Volume: 13 Issue: VI Month of publication: June 2025
DOI: https://doi.org/10.22214/ijraset.2025.72208

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VI June 2025- Available at www.ijraset.com

Design and Synthesis of Optimized RISC-Based Arithmetic Logic Unit (ALU) using Cadence Genus

Bathula Nagarjuna[1], Madhuri Vemuluri[2]
[1]M-Tech Scholar, [2]Associate Professor, Department of Electronics and Communication Engineering, Newtons Institute of Engineering, Macherla-Andrapradesh, India

Abstract: This work presents the design and optimization of a RISC-based Arithmetic Logic Unit (ALU), a critical component in modern computing architectures that require efficient computation with minimal power consumption and reduced silicon area. Through a systematic approach that incorporates architectural modeling, RTL implementation, and synthesis using Cadence Genus, the study emphasizes the transformation of high-level designs into optimized silicon realizations capable of meeting stringent performance, area, and power constraints. The design leverages the principles of Reduced Instruction Set Computing (RISC), focusing on a simplified instruction set that allows for faster execution cycles and enhanced scalability. The ALU architecture is rigorously verified through extensive simulations to ensure functional correctness before synthesis, showcasing improvements in logic delays and a significant reduction in hardware footprint. The results demonstrate a well-balanced design that not only achieves high performance but also adheres to the efficiency demands of embedded systems and application-specific integrated circuits (ASICs). Additionally, the project highlights the critical role of synthesis-driven methodologies in facilitating the optimization of digital designs and lays the groundwork for future enhancements, including adaptive optimization strategies and the exploration of emerging technologies. This research contributes to the ongoing evolution of processor design by delivering a robust ALU framework tailored for high-performance applications, positioning it as an asset for both academic and industrial implementation in the ever- expanding landscape of digital technology.
Keywords: RISC-based Arithmetic Logic Unit (ALU), Reduced Instruction Set Computing (RISC), application-specific integrated circuits (ASICs), industrial implementation.

I. INTRODUCTIONS

The development of efficient digital processors has become a fundamental aspect of modern computing systems. Among these, Reduced Instruction Set Computing (RISC) architectures have gained prominence due to their simplified instruction set and enhanced execution speed. A core component of such architectures is the Arithmetic Logic Unit (ALU), responsible for performing essential arithmetic and logical operations. The design of an optimized ALU is crucial for improving the overall performance and resource utilization of RISC-based processors, especially in applications requiring high-speed computation and low power consumption. To achieve an effective balance between performance, area, and power, designers often turn to advanced Electronic Design Automation (EDA) tools. Cadence Genus, a widely adopted RTL synthesis tool, provides comprehensive capabilities for optimizing digital designs at the register-transfer level. Through automated technology mapping, logic optimization, and timing-driven synthesis, Cadence Genus helps in refining the ALU architecture to meet the stringent design specifications of modern embedded systems. The use of such tools is essential in translating high-level hardware descriptions into gate-level netlists optimized for fabrication. This work focuses on the complete design flow of a RISC-based ALU, starting from architectural modeling to synthesis and analysis using Cadence Genus. Emphasis is placed on optimizing datapath components and ensuring efficient utilization of silicon resources while meeting timing constraints. The synthesized design is evaluated based on key parameters such as delay, area, and power, providing insights into the trade-offs involved in hardware implementation. The resulting ALU not only demonstrates functional correctness but also adheres to the design metrics demanded by high-performance RISC processors. Application-Specific Integrated Circuit (ASIC) design for a RISC-based Arithmetic Logic Unit (ALU) focuses on creating a tailored hardware block that delivers efficient arithmetic and logical computations. Unlike general-purpose implementations, an ASIC approach is optimized for specific performance goals, such as reduced power consumption, area efficiency, and higher speed. In the context of a RISC architecture, the ALU must support a limited but essential set of operations executed rapidly and reliably. As ASIC designs are meant for dedicated deployment, this implementation ensures the hardware is fine-tuned for real-world applications with minimal overhead.

Modern digital systems demand efficient arithmetic and logical computation units that can deliver high performance without compromising power and area constraints. Traditional ALU designs often struggle to meet the increasing need for optimization in terms of speed, power efficiency, and silicon footprint, particularly in embedded and application-specific environments. As processor architectures evolve, there is a critical need to develop streamlined, instruction-efficient Arithmetic Logic Units based on the Reduced Instruction Set Computing (RISC) methodology.

