Journal of VLSI Signal Processing 35, 155–178, 2003

Discrete Wavelet Transform: Architectures, Design and Performance Issues

MICHAEL WEEKS
Department of Computer Science, Georgia State University, Atlanta, Georgia 30303

MAGDY BAYOUMI
Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, Louisiana 70504

Received July 22, 2002; Revised July 22, 2002; Accepted October 2, 2002

Abstract. Due to the demand for real time wavelet processors in applications such as video compression [1], Internet communications compression [2], object recognition [3], and numerical analysis, many architectures for the Discrete Wavelet Transform (DWT) systems have been proposed. This paper surveys the different approaches to designing DWT architectures. The types of architectures depend on whether the application is 1-D, 2-D, or 3-D, as well as the style of architecture: systolic, semi-systolic, folded, digit-serial, etc. This paper presents an overview and evaluation of the architectures based on the criteria of latency, control, area, memory, and number of multipliers and adders. This paper will give the reader an indication of the advantages and disadvantages of each design.

Keywords: wavelet transforms, computer architecture, digital filters, digital signal processors, discrete transforms

1. Introduction

The Discrete Wavelet Transform (DWT) is discrete in time and scale, meaning that the DWT coefficients may have real (floating-point) values, but the time and scale values used to index these coefficients are integers. A signal is decomposed by DWT into one or more levels of resolution (also called octaves), as shown in Fig. 1, where a 1-dimensional signal is decomposed into 3 octaves. Figure 2 shows a one-dimensional, one-octave DWT. It includes the analysis (wavelet transform) on the left side and the synthesis (inverse wavelet transform) on the right side. The low-pass filter produces the average signal, while the high-pass filter produces the detail signal. In multi-resolution analysis, the average signal at one level is sent to another set of filters (Fig. 1), which produces the average and detail signals at the next octave [4]. The detail signals are kept, but the higher octave averages can be discarded, since they can be re-computed during the inverse transform. Each channel's outputs have only half the input's amount of data (plus a few coefficients due to the filter). Thus, the wavelet representation is approximately the same size as the original. The DWT can be 1-Dimensional, 2-D, 3-D, etc. depending on the signal's dimensions.

The 2-D transform is simply an application of the 1-D DWT in the horizontal and vertical directions [5], at least for the separable case. Figure 3 shows the 2-dimensional (separable) transform for one octave. The non-separable 2-D transform works differently from the one shown, since it computes the transform based on a 2-D sample of the input convolved with a matrix, but the results are the same. The separable idea can be extended to the 3-D DWT, shown in Fig. 4.

The low-pass filter applies a scaling function to a signal, while the high-pass filter applies the wavelet function. The scaling function allows approximation of any given signal with a variable amount of precision [5, 6]. Applying the following difference equation with the scaling function's coefficients, h, gives an approximation of the signal. This is also known as the low-pass output, where W are the scaling coefficients, and j represents the octave, except in the case of $W(0, n)$, which

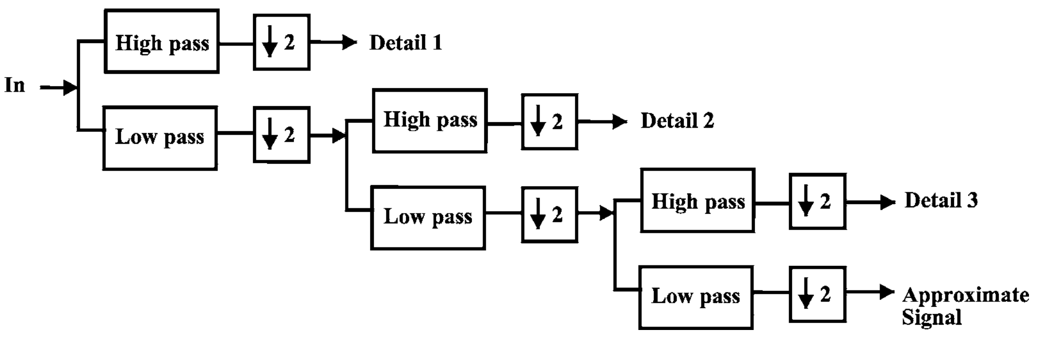


Figure 1. Three octave decomposition of a 1-D signal.

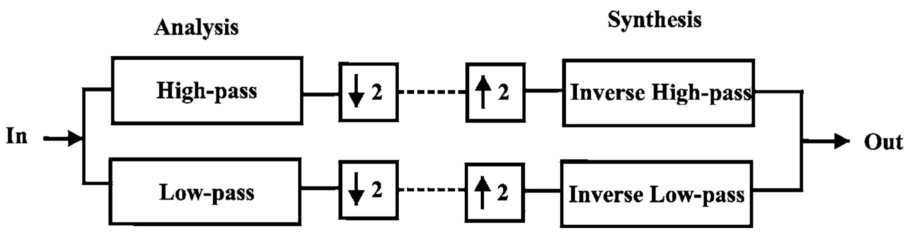


Figure 2. A 1-Dimensional, 1-octave DWT and Inverse DWT.

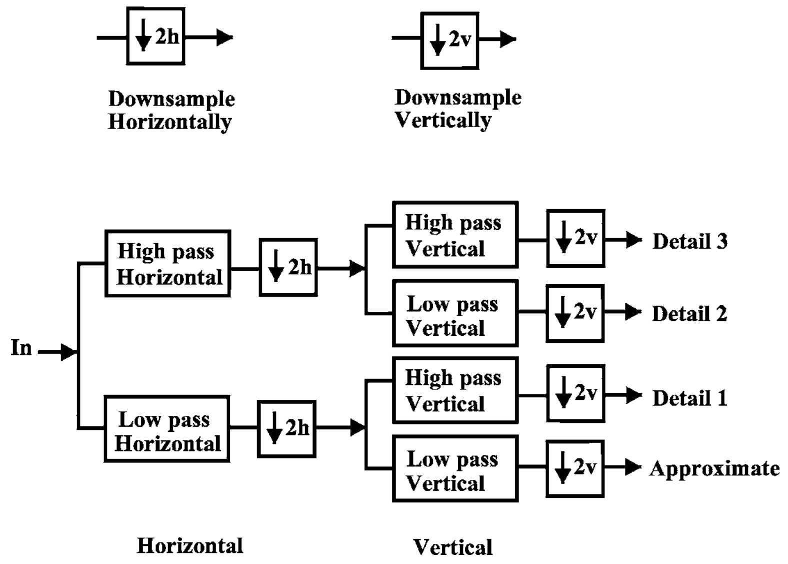


Figure 3. A 2-dimensional, 1-octave DWT.

is the original signal:

$$W(j,n) = \sum_{m=0}^{2n} W(j-1,m)h(2n-m) \qquad (1)$$

Convolution with the wavelet function's coefficients, g, produces the detail signal, also called high-pass output W_h [5–10]:

$$W_h(j,n) = \sum_{m=0}^{2n} W(j-1,m)g(2n-m) \qquad (2)$$

The DWT of a 1-D signal can be computed recursively using a filter pair with the fast pyramid algorithm,

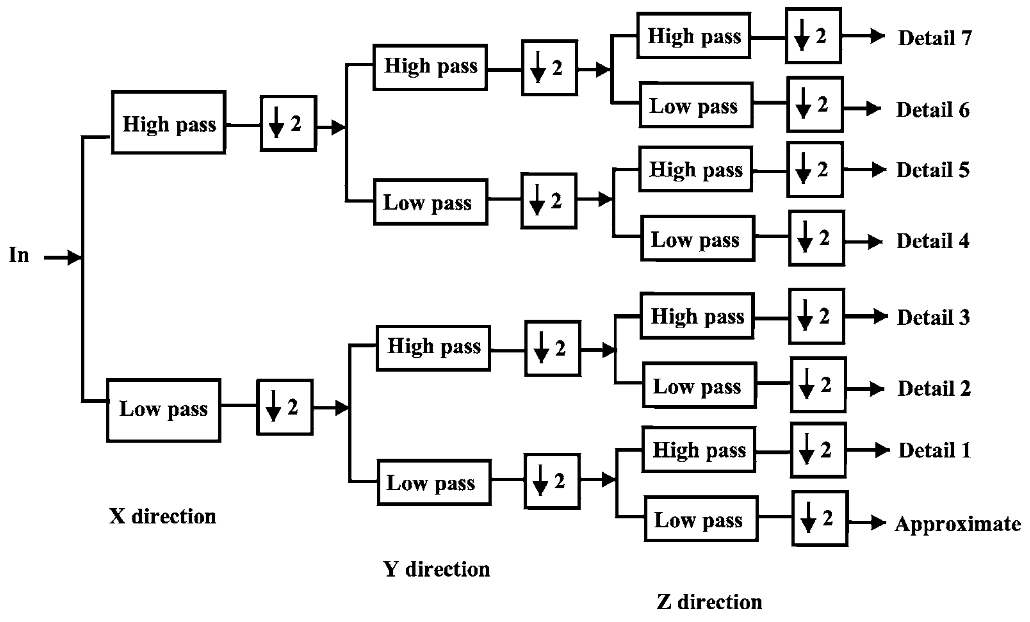


Figure 4. A 3-dimensional, 1-octave DWT.

by Mallat and Meyer [4], Fig. 1. It has a complexity of O(N), with an input of N samples. Other transforms typically require O(N^2) calculations. Even the Fast Fourier Transform takes O($N \log N$) computations. The fast pyramid algorithm gets its efficiency by halving the output data of each channel, otherwise known as down sampling. Since every octave uses half the number of data as the previous octave, the maximum number of octaves, J, can be found by setting 2^J equal to the input length, i.e. $2^J = N$, and the DWT generates approximately $N/2^j$ outputs for each octave j. However, practical applications limit the number of octaves. The number of operations is proportional to $2t(1 - (1/N))$, where t represents the number of operations in the first octave. Therefore, the DWT has a lower bound of $\Omega(N)$ multiplications [11]. Based on the work of [4], Fridman and Manolakos [12] include a schedule for their folded wavelet architecture that directly improves the effectiveness of hardware. In essence, it devotes every other time slot to compute the first octave, allowing the computations of higher octaves to fill in between [13].

Several architectures have been proposed to perform the wavelet transform, mainly: systolic architectures, semi-systolic, space-multiplexed, time-multiplexed, folded, digit-serial, block-based, and non-separable ones. These architectures are analyzed in this paper. Architecture comparisons are also given, based on several design and performance issues: latency, control, area, memory, and the number of multipliers and adders.

This paper is organized as follows. The architectural considerations are analyzed in Section 2, followed by 1-D DWT architectures in Section 3. Section 4 deals with 2-dimensional DWT architectures, and Section 5 covers the 3-D case. Finally, Section 6 summarizes this study and analysis.

2. Architecture Considerations

The performance and cost of DWT architectures are influenced and affected by several design factors, mainly latency, control, area, memory/storage, and precision. Their impact varies based on the application, wavelet coefficients, and type of architecture. These architecture considerations are discussed in this section.

2.1. *Design Issues*

Filters, the main computational kernels in DWT computation, can be implemented either in serial or parallel. The basic serial (systolic or semi-systolic) filter design uses L multiply and accumulate cells (MACs), accommodating L wavelet coefficients (L is the width of the filter). During a single time step, each MAC performs one multiplication and one addition, so $1/L$ of an answer is generated at a time. Since the serial filter is pipelined, it works on L calculations at the same time. In a parallel filter, the inputs come to the multipliers simultaneously. The parallel filter has L multipliers, one for each wavelet coefficient. The multiplications are done in parallel, then their outputs are fed to a tree of $L - 1$ adders to compute the results in the shortest time [8]. The adder tree has a latency of: (time needed for one addition) $\times \log(L)$. Adding the time needed for a multiplication to this gives the filter latency. Another design question is "how many wavelet coefficients should be used?" This is a parameter of the application, which is typically between 4 and 12 filter taps [12]. The size and type of the wavelet filters do not matter to the architectures described here, since most of them are scalable. Both the low pass and high pass computations can be done either by doubling the multiply-accumulate hardware, or by doubling the clock frequency and having the hardware do two computations per original clock period; i.e. the computation can be multiplexed in space or time [13–16]. This design choice has the obvious tradeoff of speed versus area savings.

In the DWT computation, there are not exactly $N/2$ filter outputs for N inputs, because the filter implementation lengthens the outputs by the number of the delay stages within the filter [17]. There are two ways to deal with this problem. The first is zero padding, where any value after the last input is assumed to be 0. This adds a few extra outputs, equaling the number of delay stages, but this is tiny compared to the input size. The second solution assumes that the input is circular. Circular input means that the input starts over at the beginning when it reaches the end. For 1-D data, imagine a circle. For 2-D, a torus describes the input. Both solutions allow perfect reconstruction. Zero padding is easier to implement in terms of routing. Some applications, such as the FBI fingerprint compression project [18], only need one specific wavelet filter on the wavelet analysis/synthesis chip. Therefore, the programmable filters can be replaced by fixed filters, which reduce the area required. Programmability gives the user flexibility, however. Most DWT chips allow coefficients to be loaded.

