

Area-Power Efficient Lifting-Based DWT Hardware for Implantable Neuroprosthetics

Awais M. Kamboh, Matthew Raetz, Andrew Mason and Karim Oweiss
Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA

Abstract— Discrete wavelet transform (DWT) has been shown to provide exceptionally efficient data compression for neural records. This paper describes an area-power minimized hardware implementation of the lifting scheme for multi-level, multi-channel DWT. Performance tradeoffs and key design decisions for implantable neuroprosthetics are analyzed. A 32-channel, 4-level version of the circuit has been custom designed in 0.18 μm CMOS and occupies only 0.16mm².

I. INTRODUCTION AND MOTIVATION

Neuroprosthetic devices that utilize microelectrode arrays to monitor neural activity show great promise in many biomedical applications. However, the ability to implant devices into living bodies is limited, in part, by volume and power dissipation constraints set to ensure neighboring tissue and nerves are not damaged. 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 limited hardware. Discrete wavelet transform (DWT) is a very effective method for compressing neural data [1]. The non-zero 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 neuroprosthetic devices, a multi-channel, multi-level implementation is preferred 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 multi-channel, multi-level DWT in real time is highly desirable. In contract to traditional DWT applications, neuroprosthetics can afford long computation intervals, up to 40 μsec [1], permitting hardware to prioritize power and area efficiency over speed. In prior work we have identified an optimal architecture for multi-channel, multi-level DWT in neuroprosthetic applications [2]. The approach relies on the 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 [1].

This paper describes the design of a highly area and power efficient circuit that implements multi-channel, multi-level lifting-based DWT in real time. Results for a 32-channel, 4-level DWT realization are presented.

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

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

where B_0 - B_7 are the eight constant filter coefficients and subscripts of the other variables represent the time sample, 0 is the current sample, -1 is 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 high signal to noise ratio [3]. Based on this analysis, we have chosen 10-bits data and 5-bit coefficients (including sign bit) for our hardware implementation. Derived from our prior system-level analysis [2], we have found that the power-area product can be minimized by a hardware architecture that sequentially evaluates the DWT of multi-channel data in real time. Fig. 2(a) describes the DWT architecture resulting from our circuit-level design efforts. It is composed of a customized computation core (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.

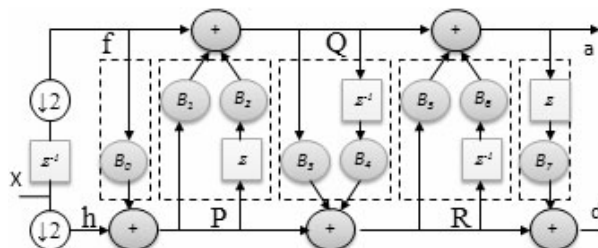


Figure 1: Data flow diagram for Lifting DWT.

This work was supported by NIH grant NS047516-01A2 and the National University of Sciences and Technology, Islamabad, Pakistan.

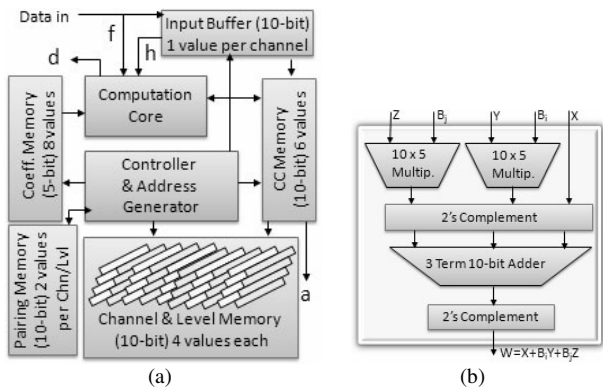


Figure 2. (a) System diagram for sequential calculation of DWT. (b) Computation Core for lifting DWT.

III. COMPUTATION CORE

Analysis of the ‘symmlet4’ lifting factorization and resulting equations in (1) shows a noticeable computation regularity; all arithmetic operations can be expressed in the general form of $W=X+ B_iY+B_iZ$. This regularity can be exploited to minimize hardware by implementing a single computation core (CC) that sequentially executes all computational steps in (1) [3]. Sequential reuse of the same hardware significantly reduces chip real estate requirements without impacting performance in this low bandwidth application [2].

