EFFICIENT VLSI ARCHITECTURE OF LIFTING-BASED WAVELET PACKET TRANSFORM FOR AUDIO AND SPEECH APPLICATIONS

Chao Wang[12], and Woon Seng Gan[2]

[1]Center for Signal Processing, Nanyang Technological University
[2]DSP Lab, School of Electric and Electronics Engineering, Nanyang Technological University

ABSTRACT

This paper presents a novel VLSI architecture for discrete wavelet packet transform (DWPT). By exploiting the in-place nature of the DWPT algorithm, this architecture has an efficient pipeline structure to implement high-throughput processing. Folded architecture for lifting-based wavelet filters is proposed to compute wavelet butterflies in different groups simultaneously, at each decomposition level. Internal pipelining and by-pass mode are employed on each processing element to increase computation throughput and provide easy configuration for arbitrary decomposition, respectively. According to the comparison results, our proposed VLSI architecture is more efficient than previous proposed architectures in terms of arithmetic operations, storage requirement, and throughput.

1. INTRODUCTION

In the last decades, discrete wavelet transform (DWT) [1] has been successfully used in a wide range of applications, including numerical analysis, signal analysis, image and video coding, pattern recognition, statistics, and physics. As a generalization of DWT, discrete wavelet packet transform (DWPT) provides good temporal and spectral resolutions in arbitrary regions of the time and frequency (TF) plane [1-2]. Due to the characteristic of flexible TF decomposition, DWPT has also been widely used in many applications, especially in speech and audio coding, speech enhancement, speech recognition, hearing aid, etc.

Many VLSI architectures have been proposed for computing DWT in the past. However, it is not the case for DWPT. There are very few papers on the development of specific VLSI architectures for DWPT. Wu *et al* [3] designed a programmable processor using two-buffer memory system of size $2N$ and a single MAC (multiplier-accumulator) to calculate different subbands. The Trenas' architecture [4] used a single PE (processing element) consisting of L multipliers working in parallel and L-1 adders for each low-pass and high-pass filters (L is the number of filter taps) to increase the computation throughput. Trenas *et al* [5] also proposed a pipelined architecture, applying a series of J PEs (MACs) communicated by J memory banks to compute each level of the total J levels. A parallel architecture for computing lifting-based DWPT was presented by Arguello [6]. It consists of a group of PEs (MACs) operating in parallel on the data prestored in a memory bank. The main drawback of all the existing methods is that they all use on-chip memory to store the intermediate coefficients and involve intense memory access during the computation process, which consume large silicon area and power dissipation. They also require dedicated control circuitry to generate the correct sequence of read and write addresses.

In this paper, we present a novel VLSI architecture for computing multi-level DWPT. By exploiting the parallelism and regularity of the DWPT algorithm, an efficient pipeline architecture is proposed to implement high-throughput processing. A folded PE for lifting-based wavelet filters is proposed to interleave the multiple groups of butterfly computation on a single PE at each level. Without on-chip memory requirement and access, this architecture is not only area-efficient but also power-efficient.

2. IN-PLACE WAVELET PACKET TRANSFORM ALGORITHM

DWT and DWPT are often implemented by a tree-structure filterbank [1]. In the DWT, only the outputs of the low-pass filters are further processed at the next decomposition level, while the outputs of the high-pass filters remain as the final results. However, DWPT allows further processing of both outputs of the low-pass filters and those of the high-pass filters at the next level, which provides the flexibility to produce arbitrary decomposition of the input signal on the TF plane.

