Improved Memory Efficient Computing Unit DWT Architecture For Satellite Images

A. Azhagu Jaisudhan Pazhani, P. Gunasekaran and A. Rameshbabu
Department of ECE, Ramco Institute of Technology, Rajapalayam, India
E-mail: alagujaisudhan@gmail.com, mailtogunasekar@gmail.com, rameshbabu@ritrjpm.ac.in

Keywords: VLSI architecture, look-up table, distributed arithmetic, DWT, 2D, ROM, memory

Received: February 21, 2024

The 2D Discrete Wavelet Transform is a signal transform that is frequently used in picture and video compression. It is a computationally costly signal transform. VLSI implementation of 2D DWT is susceptible to a set of restrictions such as area and power consumption due to its increasing use in high data rate communication and storage in portable and handheld devices. The Distributed Arithmetic architecture is one of several architectures for constraint-driven VLSI implementation of 2D DWT that have been developed in recent years. The Distributed Arithmetic architecture is used efficiently to execute inner product computations, eliminating the need for multiplication and increasing computation speed. Filtering is the most power-intensive process in DWT, and multipliers are more expensive, so in Distributed Arithmetic architecture, multipliers are substituted with shifts and ROM lookup tables. However, as the number of filter coefficients grows, the size of the ROM look-up table grows, which can be decreased using the lookup table compression technique. In this paper, an Improved Memory Efficient Distributed Arithmetic Architecture for DWT has been proposed. The look-up table is used to stock the inner product values and then compressed. The performance of the improved LUT compressed algorithm is superior than the existing technique.

Povzetek: Predlagana je optimizirana pomnilniško učinkovita VLSI arhitektura za 2D DWT pri obdelavi satelitskih slik. Z uporabo porazdeljene aritmetike in stiskanja LUT zmanjša stroške računanja, izboljša hitrost in učinkovitost za aplikacije z visoko hitrostjo prenosa podatkov.

1 Introduction

Wavelet-based approaches are used to tackle complicated problems in math and engineering, with current applications including data compression, signal processing, image processing, pattern recognition, computer graphics, aeroplane and submarine detection, and other medical imaging technologies. A wavelet is an orthogonal function that may be applied to a limited set of data in the sense of the Discrete Wavelet Transform (DWT).

Mohanty B.K. Meher P.K. introduced a distributed arithmetic (DA) formulation for DWT computation utilising 9/7 filters in 2009, and transferred it to bit-parallel and bit-serial architectures for high-throughput and low-hardware implementations, respectively. For low-hardware solutions, the bit-serial structure processes the input vector's bit-slices in a serial fashion, whereas the bit-parallel structure processes all the bit-slices in parallel for high-throughput computing. The hardware usage efficiency of the bit-parallel structure is 100 percent. The suggested DA DWT structure has a much greater throughput rate and requires less area-delay product than conventional multiplier-less arrangements.

To process N-bit input operands, the fundamental serial architecture needs N clock cycles [3]. The primary disadvantage of the serial DA design is that it consumes more clock cycles and the filter's performance is slow. To expedite the procedure, it is preferable to apply the DA in parallel. The input data is separated into even and odd samples based on their location in the parallel implementation. Even samples convolve with even and odd filter coefficients, while odd samples convolve with the same set of coefficients at the same time [2]. The result is achieved concurrently for both even and odd input samples. The number of clock cycles is lowered, resulting in faster processing and less memory.

Distributed arithmetic calculations are bit-serial in nature in their most evident and direct form, i.e., each bit of the input samples must be indexed before a new output sample becomes available. When the input samples are represented with B bits of accuracy, an inner-product computation takes B clock cycles to complete. By replicating the LUT and adder tree, a parallel realisation of distributed arithmetic allows multiple bits to be processed in one clock cycle. The odd bits are sent to one LUT and adder tree in a 2-bit parallel implementation, while the even bits are fed to an identical tree. To suitably weight the outcome, the bit partials are left shifted and added to the even partials before aggregating the aggregate. All input bits can be calculated in parallel and then concatenated in a shifting adder tree in the extreme scenario [4].

An LUT, a cascade of shift registers, and a scaling accumulator make up the distributed arithmetic implementation of the Daubechies 8-tap wavelet FIR filter. All potential sums of the Daubechies 8-tap wavelet coefficients are stored in the LUT. The bit-wide output is delivered to the bit serial shift register cascade, one bit at a time, as the input sample is serialised. The input sample is stored in a bit-serial format in the cascade, which is then utilised to generate the requisite inner-product computation. The shift register cascade's bit outputs are utilised as address inputs to the LUT. The scaling accumulator adds together partial LUT results to generate a final result at the filter output port.

The benefit of utilising DA for a wavelet with a greater number of coefficients, on the other hand, may be lost over time due to a huge rise in memory size. The needed number of table entries is 2n. As the number of filter coefficients 'n' rises, the size of the look-up database grows exponentially.