Most architectures for the DWT are based on fixed-point number representation. One can think of the fixed-point number as an integer with a scale factor, similar to a radix point. Treating the data values as integers greatly reduces the complexity of the hardware design, makes performing multiplications faster, and requires less silicon area. Floating-point allows a greater range of numbers, but the wavelet transform does not need the floating-point range, due to compact support and small coefficients [19, 20].

The Inverse Discrete Wavelet Transform (IDWT) architectures come in two forms. One assumes that all of the DWT outputs are stored in memory (the off-line case), while the second assumes that the DWT analysis happens right before the IDWT synthesis (the on-line case) [21]. Latency concerns are important in both cases. The latter case requires a buffer of size 2^{J+1} between the DWT and IDWT, where J is the number of octaves. Due to the similarity between the DWT and IDWT, the same hardware can compute both functions, with a few modifications. Therefore, most designs concentrate only on the DWT. Some designs, such as the Wavelet Transform Processor (WTP) by Aware [22], have both the DWT and IDWT built in.

2.2. *Precision*

The input data and output data precision are important to a signal processing architecture design. Often, video applications assume an input precision of 8 bits, corresponding to 256 shades in the greyscale. However, for other applications, the input precision could be 12 bits or more corresponding to the input sampling hardware, e.g. an analog to digital converter.

The intermediate coefficients are larger than the original data, since the intermediate coefficients consist of a summation of terms multiplied by the wavelet. The rate of growth depends on the wavelet, but typically this rate does not exceed a factor of 2 [20, 23]. To compute lower octaves, the intermediate coefficients are again passed through filters, which increase the range for each octave computed. Most designs use fixed-point representation. For example, a data value of 8 bits will need a 9th bit after the 1st octave computation, then a 10th bit for the 2nd octave computation, and so on. Every time an intermediate coefficient passes through a filter, its range grows. Such bit growth is applicable to multi-dimensional wavelet transforms, too. For the multi-dimensional architectures introduced in Sections 4 and 5, the internal multipliers and adders will need to handle data as wide as the outputs. The signal-to-noise ratio (SNR) measures the difference between the floating-point DWT coefficients and the ones rounded off due to finite precision.

2.3. Control

There are several types of control: centralized, flow, and pre-stored. A centralized controller generates the signals in one spot and distributes them to the other components directly. Flow control sends the control signals (tags) from processing element (PE) to PE along with the partially computed data. A designer might also build the control into the PE's, called pre-stored control. Centralized control may be easier to implement, but may impact the scalability and increase the area, so the choice of control is important to a good design [12]. In dealing with architectures, we subjectively classify the control as simple, moderate, or complex.

2.4. Area

Area is always an important consideration. For the DWT, the multiplication hardware, the interconnections, and the storage will each add to the architecture's size. Therefore, an efficient design will minimize these factors. Area is expressed in terms of gate count, μm^2, or λ^2. If no other way is available, it will be given in asymptotic complexity. Along with area, the number of multipliers and adders indicate the size of the hardware. The number of multipliers and adders are affected by several design choices, such as whether serial or parallel filters are used, whether time or space multiplexing is employed, whether or not a lattice design is applied, and to what degree folding is used. The number of octaves and the number of wavelet coefficients has a direct relationship to the number of adders, multipliers, and multiply-accumulate cells.

2.5. Memory

Partial computations and input values must be stored within the hardware. Several types of storage units are available, such as MUX-based, systolic, RAM, reduced storage, and distributed [21]. The MUX-based storage unit has an array of storage cells that form serial-in-parallel-out queues, such as in the pipelined design [16], where one queue is used per octave. Multiplexors determine which cell values are sent to the filters. This approach is regular, but it is not scalable. The semi-systolic storage unit is similar to the MUX-based unit, except that busses and tri-state buffers are used in place of multiplexors. This allows more octaves to be added without changing the multiplexor's size, and it keeps the regularity. The semi-systolic unit has long busses, which is a disadvantage. In the RAM based storage unit, a large RAM replaces all of the storage cells, as shown in [24–26], but this design is not as scalable. The reduced-storage unit uses the forward-backward allocation scheme to move data through the unit, allowing for a minimum of storage cells (registers) [27]. Minimizing storage like this makes control more complex, and decreases regularity and scalability. The distributed architecture has local storage on each processor in the filter, which is analogous to parallel computers [15]. Additionally, each processor needs to have multiplexors and demultiplexors in order to move data to another processor's memory.

3. 1-D Wavelet Architectures

One-dimensional architectures can be classified into many types, the main ones are: space multiplexed, systolic array, time multiplexed, folded, and digit-serial. There are techniques for improving these designs, which include lattice, pipelining/register networking, combined DWT and IDWT, and approximating results. However, each improvement involves a certain trade-off: for example, lattice uses less space at the expense of a slower speed. Examples of each category will be discussed below. Architectures are often designed with applications in mind. For 1-D transforms, applications may include denoising a nuclear magnetic resonance (NMR) signal, compressing seismic information [18], and identifying noisy FM signals [28].

3.1. Space Multiplexed Architectures

One of the most common DWT architectures is the fully pipelined, 1-D DWT architecture of Knowles [16], Fig. 5. It uses a low pass and a high pass filter, which is an example of space multiplexing. Serial-in-parallel-out (SIPO) shift registers store the filter inputs, demonstrating multiplexor-based storage. One queue stores data from the input stream, while the others store data generated from the low pass filters. A design with J octaves will require J shift registers, and the shift registers should be as deep as the longest filter. Because of the scheduling, the contents of each shift register will be used right after it fills up, but before it needs to store another value. The shift registers are connected to a multiplexor, which sends the contents of the correct shift register to the filter pair. Both filters get the same inputs. They perform the convolution in parallel, and

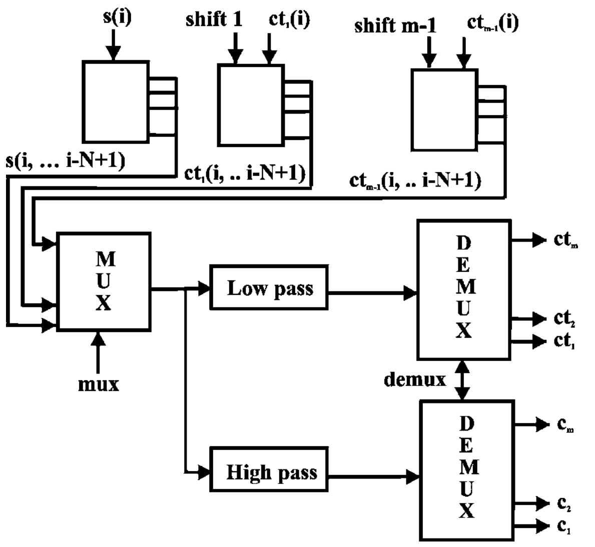


Figure 5. Pipelined architecture [16].

pass along the results. The high pass filter's output goes to a demultiplexor. It decides which channel to send the output to, based on the order of the octave. Another demultiplexor routes the low pass filter's output back to the appropriate shift register. A small control unit generates the signals to run the multiplexor, demultiplexors, and the shift registers. Together, these parts form a 1-D DWT processor [16].

3.2. Systolic Arrays

The main features of systolic arrays are regularity, locality, and scalability. A semi-systolic array has mostly local interconnections, so it retains scalability while allowing more flexibility to the designer, and the PEs can be less complex. Naturally, array architectures have distributed memory and control. This allows the PEs designed for one case to be slightly modified or arranged to solve another case. For example, a designer might want an existing DWT architecture to handle a wider filter length. Expanding a parallel filter for 2 more coefficients is not so easy, since another level of adders will need to be included in the adder tree.

Fridman and Manolakos developed a set of systolic and semi-systolic arrays [13]. The semi-systolic 1-D DWT processing array meets the lowest possible latency, shown in Fig. 6 for the 4-octave DWT. It has an

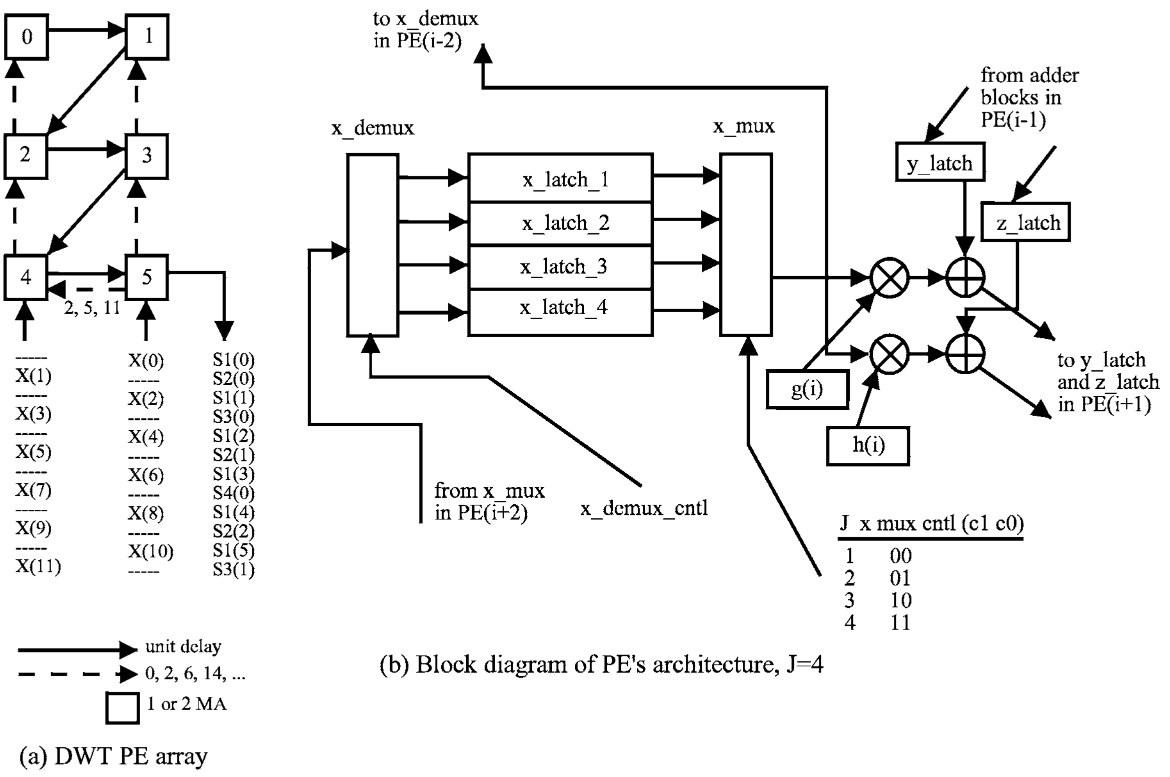


Figure 6. Systolic PE array architecture [13].

efficiency of $1–2^{-J}$, which gets closer to 100% as the number of octaves increase. Each PE needs memory registers, depending on the total number of octaves. This design is simple and modular, with distributed control. Due to the scalability of the design, it translates to 2-D [13], as shown in the architecture of Chen [29]. Another 1-D DWT processing array is described in [30] for three octaves. It has latency of $3N/2$. Three octaves are optimal for this design, with 58% utilization. Having four or more octaves increases the latency, due to data collisions of the scheduling. The PEs of this DWT architecture have 5 memory registers each, 2 registers plus one additional register per octave. The design needs L PEs, where L designates the number of wavelet coefficients. The systolic arrays have low latency and are very efficient. An example of time multiplexed systolic design is shown below.

3.3. Time Multiplexed Architectures

Syed and Bayoumi developed a systolic architecture for the 1-D DWT [14], and noted that only half of the PEs do calculations at a time. The other PEs stay idle due to the downsampling operation. To improve hardware utilization, a register was added for the high-pass coefficient. During their idle time, the PEs will calculate the details. The input signal must be kept for an additional clock cycle, but this change increases the hardware utilization and eliminates the need for a high-pass filter. In effect, this architecture maps the high-pass computations in time instead of space. Figure 7 shows the PE array for a 3-octave DWT. With a slight modification, this architecture can be used to compute the inverse DWT as well. This is a good example of a time multiplexed architecture, where area is roughly half that of a space multiplexed design.