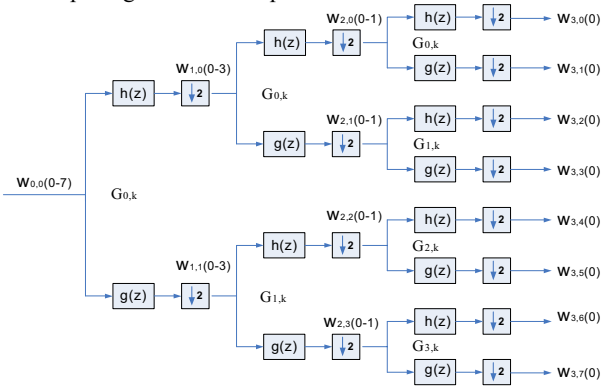

Fig. 1. A complete 3-level tree structure filter bank

Figure 1 shows a complete 3-level tree for DWPT. At each level j (j=0,1,2), each sequence $w_{j,i}(n)$ (i=0,1,…,2^j-1) from the previous level is fed into low-pass and high-pass filters of $h(z)$ and $g(z)$, respectively. From a multiresolution point of view, the output from the high-pass filter contains the detailed information of the input, while that from the low-pass filter contains the approximated information. Thus, at each resolution level, wavelets can

decompose the signal into approximated and detailed signals at the next subsequent level.

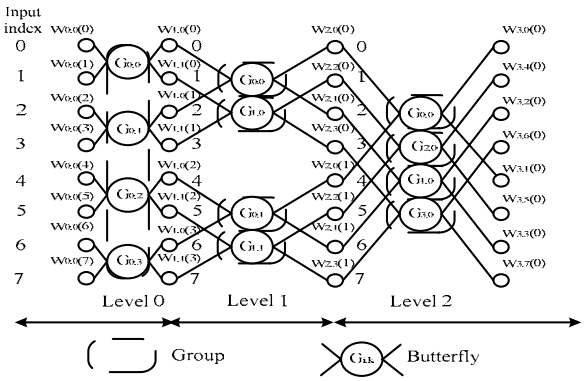

Fig. 2. Signal flow graphs of 3-level full-decomposition WPT

The 8-point 3-level DWPT full decomposition can also be described by a signal flow graphs (SFG) in Fig. 2. The wavelet butterfly in the SFG represents the low-pass and high-pass filters with downsamplers in Fig. 1. The first output and the other output of each butterfly are the low-pass filter and high-pass filter coefficients, respectively. Generally, there are $J=\log_2 N$ butterfly stages or levels. The $N/2$ butterflies at each level are localized in groups, where they process the coefficients from the same frequency band at the previous level. The number of groups (2^j) doubles at each new level, while the number of butterflies in each group ($N/2^{j+1}$) is reduced by 2. The k^{th} butterfly in the i^{th} group at each level is denoted as $G_{i,k}$. The notation $w_{j,i}(t)$ is expressing the t^{th} (intermediate) wavelet coefficient in the i^{th} group at the j^{th} level.

While computing J-level direct DWPT, each wavelet butterfly produces two new coefficients from two samples read from the memory, together with the previous L-2 old samples (L is the number of the wavelet filter taps). Since L-2 samples are shared by the input sequences to every two consecutive butterflies (eg, $G_{0,0}$ and $G_{0,1}$ at level 1) in the same group, the shared data can be buffered by extra storage provided by the butterflies [4]. Thus, the two new coefficients can be stored back into the same memory locations occupied by the two input data. Therefore, in-place computation is allowed. Observe from Fig. 2 that at each level j, $N/2$ butterflies are computed with their input indices at a distance of 2^j (1,2,4, …,$N/2$), as they belong to 2^j different groups. The proposed VLSI architecture presented in the next section employs this observation of the in-place DWPT computation.

3. PROPOSED EFFICEINT VLSI ARCHITECTURE

In this section, we present an efficient VLSI architecture for lifting-based DWPT, similar with the one proposed in [7] to implement high-speed FFT operation. Without loss of generality, we consider a 3-level Daubechies-4 DWPT with a complete tree in this study.

