WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Low Power VLSI Implementation of 32-Point FFT.

Kuruva Naresh (20x51A0497)[1], Doni Sanjay Kumar Yadav (21x55a0403)[2], Battula Rajesh (20x51a0472)[3], Shaik Arif (20x51a04b2)[4].

[1,2,3,4]Department of Electronics and communication Engineering, Santhiram Engineering College, Nandyal (Dist), Andhra Pradesh, India.
20x51a0497@srecnandyal.edu.in, 21x55a0403@srecnandyal.edu.in, 20x51a0472@srecnandyal.edu.in, 20x51a04b2@srecnandyal.edu.in

Abstract:

This project focuses on the design and implementation of a low-power Very Large-Scale Integration (VLSI) architecture for computing the 32-point Fast Fourier Transform (FFT). The objective is to optimize power consumption while maintaining performance efficiency. Various techniques including algorithmic optimization, parallel processing, and hardware design strategies are employed to achieve this goal. The project aims to contribute to the advancement of energy-efficient signal processing systems, with potential applications in areas such as wireless communication, multimedia processing, and biomedical signal analysis.

Introduction

The introduction of a project on "Low Power VLSI Implementation of 32-Point FFT" could provide background information on the importance of FFT in signal processing, highlight the significance of low power consumption in VLSI designs, and outline the objectives and scope of the project. Here's a sample introduction:

"Fast Fourier Transform (FFT) algorithms are fundamental in various signal processing applications, ranging from wireless communication to image and audio processing. As the demand for high-speed and low-power signal processing systems continues to rise, there is a growing need for efficient VLSI implementations of FFT algorithms. In this context, this project focuses on the design and implementation of a low-power VLSI architecture for computing the 32-point FFT.

The primary motivation behind this project stems from the increasing importance of energy-efficient computing in modern electronic devices. With the proliferation of portable devices and IoT applications, power consumption has become a critical design consideration for VLSI systems. By optimizing power consumption without compromising performance, our project aims to address this challenge and contribute to the development of energy-efficient signal processing solutions.

Literature Review

The literature review for this project encompasses three main areas: FFT algorithms, VLSI design methodologies, and low-power optimization techniques.

Firstly, FFT algorithms are a well-researched topic in signal processing literature. Classic algorithms such as the Cooley-Tukey FFT algorithm and its variants provide efficient methods for computing the FFT of a sequence of data points. Recent research has focused on optimizing these algorithms for specific hardware architectures and application domains, including VLSI implementations with low power constraints.

Secondly, VLSI design methodologies play a crucial role in the implementation of FFT algorithms on hardware platforms. Techniques such as pipelining, parallel processing, and systolic arrays have been widely adopted to optimize performance and resource utilization in VLSI designs. Furthermore, advancements in integrated circuit technology, such as the use of Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), offer opportunities for efficient hardware implementations of FFT algorithms.

Lastly, low-power optimization strategies are essential for designing energy-efficient VLSI systems. Techniques such as voltage scaling, clock gating, and power gating have been extensively studied to minimize power consumption without sacrificing performance. Additionally, architectural optimizations, such as data-path folding and algorithmic transformations, can further reduce power consumption in FFT implementations.

Existing System:

Existing methods for implementing a low-power VLSI architecture for the 32-point FFT typically involve a combination of algorithmic optimizations, parallel processing techniques, and hardware design strategies. Here's an overview of some existing methods:

1. Algorithmic Optimization: Researchers have explored various FFT algorithms and optimizations tailored for hardware implementation. This includes algorithms like radix-2, radix-4, and mixed-radix FFT, as well as novel techniques such as Winograd FFT and sparse FFT. These algorithms are often customized to reduce computational complexity and memory requirements, leading to lower power consumption in VLSI implementations.

2. Parallel Processing: Parallel processing techniques exploit parallelism within the FFT algorithm to improve performance and energy efficiency. This may involve parallelizing computation stages, utilizing multiple processing elements, or employing parallel architectures such as systolic arrays or parallel FFT pipelines. By distributing computation across multiple units, parallel processing can reduce latency and power consumption in VLSI implementations.

