iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68719

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

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 IV Apr 2025- Available at www.ijraset.com

Design and Analysis of 32-bit Interlock Collapsing ALU in VLSI

Dr. R. Sravanthi[1], T. BeaulahMery[2]
[1]*Associate Professor,* [2]*Student, Dept.Electronics & Ccmmunication Eng, PBR -VITS Kavali*
[1]*Sravanthi.r@visvodayata.ac.in*
[2]*tbeaulahsony97@gmail.com*

Abstract: ***A device capable of executing interlocked fixed point arithmetic logic unit (ALU) instructions in parallel with other instructions causing the execution interlock is presented. The device incorporates the design of a 3-1 ALU and can execute two's complement, unsigned binary, and binary logical operations. It is shown that status for ALU operations using a 3-1 ALU can be determined in a parallel fashion, resulting in the compliance of the proposed device with predetermined architectural behavior of single instruction execution. The device requires no more logic stages than does a 3-1 binary adder using a carry-save adder (CSA) followed by a carry-lookahead adder (CLA) design. They are executed in serial like any serial machine. It takes more than one cycle to execute multiple instructions causing performance degradation. The solution requires a special kind of device called "Interlock collapsing ALU". The Interlock Collapsing ALU, unlike conventional 2-1 ALU's is a 3- 1 ALU. The proposed device executes the interlocked instructions in a single instruction cycle, unlike other parallel machines, resulting in high performance. The resulting implementation demonstrates that the proposed 3-1 Interlock Collapsing ALU can be designed to outperform existing schemes for ICALU, by a factor of at least two. The ICALU is implemented in VHDL. Its functionality is verified through simulation.***

Keywords: ALU, Interlock collapsing, ICALU, parallel processing, computer, architecture, parallel machines Introduction.

I. INTRODUCTION

An important area in computer architecture is parallel processing. Machines employing parallel processing are called parallel machines. A parallel machine executes multiple instructions in one cycle. However, parallel machines have a limitation, they cannot execute interlocked instructions. Parallel machines cannot execute interlocked instruction concurrently. Interlocked instructions or instruction with dependencies cannot be executed concurrently in a parallel machine, thus degrading the performance of the machine[1]. The thesis investigates a solution, called, "interlock collapsing", to execute these interlocks concurrently. The solution requires a special kind of a device called the Interlock collapsing ALU. The Interlock collapsing ALU, unlike conventional 2-1 ALU's, is a 3-1 ALU[2][3]. The proposed ALU, in addition to collapsing these interlocks also should be implemented in identical stages as the conventional ALU's. A functional model of the ICALU is assumed initially. The functional model is optimized by optimizing the model's individual blocks. The design and optimization of each block is discussed in separate chapters Maintaining the Integrity of the Specifications. Finally, two parallel machines with and without the ICALU are compared with regard to their execution times. The effect of variation of percentage interlocks in a given code on the execution times and the percentage speed ratio of the parallel machines is studied [4]. The ICALU is implemented in VHDL. Its functionality is verified through simulation.

II. LITERATURE REVIEW

High-performance 3-1 interlock collapsing ALU[5]. An ALU that allows the execution of most execution interlocks in a single machine cycle, is presented. We focus on reducing the Boolean equations describing the device and the incorporation of new mechanisms in the interlock collapsing ALU design. In particular, we focus on the reduction of the critical path, regarding delay, for the interlock collapsing ALU implementation. It is shown that the delay associated with the implementation of the proposed device, in terms of logic stages, assuming a commonly available CMOS technology, is equivalent to the number of logic stages required for the implementation of a 3-1 binary adder.

Interlocked fixed point arithmetic logic unit in parallel[6]. A device capable of executing interlocked fixed point arithmetic logic unit 4 (ALU) instructions in parallel with other instructions causing the execution interlock is presented. The device incorporates the design of a 3-1 ALU and can execute two's complement, unsigned binary, and binary logical operations.

It is shown that status for ALU operations using a 3-1 ALU can be determined in a parallel fashion, resulting in the compliance of the proposed device with predetermined architectural behaviour of single instruction execution.
The device requires no more logic stages than does a 3-1 binary adder using a carry-save adder (CSA) followed by a carry-lookahead adder (CLA) design. Design considerations using a commonly available CMOS technology are also reported, indicating that the device will not increase the machine cycle of an implementation.

