Synthesis of Memory-based VLSI Architectures for Discrete Wavelet Transforms*

Seonil Choi, Jongwoo Bae and Viktor K. Prasanna
Integrated Media Systems Center
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2562
WWW:http://www.usc.edu/dept/ceng/prasanna/home.html
{seonil, jongwoo, prasanna}@halcyon.usc.edu

ABSTRACT

We propose novel VLSI architectures for computing the Discrete Wavelet Transforms. The proposed architectures employ a memory-based approach. ROM look-up tables are used for the implementation of complex computational modules. Compared with known architectures that employ traditional hardware computational modules, the proposed architectures are faster and are area-efficient. The memory-based architecture is used to implement the block-based DWT with parallel I/O. The resulting architectures are area-efficient and have high throughput and low latency. These architectures are suitable for low-power single-chip implementations which are useful for DWT-based mobile/visual communication systems.

1 Introduction

Discrete Wavelet Transform (DWT) [4] is an alternative to the existing time-frequency representations such as DFT and DCT, and is becoming popular in many signal and image processing applications. It can be used as a viable tool especially in image compression because of its capability for multi-resolution representation.

Recently, several VLSI architectures have been proposed to realize single chip designs for DWT [1, 5]. The architectures consist of three major parts, a computational module, a memory and a routing/scheduling mechanism. The computational module consists of multipliers and adders. In the design of hardware-based architectures, the computational modules are highly pipelined at the cost of increased area. The area of computational module is as dominant as that of memory in general. In this paper, we propose memory-based VLSI architectures for block-based DWTs with parallel I/O [1]. ROM look-up tables are used to implement the computational modules. The resulting architectures have higher throughput and lower latency, and are highly area-efficient compared with architectures using hardware computational modules.

* This research was supported in part by NSF under grant CCR-9317301 and in part by (Defense) Advanced Research Projects Agency under contract DABT63-95-C-0119.

In Section 2, we briefly describe the DWT algorithm. In Section 3, the memory-based approach for realizing multiplier is explained. We also show our architecture for DWT filters and the overall architecture. The overall area of our design and the latency are also analyzed. Concluding remarks are made in Section 4.

2 DWT Algorithm

The wavelet transform decomposes signals at one resolution into signals at a lower resolution called approximation signals and remaining signals called detail signals. The approximation signals correspond to low-pass filtered signals and the detail signals are high-pass filtered signals. Thus, subsequent levels can add more detail to the information content. For the sake of completeness, we review a well known algorithm called Pyramid Algorithm developed by Mallat [4]. Practical applications of the wavelet transforms have been proposed based on this. The general 1-Dimensional Discrete Wavelet Transform (1D DWT) algorithm is represented as

$$A_{2^j} f(t) = A_{2^{j+1}} f(t) * h(2t)$$

$$D_{2^j} f(t) = A_{2^{j+1}} f(t) * g(2t)$$

$$\text{where } t = 0, 1, ..., 2^j - 1; \quad j = 0, 1, ..., J-1.$$

The original signals $f(t)$ have the resolution 2^J. By 2:1 down-sampling using the wavelet filters, $f(t)$ are decomposed into two signals at half-rate, or at half resolution. $A_{2^j} f(t)$ are called the discrete approximation signals of $f(t)$ at resolution 2^j and $D_{2^j} f(t)$ are called the discrete detail signals at resolution 2^j. The quadrature mirror filters, $h(t)$ and $g(t)$, are a low-pass filter and a high-pass filter derived from the wavelet, respectively. If the signals $A_{2^{j+1}} f(t)$ at resolution 2^{j+1} are convolved with $h(t)$, the approximation signals $A_{2^j} f(t)$ at the next level are extracted. By convolving with $g(t)$, the detail signals $D_{2^j} f(t)$ at the next level are extracted.

Obviously, this process can be iterated on one or both the signals. Figure 1 shows the computations for the case of 3-levels. To achieve finer resolution,

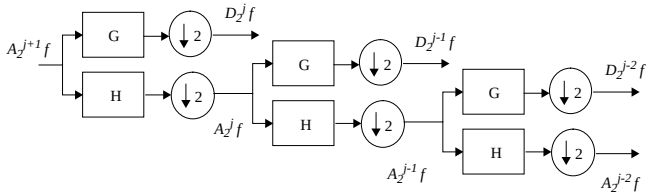

Figure 1: 1D DWT Filter Bank Structure

the scheme is iterated using the approximation signals $A_{2^j}f(t)$. One iteration of the scheme on the first approximation signals $A_{2^{j+1}}f(t)$ creates new approximation signals $A_{2^j}f(t)$ and detail signals $D_{2^j}f(t)$. Each further iteration halves the resolution of the approximation signals. During each iteration, the current detail signals correspond to the difference between the previous approximation signals and the current approximation signals. In block-based DWT, the original image is divided into smaller blocks. They are processed separately using 1D DWT.

