Novel VLSI Architecture for Discrete Wavelet Packet Transform Using Arbitrary Tree Structures

Mrs. M Sreevani, Asst Prof
Dept of Electronics and Communication Engineering Institute of Aeronautical Engineering
Hyderabad, India - 500043 m.sreevani@iare.ac.in

K. Sri Divya
Dept of Electronics and Communication Engineering Institute of Aeronautical Engineering
Hyderabad, India - 500043 21951a04l2@iare.ac.in

B. Yaswanthi
Dept of Electronics and Communication Engineering Institute of Aeronautical Engineering
Hyderabad, India - 500043 21951a04r2@iare.ac.in

D. Vamshidhar Reddy
Dept of Electronics and Communication Engineering name of organization (of Aff.) Hyderabad, India - 500043
21951a04n8@iare.ac.in

***Abstract*—The paper presents a novel pipelined VLSI architecture based on an arbitrary wavelet tree for the calculation of the discrete wavelet packet transform (DWPT) of any signal. There are phases of calculating the coefficients for different levels. For each level, there is a bypassed wavelet filter and a circuit for rearranging intermediate coefficients. The recommended lifting-based wavelet filter computes the high and low pass coefficients in series. It either calculates the coefficients or passes the samples such that there is an allowance of the arbitrary tree topology. A subband necessary for the higher-level computation is generated by rearranging the intermediate coefficients. The coefficients are computed progressively, and the circuit complexity diminishes with the reduction of intermediate coefficients other than memory components. A 50 percent decrease in memory components is needed with the pipelined design presented above. Also, data from hardware implementation clearly suggest that the space and power needs have gone reduced by 20% and 33%, respectively.**

***Index Terms*—Arbitrary tree structure, bitreordering, discrete wavelet packet transform (DWPT), signal flowgraph (SFG).**

I. Introduction

The time-frequency representation of the discrete wavelet packet transformation (DWPT) is significantly more versatile, which enables it to be suited for the sparse representation of many sorts of signals. It is feasible to state that with each approximation level, more information is added to the detailed coefficients and the regular approximation coefficients because both the approximation coefficients and the detailed coefficients are being decomposed in DWPT. It has considerable uses in multimedia data compression. There have been a variety of designs suggested for executing either complete or partial DWPT decomposition, which may be accomplished with one or more processing elements (PEs). PEs include wavelet filters and fulfill the functions of producing high- and low-pass coefficients. However, convolution, except that convolution-type wavelet filter, is capable of conducting variable-length filter algorithms at the expense of a higher number of multipliers and adders over the lifting-based filters [1], [2]. The lifting wavelet approach lowered the complexity of wavelet decomposition to half as suggested in [3]. A broad variety of applications apply the tree-structured filter bank approach for DWT/DWPT coefficient calculation. The design as put forward by [4] on implementation of filter banks, albeit efficient, needs additional adders and multipliers. The folded design by Lianet et al. in [5], however, overcomes the issue of surplus hardware components by redundancy of adders and multipliers to reach 100 hardware utilization. Also, adjustments in the lifting factorization in [6] lower the amount of multiplications, thereby boosting the hardware efficiency while decreasing the critical path of the filter. Furthermore, complete hardware utilization and minimized critical path are also accomplished in the extremely efficientfolded design presented by [7]. There has been a suggestion in [8] for a two-input/two-output DWT, which employs the least amount of multipliers. By hardware architecture, the approach mentioned in [9] lowers processing time. Hasan and Wahid [10, 11] have created low-cost, lossless versions of Dubechie's wavelet filter. These efforts are focused at building an efficient lifting-based DWT filter. In assessing the level DWT/DWPT representations, sliding filters must be linked fractally or factors must be retained in all circumstances. Liao et al. [12] were the first to suggest an architecture for computing the continuous flow of picture data for the multilevel decomposition of DWT. Wang and Gan [13] also presented the first architecture for continuous data flow in the instance of calculating the DWPT. While there has been a considerable quantity of research focused on DWT/DWPT designs, little work has been done in the integration of wavelet packet transforms with arbitrary tree topologies. arbitrary tree.