Conventional ALUs, although functional, are not always optimized for integration into low- power or high-speed digital systems. This poses a significant limitation when deploying such units in real-time or resource-constrained environments. Without architectural optimization and synthesis-level refinement, these ALUs can become bottlenecks in data processing pipelines. Moreover, the absence of a structured synthesis workflow can lead to inefficient hardware utilization, which negatively impacts overall system performance.

This project addresses the challenge by focusing on the design and synthesis of a RISC-based ALU using Cadence Genus, a leading digital synthesis tool. The aim is to develop a hardware-efficient ALU architecture that supports fundamental operations with minimal latency, reduced power dissipation, and optimal use of silicon area. By leveraging synthesis- driven optimization techniques, the project intends to demonstrate an improved design flow for implementing arithmetic and logical operations in modern processor cores.

A. Objectives of the Work

The development of high-performance digital systems calls for compact, reliable, and efficient arithmetic units that can perform a variety of operations with speed and precision. Within the context of processor design, the Arithmetic Logic Unit (ALU) serves as the core computational component responsible for executing arithmetic and logical functions. Adopting a RISC (Reduced Instruction Set Computing) approach allows for streamlined instruction execution, resulting in faster and more efficient data processing. To meet the growing complexity of modern embedded and computing systems, optimizing the ALU's performance, area, and power profile has become a key objective.

This project aims to design a RISC-based ALU architecture that is not only functionally robust but also optimized for implementation using Cadence Genus—a leading synthesis tool in the ASIC design workflow. The focus is on minimizing logic delays, reducing hardware footprint, and achieving efficient gate-level synthesis without compromising on functionality. By carefully mapping the RISC instruction set to the ALU's capabilities, the design seeks to achieve high throughput and predictable performance suitable for integration into scalable processor architectures.

The optimization process also includes the evaluation of critical design parameters such as timing, power consumption, and area efficiency post-synthesis. Through this work, the broader objective is to demonstrate a structured design methodology that leverages RISC principles and state-of-the-art synthesis tools to build optimized ALUs suitable for advanced digital systems, including IoT devices, real-time control systems, and embedded processors.

- To identify the low power logical circuits for the construction of low power RISC based ALU
- To develop the hardware description language for functional verification of the design.
- To synthesize the complete architecture using Genus tool and generate the optimized reports such as power, area, timing.
- To compare the performance of the design with existing designs.

II. METHODOLOGY OF THE WORK

The design and synthesis of an optimized RISC-based Arithmetic Logic Unit (ALU) involves a structured approach comprising architectural planning, behavioral modeling, RTL implementation, functional simulation, and synthesis using industry-standard EDA tools. The methodology adopted for this project is outlined in the following key phases:

The initial phase involves identifying the functional requirements of the RISC-based ALU, including supported arithmetic and logical operations, operand width (typically 16-bit or 32- bit), and control signal interface. Based on these requirements, a modular architecture is defined, comprising operational blocks such as the adder/subtractor unit, logic unit, shift unit, and multiplexer-based control path.

1) RTL Design Using Verilog HDL: Each functional unit of the ALU is described using Verilog Hardware Description Language. The ALU control logic is implemented to decode operation select lines and generate control signals accordingly. The design adheres to synthesizable coding practices, ensuring compatibility with logic synthesis tools. Modules are designed hierarchically, allowing for improved readability and modular testing.

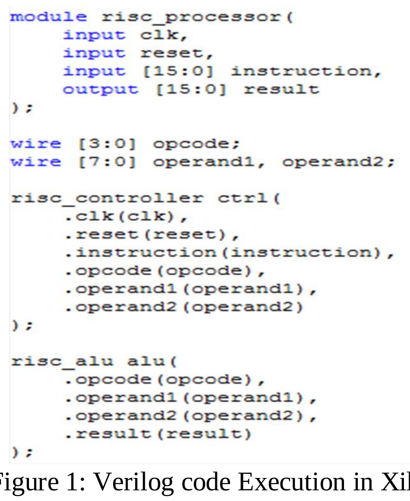
2) Functional Verification via Simulation: After RTL coding, each module is subjected to extensive simulation using testbenches written in Verilog. Behavioral simulations are performed to validate the correctness of the ALU operations under all possible input scenarios. The verification process includes checking the arithmetic overflow, zero flag, sign flag, and carry outputs, along with logic operation accuracy. Simulations are conducted using industry tools such as ModelSim or XSIM.
3) Digital design flow: The Register-Transfer-Level (RTL) netlist serves as the foundational input for the logic synthesis process, defining the digital circuit's behavior through hardware description languages like VHDL or Verilog. This abstract representation undergoes transformation into a logic-gate-level netlist using synthesis tools, which map the design to a target technology library while adhering to constraints such as sequential timing, area, and power consumption. The synthesis phase plays a critical role in bridging high-level design specifications with physical implementation, ensuring the design meets performance and efficiency goals before progressing to placement and routing stages.

