The implementations are based on the lifting-scheme for wavelet computation and integer fixed-point precision arithmetic, which minimize computational load and memory requirements. In particular, we showed that a parallel architecture could be efficiently designed to maximize the data throughput and minimize the latency making it more suitable for managing critical temporal information. The sequential architecture is more suited to size constrained applications such as low-density electrode arrays but at the expense of larger latency. By utilizing the difference between the data sampling rate and available computation bandwidth, both architectures yield significant data reduction while maintaining real-time operation. Our most recent results using layout techniques provide the information necessary to compute actual power consumption, as well as estimating the actual size of the individual components [7]. This will be used to determine the exact number of channels to be processed, and the maximum operating frequency for on-chip, real-time

streaming of neural information, which can effectively maximize hardware usage.

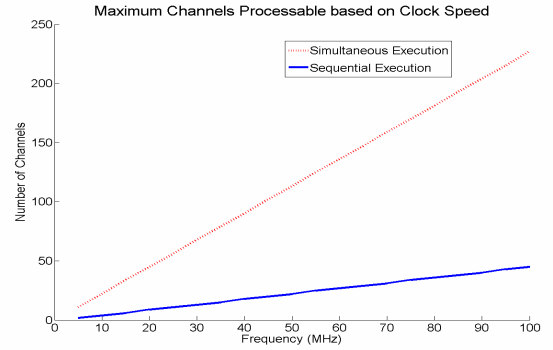


Fig 3. Maximum number of channels to be processed versus clock speed ($T_m = 4T_a$, $T_a = 2$, $S_o = 2$, $N_d = 2$)

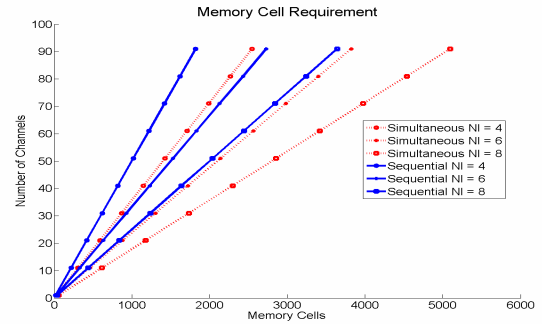


Fig 4. Memory requirements to compute a N_l level DWT, versus number of channels simultaneously processed.

REFERENCES

- [1] 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. of the IEEE*, (92): 1, pp: 76-97, 2004.
- [2] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*4(3), pp. 245-267, 1998.
- [3] K.G. Oweiss, D.J. Anderson, M.M. Papaefthymious, "Optimizing Signal Coding in Neural Interface System-On-A-Chip Modules," *Proc. of 25th IEEE Int. Conf. on Eng. in Medicine and Biology*, pp. 2016-2019, Cancun, Mexico, September 2003.
- [4] K. Andra, C Chakarbarti, and T. Acharya "A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform" *IEEE Trans. Signal Processing*, vol 50, no 4, pp 966-977, 2002
- [5] H. Liao, M.Kr. Mandal, B.F. Cockburn, "Efficient architectures for 1-D and 2-D lifting-based wavelet transforms," *IEEE Transactions on Signal Processing*, volume: 52 , Issue: 5 , May 2004 pp:1315 – 1326
- [6] Y. Suhail, K.G. Oweiss, "A Reduced Complexity Integer Lifting Wavelet Based Module for Real-Time Processing in Implantable Neural Interface Devices," *26th IEEE Int. Conf. on Eng. in Medicine and Biology*, pp. 4552-4555, September 2004.
- [7] K. G. Oweiss, A. Mason, K. Thomson, J. Li, Y. Suhail. "A Scalable Wavelet Transform VLSI Architecture for Real-Time Neural Signal Processing in Implantable Multichannel Neuroprosthetic Devices," *IEEE Trans. On Circuits & Systems*, *in review*.