Abstract: The digital data can be transformed using Discrete Wavelet Transform (DWT). The images need to be transformed without losing of information. The Discrete Wavelet Transform (DWT) was based on time-scale representation, which provides efficient multi-resolution. Here Low Pass filter coefficients and the High Pass filter coefficients filter give lossless mode of information as per the JPEG 2000 Standard. The new JPEG2000 and MPEG4 still image and video compression standards are based upon the DWT and are shown to produce superior results over their previous incarnations that do not use the DWT. The discrete wavelet transform (DWT) is being increasingly used for image coding. This is due to the fact that DWT supports features like progressive image transmission (by quality, by resolution), ease of transformed image manipulation, region of interest coding, etc. DWT has traditionally been implemented by convolution. Such an Implementation demands both a large number of computations and a large memory features that are not desirable for either high-speed or low-power applications. Recently, Available on chip memory storage and external memory bandwidth determine the range of parallel architecture choices and the degree of design scalability and signal optimization.

Keywords: Discrete wavelet transform; signal and memory optimization; VHDL; 2D architecture.

I. INTRODUCTION
Recently, there has been a tremendous increase in the application of wavelets in many scientific disciplines. Typical applications of wavelets include signal and image processing numerical analysis, with memory optimization. While the wavelet transform offers a wide variety of useful features, it is computation intensive. Furthermore, in contrast to other transforms, such as Fourier transform or discrete cosine transform, it is not block based, which makes it difficult to implement in a parallel representation. Several VLSI and FPGA architectural solutions for the discrete wavelet transform have been proposed in order to meet the real time requirements in many applications. These solutions include parallel filter architectures, linear array architectures, multigrid architectures, and 2D block based architectures. Most of these implementations are special purpose parallel processors developed for specific wavelet filters and/or wavelet decomposition trees that implement high level abstraction of the standard pyramid algorithm. In addition, some are complex designs requiring extensive user control. Knowles proposed systolic-array-based architectures without multipliers for the 1-D and 2D DWT, but these architectures are not suitable for all wavelets. We proposed a systolic-parallel architecture for the 2D DWT based on the recursive pyramid algorithm, but due to the approximations involved these architectures cannot be used when exact reconstruction is required. We proposed a sequential implementation of the polyphase representation of the DWT suitable for the Xilinx Virtex FPGAs. Yong-Hong et al. presented a parallel architecture that can compute low pass and high pass DWT coefficients in the same clock cycle. King-Ch et al. implemented the operator correlation algorithm of the 2D DWT.

However, these FPGA implementations are aimed at specific filterbanks, do not support block-based transform, or do not handle block boundaries efficiently. There is a clear need for a fast hardware DWT that allows flexibility in customizing the wavelet transform with regard to the filters being used and the structure of the wavelet decomposition. In many image processing applications, including compression, denoising and enhancement, it is critical to compute the 2D wavelet transform in real-time. Field programmable gate arrays (FPGAs) offer a suitable platform (cost effective and highly flexible) for such an implementation. FPGA-based systems represent a new paradigm in the industry – a shift away from a full custom ASIC solutions for each application, to a single hardware assembly (FPGA) that can be reconfigured to accommodate multiple applications.

II. PROPOSED ARCHITECTURE
Working Principle
A. 2D DWT Architectures for Hardware Implementations
The DWT, as represented by the Mallat style multilevel octave-band decomposition system, which uses a two-channel wavelet filter bank, is very computation intensive. This decomposition can be implemented as a pyramidal recursive filtering operation using the corresponding filter banks as shown in Figure 1. We will refer to it as the standard algorithm. The process for the 2D DWT decomposition for each level is implemented with a cascaded combination of two 1-D wavelet transforms. The standard algorithm is constrained by large latency, a high computational cost and the requirement for a large buffer size to store intermediate results, which makes it impractical for real time applications with memory 40 constraints. An alternative representation, requiring fewer computations, is the lifting algorithm which will be the basis of our implementations in this chapter.
Recently, as wavelets gained popularity. Most

n NxN image, using

approach is that it requires data

- (½)