A recent 2D DWT implementation on the NVidia GeForce GTX TITAN Black GPU was proposed in [7]. The authors of the paper [7] used a register-based technique to propose their DWT algorithm, which they claimed was four times quicker than existing GPU-based software implementations of DWT.

Darji et al. [8] presented a lifting DWT-based multiplier-less 1D/2D DWT architecture. They employed an innovative z-scanning method to reduce the transposing buffer size to 0 by using an innovative z-scanning method. Their temporal buffer size, on the other hand, is proportional to the number of input data points. Their requirement for adders is likewise quite great. Other newer methods may be able to outperform their architecture in terms of real-time image decomposition. 9/7 and 5/3 filter architectures were proposed by Meher et al. [9]. They offered 9/7 and 5/3 architectures with and without pipelines, as well as reconfigurable 9/7 and 5/3 systems. They concentrated on drastically lowering the size of the area and memory. Despite the fact that their design is space-efficient and their working speed is sufficient, there is still room to reduce their CP and thus increase the maximum operating frequency, which is a critical design component for real-time signal processing.

A multiplier-less lifting-based 2D DWT architecture was proposed in the work [10]. A flipping-based 2D DWT architecture was also presented in the same paper [10]. The inherent low critical-path delay of flipping-based architecture might be realised utilising lifting-based DWT design, according to the paper [10]. To validate the contributions, both designs were compared to other existing works. Despite the fact that the designs provided in [10] claim to greatly minimise critical-path delays, the critical-path delays of both lifting- and flipping-based architectures are significantly higher than any convolutional DWT architecture. As a result, there is plenty of room for improving timing performance.

In the work of Hegde et al. [11], the authors proposed one lifting- and flipping-based DWT architecture which is memory and power efficient. They used area consumption, critical-path delay, and power consumption as the main performance metrics. They proposed ‘look-up table’ (LUT)-based multiplier to reduce area and critical-path delay. They developed the architecture using gate-level HDL language and provided the ASIC implementation details. By proposing LUT-based multiplier, they successfully achieved to reduce the critical-path delay and area consumption of their multiplier than any conventional popular multiplier. However, they did not completely omit multipliers from their designs. Therefore, their design’s critical-path delay and power consumption are greater than any other multiplierless design. Moreover, LUT-based design uses a lot of registers or memory. Therefore, their design is also memory extensive.

We are now concentrating on briefly mentioning some of the most current works in the domain of DWT architectural design, having discussed some of the most recent and benchmark works in the subject. The authors introduced 1D/2D DWT architectures based on floating-point multiply and accumulator circuit' (MAC) units in their paper [12]. The 45 nm CMOS technology was used to implement the design. Though the validation and verification of the work is commendable, the performance in terms of critical-path delay, CT, and memory consumption should be improved further.

The study given in [13] is about the LeGall 5/3 DWT filter's DA-based DWT architecture. The work was implemented on an Altera FPGA, and the design's quality was compared to that of previous DWT-based works to demonstrate its superiority. However, there is still a lot of room for improvement in terms of area usage, power consumption, and operation speed with the DWT architecture. The authors of the paper [14] described a LeGall 5/3 DWT filter with a 1D DWT architecture based on 'canonical sign digit' (CSD)-based DA.

The authors used the CSD-based DA approach to propose a hardware-efficient DWT architecture that only required seven adders, a few shift registers, and multiplexers. However, their clock period is 100 ns [14]. This means that the working frequency of their design is only 10 MHz, which is far too low for many real-time applications. The work of [15] offered another major and current DWT architecture. A dual-memory controller-based 2D DWT architecture with a focus on real-time image processing was presented in the study [15]. The design's memory requirements were said to be streamlined to allow for real-time image processing.

An architecture that reduces the number of adders in a 1D Daub-4 filter module architecture and enhances the conventional Daub-4 very large-scale integration (VLSI) architecture design was proposed by Tiancai Lan et al [16]. The input image has a size of N × N matrix, and the output result is saved in the TM. Four sub-bands are obtained by reading the high and low frequencies one at a time to the second Daub-4 filter following the first Daub-4 filter's process.

Hussin et al. [17] proposed the 2D DWT and Huffman encoding for image compression. Once the input image has been chosen, the first step begins with RGB layer division. Next, superfluous image data at each RGB layer is eliminated using the lossy compression (DWT) technique. The output of the DWT process is then encoded and stored using lossless compression (the Huffman encoding approach).

The major purpose of this study is to create a DWT with a memory-efficient multiplier-less architecture. In DWT filtering, the distributed arithmetic architecture is used to produce multiplier-less computing. The size of the ROM look-up table increases when the filter coefficients rise in DWT with DA architecture, which can be lowered by employing a more effective LUT compression mechanism.

