AN EFFICIENT BUFFER-BASED ARCHITECTURE FOR ON-LINE COMPUTATION OF 1-D DISCRETE WAVELET TRANSFORM

Chengjun Zhang, Chunyan Wang and M.Omair Ahmad, Fellow, IEEE

Center for Signal Processing and Communications
Department of Electrical and Computer Engineering
Concordia University
Montréal, Québec, Canada H3G 1M8

ABSTRACT

In this paper, we propose a flexible architecture that performs the computation of the discrete wavelet transform requiring a small memory space and is capable of operating at high sampling rate. The architecture employs two filtering blocks to compute the transform and one buffer to store the intermediate results. Each filtering block has two processing units that operate independently in parallel using a two-phase scheduling. An efficient scheme for the synchronization of the data flow among the three blocks is provided in order to minimize the buffer size and increase the speed of operation. Verilog and HSPICE simulation results are presented to show that the proposed architecture is more efficient for the computation of a fully decomposed discrete wavelet transform with high-tap filters than some other existing architectures in terms of their areas and speed of operations.

1. INTRODUCTION

The discrete wavelet transform (DWT) [1] is playing an increasingly important role in the area of signal processing. As an analysis tool, the wavelet transform decomposes the signals into different frequency sub-bands with appropriate time resolutions. The higher the frequency band, the finer the time resolution becomes. According to this property, the discrete wavelet transform provides an alternative to Fourier analysis for an efficient representation of a signal and its analysis using adaptive time-frequency windows. The structure to perform the DWT is usually based on a binary tree [1] consisting of lowpass (LP) and highpass (HP) filter banks. To make the DWT practical for on-line applications, the structure realizing the DWT must be designed to operate at high speed. Accordingly, different kinds of architectures for VLSI implementation have been proposed to perform fast DWT with efficient hardware utilization. Most of the architectures can be categorized into two groups, non-cascaded and cascaded. An architecture belong to the non-cascaded group performs the DWT using a single computation block with parallel [2] or systolic [3] filtering structures and a number of registers to store the intermediate results. The DWT is performed using the single computation block for all the levels using a recursive pyramid algorithm (RPA) [1], but the time to perform the entire DWT is long. Moreover, the design of the memory is complicated in the case when the filter length L or the number of resolution levels J is large. One of the solutions addressing these problems is to use cascaded structures, such as those proposed in [4] and [5]. A cascaded structure has J blocks for the J-level DWT, one for each level, performing in parallel. The time taken to perform the DWT in this case can be smaller than that of the non-cascaded ones by a factor of J, at the expense of an increased hardware. On one hand, with the characteristic of higher speed, the cascaded structure can be more suitable for on-line DWT applications. On the other hand, however, the cascaded blocks in such a system could be different from each other, thus increasing the complexity of the design and lowering the scalability of the circuits realizing the structure.

In this paper, we propose an architecture for VLSI implementation to compute 1-D DWT. The design of this architecture focuses on improving the speed and on reducing the area required for the implementation. Also, the proposed architecture aims at providing scalability to the circuit realizing the DWT. Section 2 describes the proposed architecture. In Section 3, simulation results are presented to evaluate the performance of the design by comparing it with that of other structures. Finally, in Section 4, certain conclusions are made and some of the features of the proposed architecture highlighted.

2. PROPOSED ARCHITECTURE FOR DWT COMPUTATION

Most of the DWT architectures, aiming at reducing the size of the buffer, use recursive pyramid algorithm presented in [1] to implement the transform of all levels simultaneously. However, in this implementation, the size of the buffer still increases with the filter length. If the

DWT is performed serially, level by level, the buffer is used to store the intermediate results of one level at a time. The required buffer space, in this case, usually depends on the number of samples in the data sequence, and the amount of data in the first level could be very large. Thus, it is necessary to devise a new approach to handle a large number of signal samples with a minimum-size buffer. To this end, two points should be noted. Firstly, in the filtering process with down sampling [1], the amount of data generated for the computation of one level is one-half of that in the previous level. This means that the buffer space needed to store the data generated in the first level is almost equal to that in all other levels together. Thus a buffer space can be partitioned into two parts, one for the data generated in the first level and the other for all those in the succeeding levels. Secondly, the data stored in the buffer from one level is to be used only for the next-level

