A VLSI ARCHITECTURE FOR REAL-TIME HIERARCHICAL ENCODING/DECODING OF VIDEO USING THE WAVELET TRANSFORM

Mohan Vishwanath
Computer Science Lab
Xerox PARC,
Palo Alto, Ca 94304, USA
mohan@parc.xerox.com

Chaitali Chakrabarti *
Dept. of Electrical Engg.,
Arizona State University
Tempe, AZ 85287, USA
chaitali@asu.edu

ABSTRACT

Novel on-line algorithms and architectures for hierarchical coding using the wavelet transform are presented. These algorithms/architectures compute the decomposition-reconstruction cycle of the wavelet transform with minimum latency and buffering for any given blocking factor. A hierarchical scheme which enables cheap video conferencing and/or multicast over a heterogeneous network is also presented. This architecture supports single chip implementations of the encoder, the decoder, and the transcoder for some choices of the wavelet filter and vector quantization schemes.

1. INTRODUCTION

Hierarchical coding of image data enables one to access the image at different quality levels or resolutions. The two main applications of hierarchical coding have been progressive transmission and multiuse environments. While existing standards have dealt with these typical scenarios, there has been very little work done in what is becoming a very important application, namely, **interactive multicast over multiple rate channels**. A typical example (and the target application for this paper) is teleconferencing over a heterogeneous network made up of (say) ATM, IP, ISDN and wireless. Some form of transcoding has to be done at the gateway between these components due to the basic rate mismatch between them. This can be done in a brute-force manner by uncompressing the video stream and re-compressing with a higher compression ratio. This is not only expensive in hardware, but it also introduces a considerable latency. The transcoding should be done in an **on-line** fashion (with minimal latency/buffering) due to the interactive nature of the application and to reduce hardware costs. The aggregate latency between the sender and the receiver(s) of the video stream is the sum of the latency in the Encoding-Transcoding-Decoding (ETD) process and the delay in the network (usually non-deterministic from the applications point of view). In this paper we do not consider the network delay, and concentrate on reducing the latency in the Encoding-Transcoding-Decoding (ETD) process. We propose a flexible on-line scheme for the hierarchical ETD process which can be efficiently implemented in VLSI.

The main contributions of this paper are listed below.

- A flexible scheme for hierarchical encoding/decoding and transcoding in an on-line manner using the Discrete Wavelet Transform (DWT) is proposed.
- A new (on-line) VLSI algorithm/architecture for the DWT/IDWT which does the decomposition-reconstruction cycle with minimum latency and buffering for any given blocking factor is proposed. This is in contrast to previously proposed on-line algorithms [1] [2] [3] which were optimized for the DWT and hence incurred a latency of $\approx \frac{N}{2}$ cycles.

*This work was supported in part by a grant from NSF, no. MIP-9309504.

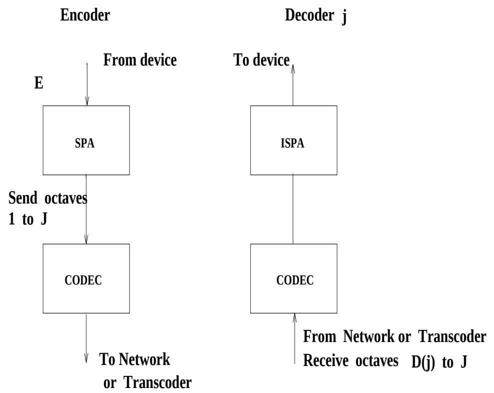

Figure 1. **An Overview of the Hierarchical Coding Scheme.**

2. HIERARCHICAL CODING SCHEME

Some of the goals of the proposed hierarchical coding scheme are (i) the latency of the ETD process should be as small as possible (less than one frame delay) (ii) both software and hardware implementations of the transcoder should be cheap, (iii) the level at which transcoding is done should scale from the bitstream level to the packet level and (iv) the on-line transform coding (in this case the DWT) should be flexible enough to accomodate a number of coders (scalar quantization, vector quantization etc.).