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, and the constant 5-bit filter coefficients could be stored in either form. The CC multipliers are required to perform 10x5 operations. Multipliers operating on two’s complemented values need the multiplicand and the multiplier to have an equal number of bits. Implementing 10x10 two’s complemented multiplication was found to increase the CC size by about 70%. 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. 2(b), the computation core 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.

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 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 [4] and shown in Fig. 3 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 performance or reliability. On 0.18 μm CMOS process, this 16 transistor cell was designed to dissipate 0.4 μW power and occupy 41.5 μm^2 . It was utilized to construct a 10-bit adder for the three-term adder block and a 13-bit adder for the 10x5 multiplier.

B. Multipliers

The CC requires two multipliers to perform 10x5 operations. 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 [5]. Two different recoders and partial product generators were considered both requiring 18 transistors each. Two partial product generator, NPR3b and NPR3b, were tailored to this application, and it was determined that, while power consumption is similar, NPR3b needs slightly fewer transistors (14 vs. 16) and is preferred. The 3:2 and 2:1 compressors are full and half adders respectively. The array multiplier ANDs every bit of the multiplicand with every bit of the multiplier 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, power and delay measurements of the custom designed adder and array multiplier sub-modules selected for use in our DWT CC.

TABLE I. CC TRANSISTOR COUNTS AND WORST CASE VALUES

| Sub-module | # of TX | Area(μm^2) | Power(μW) | Delay (ns) |
|------------|---------|-------------------------|------------------------|------------|
| Adder | 160 | 418 | 3.834 | 0.404 |
| Multiplier | 733 | 2858 | 42.76 | 4.12 |

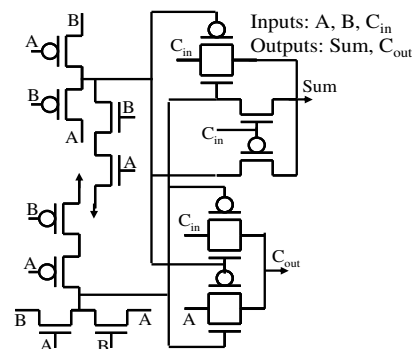


Figure 3. Single bit full adder cell used in the CC adders [4].

IV. MEMORY MODULES

A. Multi-channel/level memory module

Neuroprosthetic 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 multi-level decomposition in real time. Analysis of our DWT implementation indicates that there is sufficient bandwidth within a single computation core to process well over 100 data channels sequentially [3]. 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. Every level and every channel requires a corresponding 40-bit memory register. In our 32-channel, 4-level design, this amounts to 128 registers. The channel/level memory was implemented using an array of standard 6T SRAM cells. 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 lists the area and address decoding cost for several channel/level configurations.

Because DWT operates on data pairs, there is an idle cycle between each computation wherein the first sample of the pair is acquired. Thus, input data buffer of size equal to the number of channels is required to store input samples during the idle cycles. This idle state can be used to compute all results for higher (beyond one) levels of decomposition, where the input data is available from previous computations [3].

TABLE II. TYPICAL CHANNEL/LEVEL MEMORY CONFIGURATIONS

| Chn/Lvl | No. of Tx | Total Area (μm^2) | Decoder Area (μm^2) | Decoder % of total area |
|--------------|-----------|--------------------------------|----------------------------------|-------------------------|
| 2-Chn/2-Lvl | 1748 | 4318 | 618 | 14.33 % |
| 8-Chn/4-Lvl | 9276 | 28857 | 2351 | 8.14 % |
| 32-Chn/4-Lvl | 34814 | 129830 | 9389 | 7.23 % |