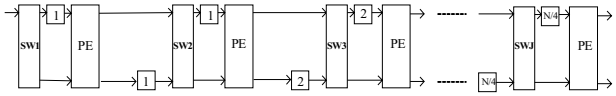

Fig. 3. DDC pipeline block diagram

3.1. Proposed pipeline DDC architecture

By exploring the in-nature parallelism and regularity in the data flow of DWPT as described in Fig. 2, we derived a pipeline VLSI architecture called double-path delay commutator (DDC) architecture. The DDC architecture has J pipeline stages, of which each is corresponding to one of the J levels in the DWPT. It consists of J processing elements to compute wavelet butterflies, and a series of delay commutators (switches and shift registers) which reorder the data coming from the PE in one given stage, so as to present them in the right order for the PE in the next stage. Figure 3 shows how the PEs are interleaved by the delay commutators, as well as the flow of data through the pipeline structure.

The key idea of the DDC architecture is derived from the observation that at each level j, $N/2$ butterflies are computed with their indices at a distance of 2^j. This property of the DWPT algorithm leads itself to rapid pipelining by appropriate data delay and switching in each stage. In this architecture, the input sequence

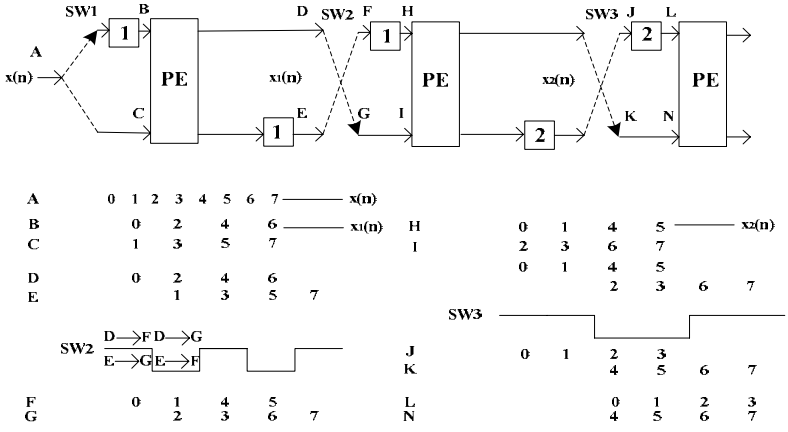

Fig. 4. DDC architecture for 3-level DWPT and its data flow graph

is broken into two parallel data streams flowing forward, with correct "distance" between data elements entering the PEs by proper delays in each pipeline stage. The DDC architecture for a 3-level full decomposition DWPT is given in Fig. 4. The data flow containing input indices at each stage with delay and switching illustrates how the pipeline processing of this architecture operates. The number of unit delay elements doubles at each subsequent stage. The switching frequency of the switch at a given stage is always half of that of the switch at the previous stage. By the means of appropriate delay and switching as shown in Fig. 4, this architecture can process the samples continuously in the exact way specified by Fig. 2. Due to its regular structure and its simple control, this DDC architecture is a good choice for high-speed, high-level DWPT.

3.2. Proposed folded PE for lifting-based DWPT

Daubechies and Swedlens [8] proved that any FIR wavelet filter can be factored into a cascade of lifting steps. By factoring the polyphase matrix for the wavelet filter into a finite product of upper and lower triangular matrices and a diagonal normalization matrix, the computational complexity required by DWT can be reduced by up to 50% [8]. The factorization of the polyphase matrix for the Daubechies-4 (D4) wavelet filter is given by:

$$P(z)=\begin{bmatrix}1 & -\sqrt{3}\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \frac{\sqrt{3}}{4}+\frac{\sqrt{3}-2}{4}z^{-1} & 1\end{bmatrix}\begin{bmatrix}1 & z\\ 0 & 1\end{bmatrix}\begin{bmatrix}\frac{\sqrt{3}+1}{\sqrt{2}} & 0\\ 0 & (\frac{\sqrt{3}+1}{\sqrt{2}})^{-1}\end{bmatrix} \quad (1)$$