3 Proposed Architectures

In this section, we first outline how to perform multiplication by using memory-based architecture. Following this, we briefly explain an architecture for DWT filter bank. Using this we show complete design for block-based DWT.

3.1 Memory-based Multiplier

In computing the DCT, DFT and DWT, multipliers are the fundamental computing elements. Since these multipliers consume significant area, the number of multipliers and adders that can be employed on a chip is limited. The memory-based approach provides an efficient way to replace multipliers by small ROM tables such that the DWT filter can attain high computing speeds with a small silicon area.

Traditionally, multiplication is performed using logic elements such as adders, registers etc. However, multiplication of two n-bit input variables can be performed by a ROM table of size 2^{2n} entries. Each entry stores the precomputed result of a multiplication (See Figure 2 (a)). The speed of the ROM table look-up is faster than that of hardware multiplication if the look-up table is stored in the on-chip memory. In general, 2^{2n}-word ROM table is too large to be practical. In DWT, one of the input variables in the multiplier can be fixed. Therefore, a multiplier can be realized by 2^n entries of ROM as shown in Figure 2 (b). Moreover, the input variables can be partitioned into smaller variables and smaller ROM tables can be used. If we divide the n-bit input to two $n/2$-bit inputs, a multiplier can be implemented by ROMs of size $2^{(n/2)+1}$ entries and an adder (See Figure 2 (c)). The sum of the sizes of resulting ROM tables is smaller than the size of the original ROM table. This

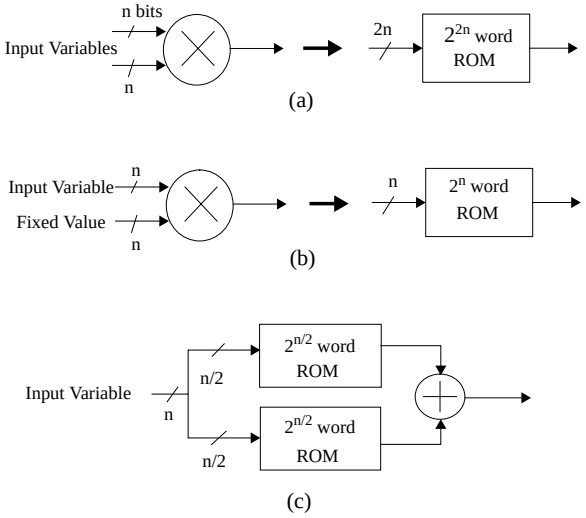

Figure 2: ROM Table Approach for Multiplication

technique has been used in many applications [6, 7].

3.2 Design of Filters

The architecture for computing the DWT consists of computational modules and routing/scheduling modules as shown in Figure 3. The computational modules of DWT consist of filters, $h(t)$ and $g(t)$, as defined in Section 2. The transfer function of these filters can be represented as

$$G(z) = g_0 + g_1 z^{-1} + \cdots + g_{l-1} z^{-(l-1)}$$

$$H(z) = h_0 + h_1 z^{-1} + \cdots + h_{l-1} z^{-(l-1)}$$

where l is the size of the filter.

The DWT filters consist of filter coefficients, g_i, and delays, z_{-i}. The DWT coefficients are generated after applying the high-pass and the low-pass filter. The delays needed in the filter are managed in the routing/scheduling modules. Assume that the size of input variables is n bits. If we multiply the l input variables with the filter coefficients in parallel, l n-bit multipliers and $\{l-1\}$ $2n$-bit adders are needed (See Figure 4 (a)). If traditional hardware is used, then the latency of these multipliers and adders as well as the chip area of the overall design can be large.

In the proposed memory-based architecture the multiplications are performed using ROM tables. The size of the ROM table needed to implement a filter with l input variables is 2^{nl}. We partition each input variable into small segments. Assume that the size of a segment is k bits. We assign a look-up table to $\{l \times k\}$-bit variables as shown in Figure 4 (b). The size of the look-up table is 2^{kl}. After the lowest segment(s_0) is looked up, the next higher segment(s_1) is looked up in the ROM table. The result of the previous look-up is added with