B. Coefficient, pairing and computation core memory modules

Due to the sequential reuse of CC hardware, intermediate results from the evaluation of (1) must be stored in computation core memory for use in subsequent filter steps. A total of six 10-bit registers are required to hold intermediate values during a given computation cycle. All the registers have parallel and serial load capability. When performing calculations for higher levels a pairing memory is used to store input values. This memory must contain two 10-bit values for every channel and level except the highest

level. This memory block must write only one 10-bit value but be able to read all 20 bits stored for a channel/level. Due to this read/write structure this memory block was built separate from the channel/level memory and implemented as two separate blocks of SRAM controlled by the same address decoder circuitry. An enable bit for each block is used to select between them. A separate coefficient memory block is required to store the eight 5-bit filter coefficients. Because this data is static, it can be hardwired and effectively implemented as a ROM.

C. Power saving strategies for memory

To reduce power consumption in the memory, a divided bit line structure was used [6] to reduce the overall bit line capacitance that is charged/discharged during read and write cycles. The bit lines were divided into sub bit lines that connect to only eight SRAM cells. Some extra logic was contained in the decoder to control access to these sub bit lines. In addition, the access transistor in the six-transistor SRAM cell is only turned on once during a read cycle to reduce switching power. With reduced bit line capacitance, the SRAM could be implemented without a sense amplifier, which eliminates the need to access both sides of the SRAM cell and reduces bit line charging/discharging currents. This also helps to reduce the overall memory power consumption.

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. Two different controller options were considered, an instruction based microprocessor and a state machine based controller. The repetitive and sequential nature of the required controller functions warrants the use of a state machine based design. Three different phases of controller operation containing a total of eight clock cycles (per channel per level) are required to process each sample. One read cycle is followed by five calculation cycles for the CC and two write cycles. The time period allowed to complete these cycles dictates the clocking frequency of the circuit. For 32 data channels and a sampling frequency of 25 kHz, a clock frequency of 6.4MHz is required. Note that increase in the number of levels has no effect on the clocking as computation of higher level results can be completed in cycles not used by lower levels as can be seen in Fig. 4.

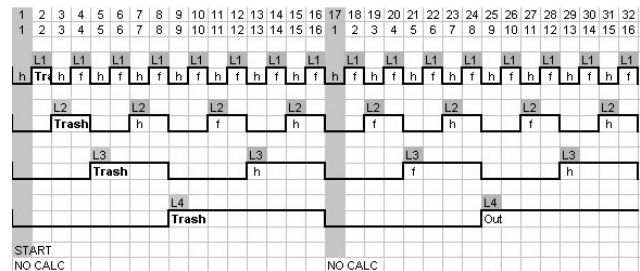


Figure 4. Use of CC for different levels per channel.

For the state machine to work correctly, the number of incoming channels and the required levels of DWT should be known beforehand. In our implementation the incoming data

from multiple channels is assumed to be multiplexed onto a single data line, so the system accepts input from one data bus and outputs data on another.

The state machine controller depends on a counter to keep track of its states and control the DWT block at different levels of system hierarchy. The three least significant bits control the eight phases of operation of CC and memory for a single data pair irrespective of channel or level being processed. For a 32 channel, 4 level system, the next six bits control the channel being processed. This section also directs the input stream to the input buffer or the computation core. The most significant channel bit in conjunction with the three level bits are used to represent the level being processed where each bit represents a separate level. 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 using 4 bits to represent the levels we are able to recognize idle cycles for use in calculating higher level results and also to manage storing the approximation coefficient.

Fig. 5 describes the part of the state machine controlled by the three least significant bits of the counter, i.e. the complex data movement within the CC memory and its interaction with other memory blocks. The top row mentions register names where X, Y and Z are registers connected to respective inputs of the CC while M1, M2 and M3 are intermediate memory registers. Values are read into the CC memory in the first cycle, followed by five cycles of calculations and data shifts. The last computation cycle produces two results; the detail coefficient is transmitted out while the approximation coefficient is transmitted or retained based on the level being calculated. The 4 values in CC memory, f_0 , P_0 , Q_0 and R_0 are stored in channel/level memory to be reused for higher level calculations.