The overall scheme for the encoder, the decoder and transcoder is shown in Figure 1. The blocks labeled SPA (Symmetric Pyramid Algorithm) and ISPA (Inverse SPA) are the transform coders which compute the DWT and IDWT. Details of these algorithms and architecture are deferred to sections 3 and 4, respectively. E is an encoder which encodes an input frame and outputs octaves 1 to J (maximum resolution at octave 1 and minimum at octave J). The jth decoder operates on octaves $D(j)$ to J, where $D(j)$ depends on the data rate requested by the decoder's host, the quality of the desired image, etc. The encoder does not output all the octaves at the same bit rate. In fact, the encoder assigns a fixed number of bits to each octave output. This assignment is a function of the energy, and will not be described here. Each decoder reads in a fixed number of bits corresponding to an octave output. In summary, each decoder reads in some or all of the output octaves transmitted by the encoder, and some or all of the bits corresponding to an octave output. The transcoder is a simple state machine which handles this mismatch of the data rates of the encoder and the decoder.

The motivation behind our approach is based on the following observation. When the current lowest octave of the transform coded (DWT) data is dropped, there is a factor of 4 reduction in the amount of uncompressed data (and subsequently size of the image). Software simulations of our scheme have shown that if we use scalar quantization followed by run-length and Huffman encoding on the transform coded data and then drop an octave of this compressed data, the reduction is now by a factor of around 2. This factor can be made close to 3 if we use VQ. Thus computing 5 octaves (and encoding it using VQ) of a 512×512 image gives us a dynamic range of over 200, ie., the fastest link (receiver) on the network can be 200 times faster than the slowest link (receiver). Our scheme provides a video stream to both recievers with the video quality (for same sized image) being directly proportional to the link (receiver) speed without touching uncompressed data.

In this paper we consider two coding schemes for the subband coefficients. Both schemes have the following properties. (i) They are computable using on-line algorithms, and (ii) they scale the SPA-ISPA schedule in an uniform manner across all octaves.

- Scheme 1: The subband coefficients are scalar quantized using PCM. Error correction is required for the coefficients of the lowest subbands. Another possibility is scalar quantization followed by run-length encoding and Huffman coding. Both these schemes operate on individual outputs of the SPA, and do not require the outputs to be blocked.
- Scheme 2: The subband coefficients are grouped into blocks, and then vector quantized. For our application, it is desirable that the subband coefficients be compressed in multiple stages. This is necessary to support the difference in precision that is required by the various decoders. For eg., Residual vector quantization (RVQ) is a scheme that is suitable for our application [6].

2.1. Encoder and Decoder Architectures

In the first scheme, the encoder merely consists of the 2-D SPA module followed by a coding module, while the decoder consists of the decodong module followed by the 2-D ISPA module. In the second scheme, the encoder consists of a 2-D SPA module followed by a raster to block converter and a VQ encoding module. The decoder consists of a VQ decoding module followed by a block to raster converter and a 2-D ISPA module. The architectures for the encoder and decoder of the second scheme are described in Figures 2 and 3. The raster to block converter is required at the encoder since the output of the 2-D SPA unit which is in raster format has to be converted into the block format before applying the VQ. We consider 2 cases; (i) When the block size (or vector dimension) is $1 \times m$. (ii) When the block size is $b \times m$, $b > 1$. In the first case, each raster to block converter consists of mk-bits of storage. Thus if there are J octaves the total storage required is Jmk-bits. For a typical color video application $J = 5$, $m = 8$ and the precision $k = 16$. Thus the tota storage for such an application would be 640-bits + VQ codebook size + storage requirements of the SPA (or ISPA) architecture (see section 4). In this case the raster to block converter (or block to raster converter) storage is trivial compared to the VQ codebook size. The total decomposition-reconstruction latency in this case is (latency of 2-D SPA-ISPA + latency of the VQ).

