Systolic and Load Balanced Structure for Full Wavelet Transform and Wavelet Packet Transform

Dongming Peng and Mi Lu
Electrical Engineering Department
Texas A&M University
College Station, TX 77843
mlu, peng@ee.tamu.edu

1 Introduction
The discrete Full Wavelet Transform (FWT) and the discrete Wavelet Packet Transform (WPT) are algorithms of multi-stage wavelet transforms, in which some or all of the results from earlier stages of transform are continuously transformed in later stages. The two algorithms have received much attention in many areas including communication systems[2][5], image processing and compression[4][6], etc. There have been many reports on the architectures[7][8][9][10] for multi-stage Discrete Wavelet Transform (DWT) as shown in Figure 1(d), yet few on the architectures for FWT and WPT. We contribute to pipelining the algorithms of 2-D as well as 1-D FWT and WPT by using special-purpose architectures based on a new systolic and load balanced structure proposed in this paper. Our designs are desirably area efficient and of real-time performance, without involving any memory manipulation. These advantages are especially attractive to the systems that need efficient and fast hardware.

2 Architecture for Computing the 1-D FWT

The notations used in the algorithm for 1-D FWT are shown in Figure 1(e). The algorithm is calculated by the following equations at various stages:

\[ C^{(i+1,2j)}[n] = \sum_{k} h_{low}[2n - k] \times C^{(i,j)}[k] \] (1)

\[ C^{(i+1,2j+1)}[n] = \sum_{k} h_{high}[2n - k] \times C^{(i,j)}[k] \] (2)

where Eq. 1 corresponds to the low-pass wavelet filtering, Eq. 2 to the high-pass wavelet filtering, and \( C^{(i,j)}[n] \) represents the \( n^{th} \) point datum in the \( j^{th} \) output signal sequence at the \( i^{th} \) stage of FWT decomposition. \( C^{(i,j)} \) is the low-pass output if \( j \) is even, otherwise the high-pass output.

We introduce a new synchronized wavelet filter implementation[10], which is used in our system as a Processing Unit (PU), whose structure and brief explanation are in Figure 2(a). It rearranges the calculation of wavelet filtering so that the filter is cut to half taps based on the symmetry between the negative and positive wavelet filter coefficients. \( z \) and \( c \) are the input sequence, low- and high-pass filtering output sequence respectively. While sequence \( z \) is fed into the PU with one datum per clock cycle, the data in sequence \( a \) are calculated on even indexed clock cycles and the data in sequence \( c \) are calculated on odd indexed clock cycles. The PU in Figure 2(a) can be extended to a parallel format as in Figure 2(b) where if a number of data from sequence \( z \), e.g., \( x_{k+9}, x_{k+7}, \ldots, x_{k} \) are fed to the PU in parallel in a clock cycle, then \( x_{k+9}, x_{k+8}, \ldots, x_{k+1} \) are fed in the next cycle.

With the PU, our basic scheme is to assign to each PU a transform stage, and feed the data sequences to PUs in either a sequential way as shown in Figure 2(a) or a parallel way as shown in Figure 2(b). For example, \( z \) is sequentially fed to PU1, and \( a \) and \( c \) are alternatively generated and supplied to PU2. Two groups of data from \( a \) and \( c \) are alternatively fed to PU3 with each group of data being in parallel, and PU3 takes turns to use them as inputs to calculate relative convolutions every other clock cycle. Two linear systolic arrays are necessary to store and shift the recently generated data per clock cycle, one for \( a \) and the other for \( c \). Each cell in the arrays has a port connected to PU2, so that the two arrays can alternatively supply PU2 with a group of parallel data from \( a \) or \( c \). The length of either array is \( N \), the size of the wavelet filter. The designs of more transform stages can be figured out in a similar way.

The architecture for three transform stages is illustrated in Figure 3. \( SA_1, SA_2, SA_3, SA_4, SA_5 \) and \( SA_6 \) are linear semi-systolic arrays as described above and supply the relative PUs with parallel data. \( DM_1, DM_2, DM_3, DM_4 \) and \( DM_5 \) are switches selecting appropriate data feed to PUs at different time so that the PUs perform the correct computation at every clock cycle. The local input/output relations at different time for PU1, PU2 and PU3 are shown in Table 1. It is demonstrated in Table 1 that 1) PUs have balanced loads; 2) the FWT has been pipelined; 3) the output of the system is generated at a uniform rate, the same as that of the input; 4) the intermediate data are calculated as earlier as possible. The architecture in Figure 3 whose behavior is shown in Table 1 is a new systolic tree integrating the systolic implementation of tree structures and the systolic linear arrays. Its sketch is composed of a number of trees in systolic implementation where each tree has a PU as its root to process data in systolic ways, and each leaf node on the trees by itself is a linear systolic array (SA).

