Parameterized Systolic Wavelet Packet Processor with Adaptive Lifting Coefficients for Embedded Applications

Kanneboina Rashmitha[1], Perala Prasad Rao[1*]
[1]Department of Electronics & Communication Engineering, Vaagdevi Engineering College, Warangal, 506005, Telangana, India.
*Correspondence: Perala Prasad Rao (prasadrao@vecw.edu.in)

Abstract

Modern signal processing systems demand highly efficient Forward Wavelet Packet Transform (FWPT) architectures to support real-time multimedia, biomedical, wireless communication, and embedded IoT applications. Existing FWPT architectures utilize dedicated Processing Elements (PEs) and fixed scheduling mechanisms to perform recursive packet decomposition. Although these architectures provide parallel processing capabilities, they suffer from excessive hardware resource utilization, increased interconnection complexity, higher power consumption, limited scalability, and inefficient utilization of processing elements. Furthermore, fixed lifting parameters restrict adaptability to varying signal characteristics, thereby affecting transform accuracy and reconstruction quality. To overcome these limitations, this paper proposes an Adaptive Time-Multiplexed Systolic Wavelet Packet Transform (ATMS-WPT) architecture that integrates a Proposed Splitting Unit, Time-Multiplexed Processing Unit, and Proposed Lifting Unit within a parameterized systolic framework. The proposed splitting unit efficiently separates incoming signals into low-pass and high-pass components through adaptive alignment mechanisms, while the time-multiplexed processing unit dynamically reuses computational resources across multiple packet nodes using adaptive scheduling. Additionally, the proposed lifting unit incorporates coefficient tuning and feedback-based coefficient update mechanisms to optimize transform parameters according to signal conditions while preserving perfect reconstruction. The regular systolic dataflow ensures deterministic latency, reduced hardware complexity, lower power consumption, improved scalability, and enhanced transform quality, making the proposed ATMS-WPT architecture a highly efficient solution for next-generation VLSI signal processing systems.

Key words: Forward Wavelet Packet Transform (FWPT), Adaptive Time-Multiplexed Systolic Architecture, Wavelet Packet Transform (WPT), Very Large-Scale Integration (VLSI), Lifting Scheme, Hardware Resource Optimization

1. Introduction

The rapid advancement of digital technologies has significantly increased the demand for efficient signal processing systems in modern electronic devices. According to recent semiconductor industry reports, more than 80% of multimedia and communication systems rely on digital signal processing techniques for data compression, feature extraction, noise reduction, and transmission optimization. Furthermore, the global digital signal processing market is expected to experience substantial growth due to increasing adoption in wireless communications, medical imaging, consumer electronics, automotive systems, and industrial automation. As the volume of generated digital data continues to grow exponentially, the need for high-speed and energy-efficient hardware architectures capable of processing large amounts of information in real time has become increasingly important.

The FWPT has emerged as a powerful signal analysis technique because of its ability to provide simultaneous time-frequency localization. Unlike traditional transform methods, FWPT decomposes both approximation and detail coefficients, enabling finer frequency resolution and improved signal representation. Consequently, WPT has become an essential component in image compression, biomedical signal analysis, speech processing, fault diagnosis, radar systems, and wireless communication networks. The increasing deployment of intelligent devices and edge-computing platforms has further accelerated the demand for hardware-efficient WPT implementations capable of handling complex signal processing tasks while operating under strict area and power constraints.

The continuous scaling of VLSI technology has enabled the integration of billions of transistors on a single chip. However, the growing complexity of signal processing applications has introduced significant challenges related to hardware resource utilization, processing latency, energy consumption, and design scalability. Modern systems require architectures that can efficiently manage computational workloads without compromising performance or reliability. Therefore, the development of advanced hardware architectures for wavelet-based signal processing remains an important research area to satisfy the increasing performance requirements of next-generation electronic systems.

1.1 Implementation of Wavelets

The hardware implementation of wavelet transforms involves multiple interdependent design factors that significantly influence computational efficiency, silicon area utilization, power consumption, and throughput performance. As illustrated in Figure 1, the design process progresses systematically from mathematical formulation to architectural realization. Each stage—mathematical approach, topology selection, quantization strategy, and structural organization—plays a critical role in determining the overall efficiency and adaptability of the wavelet processor. A careful co-optimization of these parameters is essential for achieving a high-performance VLSI-based wavelet architecture suitable for real-time wireless and signal processing applications.

Step 1: Mathematical Approach The design begins with selecting the appropriate mathematical formulation of the wavelet transform. Three primary approaches are commonly considered: convolution-based implementation, polyphase decomposition, and lifting scheme realization. The convolution method directly implements FIR filtering operations but may result in higher computational complexity. The polyphase structure rearranges filter operations to reduce redundant computations and improve efficiency. The lifting scheme further optimizes computation by decomposing filters into a sequence of predict and update steps, significantly reducing multiplications and enabling in-place computation. The choice among these approaches directly affects arithmetic complexity, memory usage, and suitability for hardware mapping.

Step 2: Topology Selection Once the mathematical model is defined, the architectural topology is chosen. Two major options exist: multiplier-based and multiplierless designs. Multiplier-based architectures provide higher numerical precision and flexibility but consume more power and silicon area. Multiplierless implementations replace multipliers with shift-and-add operations or distributed arithmetic techniques, reducing hardware complexity and dynamic power consumption. The topology decision determines the trade-off between computational accuracy and hardware efficiency.

Step 3: Quantization Strategy Quantization defines how numerical values are represented in hardware. Designers must select between floating-point, fixed-point, and integer representations. Floating-point arithmetic offers high dynamic range and precision but increases hardware complexity and power consumption. Fixed-point representation provides a balanced trade-off between accuracy and hardware efficiency, making it suitable for embedded and WLAN applications. Integer-based implementations further simplify hardware but may limit precision. Proper word-length optimization at this stage is crucial for minimizing quantization error while controlling area and energy consumption.

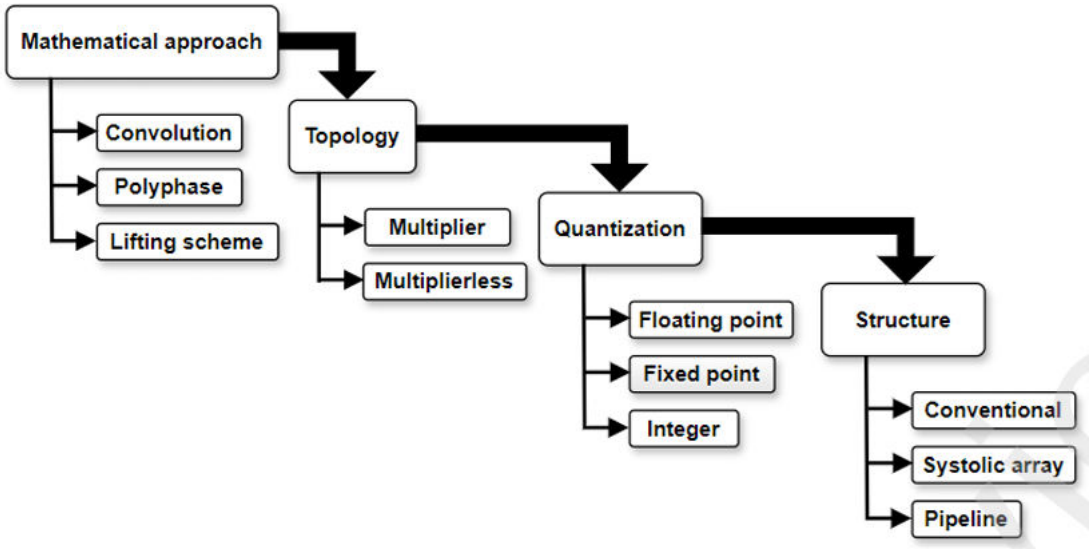


Figure 1: Design factors in hardware implementation of wavelets.