- B2 required to complete the filtering

- (½)(s

- o several blocks and 43 operate on

43x38

www.ijtre.com

multiple image blocks simultaneously. A known

multi

the recursive filtering operation and the

architecture still requires the same intensive computations of

would merge the results to complete the DWT. While this

The standard DWT algorithm operates on the whole image in

C. Proposed Parallel Architectures for the 2D DWT

The standard DWT algorithm operates on the whole image in

a sequential manner. An improved implementation would

partition the image into several blocks and 43 operate on
each block independently and in a parallel manner, and then

would merge the results to complete the DWT. While this
architecture still requires the same intensive computations of
the recursive filtering operation and the same memory
requirements, the computation can be sped-up if one uses a
multi-processor system or identical parallel hardware
implementations of the filtering blocks that can operate on
multiple image blocks simultaneously. A known disadvantage
of such an approach is that it requires data exchanges between neighboring blocks at each
decomposition level of the discrete wavelet transform, and

hence an additional overhead due to inter-processor
communications. We consider three parallel implementations
based on the lifting factorization of the DWT. The standard
overlapping algorithm shown in Figure eliminates the
blocking artifacts and imposes relatively simple control
complexity, but has high computational cost and requires
high on-chip buffering of data. For an NxN image, using
DWT filters of length less than or equal to L, and partitioned
into S Blocks, the number of additional filtering operations
for a 1 level 2D DWT decomposition vs. a non-overlapped
approach is: 2N*L*(S-1).

III. THEORETICAL FRAMEWORK

MEMORY OPTIMIZATION

A. Memory Bandwidth Considerations and Storage

Consider the stripe-parallel design shown in Figure 3.6. After
the completion of 1 level DWT decomposition, the number of
transitional boundary states generated at the first boundary
of B1 and B2 is:

M_1 = \frac{1}{2} F * N

From B1

From B2

M_2 = \frac{1}{2} F * N * \frac{1}{2}

where memory is measured here in number of pixels, j and

\( l \) are the ceiling and the floor operators to accommodate odd
length DWT filters at the stripe boundaries.

This results from the absence of image data along the
boundaries of B1 and B2 required to complete the filtering
operations. After the completion of 2 decomposition levels
additional transitional boundary states are generated at the
same boundary

\[
\begin{align*}
F & = \text{length of the longest filter} \\
J & = \text{DWT decomposition levels} \\
S & = \text{Number of DWT line processors (blocks/stripes)} \\

\text{From B1} & \quad M_1 = \frac{1}{2} F * N * \frac{1}{2} \\
\text{From B2} & \quad M_2 = \frac{1}{2} F * N * \frac{1}{2}
\end{align*}
\]

Hence the memory required to hold transitional boundary
states for each boundary is

\[
M = F \sum_{i=0}^{J-1} N * \left( \frac{1}{2} \right)^i
\]

\[
M_{\text{total}} = F \sum_{i=0}^{J-1} N * \left( \frac{1}{2} \right)^{(S-1)}
\]

To minimize external memory I/O bandwidth, the decision
was made earlier to use
the cascaded architecture, i.e., pipeline row and column
filtering. Assuming a FIFO
buffer length of N (image width), the memory required for
row buffering is:

\[
M_{\text{FIFO}} = F * N
\]

and total needed memory for FIFO buffers is

\[
M_{\text{FIFO, total}} = F * N * S
\]
For example: for an image of 512x512 pixels, a 3 level (9,7) DWT decomposition with a partition size of 4 stripes, the total required on-chip RAM (BRAM) measured in pixels is: 9*(512+256+128)*4 + 9*512*4 = 42K bytes Actual required BRAM may need to be 84K bytes to account for dynamic expansion in DWT domain.