For the second case, each raster to block converter consists of bN_jk-bits of memory and a simple address generator, where the block size (or vector dimension) is $b \times m$, the

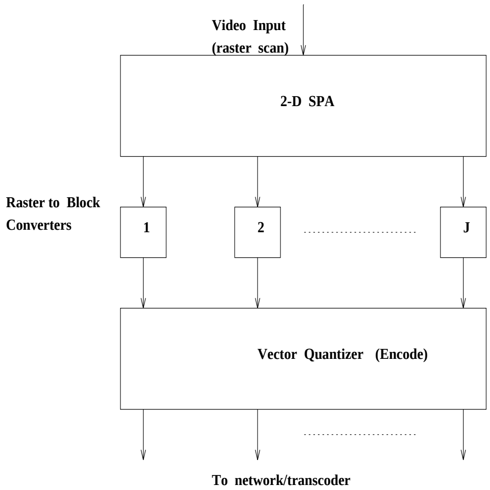

Figure 2. **The Encoder Architecture for the second scheme**

number of outputs per line of octave j is N_j and the output precision of the 2-D SPA unit is k. For the 2-D DWT, if the input frame size is $N \times N$, then the number of outputs of octave j is $3 \times \left(\frac{N}{2^j} \times \frac{N}{2^j}\right)$. Thus the total storage required by the raster to block converters is $bNk\left(3 - \frac{1}{2^{J+1}}\right)$ bits, ie., less than $3bNk$ bits. The storage requirements of the block to raster converter is the same as that of the raster to block converter. The storage requirements of the SPA (or ISPA) unit is $2LNk$, where L is the length of the wavelet filter. Thus the total storage requirements of the encoder (decoder) is less than $3bNk + 2LNk+$ (VQ codebook size)- bits. For a typical color video application we have $N = 640$, $k = 16$, $L = 4$ and $b = 4$, thus the total storage is less than 24 KBytes + VQ codebook size. For a 4×4 vector dimension and 0.5 bits per pixel rate, the size of the codebook for full search is about 16 KBytes. Thus the total storage requirements in the worst case is about 40 KBytes (in contrast the storage required by a 640×480 color frame is about 512 KBytes). The storage can either be on-chip RAM or a cheap 8Kx8 static RAM chip. The total latency in this case is (latency of 2-D SPA-ISPA + latency of the raster to block converter + latency of the VQ). The latency of the raster to block converter (for the first octave output, ie., the worst case) is bN output cycles. We now present brief descriptions of the SPA-ISPA algorithms and their architectures.

3. FORWARD-INVERSE DISCRETE WAVELET TRANSFORM

For a description of the DWT and standard algorithms for it, please refer to [4] [5]. In the existing on-line implementations of the DWT [2], [3], the outputs are computed in such a way that the output of the DWT block is not compatible with the input of the IDWT block. As a result, large buffers are required to permute the data in between the two blocks. The latency of these implementations is also quite large. In this paper, we propose algorithms and architectures for computing DWT and IDWT which require (i) minimal buffering, and (ii) minimal latency. We consider the separable case of the 2-D DWT, which is the most commonly used DWT scheme for image coding. For the separable case, the algorithms for 2-D are simple extensions of the algorithms for 1-D. Thus for notational simplicity, in this section, we present the running algorithms for only the 1-D case. The 2-D algorithm basically consists of applying the 1-D algorithms on all rows and transposing the intermedi-

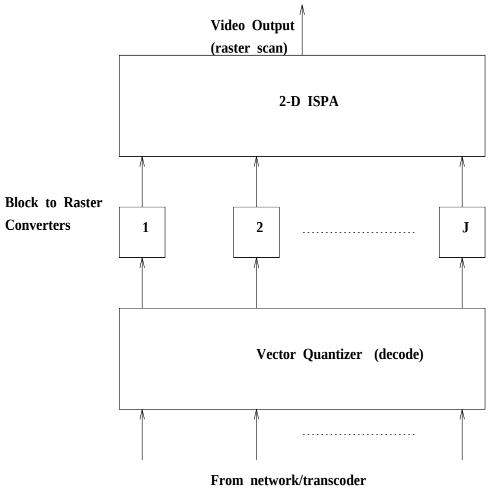

Figure 3. **The Decoder Architecture for the second scheme**

ate result, and applying the 1-D algorithms on the columns to get the final result.