The size of the LUT can be lowered by counting the number of toggles between each pair of entries and compressing the result. The idea behind compressing the table is to reduce the amount of bit transitions per column as much as possible, then save the indices just where a bit toggling occurs rather than the entire column. Using the look-up table decoding approach, the needed inner product value is created from the compressed look-up table.

The following is a breakdown of the paper's structure. The DA architecture for DWT implementation was covered in part II. The suggested DA-based DWT architecture with better compression algorithm is described in Section III. In section IV, the findings and debates are discussed. Section V brings the paper to a close.

2 Distributed arithmetic architecrure for dwt implementation

FPGA implementation may be difficult due to their lack of arithmetic capabilities compared to general-purpose DSP processors. The reprogrammable configuration of FPGA is, nevertheless, its most significant benefit. Field Programmable Gate Arrays (FPGAs) are utilized in this study to implement DWT in hardware. With a large reduction in calculation time, DWT gives enough information for analysis and synthesis of the original signal.

The DA-based DWT has several uses in science, engineering, mathematics, and computer science. The use of DWT as an analogue filter bank in biomedical signal processing for the creation of low-power pacemakers, as well as in ultra-wideband wireless communications, is demonstrated.

To disguise the multiplications, DA is a bit level rearrangement of a multiply accumulate. It's a useful strategy for shrinking parallel hardware multiply accumulates that's ideally suited to FPGA designs. Since its introduction over two decades ago, DA has been frequently employed in VLSI implementations of DSP systems. The majority of these applications rely heavily on computing, with multiplication and/or addition being the most common operations. The key benefit of the distributed arithmetic technique is that it speeds up the multiply process by computing and storing all potential medium values in a ROM. After that, the input data may be used to address the memory and the result directly.

Formulation of algorithm

An illustration of normal Multiply Accumulate (MAC) operation

$$y = A_1X_1 + A_2X_2 + \ldots\ldots\ldots\ldots A_iX_i \tag{1}$$

A_i = Coefficient, X_i = Input

Distributed arithmetic implementation of DWT

Let Xk be a N-bits scrambled 2's complement number $|X_k|<1$

X_k: $\{b_{k0}, b_{k1}, b_{k2}\ldots\ldots, b_{k(N-1)}\}$,

Where b_{k0} is the sign bit

X_k is expressed as

$$X_k = -b_{k0} + \sum_{n}^{N} \tag{2}$$

Substitute equation (2) in equation (1),

$$y = \sum_{k=1}^{k} A_k + \sum_{n}^{N}$$

$$y = \sum_{k=1}^{k} b_{k0}A_k + \sum_{k=1}^{k} A_k \sum_{n=1}^{N-1}(A_k b_{kn})\, 2^{-n}$$

$$y = -\sum_{k=1}^{k} b_{k0}A_k + \sum_{k=1}^{k}\sum_{n=1}^{N-1}(A_k b_{kn})2^{-n} \tag{3}$$

Expanding this part

$$y = -\sum_{k-1}^{k} b_{k0}A_k + \sum_{k=1}^{k}(A_k b_{k1})2^{-1} + (A_k b_{k2})2^{-2} + \cdots + \left(A_k b_{(N-1)}\right)2^{-(N-1)} \tag{4}$$

$$\begin{aligned} y = -[b_{10}A_1 + b_{20}A_2 + \cdots + b_{k0}A_k] \\ + \left[(b_{11}A_1)2^{-1} + (b_{12}A_1)2^{-2} \right. \\ \left. + \cdots + b_{1(N-1)}A_1 2^{-(N-1)}\right] + \cdots \\ + \left[(b_{k1}A_k)2^{-k} + (b_{k2}A_k)2^{-k} \right. \\ \left. + \cdots (b_{k(N-1)}A_k)2^{-(N-1)}\right] \end{aligned}$$

$$y = -\sum_{k=1}^{k} b_{k0}\, A_k + \sum_{n=1}^{N-1}[\, b_{1n}A_1 + b_{2n}A_2 + \ldots + b_{kn}A_k]\, 2^{-n} \tag{5}$$

$$y = -\sum_{k=1}^{k} A_k(b_{k0}) + \sum_{n=1}^{N-1}\left[\sum_{k=1}^{k-1} A_k b_{kn}\right]\, 2^{-n} \tag{6}$$

Because each b_{kn} can only take on values of 0 and 1, there are only 2k potential possibilities. The memory holds the result y after N such cycles.