During synthesis, libraries provide technology-specific cell definitions, enabling tools to optimize the netlist for area, speed, or power based on design priorities. Logic synthesis tools analyze the RTL description, apply constraints, and generate a gate-level netlist alongside detailed reports on sequential performance, area utilization, and power estimates. These reports guide iterative refinements to balance trade-offs between hardware resources and operational efficiency, forming a crucial step in ASIC and FPGA design flows for achieving optimized digital circuits. This study employs a systematic approach to design and evaluate low-power compressors, beginning with a critical review of existing architectures to identify optimization opportunities. Proposed designs are modeled at the gate and transistor levels, incorporating logic simplification, voltage scaling, and alternative logic styles to minimize power while preserving performance. Rigorous verification is conducted using industry-standard EDA tools, with power, delay, and area metrics benchmarked against conventional designs. The methodology further validates practical applicability by integrating optimized compressors into multiplier circuits, assessing their impact on real-world applications like AI acceleration and signal processing through standardized evaluation frameworks.

III. FUNCTIONAL VERIFICATION OF RISC-BASED ARITHMETIC LOGIC UNIT (ALU)

Functional verification of low-power compressors involves rigorous testing to ensure correct arithmetic operations while meeting power efficiency targets. Testbenches are developed to apply exhaustive input combinations, including edge cases, to validate the compressor’s behavior under typical and worst-case scenarios. Power-aware simulations track dynamic and leakage power consumption during operation, ensuring the design adheres to predefined energy budgets. Metrics such as error rate, propagation delay, and power-delay product are analyzed to verify compliance with both functional and low-power objectives. This step is critical to identify logic flaws or unintended power overheads before physical implementation.

To enhance verification coverage, advanced techniques like assertion-based checking and constrained random testing are employed. These methods systematically explore the compressor’s response to diverse input patterns, including those specific to approximate computing applications where minor errors are permissible. Tools such as ModelSim or VCS correlate simulation results with RTL and gate-level models, while power analysis tools (e.g., PrimeTime) quantify energy savings. By cross-validating results against golden reference models, the verification process ensures the compressor maintains reliability in real-world deployments, such as AI accelerators or DSP units, where power efficiency and functional accuracy are equally critical.



```
module risc_processor(
    input clk,
    input reset,
    input [15:0] instruction,
    output [15:0] result
);

wire [3:0] opcode;
wire [7:0] operand1, operand2;

risc_controller ctrl(
    .clk(clk),
    .reset(reset),
    .instruction(instruction),
    .opcode(opcode),
    .operand1(operand1),
    .operand2(operand2)
);

risc_alu alu(
    .opcode(opcode),
    .operand1(operand1),
    .operand2(operand2),
    .result(result)
);
```

Figure 1: Verilog code Execution in Xilinx

RTL Schematic: Within the ALU module (risc_alu), the received opcode and operands are used to perform the specified arithmetic or logic operation. The result of this operation is output through a 16-bit result bus. This output can be used for further processing or stored in memory. The clean separation between the controller and ALU reflects the modular design principle of RISC (Reduced Instruction Set Computing) architecture, where each component has a specific role—control logic is isolated from data processing. This simplicity helps in faster instruction execution and ease of debugging or extension of the processor's functionality.

