IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Design And Performance Analysis Of Subsystem Components Of Risc Processor

[1]G Nandan Kumar Rao, [2]Dr. Jeeru Dinesh Reddy

[1]Student, [2]Assistant Professor
[1]Department of ECE
[1]B.M.S College of Engineering, Bengaluru, India

Abstract: This report presents the design and performance analysis of key subsystem components in a 16-bit Reduced Instruction Set Computing (RISC) processor. The study focuses on three primary adder implementations: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA). These adders play a vital role in defining the arithmetic processing capabilities of the RISC processor. The project evaluates the impact of each adder architecture on critical performance metrics, including execution speed, area utilization, and power consumption. Simulations were conducted using Xilinx ISE, and performance was measured to assess the efficiency of each adder in various scenarios. Results indicate that while the CLA provides superior speed due to minimized carry propagation delays, the RCA excels in space efficiency, making it more suitable for compact designs. The CSA offers a balanced performance between speed and power efficiency, making it a viable option for high-performance computing tasks. Overall, the design and analysis provide valuable insights into the trade-offs between speed, area, and power consumption in processor architecture, leading to an optimized RISC processor design

Index Terms : **RISC – Reduced Instruction Set**

I. INTRODUCTION

Reduced Instruction Set Computing (RISC) is a type of computer architecture that emphasizes a small, highly optimized set of instructions. Its primary goal is to simplify instruction execution by using fewer fundamental operations, contrasting with Complex Instruction Set Computing (CISC), which uses a more extensive instruction set.

Key characteristics of RISC include a Simplified Instruction Set, which consists of basic operations for easier decoding and execution. Uniform Instruction Format and Load/Store Architecture simplify fetching, decoding, and memory operations. Register Usage is prominent, reducing memory access for improved performance. Execution breaks down instruction processing into stages for simultaneous execution. RISC processors also use Hardwired Control Units, enhancing speed through direct control signal generation, and their design favors Compiler Optimization for efficient code execution

A 16-bit RISC processor handles 16-bit data chunks and follows the core principles of RISC. It uses a simplified instruction set for efficient single-cycle execution

With Registers reducing memory access, the processor improves performance. The Load/Store Architecture ensures operations occur in registers, while Fixed-Length Instructions allow faster execution and decoding. The processor's design also focuses on Low Power Consumption, suitable for embedded and low-power applications.

ALU- Arthematic and Logical UnitThe ALU performs arithmetic and logical operations. It takes input from the instruction's opcode and operands, processing them and storing results in registers.

Register Bank: The register bank stores intermediate data, providing a space for the ALU to store rsults. The Z-Flag indicates operation success or failure,and registers improve efficiency in memory based data operations

CONTROL UNITS AND REGISTERS

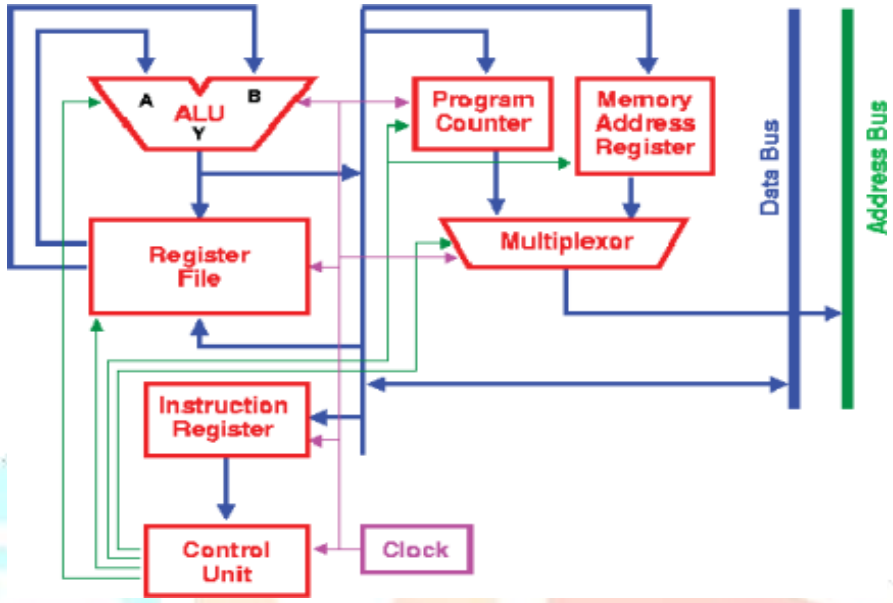
Instruction Register (IR): Holds the instruction to be executed. The Control Unit decodes the opcode, retrieves operands, and executes the instruction with the help of timing signals.

Program Counter (PC): The PC points to the next instruction in memory, ensuring smooth instruction flow in the processor

.

Memory Address Register (MAR): The MAR holds the memory address for the current instruction, loading instructions into

Multiplexer (MUX): The multiplexer selects inputs and provides a single output, controlling data flow within the processor



Ripple Carry Adder (RCA):

performs basic arithmetic, sequentially processing bits and carry-outs. Its simplicity and space-efficiency make it ideal for a 16-bit processor.

Carry Look-Ahead Adder (CLA):

CLA precomputes carry signals to reduce delays, significantly improving arithmetic performance in the 16-bit processor

Carry Select Adder (CSA):

CSA parallelizes carry paths, optimizing speed and efficiency, making it a vital part of the processor's arithmetic unit

II. PROBLEM STATEMENT

The project aims to address the critical challenge of optimizing the design and performance of a 16-bit Reduced Instruction Set Computing (RISC) processor by employing distinct adder datapath logic cells. The specific focus lies on three primary adder implementations: Ripple Carry Adder (RCA), Carry Look- Ahead Adder (CLA), and Carry Select Adder (CSA). Each adder logic cell introduces unique characteristics influencing aspects such as execution speed, area utilization, and power consumption. The challenge is to comprehensively investigate the impact of these varied adder datapath logics on the overall efficiency of the 16-bit RISC processor.