Hardware reduction in DA method

Figure 2.1 gives the hardware realization of the original equation (3) and for this original equation, the hardware utilization is high. The DA approach decreases hardware use, allowing the operation to run faster.

$$y = -\sum_{k=1}^{k} A_k(b_{k0}) + \sum_{k=1}^{k}\sum_{n=1}^{N-1}(A_k b_{kn})\, 2^{-n} \quad (7)$$

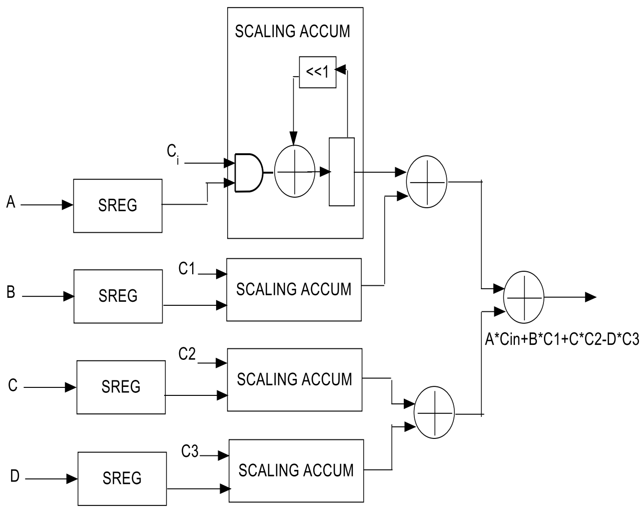


Figure 2.1: Hardware utilization for original equation

Figure 2.2 shows the hardware utilization in bit level rearrangement. In that hardware is reduced compared to original equation

$$y = -\sum_{k=1}^{k} A_k(b_{k0}) + \sum_{n=1}^{N-1}\left[\sum_{k=1}^{k-1} A_k b_{kn}\right] 2^{-n} \quad (8)$$

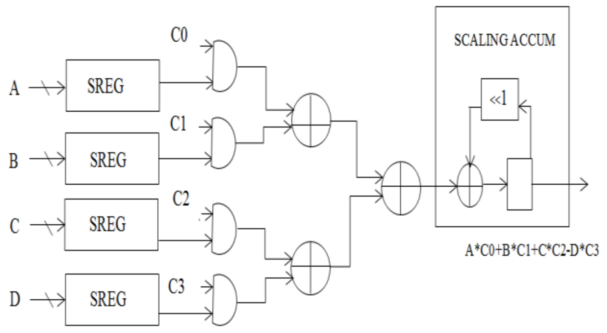


Figure 2.2: Hardware utilization in bit level rearrangement

DA architecture

The LUT, Shift registers, and scaling accumulator make up the DA architecture of a FIR filter. Various sums of the four coefficients make up the LUT data. The operands are loaded into the registers through a register chain in the shift registers. Depending on whether a serial or parallel architecture is used, the operands are then shifted 'n' bits at a time. In the scaling accumulator, the output of the DA LUT is added to the scaled output. It's made with an M-bit adder and a N+M-bit shift register at the output.

Serial DA architecture

As illustrated in Figure 2.3, the basic serial architecture requires N clock cycles to handle N-bit input operands. The LUT, adder tree, and scaling accumulator are all part of the critical path in the DA architecture, which runs from the input shift register to the output. The critical path delay is dominated by adder delays without the pipeline registers. When the design is fully pipelined, the significant fan-out loading delay incurred at the output of the shift register feeding the DA LUT inputs entirely masks the adder delays. If the loading factor is taken into account, the adder delays dominate the critical route latency, which may be considerably reduced by applying the technique outlined in. However, there will be little benefit from adopting quicker adder stages until the fan-out delays are addressed.

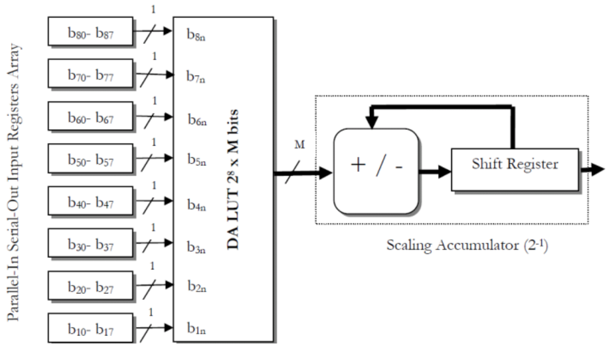


Figure 2.3: Serial DA architecture

The implementation findings show that by using parallelism with more than one bit at a time, the performance of DA systems may go up virtually linearly. Adding parallelism is the same as repeating the fundamental structure as many times as needed, each of which may function independently without clock frequency deterioration caused by pipelining.

Due to pipelining, the frequency of both operations stays the same. Furthermore, because each stage of the DA calculation is only a single basic FPGA element, the highest potential clock frequency for a particular FPGA device may be exploited. The main drawback of the serial DA architecture is, it requires more clock cycles and the speed of filter is low.

Parallel DA architecture

The procedure will be slower because the DA architecture is bit serial in nature. A parallel distributed arithmetic architecture is built to speed up the procedure [4]. Figure 2.4 depicts the parallel DA architecture. The input data is separated into even and odd samples based on their location in parallel implementation. Filter coefficients are also divided into even and odd samples. Even samples convolve with even and odd filter coefficients, while odd samples convolve with the same coefficients at the same time.