3 Architecture for Computing the 2-D FWT

Lemma 1. Exchanging the order of row-wise transforms and column-wise transforms across the transform stages does not change the final results of separable 2-D FWT.

Proof: The separable 2-D FWT equations are as follows[11]:

\[ C^{(i+1, j+i+1)}[n_1, n_2] = \sum_{k_1} \sum_{k_2} F_1[2n_1 k_1] \times \]
systems and \( N \) is hundreds or thousands of times greater than \( N_w \) (the width of the wavelet filter) for most applications of 2-D wavelet transforms (such as in image/video processing systems), \( T_u \) is much more efficient than a memory regularly used in 2-D wavelet transforms which at least contains \( N \times N \) cells. endproof.

Based on Lemma 1 and Lemma 2, we implement the 2-D FWT by cascading two 1-D FWT architectures but replacing the SA with the TU within the second 1-D FWT architecture, as proposed in Figure 6. TUs supply the \( P_{u1} \), \( P_{u3} \) and \( P_{u5} \) with the transposed data. Three stages of the row-wise DWT are firstly calculated when matrix \( X \) is transposed in the order indexed in Figure 5(b), and \( P_{u6} \), \( P_{u4} \) and \( P_{u5} \) perform the three stages of column-wise DWT. \( X\) takes \( N_w \) column-wise data items from \( T_u \) and performs the column-wise wavelet filtering. The output \( Z \) is \( C(C(C(R(R(X)))))) \) or \( C(R(C(R(C(R(X)))))) \), thus the 3-stage 2-D FWT is achieved in real time. As shown in Figure 3, the PU switches between the calculations of high- and low-pass filtering periodically. Accordingly, the local input/output relations for every PU and switch (the DM) at different time are shown in Table 2. The real-time output sequence periodically gives the data in every generated subband according to Table 2.

4 Extension to the WPT

We add a value field Node Expansion or \( NE(i,j) \) to each node \((i,j)\) in wavelet expansion tree, which corresponds to the \( j \text{th} \) subband in the \( i \text{th} \) transform stage. \( NE(i,j) \) equals \( TRUE \) if the corresponding subband needs to be decomposed in the next stage; \( NE(i,j) \) equals \( FALSE \) otherwise.

To map the WPT to an architecture comparable to that in Figure 3, the only extra work needed is with the NE value field. It has been shown that the SAs in Figure 3 is corresponding to the transformed subbands in Section 2. So we attach one extra bit (called NE cell) to the end of each SA, and use such a bit to store the NE status of the corresponding subband. The PU takes the corresponding SA’s normal content in parallel with this extra bit as its input. If this bit is asserted (TRUE), the PU performs the same transform on the input data as described in Section 2. Otherwise the PU leaves the data unchanged and transfers them to its output port. To prevent these data from being processed by the upcoming PUs, there are wires connecting different NE cells so that once an NE bit is FALSE the relative NE cells belonging to the later stages are set to FALSE too. In this way, the WPT has been realized by a structure with only a little modification on the architecture shown in Figure 3. The implementation of 2-D WPT can be figured out similarly.

5 The Performance Analysis and Conclusion

Because all the PUs have been processing data and all the storage cells have been storing and transferring data as long as the input sequence is fed to the system, the hardware utilizations for the architectures shown in Figure 3 and Figure 6 are both 100% (disregarding the overhead which is independent of the input size). In the 1-D architecture, since the response of the first datum from input sequence reaches
the system output port through PU1, PU2, PU3 in 3 cycles according to Table 1, the overhead is only 3 cycles. By the similar reason, the overhead is 6 cycles for the 2-D case in Figure 6. The bounday effect whose size is independent of the size of input signals is disregarded due to the space limit of this paper, but it can be resolved by a little adjustment of the hardware. Since each MAC (Multiply-Accumulate Cell) for the convolution computation is essentially a p-bit unit (p is the precision of operations), the minimal area is O(p^2). So in the case of 1-D architecture, the required area for MACs is O(pN^2 Log(C)), assuming C is the total number of subbands of the transform. The required area for the storage cells is O(pN^2 C). Thus Axr×Time=O(pN^2 CN). This is even not worse than the work for DWT computation presented in [7][8][9], given the higher complexity of FWT demanding more time and storage for its computation compared with DWT. The implementation for 2-D transforms can achieve the similar performance based on the scheme designed for 1-D transforms.

Unlike many other designs for wavelet transforms, our schemes are scalable to any length of wavelet filters and any stage of decompositions. The architectures require practical I/O rates to achieve real-time performance and need no input buffering. The designs are not memory-based and need little storage in contrast to the large size of input sequence. Since the operations of each device are periodic, the control signals are generated locally and no external control signal is necessary, thus the corresponding amount of non-local wiring is small.

References