III. OBJECTIVE

Define RISC ISA

The objective is to develop a streamlined RISC Instruction Set Architecture (ISA) that prioritizes simplicity and efficiency. This involves creating a minimal set of instructions that can perform essential operations effectively while minimizing complexity. A reduced instruction set is expected to enhance the processor's performance, optimize resource use, and simplify programming. The goal is a compact ISA that executes instructions rapidly,

ultimately improving processing speeds and overall processor performance

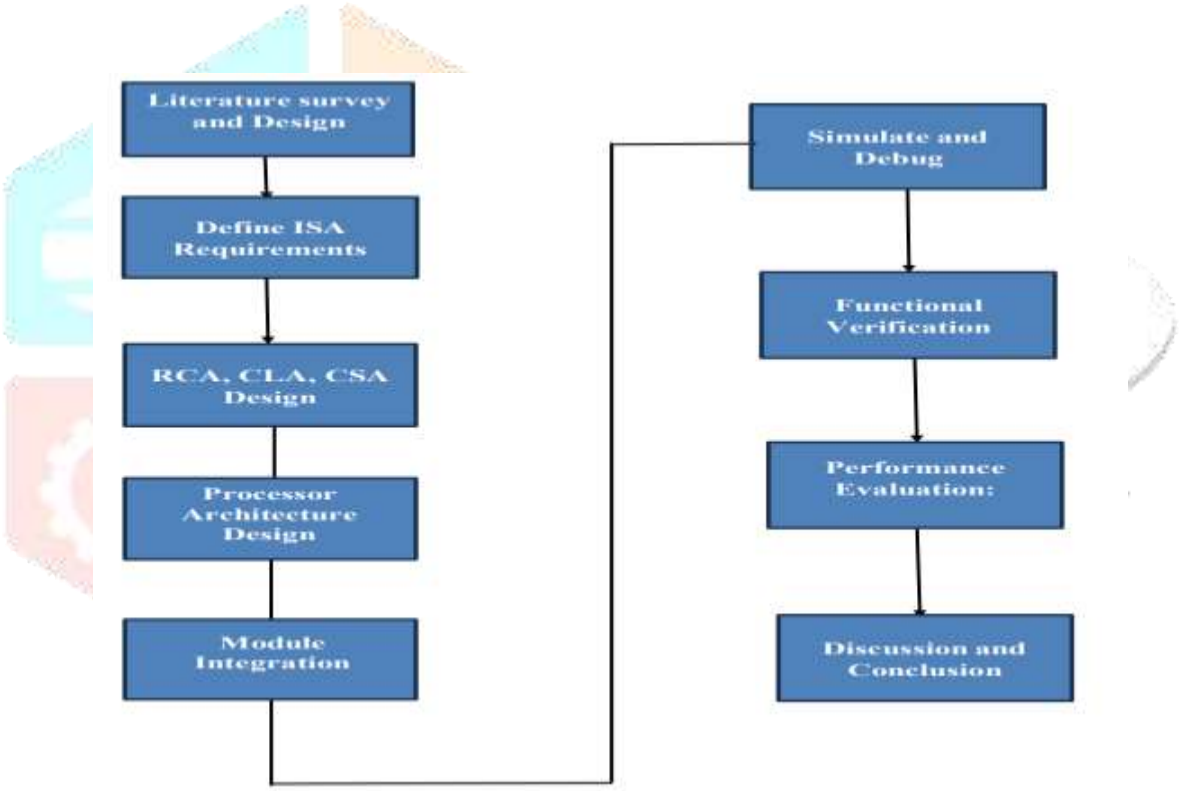
Implement Different Adders:

Enhancing arithmetic capabilities by integrating various adder architectures is another key objective. The processor will include ripple carry adders, carry select adders, and Wallace tree multipliers, each offering specific advantages in terms of speed, efficiency, and area utilization. Incorporating these diverse adder logics will enable the processor to handle arithmetic operations efficiently, supporting a range of computational tasks.

Ensure Correctness:

Verifying the processor design rigorously is essential for reliability. Extensive simulation and testing will assess all functionalities, including instruction behavior, data handling, and component interactions. This verification phase ensures early identification of potential issues, resulting in a robust and dependable processor design

IV. METHODOLOGY



This study outlines the development of a RISC processor design using optimized adder circuits, structured into nine distinct phases. First, a Literature Survey was conducted to assess current RISC processor designs and adder circuits like RCA, CLA, and CSA, forming the foundation for our design approach. Second, ISA Requirements were defined, outlining instruction formats, addressing modes, and essential operations, customized to support the integrated adders.

The RCA, CLA, and CSA Design phase focused on developing the adders, with RCA for simplicity, CLA for speed enhancement through carry prediction, and CSA for efficient multi-operand addition. In Processor Architecture Design, the ISA and adder designs were integrated into the processor's core components—ALU,

control unit, registers, and data paths. Model Integration followed, where the adders were connected to enable seamless data flow and arithmetic processing within the architecture.