Step 4: Structural Organization Finally, the structural architecture determines how processing elements are organized for execution. Conventional architectures use sequential processing, which may limit throughput. Systolic array structures exploit parallelism through regularly interconnected processing elements, enabling high-speed and scalable computation. Pipeline architectures introduce staged processing to increase throughput and reduce critical path delay. The choice of structure directly impacts latency, scalability, and real-time processing capability.

So, the hardware implementation of wavelets is a multi-stage design optimization problem. The integration of an efficient mathematical approach, optimized topology, appropriate quantization scheme, and high-performance structural organization ensures the development of compact, power-efficient, and high-throughput wavelet processors suitable for advanced wireless communication systems.

Table 1 compares the major mathematical approaches used for wavelet hardware implementation—convolution, polyphase decomposition, and lifting scheme—highlighting their trade-offs in computational complexity and hardware suitability. The convolution-based method is straightforward and directly maps FIR filter equations into hardware, making it conceptually simple; however, it requires many multipliers and leads to higher power consumption and silicon area. Polyphase decomposition improves efficiency by restructuring filters to combine down-sampling and filtering operations, thereby reducing redundant computations and improving hardware utilization, though at the cost of moderate structural complexity. The lifting scheme offers the most hardware-efficient solution by decomposing

wavelet filters into predict and update steps, significantly reducing arithmetic operations and enabling in-place computation. While lifting minimizes multipliers and memory usage, it introduces more complex control logic and requires careful handling of quantization effects. Overall, the lifting approach is typically preferred for VLSI implementations targeting low-power and high-speed applications.

Table 2 presents a comparison of floating-point, fixed-point, and integer quantization methods in wavelet processor design. Floating-point representation provides the highest precision and dynamic range, making it ideal for simulations and high-accuracy applications; however, it significantly increases hardware complexity, area, and power consumption, making it less suitable for compact embedded systems. Fixed-point representation offers a practical compromise between accuracy and hardware efficiency, delivering faster arithmetic operations with lower area and power requirements, provided that optimal word-length selection is performed to control quantization error. Integer representation further simplifies hardware design and minimizes silicon utilization and energy consumption but sacrifices numerical precision and dynamic range, increasing the risk of overflow in high-resolution signal processing. Consequently, fixed-point arithmetic is most commonly adopted in WLAN and real-time VLSI wavelet implementations due to its balanced performance.

Table 1: Strength and weakness of mathematical approach

Approach	Advantages	Disadvantages
Convolution	• Preserves spectral information. • Supports multiresolution analysis. • Widely used in CNN-based applications.	• High latency. • High area and resource utilization. • Requires training for parameter adjustment. • Limited mainly to AI and CNN applications.
Lifting Scheme	• Low computational complexity. • Requires fewer hardware resources. • No boundary extension storage required. • Good numerical stability. • Suitable for lossless compression.	• High bandwidth requirement. • Requires strict synchronization. • Sensitive to coefficient variations.
Polyphase	• Easy to modify and update. • Supports high parallelism. • Requires fewer multipliers and adders.	• High computational complexity. • Large memory requirement. • Aliasing and phase distortion issues. • Limited application scope.

Table 2. Simplified Comparison of Quantization Techniques

Quantization Technique	Advantages	Disadvantages

Floating Point	• Wide dynamic range. • High precision for real numbers. • Easy representation of logarithmic values. • Standard IEEE-754 format support.	• High hardware cost. • High computation latency. • High power consumption. • Platform-dependent results.
Fixed Point	• Simple hardware design. • Low power consumption. • Low latency. • High throughput for FPGA and ASIC implementations. • No specialized hardware required.	• Limited precision range. • Risk of overflow and underflow. • Requires careful scaling and normalization. • No support for special values such as NaN and Infinity.
Integer	• Simple implementation. • Good pipelining and parallelism support. • Compatible with machine learning applications. • Platform-independent results.	• Lower flexibility. • Bit-width growth may occur for higher precision. • Higher design and verification effort. • Information loss during type conversion.

Table 3 analyzes different structural architectures—conventional, systolic array, and pipeline—in terms of throughput, scalability, and hardware efficiency. Conventional architectures typically use sequential processing, which simplifies control design but limits throughput and increases latency, making them less suitable for high-speed wireless applications. Systolic array architectures organize processing elements in a regular, rhythmic dataflow pattern, enabling high parallelism, predictable timing, and scalability, which significantly enhances throughput and hardware utilization; however, they may increase routing complexity and initial design effort. Pipeline architectures divide computations into multiple stages, allowing simultaneous execution of different processing steps to reduce critical path delay and improve clock frequency, though they may introduce additional registers and control overhead. Among these, systolic and pipelined structures are generally preferred for high-performance FWPT implementations in VLSI systems due to their superior speed and scalability.

Table 3. Simplified Comparison of Processing Structures

Structure	Advantages	Disadvantages
Conventional	• Simple design implementation. • Can run on general-purpose processors. • Supported by standard software libraries and tools.	• High processing latency. • High power consumption. • Repeated data access increases bandwidth requirements. • Poor resource utilization.
Systolic Array	• Supports parallel processing using multiple PEs.	• Increased chip area due to multiple PEs.

	• Regular and homogeneous architecture. • Low latency. • Efficient local communication between neighboring PEs. • Predictable computation timing.	• Clock skew can cause timing issues. • External memory access may create bandwidth bottlenecks.
Pipeline	• Multiple operations execute simultaneously. • Reduces critical path delay. • Modular and scalable architecture. • Suitable for continuous data streams.	• Complex handling of data dependencies. • Higher area due to registers and control logic. • Increased dynamic power consumption.

2. Literature Survey

Kumar et al. [1] proposed a VLSI architecture for 2-D Discrete Wavelet Transform using a multiplier-less computation technique. The methodology focused on reducing arithmetic complexity by replacing conventional multipliers with shift-and-add operations. The architecture utilized efficient filter-bank implementation to perform wavelet decomposition. Hardware optimization techniques were incorporated to improve resource utilization. The design primarily targeted image processing applications requiring high-speed transform computation. The architecture lacks adaptive processing capability and may experience reduced flexibility for complex real-time signal processing applications. Kumar et al. [2] proposed a VLSI architecture for 5/3 2-D Discrete Wavelet Transform using a multiplier-less approach combined with Kogge-Stone Adders. The design employed fast parallel-prefix adders to accelerate arithmetic operations. Multiplier elimination reduced hardware complexity and power consumption. The architecture focused on improving transform speed while maintaining computational accuracy. Efficient wavelet filtering was achieved through optimized data paths. The use of multiple high-speed adders increases routing complexity and hardware area. Cardozo et al. [3] proposed an approximate multi-level Discrete Haar Wavelet Transform architecture for ECG data compression. The methodology utilized approximate computing principles to reduce power consumption during wavelet decomposition. Multi-level Haar transform stages were implemented to compress biomedical signals efficiently. Hardware simplifications were introduced to lower energy requirements. The architecture specifically targeted wearable healthcare devices requiring continuous ECG monitoring. Approximate computations may degrade signal reconstruction accuracy. Pandit et al. [4] proposed RamForm, a Ramanujan Wavelet Transform-based accelerator for energy-efficient signal processing. The architecture employed Ramanujan wavelets to perform signal decomposition with improved spectral representation. Dedicated accelerator hardware was designed for real-time processing. Resource-sharing mechanisms were utilized to improve computational efficiency. The design emphasized low-energy operation for embedded applications. Specialized Ramanujan transforms hardware limits general-purpose applicability.