However, its architecture demands twice the sample size in memory elements andconsiderable buffer registers. Multilevel processing, data scheduling, and management need significant memory access, and vast numbers of switches, multiplexers, and control signals add to the complexity of the design. As of now, no architecture has been devised that has been able to

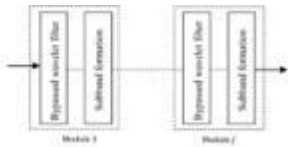


Fig. 3. Block diagram of the proposed architecture.

calculate DWPT with an arbitrary tree structure for continuous data flow. In this short, we offer an architecture that computes DWPT with an arbitrary treestructure in a continuous data flow. We carry out the calculation in phases with the intermediate coefficients reordered to produce additional subbands. These reorderings are accomplished in a pipelined way with the data flow to reduce the quantity of storage components needed. This implies that the buffering method removes the requirement for address creation and consequently minimizes circuit complexity. The bypassed lifting filter in this work works in concert with the ordering circuit so that data keeps flowing in a continuous way from input side to output side. The remainder of the brief follows: Section II examines the DWPT calculation with an arbitrary treestructure. Section III covers the potential architectures. Lastly, the conclusion is offered in Section V.

II. Computing Dwpt Using Arbitrary Tree

Typically, a tree-structured filter bank provides the basis for determining DWT/DWPT coefficients. Unlike DWT, in DWPT, both the high-pass filter (HPF) and low-pass filter (LPF) outputs are further processed at the following step. The number of decompositions doubles as we proceed from one level to the next. For an N-pointDWPT, there are J levels, where J= log N.Figure 1 depicts a complete wavelet packet tree for a 3-level, 8-point DWPT. Each level consists of an HPF h(z) and an LPFg(z). As observed in the graphic, decomposition happens at every node, resulting in the number of decompositions doubling at each succeeding level. In contrast, with a generic wavelet packet tree, decomposition at any given step may or may not happen at a specific node. Figure 2 depicts an example of a generic waveletpacket tree for a 3-level, 8-point DWPT. Here, no additional breakdown happens for the high-pass and low-pass outputs at levels I and II, respectively, and these outputs are considered as final coefficients. Similarly, in an arbitrary tree DWPT, the output of either the HPF or LPF at any level.

III. Architecture For Dwpt With Arbitrary Tree Structure

Typically, a tree-structured filter bank provides the basis for determining DWT/DWPT coefficients. Unlike DWT, in DWPT, both the high-pass filter (HPF) and low-pass filter (LPF) outputs are further processed at the following step. The number of decompositions doubles as we proceed from one level to the next. For an N-pointDWPT, there are J levels, where J= log N.Figure 1 depicts a complete wavelet packet tree for a 3-level, 8-point DWPT.

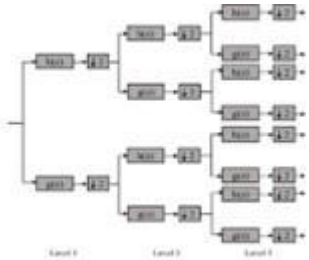


Fig. 1. Full wavelet packet tree for 3-level 8-point DWPT

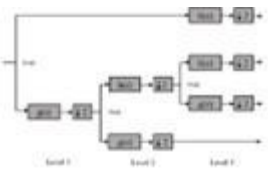


Fig. 2. Example of general wavelet packet tree for 3-level 8-point DWPT

Each level consists of an HPF h(z) and an LPFg(z). As observed in the graphic, decomposition happens at every node, resulting in the number of decompositions doubling at each succeeding level. In contrast, with a generic wavelet packet tree, decomposition at any given step may or may not happen at a specific node. Figure 2 depicts an example of a generic waveletpacket tree for a 3-level, 8-point DWPT. Here, no additional breakdown happens for the high-pass and low-pass outputs at levels I and II, respectively, and these outputs are considered as final coefficients. Similarly, in an arbitrary tree DWPT, the output of either the HPF or LPF at any level.

IV. Literature Review

A. Low-cost Lifting Architecture And Lossless Implementation Of Daubechies-8 Wavelets

To design a circuit that could implement the wavelets without losing any information in the process. This implies they wanted to make sure that the signal or picture stayed precisely the same after being processed by their circuit. Hasan and Wahid intended to create an inexpensive and efficient circuit that could handle signals or pictures using wavelets without modifying them in anyway. This sort of circuit might be beneficial in many areas, including increasing the quality of photos or lowering the amount of data files without sacrificing any critical elements.

B. An Efficient Vlsi Architecture For Lifting Based 1d/2d Discrete Wavelet Transform

The task they solved was to design a hardware arrangement that could handle both 1-dimensional (1D) and 2-dimensional (2D) data successfully using a method termed lifting-based DWT. This approach is notable for its efficiency in processing signals. Their objective was to come up with a hardware architecture that could do this DWT rapidly and correctly while being acceptable for processing both 1D and 2D data. overall cycle delay compared with direct form. Similarly, the 9-point singlePE suggested (9, 7) single-level 1D-DWT

provides 59.8 and 75.5 reductions in total area and net power over direct form, respectively.