The sixth phase involved Simulation and Debugging to test the processor's functionality and identify errors, ensuring alignment with design objectives. Functional Verification was then performed to confirm the processor's compliance with ISA specifications, verifying the accuracy of RCA, CLA, and CSA operations. During Performance Evaluation, metrics like speed and efficiency were measured to assess the processor's handling of arithmetic operations with different adders. Finally, in Discussion and Conclusion, results were summarized, addressing challenges, design success, and suggesting future enhancements for improved processor performance

V. RESULTS

The results section presents the outcomes of implementing the RISC processor design, Using Ripple carry adder, Carry Look Ahead Adder and Carry select Adder. This includs performance metrics and functionality validation. This analysis evaluates the processor's efficiency by highlighting Area , Time and Power consumed by respective adders

1. Schematic Results

.

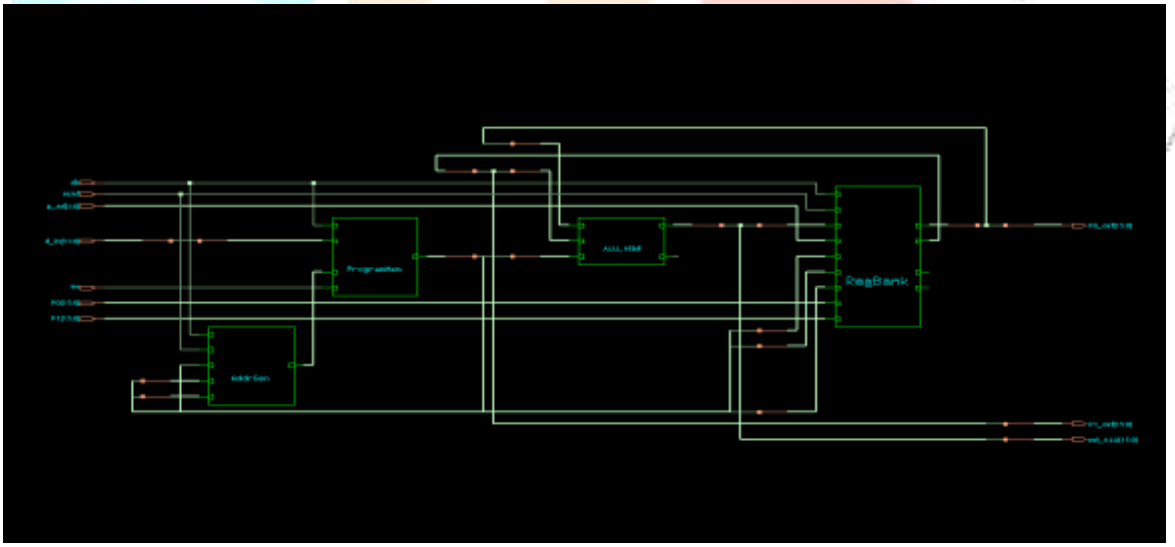


Fig1.1: Overall View

Fig 1.2 : Address Generator

The following image illustrates the address generator of the RISC processor, which was synthesized to optimize address computation and enhance overall processing efficency

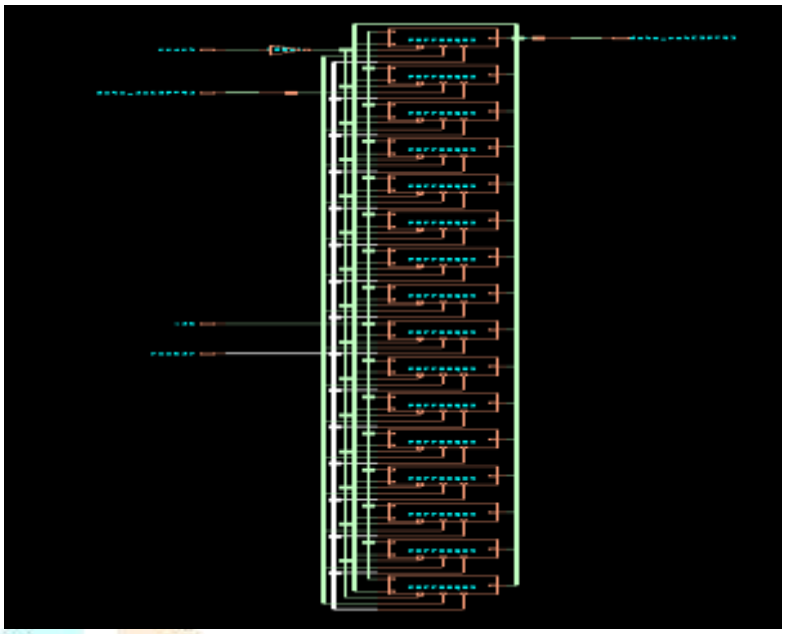


Fig 1.3 : Register Bank

The image depicts the register bank of the RISC processor, showcasing the synthesized design that facilitates efficient data storage and retrieval across the processor's execution stages