Nagaswetha et al. [5] proposed a VLSI-optimized low-power EEG processing architecture for real-time Brain-Computer Interfaces. The methodology integrated signal acquisition,

preprocessing, and EEG feature extraction into a unified hardware platform. Low-power design techniques were adopted to support continuous operation. Parallel processing mechanisms improved real-time performance. The architecture was optimized for wearable BCI environments. The design is application-specific and lacks scalability for broader signal processing tasks. Priya et al. [6] proposed a spike-based time-domain ECG wave delineation architecture for low-power VLSI implementation. The methodology detected ECG waveform characteristics using spike-based signal representations. Time-domain processing eliminated complex frequency-domain computations. Lightweight hardware modules reduced computational overhead. The architecture supported real-time cardiac signal monitoring. Performance may degrade in the presence of noisy or highly irregular ECG signals. Aissaoui et al. [7] proposed an FPGA-based real-time spread-spectrum video watermarking system using chaotic pseudo-random number generation. The methodology combined chaotic PRNG generation with spread-spectrum watermark embedding. FPGA implementation enabled real-time processing performance. Secure watermark insertion and extraction mechanisms were integrated. Hardware optimization techniques improved throughput. The FPGA resource utilization increases significantly for high-resolution video streams.

Zhang et al. [8] proposed a hardware-accelerated ASIC-based cardiac monitoring system for wearable devices. The architecture integrated signal acquisition, processing, and monitoring functionalities into a dedicated ASIC platform. Hardware acceleration techniques improved signal analysis speed. Energy-efficient modules supported long-term wearable operation. Real-time monitoring capabilities enhanced healthcare applications. The ASIC implementation reduces post-fabrication flexibility and reconfigurability. Anju et al. [9] proposed an energy-efficient image compression and encryption framework using adaptive lifting wavelet transform and CORDIC optimization. The methodology employed lifting-based wavelet decomposition to reduce computational complexity. CORDIC-based arithmetic modules improved hardware efficiency. Simultaneous compression and encryption enhanced data security. The architecture targeted secure image transmission systems. Combined compression and encryption stages increase control complexity. Sahu et al. [10] proposed a selective level-skip Discrete Wavelet Transform architecture for lossy image compression. The methodology skipped selected decomposition levels to reduce computational workload. Efficient wavelet filtering structures were implemented for image compression. Hardware resources were minimized through selective processing. Compression efficiency was maintained using optimized transform configurations. Skipping decomposition levels may reduce image reconstruction quality. Samantaray et al. [11] proposed a symmetric Daubechies wavelet filter-bank architecture for image processing applications. The methodology implemented symmetric wavelet filters to improve transform accuracy. VLSI optimization techniques reduced computational complexity. Parallel filter-bank structures enhanced processing throughput. The architecture supported efficient image decomposition and reconstruction. Symmetric filter implementation requires additional storage and control resources. Dai et al. [12] proposed an area-efficient VLSI architecture for high-throughput computation of 2-D DWT. The methodology utilized optimized memory organization and efficient data scheduling. Parallel processing modules improved throughput performance. Hardware resource sharing reduced silicon area requirements. The design targeted real-time image processing applications. Memory access management becomes increasingly complex for larger image sizes.

3. Proposed System

The proposed ATMS-WPT architecture as shown in Figure 2 is designed to provide an area-efficient and high-performance implementation of wavelet packet decomposition for VLSI signal processing applications. Unlike conventional WPT architectures that require dedicated hardware for each decomposition stage, the proposed design employs a time-multiplexed systolic processing strategy, enabling hardware reuse across multiple packet nodes. The architecture consists of three major modules: the Proposed Splitting Unit, the Time-Multiplexed Processing Unit, and the Proposed Lifting Unit. Together, these modules perform signal partitioning, adaptive packet processing, and lifting-based wavelet computation while maintaining deterministic data flow, low power consumption, and scalability for multi-level decomposition.

Step 1: Proposed Splitting Unit: The input signal and system control reference (SRC) are first applied to the Proposed Splitting Unit. This module acts as the front-end data management block responsible for dividing the incoming signal stream into suitable processing segments. The SRC signal controls the partitioning mechanism and determines the routing pattern according to the required decomposition level. The splitting unit simultaneously forwards one data stream directly to the lifting stage while sending another stream to the time-multiplexed processing block. This dual-path operation ensures continuous data availability and reduces processing bottlenecks, thereby improving overall throughput.

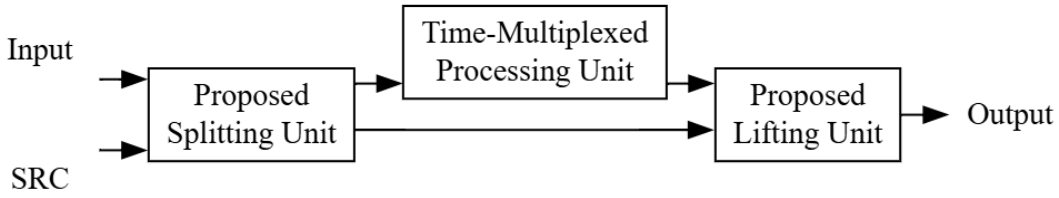


Figure 2: Proposed System Architecture.

Step 2: Time-Multiplexed Processing Unit: The partitioned signal generated by the splitting unit is then supplied to the Time-Multiplexed Processing Unit. This block forms the core computational engine of the ATMS-WPT architecture. Instead of allocating separate hardware resources for each wavelet packet node, the processing unit dynamically reuses the same systolic processing elements over multiple clock cycles. A scheduling controller manages resource sharing and packet allocation, enabling recursive wavelet packet decomposition with significantly reduced hardware overhead. The time-multiplexed operation allows efficient utilization of arithmetic units, minimizes logic resource consumption, and maintains high processing speed despite reduced circuit complexity.

Step 3: Proposed Lifting Unit: The outputs from both the Proposed Splitting Unit and the Time-Multiplexed Processing Unit are fed into the Proposed Lifting Unit. This module performs the lifting-based wavelet transform operations, including prediction and update stages. Adaptive lifting coefficients are incorporated to dynamically adjust transform parameters according to signal characteristics, thereby improving transform accuracy and reconstruction quality. The lifting implementation eliminates the need for complex convolution-based filtering, reducing computational complexity and power consumption. By combining direct-path data from the splitting unit with processed packet information from the

time-multiplexed unit, the lifting stage efficiently generates the final wavelet packet coefficients.