3.4. Folded Architectures

Folding is the process of performing multiple operations with one processor [29]. Regardless of how many octaves of decomposition are required, the 1-D folded design only needs 1 low pass and 1 high pass filter, Fig. 8 [27]. This design shows a high-pass filter on top, with a low-pass filter below. Notice that the high-pass (detail) outputs are simply sent off-chip. The results from the low-pass filter are passed along to registers to the right of the filter, and are periodically sent off-chip. While the filters use the 4 latest inputs every-other clock cycle, the multiplexors send along previous low-pass results during the odd clock cycles. Multiplexors pass on results from the first octave on clock cycle 3, and every 4th clock cycle after that. Similarly, in every clock cycle beginning with the 5th and in increments of 8, the multiplexors send along results from the second octave. Naturally, this example is specific to a 3-octave decomposition. Folding has the disadvantages of long wires for interconnection, and the filters are not used to their full potential. The 3-octave analysis above shows a utilization of 7/8. Folding does have advantages of low latency, as well as a flexible word-length. Categories of folded architectures are serial, parallel, and serial-parallel (for the 2-D case), depending on the manner which the filter requires the inputs. In a folded architecture, a pair of filters is used, and the output from the low-pass filter feeds back to the input. This allows

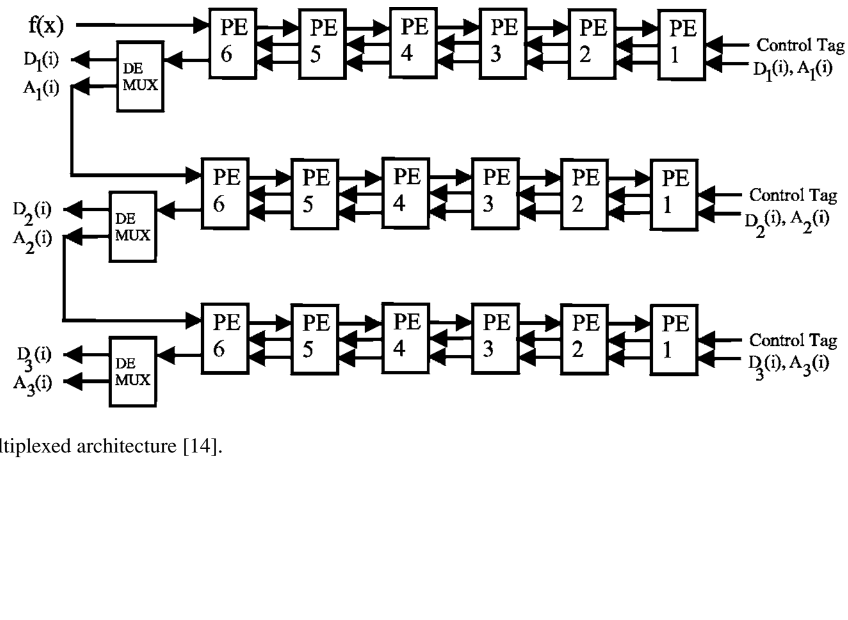


Figure 7. Time multiplexed architecture [14].

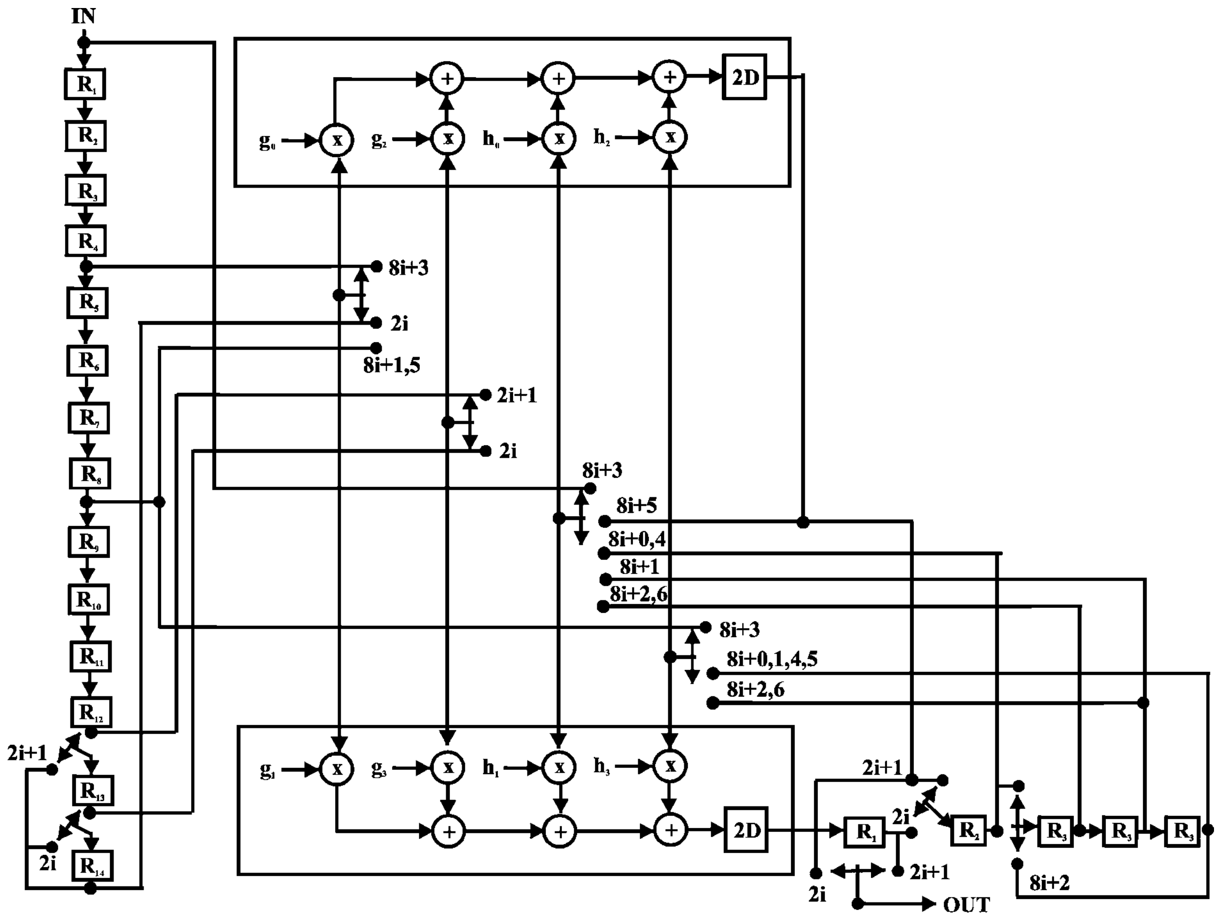


Figure 8. Folded architecture [27].

multiple octaves to be computed by a single pair of filters.

The DWT requires a special folding algorithm, since the octaves have different amounts of calculations. For example, if the first octave does N computations, then the second octave only does $N/2$ computations. Most folding algorithms assume a constant number of calculations, and are thus not appropriate for the DWT. The design of the folding architecture uses "lifetime analysis" to figure out the minimum number of registers which the design should include. The lifetime analysis algorithm is given by Parhi and Nishitani [27].

Yu et al. [31] show an improved 1-D folded architecture by including an additional filter pair. The first filter pair computes the first octave, while the second one works on all lower octaves. This folding modification increases the throughput. These folded architectures have low storage requirements, and are fast and efficient.

3.5. *Digit-Serial Architectures*

The digit-serial design is based on processing a certain number of bits per cycle, known as the digit-size. A digit-serial architecture (also called digit pipelining) uses the input digits one after the other, and the outputs are produced similarly. Digit-serial architectures have a high utilization of hardware, with less routing than the folded design. Digits are smaller than words, i.e. for the first octave, the digits are half the size of the input words. A data converter breaks the input words into 2 digits, but the converter stage increases the latency. The arithmetic components are smaller, but are not as fast, compared to a folded design. Naturally, the design requires a data format converter to reconcile the constant clock period with the varying data rate. Similar to the digit-serial approach are short-length filtering algorithms. Short-length algorithms try to eliminate calculations by taking advantage of calculation redundancy.

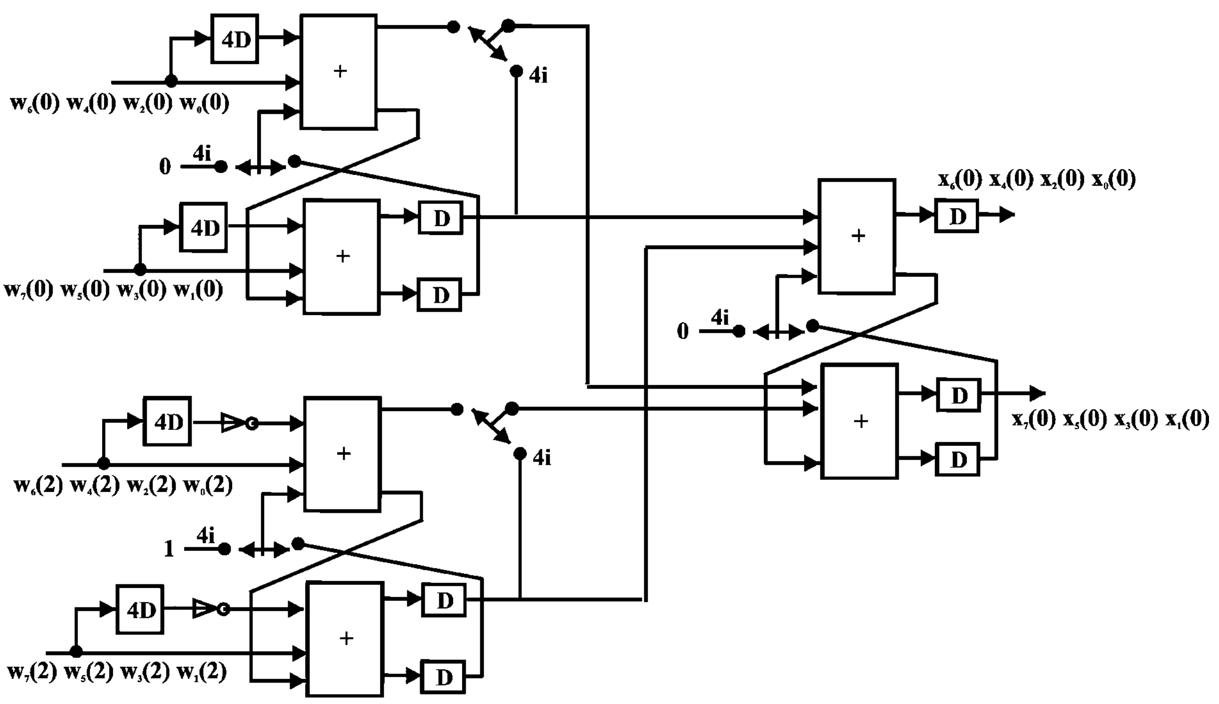

4D
+
4i
$w_6(0)$ $w_4(0)$ $w_2(0)$ $w_0(0)$
0
4i
4D
D
D
$w_7(0)$ $w_5(0)$ $w_3(0)$ $w_1(0)$
$x_6(0)$ $x_4(0)$ $x_2(0)$ $x_0(0)$
D
0
4i
$x_7(0)$ $x_5(0)$ $x_3(0)$ $x_1(0)$
4D
4i
$w_6(2)$ $w_4(2)$ $w_2(2)$ $w_0(2)$
1
4i
4D
$w_7(2)$ $w_5(2)$ $w_3(2)$ $w_1(2)$

Figure 9. Digit-serial architecture [27].

The DWT digit-size varies with the octave j. The jth octave processes *wordlength*/2^j bits at a time. As before, lifetime analysis allows the designer to minimize the number of registers needed in the data converters. Due to the downsampling implicit with the DWT, the designer can view the computations as additions of even and odd parts. Therefore, the architecture can send out the even part of the output followed by the odd part in order to keep the data rate consistent.

The digit-serial architecture [27] computes each octave with different sized processors, Fig. 9, again with a 4-tap filter. A wavelet with more coefficients requires more registers, so the application's needs affect the area and latency of the final design. An assumption is made that there should be 2 pipeline stages within this particular design. Another design consideration deals with handling intermediate results. To juggle them, one should add an output converter unit. For example, an output from octave 1's low pass filter will be used as input to both the low and high pass filters of octave 2. Finally, ripple carry adders in the design affect the overall speed, although Parhi and Nishitani note that the speed would still be practical for video applications [27].

The digit-serial design allows the designer to use local interconnections, which reduces routing. Also, one clock operates the entire design. It utilizes the hardware 100%. The lower octaves need less supply voltage because of the smaller digit sizes, which lowers the overall power consumption. However, the increased amount of processing, i.e. the data conversion, increases the latency. Also, the wordlength must be a multiple of 2^J, where J represents the number of octaves [27]. The digit-serial design is not as flexible as the folded design, but it requires less power.

3.6. Specialized Architectures

There are techniques for making these DWT architectures more efficient in terms of area and time, which include lattice, pipelining, combined DWT and IDWT, and approximating results. As mentioned previously, each improvement involves a trade-off, though the trade-off can be very desirable, like when speed is sacrificed for flexibility. Thus, we present several specialized architectures and note their positive and negative attributes.