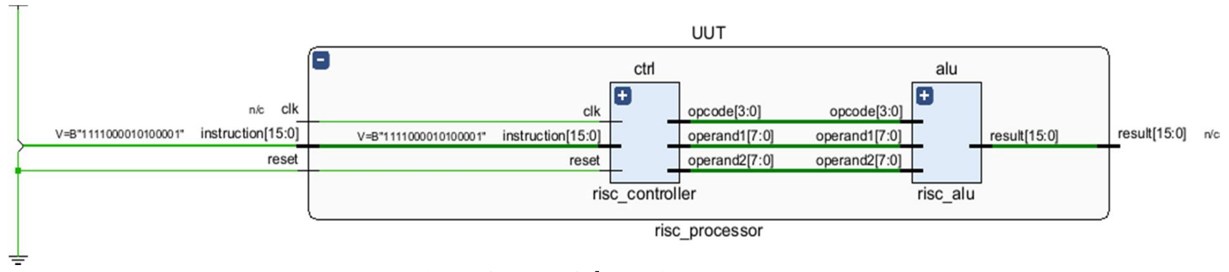


Figure 2: RTL Schematic

A. Simulation Results

The simulation waveform represents the timing behavior of a RISC-based ALU in operation. Key signals shown include the clock (`clk`), reset, instruction input (`instruction[15:0]`), and output result (`result[15:0]`). The clock signal drives the processor, and each rising edge triggers the next step in execution. The reset signal is held low (inactive), allowing the processor to operate normally. Initially, the instruction bus contains undefined data (`XXXX`), and the result is zero. As valid instructions are fed at subsequent clock cycles (such as `00a4`, `1322`, `2067`, etc.), the result changes accordingly, reflecting the ALU's execution of operations encoded in those instructions.

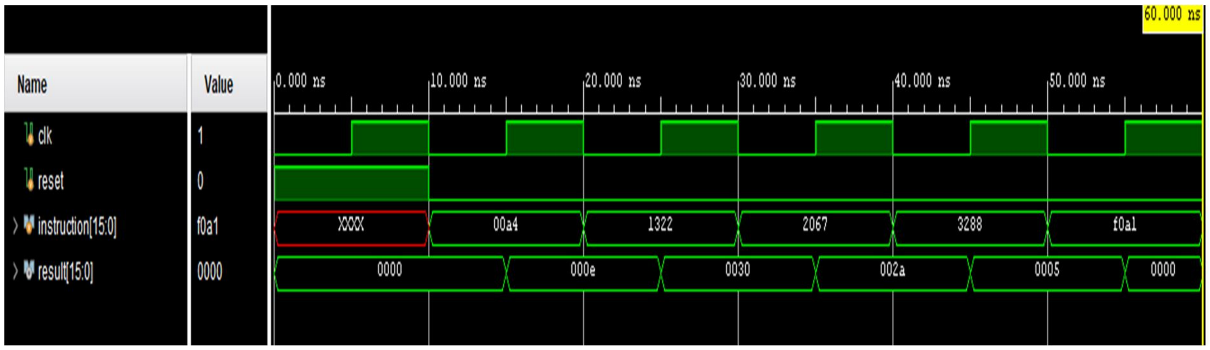


Figure 3: Functional verification of the results

Table 1: possibly no operation or end-of-instruction

Time (ns)	instruction	operation	results
10	00A4	ADD	14
20	1322	SUB	48
30	2067	MUL	42
40	3288	DIV	5
50	F0A1	HALT/IO	0

B. Synthesis through FPGA

The synthesis illustrates the layout view of the synthesized RISC-based ALU on an FPGA, showcasing the post-synthesis floorplan generated after logic synthesis and mapping. Each colored block (e.g., X0Y0, X1Y1) represents configurable logic blocks (CLBs) and their allocated regions on the FPGA fabric. These regions are automatically defined during the place-and-route phase, where logic gates and interconnections required for the ALU are optimally positioned to minimize delay and resource usage. The design appears to be modular, with functional units of the ALU distributed across different logic regions, helping to balance routing congestion and ensure timing closure.

This layout reflects a well-structured synthesis process where the high-level RTL code of the RISC-based ALU has been translated into a gate-level netlist and finally mapped onto the FPGA's architecture. The presence of multiple logic tiles and routing tracks indicates efficient resource utilization and parallelism within the ALU operations. Each logic block handles different instruction decoding, arithmetic, or control tasks, depending on its logical assignment. Overall, the floorplan confirms that the RISC-ALU has been successfully synthesized, placed, and routed for FPGA implementation, indicating readiness for bitstream generation and hardware testing.