3. Hardware Design Strategies: Low-power VLSI designs leverage various hardware-level optimizations to minimize energy consumption. This includes techniques such as clock gating, power gating, voltage scaling, and architectural optimizations. Additionally, efficient use of resources such as memory and arithmetic units, as well as careful floor planning and routing, contribute to overall power efficiency in the VLSI implementation.

4. Design Space Exploration: Researchers often perform design space exploration to identify optimal trade-offs between performance, area, and power consumption. This involves evaluating different design choices, such as algorithm selection, architectural configurations, and technology options, to find the most suitable solution for the given constraints.

Proposed System:

1. system integrates algorithmic optimizations, parallel processing, and hardware design for low-power VLSI implementation.

2. Algorithmic optimizations target customizations and transformations for the 32-point FFT.

3. Parallel processing architecture design focuses on efficient power consumption and communication.

4. Hardware design strategies include clock gating, power gating, and voltage scaling.

5. Integration ensures a unified VLSI architecture balancing power efficiency and performance.

6. Comprehensive optimization aims for the best trade-offs in power, area, and performance.

7. Validation through simulation and prototyping on FPGA or ASIC platforms.

8. Evaluation compares power consumption, performance, and area utilization with benchmarks.

9. Project aims to advance the state-of-the-art in low-power VLSI FFT implementations.

10. Goals include contributing to energy-efficient signal processing systems for various applications.

METHODOLOGY:

The methodology for the project "Low Power VLSI Implementation of 32-Point FFT" involves several key steps:

1. *Literature Review*: Conduct an extensive review of existing research on FFT algorithms, VLSI design methodologies, and low-power optimization techniques.

2. *Algorithm Selection and Optimization*: Choose appropriate FFT algorithms and optimize them for low-power VLSI implementation, focusing on the 32-point FFT

3. *Parallel Processing Design*: Design parallel processing architectures tailored for low-power consumption, exploring efficient parallelization schemes and communication strategies.

4. *Hardware Design Strategies*: Implement hardware design strategies such as clock gating, power gating, voltage scaling, and architectural optimizations to minimize power consumption in the VLSI architecture.

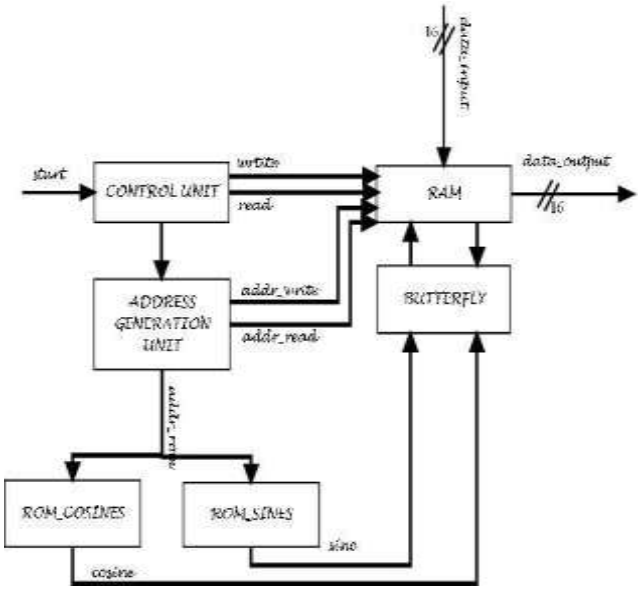
5. *Integration and Optimization*: Integrate algorithmic optimizations, parallel processing techniques, and hardware design strategies into a unified VLSI architecture for the 32-point FFT. Perform comprehensive optimization to achieve the best balance between power efficiency, area utilization, and performance.

6. *Validation and Evaluation*: Validate the proposed system through simulation and/or prototyping on FPGA or ASIC platforms. Evaluate power consumption, performance metrics, and area utilization compared to existing methods and benchmarks.