3.6.1. Combined DWT and IDWT Architectures. Aware Inc. introduced a wavelet transform processor (WTP) [22], which allows up to 6 coefficients. The user chooses the wavelet coefficients, either specifying

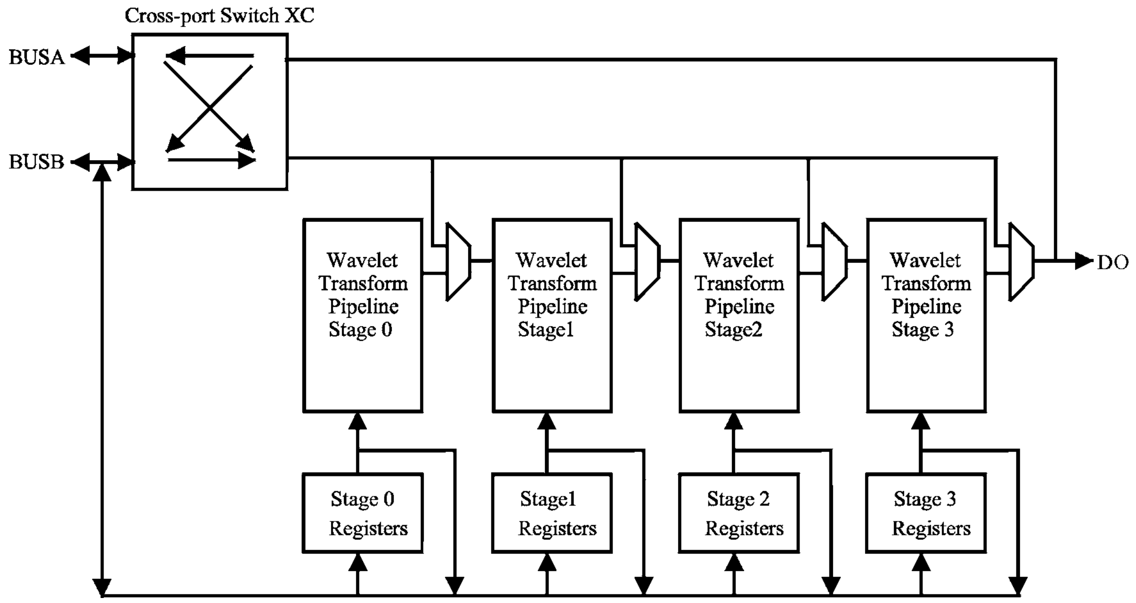

Cross-port Switch XC
BUSA
BUSB
Wavelet Transform Pipeline Stage 0
Wavelet Transform Pipeline Stage1
Wavelet Transform Pipeline Stage2
Wavelet Transform Pipeline Stage 3
DO
Stage 0 Registers
Stage1 Registers
Stage 2 Registers
Stage 3 Registers

Figure 10. The combined DWT/IDWT architecture [22].

the coefficient values or the pre-loaded 6-coefficient Daubechies transform. Input and output data can have a width of 16 bits, and can be fed at a rate of 30 MHz. The chip processes both the DWT and the IDWT. A delay of 9 clock cycles occurs between the time that the first input is received and the time that the first output is complete.

The Aware WTP is pipelined into 4 stages, Fig. 10. Each stage has registers, a multiplier, adder, and shifter. Though the multiplier is size 16×16, only 16 bits are passed on from the multiplier. Software chooses which 16 bits of the result to keep. For example, the most significant bits of the multiplication result might all be 0. Two 16 bit, bi-directional busses allow inputs and outputs to the four stages, depending on the cross-bar switch, while one output bus always supplies the outputs. A designer can cascade a number of these chips to achieve a longer wavelet. The WTP demonstrates how the DWT and IDWT can be combined on a single chip. Of course, combining these on one chip will have an impact on the chip's size, power consumption, and speed.

Sheu, Shieh, and Cheng developed a 1-D architecture that allows both the DWT and the IDWT [32]. Based on the distributed arithmetic approach, the equations of the DWT and IDWT are modified to allow two generalized parts to compute both. Assuming a 4-tap wavelet, a bit-level look up table is used to avoid a multiplication, which speeds up the transform processing.

3.6.2. Lattice Filters. A lattice structure is a variation of the direct DWT implementation, which has only half of the required MACs. It needs complex control and increased routing to allow for fewer multipliers. The filter coefficients must be symmetric [33]. A lattice structure has 2 multipliers and 2 adders and one delay with each stage. The delay elements are clocked with a period of 2T. The lattice structure has been applied to both the folded and the digit-serial designs [34]. Resulting designs have smaller area and lower power dissipation, but also have increased latency. For a good overview of the lattice structure, see [35] and [36].

3.6.3. Register Network, and RAM-Based Architectures. Vishwanath, Owens and Irwin [11] presented three 1-D DWT systolic architectures. The first architecture is similar to the systolic time-multiplexed architecture seen in Section 3, but is presented here for comparison with their other architectures. It cascades linear systolic arrays in a matrix, where each row computes one octave, while each column contains a multiply and accumulate cell (MAC) for each wavelet coefficient, Fig. 11. The input flows from left to right, while the output flows in the opposite direction. One output can be created every two clock cycles, since the cells include downsampling. Due to this timing issue, two overlapped input streams can feed into this architecture, e.g. a practical application has the architecture computing the low and high-pass

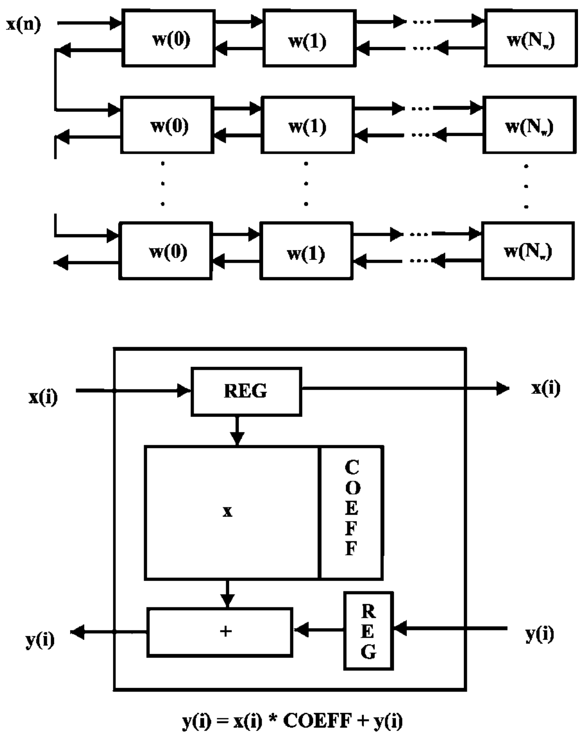


Figure 11. Systolic architecture [11].

outputs for a signal. Time mapping works for the other 2 architectures of Vishwanath et al. [11] as well. This architecture suffers from a large area requirement, and keeps the processors busy only 33% of the time.

The second architecture by these researchers uses a register network to store intermediate data. It improves the processor utilization while reducing the area, Fig. 12. The improved architecture schedules the octave outputs as in [13]. To implement this design, a routing network is used consisting of shift registers. The network has a size of (*number of wavelet coefficients*) × (*number of octaves*). A bound of the area was calculated to be O (*network size* × *precision*). The time needed to find a DWT with this architecture is $2N$ cycles.

Finally, Vishwanath et al. proposes an improved DWT architecture that has minimal area and uses RAM, Fig. 13 [11]. However, the control is very complex. The hardware schedules the inputs on-line, depending on whether the clock cycle is even or odd. Tags specifying the octaves are added to the data, for flow-control. It multiplies data only when the input tag is 1 less than the output tag. Due to the scheduling difference, the basic cell for this architecture had to be

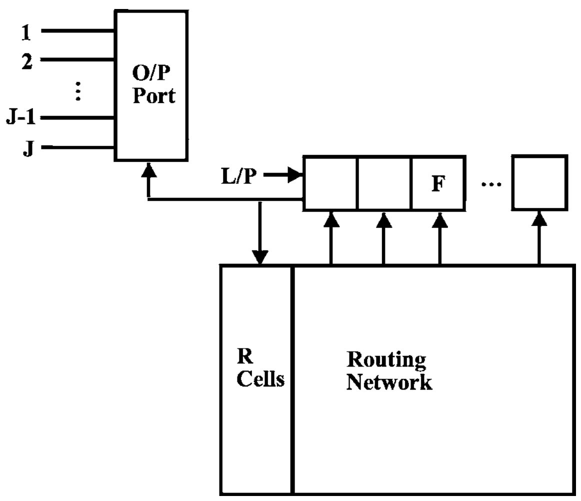


Figure 12. Register network architecture [11].

modified, so it is semi-systolic. The complexity of this design means that the clock cycle must be slower than the clocks in the other architectures.

3.6.4. Approximate 1-D DWT Architecture. Chang et al. present a plan to get faster wavelet transforms at the expense of accuracy. They use fixed-point, but also propose thresholding the calculations. If the product of a data value and a wavelet coefficient will be close to zero, then the multiplication will be skipped, saving time. The two operands are quantized to decide whether or not to perform the multiplication based on the magnitude estimate stored in a lookup table [19].

Another architecture giving a DWT approximate is the design of Lewis and Knowles [37]. They present a VLSI architecture that performs the DWT with 4 tap Daubechies filters. The key advantage of this architecture is that it does not include multipliers, which saves on space while increasing speed. It takes advantage of the wavelet coefficients and the low precision requirements of video applications. Since multiplication or division by powers of 2 can be achieved by shifting the data left or right, some multiplications can be done by shifting and adding. For example, multiplication by 32 is simply a left-shift of 5 bits. The four values used in this wavelet transform are:

$$
\begin{array}{ll}
a = (1 + \text{sqrt}(3))/8 \cong 11/32 & \text{low pass filter:} \\
b = (3 + \text{sqrt}(3))/8 \cong 19/32 & h = (a, b, c, -d) \\
c = (3 - \text{sqrt}(3))/8 \cong 5/32 & \text{high pass filter:} g(n) \\
d = (-1 + \text{sqrt}(3))/8 \cong 3/32 & = (-1)^{n+1} h(3-n)
\end{array}
$$

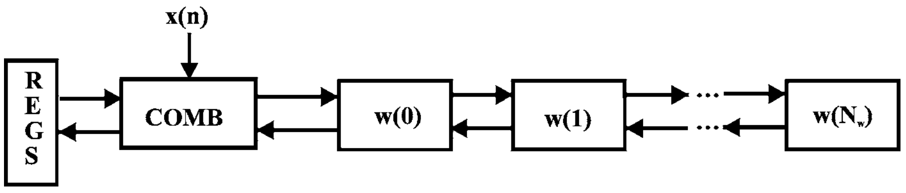


Figure 13. RAM-based architecture [11].

The Lewis and Knowles design [37] presents a good way to speed up the transform (i.e. eliminate actual multiplications) when a fixed wavelet is used. Sheu et al. also eliminated multipliers from their architecture [32]. They use a look-up table, which saves in area. These architectures have the obvious disadvantage of low precision. Also, changing the wavelet coefficients would be difficult if not impossible.

3.6.5. Summary of 1-D Architectures Direct implementation of 1-D DWT tends to be inefficient, and several alternatives have been developed. The pipelined and space-multiplexed architecture is the first discrete wavelet transformer [16]. It is not scalable, it has a large area, complex control and routing, but its latency is O(N). The folded and digit-serial architectures of Parhi [27], Aware [22], and Knowles [16] are fast but not regular. This means that scaling these architectures for more octaves, wider filters, or to higher dimensions is difficult [12]. Vishwanath, et al. developed three architectures, a systolic design, one using a register network to control the filter pair, and the third using RAM [11]. Fridman and Manolakos' architecture has the advantages of distributed memory and control, based on systematic data dependence analysis [12].

A lower latency bound of O(N) is important, it has been achieved by Vishwanath [11], Parhi [27], and Knowles [16]. Fridman's first architecture [12] also meets this lowest possible latency with an efficient semi-systolic 1-D DWT processing array. It has a latency of N or $2N$ depending on space or time multiplexing [12], and it requires $2L$ or L Multiply-Accumulate units, respectively. It can be distinguished by the following observations. Adding more processing elements can lengthen the filters, and simple changes to the PEs allow the architecture to perform more octaves. Most important, the analysis can be used for arrays other than the one specifically shown in [12], such as 2-D designs. Aware's WTP has a latency of O($N \log N$). The designs of Vishwanath et al. [11] have global routing networks, which varies in memory requirements. The systolic architectures are modular and have a lower latency bound of N [11, 14].

The folded architecture is faster, but larger. It has low latency and allows for arbitrary wordlength. The digit-serial architecture uses less power and has simpler interconnections, but the speed and wordlength are constrained. Each architecture requires the same amount of input/output pins. With these features in mind, a project designer can choose which of these architectures best suits the target application.

Chang, Liu and Chan developed a quick algorithm that yields an approximate solution [19]. Lewis and Knowles present a way to calculate the DWT with Daubechies coefficients, without multipliers [37]. These algorithmic variations can be implemented according to the application's demands. For fast, low-resolution video applications, Lewis and Knowles' multiplier-less algorithm would work well. The design of Chang, et al. sacrifices accuracy for speed, while Lewis and Knowles' design is not readily altered for a different wavelet.

For a lossy compression of an on-line source signal, speed will be more important than power consumption or space savings. For example, a chip based on the architecture of [16] can meet the television requirements. If the manufacturers are making several variations for different sized wavelets, perhaps for different television models, then the design of [13] would be very good. If accuracy is not required, for example in compression of a voice signal, the approximate DWT [19] may work well. If the chip will be in a portable unit, such as a telephone, then savings in power dissipation will be the dominant concern, the digit-serial lattice design [34] would be good, and the folded lattice [34] would be good for an even smaller unit. For an integrated unit, space savings would allow other functions to be performed on chip, such as quantization, so a denoising circuit in a sensor might be best suited by [14]. In short, the architecture chosen for an application will depend upon the application's priorities. A comparison of 1-D DWT architectures appears in Table 1.

Table 1. Architectures for the 1-D DWT.