3.1. Forward DWT

We propose a new running algorithm, called SPA, for computing forward DWT in real-time. This algorithm is similar to the Recursive Pyramid Algorithm (RPA) [1] in that it computes the DWT in an on-line manner and requires only LJ cells of storage, where N is the input size, and L is the filter size. As in the case of RPA, the 1st octave computations take place every 4 sample periods, and the higher octave computations are scheduled between the 1st octave computations. The higher octaves are scheduled based on the following two precedence rules: (i) if the earliest instance of the ith octave clashes with that of the $(i+1)$th octave, then the ith octave gets scheduled first, (ii) the 0th output of any octave gets precedence over all but the 1st octave outputs. The second precedence rule results in a reduction in latency in a DWT-IDWT system. The output schedule generated by the SPA for $N = 16$ and $J = 4$ is as follows. Here $y_i(n)$ is the nth low pass output of the ith octave. $y_1(0), y_2(0), y_3(0), y_4(0), y_1(1), -, -, -, y_1(2), y_2(1), -, -,$ $y_1(3), -, -, -, y_1(4), -, y_2(2), y_3(1), -, y_1(5), -, -, -, y_1(6),$ $y_2(3), -, -, y_1(7)$.

3.2. Inverse DWT

We propose a new running algorithm, called ISPA, for computing inverse DWT in real-time using only $\approx LJ$ cells of storage. Let $x_i(n)$ be the nth low pass output of the ith intermediate stage. $x_0(n)$s are the IDWT outputs. In this scheme, the x_0 outputs are computed every other sample period, and the intermediate outputs are computed between the x_0 outputs. The scheduling algorithm is independent of the value of $D(j)$. In other words, two decoders which have different values of $D(j)$ will compute the outputs of the ith intermediate stage at the same time.

Define $\{x_i(k)\}$ as the set of nodes $x_{i-1}(2k), x_{i-2}(4k), \ldots,$ $x_1(2^{i-1}k)$. Though $x_{i-1}(2k)$ is computed before $x_{i-2}(4k)$, and so on, the algorithm schedules a time for $x_{i-1}(2k)$ after $x_{i-2}(4k)$. Such a scheme reduces the size of the buffer as well as the latency.

Step 1 (i) for i=J-1 to 1 , schedule $x_i(0)$.
(ii) for k=0 to N-1 , schedule $x_0(k)$.

Step 2 for k=1 to $N/2^{J-1} - 1$
(i) schedule $\{x_{J-1}(k)\}$
(ii) schedule $x_1(2^{J-2}k - 1)$
If there is a conflict between the times at which $y_{J-1}(N/2^J)$ of SPA and $x_{J-2}(N/2^{J-1})$ of ISPA, then add a delay of $\beta = y_{J-1}(N/2^J) - x_{J-2}(N/2^{J-1}) + 1$, to all the nodes scheduled so far.

Step 3 for P=J-2 to 2
for m=0 to $N/2^{P+1} - 1$
(i) schedule $\{x_P(2m+1)\}$
(ii) schedule $x_1(2^{P-1}(2m+1) - 1)$.

Step 4 for i=1 to J-1
schedule the remaining x_i nodes.

The output schedule generated by the ISPA for $N = 16$ and $J = 4$ is as follows:
$x_3(0), x_2(0), x_1(0), x_0(0), -, x_0(1), x_1(1), x_0(2), x_2(1), x_0(3),$ $x_1(2), x_0(4), x_3(1), x_0(5), x_1(3), x_0(6), x_2(2), x_0(7), x_1(4), x_0(8),$ $-, x_0(9), x_1(5), x_0(10), x_2(3), x_0(11), x_1(6), x_0(12), -, x_0(13),$ $x_1(7), x_0(14), -, x_0(15)$.

3.3. Blocked SPA and ISPA

In Scheme 2, the SPA outputs have to be blocked before being sent to the VQ unit. Similarly, in the decoder, the ISPA unit receives a block of data. Clearly, the blocking procedure will increase the latency of the system. Let J be the number of octaves that have been blocked, and B is the block size, where B is a power of 2, and $B < N/2^J$. Then the ISPA schedule for block size B is the same as the ISPA schedule for B=1 (section 3.2) delayed by an additional amount of $\alpha = y_J(p + B - 1) - x_{J-1}(2p) + 1$, where $p = N/2^{J+1}$. Note that α does not take into account the time that it takes to encode and decode using VQ.

4. ARCHITECTURES

The overall architectures of the two proposed schemes differ because of the differences in the coding techniques employed. However, in the transform coding part of the architecture, the change is minimal. Recall from the previous section, that a change in the blocking factor B merely shifts the schedules of both the SPA and the ISPA. This is reflected by a small change in the state machine. In this section we first describe the transform coding architectures for the 1-D SPA and ISPA, and then describe how they can be extended to 2-D.