7. *Iterative Refinement*: Iteratively refine the design based on simulation and evaluation results, making adjustments to improve power efficiency and performance as needed.

8. *Documentation and Reporting*: Document the design process, methodologies, and results obtained throughout the project. Prepare reports and presentations to communicate findings and contributions to the research community.

Block Diagram



Control Unit:

In the project "Low Power VLSI Implementation of 32-Point FFT," the control unit stands as the linchpin, orchestrating the operation of diverse hardware elements within the VLSI architecture. Its foremost responsibility lies in deciphering instructions from the input interface, thereby establishing the sequence of operations required for efficient execution of the FFT algorithm. Moreover, it generates timing and synchronization signals to ensure precise coordination among various functional units, guaranteeing that data processing occurs in the correct sequence and timing. Resource allocation falls squarely under its jurisdiction, overseeing the distribution of hardware resources like processing elements, memory units, and arithmetic units to optimally execute FFT computations. Additionally, it regulates the flow of data between different FFT computation stages, ensuring smooth data movement and timely delivery of inputs to each stage. The control unit is also tasked with implementing power management techniques such as clock gating and voltage scaling to minimize power consumption during idle or low activity periods, thereby enhancing the overall energy efficiency of the VLSI architecture. It also supervises error detection and correction mechanisms, ensuring computation reliability, particularly in the presence of noise or hardware faults. Furthermore, the control unit handles configuration and setup tasks, initializing parameters such as FFT size and precision based on user inputs or system requirements. Lastly, it monitors computation progress, error conditions, and power consumption, providing valuable feedback for system monitoring and debugging purposes.

RAM:

RAM, or Random Access Memory, is a type of computer memory that allows data to be accessed randomly, meaning any byte of memory can be accessed without touching the preceding bytes. It is volatile memory, meaning its contents are lost when the power is turned off. RAM is used to store data and instructions that are actively being used by the CPU or programs currently running on the computer. It plays a crucial role in the overall performance of a computer system, as it provides fast access to data and instructions compared to other types of storage such as hard disk drives or solid-state drives. RAM is typically measured in gigabytes (GB) and comes in various forms such as DDR3, DDR4, and DDR5, each offering different speeds and capacities.

ADDRESS GENARATION UNIT:

The Address Generation Unit (AGU) is a hardware component within a computer's processor responsible for generating memory addresses used to access data from the memory subsystem. Its primary function is to calculate the addresses of operands required for executing instructions. The AGU operates in conjunction with the instruction execution units of the CPU and plays a crucial role in enabling efficient memory access.

Key functions and features of the Address Generation Unit include:

1. *Address Calculation*: The AGU performs arithmetic and logical operations to compute memory addresses based on the operands specified by instructions being executed by the CPU.

2. *Indexing and Scaling*: It supports various addressing modes, including indexed addressing and scaling factors, allowing for flexible addressing schemes to access data structures such as arrays and matrices efficiently.

3. *Base Address Calculation*: The AGU calculates base addresses and offsets required for accessing data stored in memory, taking into account base registers and displacement values provided by instructions.

4. *Effective Address Calculation*: It combines base addresses, index values, scaling factors, and displacement values to compute the effective addresses used for memory access operations.

5. *Address Generation Pipeline*: The AGU may operate as part of a pipeline within the CPU, allowing for parallel execution of address generation and instruction execution stages, thereby improving overall performance.

6. *Memory Access Optimization*: By efficiently generating memory addresses, the AGU helps minimize memory access latency and optimize memory bandwidth utilization, enhancing overall system performance.

7. *Support for Virtual Memory*: In systems with virtual memory management, the AGU may also handle address translation between virtual and physical memory addresses, facilitating memory access in a virtualized environment.