VI. RESOURCE REQUIREMENTS

Table III lists number of transistors required for different modules of the DWT system. The corresponding area consumed, including routing, by each module is also shown. The dominance of channel/level memory over other modules is evident in the transistor counts and area consumption, where increasing the number of channels by a factor of four results in roughly a 300% increase in both the number of transistors and the area consumption (312% number of transistors, 296% area).

TABLE III. TRANSISTOR COUNT AND AREA FOR HARDWARE MODULES

| Module | No. of Tx | Area (μm^2) |
|------------------------------------|-----------|--------------------------|
| Controller | 517 | 3834 |
| Comp Core | 1786 | 13828 |
| Comp Core Memory | 396 | 1775 |
| Coefficient Memory | 264 | 1114 |
| Pairing Memory per channel/level | 120* | Varies |
| Input Buffer per channel | 60* | Varies |
| Per Channel or Level Memory | 240* | Varies |
| Complete DWT system: 8 Chn, 4 Lvl | 13465 | 55248 |
| Complete DWT system: 32 Chn, 4 Lvl | 41943 | 163377 |

*Counts exclude transistors for address decoding circuitry

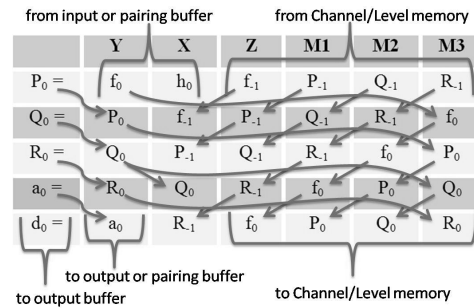


Figure 5. Data movement in CC memory during different phases of controller operation per sample per channel.

VII. CONCLUSIONS

Lifting-based DWT using ‘symmlet 4’ wavelet filters was shown to minimize the required area for hardware by sequential evaluation of filter equations using a specially designed computation core. The adders were based on an area-power efficient transmission gate based full adder cell. For the required 10x5 bit operations, the Booth multiplier was found to be more expensive than an array multiplier after optimization. These power efficient adders and multipliers are arranged into a unique dedicated computation core whose sequential reuse results in an area efficient DWT system. Four different memory modules were implemented using a combination of SRAM and ROM to improve system performance. A state machine based controller was designed to manage the complex data flow. The system is designed to pseudo-simultaneously process multiple channels of incoming neural signals. Furthermore, its ability to perform multi level DWT enables very high data compression while maintaining signal fidelity. Results show that this lifting based architecture has low area and power requirements for multi-channel, multi-level applications. Our implementation in 0.18 μm CMOS requires only 0.16 mm^2 of area to process 32 channels of data at 4 levels of DWT in real time. The small size and low power consumption of the system makes it highly suitable for use in implantable devices.

REFERENCES

- [1] K. G. Oweiss, “A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces,” *IEEE Tran. on Biomed. Eng.*, vol. 53, no. 7, pp. 1364-1377, 2006.
- [2] A. M. Kamboh, and A. Mason, “Comparison of lifting and B-spline DWT implementations for implantable neuroprosthetics,” *IEEE Int. Conf. on Sensors*, Oct 2006.
- [3] A. Mason, J. Li, K. Thomson, Y. Suhail, and K. Oweiss, “Design optimization of integer lifting DWT circuitry for implantable neuroprosthetics,” *IEEE-EMBS Int. Conf. on Microtechnologies in Medicine and Biology*, pp. 112-115, May 2005.
- [4] A. Shams, T. K. Darwish, and M. A. Bayoumi, “Performance analysis of low-power 1-bit CMOS full adder cells,” *IEEE Trans. VLSI systems*, vol. 10, no. 1, pp. 20-19, Feb 2002.
- [5] Z. Huang, “High-Level Optimization Techniques for Low-Power Multiplier Design,” PhD dissertation, Univ. of California, Los Angeles, June 2003.
- [6] B.-D. Yang, L.-S. Kim, “A low power SRAM using hierarchical bit line and local sense amplifiers,” *IEEE Journal of Solid State Circuits*, vol. 40, no. 6, pp. 1366-1376, June 2005.