One of the implementation for the lifting-based D4 wavelet forward transform in [8] is expressed as follows:

$$\begin{aligned}
&s^1(l)=x(2l)+\sqrt{3}x(2l+1),\\
&d^1(l)=x(2l+1)-\sqrt{3}/4\,s^1(l)-(\sqrt{3}-2)/4\,s^1(l-1),\\
&s^2(l)=s^1(l)-d^1(l+1),\\
&s(l)=(\sqrt{3}-1)/\sqrt{2}\,s^2(l),\\
&d(l)=(\sqrt{3}+1)/\sqrt{2}\,d^1(l).
\end{aligned} \quad (2)$$

Note that we start with a sequence $x(2l)$ and $x(2l+1)$ to represent the even and odd indexed samples, respectively. The intermediate values during the lifting are denoted as $s^k(l)$ and $d^k(l)$ (k=1,2,…), where l is the time index. The sequence $s(l)$ and $d(l)$ represent low-pass and high-pass filter coefficients, respectively. From (2), we derive the direct implementation of the D4 wavelet forward transform as depicted in Fig. 5 (n=1 in this case). The internal pipeline registers are ignored in this section. However, it is not suitable for the DDC architecture.

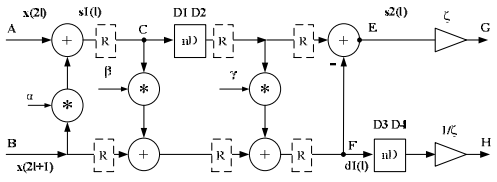

Fig. 5. Proposed folded architecture of lifting-based D4 wavelet filter (n=2^j) with internal pipeline stages, where α=sqrt(3), β=-sqrt(3)/4, γ= -(sqrt(3)-2)/4, and ζ= (sqrt(3)+1)/ sqrt(2). Note that the intermediate signals $s^1(l)$, $s^2(l)$ and $d^1(l)$ are represented here by $s1$(l), $s2$(l) and d1(l), respectively. "D" and "R" represent delay registers and internal pipeline registers.

A folded architecture of the lifting-based D4 wavelet filter is proposed with the DDC architecture. At each stage, multiple groups of butterfly operations are mapped on one D4 wavelet filter, as depicted in Fig. 5. It is observed that any two consecutive butterflies in the same group at level j, are separated by 2^j-1 (0,1,3,…,N/2-1) butterflies from the other groups (see Fig. 2). In order to interleave the butterfly operations form different groups at level j, n=2^j unit delay registers can be placed at each unit delay element position in the PE. Therefore, signal $s^1(l)$ is delayed by 2^j cycles for the next subsequent butterfly computation belonged to the same group at level j. For the synchronization with $s^2(l)$, $d^1(l)$ is also delayed by 2^j cycles for waiting new $d^1(l)$ sample to calculate $s^2(l)$. At the meanwhile, the PE can still process the next pair of inputs to calculate the butterfly from the next group without pausing. As an example, the data flow shown in Table I illustrates how two groups of butterfly calculations are interleaved on a single folded PE at the 2nd level of 3-level D4 DWPT.