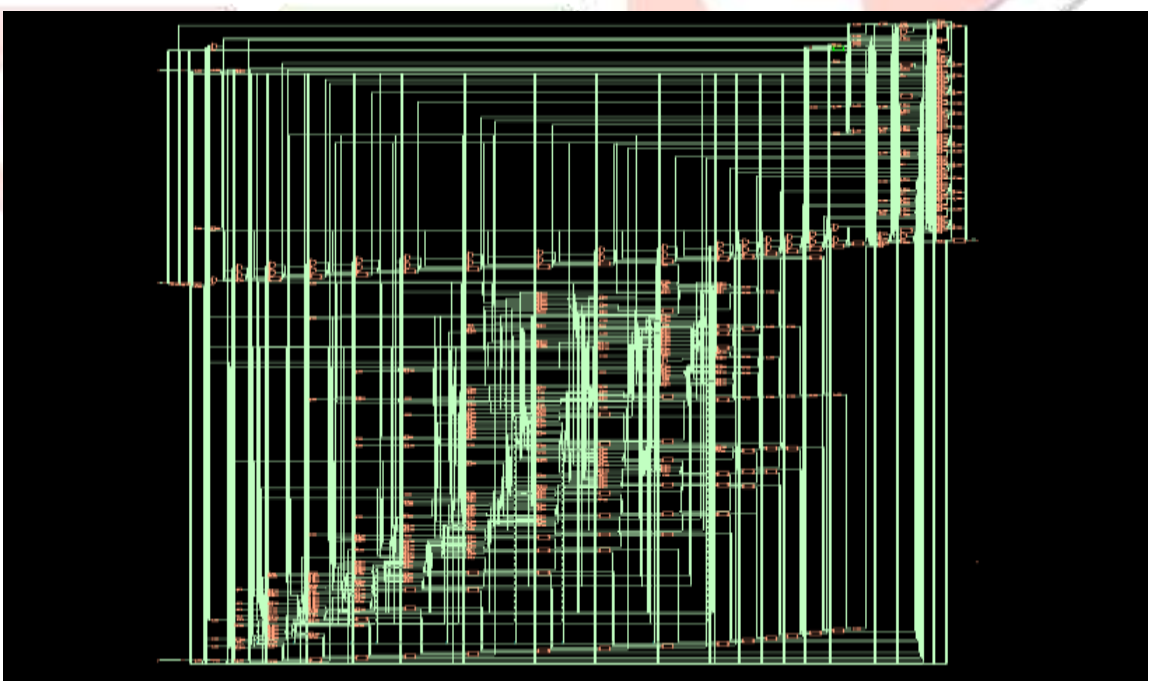


Fig 1.4 : Arthematic and Logical Unit

Figure shows the Arithmetic Logic Unit (ALU) of the RISC processor, illustrating its synthesized design that performs critical arithmetic and logical operations essential for Processing instructions

2. Simulation Results

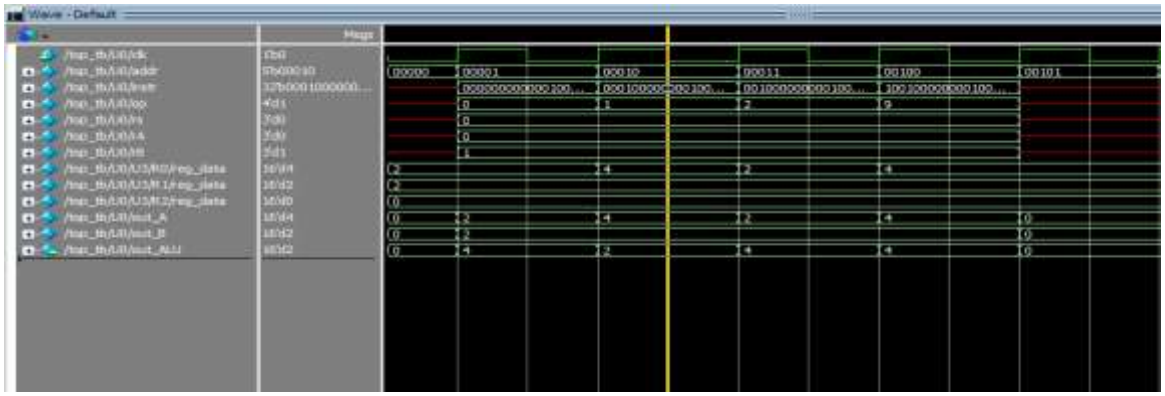
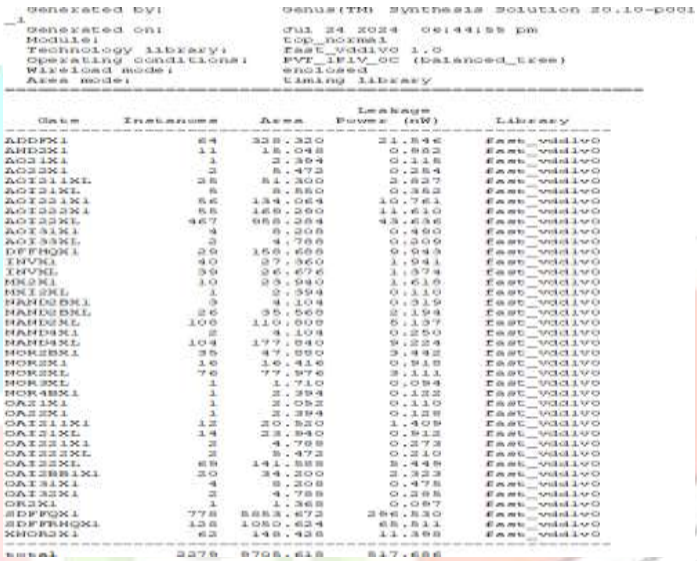