III. PROPOSED MODEL

An important area in computer architecture is parallel processing. Machines employing parallel processing are called parallel machines. A parallel machine executes multiple instructions in one cycle. However, parallel machines have a limitation, they cannot execute interlocked instructions. They are executed in serial like any serial machine. It takes more than one cycle to execute multiple instructions causing performance degradation. In addition, there is hardware underutilization as a result of serial execution in parallel machine. The solution requires a special kind of device called “Interlock collapsing ALU”. The Interlock Collapsing ALU, unlike conventional 2-1 ALU’s is a 3- 1 ALU[2][3]. The proposed device executes the interlocked instructions in a single instruction cycle, unlike other parallel machines, resulting in high performance. The resulting implementation demonstrates that the proposed 3-1 Interlock Collapsing ALU can be designed to outperform existing schemes for ICALU, by a factor of at least two. The ICALU is implemented in VHDL. Its functionality is verified through simulation.
In this chapter all the designed components are put together to implement the ICALU. Also, ALU1 is created using the designed components. Finally, the Interlock collapsing ALU unit is implemented which consists of both ALU1 and ICALU. The chapter also estimates the relative delay.

A. Reduced ICALU Model

Resulting from the design of the various stages in the preceding chapters a reduced ICALU is obtained. The result was the elimination of the multiplexers M2 and M3 and also better implementations of the Pre and Post-CLA Logic Blocks.

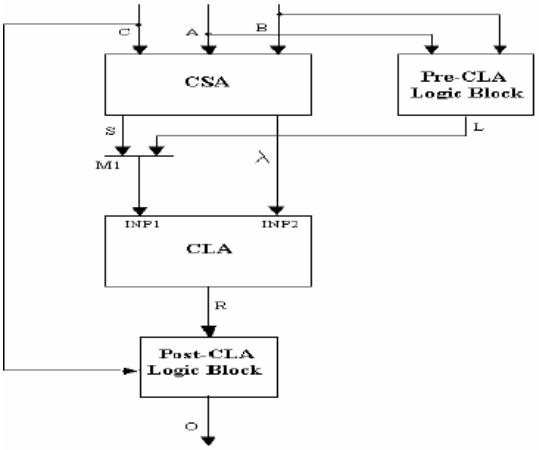


Figure 1: Reduced Dataflow Model of Icalu

B. Alu1 Model

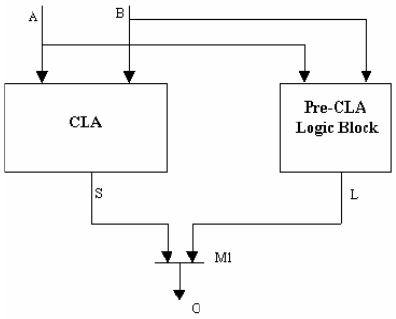


Figure 2: Dataflow Model of ALU

IV. DESIGN ISSUES OF ICALU

The control signals for multiplexer are K_{12} and K_{13} and are set as follows :

I) CATEGORY 1 (ARITHMETIC) :

$K_{12} = 1$ and, $K_{13} = 0$;

Output of ALU1 = O = A ± B.

II) CATEGORY 2 (LOGICAL) :

$K_{12} = 0$ and, $K_{13} = 1$;

Output of ALU1 = O = A LOP B.

The values of control signals are summarized.

A. Implementation :

The ALU1 is implemented using the block diagram above. The components CLA and PREBLK are the adder and the logic block respectively, for ALU1. The program forentity ALU1.

B. Interlock Collapsing ALU Unit :

The Interlock collapsing ALU unit consists of ALU1 and the ICALU operating in parallel. The block diagram of the Interlock collapsing unit was shown in Chapter 1, Fig 1. The program for entity ICUNIT.

C. Estimation Of Relative Delay Between ALU And ICALU :

In this section the relative delay between the ALU1 in Fig 2 and the ICALU in Fig 1 is estimated. The relative delay is the difference between the delay of ALU1 and the ICALU. The delay is required to find out the instruction cycle length. The delay of a device can be estimated by taking a logic gate count from the input to the output[7]. Only the delay between both ALU's considered because all other stages in their respective paths are identical, hence they also have identical delays.

Now, compare Fig 1 (ICALU) and Fig 2 (ALU1).

By elimination, it is deduced that the ICALU has two additional stages when compared to the ALU1 which are :

i) The CSA and,

ii) The Post-CLA Logic Block.

The procedure is :

1) The CLA and multiplexers are common to both theALU's. Hence they can be eliminated.
2) The extra stages in the ICALU path are the CSAand the Post-CLA Logic Stage.