In general, at each level, only one D4 wavelet filter is folded

TABLE I
DATA FLOW FOR THE FOLDED D4 PE AT THE 2ND LEVEL IN FIG. 5

Cycle	Input A B	C	D1, D2	E F	D3, D4	Output G H
1	$w_{1,0}(0)$ $w_{1,0}(1)$	$s1_{1,0}(0)$		 $d1_{1,0}(0)$		
2	$w_{1,1}(0)$ $w_{1,1}(1)$	$s1_{1,1}(0)$	$s1_{1,0}(0)$	 $d1_{1,1}(0)$	$d1_{1,0}(0)$	
3	$w_{1,0}(2)$ $w_{1,0}(3)$	$s1_{1,0}(1)$	$s1_{1,1}(0)$, $s1_{1,0}(0)$	$s2_{1,0}(0)$ $d1_{1,0}(1)$	 $d1_{1,1}(0)$, $d1_{1,0}(0)$	$s_{1,0}(0)$ — $w_{2,0}(0)$ $d_{1,0}(0)$ — $w_{2,1}(0)$
4	$w_{1,1}(2)$ $w_{1,1}(3)$	$s1_{1,1}(1)$	$s1_{1,0}(1)$, $s1_{1,1}(0)$	$s2_{1,1}(0)$ $d1_{1,1}(1)$	 $d1_{1,0}(1)$, $d1_{1,1}(0)$	$s_{1,1}(0)$ — $w_{2,2}(0)$ $d_{1,1}(0)$ — $w_{2,3}(0)$
5			$s1_{1,1}(1)$, $s1_{1,0}(1)$	$s2_{1,0}(1)$	 $d1_{1,1}(1)$, $d1_{1,0}(1)$	$s_{1,0}(1)$ — $w_{2,0}(1)$ $d_{1,0}(1)$ — $w_{2,1}(1)$
6			$s1_{1,1}(1)$	$s2_{1,1}(1)$	 $d1_{1,1}(1)$	$s_{1,1}(1)$ — $w_{2,2}(1)$ $d_{1,1}(1)$ — $w_{2,3}(1)$

Note that the intermediate signals $s^1(l)$, $s^2(l)$ and $d^1(l)$ are represented here by $s1_{j,i}$(l), $s2_{j,i}$(l) and $d1_{j,i}$(l), respectively. j and i denote the decomposition level and the group, respectively. D1, D2, D3 and D4 represent the signals at the outputs of the unit delay registers as specified in Fig. 5. Ignore the pipeline registers in this example.

to compute the 2^j groups of butterflies by placing $n=2^j$ unit delay registers into the corresponding unit delay locations. In this way, the PE at each level can compute the butterflies exactly specified by Fig. 2.

3.3. Evaluation and implementation

We proposed a folded architecture of the lifting-based D4 wavelet filter suitable for the DDC architecture. As shown in Fig. 5, the critical path for the D4 wavelet filter can be reduced from $(3T_m+4T_a)$ to (T_m+T_a) by adding 6 extra pipeline registers and dividing the PE into 4 pipeline stages. T_m and T_a are time required for performing one multiplication and one addition, respectively. Thus, the processing time for computing one 3-level D4 DWPT is $4(T_m+T_a)$. Generally, the processing time for computing a frame is $N/2$ cycles. Storage registers are $3(N-1)$ delay registers and $6\log_2 N$ internal pipeline registers.

TABLE II
COMPARISON OF SEVERAL ARCHITECTURES FOR D4 DWPT

Author	Architecture	Arithmetic operation	Storage (Size)	Processing Cycle
Wu [3]	Single PE Conv-based	14	Memory ($2N$)	-
Trenas [4]	Single PE Conv-based	14	Memory (N)	$JN/2$
Trenas [5]	Pipeline Conv-based	14	Memory ($4(N-1)$)	$LN=4N$
Arguello [6]	Parallel Lifting-based	12	Memory (N)	$1.5N$ (J PEs)
Proposed	Pipeline Lifting-based	9	Registers ($3(N-1)+6J$)	$N/2$

The comparison of several architectures with our proposed architecture is listed in Table II. Our method has the least computational complexity due to the lifting-scheme employed. The significant increase in arithmetic operations in Arguello's parallel approach [6] comes from the regularization of lifting steps. Our scheme has the shortest processing cycles for N-point final coefficients. Wu's single PE method [3] uses two-buffer memory system to obtain real-time processing within one frame period, which use a high clock frequency rate and consume a large silicon area. Trenas's single-PE scheme [4] has relatively long computational time for high-level DWPT. Trenas's pipeline architecture [5] has a longer time when it implements a long-tap wavelet filter. Arguello's approach [6] has a processing time of $1.5N$ cycles when it applies J PEs operating in parallel. Considering that other architectures need extra memory to buffer the data, our proposal is more area-efficient because it only uses shift registers and a few pipeline registers. Our architecture is also likely to be more energy-efficient not only because of fewer arithmetic computations required but also by avoiding memory access and data fetching on long wires [9].