It is possible to receive results for both even and odd samples of input at the same time. The number of clock cycles is lowered, resulting in faster processing and less memory. The registers are loaded with the input values for each cycle, and then the reloading procedure to registers is enabled for the following set of cycles. The serial shift register, which must access the look-up table, will receive the input x[n]. The old value will be moved into the next register when the new input arrives in the first register. Similarly, as new values enter registers, the old values are removed from the registers.

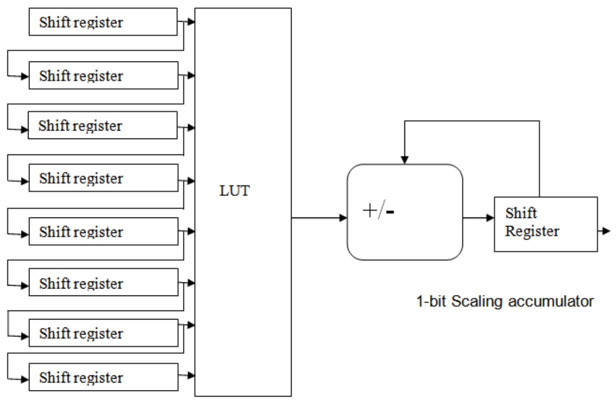


Figure 2.4: Parallel DA architecture

Consider the bit locations and retrieve the values of inputs from that bit position to get the address from the input values. Consider the LSBs of all serial registers to determine the initial address, for example. The initial position value will be generated using this address. Obtain all of the bit position addresses and the accompanying values from the look-up table in the same manner. Shift the values by the bit position value and provide them to the adder during adding. Finally, the output, which is the convolution of the filter coefficients and the inputs, will be generated.

Both the high-pass and low-pass filters will be built using the same design. If the input is 8 bits long, the convolved value takes 8 clock cycles to compute. The filter operations are stated using floating point arithmetic while computing the wavelet coefficients. In practice, though, integer arithmetic is employed. The filter coefficients are shortened as a result. The precision of the calculated coefficients suffers as a result of this reduction.

3 Proposed memory efficient da architecture for dwt

Implementing DWT with DA architecture may improve computation speed, but it will also increase memory size as the number of wavelet coefficients grows. The multi-level decomposition requires a high level of DWT implementation complexity. As a result, the benefit of employing DA will be effectively gone. The size of the look-up table in the DA architecture for DWT is reduced using a novel way. A table compression approach, as shown in Figure 3.1, can be used to minimize the size of the look up table required to record all possible combinations of input in DA architecture. The algorithm for compressing the LUT is the same as that used to save a processor's assembly language instructions [5]. A similar approach can be used to reduce the number of LUTs in DA architecture [1].

After going through high pass and low pass filters, the DWT coefficients are created. The filter coefficients are convolved with inputs to perform the filter operation with N input variables. The coefficients are fixed in this case. Binary can be used to represent inputs. The inputs are scaled to have absolute values less than one. In ROM look-up tables, the inner product for several inputs can be computed and saved in advance. If there are n wavelet coefficients, the look-up table will be 2n. All LSBs are assumed to be the first to receive data. Similarly, all bit positions are determined, and the look-up database is used to determine the appropriate values.