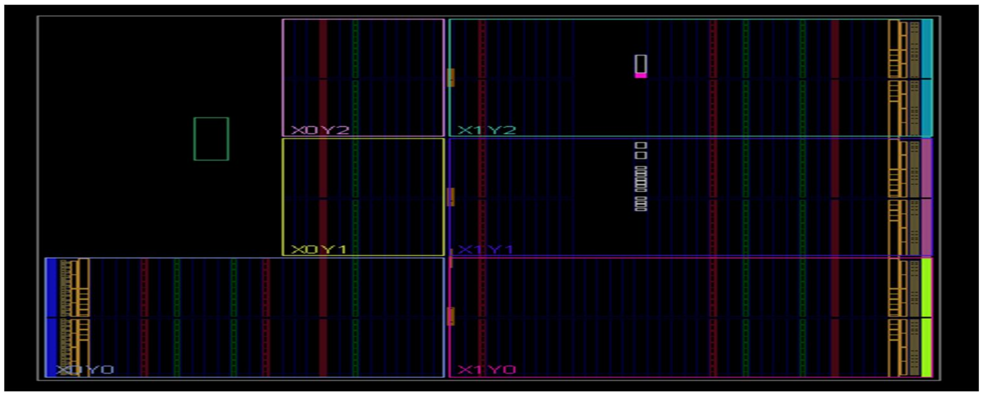
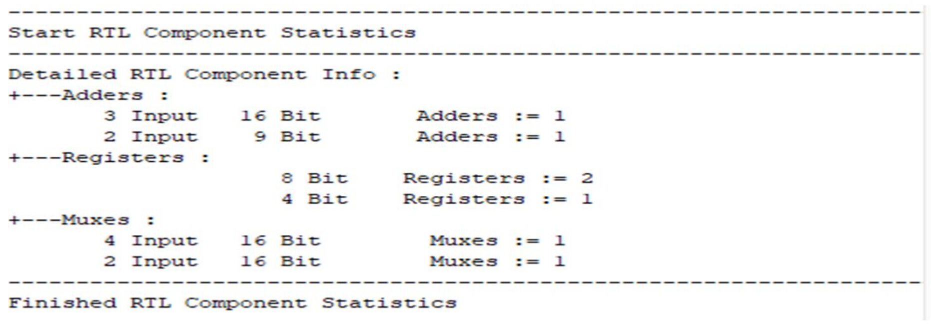


Figure 4: FPGA Synthesis Implementation



```
---------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
       3 Input   16 Bit       Adders := 1
       2 Input    9 Bit       Adders := 1
+---Registers :
                  8 Bit    Registers := 2
                  4 Bit    Registers := 1
+---Muxes :
       4 Input   16 Bit        Muxes := 1
       2 Input   16 Bit        Muxes := 1
---------------------------------------------------------------------
Finished RTL Component Statistics
```

The functional verification and implementation of a RISC-based Arithmetic Logic Unit (ALU) using a low-power 4:2 compressor design in Verilog HDL. It emphasizes rigorous testing using simulation tools (e.g., ModelSim) to ensure functional correctness, low power consumption, and reliability for real-world applications. The synthesis enhances multiplication efficiency by reducing logic levels and power usage. The RTL schematic demonstrates a modular RISC architecture with separate controller and ALU components. Simulation waveforms confirm proper instruction decoding and operation execution across clock cycles, validating the design's functionality and synchronization.

IV. SYNTHESIS OF LOW POWER COMPRESSOR

Simulation following synthesis is essential to verify that the low-power optimizations do not affect the intended behavior of the design. Post-synthesis simulation uses the gate-level netlist generated by Genus and includes switching activity to reflect real power usage scenarios. The waveform outputs confirm that the compressed partial product generation logic remains intact, and the timing requirements are satisfied. This step ensures the compressor is not only functionally correct but also optimized for deployment in power- sensitive applications such as portable communication devices or battery-operated systems.

The synthesis procedure involves converting the RTL (Register Transfer Level) Verilog code into a gate-level netlist using a synthesis tool like Cadence Genus. The process begins by importing the RTL design and setting up the design constraints, including timing, area, and power requirements. The tool then analyzes the design, performs optimization, and maps the logic to technology-specific standard cells. During this step, techniques such as logic minimization, retiming, and resource sharing are applied to improve performance and reduce power consumption. Finally, the tool generates reports detailing timing, area, and power estimates, and outputs a gate-level netlist ready for further verification or physical implementation.