Fig 2.1 Simulation Result of Risc Processor

3. Area and Power Results



Gate	Instances	Area	Leakage Power (nW)	Library
ADDFX1	64	328.320	21.846	fast_vdd1v0
AND3X1	11	15.048	0.863	fast_vdd1v0
AO31X1	1	2.304	0.115	fast_vdd1v0
AO32X1	2	5.472	0.254	fast_vdd1v0
AOI21XL	25	51.300	2.827	fast_vdd1v0
AOI21XL	5	8.550	0.362	fast_vdd1v0
AOI221X1	56	134.064	10.761	fast_vdd1v0
AOI32X1	55	168.260	11.610	fast_vdd1v0
AOI32XL	467	868.284	43.636	fast_vdd1v0
AOI31X1	4	8.208	0.460	fast_vdd1v0
AOI33XL	2	4.788	0.309	fast_vdd1v0
DFFHQX1	29	158.688	9.943	fast_vdd1v0
INVX1	40	27.360	1.941	fast_vdd1v0
INVXL	39	26.676	1.374	fast_vdd1v0
MX2X1	10	23.940	1.616	fast_vdd1v0
MX2XL	1	2.394	0.110	fast_vdd1v0
NAND2BX1	3	4.104	0.319	fast_vdd1v0
NAND2BXL	26	35.568	2.194	fast_vdd1v0
NAND2XL	106	110.808	5.137	fast_vdd1v0
NAND3X1	2	4.104	0.250	fast_vdd1v0
NAND3XL	104	177.840	9.224	fast_vdd1v0
NOR2BX1	35	47.880	3.442	fast_vdd1v0
NOR2X1	16	16.416	0.918	fast_vdd1v0
NOR2XL	76	77.976	3.111	fast_vdd1v0
NOR3XL	1	1.710	0.094	fast_vdd1v0
NOR4BX1	1	2.394	0.122	fast_vdd1v0
OA21X1	1	2.052	0.110	fast_vdd1v0
OA22X1	1	2.394	0.128	fast_vdd1v0
OAI211X1	12	30.320	1.409	fast_vdd1v0
OAI21XL	14	23.940	0.913	fast_vdd1v0
OAI221X1	2	4.788	0.273	fast_vdd1v0
OAI222XL	2	5.472	0.210	fast_vdd1v0
OAI22XL	69	141.588	3.448	fast_vdd1v0
OAI2BB1X1	30	34.200	2.323	fast_vdd1v0
OAI31X1	4	8.208	0.475	fast_vdd1v0
OAI32X1	3	4.788	0.388	fast_vdd1v0
OR3X1	1	1.368	0.097	fast_vdd1v0
SDFFQX1	778	5853.672	286.830	fast_vdd1v0
SDFFRHQX1	138	1050.624	65.811	fast_vdd1v0
XNOR2X1	62	148.428	11.388	fast_vdd1v0
total	3279	9705.618	517.686	

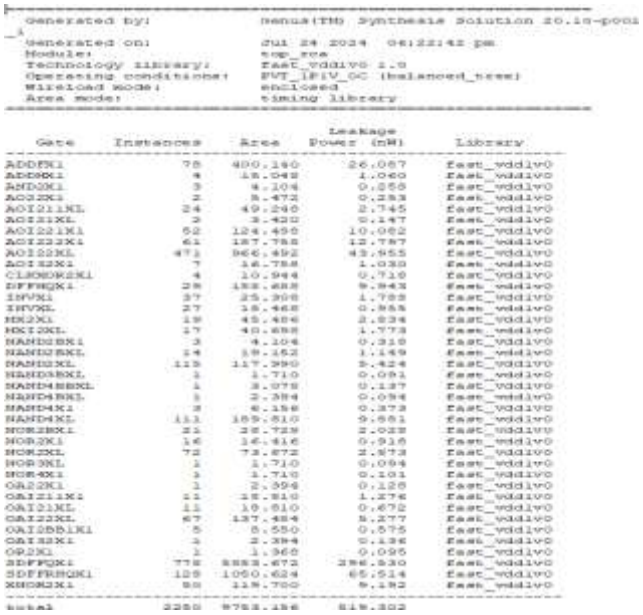
- **Risc**

Fig 3.1 Area and Power of Risc

Total Area: 9705.618 um²

Total Power (Leakage): 517.686 nW

- **RCA** (Ripple Carry Adder)



```
  Generated by:           Genus(TM) Synthesis Solution 20.10-p001
  Generated on:           Jul 24 2024  06:22:42 pm
  Module:                 top_rca
  Technology library:     fast_vddlv0 1.0
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
```