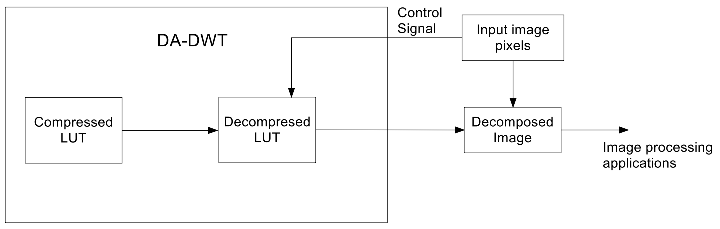


Figure 3.1: Memory reduced DA architecture

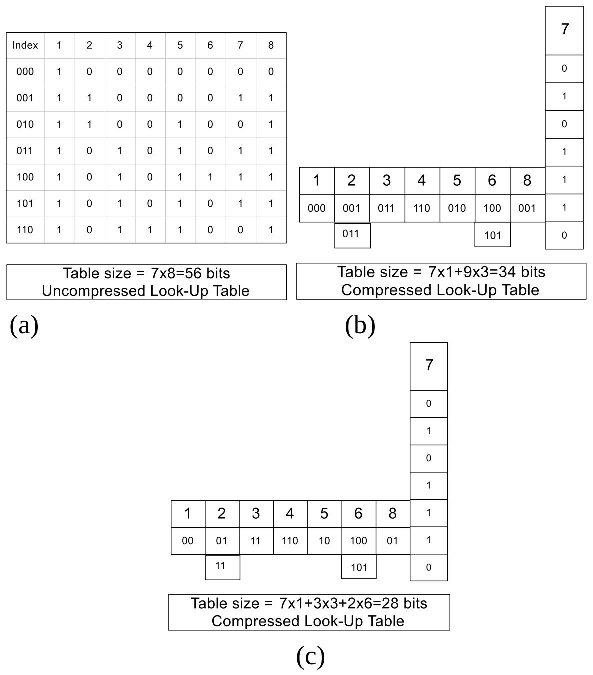
LUT encoding algorithm

The size of the LUT can be lowered by counting the number of toggles between each entry and compressing them [1]. The idea behind compressing the table is to reduce the amount of bit transitions per column as much as possible, then save the indices just where a bit toggling occurs rather than the entire column. Figure 2 displays an example of a LUT with seven symbols, each with eight bits. The table is 56 bits in size (before compression). There are 8 distinct binary words in the table, with an index length of 3 bits. As a result, if the column contains no more than two transitions, it can be compressed. Seven columns will be compressed in this example, but one column will remain uncompressed. After compression, the table's size is reduced to 34 bits (from 56 bits before). FPGA RAM blocks are used to hold the compressed table.

If the lookup table compression is modified using the following steps auxiliary compression can be achieved. The steps to be incorporated in the modified lookup table compression are as follows:

```
Total number of locations:  LUT size: $2^n$
        if index< 2n/2
          use rep with (n-1)-bits
        else
          n-bits
```

Using the above steps the table is further compressed as shown in Figure 3.2. Hence the LUT compression of 28 bits can be achieved.



Index	1	2	3	4	5	6	7	8
000	1	0	0	0	0	0	0	0
001	1	1	0	0	0	0	1	1
010	1	1	0	0	1	0	0	1
011	1	0	1	0	1	0	1	1
100	1	0	1	0	1	1	1	1
101	1	0	1	0	1	0	1	1
110	1	0	1	1	1	0	0	1

Figure 3.2: a) Uncompressed LUT b) Existing Compressed LUT c) Improved LUT Compression

Using the LUT compression methodology and the improved LUT compression, the size of the compressed LUT is decreased by 39.28% and 50%, respectively. Thus the modified LUT can be an efficient method for compressing the DWT coefficients.

LUT decoding algorithm

The needed inner product value is created from the compressed look-up table in this decoding process. When a certain input to a look-up table comes, it determines its location in each compressed table column.

- If the input is greater or equal to the compressed look-up table value, then generate '1'
- If the input is lesser to the compressed look-up table value, then generate '0'

The uncompressed table columns' original bits are received straight from the ROM.

DA DWT architecture

The parallel implementation of DA architecture is exposed in Figure 3.3. The input data is separated into even and odd samples based on their location in parallel implementation. As a result, even samples convolve with even and odd filter coefficients, whereas odd samples convolve with the same set of coefficients. The results for both even and odd samples of input are obtained. Here number of clock cycles are abridged which results in increased speed and decreased memory.

To access the LUT, the same number of registers must be used for accessing filter quantities. The data will be sent into a serial shift register, which will need to consult the look-up table. The old value will be moved into the next register when the new input arrives in the first register.

Similarly, when new values enter registers, the old values are removed from the registers by examining bit positions and determining the values of inputs based on that bit position. Finally, all bit position addresses are obtained from the look-up table and are given as input to adder by shifting its values. Finally, the result, which is the convolution of the filter coefficients and the inputs, will be achieved.

The DA architecture speeds up the operation by lowering memory use, but as the size of the look-up table grows larger, the decoding process becomes more time demanding.

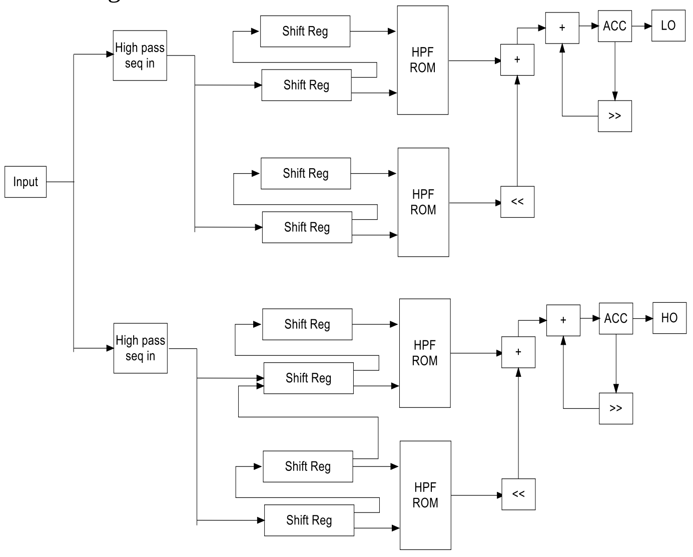


Figure 3.3: DA DWT architecture

4 Results and discussion

In this work, the distributed arithmetic architecture for DWT is designed and simulated using Verilog in MODEL SIM 0.61xd. Simulation verifies the functionality of both high pass and low pass filters. Then it is synthesized into Spartan3E FPGA platform using Xilinx ISE Design Suite 13.2.