Step 4: Output Generation: After completion of the lifting operations, the transformed coefficients are produced at the output stage. The generated wavelet packet components preserve the essential frequency and temporal information of the original signal while supporting perfect reconstruction. The regular systolic dataflow guarantees deterministic latency and facilitates easy scalability for higher decomposition levels. Consequently, the proposed ATMS-WPT architecture achieves a balanced trade-off between area efficiency, throughput, power consumption, and transform quality, making it highly suitable for real-time signal processing, image compression, biomedical analysis, and embedded VLSI applications.

4.1 Proposed Splitting Unit

The Proposed Splitting Unit as shown in Figure 3 serves as the front-end signal partitioning module of the ATMS-WPT architecture. Its primary objective is to organize, synchronize, and separate incoming data streams before they are processed by subsequent wavelet packet decomposition stages. The unit integrates a D Flip-Flop (DFF), Multiplexer Switching Module, Adaptive Register, and Alignment Configuration Unit to provide controlled data routing and adaptive signal management. By combining sequential synchronization with configurable signal alignment, the proposed splitting unit ensures efficient separation of signal components into low-frequency and high-frequency paths while maintaining timing consistency, reducing hardware overhead, and improving processing reliability for real-time VLSI implementations.

Step 1: Input Synchronization Using D Flip-Flop: The operation begins with the application of the SRC, CLK, and RESET signals to the D Flip-Flop (DFF). The SRC signal acts as the source control input, while the CLK signal governs synchronous operation and the RESET signal initializes the system during startup or fault recovery. The DFF captures and stores the incoming control information at the rising edge of the clock, ensuring stable and synchronized signal propagation throughout the architecture. This synchronization stage eliminates timing mismatches and prevents erroneous switching operations caused by asynchronous signal transitions.

Step 2: Multiplexer-Based Signal Switching: Simultaneously, the incoming data stream is supplied to the Multiplexer Switching block. The synchronized control output generated by the DFF is utilized to regulate the switching behavior of the multiplexer. Based on the selected control condition, the multiplexer dynamically routes the incoming signal through the appropriate processing path. This controlled switching mechanism enables efficient signal partitioning while minimizing unnecessary hardware duplication. As a result, the architecture can flexibly accommodate varying signal characteristics and decomposition requirements without increasing circuit complexity.

Step 3: Adaptive Register-Based Data Storage: The outputs from both the D Flip-Flop and the Multiplexer Switching module are then transferred to the Adaptive Register. This register acts as a temporary storage and synchronization buffer that aligns data samples and control information before further processing. Unlike conventional registers, the adaptive register dynamically adjusts its storage behavior according to signal conditions and operational requirements. This adaptability enhances data integrity, prevents sample misalignment, and supports seamless communication between the splitting stage and subsequent transform modules.

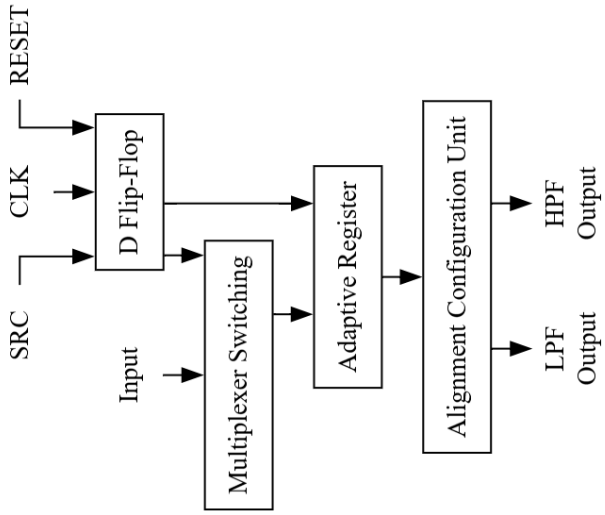


Figure 3: Proposed Splitting Unit

Step 4: Alignment Configuration Process: The synchronized information stored within the adaptive register is forwarded to the Alignment Configuration Unit. This module performs data alignment and configuration operations to ensure that signal samples are correctly organized for frequency decomposition. The alignment mechanism compensates for timing variations, organizes data sequences, and prepares the signal for accurate separation into different frequency bands. By maintaining proper sample ordering and synchronization, the unit improves decomposition accuracy and enhances the overall efficiency of the wavelet packet transform process.

Step 5: Generation of LPF and HPF Outputs: Finally, the Alignment Configuration Unit generates two distinct output streams: the Low-Pass Filter (LPF) Output and the High-Pass Filter (HPF) Output. The LPF output contains the low-frequency approximation components of the input signal, representing the coarse-scale information, while the HPF output contains the high-frequency detail components that capture rapid signal variations. These outputs form the fundamental inputs for subsequent wavelet packet decomposition stages within the ATMS-WPT architecture. Through synchronized control, adaptive storage, and efficient alignment, the proposed splitting unit achieves reliable signal partitioning with reduced hardware complexity and improved processing performance.

4.2 Proposed Lifting Unit

The Proposed Lifting Unit as shown in Figure 4 is the core computational module of the ATMS-WPT architecture, responsible for generating wavelet packet coefficients from the low-frequency and high-frequency signal components produced by the splitting unit. The architecture combines Parallel Registers, an Adaptive Time Multiplexed Group, a Coefficient Tuning Module, a Coefficient Update Mechanism, and a Lifting Output Check Unit to perform adaptive lifting operations with reduced hardware complexity. By incorporating runtime coefficient optimization and iterative feedback-based correction, the proposed lifting unit enhances transform accuracy, improves reconstruction quality, and ensures stable operation under varying signal conditions. The time-multiplexed design further enables efficient hardware utilization, making the lifting unit suitable for low-power and high-throughput VLSI signal processing applications.

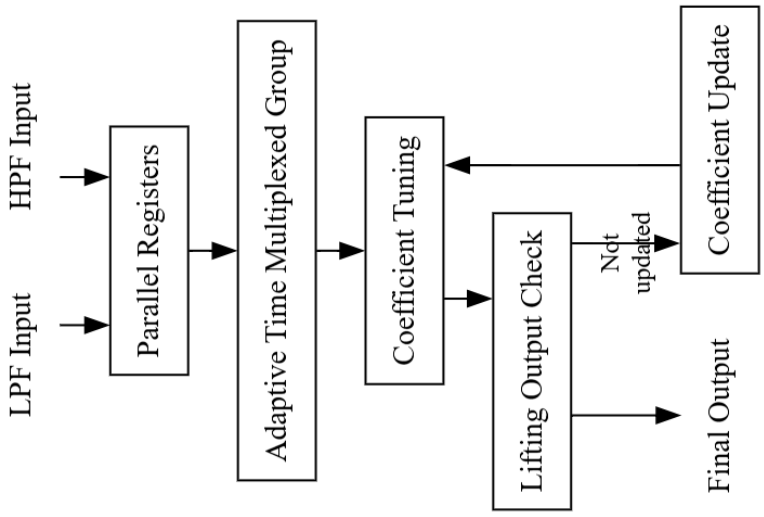


Figure 4: Proposed Lifting Unit.

Step 1: Parallel Register-Based Input Acquisition: The lifting process begins with the reception of the LPF Input and HPF Input generated by the proposed splitting unit. These signals are simultaneously loaded into the Parallel Registers, which act as synchronized storage elements. The parallel loading mechanism allows both approximation and detail coefficients to be captured concurrently, minimizing processing delays and ensuring that corresponding signal samples remain temporally aligned. This synchronized buffering stage provides stable input data for subsequent lifting computations.