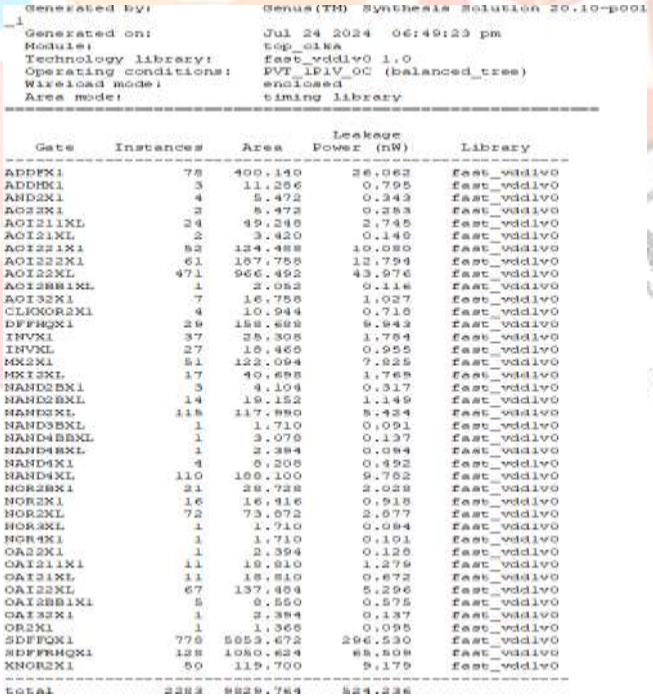
Gate	Instances	Area	Leakage Power (nW)	Library
ADDFX1	78	400.140	26.067	fast_vddlv0
ADDHX1	4	15.048	1.060	fast_vddlv0
AND2X1	3	4.104	0.255	fast_vddlv0
AO22X1	2	5.472	0.283	fast_vddlv0
AOI211XL	24	49.248	2.745	fast_vddlv0
AOI21XL	2	3.420	0.147	fast_vddlv0
AOI221X1	52	124.488	10.082	fast_vddlv0
AOI222X1	61	187.758	12.787	fast_vddlv0
AOI22XL	471	966.492	43.955	fast_vddlv0
AOI32X1	7	16.758	1.030	fast_vddlv0
CLKXOR2X1	4	10.944	0.710	fast_vddlv0
DFFHQX1	29	158.688	9.943	fast_vddlv0
INVX1	37	25.308	1.783	fast_vddlv0
INVXL	27	18.468	0.955	fast_vddlv0
MX2X1	19	45.486	2.834	fast_vddlv0
MXI2XL	17	40.698	1.773	fast_vddlv0
NAND2BX1	3	4.104	0.310	fast_vddlv0
NAND2BXL	14	19.152	1.149	fast_vddlv0
NAND3XL	115	117.990	5.424	fast_vddlv0
NAND3BXL	1	1.710	0.091	fast_vddlv0
NAND4BBXL	1	3.078	0.137	fast_vddlv0
NAND4BXL	1	2.394	0.094	fast_vddlv0
NAND4X1	4	9.234	0.492	fast_vddlv0
NAND4XL	110	188.100	9.762	fast_vddlv0
NOR2BX1	21	28.728	2.028	fast_vddlv0
NOR2X1	16	16.416	0.918	fast_vddlv0
NOR2XL	72	73.872	2.677	fast_vddlv0
NOR3XL	1	1.710	0.094	fast_vddlv0
NOR4X1	1	1.710	0.101	fast_vddlv0
OA22X1	1	2.394	0.126	fast_vddlv0
OAI211X1	11	18.810	1.276	fast_vddlv0
OAI21XL	11	10.610	0.672	fast_vddlv0
OAI22XL	67	137.484	5.277	fast_vddlv0
OAI2BB1X1	5	8.550	0.575	fast_vddlv0
OAI33X1	1	2.394	0.136	fast_vddlv0
OR2X1	1	1.368	0.095	fast_vddlv0
SDFFQX1	778	5883.672	286.530	fast_vddlv0
SDFFRHQX1	128	1050.624	65.514	fast_vddlv0
XNOR2X1	50	119.700	9.182	fast_vddlv0
total	2288	9753.156	519.302	

Fig 3.2 Area and Power of RCA

Total Area: 9753.156 um²

Total Power (Leakage): 519.302 nW

- **CLKA**(Carry Look Ahead Adder)



```
  Generated by:           Genus(TM) Synthesis Solution 20.10-p001
  Generated on:           Jul 24 2024  06:49:23 pm
  Module:                 top_clka
  Technology library:     fast_vddlv0 1.0
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
```

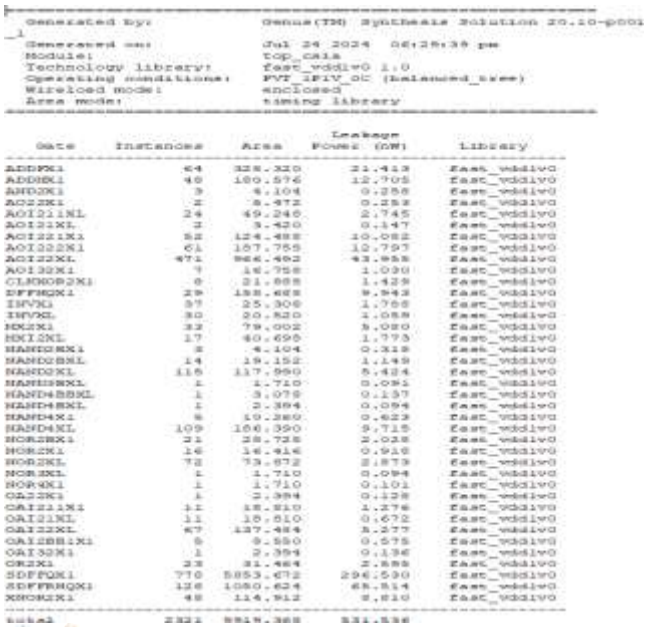
Gate	Instances	Area	Leakage Power (nW)	Library
ADDFX1	78	400.140	26.062	fast_vddlv0
ADDHX1	3	11.286	0.795	fast_vddlv0
AND2X1	4	5.472	0.340	fast_vddlv0
AO22X1	2	5.472	0.283	fast_vddlv0
AOI211XL	24	49.248	2.745	fast_vddlv0
AOI21XL	2	3.420	0.140	fast_vddlv0
AOI221X1	52	124.488	10.080	fast_vddlv0
AOI222X1	61	187.758	12.794	fast_vddlv0
AOI22XL	471	966.492	43.976	fast_vddlv0
AOI2BB1XL	1	2.052	0.116	fast_vddlv0
AOI32X1	7	16.758	1.027	fast_vddlv0
CLKXOR2X1	4	10.944	0.710	fast_vddlv0
DFFHQX1	29	158.688	9.943	fast_vddlv0
INVX1	37	25.308	1.784	fast_vddlv0
INVXL	27	18.468	0.955	fast_vddlv0
MX2X1	51	122.094	7.825	fast_vddlv0
MXI2XL	17	40.698	1.769	fast_vddlv0
NAND2BX1	3	4.104	0.317	fast_vddlv0
NAND2BXL	14	19.152	1.149	fast_vddlv0
NAND3XL	115	117.990	5.424	fast_vddlv0
NAND3BXL	1	1.710	0.091	fast_vddlv0
NAND4BBXL	1	3.078	0.137	fast_vddlv0
NAND4BXL	1	2.394	0.094	fast_vddlv0
NAND4X1	4	9.234	0.492	fast_vddlv0
NAND4XL	110	188.100	9.762	fast_vddlv0
NOR2BX1	21	28.728	2.028	fast_vddlv0
NOR2X1	16	16.416	0.918	fast_vddlv0
NOR2XL	72	73.872	2.677	fast_vddlv0
NOR3XL	1	1.710	0.094	fast_vddlv0
NOR4X1	1	1.710	0.101	fast_vddlv0
OA22X1	1	2.394	0.126	fast_vddlv0
OAI211X1	11	18.810	1.279	fast_vddlv0
OAI21XL	11	10.610	0.672	fast_vddlv0
OAI22XL	67	137.484	5.296	fast_vddlv0
OAI2BB1X1	5	8.550	0.575	fast_vddlv0
OAI33X1	1	2.394	0.137	fast_vddlv0
OR2X1	1	1.368	0.095	fast_vddlv0
SDFFQX1	778	5853.672	296.530	fast_vddlv0
SDFFRHQX1	128	1080.624	65.508	fast_vddlv0
XNOR2X1	50	119.700	9.179	fast_vddlv0
total	2283	9829.764	524.236	