Simulation and synthesized results for single level DWT

The synthesized results for the suggested design are presented in Figure 4.1 for the low pass and high pass filters. The Parallel DA-DWT Architecture reads input vectors from a ROM. The shredded outputs are saved, and simulated waveforms are used to illustrate them.

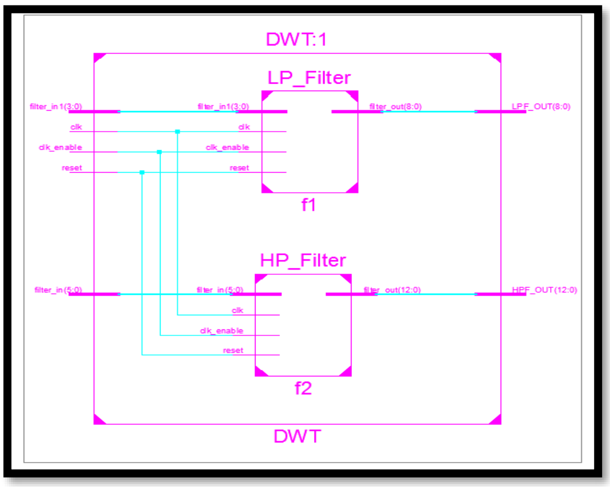


Figure 4.1: Synthesized result of single level DWT

Comparison of uncompressed and compressed DA ROM Size

The Table I give the memory size of the look-up table for low pass and high pass filter with uncompressed DA is reduced to 60% and 40% compared to compressed DA respectively. The proposed technique gives the compression efficiency of 50% for low pass and 72% for high pass filter whereas the existing technique gives the compression efficiency of 60% for low pass and 63% for high pass filter.

Table 1: Comparison of distributed arithmetic schemes

Architecture	**Memory size (ROM) Lowpass filter**	**Memory size (ROM) Highpass filter**
Uncompressed DA [1]	80 bits	256 bits
Existing Compressed DA [8]	48 bits	96 bits
Proposed Improved DA	40 bits	72 bits

Performance comparision

The performance comparison of different architecture for DWT is given in Table II.

Table 2: Performance comparison of various DWT architecture

Scheme	**Level = 1**
Filter implementation [9]	16 multipliers
Lifting implementation [8]	6 multipliers
Serial DA based implementation [14]	43 adders
Compressed DA based implementation	4 adders 4 subtractors

The Table II gives the requirement of adder and multiplier for different architectures to design DWT. The filter based implementation involves direct multiplication for inner product calculation in the filter, which requires more number of multipliers. The filter based implementation of DWT for single level requires 16 multipliers. The lifting scheme is implemented to reduce the arithmetic computation which requires 6 multipliers to implement the DWT for single level. The serial DA based architecture involves multiplier less operation for inner product calculation; it requires 43 adders to design single level DWT. The proposed method reduces up to 4 adders and 4 subtractors.

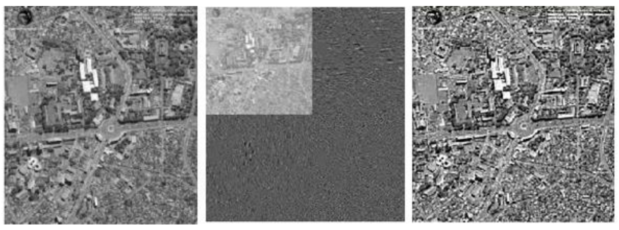
Hardware utilization comparision

The Table III gives the device utilization of DA architecture. It is less compared to convolution based architecture. The DA architecture uses LUT instead of multiplier for MAC unit to get inner product calculations.

Table 3: Hardware utilization comparison

LOGIC UTILIZATION	**CONVOLUATION BASED ARCHITECTURE (one level) [6]**	**DA BASED ARCHITECTURE (one level)**
Number of slices Flip Flops	47	102
Number of 4 input LUTs	294	115
Total number of occupied slices	209	91
Number of bonded IOBs	91	35
Number of BUFGMUXs	1	1

Images transform comparisons using 2D-DWT



(a) Input image (b) DWT Processed image (c) Output image

5 Conclusion

The memory efficient DA architecture for discrete wavelet transform is implemented using Spartan 3E FPGA. The DA architecture is built on the Look-up table technique for effective inner product computation. When using DA architecture to implement DWT, the size of the ROM look-up table grows as the filter coefficients grow. The revised look-up table compression technique reduces the size of the LUT up to 115. The compressed LUT is kept in the FPGA's ROM. Data can be decrypted by decompressing the table while conducting DWT calculation. The memory-based method enables the Parallel DA-DWT to achieve high computation speeds while using a little silicon area by replacing multipliers with compact ROM tables. Saving adders, quick processing time, regular flow of data, and minimal control complexity are all advantages of the suggested architecture, making it suited for image compression systems. The proposed method reduces the memory size from 80 bits to 40 bits for LPF and 256 bits to 72 bits for HPF, but the decoding process will be time consuming while increasing the filter coefficients. The focus of future research will be on improving the speed of retrieval from LUTs and quick decoding.