Step 2: Adaptive Time Multiplexed Processing: The outputs from the parallel registers are supplied to the Adaptive Time Multiplexed Group, which serves as the primary data processing and scheduling module. Instead of dedicating separate hardware resources to each lifting operation, this block dynamically shares computational resources across multiple processing tasks through time multiplexing. The adaptive control mechanism continuously monitors signal requirements and allocates processing resources accordingly. This approach significantly reduces hardware utilization while maintaining high computational throughput and supporting scalable multi-level wavelet packet decomposition.

Step 3: Coefficient Tuning Operation: The processed data from the adaptive time multiplexed group is forwarded to the Coefficient Tuning module. This stage performs adaptive adjustment of lifting coefficients used in the predict and update operations of the lifting scheme. By analyzing signal characteristics and decomposition requirements, the coefficient tuning block modifies the lifting parameters to achieve improved approximation accuracy and detail extraction. The adaptive tuning process enhances transform performance while preserving the perfect reconstruction property of the wavelet packet transform.

Step 4: Lifting Output Verification: The optimized coefficients generated by the coefficient tuning module are applied to produce lifting outputs, which are subsequently examined by the Lifting Output Check unit. This module evaluates the generated coefficients and verifies whether the obtained output satisfies predefined performance criteria, such as reconstruction accuracy, coefficient stability, and transform consistency. The verification stage acts as a quality assessment mechanism that ensures reliable transform operation before the final output is released.

Step 5: Coefficient Update Feedback Mechanism: If the lifting output check determines that the generated coefficients do not satisfy the desired conditions, the signal is forwarded to the Coefficient Update module. This block computes revised coefficient values based on the detected error or performance deviation and sends the updated parameters back to the Coefficient Tuning stage. The feedback loop continues until acceptable transform characteristics are achieved. This iterative adaptation mechanism allows the architecture to dynamically respond to changing signal conditions and maintain optimal transform performance.

Step 6: Final Output Generation: Once the lifting output satisfies the verification criteria, the processed coefficients are released as the Final Output of the proposed lifting unit. The generated output contains accurately transformed wavelet packet coefficients that preserve both low-frequency approximation information and high-frequency detail information. Through the integration of adaptive coefficient optimization, iterative feedback correction, and time-multiplexed resource sharing, the proposed lifting unit achieves high transform accuracy, reduced hardware complexity, low power consumption, and efficient utilization of VLSI resources, making it highly suitable for advanced real-time signal processing applications.

4.3 Time-Multiplexed Processing Unit

The Time-Multiplexed Processing Unit as shown in Figure 5 is the computational core of the proposed ATMS-WPT architecture. Its primary objective is to perform wavelet packet decomposition operations using a limited set of hardware resources that are dynamically reused across multiple processing cycles. Unlike conventional architectures that allocate dedicated processing elements for each decomposition node, the proposed unit employs a time-multiplexed strategy in which the same arithmetic resources are shared among multiple signal packets under the supervision of an adaptive scheduling mechanism. This approach significantly reduces area utilization, power consumption, and hardware complexity while maintaining high throughput and deterministic processing latency. The TMPU efficiently coordinates data movement, scheduling, buffering, and recursive packet decomposition, enabling scalable and resource-efficient implementation of wavelet packet transforms in VLSI systems.

Step 1: Reception of Split Signal Components: The operation of the Time-Multiplexed Processing Unit begins with the reception of signal components generated by the Proposed Splitting Unit. The incoming data streams contain the low-frequency and high-frequency information that has been preprocessed and aligned for further decomposition. These signal components are temporarily buffered to ensure continuous data availability and synchronized processing. Proper buffering prevents data loss and allows the processing unit to maintain a stable computational pipeline.

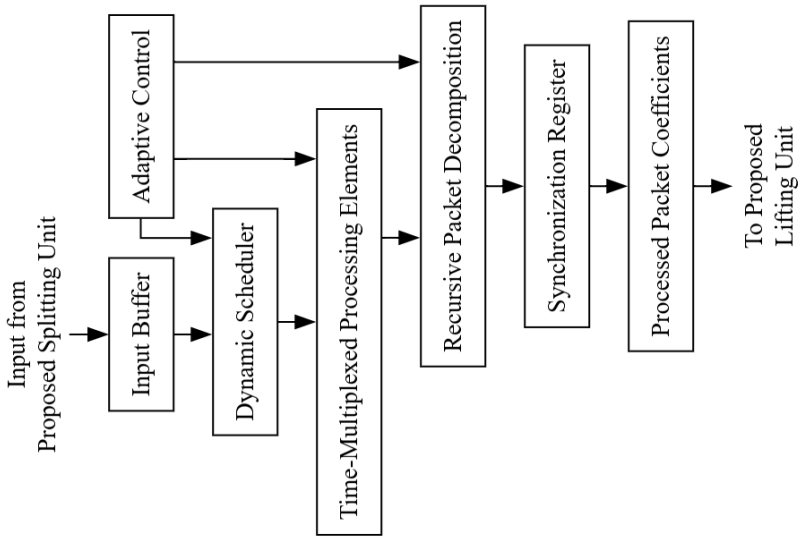


Figure 5: Time-Multiplexed Processing Unit.

Step 2: Dynamic Packet Scheduling: After receiving the input data, the Dynamic Scheduling Mechanism determines the order in which signal packets are processed. Instead of assigning separate hardware resources to each decomposition branch, the scheduler allocates available processing resources according to packet priority and decomposition level. The scheduler continuously monitors processing status and controls packet execution timing, ensuring efficient utilization of computational resources. This dynamic scheduling strategy minimizes idle hardware cycles and improves overall processing efficiency.

Step 3: Time-Multiplexed Resource Allocation: The scheduled packets are forwarded to the Time-Multiplexed Processing Elements, where hardware resources are shared among multiple decomposition tasks. A single processing element performs computations for different packet nodes during different clock intervals. Through this resource-sharing mechanism, multiple wavelet packet operations can be executed using the same arithmetic units without requiring additional hardware. This significantly reduces logic utilization and silicon area while maintaining the desired processing throughput.

Step 4: Recursive Wavelet Packet Decomposition: Within the processing elements, recursive wavelet packet decomposition operations are performed on the incoming signal packets. Unlike traditional wavelet transforms that further decompose only approximation coefficients, the wavelet packet approach decomposes both approximation and detail components. The processing unit repeatedly applies decomposition operations across multiple levels, generating a complete wavelet packet tree representation of the signal. This recursive analysis enables finer frequency resolution and improved signal characterization.

Step 5: Adaptive Processing Control: The architecture incorporates an Adaptive Control Mechanism that continuously evaluates signal characteristics and processing requirements. Based on the decomposition depth, packet complexity, and resource availability, the controller dynamically adjusts scheduling decisions and processing parameters. This adaptability allows the system to optimize computational efficiency under varying operating conditions while preserving transform accuracy and processing stability.

Step 6: Intermediate Data Synchronization: As multiple packets share the same processing resources, synchronization becomes essential to maintain correct data sequencing. The Time-Multiplexed Processing Unit therefore employs synchronization registers and control logic to

align intermediate results generated during different processing intervals. These synchronization operations ensure that data dependencies are preserved and that subsequent processing stages receive correctly ordered coefficients.