Fig 3.3 : Area and Power of CLKA

Total Area: 9829.764 um²

Total Power (Leakage): 524.236 nW

- **CSLA** (Carry Select Adder)



Gate	Instances	Area	Leakage Power (nW)	Library
ADDFX1	64	328.320	21.413	fast_vdd1v0
ADDHX1	40	180.576	12.705	fast_vdd1v0
AND2X1	3	4.104	0.258	fast_vdd1v0
AO22X1	2	5.472	0.253	fast_vdd1v0
AOI211XL	24	49.248	2.745	fast_vdd1v0
AOI21XL	3	3.420	0.147	fast_vdd1v0
AOI221X1	52	124.488	10.082	fast_vdd1v0
AOI222X1	61	157.755	12.797	fast_vdd1v0
AOI22XL	471	986.463	43.855	fast_vdd1v0
AOI32X1	7	16.758	1.030	fast_vdd1v0
CLKXOR2X1	0	31.865	1.429	fast_vdd1v0
DFFHQX1	29	138.468	9.943	fast_vdd1v0
INVX1	57	25.308	1.766	fast_vdd1v0
INVXL	30	20.520	1.059	fast_vdd1v0
MX2X1	32	79.002	5.080	fast_vdd1v0
NAND2XL	17	40.698	1.775	fast_vdd1v0
NAND3BX1	2	4.104	0.318	fast_vdd1v0
NAND3BXL	14	19.152	1.149	fast_vdd1v0
NAND3XL	115	117.990	5.424	fast_vdd1v0
NAND4BBXL	1	1.710	0.091	fast_vdd1v0
NAND4BBXL	1	3.078	0.137	fast_vdd1v0
NAND4BXL	1	2.394	0.094	fast_vdd1v0
NAND4X1	6	10.260	0.623	fast_vdd1v0
NAND4XL	109	166.590	9.715	fast_vdd1v0
NOR2BX1	21	28.728	2.038	fast_vdd1v0
NOR2X1	16	16.416	0.910	fast_vdd1v0
NOR2XL	72	73.872	3.873	fast_vdd1v0
NOR3XL	1	1.710	0.094	fast_vdd1v0
NOR4X1	1	1.710	0.101	fast_vdd1v0
OA22X1	1	2.394	0.128	fast_vdd1v0
OAI211X1	11	18.810	1.276	fast_vdd1v0
OAI21XL	11	10.810	0.672	fast_vdd1v0
OAI22XL	67	137.484	5.277	fast_vdd1v0
OAI2BB1X1	5	9.550	0.575	fast_vdd1v0
OAI32X1	1	2.394	0.136	fast_vdd1v0
OR2X1	23	31.464	2.888	fast_vdd1v0
SDFFQX1	770	5853.672	296.530	fast_vdd1v0
SDFFRHQX1	126	1080.624	65.514	fast_vdd1v0
XNOR2X1	48	114.912	8.810	fast_vdd1v0
total	2321	9919.368	531.536	

Fig 3.4 Area and Power of CSLA

Total Area: 9919.368 um²

Total Power (Leakage): 531.536 nW

4. Delay Output

RISC : In the RISC processor design the critical path delay from the first register to the last register measures 1215 ps, reflecting the overall time required for data to propagate through the entire pipeline of the processor.

RCA : In the RISC processor design utilizing a Ripple Carry Adder, the critical path delay from the first register to the last register measures 1300 ps, reflecting the overall time required for data to propagate through the entire pipeline of the processor.