Author contributions statement

"All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by [A. Azhagu Jaisudhan Pazhani], [P. Gunasekaran] and [A. Rameshbabu]. The first draft of the manuscript was written by [A. Azhagu Jaisudhan Pazhani]. All authors read and approved the final manuscript."

Conflict of interest

There is no conflict of interest in this paper regarding publication.

Data availability statement

The data that supports the findings of this study are available within the article.

Funding

No funding was received for this study.

References

[1] Remya Ajai A S, Nithin Nagaraj (2012), "A Novel methodology For Memory Reduction in Distributed Arithmetic Based DWT" International Conference on Communication Technology and System Design procedia Engineering 30, pp. 226-233.

[2] K.B. Sowmya, Dr. SavitaSonoli and M. Nagabushanam (2012), "Implementation of Parallel DA Technique for DWT-IDWT on FPGA for Image Compression", International Journal of Power Systems and Integrated Circuits, Vol. 2, pp 143 – 148.

[3] Mohanty B.K. Meher P.K (2009), "Efficient multiplier less designs for 1-D DWT using 9/7 filters based on distributed arithmetic", Dept. of Electronics and Communication Engineering., Jaypee Inst. of Eng. & Technol., Guna district, India, vol 1.1, no 6, pp 364 – 367.

[4] Al-Haj AM (2005), "An FPGA-Based Parallel Distributed Arithmetic Implementation of the 1-D Discrete Wavelet Transform," Informatica vol. 29, no 2, pp 241-247.

[5] Xixin Cao, QingqingXie, ChunganPeng, Qingchun Wang (1996), "An Efficient VLSI Implementation of Distributed Architecture for DWT" IEEE Transaction VLSI System, vol. 2, no 6, pp. 521-543.

[6] Basant kumar mohanty, Pramod kumar (2013), "Memory-Efficient High-Speed Convolution-Based Generic Structure for Multilevel 2-D DWT" IEEE Transaction on circuits and systems for video technology, vol. 23, No. 2.

[7] Enfedaque, P, Auli-Llinas F, Moure J.C (2014), "Implementation of the DWT in a GPU through a register-based strategy" IEEE Trans. Parallel Distrib. Syst. 26(12), 3394–3406.

[8] Darji A, Arun R, Merchant S.N, Chandorkar A (2015), "Multiplierless pipeline architecture for lifting-based two-dimensional discrete wavelet transform" IET Comput. Dig. Tech. 9(2), 113–123.

[9] Meher P.K, Mohanty B.K., Swamy M.M.S (2015), "Low-area and low-power reconfigurable architecture for convolution-based 1-D DWT using 9/7 and 5/3 filters" 28th International Conference on VLSI Design, Bangalore, pp. 327–332.

[10] Mohanty B.., Meher P.K, Srikanthan T (2015), "Critical-path optimization for efficient hardware realization of lifting and flipping DWTs" IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, pp. 1186–1189.

[11] Hegde G, Reddy K.S, Ramesh, T.K.S (2018), "A new approach for 1-D and 2-D DWT architectures using LUT based lifting and flipping cell", AEU Int. J. Electron. Commun. 97, 165–177.

[12] Mohamed Asan Basiri M, Noor Mahammad S (2018), "An efficient VLSI architecture for convolution-based DWT using MAC", 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems, Pune, pp. 271–276.

[13] Aziz, F, Javed S, Iftikhar Gardezi S.E, Jabbar Younis C, Alam M (2018), "Design and implementation of efficient DA architecture for LeGall 5/3 DWT", International Symposium on Recent Advances in Electrical Engineering (RAEE), Islamabad, pp. 1–5.

[14] Gardezi S.E.I, Aziz F, Javed S, Younis C.J, Alam M, Massoud Y (2019), "Design and VLSI implementation of CSD based DA architecture for 5/3 DWT", 16th IEEE International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp.548–552.

[15] Naik P, Guhilot H, Tigadi A, Ganesh P (2019), "Reconfigured VLSI architecture for discrete wavelet transform", Soft Computing and Signal Processing. Springer, Singapore, pp. 709–720.

[16] Tiancai Lan, Chih-Hsien Hsia, Po-Ting Lai, Hsien-Wei Tseng and Cheng-Fu Yang (2022), "Memory efficient Very Large-Scale Integration Architecture of 2D Algebraic-integer-based Daubechies Discrete Wavelet
Transform", Sensors and Materials, Vol. 34, No. 9 3623–3636.

[17] M.A. Hussin, F.A. Poad, A. Joret (2021), "A comparative study on the performance of DWT and huffman compression technique on a 2D signal", J. Electron. Voltage Appl. 2 (1) 11–19.