	Type	Latency	Control	Area	Memory	MACs
[8]	Parallel filter	$T_m + T_a \log L$	Complex	O(LJK)	JL shift registers	$2L$ mults, $2(L-1)$ adders
[11]	Systolic	$2N + \log(N)$-4	Simple	O(LK log(N))	$2L(J+3)$ (48 registers)	JL MACs (12 mults, 12 adders)
[11]	Semi-systolic shift-registers	$2N$	Simple/moderate	O(LK log(N))	JL registers	L MACs
[11]	Semi-systolic RAM-based	$2N$	Complex	O(LK)	$L(J+3)$ registers	L MACs
[14]	Systolic	O(N)	Simple/moderate	11.6×7.9 mm^2 per PE	60 registers	18(8b) mults, 21(16b) adders, (18 MACs)
[16]	Pipelined (folded)	O(N)	Simple, centralized	1500 gates	JL registers	$2L$
[19]	Fast approximation	Depends on input data	Central, complex	Small	36 registers + coefficient table	Booth mult (1 adder + circuitry)
[20]	Systolic time multiplexed	O(N)	Simple	10 mm × 7 mm	49 registers (3 octs, $L = 6$)	L
[27]	Folded	O(N), 28 cycles (at 3 octaves)	Complex	More than digit-serial	164 registers	$4(L+1)$ mults, $4L$ adders
[27]	Digit-serial	70 cycles (at 3 octaves)	Simple	Less than folded	258 registers	$4\,\gamma\,(L+1)$ mults $4\,\gamma\,L$ adders
[30]	Semi-systolic PE array	$3N/2 + 1$	Simple, distributed	–	$L(J+2)$ registers	L PEs
[31]	Folded, 2 stages	O(N)	Simple	2974×2868 μm^2, 39275 transistors	JL registers	$2L$ MACs
[22]	Pipelined	O(N log(N)) for 1 octave	External	O(NK)	4 (16b), 4(3b) registers	4 MACs
[32]	No multipliers	O(N)	Simple	$5100 * 5900$ μm^2	61 registers	32 adders
[35]	Folded-lattice	74 cycles (at 3 octaves)	Complex	About half the size of folded	182 registers	$(L+3)$ mults, $2(L+1)$ adds
[34]	Digit-serial-lattice	168 cycles (at 3 octaves)	Simple	About half the size of digit-serial	257 registers	$2\,\gamma\,(L+3)$ mults $2\,\gamma\,(L+1)$ adds
[38]	Frequency domain multiplication	$2\ \log(n1) + 1$	complex	O(KJN$_1$ log(N_1))	–	JN$_1$
[39]	Analog	L (4 taps, 3 octaves)	Simple	About $1500 \times 900\ \lambda^2$	Uses voltage followers	4 quadrant mults, current adders

Note: In the architectures of [34] and [27], the character γ is used, where $\gamma = \Sigma_{i=1..K} 2^{-i}$.
L = filter length, J = number of octaves, K = data precision (i.e. number of bits per input sample), N = input size (for 1-D), M = input size (for 2-D, i.e. rows) (M = N for square images), P = input size (for 3-D, i.e. number of images), T_a = Time to perform 1 addition, T_m = Time to perform 1 multiplication, – = not addressed.

4. 2-D Wavelet Architectures

Architectures for the 2-D DWT include many of the same types as in 1-D case, presented in the previous section, as well as a few new ones. The 2-D folded and semi-systolic architectures are similar to the 1-D versions, while the block-based, and non-separable architectures do not have a 1-D equivalent. A 2-D digit-serial design is also feasible. Applications for the 2-D transform include image compression, and speeding up matrix algebra [18]. Two-dimensional data get the most correlation from the transform when the second dimension is considered. In other words, one could treat 2-D data as 1-D data, and perform the 1-D DWT on it, but the transform will be less effective. The DWT compacts the majority of the signal's energy into the low pass outputs. The resulting approximate signal from a 2-D transform will be 1/4 the size of the original, while the approximate signal from a 1-D transform will only be $1/2$ the size of the original. Therefore, the 2-D transform is more efficient for 2-D data than a 1-D transform, since it compacts the signal's energy (i.e. the approximate signal) into less space.

When the DWT is separable, it means that the 2-D transform is simply an application of the 1-D DWT in the horizontal and vertical directions. In other words, filtering in both the horizontal and vertical directions performs the separable 2-D discrete wavelet transform. The horizontal (row) inputs flow to one high-pass filter and one low-pass filter. The filter outputs are downsampled by 2, meaning that every other output is discarded. At this point, a 1-D transform would be complete for one octave. The 2-D transform sends each of these outputs to another high-pass and low-pass filter pair that operates along the columns. Outputs from these filters are again downsampled. One octave of the 2-D transform results in four signals, with each of the four signals only one-fourth the size of the original input. The algorithm sends the low-low output (the signal's approximation) to the next stage in order to compute the next octave. This process repeats for all octaves. Separable 2-D architectures can be generated from 1-D designs, such as the case of the 2-D architecture of [29], which is based on the 1-D design of [13]. The non-separable 2-D DWT does not do the row and column transforms, but instead generates the 4 outputs of an octave decomposition directly.

4.1. The 2-D Folded Architecture

Folded architectures for the 2-D DWT come in three varieties. The first is serial-parallel, also called systolic-parallel [40], where the computations along the X (horizontal) axis are done with serial filters, while the Y (vertical) axis calculations are done in parallel. The second variety is to use parallel filters along both dimensions [8]. The third one is a direct architecture [40], it uses one filter combined with a multiplexor and RAM to perform all calculations. It is similar to the other folded architectures, but it takes longer to perform the transform since it only uses 1 filter pair.

The serial-parallel architecture has 2 serial horizontal (row) filters, and 2 parallel vertical (column) filters, Fig. 14. A storage unit buffers the data sent to the parallel filters, and a second storage unit buffers the low-low-pass output before it feeds back into one of the serial filters. The first storage unit has an approximate size of $2KN$, where K represents the data precision, while the second storage unit has a size of N. The serial-parallel architecture could be easily redesigned as a fully parallel architecture. The fully parallel architecture replaces the multipliers with programmable multipliers, which eliminates half the multipliers. Both the serial-parallel and fully parallel architectures are scalable.

The processing load does not distribute evenly in a parallel environment. Folding can be used to make the architecture more efficient since it requires only

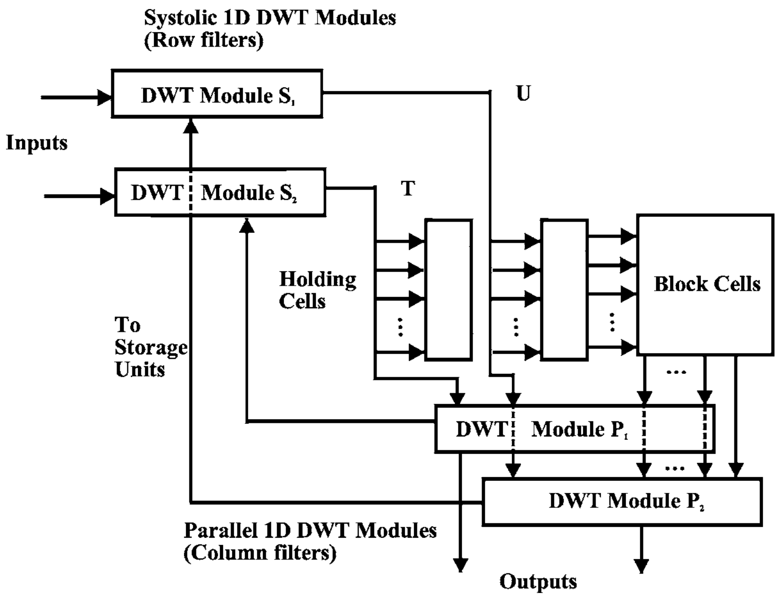


Figure 14. Serial-parallel 2-D architecture [40].

one pair of filters. As in the 1-D case, a folded architecture requires a large storage size. Without register minimization, a 1-D folded architecture needs memory storage of N words, where N is the length of the input. A 2-D folded architecture, operating on $N \times N$ sized data, requires a memory size of N^2 words, since N^2 outputs will need to be stored in the worst case (without minimization). Register minimization is more difficult to perform on a 2-D design, and takes away from the modularity while increasing the complexity. Besides a large storage size, the 2-D folded architecture has a long latency [21].

Vishwanath et al. devised a 2-D algorithm that runs in real-time, and is similar to the fast pyramid algorithm [40]. The 1st octave has a calculation, followed by 4 cycles where the lower octaves are scheduled. This process repeats until completion. The calculations for the IDWT are scheduled similarly, with the final reconstruction getting a time slot every other cycle. The intermediate reconstructions are scheduled in between. Interleaving the octave's calculations in this manner reduce the DWT/IDWT storage requirements and overall latency. When using a vector quantizer, the outputs must be in block form, and the inverse transform must be set up to receive blocks. Putting data in block form adds to the latency, especially in the inverse transform. This architecture has the advantages of minimum latency, minimum buffering, and single chip implementations for an encoder, decoder, and transcoder. The scheme is hierarchical, which means that the image can be accessed at different resolutions. Hierarchical coding applies to progressive transmissions and multipurpose uses. This architecture takes $N^2 + N$ cycles to compute the 2-D DWT or IDWT. It has an area of O(NLk). The 2-D implementation needs additional memory to act as holding cells between the horizontal and vertical filters [5]. The work of Vishwanath et al. [40] addresses the problem of interactive multi-cast over channels with multiple rates, for example, teleconferencing over mixed networks.

4.2. Semi-Systolic Architecture

The Limqueco and Bayoumi sequential architecture [15], shown in Fig. 15, is optimized for a 2 octave, 2-D DWT, where it has 100% utilization. With modifications to the architecture, it can be expanded to include more octaves, but the hardware utilization drops. This architecture handles downsampling with time multiplexing, where an even and an odd coefficient share one processing element. The processing element (PE) alternatively uses the even and odd coefficients, resulting in a need for only half the number of PEs, while improving the throughput by about 50%. A similar design of breaking the inputs into even and odd streams can be found in [41]. Simply adding more PEs to the filter can expand the number of filter coefficients. The filter computes the high and low pass outputs at the same time, which produces 2 outputs every other cycle. From the horizontal filter, the outputs are fed to the vertical filter one per cycle, instead of 2 outputs per every other cycle with nothing in between. Staggering the outputs this way allows the vertical filter to be utilized 100% of the time. The second octave of this architecture has only one filter. It computes both low and high pass outputs, for both horizontal and vertical directions. The second octave uses 2 register banks for intermediate data storage. It requires adding one more

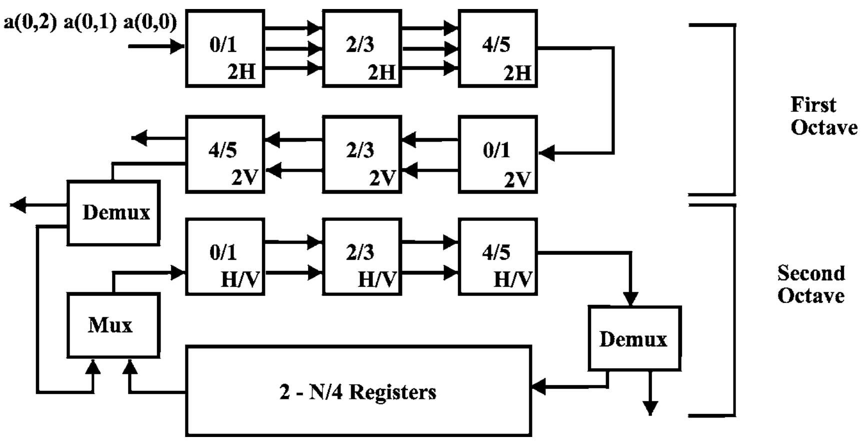


Figure 15. Serial 2-D architecture [15].

filter and removing one register bank to expand this architecture for more than 2 octaves. Though the register bank's memory decreases, memory is added to the PEs. Since the PEs have different amounts of internal storage, this architecture is semi-systolic. A 2-D DWT for 3 octaves has a utilization of 91% for this architecture, but the utilization increases for additional octaves.

4.3. Block-Based 2-D Architectures

The block-based architecture requires that the input signal be available as a block, instead of just a row or column. The block-based architecture needs a data converter, which reduces scalability and increases the routing. The data conversion is necessary between octave decompositions. One example of the block-based 2-D architecture is in [42], Fig. 16, where the C units are data converters, and the F units are filters. The design is similar to the structure used for the 2-D Discrete Cosine Transform (DCT). The transform is "lapped" since the data blocks are overlapped. To perform a transform on an $n \times n$ block, an overlap of $(L - 1)$ samples is needed in both the horizontal and vertical directions. The samples are given to the 1-D filtering module a column at a time. The 1-D processor outputs 2 inner products every other cycle, and these can be shifted to be output one at a time (1 output per cycle). Lifetime analysis can be used with the design to minimize the registers needed in the converters [34]. Other examples of a block-based 2-D architecture can be found in [43] and [44]. Block-based architectures can be very efficient in terms of memory use, or can perform the transform very quickly [23].