B. For signal optimization
The pyramidal algorithm for 2-D DWT with separable wavelet bases is given by the following equations.

\[ X'(i, j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} h(k)h(2k)X^{-1}(2i-k,2j-k) \]
\[ Y'(i, j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} g(k)h(2k)X^{-1}(2i-k,2j-k) \]
\[ U'(i, j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} g(k)h(2k)X^{-1}(2i-k,2j-k) \]
\[ V'(i, j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} g(k)h(2k)X^{-1}(2i-k,2j-k) \]

Where \( X'(i, j), Y'(i, j), U'(i, j) \) and \( V'(i, j) \) are the coefficients of s-th stage of 2-D DWT and \( i, j = 0, 1, \ldots \), for low-pass sub-bands and high-pass filters used for decomposing the input data matrix of size \( (N \times N) \) into four sub-bands. The pyramidal algorithm pertaining to the low-low sub-band computation \([X'(i, j)]\) may be decomposed into two stages of computation as:

\[ X'(2i, 2j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} h(k)W^0(k)X^{-2}(k)(2i-2j-k) \]
\[ W'(2i, 2j) = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} h(k)X^{-2}(k)(2i-k, j) \]

in z-domain representation:

\[ W'(Z_1, Z_2) = W^{-1}(Z_1, Z_2) \]
\[ X'(Z_1, Z_2) = W'(Z_1, Z_2) \]

Where, the input and output of equation are time multiplexed in z-domain. Equations can also be derived for \( Y'(i, j) \), \( U'(i, j) \) and \( V'(i, j) \) in z-domain and the 2-D DWT, therefore may be computed.

Design of an Efficient VLSI Architecture for 2D DWT Wavelet Image Processing

The selected low pass or high pass filter are FIR (finite impulse response) filters. The transfer functions for these filters are as, Transfer function for low pass filter,

\[ H(z) = h_0H^{-1} + h_1H^{-2} + h_2H^{-3} + h_3H^{-4} + h_4H^{-5} \]
\[ G(z) = g_0H^{-1} + g_1H^{-2} + g_2H^{-3} + g_3H^{-4} + g_4H^{-5} \]

IV. IMPLEMENTATION RESULTS AND DISCUSSIONS
A. IMPLEMENTATION RESULTS
RESULT OF MEMORY OPTIMIZATION

<table>
<thead>
<tr>
<th>Number of</th>
<th>ONE DWT MODEL</th>
<th>TWO PARALLEL DWT MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice</td>
<td>3380 (10% of total available slices)</td>
<td>7267 (22%)</td>
</tr>
<tr>
<td>Flip Flops</td>
<td>3488 (5%)</td>
<td>7499 (11%)</td>
</tr>
</tbody>
</table>

Table1: Resources Utilization for the Overlap-State Implementation

Table2: Resources Utilization and Throughput Comparisons to other Optimized Methods

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Cast Inc.</th>
<th>“Software pipelined”</th>
</tr>
</thead>
<tbody>
<tr>
<td>(LB-2D)</td>
<td>Virtexs 2</td>
<td>Virtexs 2</td>
</tr>
<tr>
<td></td>
<td>Virtexs 2</td>
<td>Virtexs E</td>
</tr>
</tbody>
</table>

| Number of | 2227 (22%) |
| Slices    | 966 (98%)  |
| BRAM       | 84k (75%)  |

| Number of Multipliers | 16 (5%) | 34 (10%) |

<table>
<thead>
<tr>
<th>Performance (MBytes/sec)</th>
<th>9</th>
<th>98</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock 100 MHz Frequency (MHz)</td>
<td>51</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>Image Size</td>
<td>256x256</td>
<td>1024x1024</td>
<td>512x512</td>
</tr>
<tr>
<td>DWT Trans. Level</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

B. Performance Evaluation
(1) for memory optimization

![Performance Parallel Algorithms](image)

Fig2. Performance algorithms
V. CONCLUSIONS

We presented, in thesis, a methodology for parallel implementation of memory and signal optimization using DWT on FPGAs. We investigated and analyzed parallel and efficient hardware implementations targeting state-of-the-art FPGAs. We addressed practical considerations and various design choices and decisions at all design stages to achieve an efficient DWT implementation, subject to a given set of constraints and limitations. We presented a specific optimization representation for the DWT that provides architectures suitable for efficient hardware implementation, and a novel data transfer method that provides seamless handling of boundary and transitional states associated with parallel implementations.

Also in recent years, several architectures have been proposed for 2-D discrete wavelet transform. However, the hardware of these architectures needs to be further improved. Therefore, in this paper, we have proposed an efficient recursive architecture for 2-D DWT. The advantages of the proposed architecture are saving adders, multipliers, simple control complexity, and complete hardware utilization, making this design suitable for image processing systems.

REFERENCES