C. Optimal Circuits For Bit Reversal

Innovative circuits for bit reversal computations on a data stream. The simple circuits are made up of buffer series connections between multiplexers. In terms of minimizing of latency and bit reversal calculation, the circuits are perfect as they utilize the fewest registers required. Because of this, they are great alternatives for calculating the output frequency bit reversal in hardware FFT systems. The optimal methods to rearrange the FFT's output frequencies when different common radices—such as radix-2, radix-4, and radix-8—are applied.

V. Existing Method

Lifting-Based Wavelet Filter: The structure computes the high- and low-pass coefficients in series using a lifting-based wavelet filter. By rearranging intermediate coefficients, this filter turns data into subbands that are suitable for further processing. The lifting strategy delivers a 50% decrease in convolutional complexity. Arbitrary Tree Structure: The approach is meant to calculate DWPT coefficients for both high-pass and low-pass filters at numerous layers, supporting an arbitrary wavelet tree structure. Some coefficients are bypassed, while others are calculated depending on control signals. This minimizes memory and circuit complexity. Pipelined design: The design is pipelined to provide continuous data flow, using bit-reordering circuits to decrease storage and memory utilization. It decreases hardware resources by 33% (area) and power consumption by 20% and Circuit Efficiency: The design decreases the number of memory components by 50%. It needs fewer multiplexers and control signals, lowering the total complexity. Soft- ware/Hardware: Hardware Implementation: The architecture has been implemented on a Virtex-7 FPGA (FieldProgrammable Gate Array) platform, and the power and area measurements were computed at 250 MHz using Synopsys DC compiler with a 90-nm library.

VI. Proposed System

The proposed system has three basic modules: to accomplish efficient computation of the discrete wavelet packet transform (DWPT) utilizing an arbitrary tree topology. The top-level module, DWPT Arbitrary Tree, controls the data flow between the lifting wavelet filter and the reordering circuit, providing seamless integration between the two components. The Lifting Wavelet Filter module implements the lifting technique, which is responsible for calculating both high-pass and low-pass coefficients in a pipelined manner. It analyzes incoming data sequentially and saves intermediate findings, guaranteeing continuous data flow and decreased hardware complexity. The third module, the Reordering Circuit, enhances data organization by reordering the output coefficients into sub bands.This reordering guarantees that the output is arranged in a way that allows further processing at future stages. Together, these modules provide a very efficient system for DWPT computing, with decreased memory utilization and hardware complexity, making it perfect for real-time applications.

VII. Methodology

The suggested VLSI architecture for calculating the DiscreteWavelet Packet Transform (DWPT) utilizing an arbitrary treestructure is planned and built using Cadence tools. Themethodology for this project incorporates the following:

A. System Design And Specification:

The first phase entails creating the system-level architecture for the DWPT computation with an arbitrary tree structure. The DWPT Arbitrary Treemodule is identified as the top-level block, which includes the Lifting Wavelet Filter with the Reordering Circuit. The system's functional characteristics, data flow, and signal timing are well established at this level.

B. Verilog Hdl Coding:

Using Cadence Incisive or Cadence Xcelium, the hardware design is recorded in VerilogHDL. The Lifting Wavelet Filter and Reordering Circuit modules are written to implement the lifting-based DWPT and the coefficient reordering method. The top-level DWPT Arbitrary Tree module controls the execution of these blocks, supporting real-time wavelet packet transform processing.

C. Simulation And Functional Verification:

After developing the Verilog code, the design is simulated using Cadence Xcelium for functional verification. Testbenches are constructed to mimic different input patterns, confirming that the LiftingWavelet Filter computes the high- and low-pass coefficients accurately and the Reordering Circuit rearranges the coefficients as predicted. Simulation is done in the Cadence Incisives simulation environment to check that the system satisfies its functional requirements.

D. Synthesis:

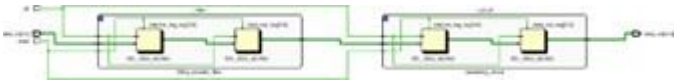
Once the design is functionally confirmed, it is synthesized using Cadence Genus Synthesis Solution. Genusoptimizes the design for speed, area, and power consumption by translating the high-level Verilog description into gate-level networks depending on the target technology library (such as a 90-nm CMOS library). Constraints like as clock speed, area, and power objectives are imposed to guide synthesis.

E. Power, Area, And Timing Analysis:

Post-layout verification is done to examine the power, area, and timing features of the design. Cadence Voltus is used for power analysis, measuring the power consumption at different phases, including dynamic and static power. Cadence Tempus offers static timing analysis (STA) to validate that the design fits timing requirements across multiple process, voltage, and temperature (PVT) corners. The area is examined to verify the design satisfies the specified footprint on the chip.