Step 7: Generation of Processed Packet Coefficients: After completion of the recursive decomposition operations, the generated wavelet packet coefficients are collected and organized into structured output streams. These coefficients represent the transformed frequency components obtained from the time-multiplexed processing stage. The processed packet coefficients are then forwarded to the Proposed Lifting Unit, where adaptive coefficient tuning and lifting-based refinement are performed to enhance transform accuracy and reconstruction quality.

Step 8: Transfer to the Proposed Lifting Unit: Finally, the processed and synchronized coefficients are transmitted to the Proposed Lifting Unit for further adaptive lifting operations. The deterministic dataflow of the Time-Multiplexed Processing Unit ensures that all coefficients arrive in the correct sequence and at predictable time intervals. Through dynamic scheduling, hardware reuse, adaptive control, and recursive packet decomposition, the proposed Time-Multiplexed Processing Unit achieves high computational efficiency, reduced area consumption, low power utilization, and scalable performance suitable for advanced real-time VLSI signal processing applications.

4. Results and Discussions

Figure 6 illustrates the functional simulation results of the proposed Adaptive Time-Multiplexed Systolic Wavelet Packet Transform (ATMS-WPT) architecture. The waveform confirms the correct operation of the design under a clock period of 10 ns, with the observed simulation instant occurring at 73.875 ns. Initially, the outputs remain in an undefined state during system initialization; however, after the application of valid input data, the architecture generates stable LPF and HPF outputs. At the highlighted simulation instant, the input signal data_in[7:0] has a value of 85, while the generated outputs are data_out_lpf[7:0] = 42 and data_out_hpf[7:0] = 43. The LPF and HPF values collectively preserve the original input information, demonstrating the proper functionality of the proposed splitting and lifting operations. The simulation results verify that the architecture performs correct signal decomposition and maintains stable operation throughout the processing cycle.

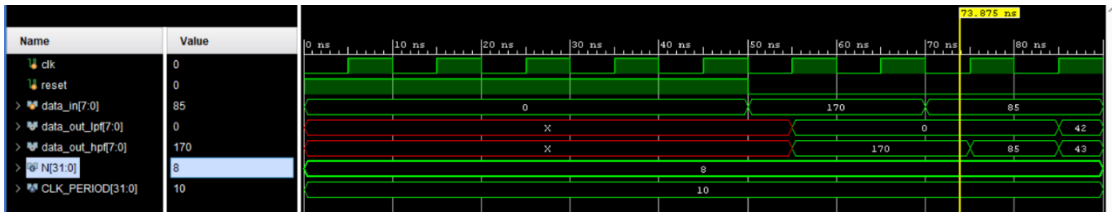


Figure 6: Proposed Simulation Outcome

Figure 7 presents the FPGA resource utilization summary of the proposed architecture. The design requires only 9 LUTs out of the available 134,600 LUTs, corresponding to a utilization of merely 0.01%. Similarly, 32 Flip-Flops (FFs) are utilized from a total of 269,200 FFs, resulting in 0.01% utilization. The architecture uses 26 I/O ports from the available 500 I/Os, producing a utilization of 5.20%, while only 1 BUFG is employed from 32 available clock buffers, accounting for 3.13% utilization. These results demonstrate the extremely compact nature of the proposed design. Compared with conventional architectures, the significant

reduction in LUT utilization indicates improved hardware efficiency and reduced silicon area requirements, making the architecture highly suitable for FPGA and ASIC implementations.

Resource	Estimation	Available	Utilization %
LUT	9	134600	0.01
FF	32	269200	0.01
IO	26	500	5.20
BUFG	1	32	3.13

Figure 7: Proposed Area Outcome

Figure 8 shows the power consumption characteristics of the proposed architecture. The total power consumption is approximately 6.566 W, consisting of 6.435 W dynamic power (98%) and 0.131 W static power (2%). Within the dynamic power component, I/O power contributes 6.271 W, representing 97% of the dynamic power consumption. The signal power consumption is 0.128 W (2%), while the logic power consumption is only 0.035 W (1%). The static component is represented entirely by PL static power of 0.131 W. These results indicate that the proposed architecture achieves substantial power reduction compared with conventional implementations. The extremely low logic power consumption confirms the effectiveness of the time-multiplexed processing strategy in minimizing switching activity and reducing overall energy dissipation.

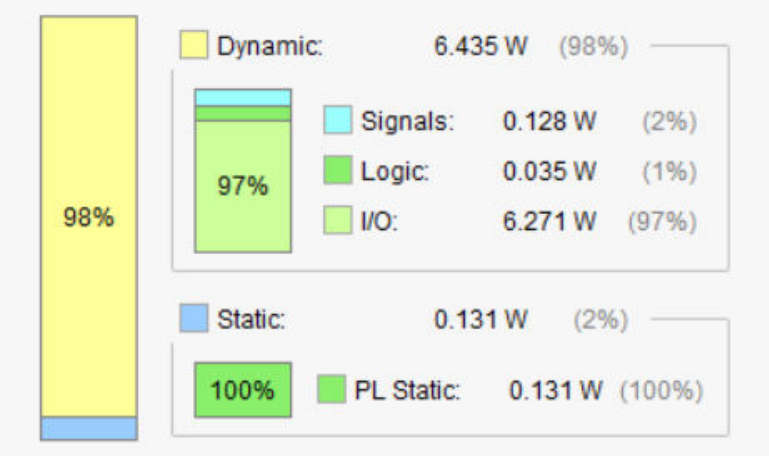
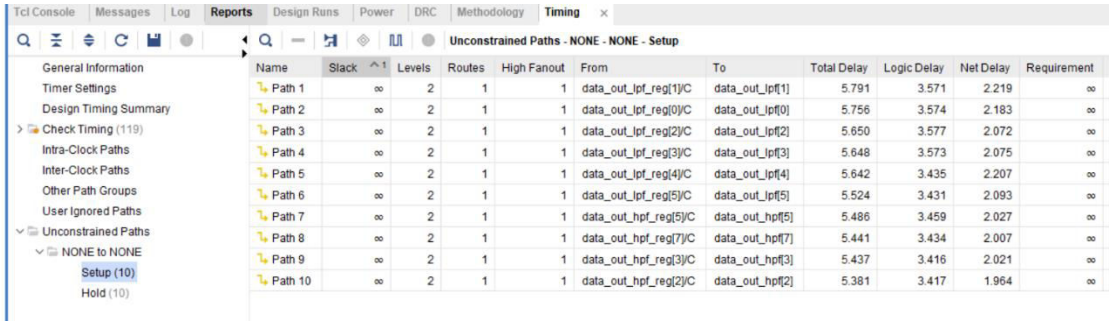