The architecture for both the SPA and the ISPA consist of systolic arrays to compute the low pass and the high pass outputs, and a storage unit to store $\approx LJ$ intermediate outputs. Figure 4 describes the block diagram of such an architecture. Since the systolic array implementations are well known, in this section we will briefly describe the storage units in both the architectures. Note that in the SPA architecture the inputs to both the filters is the same while the output of the SPA is the output of the high-pass filter. In the ISPA architecture the inputs to the two filters are different while the output of the ISPA is the sum of the outputs of the two filters.

SPA: The storage unit for the SPA architecture consists of J shift registers, each of length L. The ith shift register stores the data required for the computation of the ith output. A control unit generates control signals to appropriately gate data in/out of the shift registers.

ISPA: The storage unit for the ISPA architecture consists of 2 subunits: unit 1 stores the DWT outputs generated by SPA, and unit 2 stores the IDWT intermediate outputs. In every sample period, $L/2$ data from unit 1 and $L/2$ data from unit 2 are fed to the systolic array. Both the units consist of J shift registers. The length of the unit 1 shift registers are $(L/2 + a_i)$, where $a_i = round(\frac{x_{i-1}(N/2^i) - y_i(N/2^{i+1})}{2^{i+1}})$, while the length of the unit 2 shift registers are $L/2$. For blocked ISPA, the length of the unit1 shift registers are $(L/2 + b_i)$, where b_i is the

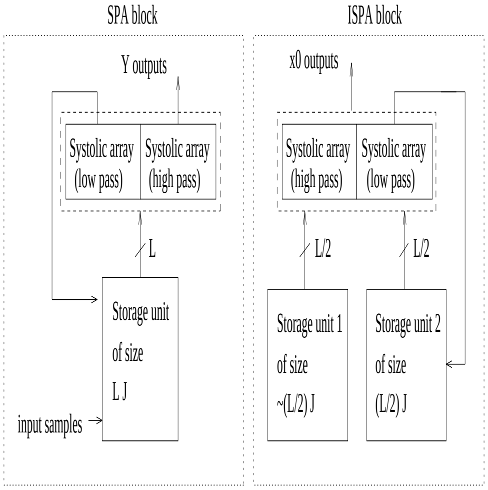

Figure 4. **Block diagrams of the SPA and ISPA architectures**

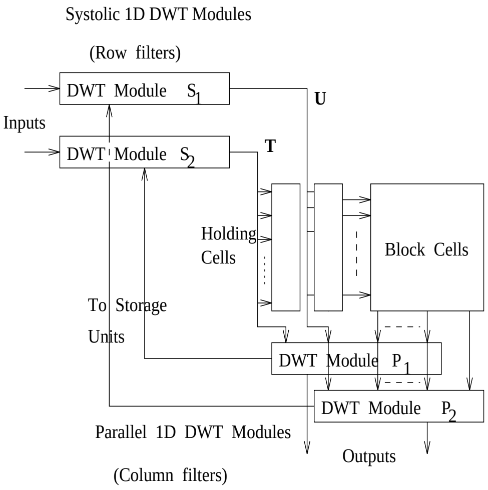

Figure 5. **The 2D DWT Architecture**

smallest number that is closest to a_i, and is also a multiple of the blocking factor B. As in the case of SPA, control signals are generated to read-in/read-out data from the shift registers at the scheduled times.

4.1. 2-D Extensions

For the separable case, the 2D DWT (IDWT) consists of computing the 1D DWT (IDWT) of each of the N columns of $\mathbf{X}$ and then computing the 1D DWT (IDWT) of each of the N resulting rows. Each one of the 1D DWT (IDWT) can be computed using the SPA (ISPA) described above. The resulting architecture is shown in Figure 5. This architecture uses systolic modules to do the DWT in one dimension (S_1, S_2) while the other dimension is handled by parallel filters (P_1, P_2). At each stage of the separable 2-D DWT there is a decimation by 2, ie., there is only one output produced for every 2 inputs [4]. This process is parallelized by using 2 modules (S_1, S_2 and P_1, P_2) in tandem.