4.4. Non-Separable 2-D Architecture

Another version of block-based design is the non-separable architecture for the 2-D DWT. It uses a 2-D parallel-serial filter for each octave's output. This is shown in Fig. 17 [31]. It has a storage size of $N(2L - 1)$. The non-separable approach performs the decomposition without performing the DWT on rows and columns. Instead, the input feeds to 4 filter units, each calculating the high-high, low-high, high-low or low-low pass results. The filter units have L-inputs each, and essentially perform a matrix multiplication to generate the outputs. The design needs no transpose memory between rows and columns, which eliminates some memory and delay. It is efficient for data sent in a parallel manner. The main difference between this architecture and the block-based design is the way it handles inputs as bands (entire rows at a time) instead of blocks.

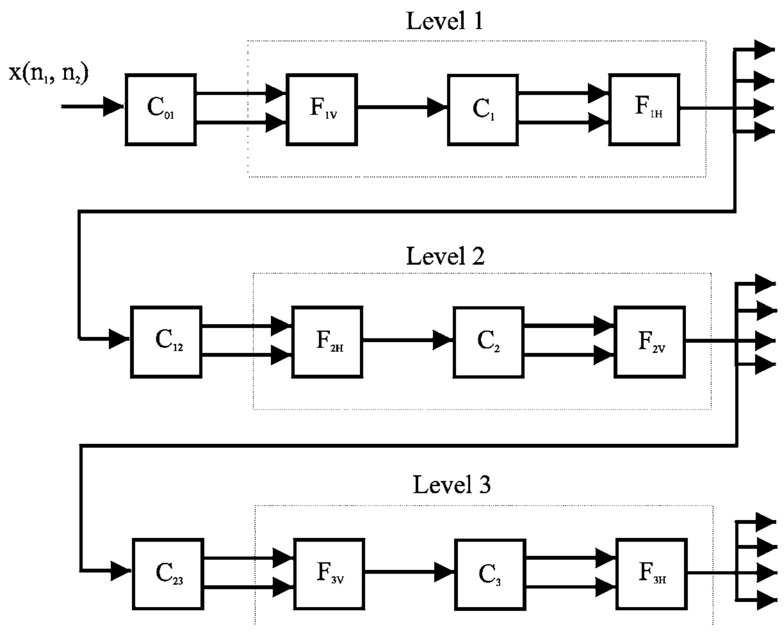


Figure 16. Block-based 2-D architecture [42].

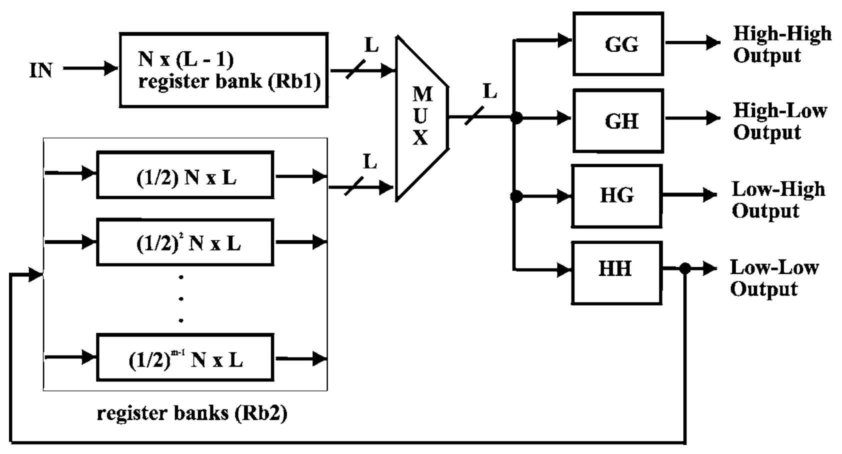


Figure 17. Non-separable 2-D architecture [39].

4.5. Architectural Improvements—The Lattice Structure

Acharya et al. form a 2-D DWT module from 2 1-D modules [45, 46], Fig. 18. A transpose circuit is used to transfer the horizontal (row-wise) outputs to the vertical (column-wise) DWT module. The design is specifically for a 9–7 biorthogonal spline filter, where the low pass filter has 9 coefficients, and the high pass filter has 7 coefficients. Since the coefficients are symmetric, 2 of

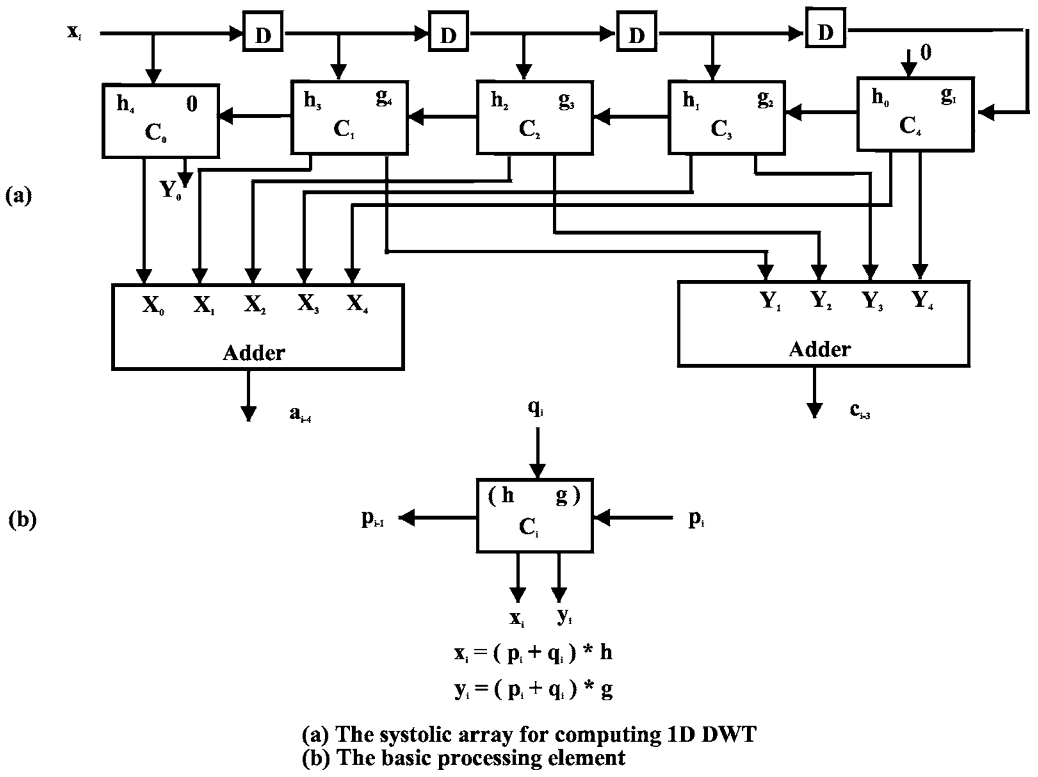


Figure 18. Lattice 2-D architecture [45, 46].

the low pass filter coefficients have the same value, while 3 of the high pass filter coefficients share the same value. Thus, only 5 and 4 coefficients need to be specified to the architecture.

The first few cycles allow the inputs to load. On the following clock cycles, it generates a high or a low pass filter output. On the trailing edge of an even clock cycle, a high pass filter output becomes available. The low pass filter outputs come on the trailing edge of odd clock cycles. Since the architecture generates an output for every clock cycle, the architecture is 100% utilized [45]. This architecture is specific for the biorthogonal spline wavelet, so it is not flexible.

4.6. Summarizing 2-D Architectures

The 2-D folded architectures come in three varieties. The direct one is the simplest, but gives lowest performance. The serial-parallel and the fully parallel designs can be thought of as variations of the direct architecture, with filter pairs for the vertical as well as the horizontal dimensions. The direct method has O(N^2) memory units, while the serial-parallel design uses O(*NL*) holding cells. The fully parallel design has L^2 programmable multipliers, with $L^2 - 1$ adders. It has a memory of O(*JLN*) shift registers. The fully parallel architecture is the fastest of the three folded designs. The 2-D lattice design in [45] is specific for a biorthogonal spline filter. It takes advantage of the coefficients to minimize the operations needed to generate the results.

In the semi-systolic serial architecture of [15], each PE has a different amount of memory that makes scalability a problem. The architecture is fast, efficient, moderately modular, has localized wiring, but large memory is required. High- and low-pass filter calculations are done simultaneously, with 16 bit multipliers.

Most single-chip signal processing implementations are done with block-based architectures, since they show the most promise. But these designs assume that an arbitrary input pattern is available, which will not be a realistic assumption for all applications. Like block-based, the non-separable 2-D DWT architecture presented in [31] has some attractive features. However, it is not conclusively better than the separable designs. It has $2L$ MACs, $2L - 1$ adders, and $N(2L - 1)$ registers, with a delay of $N^2 + N$. In comparison, the fully parallel architecture has about N^2 delay, while the serial-parallel design has a $N^2 + N$ delay. Table 2 compares the 2-D DWT architectures.

Applications for the 2-D DWT include image processing [47], 2-D signal compression, and fingerprint storage [48]. Potential markets include on-line video compression and decompression (codec) systems. For a stand-alone system, the folded architecture [8, 21] would work well, especially the parallel-parallel model. A simpler model, using only 2 octaves of decomposition, could take advantage of the semi-systolic architecture of [15]. A portable device with a smaller screen would need a less power consuming processor, where the block-based architecture would be suited [42]. When bandwidth is not a constraint, perhaps between equipment sharing a dedicated line, the non-separable design would allow parallel computation of the 2-D DWT [31]. As in the previous section, the architecture chosen for an application will depend upon the application's priorities.

Table 2. Architectures for the 2-D DWT.

	Type	Latency	Control	Area	Memory	MACs
[5]	Systolic	$N^2 + N$	–	O(NLK)	2NL	$2L$ MACs, $4L$ mults, $4L$ adders
[8]	Non-separable	$T_m + 2T_a \log(L)$ O(N^2)	Complex	O(NLK)	NL registers adders	$2L^2$ mults $2(L^2 - 1)$
[29]	Systolic (see Parhi93)	–	Simple	–	"Y dimension buffer"	18 mults, 18 adders, (12 PE's)
[31]	Parallel-systolic, non-separable	$N^2 + N$	Simple	$7702 \times 8187\ \mu m^2$, 567899 transistors	$N(2L - 1)$	$2L$ MACs, $L - 1$ adders
[37]	No multipliers	–	Local	Small, about 1/8 "normal" filter	–	8 adders
[43]	Semi-systolic time-multiplexed	$N^2 + N$	Simple, distributed	O(3L(4 K gates + memory))	$29N$	$L/2$ PEs
[45]	Time-multiplexed, systolic	–	Simple	–	"Transpose circuit"	–

5. Architectures for the 3-D DWT

Methods to compress a sequence of images have been proposed using both the 2-D DWT [49] and the 3-D DWT [50]. But three-dimensional data, such as that produced by medical applications, get best results from a true 3-D transform [51]. Video compression, magnetic resonance imaging (MRI) compression [52] and noise reduction between frames of a video sequence are applications for the 3-D transform [18]. Weeks and Bayoumi developed 2 architectures for the 3-D DWT [53, 54]. The first architecture, 3DW-I, is a folded design where a single filter pair performs the calculations for one dimension. The second architecture, 3DW-II, is a block-based architecture. The 3DW-I design does fewer calculations, but the 3DW-II design is smaller and can run in parallel. These architectures are detailed in the next two sections.

5.1. Folded 3-D Architecture

Figure 19 shows how the conceptual model of the 3-D could be implemented directly, assuming the filters are folded and space-multiplexed. This design is the 3DW-I architecture. It uses semi-systolic filters, each containing L_i Multiply-Accumulate Cells (MACs). Here, L_i stands for the number of filter taps in the ith dimension. Each MAC has a number of shift registers, dependent upon the dimension. For each dimension, the number of computations stays the same. The number of implemented filters doubles due to branching, but the amount of data passing through the filters is cut in half by the downsampling operation. The data are used by every other MAC, which eliminates the need for explicit downsampling. The data are sent from the even MACs to other even MACs, while the odd MACs send data to the next odd MAC, similar to the folded 1-D filter [13]. Doubling the amount of registers in each MAC allows it to alternate between two data streams, which simulations confirmed.

The 3DW-I architecture's folded design allows scalability to a longer wavelet. It has simple, distributed control, since each processing element has few functions: shift inputs, multiply and add. The MACs have few internal registers, and any two MACs in a filter have the same amount of registers. The 3DW-I is cascadable, like other folded designs. Finally, the semi-systolic filters generate results in a low number of clock cycles, relative to the data size.

5.2. Block-Based 3-D DWT Architecture

In the second 3-D DWT architecture, 3DW-II, the data is asserted as blocks instead of the row-column fashion, eliminating the need for large on-chip memory. Therefore, the 3DW-II processor needs a data block of size $L_1 \times L_2 \times L_3$ to compute the 8 output streams of the 3-D DWT. This architecture processes data blocks from along the X dimension, Y dimension, then the Z dimension. The blocks are read, skipping every other block horizontally, vertically, and between images in order to take downsampling into account.