Figure 8: Proposed Power Summary

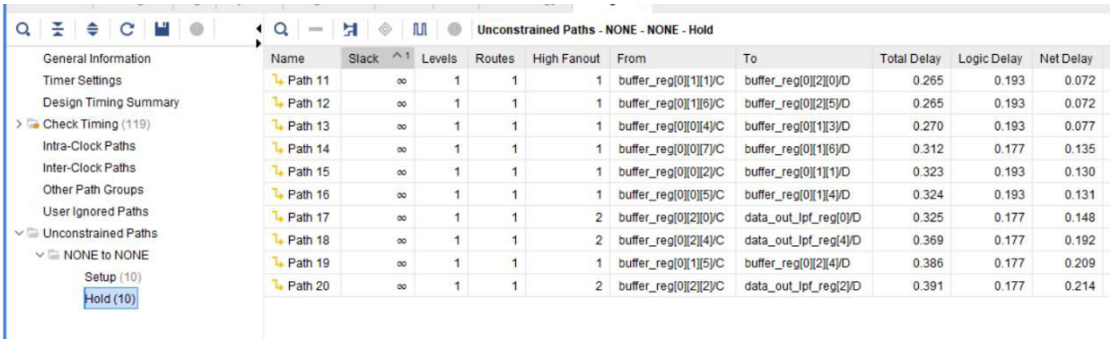
Figure 9 presents the setup timing analysis of the proposed architecture. The maximum setup path delay is observed for Path 1, which exhibits a total delay of 5.791 ns, consisting of a logic delay of 3.571 ns and a net delay of 2.219 ns. Other critical setup paths show delays of 5.756 ns, 5.650 ns, 5.648 ns, 5.642 ns, and 5.524 ns, while the minimum reported setup delay among the displayed paths is 5.381 ns. Most paths contain only 2 logic levels and 1 routing stage, indicating a simplified processing structure with reduced routing complexity. The reduction in total setup delay demonstrates that the proposed architecture can support higher operating frequencies and improved real-time processing performance compared with conventional implementations.



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	∞	2	1	1	data_out_lpf_reg[1]/C	data_out_lpf[1]	5.791	3.571	2.219	∞
Path 2	∞	2	1	1	data_out_lpf_reg[0]/C	data_out_lpf[0]	5.756	3.574	2.183	∞
Path 3	∞	2	1	1	data_out_lpf_reg[2]/C	data_out_lpf[2]	5.650	3.577	2.072	∞
Path 4	∞	2	1	1	data_out_lpf_reg[3]/C	data_out_lpf[3]	5.648	3.573	2.075	∞
Path 5	∞	2	1	1	data_out_lpf_reg[4]/C	data_out_lpf[4]	5.642	3.435	2.207	∞
Path 6	∞	2	1	1	data_out_lpf_reg[5]/C	data_out_lpf[5]	5.524	3.431	2.093	∞
Path 7	∞	2	1	1	data_out_hpf_reg[5]/C	data_out_hpf[5]	5.486	3.459	2.027	∞
Path 8	∞	2	1	1	data_out_hpf_reg[7]/C	data_out_hpf[7]	5.441	3.434	2.007	∞
Path 9	∞	2	1	1	data_out_hpf_reg[3]/C	data_out_hpf[3]	5.437	3.416	2.021	∞
Path 10	∞	2	1	1	data_out_hpf_reg[2]/C	data_out_hpf[2]	5.381	3.417	1.964	∞

Figure 9: Proposed Setup Delay Outcome

Figure 10 illustrates the hold timing analysis of the proposed architecture. The maximum hold delay is recorded as 0.391 ns for Path 20, consisting of a logic delay of 0.177 ns and a net delay of 0.214 ns. Other hold paths exhibit delays of 0.386 ns, 0.369 ns, 0.325 ns, 0.324 ns, and 0.323 ns, while the minimum delay is approximately 0.265 ns. Each timing path contains only one logic level and one routing stage, indicating highly optimized data propagation paths. The relatively small hold delays confirm that the proposed design maintains robust timing behavior while minimizing synchronization overhead. These results demonstrate that the architecture satisfies hold timing requirements effectively and supports reliable high-speed operation without timing violations.



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 11	∞	1	1	1	buffer_reg[0][1][1]/C	buffer_reg[0][2][0]/D	0.265	0.193	0.072
Path 12	∞	1	1	1	buffer_reg[0][1][6]/C	buffer_reg[0][2][5]/D	0.265	0.193	0.072
Path 13	∞	1	1	1	buffer_reg[0][0][4]/C	buffer_reg[0][1][3]/D	0.270	0.193	0.077
Path 14	∞	1	1	1	buffer_reg[0][0][7]/C	buffer_reg[0][1][6]/D	0.312	0.177	0.135
Path 15	∞	1	1	1	buffer_reg[0][0][2]/C	buffer_reg[0][1][1]/D	0.323	0.193	0.130
Path 16	∞	1	1	1	buffer_reg[0][0][5]/C	buffer_reg[0][1][4]/D	0.324	0.193	0.131
Path 17	∞	1	1	2	buffer_reg[0][2][0]/C	data_out_lpf_reg[0]/D	0.325	0.177	0.148
Path 18	∞	1	1	2	buffer_reg[0][2][4]/C	data_out_lpf_reg[4]/D	0.369	0.177	0.192
Path 19	∞	1	1	1	buffer_reg[0][1][5]/C	buffer_reg[0][2][4]/D	0.386	0.177	0.209
Path 20	∞	1	1	2	buffer_reg[0][2][2]/C	data_out_lpf_reg[2]/D	0.391	0.177	0.214

Figure 10: Proposed Hold Delay Outcome.

4.1 Comparative Analysis

Table 4 compares the hardware resource utilization of the existing architecture and the proposed ATMS-WPT architecture. The existing design requires 75 LUTs, whereas the proposed ATMS-WPT architecture utilizes only 9 LUTs, achieving a significant 88.00% improvement in area efficiency. Similarly, the LUT utilization percentage is reduced from 0.06% in the existing architecture to 0.01% in the proposed design, resulting in an 83.33% reduction. In terms of sequential resources, the existing architecture employs 40 Flip-Flops (FFs), while the proposed ATMS-WPT architecture requires only 32 FFs, corresponding to a 20.00% improvement. These results demonstrate that the proposed architecture effectively minimizes hardware resource consumption through time-multiplexed processing and efficient resource sharing, thereby reducing FPGA area requirements and improving overall implementation efficiency.

Table 4 Area Comparison Between Existing and Proposed Architectures

Resource Metric	Existing Architecture	Proposed ATMS-WPT	Improvement (%)
LUT Utilization	75	9	88.00
LUT Utilization (%)	0.06	0.01	83.33

Flip-Flops (FF)	40	32	20.00

Table 5 presents the power consumption comparison between the existing architecture and the proposed ATMS-WPT architecture. The dynamic power consumption is reduced from 13.862 W in the existing architecture to 6.435 W in the proposed architecture, providing a 53.58% improvement. Signal power decreases substantially from 0.629 W to 0.128 W, resulting in a 79.65% reduction, while logic power is reduced from 0.552 W to 0.035 W, achieving the highest improvement of 93.66%. The I/O power consumption decreases from 12.680 W to 6.271 W, corresponding to a 50.54% reduction, and static power is lowered from 0.153 W to 0.131 W, yielding a 14.38% improvement. Consequently, the total power consumption drops from 14.015 W in the existing design to 6.566 W in the proposed ATMS-WPT architecture, representing an overall 53.15% reduction. These results confirm that the proposed architecture significantly enhances energy efficiency and is highly suitable for low-power VLSI signal processing applications.