The schematic represents a Register-Transfer Level (RTL) block diagram synthesized using the Cadence Genus tool, showcasing a RISC-based Arithmetic Logic Unit (ALU) integrated with a controller. The input signals—clock (clk), reset (reset), and a 16-bit instruction (instruction[15:0])—are fed into the control unit (ctrl), which decodes the instruction and generates control signals. These signals determine the operations to be executed by the ALU. The controller produces the opcode, operand1, and operand2, which are routed to the ALU module. The modular design separates control logic and datapath operations, facilitating hierarchical synthesis and efficient verification.

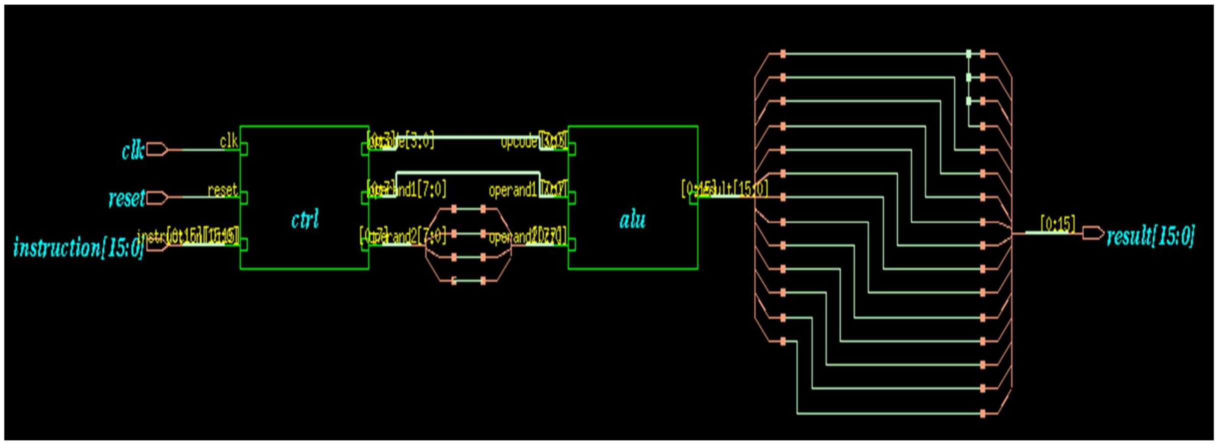
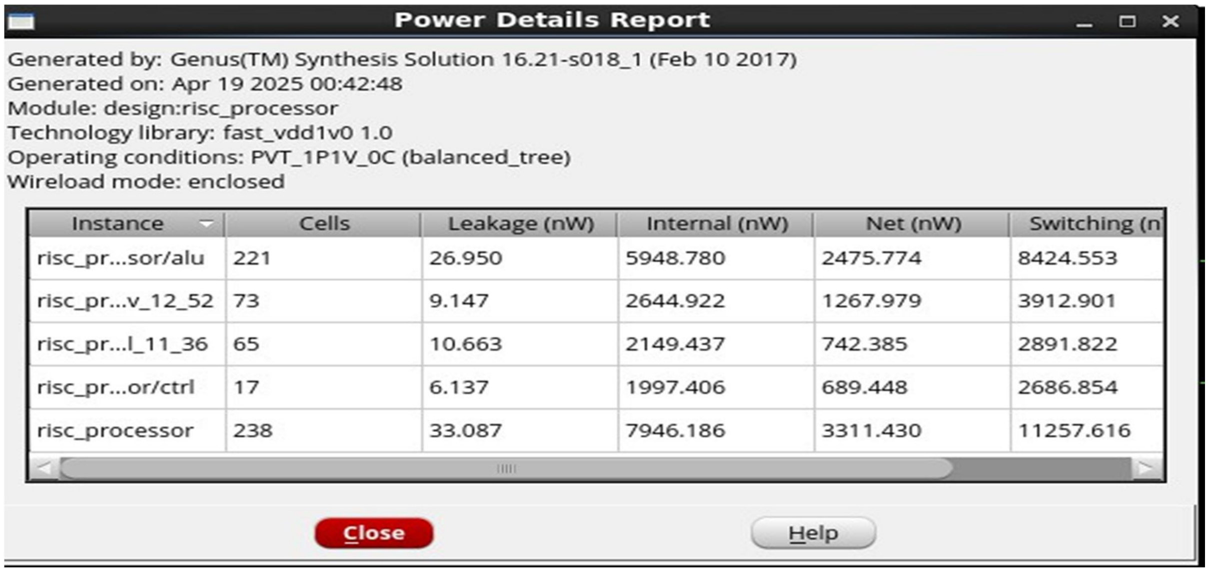