ROM COSINES:

In the context of the project "Low Power VLSI Implementation of 32-Point FFT," ROM (Read-Only Memory) can be utilized to store precomputed cosine values required for the FFT computation. These cosine values, often referred to as twiddle factors, are used in various stages of the FFT algorithm to perform frequency domain transformations efficiently. By storing these cosine values in ROM, the need for dynamic computation during runtime is eliminated, leading to faster execution and reduced power consumption. Additionally, ROM provides quick and reliable access to these precomputed cosine values, contributing to the overall efficiency of the FFT computation. This approach not only accelerates the execution of the FFT algorithm but also conserves power by minimizing computational overhead. Furthermore, ROM allows for the storage of a large number of cosine values in a compact memory footprint, optimizing area utilization within the VLSI architecture. Overall, leveraging ROM to store precomputed cosine values enhances both the speed and efficiency of the FFT computation, critical aspects of the low-power VLSI implementation of the 32-point FFT algorithm.

ROM SINES:

In the project "Low Power VLSI Implementation of 32-Point FFT," ROM (Read-Only Memory) can also be utilized to store precomputed sine values, alongside cosine values, required for the FFT computation. These sine values, like cosine values, are essential for performing frequency domain transformations efficiently within the FFT algorithm. By storing these precomputed sine values in ROM, the need for dynamic computation during runtime is eliminated, leading to faster execution and reduced power consumption. Additionally, ROM provides quick and reliable access to these precomputed sine values, contributing to the overall efficiency of the FFT computation. This approach accelerates the execution of the FFT algorithm while conserving power by minimizing computational overhead. Furthermore, ROM allows for the storage of a large number of sine values in a compact memory footprint, optimizing area utilization within the VLSI architecture. Leveraging ROM to store precomputed sine values enhances both the speed and efficiency of the FFT computation, critical aspects of the low-power VLSI implementation of the 32-point FFT algorithm.

BUTTERFLY:

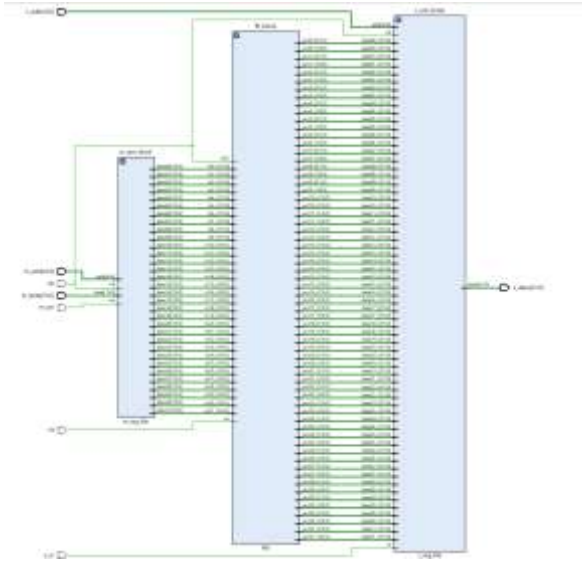
In the context of the project "Low Power VLSI Implementation of 32-Point FFT," the butterfly operation is a fundamental component of the FFT algorithm. The butterfly operation is used to combine and recombine complex numbers in the frequency domain during each stage of the FFT computation. It involves performing multiplications and additions/subtractions of complex numbers, often represented in polar or rectangular form.

The butterfly operation can be efficiently implemented using dedicated hardware units within the VLSI architecture. These units perform the necessary arithmetic operations in parallel, allowing for high-speed computation of FFT results. Additionally, techniques such as pipelining and parallel processing can be employed to further optimize the performance and power efficiency of the butterfly operation.