CLKA : In contrast, when implementing the Carry Select Adder with an optimized design, the delay from the first to the last register is reduced to 1290 ps, demonstrating an improvement in the data propagation time through the processor's pipeline

In the RISC processor design using a Carry Select Adder, the delay from the first to the last register is 1300 ps, indicating the time required for data to traverse the full pipeline of the processo

VI. CONCLUSION

This study evaluated the performance of an Ideal RISC Processor and three distinct adder architectures—Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA)—analyzing their impact on total area, power (leakage), and critical path delay. The Ideal RISC Processor emerged as the most compact and power-efficient, occupying the smallest area at 9705.618 µm² and exhibiting the lowest leakage power of 517.686 nW. Each adder type increased the processor's area slightly, with the CSA occupying the largest area at 9919.368 µm². Power consumption followed a similar trend, where the CSA exhibited the highest leakage at 531.536 nW, indicating a trade-off between performance and resource utilization across different adder architectures

offs. The Ideal RISC Processor achieved the shortest delay at 1215 ps, reflecting its efficient design without additional adder complexities. Among the adders, the CLKA proved the fastest, with a minimal delay of 1290 ps, whereas both the RCA and CSA had a delay of 1300 ps. This suggests that while the CLKA slightly increases the processor's area and power, it provides an optimal balance of speed and efficiency.

Final Conclusion

The corrected delay values underscore that the Ideal RISC Processor is the best choice for applications requiring minimal area, power, and delay, confirming its optimality in terms of size and efficiency. The CLA, with its reduced critical path delay, presents the most efficient adder choice, balancing speed with moderate area overhead. In contrast, the RCA and CSA show equal delay, but the CSA demands significantly more area and power, rendering it less favorable in contexts where compactness and low leakage are critical. This study concludes that while each adder architecture offers unique advantages, the Ideal RISC Processor provides the most balanced performance, with CLKA as a viable option for designs prioritizing delay over area.

.**Comparison Table**

Parameters	Area	Delay	Power
RISC Processor	9705.618 um^2	1215ps	517.686 nW
RCA(Ripple carry adder)	9753.156 um^2	1300ps	519.302 nW
CLKA(Carry Look Ahead Adder)	9829.764 um^2	1290ps	524.236 nW
CSLA (Carry Select Adder)	9919.368 um^2	1300ps	536

VII. ACKNOWLEDGEMNT

We would like to acknowledge thank everyone and Assistant Professors of Dr. Jeeru Dinesh Reddy Dept. of ECE BMS college of Engineering for his support in designing this project

REFERENCES

[1]. P. Trivedi and R. P. Tripathi, "Design & analysis of 16 bit RISC processor using low power pipelining," International Conference on Computing, Communication & Automation, Greater Noida, India, 2020, pp. 1294-1297, doi: 10.1109/CCAA.2015.7148575

\[2]. R. Bhat, D. Pandey, F. Ahmad and P. Gupta, "Analysis and Optimization of 16-bit RISC Processor and 32-bit MIPS Processor: A Review," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-6, doi: 10.1109/CONIT59222.2023.10205898.

[3]. C. Venkatesan, M. T. Sulthana, M. G. Sumithra and M. Suriya, "Design of a 16-Bit Harvard Structure RISC Processor in Cadence 45nm Technology," 2019 5th International Conference on Advanced Computing &

Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 173-178, doi: 10.1109/ICACCS.2019.8728479.

[4]. S. S. Khairullah, "Realization of a 16-bit MIPS RISC pipeline processor," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-6, doi: 10.1109/HORA55278.2022.9799944

[5]. 8A. P. Singh, A. Rai, A. Rajput, P. C. Joshi and A. Prakash, "Design and Analysis of High Speed RISC Processor Using Pipelining Technique," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 1642- 1645, doi: 10.1109/ICAC3N56670.2022.10074423

[6]. B. Ramkumar and H. M. Kittur, "Low-Power and Area-Efficient Carry Select Adder," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 2, pp. 371-375, Feb. 2012, doi: 10.1109/TVLSI.2010.2101621

[7]. Y. Xu et al., "Towards Developing High Performance RISC-V Processors Using Agile Methodology," 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, 2022, pp. 1178-1199, doi: 10.1109/MICRO56248.2022.0008

[8]. A. -S. Gheorghe and C. Burileanu, "Savage16 - 16-bit RISC architecture general purpose microprocessor," CAS 2010 Proceedings (International Semiconductor Conference), Sinaia, Romania, 2010, pp. 521-524, doi: 10.1109/SMICND.2010.5650480

[9]. N. A. Gudala, T. Ytterdal, J. J. Lee and M. Rizkalla, "Implementation of High Speed and Low Power Carry Select Adder with BEC," 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 2021, pp. 377-381, doi: 10.1109/MWSCAS47672.2021.9531750.

[10]. M. Johns and T. J. Kazmierski, "A Minimal RISC-V Vector Processor for Embedded Systems," 2020 Forum for Specification and Design Languages (FDL), Kiel, Germany, 2020, pp. 1-4, doi: 10.1109/FDL50818.2020.9232940.

[11]. Mamun B, Shabiul I. and Sulaiman S ,“A Single Clock Cycle MIPS RISC Processor Design using VHDL

[12]. J. C. Wright et al., "A Dual-Core RISC-V Vector Processor with On-Chip Fine-Grain Power Management in 28-nm FD-SOI," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 28, no. 12, pp. 2721-2725, Dec. 2020, doi: 10.1109/TVLSI.2020.3030243.

[13]. M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner and L. Benini, "Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor with Multiprecision Floating-Point Support in 22-nm FDSOI," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 28, no. 2, pp. 530- 543, Feb. 2020, doi: 10.1109/TVLSI.2019.2950087.

[14]. Á. B. de Oliveira et al., "Evaluating Soft Core RISC-V Processor in SRAM-Based FPGA Under Effects," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1503- 1510, July 2020, doi: 10.1109/TNS.2020.299572