CONCLUSION

DWT provides an efficient computing method for sparse representation of wide class of signals. The DWT analyzes the lower frequency sub-bands implicitly ignoring any information embedded in the higher frequency components. The multi-level 2-D DWT is widely used in various multimedia applications such as image analysis, sub-band coding and image compression, volumetric data compression etc. The 2-D DWT algorithm is computationally intensive and many of its applications require real-time computation to deliver better performance. Efficient realization of 2-D DWT in dedicated system has great practical interest for low-power and resource constrained DSP applications. With the advancement of VLSI technology, high density and low-cost memory chips are rapidly evolving in recent years. DSP algorithms are implemented in FPGA or ASIC for real-time applications. FPGA offers high capacity programmable devices for realization of complex DSP algorithms. FPGA uses a fixed architecture and offers specific types of memory and logic resources for realization of digital systems. On the other hand, modern synthesis tools offer a wide range of logic and memory components for realization of ASIC systems. Therefore, ASIC system offers higher-performance and consumes less-power compared to FPGA systems. However, ASIC system does not allow reusing its resources through programming unlike the FPGA. Therefore, FPGA offers rapid prototyping the complex DSP algorithms with lesser performance than the ASIC. In general ASIC implementation is preferred for computation intensive algorithms and high volume applications. Several design schemes have been suggested in the last two decades for efficient implementation of 2-D DWT in VLSI system. Researchers have adopted different algorithm formulation, mapping scheme, and architectural design methods to reduce the computational time, arithmetic or memory complexities of 2-D DWT.

Chapter 2 presents the literature review on computation schemes and architectural designs are considered to find an efficient hardware design for 2-D DWT. Literature survey reveals that different types computation schemes such as convolution scheme, lifting and flipping scheme have been considered to reduce the combinational logic complexity and critical-path delay of 2-D DWT structure. Different types of mapping schemes such as RPA and folded scheme have been proposed to improve the utilization efficiency of the hardware design. Parallel architectures

also have been proposed to reduce memory complexity of 2-D DWT. It has been observed that DWT designs involve data-selectors (multiplexors and de-multiplexors) apart from multiplier, adder and memory elements. DWT structures use data-selectors for time-multiplexing data sequences to improve resource utilization of the hardware design while performing down-sample filter computation. Data-selector complexity depends on block size (throughput rate) of folded and RPA designs. In case of parallel designs, data-selector complexity depends on mapping algorithm, input block-size and decomposition levels. The parallel design of 2-D DWT involves a large size input data-block for higher decomposition-level. The data-selector complexity of such parallel designs is relatively large compared to the folded and RPA designs and that affect the area-delay efficiency of the design substantially. However, the data-selector complexity is overlooked in the existing parallel designs. Also, it is observed that the arithmetic unit of parallel design of large block size contributes almost comparable amount area as the on-chip memory unit. However, in the existing parallel structures the design effort is mostly focused on the on-chip memory optimization. Since the parallel structure of large block size involves hundreds of multipliers, use of an optimized hard multiplier could improve the area-delay efficiency substantially. Few memory-less designs have been proposed in the technical literature for low-complexity realization of DWT, but these designs use convolution scheme. Look-up-table based multiplier design could be a better choice for the parallel designs based lifting scheme which is yet to be explored.

Chapter 3 presents a review of data access schemes considered to develop 2-D DWT computing structures. In parallel design, data-blocks are reordered and time-multiplexed at different stages of DWT computation. The data-reordering format depends on the input data-access scheme used by the parallel design. The input-output data-flow of different levels of 2-D DWT computation is studied to find the reordering mechanism of intermediate data-blocks. Based on this study a novel data-access scheme is formulated to avoid data-multiplexing which is common in the existing parallel architectures. A block formulation is presented for vector computation of multi-level lifting 2-D DWT. Using the proposed block-formulation a generic processing unit design is presented. The proposed generic design is controlled by a single parameter *i.e.* the input-vector size. A regular and modular parallel architecture is derived using the generic processing unit design. The proposed parallel architecture is easily scalable for higher block-sizes as well as

higher DWT levels without sacrificing its circuit regularity and modularity. This is an important feature of the proposed architecture. The proposed parallel architecture is, therefore, suitable for high-throughput realization of multi-level lifting 2-D DWT.