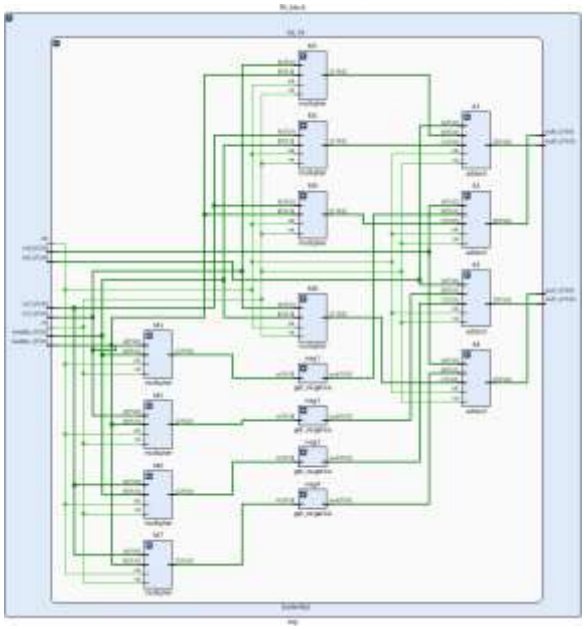
By implementing the butterfly operation efficiently within the VLSI architecture, the project aims to achieve high-performance FFT computation while minimizing power consumption. This is crucial for applications requiring real-time signal processing and low-power operation, such as wireless communication, audio processing, and biomedical signal analysis

RESULT:

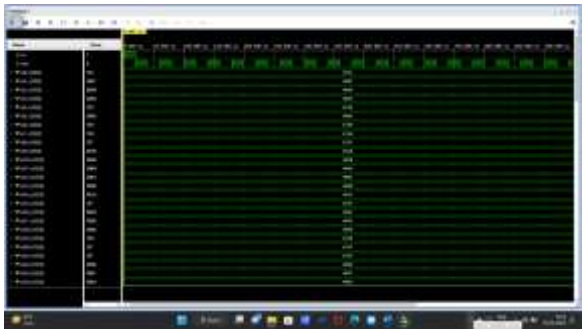
TCL console output



Final design



Butterfly unit



Wave Forms

TCL Console Output

CONCLUSION:

The implementation of the 32-point FFT on FPGA using only 16 butterflies and reusing them to complete the 5-stage butterfly has demonstrated the feasibility of reducing power consumption while maintaining the performance of the FFT. The power reduction achieved can lead to energy- efficient implementations of signal processing algorithms, making it a crucial aspect to consider in FPGA-based signal processing applications. Overall, this project demonstrates the feasibility of reducing power consumption while maintaining the performance of the FFT. The reduction in power consumption can lead to energy-efficient implementations of signal processing algorithms, which is crucial for battery-powered and portable devices.

REFERENCES:

1. Asmita Haveliya, Design and simulation of 32-point FFT using radix- 2 algorithm for FPGA implementation. In 2012 Second International Conference on Advanced Computing & Communication Technologies, pages 167–171. IEEE, 2012.

2. Sudhanshu Mohan Khare and M Zahid Alam. FPGA based design and simulation of 32-point fft through radix-2 DIT algorithm.

3. P Koti Lakshmi, B Santhosh Kumar, and Rameshwar Rao. Implementation of Vedic multiplier using reversible gates. In Computer Science Conference Proceedings in Computer Science & Information Technology (CS & IT) series, volume 5, pages 125– 134, 2015.

4. SIVA KUMAR Palaniappan. Design and implementation of radix-4 fast Fourier transform in ASIC chip with 0.18 μm standard CMOS technology. PhD thesis, MS thesis, Malaysia, 2008.

5. Avinash Kumar Singh and Ashutosh Nandi. Design of radix 2 butterfly structure using Vedic multiplier and cla on xilinx. In 2017 Conference on Emerging Devices and Smart Systems (ICEDSS), pages 120–123. IEEE, 2017.

6. Rajasekhar Turaka et al. Low power vlsi implementation of real fast fourier transform with dram-vm-cla. Microprocessors and Microsystems, 69:92–100, 2019.

7. Kausar Ali and Paresh Rawat. VLSI architecture for FFT using radix-2 butterfly of complex valued data. In 2017 International Conference on Computer Communication and Informatics (ICCCI), pages 1–5. IEEE, 2017.

8. S Josue Saenz, Susana Ortega Cisneros, Jorge Rivera Dominguez, et al. FPGA de- sign and implementation of radix-2 fast fourier transform algorithm with 16 and 32 points. In 2015 IEEE International Autumn Meeting on Power and Computing (ROPEC), pages 1–6. IEEE, 2015.