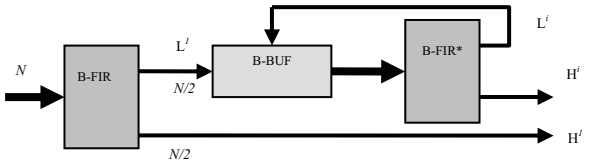

Figure 1. Proposed architecture for the computation of 1-D DWT with two dedicated filtering blocks, B-FIR and B-FIR*, and a special buffer, B-BUF.

computation. Thus, if the computations of all the levels are to be performed in parallel, at least part of the data produced by one level must be available in the buffer for use by the next level. Based on these two points, we now propose an architecture containing one buffer and two filtering blocks, one for the first level and the other for all the remaining levels, to compute 1-D DWT. The basic structure is shown in Figure 1. The block performing the computation of the first level is indicated as B-FIR and that for the computation of all other levels as B-FIR*. The quantity N denotes the number of the input samples, and L^i and H^i are the outputs of the HP and LP filters at level i, where i = 2, 3,...,J. A buffer, denoted as B-BUF, is placed between the two filtering blocks and is designed to suit the requirement of the input data supply for B-FIR*.

For a fully decomposed DWT, if the number of

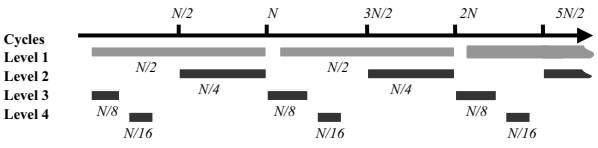

Figure 2. Timing diagram for the operations performed by the blocks B-FIR and B-FIR*. The grey bars represent the computations of the first level performed by the block B-FIR. The black bars indicate those of the other levels performed by B-FIR*. The number under the bar represents the quantity of intermediate results of given level.

resolution levels is J, the sequence length N is equal to 2^J. It should be mentioned that the computation of the first level requires N cycles to produce $\frac{N}{2}$ intermediate sample points, that of the second level requires $\frac{N}{2}$ cycles to produce $\frac{N}{4}$ points, and that of the third level, $\frac{N}{4}$ cycles to produce $\frac{N}{8}$ points, and so on. Based on these requirements, we present a timing diagram shown in Figure 2, for the operations to take place in parallel by the two filtering blocks. After $\frac{N}{2}$ clock cycles, $\frac{N}{4}$ intermediate results of the first level are produced by B-FIR block and thus the block B-FIR* can start to perform the computation for the second level. The operations of B-FIR* are timed in such a way that the computations for the other levels are performed one by one, as shown in Figure 2, so that the intermediate results are used immediately after their generation for performing operations of the next level. The length of the B-BUF can be only $(\frac{N}{4}+1)$. In this way, the computations of the DWT is divided into two sets of parallel operations, one for the first level lasting N cycles and the other for all the other levels lasting N-1 cycles.

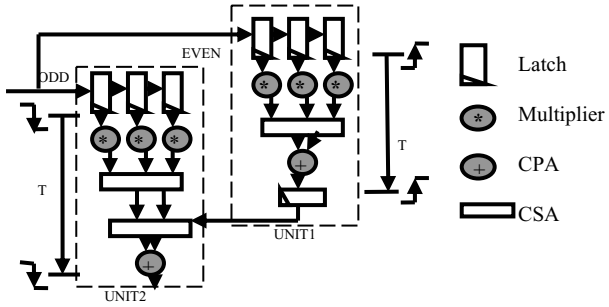

Figure 3. Details of the filtering block, B-FIR or B-FIR*.

The structure of the filtering block B-FIR or B-FIR* using 6-tap filters is shown in Figure 3. It has two processing units, UNIT1 and UNIT2, operating in parallel. The input sequence is spilt into two sub-sequences, one consisting of the odd numbered samples and the other of the even numbered samples of the input sequence. These two sub-sequences are applied to the latches of the two processing units, at the rising and falling edges of the clock, respectively. Each of the filtering blocks, B-FIR and B-FIR*, performs the HP and the LP filtering alternatively. Each filtering block outputs one sample of data per cycle. As the HP and LP filtering operations are performed alternately, the LP (or HP) produces one sample every other cycle. In the two filtering blocks, the filter coefficients, h_i (for HP) and g_i (for LP), where $g_{L-1-k} = (-1)^k * h_k$ (k = 1, 2, …, L), are loaded at the same time. When UNIT1 performs the LP (HP) filtering with the even-numbered samples, UNIT2 does the HP (LP)