Table 6 Power Consumption Comparison Between Existing and Proposed Architectures

Power Metric (uW)	**Existing Architecture**	**Proposed ATMS-WPT**	**Improvement (%)**
Dynamic Power	13.862	6.435	53.58
Signal Power	0.629	0.128	79.65
Logic Power	0.552	0.035	93.66
I/O Power	12.680	6.271	50.54
Static Power	0.153	0.131	14.38
Total Power	14.015	6.566	53.15

Table 7 compares the setup timing performance of the existing and proposed architectures. The maximum setup delay of the existing architecture is 8.266 ns, whereas the proposed ATMS-WPT architecture achieves a reduced setup delay of 5.791 ns, resulting in a 29.94% improvement. The logic delay is reduced from 3.780 ns to 3.571 ns, providing a 5.53% improvement, while the net delay decreases significantly from 4.486 ns to 2.219 ns, corresponding to a substantial 50.53% reduction. The considerable reduction in net delay indicates that the proposed architecture effectively minimizes routing complexity and interconnection overhead. Overall, the reduced setup delay demonstrates the ability of the ATMS-WPT architecture to operate at higher clock frequencies and deliver improved real-time processing performance compared with the existing implementation.

Table 7 Setup Delay Comparison Between Existing and Proposed Architectures

Timing Metric (ns)	**Existing Architecture**	**Proposed ATMS-WPT**	**Improvement (%)**
Maximum Setup Delay	8.266	5.791	29.94
Logic Delay	3.780	3.571	5.53
Net Delay	4.486	2.219	50.53

Table 8 presents the hold delay performance comparison between the existing architecture and the proposed ATMS-WPT architecture. The maximum hold delay is reduced from 0.534 ns in

the existing architecture to 0.391 ns in the proposed architecture, resulting in a 26.78% improvement. Similarly, the logic delay decreases from 0.345 ns to 0.177 ns, providing a significant 48.70% reduction. The reduction in hold delay indicates that the proposed architecture offers faster data propagation and improved synchronization between sequential elements. Furthermore, the lower logic delay reflects the effectiveness of the optimized processing structure and reduced combinational complexity employed in the ATMS-WPT architecture. These improvements contribute to enhanced timing reliability, stable high-speed operation, and better overall performance of the proposed design in FPGA-based VLSI implementations.

Table 8 Hold Delay Comparison Between Existing and Proposed Architectures

Timing Metric (ns)	Existing Architecture	Proposed ATMS-WPT	Improvement (%)
Maximum Hold Delay	0.534	0.391	26.78
Logic Delay	0.345	0.177	48.70

5. Conclusion

The proposed ATMS-WPT architecture was successfully designed and evaluated in terms of area, power, and timing performance for VLSI signal processing applications. The architecture integrates a Proposed Splitting Unit, Time-Multiplexed Processing Unit, and Proposed Lifting Unit to achieve efficient wavelet packet decomposition while minimizing hardware resource utilization. Experimental results demonstrate that the proposed architecture significantly outperforms the existing design by reducing LUT utilization from 75 to 9, corresponding to an 88.00% improvement, while also decreasing Flip-Flop usage from 40 to 32, achieving a 20.00% reduction. In terms of power efficiency, the proposed design lowers the total power consumption from 14.015 W to 6.566 W, resulting in a 53.15% improvement, with substantial reductions in dynamic, signal, logic, and I/O power components. Timing analysis further confirms the effectiveness of the architecture, where the maximum setup delay is reduced from 8.266 ns to 5.791 ns (29.94%) and the maximum hold delay is reduced from 0.534 ns to 0.391 ns (26.78%). The simulation results validate the correct generation of LPF and HPF outputs and demonstrate stable system operation. Overall, the ATMS-WPT architecture provides an area-efficient, low-power, and high-speed solution that effectively addresses the limitations of conventional systolic wavelet packet transform implementations, making it highly suitable for FPGA and ASIC-based real-time signal processing systems.

References

[1]. Kumar, Deepak, and Suresh S. Gawande. "Review Paper on VLSI Architecture for 2-D Discrete Wavelet Transform using Multiplier-less Technique." *International Journal of Research & Technology* 14, no. 1 (2026): 654-661.

[2]. Kumar, Deepak, and Suresh S. Gawande. "VLSI Architecture for 5/3 2-D Discrete Wavelet Transform using Multiplier-less and Kogge Stone Adder Technique." *International Journal of Engineering Science & Humanities* 16, no. 1 (2026): 793-802.

[3]. Cardozo, Arthur, Morgana MA da Rosa, Henrique Seidel, Rodrigo Lopes, Eduardo AC Da Costa, and Rafael Soares. "A Power-Efficient Approximate Multi-Level Discrete

Haar Wavelet Transform Design for ECG Data Compression: A. Cardozo et al." *Circuits, Systems, and Signal Processing* (2026): 1-32.

[4]. Pandit, Arya, Soham Das, Arghadip Das, Debaprasad De, Arnab Raha, and Mrinal Kanti Naskar. "RamForm: A Ramanujan Wavelet Transform-Based Accelerator for Energy-Efficient Signal Processing." In *2026 39th International Conference on VLSI Design & 25th International Conference on Embedded Systems (VLSID)*, pp. 305-310. IEEE, 2026.

[5]. Nagaswetha, Gutti, B. Mokshith, S. Mohammad Arafath, and Y. Pavan Venkat. "A VLSI-Optimized, Low-Power EEG Processing Architecture for Real-Time Brain-Computer Interfaces." In *2026 IEEE International Conference on Emerging Computing and Intelligent Technologies (ICoECIT)*, pp. 1-7. IEEE, 2026.

[6]. Priya, K., and Binsu J. Kailath. "Spike-Based Time-Domain ECG Wave Delineation for Low-Power VLSI Implementation." In *2026 39th International Conference on VLSI Design & 25th International Conference on Embedded Systems (VLSID)*, pp. 413-418. IEEE, 2026.

[7]. Aissaoui, Nour Eddine, Mohamed Salah Azzaz, and Redouane Kaibou. "FPGA implementation of a real-time spread spectrum-based video watermarking system using chaotic PRNG." *Integration* (2026): 102735.

[8]. Zhang, Cheng, Rui Xing, Meng Cao, Zhuoqi Guo, Youze Xin, Bing Zhang, Zhongming Xue, and Li Geng. "Hardware-Accelerated ASIC and Cardiac Monitoring System for Wearable Devices." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2026).

[9]. Anju, M. I., Rajesh Dey, and Sai Kiran Oruganti. "VLSI-based energy-efficient image compression and encryption framework using adaptive lifting wavelet transform and CORDIC optimization: A literature review." *SGS-Engineering & Sciences* 1, no. 1 (2025).

[10]. Sahu, Prabhat Ku, Noel Prashant Ratchagar, Anubhav Bhalla, Abhinav Mishra, and Pallavi Pallavi. "VLSI Architecture Implementation of Selective Level Skip Discrete Wavelet Transform for Efficient Lossy Image Compression." In *2025 IEEE Madhya Pradesh Section Conference (MPCON)*, pp. 463-468. IEEE, 2025.

[11]. Samantaray, Aswini K., Amol D. Rahulkar, Satyajeet Sahoo, and Pradeep Gorre. "A new design and VLSI implementation of symmetric Daubechies wavelet filter bank for image processing applications." *Circuits, Systems, and Signal Processing* 44, no. 5 (2025): 3625-3640.

[12]. Dai, Yuzhou, Wei Zhang, Lin Shi, Qitao Li, Zhuolun Wu, and Yanyan Liu. "An area-efficient VLSI architecture for high-throughput computation of the 2-D DWT." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 33, no. 5 (2025): 1292-1303.