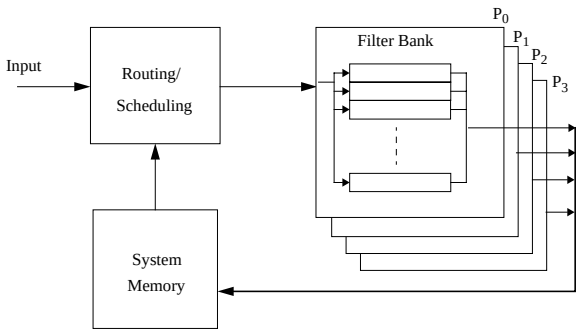

Figure 3: DWT Architecture

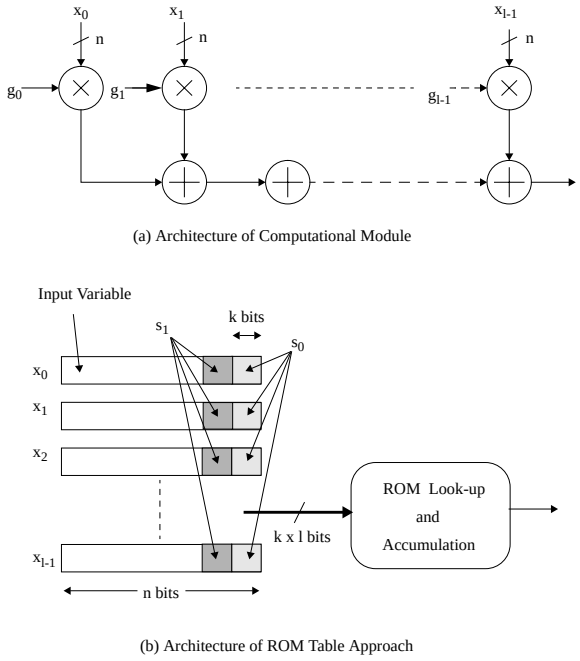

Figure 4: Proposed Architecture for a Filter Bank

the result of the current look-up using the adder. The previous result is shifted right by k bits before the addition. The latency of this architecture is n/k clock cycles. If the length of the input variable increases, we need to increase the number of iterations. This will increase the latency of the computation as well as the size of the input registers. If the filter size increases, the number of input variables increases. As explained in Section 3.1, ROM table can be partitioned into smaller size of ROM tables. However, this increases the number of adders and increases also the latency. Based on the real-time needs, we can choose the size of the ROM table and the number of adders.

Figure 5 illustrates the details of the architecture for $l = 4$, $n = 8$ and $k = 2$. To implement a filter, we need four 8-bit multipliers and three adders. However, with the proposed ROM table approach, we need two 2^4-word

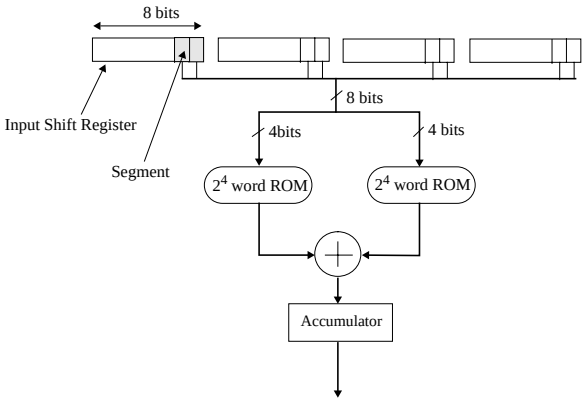

Figure 5: Proposed Architecture for l=4

ROMs, an adder and an accumulator. The output of the accumulator is the DWT coefficient.

The four input variables are stored in four input registers. The shift registers shift the segment to the right by 2 bits every clock cycle. Four 2-bit segments of each input variable play a role on addressing the ROM table. This 8-bit input is partitioned into two 4-bit inputs to look up the ROM table. After looking up the ROM table, the partial results are added in the adder. We iterate this process four times, using the ith segments during the ith iteration. After four clock cycles, DWT coefficient is available in the accumulator.

In computing the wavelet coefficients the filter operations are specified using floating point arithmetic. However, integer arithmetic is used in practice. Thus, the filter coefficients are truncated. This truncation reduces the accuracy of the computed coefficients and hence affects the reconstructed image quality. By using a wide length multiplier, we can reduce the truncation error and can potentially improve the image quality. In traditional hardware-based multipliers, increasing the length of the multiplier can incur significant area and delay penalties. In the memory-based architecture, this is accomplished by increasing the length of the table entries and updating associated logic. This does not significantly affect the overall chip area and offers flexibility in implementation. We are currently evaluating chip size/image quality tradeoffs.

3.3 Overall Architecture

Various VLSI architectures have been developed to realize area-efficient designs for DWT [1, 5]. In realizing high-speed, area-efficient and low-power VLSI implementation, several design parameters must be considered. These are parallel I/O, block-based computation and overlapped computation. Parallel I/O increases the throughput and can be employed to reduce the power consumption of the architecture. In block-based computation, the input image is divided into smaller blocks and processed separately. Block-based computation reduces

the memory requirement considerably. Also, latency is reduced. Overlapped computation of different computation stages and octaves is used in many architectures for computing DWT [1]. Overlapped computation reduces the latency and increases the utilization of the filter modules.