In this study, a 3-level D4 DWPT core employing the proposed VLSI architecture has been captured by VHDL and the functionality was verified by RTL and gate-level simulation. By adding a few extra registers, the folded PE in the architecture can be configured into normal or by-pass working modes to compute any required subtree. Zero-padding method as proved in [9] was used to deal with boundary extension issue. To handle the overflow problem, the internal wordlength (18-bit) in each PE is 2 bits more than the input signals. To estimate the area, timing and power information for ASIC design, we used Synopsys Design Compiler to synthesize the circuits into gate level with standard cells from TSMC 0.18μm digital library. The estimated area and clock period are $0.549mm^2$ and 10ns, respectively. The area overhead of the registers for the two folded PEs in the 3-level DWPT core is only an increase of 1.3% and 2.2% of the total area, respectively. It achieves a 53.3% reduction in hardware area with comparison to the direct mapped tree-structure architecture for the 3-level DWPT. The power dissipation for computing an 8-point DWPT is 26mW at 100MHz and 1.8V.

The proposed DDC architecture with novel folded lifting-based wavelet filter can be generalized for J-level, N-data lifting-based DWPT using any L-tap wavelet filters. The PE at each level is folded by placing 2^j unit delay registers into the corresponding unit delay positions. The throughput for computing a frame is $N/2$ cycles. Storage required are $(M+1)(N-1)$ delay registers and some internal pipeline registers, where M is the number of unit delay element in the direct implementation of lifting-based wavelet filter.

4. CONCLUSION

An efficient pipelined VLSI architecture has been presented for a lifting-based discrete wavelet packet transform (DWPT). By applying a novel folded and internally pipelined wavelet filter at each level, this new architecture can perform pipelining processing with a throughput of $N/2$-cycle for an N-point DWPT. The comparison with the other existing architectures shows that the proposed scheme is an ideal architecture for VLSI implementation of DWPT.

REFERENCES

[1] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[2] R. R. Coifmen and M. V. Vickerhauser, "Entropy-Based Algorithms for Best Basis Selection," *IEEE Trans. Information Theory*, vol. 38, pp. 713-718, 1992.

[3] X. Wu, Y. Li, and H. Chen, "Programmable Wavelet Packet Transform Processor," *IEE Electronics Letters*, vol. 35, no. 6, pp. 449-450, 1999.

[4] M. A. Trenas, J. Lopez, M. Sanchez, F. Arguello, and E. L. Zapata, "Architecture for Wavelet Packet Transform with Best Tree Searching," in *Proc. IEEE Int. Conf .on Application-Specific Systems, Architectures and Processors,* pp. 289-298, 2000.

[5] M. A. Trenas, J. Lopez, M. Sanchez, F. Arguello, and E. L. Zapata, "An Architecture for Wavelet-Packet Based Speech Enhancement for Hearing Aids," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing,* vol. 2, pp. 849-852, 2000.

[6] F. Arguello, J. Lopez, M. Sanchez, M. A. Trenas, and E. L. Zapata, "Architecture for Wavelet Packet Transform Based on Lifting Steps," *Journal of Parallel Computing,* vol. 28, no. 7-8, pp. 1023-1037, 2002.

[7] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975, pp. 600-614.

[8] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal Fourier Anal. Applicat.,* vol. 4, pp. 247-269, 1998.

[9] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient Architecture for 1-D and 2-D Lifting-Based Wavelet Transforms," *IEEE Trans. Signal Processing*, vol. 52, no. 5, pp. 1315-1326, 2004.