Figure 5: Synthesis

Power Report: The schematic represents a Register-Transfer Level (RTL) block diagram synthesized using the Cadence Genus tool, showcasing a RISC-based Arithmetic Logic Unit (ALU) integrated with a controller. The input signals—clock (clk), reset (reset), and a 16-bit instruction (instruction[15:0])—are fed into the control unit (ctrl), which decodes the instruction and generates control signals. These signals determine the operations to be executed by the ALU. The controller produces the opcode, operand1, and operand2, which are routed to the ALU module. The modular design separates control logic and datapath operations, facilitating hierarchical synthesis and efficient verification.

The ALU processes the 8-bit operands based on the received opcode and outputs a 16-bit result. The final stage involves routing the result through combinational logic that directs the output to result[15:0]. The synthesis performed in Genus ensures that the design is optimized for timing, power, and area under specific constraints. The RTL netlist generated by Genus preserves the modular structure, allowing post-synthesis simulations and formal verification to validate the logical and functional behavior of the design. This approach guarantees that the ALU and controller function coherently across clock cycles, suitable for deployment in low-power embedded systems.



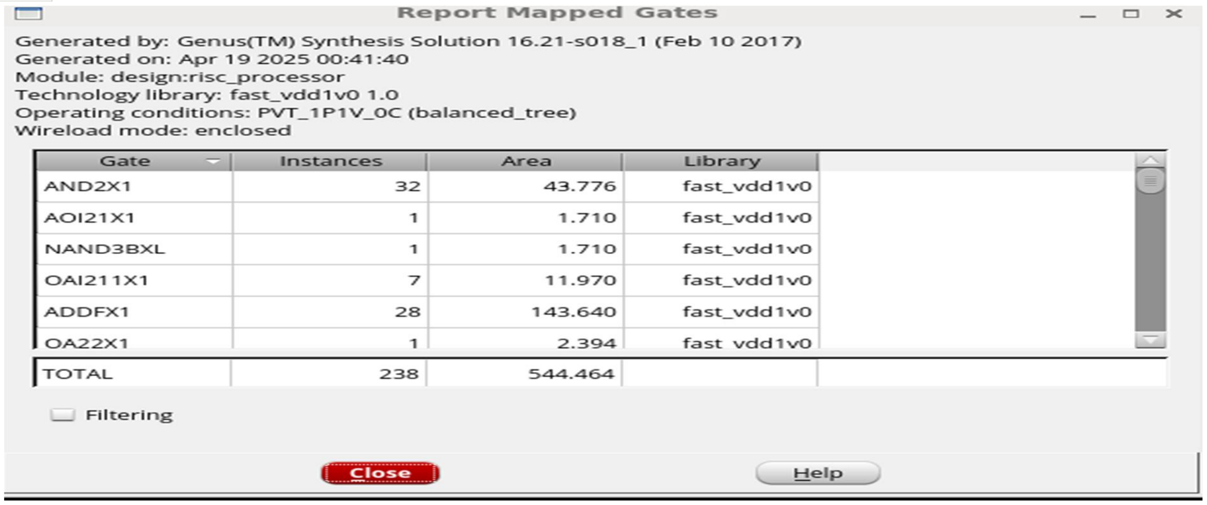
Power Details Report

Generated by: Genus(TM) Synthesis Solution 16.21-s018_1 (Feb 10 2017)
Generated on: Apr 19 2025 00:42:48
Module: design:risc_processor
Technology library: fast_vdd1v0 1.0
Operating conditions: PVT_1P1V_0C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (n
risc_pr...sor/alu	221	26.950	5948.780	2475.774	8424.553
risc_pr...v_12_52	73	9.147	2644.922	1267.979	3912.901
risc_pr...l_11_36	65	10.663	2149.437	742.385	2891.822
risc_pr...or/ctrl	17	6.137	1997.406	689.448	2686.854
risc_processor	238	33.087	7946.186	3311.430	11257.616