The block diagram of a general architecture for computing the DWT was shown in Figure 3. The architecture consists of three modules: routing/scheduling circuit, filter banks and memory. The routing/scheduling circuit schedules the input to the filter banks. This input consists of external inputs and the data forwarded from the filter banks. The design of the routing/scheduling circuit becomes complex if block-based computation and overlapped computation are incorporated. Data format converters can be used for the design of the routing circuit [2]. The number of filter banks depends on the number of parallel inputs. If block-based I/O is employed, the area of an on-chip memory becomes very small. In [1, 3], the architecture employs a register buffer of size $(l-1)(jp+2w)+4w^2$ bytes and an on-chip SRAM of size $j(l-1)(2N+w)$ bytes, where l is the filter length, $N \times N$ is the input image size, $w \times w$ is the block size, j is the number of octaves and p is the number of parallel inputs. Each variable was assumed to be 1 byte. The number of filter banks increases linearly as the number of inputs increases. The area occupied by the filter banks becomes a dominant part of the entire chip area.

The proposed memory-based filter module can be used for the design of filter banks shown in Figure 3. The architecture using memory-based filter module has lower latency and smaller area compared with the one using the traditional hardware filter banks. We estimated the area based on the architecture proposed in [1] The architecture employs the block-based I/O, the parallel I/O and the overlapped computation. The layouts were obtained using 0.6μ CMOS standard cell design(double metal layer) using COMPASS logic design tools. We used four-tap filters ($l=4$) and each input variable was 8 bits. The architecture using the memory-based filter banks reduced the core size by $30 \sim 40$ % (See Table 1). The latency of the memory-based filter banks was about 60% of that of hardware filter banks. Due to space limitations, the details of the area-estimation and simulation results have been deleted. Details can be found in the full version of the paper [9].

Type of Implementation	Frame Size ($N \times N$)	
	512×512	1024×1024
F_1 $(p=8)$	$3.244\ mm^2$	$3.667\ mm^2$
F_2 $(p=8)$	$1.695\ mm^2$	$2.066\ mm^2$
F_1 $(p=4)$	$2.271\ mm^2$	$2.567\ mm^2$
F_2 $(p=4)$	$1.187\ mm^2$	$1.446\ mm^2$

Table 1: Comparison of the total layout area of hardware filter(F_1) and memory-based filter(F_2) implementations of DWT.

4 Conclusion

We proposed memory-based VLSI architectures for computing the DWT. We implemented the multipliers of the filters by ROM look-up tables. Compared with the architectures employing hardware computation module, our architectures reduce the chip size and the latency. The reduced area can be used to improve the precision of computation and the image quality by increasing the length of the ROM table entries. Our preliminary area estimation shows that the proposed architectures reduce the overall chip area by about 30% and reduce the latency by up to 60%. Currently, we are performing simulation studies to evaluate the image quality and the word length of the look-up table to be used.

Recently, wavelets have been studied extensively by the image compression and VLSI image processing communities. Wavelet transforms offer the capability for multi-resolution representation. They have been shown to be competitive with respect to the compression ratio and the image quality compared with DCT and DFT. DWT shows the promise of being useful in many video compression applications in the near future. Our proposed approach offers fast and area-efficient DWT architectures.

References

[1] J. Bae and V. K. Prasanna, "Synthesis of VLSI Architectures for Two-Dimensional Discrete Wavelet Transforms," *IEEE Int. Conf. on Application Specific Array Processors,* July 1995.

[2] J. Bae and V. K. Prasanna, "Synthesis of a Class of Data Format Converters with Specified Delays," *IEEE Int. Conf. on Application Specific Array Processors,* July 1994.

[3] J. Bae, "Fast and Area-Efficient VLSI Architectures for Video Compression Applications", *Ph.D Thesis, University of Southern California,* June 1996.

[4] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. PAMI, vol. 11,* pp. 674-693, 1989.

[5] C. Chakrabarti, M. Vishwanath and R. M. Owen, "A Survey of Architectures for the Discrete and Continuous Wavelet Transforms," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* 1995.

[6] J. Guo, C. Liu and C. Jen, "The Efficient Memory-Based VLSI Array Designs For DFT and DCT," *IEEE Trans. Circuits and Systems - II: Analog and Digital Signal Processing, vol. 39, No. 10,* pp. 723-733, Oct. 1992.

[7] M. Sun, T. Chen and A. M. Gottlieb, "VLSI Implementation of a 16x16 Discrete Cosine Transform," *IEEE Trans. Circuits and Systems, vol. 36, No. 4,* pp. 610-617, April 1989.

[8] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine,* pp. 14-38, Oct 1991.

[9] S. Choi, J. Bae, V. K. Prasanna, "Synthesis of Memory-based VLSI architectures for Discrete Wavelet Transforms," *Manuscript, in preparation.*