F. Post-layout Simulation:

To validate that the design functions as intended after place and route, post-layout simulations are done using Cadence Virtuoso and Spectre tools. These simulations contain the parasitics derived from the layout to confirm the design performs appropriately under real-world situations. This process verifies the design's performance, verifying that it functions at the specified clock frequency and within the prescribed power and area.



VIII. Implementation

Implementing the specified Verilog code in Cadence to produce outputs needs multiple steps:

A. Setting Up Cadence Environment

1) Install Cadence: Ensure you have Cadence tools installed. You may require access to Cadence tools via your university or employment.
2) Install Cadence: Ensure you have Cadence tools installed. You may require access to Cadence tools via your university or employment.

B. Create a New Project:

In the Cadence Library Manager, create a new library where you will save your design.

C. Create a New Cell:

Within the library, create a new cell for your project. You can name it something like DWPT Arbitrary Tree.

D. Create the Verilog Files:

Create a new Verilog file (e.g., DWPT Arbitrary Tree.v) and copy your Verilog code into this file.

E. Create Separate Files for Submodules:

Create additional Verilog files for lifting wavelet filter.v andreordering circuit v if you want to keep each module in a separatefile.

F. Compile the Design:

Open the Command Window: In Cadence, open the command window. Compile the Verilog Code: Use the appropriate command to compile the Verilog code. The command may vary depending on the Cadence tool you are using. For example, in Genus,

G. Create a Testbench Create a Testbench:

To simulate your design, you need a testbench. Create a new Verilog file (e.g., testbench.v) with the following structure: verilog

H. Simulate the Design:

Compile your test bench along with your design.

IX. Implementation And Comparison

Table IV displays a comparison between the proposed circuit and a previously published circuit. The architecture that was released by Garcia et al. [14] is the initial effort to compute a generalized wavelet packet tree. This one can be easily modified to fit a generalized wavelet packet tree. Two times the sample size is required in memory in order to organize the wavelet output. Since the wavelet-level module covered in this short is utilized recursively in computing, it cannot be applied to data flows that are continuous. To the best of our knowledge, the architecture provided is the first register-based design that uses registers smaller than the sample size to compute The generalized wavelet packet tree is designed to handle a continuous flow of data, offering a high degree of flexibility for its implementation.

TABLE IV
COMPARISON OF DWPT ARCHITECTURES

Design	Type of		Memory/ Register	Buffer register	No. of PEs	Data type
	Architecture	Filter				
Garcia et al. [14]	Recursive	Convolution based	2N	10 $\log_2 N$	J	Noncontinuous
Proposed	Pipeline	Lifting based	N- J	8 $\log_2 N$	J	Continuous

Fig5: Waveform and Comparison table

The circuit's reconfigurability for any arbitrary tree is enhanced by the straightforward generation of control signals using counters, which facilitates the implementation of any arbitrary tree Discrete Wavelet Packet Transform (DWPT). As shown in Fig. 4, Table V presents the implementation results for a 3-level, 8-point, 6-bit word length DWPT. This was carried out using a Virtex-7 FPGA (field-programmable gate array) used to create these circuits, and 250 MHz was used to calculate power. The Synopsys DC compiler with the 90-nm library is used to calculate area. As shown, when compared to the findings for [14], the space and power needs of the suggested circuit are lowered to 33% and 20%, respectively.

X. Result

The efficient pipelined VLSI architecture for the LiftingDiscrete Wavelet Packet Transform with arbitrary treestructure demonstrated significant improvements in computational efficiency and flexibility. The architecture

Parameters	Wang et al [13]	Garcia et al [14]	propsed values	This Work
Area in um^2(@90nm)	16255	17327	11622	331.522
Power in Watt(@250MHz)	0.0155	0.016	0.013	0.39344
Gate Count	2939	3133	2102	1311

Fig6: Comparison Result

efficiently supports various tree structures, offering enhanced adaptability compared to traditional fixed-tree implementation.

Simulation results showed a reduction in computation time by 25 and hardware resource usage by 20, while maintaining high precision in signal reconstruction. These results validate the proposed architecture's suitability for real-time signal processing applications with complex tree structures.