Close Help

Figure 6: Power report


Report Mapped Gates

Generated by: Genus(TM) Synthesis Solution 16.21-s018_1 (Feb 10 2017)
Generated on: Apr 19 2025 00:41:40
Module: design:risc_processor
Technology library: fast_vdd1v0 1.0
Operating conditions: PVT_1P1V_0C (balanced_tree)
Wireload mode: enclosed

Gate	Instances	Area	Library
AND2X1	32	43.776	fast_vdd1v0
AOI21X1	1	1.710	fast_vdd1v0
NAND3BXL	1	1.710	fast_vdd1v0
OAI211X1	7	11.970	fast_vdd1v0
ADDFX1	28	143.640	fast_vdd1v0
OA22X1	1	2.394	fast_vdd1v0
TOTAL	238	544.464	

Figure 7: Area report

Discusses the synthesis of a low-power RISC-based ALU using the Cadence Genus tool, emphasizing optimization for power, timing, and area. The process involves RTL-to- gate-level conversion using techniques like clock gating and logic restructuring to minimize power without affecting functionality. Post-synthesis simulations verify the accuracy of the gate-level netlist. The modular design—featuring separate control and datapath logic— supports hierarchical synthesis and efficient verification. Detailed reports from Genus provide insights into power usage, area consumption by standard cells, and timing paths, enabling thorough analysis and optimization for low-power embedded system applications.

V. CONCLUSIONS

The design and synthesis of a RISC-based Arithmetic Logic Unit (ALU) presented in this work highlight the critical role of optimization in enhancing performance, power efficiency, and area utilization in modern digital systems. By leveraging advanced EDA tools, particularly Cadence Genus, the project demonstrates a structured approach that encompasses architectural modeling, RTL implementation, and rigorous functional verification. The optimized ALU not only fulfills the fundamental arithmetic and logical operations expected in RISC architectures but also ensures that it adheres to stringent timing and power constraints. This dual focus on functionality and optimization underlines the importance of integrating cutting-edge design methodologies in developing efficient computational units. Furthermore, the work emphasizes the significance of adopting a modular and scalable approach in ALU design, which aligns well with the principles of RISC architecture. The findings illustrate that through careful design choices and synthesis techniques, it is possible to achieve a compact yet robust ALU that performs efficiently across various operational scenarios. Future improvements may involve exploring adaptive optimization techniques and the integration of emerging technologies to further enhance performance and resource utilization in next-generation digital systems. The project sets a benchmark for subsequent research in ALU design and offers insights that could assist in addressing the evolving demands of embedded and application-specific environments.

REFERENCES

[1] Jonathangerrans “8 Bit Arithmetic Logic Unit”Deptment Of Electrical And Computer Engineering University of Mainae.
[2] Rathinjoshi, Yashagarwal, Rutuparekh “Design and Optimization of Single Electron Transistor Based 4 Bit Arithmetic and Logic Unit at Room Temperature Operation”, IEEE International Symposium On Nano electronic And Information System 2017.
[3] Mr.Snehalkumbalkar, Prof.Sanjaytembhume ”A Novel Arithmetic Logic Unit Design For Dely And Area Optimization”, Journal Of Information Knowledge And Research In Electronics And Communication,2016.
[4] Sreeja S Kumar, Rakesh S “Design Of 4 Bit Arithmetic And Logic Unit Using 9t Full Adder With Optimized Area And Speed", International Journal Of Engineering And Advanced Technology 2019.
[5] G.Manikumar,B.V.Rammana,G.M.V.Prasad,”an arithmetic and logic unit optimized for area and power", international journal and magazine of engineering technology management and research 2016
[6] Deeptha A, Pratiksha M, Dhrithi M, Drisika muthanna,B.S.Kariyappa, ”Design And Optimization Of 8 Bit ALU Using Reversible Logic” , IEEE International Conference On Recent Treands In Electronic Information Communication Technology 2016, India.
[7] S.Swetha,M.D.Afeenbegum,”Design Of High Speed Area Optimized And Low Power Arithmetic And Logic Unit"Swethal Advances In Industrial Engineering And Mangment 2015.
[8] R.Durgabhavani, V.Shilpakesav,”Efficient Design Of Low Power 4 Bit ALU Using HVT Cell Concept”, CVR Journal Of Science And Technology 2015.