Chapter 4 discusses Booth multiplication algorithm to develop an efficient hardware design for sign multiplication. Booth multiplier design is broadly divided into three parts as (i) partial product generator (PPG), (ii) partial product selector (PPS) and (iii) adder unit (AU). The complexity of PPG and PPS unit depends on the partial product set while the complexity of AU depends on the number of partial product row of partial product array. The higher radix Booth multiplication algorithm reduces the number of partial product rows of PPA while it increases the size of the partial product set in an exponential order. Consequently, the area-delay efficiency of higher radix Booth multiplier designs appear not efficient than the radix-4 Booth multiplier design. It has been observed that area-delay efficiency of radix-8 Booth multiplier design is closest to radix-4 Booth multiplier design. A review of radix-4 and radix-8 Booth multiplication algorithms using different encoding scheme such as anti-symmetric product coding (APC) symmetric, symmetric product coding (SPC) and symmetric anti-symmetric product coding (SAPC) is presented. Using SAPC scheme partial product rows is reduced by one-forth compared to direct Booth encoding scheme. However, the SAPC scheme introduces extra logic operation both in the PPS and AU which is an overhead. A detail complexity analysis of radix-4 and radix-8 Booth multiplication is presented to study the hardware and time complexities of multiplier designs using different Booth encoding schemes. Based on these findings, a regular partial product array for radix-8 Booth multiplication is presented with a small overhead complexity. ASIC synthesis result shows that the proposed radix-8 multiplier design for n=12, involves 6.1% less area-delay product and consumes 4.1% less power than the state-of-the art radix-4 Booth multiplier design of [Kuang *et al*. (2009)]. The proposed radix-8 multiplier design may be considered instead of the popular radix-4 Booth multiplier for 12-bit multiplications.

Chapter 5 presents a constant post-truncated radix-8 Booth multiplier design. Few redundant logic operations are created within the adder unit when n lower-order bits of 2n-bit multiplier output are truncated. An optimized adder unit design is presented after removing all such redundant logics for post-truncated fixed-width radix-8 Booth multiplier. Comparison result shows that the proposed post-truncated fixed-width radix-8 Booth multiplier design offers nearly

20.7% less ADP and consumes 18.3% less power over the existing radix-8 design optimized by Synopsys Design Compiler for post-truncation. It is observed that radix-8 multiplier design offers some area and delay saving when configured for constant multiplication, while the radix-4 multiplier design does not have this feature. It is shown that the proposed 12-bit fixed-width post-truncated radix-8 generic multiplier design involves 24.7% less ADP than the existing post-truncated radix-4 multiplier designs when both the multiplier designs are configured for constant multiplication. Block lifting 2-D DWT structure involves several multipliers. But, the interesting fact is that many of these multipliers share a common input operand. To take advantage of this feature, the proposed radix-8 generic constant multiplier is considered instead of radix-4 Booth multiplier for the block 2-D DWT structure. The block-based lifting 2-D DWT structure proposed in Chapter 3 is synthesized using the proposed radix-8 generic-constant fixed-width multiplier to demonstrate the effectiveness of proposed scheme. ASIC synthesis result shows that the lifting 2-D DWT structure of block size 16 and word length 12 offers 19.3% ADP saving and 11.5% power saving when the constant multipliers are implemented using the proposed radix-8 multiplier design instead of the existing radix-4 multiplier design. Compared with the existing block lifting 2-D DWT structure of [Hu *et.al* (2013)], the proposed block lifting 2-D DWT structure offers a saving in 24% ADP and 10% power for J=3 and block size 64, and higher saving for higher block sizes.

Chapter 6 discusses the look-up-table (LUT) based design for sign multiplication. Several design schemes have been proposed in the technical literature for efficient realization of LUT-based multiplier design. However, these LUT-multiplier designs are proposed for multiplication of unsigned numbers. The LUT must store both the positive and negative partial product terms for sign multiplication and there should be proper mechanism to read these positive and negative partial product terms from the LUT. Booth encoding scheme is used to design the LUT contents of signed multiplication. In general, (w/r) small size LUTs of size (2^p) words are required to perform w-bit sign multiplication, where r=(p-1). Symmetric and anti-symmetric coding schemes are used to reduce the LUT size. However, symmetric and anti-symmetric coding schemes involve separate address generator logic to fetch the correct partial-product from the LUT. LUT multiplier design of 12-bit multiplication using 8-words and 4-words LUT is presented. The proposed LUT multiplier is used design the lifting 1-D DWT structure which involves sign

multiplication. The lifting 1-D DWT design is used as a building block in the block-based lifting 2-D DWT structure presented in Chapter-4 to find a LUT-multiplier based design. ASIC synthesis result shows that the proposed LUT-multiplier based structure for block-size 16 and 64 involves 49% and 44% less MCP, 38% and 15% less area than those of the proposed radix-8 constant multiplier-based structure, respectively. Compared with the existing similar structure of [Hu *et. al* (2013)], the proposed LUT-multiplier based structure for block-size 16 and 64 involves 93% and 63% less ADP, and dissipates 13% and 21% less power, respectively. The proposed LUT-multiplier based structure for block-size 16 and 64 involves 92% and 52% less ADP, and consumes 8% and 13% less power than those of proposed radix-8 constant multiplier-based structure, respectively. Therefore, the proposed LUT-multiplier offers an efficient hardware design for high-speed implementation of lifting 2-D DWT.