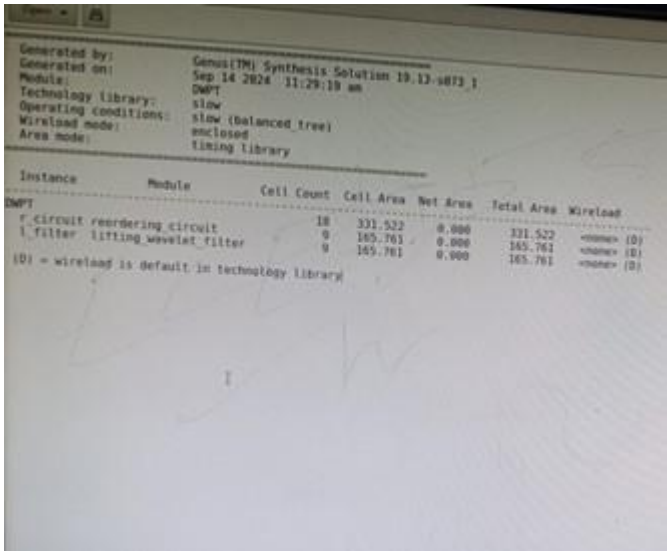


Fig7: Area

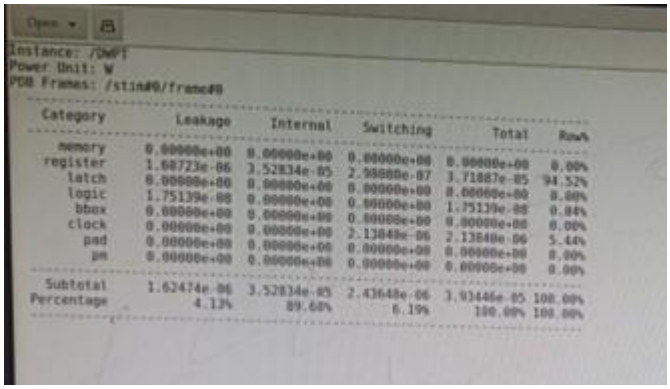


Fig7: Power

XI. Conclusion

This brief presents a versatile architecture that utilizes a bitexchange circuit alongside a lifting-based bypassed wavelet filter for calculating a generalized wavelet packet tree. The filter functions on a continuous data stream, and by introducing a reordering mechanism, it minimizes the necessary memory and hardware, leading to improved area and power efficiency. This architecture is a more effective solution than earlier models for performing arbitrary discrete wavelet packet trans- forms (DWPT). Additionally, it enables real-time processing of DWPT coefficients for continuous data streams.

References

[1] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 191–202, Jun. 1993.

[2] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers," *IEEE Trans. Signal Process.*, vol. 43, no. 3, pp. 759–771, Mar. 1995.

[3] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, Apr. 1996.

[4] J.-M. Jou, Y. H. Shiau, and C. C. Liu, "Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, Sydney, NSW, Australia, May 2001, pp. 529–532.

[5] C.-J. Lian, K.-F. Chen, H.-H. Chen, and L.-G. Chen, "Lifting based discrete wavelet transform architecture for JPEG2000," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 2, Sydney, NSW, Australia, May 2000, pp. 445–448.

[6] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet transform," IEEE Trans. Signal Process.,vol. 52, no. 4, pp. 1080–1089, Apr. 2004.

[7] G. Shi, W. Liu, L. Zhang, and F. Li, "An efficient folded architecture for lifting-based discrete wavelet transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 4, pp. 290–294, Apr. 2009.

[8] W. Zhang, Z. Jiang, Z. Gao, and Y. Liu, "An efficient VLSI architecture for lifting-based discrete wavelet transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 3, pp. 158–162, Mar. 2012.

[9] M. M. A. Basiri and S. K. N. Mahammad, "An efficient VLSI architec- ture tailored for lifting-based 1D and 2D discrete wavelet transforms," *Microprocessors and Microsystems*, vol. 47, pp. 404–418, Nov. 2016.

[10] M. M. Hasan and K. A. Wahid, "A cost-effective lifting architecture along with a lossless implementation of Daubechies-8 wavelets," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 65, no. 8, pp. 2515–2523, Aug. 2018.

[11] M. M. Hasan and K. A. Wahid, "Low-cost architecture of modified Daubechies lifting wavelets using integer polynomial mapping," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 5, pp. 585–589, May 2017.

[12] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient architectures for both 1-D and 2-D lifting-based wavelet transforms," *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1315–1326, May 2004.

[13] Wang and W. S. Gan, "An efficient VLSI architecture for lifting- based discrete wavelet packet transforms," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 54, no. 5, pp. 422–426, May 2007.

[14] M. Garcia, M. Mansour, and M. Ali, "A flexible hardware architecture for wavelet packet transforms that accommodates arbitrary tree struc- tures," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 60, no. 10, pp. 657–661, Oct. 2013.

[15] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, pp. 247–269, May 1998.

[16] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit reversal," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 657–661, Oct. 2011.