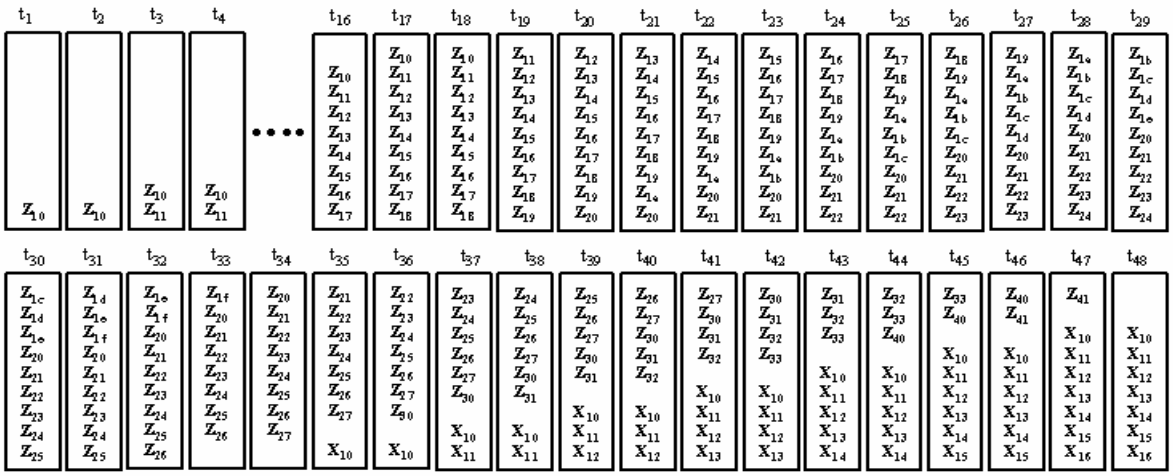

Figure 4. Data updating in the buffer for the DWT computation of a sequence with $N = 32$. Z_{ji} is the result of the computation for the j^{th} level, ($1 \le j < 5$ and $0 \le i < 32/2^{j}$). X_{ji} is the result of the computation for the next input sequence. There are two waiting cycles before B-FIR starts the DWT computation for the new sequence.

filtering with the odd-numbered samples. In this way, the hardware is fully used. The speed of the DWT computation depends on the delay of the filtering blocks consisting of the modules, multipliers, adders and latches. The carry save adders (CSA), instead of the carry propagation adders (CPA), are used between the multiplies and the CPA in the two processing units to reduce the delay of the filtering blocks.

In the proposed structure, the buffer B-BUF should be designed to produce a data flow that suits the operations of the two filtering blocks, in particular, those of the B-FIR* block. The buffer has two input paths, i.e., it accepts one output sample from each filtering block every two cycles, and it provides one output sample from its stored data, by shifting it, to B-FIR* block each clock cycle. The loading of the samples into the buffer should be specifically arranged so that the B-FIR* block can receive the correct samples shifted out of the buffer, according to the timing diagram shown in Figure 2. Figure 4 illustrates the data updating of B-BUF for the DWT computation of a 32-sample sequence. In this case, the buffer consists of nine shift registers. Each rectangle shows the buffer content during a given clock cycle, and the data shift from the bottom to the top of the buffer. The first 16 rectangles illustrate the data updating of the first 16 cycles at the beginning of the transform computation. The remaining rectangles illustrate the data updating when the circuit realizing the proposed architecture is in full operation and the buffer fully occupied.

To recapitulate, the new scheme aims at developing an architecture with improved performance to compute the DWT. The input data sequence to each filtering block is split into two sub-sequences according to odd and even numbered samples of the input sequence. To accelerate the filtering process, the application of the two sub-sequences to the two processing units in B-FIR (or B-FIR*) are timed with the rising and falling edges of the clock signal, respectively. Thus, the delay in the filtering process can be significantly reduced. Moreover, in order to reduce the area of the circuit implementing the proposed architecture, the architecture itself is designed to operate with a buffer that is much smaller than that used in most of the DWT circuits. It should also be noted that the filtering blocks are designed with a given number of taps and can fit for different number of resolution levels of the transform, thus providing certain flexibility for the DWT applications. Such a flexibility is obtained without sacrificing the area of the implementation or speed. In the following section, the results of the performance evaluation of the proposed scheme obtained through simulations are presented.