The systolic 1D DWT modules, S_1 and S_2, are similar to the SPA block described in the previous section. The only difference is that the storage units get their inputs from the parallel filter, rather than from the low pass systolic filter. The parallel filters could be designed by L multipliers followed by a $\log L$-level adder tree, or it could be based on distributed arithmetic. Since the inputs to the parallel filters come from S_1 and S_2, the output schedules for the 2-D SPA and ISPA are nearly the same as the 1-D SPA and ISPA schedules. The only difference between the 1-D and 2-D cases is an extra delay of p, say, that has to be incorporated in the 2-D schedules due to the fact that the parallel filters have a latency of p.

The operation of the architecture for SPA (ISPA is similar) is as follows;

- Start filtering (decimated by 2) of the i^{th} and $(i+1)^{th}$ rows on S_1 and S_2 respectively. The inputs for their storage units will come from P_2 and P_1, respectively.
- The outputs of these two filters (S_1 and S_2) are fed into the holding cells. The holding cells shift their contents into the block cells once every $2N$ clock cycles, i.e., once every $2N$ clock cycles, the holding and block cells shift right by two. Since these cells are fed by S_1 and S_2, the values in them are stored in the same order as the output schedule described for the 1D DWT (SPA).
- The filters P_1 and P_2 compute 4 rows each, over $2N$ cycles, one row (from each set of 4) is to be used by the storage units of S_2 and S_1 respectively. They produce the outputs in the same order as the row filters and this is exactly the order required by the storage units (and subsequently by the row filters).

The storage required by this architecture is $2NL$, since at any point in time the L latest rows of outputs (all the octaves, lowpass and highpass) from the row filter are stored. Each of the two column filters (parallel) can now operate on these $2N$ columns, each of length L, to produce 4 rows. The 2D DWT (IDWT) is computed by this architecture in N^2+N cycles with a decomposition-reconstruction latency of $2J++\beta+1$ cycles. The area complexity is $A=O(NLk)$ (since there are $2NL$ k-bit storage elements).

5. RESULTS AND CONCLUSIONS

We have presented a novel VLSI (on-line) algorithm and architecture for hierarchical coding using the wavelet transform. A novel hierarchical scheme which enables cheap video conferencing and/or multicast over a heterogeneous network has also been presented. Such a scheme can be implemented with no multipliers if the Haar wavelet or the Daubechies 4-tap wavelet is used in the SPA (and the ISPA) algorithms, and approximation schemes are used for the VQ. This would enable single chip implementations of the encoder, the decoder and the transcoder.

REFERENCES

[1] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," To appear in the *IEEE Trans. on Signal Processing*, March, 1994.

[2] C. Chakrabarti, M. Vishwanath, and R. M. Owens, " Architectures for Wavelet Transforms", *VLSI Signal Processing VI*, pp.507-515.

[3] K. K. Parhi, and T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Trans on VLSI Systems*, June 1993.

[4] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. Pattern Analysis and Mach. Intell.,*, vol.11, No.7, pp.674-693, July, 1989.

[5] O. Rioul, P. Duhamel, "Fast Algorithms for Wavelet Transforms", *IEEE Trans. on Information Theory,* Vol.38, No.2, pp.569-586, Mar., 1992.

[6] C.F. Barnes, E.J. Holder, "Classified Variable Rate Residual Vector Quantization Applied to Image Subband Coding," *Data Compression Conference 93,* pp. 272-281, 1993.

A VLSI ARCHITECTURE FOR REAL-TIME HIERARCHICAL ENCODING/DECODING OF VIDEO USING THE WAVELET TRANSFORM

Mohan Vishwanath
Computer Science Lab
Xerox PARC,
Palo Alto, Ca 94304, USA
mohan@parc.xerox.com
and Chaitali Chakrabarti [1]
Dept. of Electrical Engg.,
Arizona State University
Tempe, AZ 85287, USA
chaitali@asu.edu

Novel on-line algorithms and architectures for hierarchical coding using the wavelet transform are presented. These algorithms/architectures compute the decomposition-reconstruction cycle of the wavelet transform with minimum latency and buffering for any given blocking factor. A hierarchical scheme which enables cheap video conferencing and/or multicast over a heterogeneous network is also presented. This architecture supports single chip implementations of the encoder, the decoder, and the transcoder for some choices of the wavelet filter and vector quantization schemes.