The 3DW-II architecture is shown in Fig. 20. The control unit will be more complex for this architecture than the prestored control of the 3DW-I. This control unit will be directly responsible for selecting

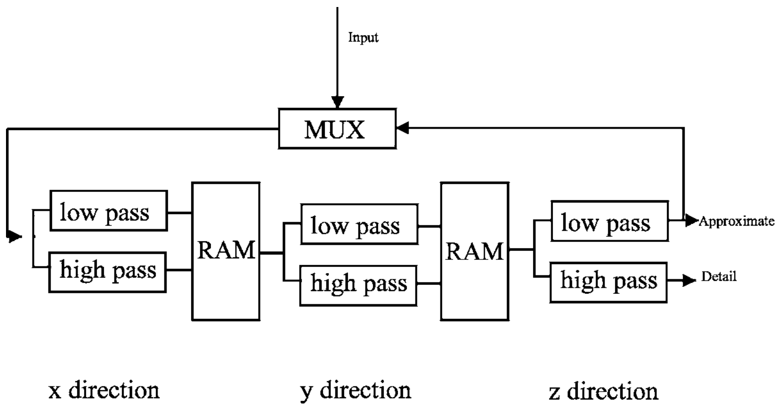


Figure 19. Folded 3-D DWT architecture [54].

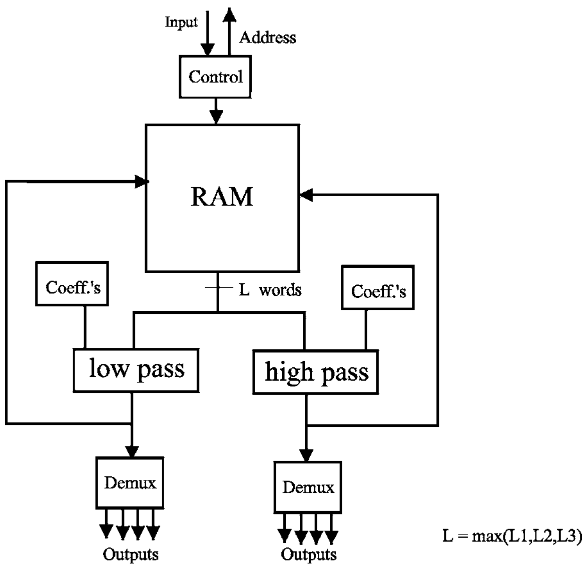


Figure 20. Block-based 3-D DWT architecture [54].

the input block from off-chip. The control unit generates the addresses needed to get the correct input block. The memory on the chip will be small compared to the input size [55]. The amount of storage needed depends solely on the filter sizes, that is, it will be $L_1 \times L_2 \times L_3 + 2 \times L_2 \times L_3 + 2 \times 2 \times L_3 + 8$. The last 8 values do not need to be stored on the chip; they are outputs. The design needs $1 \times L_2 \times L_3$ cycles to do the X calculations, followed by $2 \times L_3$ cycles to do the Y calculations, followed by 4×1 cycles for the Z calculations. Note that the last 4 cycles produce 2 outputs each. This results in $(L_2 + 2) \times L_3 + 4$ cycles per every 8 outputs.

The filters are parallel, since this produces the computation result with the least latency. In contrast, systolic filters assume that the data is fed in a non-block form such that partial calculations are done. The 3DW-I's (semi-) systolic filter is very efficient, but assumes partial calculations that result in large memory requirements. In the 3DW-II, calculations are done on one data block as an atomic operation. This means that the 3DW-II is not as efficient as the 3DW-I. Rather than store partial results for later calculations, the 3DW-II will re-compute them as needed.

5.3. Summarizing 3-D Architectures

The 3DW-I architecture is a straightforward implementation of the 3-D DWT. It allows even distribution of the processing load onto 3 sets of filters, with each set doing the calculations for one dimension. The filters are easily scalable to a larger size. The control for this design is very simple and distributed, since the data are operated on in a row-column-slice fashion. The design is cascadable, meaning that the approximate signal can be fed back directly to the input to generate more octaves of resolution. Scheduling every other time slot to do a lower octave computation works for this design. The filters are folded, allowing the multiple filters in the Y and Z dimensions to map onto a single pair for each dimension. Due to pipelining, all filters are fully utilized, except for the start up and wind-down times.

In the 3DW-I architecture, the amount of memory between filters is a concern. To get started in the Y direction, the X direction must generate enough results, one row of outputs for every wavelet coefficient in the Y dimension. Similarly, the Z filters must wait on the Y dimension filters to finish enough outputs for multiple images. Therefore, the 3DW-I needs large internal storage space.

The 3DW-II architecture has a single low/high filter pair to compute all the outputs. It requires a small amount of storage, based on the filter size, $O(L_1 \times L_2 \times L_3)$, and not the input size. The filter sizes are much smaller than the input size ($L_i \ll NMP$). The latency is small, it depends on the filter sizes and not the input size. For example, using a $4 \times 4 \times 2$ wavelet, the first output comes at the 12th clock cycle. The architecture can be used in parallel to transform the data in half the time (or less, if more than 2 are used). Complex control and the large number of clock cycles are the major drawbacks. Though the 3DW-II uses many clock cycles for a large wavelet on a large data volume, putting multiple chips in parallel will allow each chip to run at a fraction of the speed that would be required of one chip. Table 3 compares the two architectures. For fixed size data, the 3DW-I will complete the transform faster. But for variable data sizes, or when the first results are needed right away, the 3DW-II is the better choice.

Potential applications for the 3-D DWT architectures include television processing and MRI compression. The 3DW-I would be good for television, since the image size will be consistent, speed is important, and space is less of a concern. The 3DW-II works well with MRI data, or any 3-D data that can be buffered into blocks. It can run in parallel, allowing even greater speed than the 3DW-I can provide.

Table 3. Architectures for the 3-D DWT.

	Type	Latency	Control	Area	Memory	MACs
[52]	Folded, semi-systolic	O(NMP)	Simple, distributed	O(NMP)	2NMP + 2MN + $2(L_1 + L_2 + L_3)$	$2(L_1 + L_2 + L_3)$ MACs
[52]	Block-based, parallel multipliers	$L_2L_3 + 2L_3 + 4$	Complex, centralized	$O(L)$	$L_1L_2L_3 + L_2L_3 + 4L_3$	2 max(L_1, L_2, L_3) mults, 2 max(L_1, L_2, L_3)-2 adders

6. Conclusions

The Discrete Wavelet Transform presents an interesting problem for hardware designers. Many researchers have proposed methods for the 1-D and 2-D cases, as well as the 3-D case. Each architecture has advantages and disadvantages compared to the others. This paper gives an overview of several architectures and compares their performance.

Architectures for the DWT include the same computational blocks. First, finite impulse response filters are used. Most designs are scalable, so switching to a different wavelet is not a concern. Digital designs are the most common. Folding is the dominant design choice, since it is flexible and allows an architecture to do more without adding much area. In other words, doubling the amount of calculations does not mean doubling the area needed. Though Mallat's pyramid algorithm is the basic algorithm used [4], Fridman's work optimizes the algorithm with scheduling for folded architectures [30].

Options include the type of filter (serial versus parallel). Also, time versus space mapping allows the designer to trade off between area and latency. The target application will influence the architecture's design to an extent. For example, when speed is a critical factor, space mapping will give the best speed performance.

Tables 1, 2 and 3 give more information about the selected architectures examined in this paper, as well as listing other DWT architectures. These tables include several variables used to indicate architecture parameters. For example, most designs do not require a specific wavelet (Lewis and Knowles' design [37] is one exception). Instead, the designs are modifiable to accommodate any size wavelet. These variables are listed below:

J = number of octaves
K = data precision (i.e. number of bits per input sample)
L = filter length (number of taps)
N = input size (for 1-D)
N_1 = overlapped block size ($N_1 \gg L$)
M = input size (for 2-D, i.e. rows)
P = input size (for 3-D, i.e. number of images)

To demonstrate a design, J typically has the value of 3. The choice of this variable can affect the design, for example, how many registers are needed. The variable K indicates the data precision. This is also known as the data sample width, or number of bits per sample. Typical values of K are 8 and 16.

For a 2-D application, the data has dimensions of NM, though it is reasonable to assume that the image width will equal the image height; that the image is square. Three-dimensional data can be thought of as a sequence of images. While N and M give the dimensions of the images themselves, P specifies the number of images. Also in 3-D designs, the wavelets used for each dimension do not need to be the same. Thus, L_1, L_2, and L_3 are used to denote the filter lengths of the 3 wavelets used.

The Discrete Wavelet Transform can be used on 1, 2, and 3-dimensional signals. The DWT represents an input signal as one approximate signal and a number of detail signals. The representing signals combined together need no more storage space than the original signal. These signals can be sent to the synthesis procedure, to recreate the original signal without loss of information, assuming that no lossy compression is performed. Alternately, the analysis output signals can be compressed, stored, and uncompressed before being sent to the synthesis procedure. This allows the signal to be stored with little loss of information.

The current state of architectures for the Discrete Wavelet Transform typically use finite impulse response filters in parallel filter, systolic, semi-systolic, folded, and digit-serial configurations. Their scalable design makes it easy to modify them to different filter structures according to application demands, such as additional filter coefficients. Tradeoff between area and latency determines the architectural structures: serial versus parallel or time versus space mapping. For time critical applications, space mapping provides a faster transform. The architectures can handle a wide range of applications of 1, 2, and 3-D DWTs.

Acknowledgments

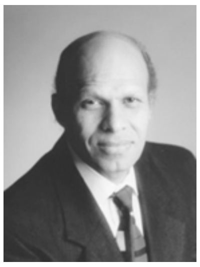
The authors would like to acknowledge the support from contract LEQSF (1996-99) RD-B-13.

References

1. J. Ozer, "New Compression Codec Promises Rates Close to MPEG," *CD-ROM Professional*, 1995, p. 24.
2. A. Hickman, J. Morris, C. Levin, S. Rupley, and D. Willmott, "Web Acceleration," *PC Magazine*, June 10, 1997, p. 10.
3. W.W. Boles and Q.M. Tieng, "Recognition of 2-D Objects from the Wavelet Transform Zero-crossing Representation," in *Proceedings SPIE*, vol. 2034, Mathematical Imaging, San Diego, July 11–16, 1993, pp. 104–114.
4. S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, 1989, pp. 674–693.
5. M. Vishwanath and C. Chakrabarti, "A VLSI Architecture for Real-Time Hierarchical Encoding/Decoding of Video using the Wavelet Transform," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '94)*, Adelaide, Australia, vol. 2, April 19–22, 1994, pp. 401–404.
6. M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1995.
7. S. Mallat, "Multifrequency Channel Decompositions of Images and Wavelet Models," *IEEE Transactions of Acoustics, Speech and Signal Processing*, vol. 37, no. 12, 1989, pp. 2091–2110.
8. C. Chakrabarti and M. Vishwanath, "Efficient Realizations of the Discrete and Continuous Wavelet Transforms: From Single Chip Implementations to Mappings on SIMD Array Computers," *IEEE Transactions on Signal Processing*, vol. 43, no. 3, 1995, pp. 759–771.
9. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley, MA: Wellesley-Cambridge Press, 1996.
10. I. Daubechies, *Ten Lectures on Wavelets*, Montpelier, Vermont: Capital City Press, 1992.
11. M. Vishwanath, R.M. Owens, and M.J. Irwin, "Discrete Wavelet Transforms in VLSI," in *Proceedings of the International Conference on Application Specific Array Processors*, Berkeley, Aug. 1–2, 1992, pp. 218–229.
12. J. Fridman and E.S. Manolakos, "Discrete Wavelet Transform: Data Dependence Analysis and Synthesis of Distributed Memory and Control Array Architectures," *IEEE Transactions on Signal Processing*, 1994, pp. 77–81.
13. J. Fridman and E.S. Manolakos, "Distributed Memory and Control VLSI Architectures for the 1-D Discrete Wavelet Transform," in *IEEE Proceedings VLSI Signal Processing VII*, La Jolla, California, Oct. 26–28, 1994, pp. 388–397.
14. S. Syed, M. Bayoumi, and J. Limqueco, "An Integrated Discrete Wavelet Transform Array Architecture," in *Proceedings of the Workshop on Computer Architecture for Machine Perception*, Como, Italy, Sept. 18–20, 1995, pp. 32–36.
15. J. Limqueco and M. Bayoumi, "A 2-D DWT Architecture," in *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, Iowa State University, Ames, Iowa, Aug. 18–21, 1996, pp. 1239–1242.
16. G. Knowles, "VLSI Architecture for the Discrete Wavelet Transform," *Electronics Letters*, vol. 26, no. 15, 1990, pp. 1184–1185.
17. T. Edwards, "Discrete Wavelet Transforms: Theory and Implementation," Technical Report, Stanford University, September 1992.
18. A. Bruce, D. Donoho, and H.-Y. Gao, "Wavelet Analysis," *IEEE Spectrum*, Oct. 1996, pp. 26–35.
19. C.C. Chang, J.-C. Liu, and A.K. Chan, "On the Architectural Support for Fast Wavelet Transform," *SPIE Wavelet Applications IV*, vol. 3078, Orlando, Florida, April 21–25, 1997, pp. 700–707.
20. A. Grzeszczak, M.K. Mandal, S. Panchanathan, and T. Yeap, "VLSI Implementation of Discrete Wavelet Transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, no. 4, 1996, pp. 421–433.
21. C. Chakrabarti, M. Vishwanath, and R. Owens, "Architectures for Wavelet Transforms: A Survey," *Journal of VLSI Signal Processing*, vol. 14, no. 1, 1996, pp. 171–192.
22. *Wavelet Transform Processor Chip User's Guide*, Bedford, MA: Aware, Inc., 1994.
23. M. Weeks, J. Limqueco, and M. Bayoumi, "On Block Architectures for Discrete Wavelet Transform," in *32nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov. 1–4, 1998.
24. M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 42, no. 3, 1994, pp. 673–676.
25. M.-H. Sheu, M.-D. Shieh, and S.-W. Liu, "A VLSI Architecture Design with Lower Hardware Cost and Less Memory for Separable 2-D Discrete Wavelet Transform," *IEEE International Symposium on Circuits and Systems (ISCAS '98)*, vol. 5, Monterey, California, May 31–June 3, 1998, pp. 457–460.
26. M. Schwarzenberg, M. Träber, M. Scholles, and R. Schüffny, "A VLSI Chip for Wavelet Image Compression," in *IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 4, Orlando, Florida, May 30–June 2, 1999, pp. 271–274.
27. K.K. Parhi and T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, 1993, pp. 191–202.
28. A. Teolis, "Identification of Noisy FM Signals using Non-Orthogonal Wavelet Transforms," *SPIE Wavelet Applications IV*, vol. 3078, April 22–24, 1997, pp. 590–601.
29. J. Chen and M. Bayoumi, "A Scalable Systolic Array Architecture for 2-D Discrete Wavelet Transforms," in *Proceedings of IEEE Workshop on VLSI Signal Processing*, vol. III, Osaka, Japan, Oct. 16–18, 1995, pp. 303–312.
30. J. Fridman and E.S. Manolakos, "Calculation of Minimum Number of Registers in 2-D Discrete Wavelet Transforms using Lapped Block Processing," in *International Symposium on Circuits and Systems*, vol. 2303, San Diego, July 24–29, 1994, pp. 91–104.
31. C. Yu, C.-A. Hsieh, and S.-J. Chen, "VLSI Implementation of 2-D Discrete Wavelet Transform for Real-Time Video Signal Processing," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 4, 1997, pp. 1270–1279.
32. M.-H. Sheu, M.-D. Shieh, and S.-F. Cheng, "A Unified VLSI Architecture for Decomposition and Synthesis of Discrete Wavelet Transform," in *Proceedings of the 39th Midwest Symposium on*