3. PERFORMANCE EVALUATION

The efficiency of the proposed approach is evaluated by Verilog and HSPICE simulations of the proposed architecture. The simulation is carried out for a 6-level DWT computation using 10-tap filters, i.e. $J = 6$, $L = 10$, and $N = 64$. A complete DWT computation can be performed in $N_{clk} = 2^{J}$ clock cycles. The minimal cycle time t_{clk} is given by

$$t_{clk} = S * t_{latch} + M * t_{cpa} + t_{csa} + t_{mul} \qquad (1)$$

TABLE I . A COMPARATIVE PERFORMANCE OF VARIOUS ARCHITECTURES FOR 10-TAP FILTER BASED DWT COMPUTATION WITH $J = 6$, $N = 64$.

Architecture	Number of multipliers (t_{mul} =6.2 ns)	Number of adders (t_{cpa}=2.4 ns, t_{csa}=0.4 ns)	Number of registers (t_{latch} =0.4 ns)	Number of clock cycles (N_{clk})	Duration of the clock cycle (t_{clk}) ns	*Latency of the filtering block in clock cycles
Parallel [2]	20	18	60	66	17.8	1
Systolic [3]	20	20	60	66	10.2	10
DRU cascaded [5]	20	18	59	66	17.8	1
Systolic cascaded [4]	40	37	60	34	11.8	5
Proposed	20	18	36	66	13.8	1.5

* The Latency is defined as the time required, measured by the number of clock cycles, for the filtering block to complete its procedure of the processing.

where t_{latch} is the delay of the latch, t_{cpa} and t_{csa} are, respectively, the delays of the two adders, CPA and CSA, and t_{mul} is the delay of the multiplier. The quantities M and S represent, respectively, the numbers of CSAs and latches in each processing unit of a filtering block.

We use HSPICE simulation to evaluate the delays of the basic modules, namely t_{latch}, t_{mul}, t_{cas} and t_{cpa}. These results are then used in the Verilog simulation. Also, the structure of these modules are needed for the HSPICE simulation in order to estimate the area of an implementation of the proposed architecture. Based on these results, t_{clk} can be calculated in a realistic manner and then verify using the Verilog simulation.

The simulation results of an implementation of the proposed architecture as well as those of four other architectures given in [2]-[5] are listed in Table I. In this table, the first three columns show the number of each of the three modules, namely, multipliers, adders, and registers used in each implementation. Assuming the area of a register to be unity, the area of an adder is determined to be 1.5 and that of a multiplier 4.2. The proposed design has a smaller number of registers than other designs have, which results in a reduced overall area. Compared to the DRU cascaded architecture of [5], that has the smallest area among the four other architectures considered in this paper, the implementation of the proposed architecture has a 16.8% less area. The fourth and fifth columns of the table illustrate that the proposed architecture provides a 22.5% improvement in the speed compared to that of the parallel and DRU cascaded architectures, but the speed is a bit slower compared to that of the systolic architecture of [3]. The systolic cascaded architecture of [4] is, however, much faster, but this was achieved at the expense of a much larger area. In terms of the latency, the proposed architecture has a significant advantage over the systolic architectures.

4. CONCLUSION

In this paper, we have proposed a new architecture for the computation of the 1-D DWT. This architecture includes two types of blocks: two filtering blocks and one buffer block. The two types of blocks have been specially designed to operate efficiently and at a high speed by introducing parallel processing in the filtering blocks and special scheme for the data flow into the buffer block. The simulation results have shown that the proposed architecture has a better performance, in terms of the speed of operation and the area of implementation, in comparison to that of other architectures in similar categories. Since the number of taps affects the filtering blocks only in changing the number of modules in the blocks, and the number of levels affects only the size of the buffer, the proposed architecture is modular and flexible.

The proposed structure can be optimized by combining some of the modules or adding pipelines in the processing units of the filtering blocks to provide a better speed performance. The architecture presented in this paper is appropriate for on-line applications, such as real-time audio coding and image compression.

REFERENCES

[1] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Trans. Signal Processing*, vol. 42, no. 3, pp. 673–677, Mar. 1994.

[2] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: A survey," *Journal of VLSI Signal Processing*, vol. 14, no. 2, pp. 171–192, Dec. 1996.

[3] K.Parhi and T.Denk, "Systolic VLSI architectures for 1-D discrete wavelet transforms," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, Nov. 1998, Pacific Grove, California.

[4] F.Marino, D.Guevorkian, and J.Astola, "Highly efficient high-speed/low-power architectures for the 1-D discrete wavelet transform," *IEEE Trans. CAS-II*, vol. 47, no. 12, pp. 1492-1502, Dec. 2000.

[5] T.Park, "Efficient VLSI architecture for one-dimensional discrete wavelet transform using a scalable data reorder unit," in *Proc. International Technical Conference on Circuits/Systems, Computers and Communications,* July 2002.