Circuits and Systems, Iowa State University, Ames, Iowa, Aug. 18–21, 1996, pp. 113–116.
33. G. Brooks, "Processors for Wavelet Analysis and Synthesis: NIFS and the TI-C80 MVP," in *Proceedings SPIE [Society of Photo-Optical Instrumentation Engineers]*, H.H. Szu (ed.), vol. 2762, Orlando, Florida, April 8–12, 1996, pp. 325–333.
34. T.C. Denk and K.K. Parhi, "Architectures for Lattice Structure Based Orthonormal Discrete Wavelet Transforms," in *Proceedings of the 1994 IEEE International Conference On Application Specific Array Processors*, 1994, pp. 259–270.
35. T.C. Denk and K.K. Parhi, "VLSI Architectures for Lattice Structure Based Orthonormal Discrete Wavelet Transform," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 44, no. 2, 1997, pp. 129–132.
36. J.T. Kim, Y.H. Lee, T. Isshiki, and H. Kunieda, "Scalable VLSI Architectures for Lattice Structure-Based Discrete Wavelet Transform," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 45, no. 8, 1998, pp. 1031–1043.
37. A.S. Lewis and G. Knowles, "VLSI Architecture for 2-D Daubechies Wavelet Transform without Multipliers," *Electronics Letters*, vol. 27, no. 2, 1991, pp. 171–173.
38. S.-K. Aditya, "Fast Algorithm and Architecture for Computation of Discrete Wavelet Transform," Ph.D. Dissertation, University of Southwestern Louisiana, 1996.
39. G. González-Altamirano, A. Diaz-Sanchez, and J. Ramírez-Angulo, "Fast Sampled-Data Wavelet Transform CMOS VLSI Implementation," in *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, Iowa State University, Ames, Iowa, Aug. 18–21, 1996, pp. 101–104.
40. M. Vishwanath, R. Owens, and M. Irwin, "VLSI Architectures for the Discrete Wavelet Transform," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 5, 1995, pp. 305–316.
41. S.-J. Chang, M.H. Lee, and J.-J. Cha, "A Simple Parallel Architecture for Discrete Wavelet Transform," in *IEEE International Symposium on Circuits and Systems (ISCAS '97)*, Hong-Kong, June 9–12, 1997, pp. 2100–2103.
42. T.C. Denk and K.K. Parhi, "Calculation of Minimum Number of Registers in 2-D Discrete Wavelet Transforms using Lapped Block Processing," *International Symposium on Circuits and Systems*, 1994, pp. 77–81.
43. J. Limqueco and M. Bayoumi, "A Scalable Architecture for 2-D Discrete Wavelet Transform," *VLSI Signal Processing IX*, San Francisco, Oct. 30–Nov. 1, 1996, pp. 369–377.
44. S.-K. Paek, H.-K. Jeon, and L.-S. Kim, "Semi-Recursive VLSI Architecture for Two Dimensional Discrete Wavelet Transform," *IEEE International Symposium on Circuits and Systems (ISCAS '98)*, vol. 5, Monterey, California, May 31–June 3, 1998, pp. 469–472.
45. T. Acharya, P.-Y. Chen, and H. Jafarkhani, "A Pipelined Architecture for Adaptive Image Compression using DWT and Spectral Classification," in *Proceedings of the Thirty-First Annual Conference on Information Sciences and Systems*, vol. II, The Johns Hopkins University, Baltimore, Maryland, March 19–21, 1997, pp. 1–17.
46. T. Acharya and P.-Y. Chen, "VLSI Implementation of a DWT Architecture," *IEEE International Symposium on Circuits and Systems (ISCAS '98)*, vol. 2, Monterey, California, May 31–June 3, 1998, pp. 272–275.
47. W.S. Lu and A. Antoniou, "Simultaneous Noise Reduction and Feature Enhancement in Images Using Diamond-Shaped 2-D filter Banks," *IEEE International Symposium on Circuits and Systems (ISCAS '97)*, Hong-Kong, June 9–12, 1997, pp. 737–740
48. C.M. Brislawn, J.N. Bradley, R.J. Onyshczak, and T. Hopper, "The FBI Compression Standard for Digitized Fingerprint Images," in *Proceedings SPIE*, vol. 2847, Denver, Aug. 4–9, 1996, pp. 344–355.
49. A.S. Lewis and G. Knowles, "Image Compression Using the 2-D Wavelet Transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, 1992, pp. 244–250.
50. A.S. Lewis and G. Knowles, "Video Compression using 3-D Wavelet Transforms," *Electronics Letters*, vol. 26, no. 6, 1990, pp. 396–398.
51. J. Wang and H.K. Huang, "Three-Dimensional Medical Image Compression Using a Wavelet Transform with Parallel Computing," *SPIE Imaging Physics*, San Diego, vol. 2431, March 26–April 2, 1995, pp. 16–26.
52. M. Weeks, "Architectures for the 3-D Discrete Wavelet Transform," Ph.D. Dissertation, University of Southwestern Louisiana, 1998
53. M. Weeks and M. Bayoumi, "3-D Discrete Wavelet Transform Architectures," *IEEE International Symposium on Circuits and Systems (ISCAS '98)*, Monterey, California, May 31–June 3, 1998.
54. M. Weeks and M. Bayoumi, "3-D Discrete Wavelet Transform Architectures," in *IEEE Transactions on Signal Processing*, vol. 50, no. 8, Aug. 2002.
55. G. Zhang, M. Talley, W. Badawy, M. Weeks, and M. Bayoumi, "A Low Power Prototype for a 3-D Discrete Wavelet Transform Processor," in *IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 1, Orlando, Florida, May 30–June 2, 1999, pp. 145–148.

Michael Weeks studied at the University of Louisville's Speed Scientific School in the Engineering Math and Computer Science department. He served as vice-president of the local Triangle chapter, as well as president of the school's ACM chapter. He received a Bachelor of Engineering Science degree in 1993, then a Master of Engineering degree in May 1994. He next enrolled at the Center for Advanced Computer Studies at the University of Louisiana at Lafayette, where he was president of the school's IEEE Computer Society. At Louisiana, he received a Master of Science degree in Computer Engineering in December 1996, followed by a Ph.D. in Computer Engineering in May 1998. He is currently an Assistant Professor in the Computer Science Department at Georgia State University, as part of the State of Georgia's Yamacraw program.
mweeks@cs.gsu.edu



Magdy A. Bayoumi is the Director of the Center for Advanced Computer Studies and the Department Head of Computer Science, University of Louisiana (UL) at Lafayette. He is Edmiston Professor of Computer Engineering and Lamson Professor of Computer Science at the University of Louisiana (UL) at Lafayette. He has been a faculty member there since 1985. Dr. Bayoumi received the B.Sc. and M.Sc. degrees in Electrical Engineering from Cairo University, Egypt; M.Sc. degree in Computer Engineering from Washington University, St. Louis; and the Ph.D. degree in Electrical Engineering from the University of Windsor, Canada. Dr. Bayoumi's research interests include VLSI Design Methods and Architectures, Low Power Circuits and Systems, Digital Signal Processing Architectures, Parallel Algorithm Design, Computer Arithmetic, Image and Video Signal Processing, Neural Networks and Wideband Network Architectures.

Dr. Bayoumi is leading a research group of 15 Ph.D. and 10 M.Sc. students in these research areas. He has graduated 15 Ph.D. and about 100 M.Sc. students. He has published over 200 papers in related journals and conferences. He edited, co-edited and co-authored 5 books in his research interest. He was the guest editor of three special issues in VLSI Signal Processing and co-guest editor of a special issue on "Learning on Silicon". Dr. Bayoumi has one patent on "On-Chip Learning". He has given numerous invited lectures and talks nationally and internationally. He has consulted in industry.

Dr. Bayoumi was the vice president for technical activities of the IEEE Circuits and Systems (CAS) Society, where he has served in many editorial, administrative, and leadership capacities. He was elected to the BoG (1996). He is one of the founding members of the VLSI Systems and Applications (VSA) Technical Committee (TC) and was the past chair. He was one of the founding members of the Neural Network TC. He is a member of the Multimedia TC. He has been on the technical program committee for ISCAS for several years (as track chair and co-chair). He has organized several special sessions and workshops at this conference. He was a co-organizer and co-chair of a forum on MEMS in ISCAS'95. He was the publication chair of ISCAS'99 and he is the special session co-chair of ISCAS'02. He is a member of the steering committee of the Midwest Symposium on Circuits and Systems (MWSCAS). He was the general chair of MWSCAS'94, and the special session chair of MWSCAS'93. He has organized many special sessions and has been on the technical program committee of the symposium for several years. He was on a panel on VLSI Education in MWSCAS'95 and a judge for the first student paper contest in MWSCAS'97. He was an associate editor of the Circuits and Devices Magazine, Transaction on VLSI Systems, Transaction on Neural Networks, and Transaction on Circuits and Systems II. He was the general chair of the 1998 Great Lakes Symposium on VLSI. He was the general chair of the VLSI Signal Processing Workshop 2000. He is on the steering committee of the International Conference on Electronics, Circuits, and Systems (ICECS). He represented the CAS Society on the IEEE National Committee on Engineering R&D policy, 1994, the IEEE National Committee on Communication and Information Policy, 1994, and the IEEE National Committee on Energy Policy, 1997.

Dr. Bayoumi serves on the ASSP Technical Committee on VLSI Signal Processing. He was one of the founders of the CS TC on VLSI. He has been a member of the Technical Program of the IEEE VLSI Signal Processing Workshop, the International Conference on Application Specific Array Processors, and the Computer Arithmetic Symposium. He was the general chair of the Workshop on Computer Architecture for Machine Perception, 1993 and he is a member of the Steering Committee of this workshop. Dr. Bayoumi is an Associate Editor of INTEGRATION, the VLSI Journal and the Journal of VLSI Signal Processing Systems. He was an associate editor of the Journal of Circuits, Systems, and Computers. He is a regional editor for the VLSI Design Journal and on the Advisory Board of the Journal on Microelectronics Systems Integration. Dr. Bayoumi served on the Distinguished Visitors Program for the IEEE Computer Society, 1991–1994. He is the faculty advisor for the IEEE Computer student chapter at UL. He won the UL 1988 Researcher of the Year award and the 1993 Distinguished Professor award at UL. He is a fellow IEEE.

Dr. Bayoumi served on the technology panel and advisory board of the US Department of Education project, "Special Education Beyond Year 2010," 1990–1993. He was the vice-president of Acadiana Technology Council. He was on the organizing committee for Acadiana's 3rd Internet Workshop, 1999. He gave the keynote speech in "Acadiana Y2K Workshop," 1999. He is a member of Lafayette Chamber of Commerce where he is a member of the Economic Development, Education, and Tourism Committees. Dr. Bayoumi was a technology columnist and writer of the Lafayette newspaper "Daily Advertiser." He is on the governor's commission for developing comprehensive energy policy for the State of Louisiana.
mbayoumi@cacs.louisiana.edu