The Pre-CLA Logic stages are not considered because in case of ALU1 it is parallel with the CLA stage and has lesser stages than the same. Figure 3 represents the instruction path of serial machine. instructions given as I_0, or the basic instruction cycle time.

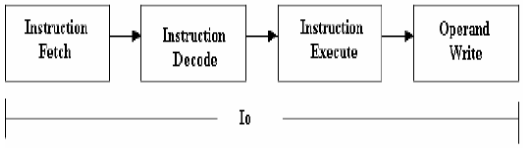


Figure 3: Phases of Instruction execution process

The time to execute an The individual stage have

D. Without ICALU

For a parallel machine there are two such paths in parallel. Fig 4 shows instruction execution Where as, in case of the ICALU it is in parallel and has the same delay as the CSA. The logic delay of both stages are :

I) Carry Save Adder(CSA)

To estimate this consider (3.13a) and (3.14) which represent the input-output transformations of the CSA sum and carry respectively. Both are in parallel.

$$SUM = S_i = A_i \ V \ B_i \ V \ C_i \quad (1)$$

$$\lambda_{i+1} = K_2 \ A_i \ B_i + K_1 \ B_i \ C_i + K_1 \ A_i \ C_i + K_3 \ C_{i+1}. \quad (2)$$

(1) and (2) can each be implemented in one gate delay using custom-built CMOS libraries. A ± 3 X 4 AO gate can serve this purpose ('+' represents AND-OR and '-' represents AND-OR-INVERT). The delay of this gate is assumed to be 1 gate stage as that of any other gate in the assumed libraries.

II) POST-CLA Logic Block

Similarly, equation 3 can be implemented in one gatedelay by the AO gate.

Li = Lli KPRE1 + Lri KPRE1 + Lli Lri KPRE2 + Lli LriKPRE3 (3)

Thus, total relative gate delay of the ICALU over the ALU1 =

Logic delay due to CSA stage + Logic delay due to Post-CLALogic Stage =1+1 = 2.

III) Carry Look Ahead Adder (CLA)

The CLA is faster than the ripple carry adder. The carry input to each stage is generated directly, instead of allowing the carry to ripple from one stage to another. Figure 5.3 shows the block diagram of a CLA[8]. The Boolean expression for each carry block can be defined by using the carryout expression of a full adder. It is given as

$C_{i+1} = x_i \ y_i + C_i \ (x_i + y_i)$(4)

Where, i=0,……..,N, and, N= number of bits in each number. For example, the output of first carry block:

$C_1 = x_0 \ y_0 + C_0 \ (x_0 + y_0)$ (5)

One of the most basic adders is the ripple carry adders. The addition is similar to that of paper and pencil addition. In figure 4 it will shows the 4-bit Carry Look Ahead Adder(CLA).

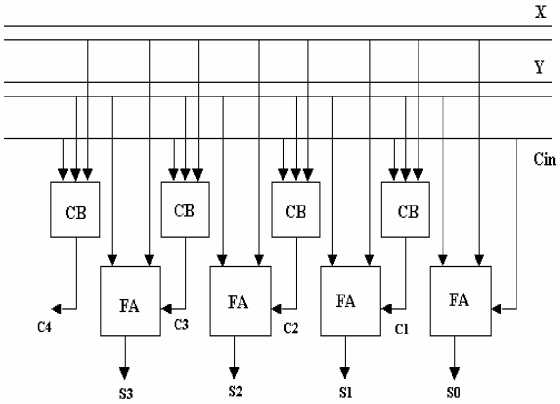


Figure 4: A 4-bit Carry Look Ahead Adder

IV) PRE – CLA Logic Block

The Pre-CLA Logic Block has to perform AND, OR, XOR & their inverts NAND, NOR, XNOR respectively. It also provides inputs to INPUT 1 of CLA for categories 2 to 4. With these signals the expression for the output of the Pre-CLA Logic Block 'L' is expressed as:

Li = Ai Bi KAND KINV + Ai KOR KINV + Bi KOR KINV + Ai Bi KXOR KINV + Ai Bi KXOR KINV + Ai KAND KINV + Bi KAND2 KINV2 + Ai Bi KOR KINV + Ai Bi KXOR KINV + Ai Bi KXOR KINV (4)

where, ($0 \leq i \leq 31$)

In the equation 4, the Logic that can be used for the design to developed the Pre-CLA logic block. This equation is modified from the virginal expression can be used to build the logic The operation is the same as that for the parallel machine when the sequence is non-interlocked

Table 1: Control Signals to PRE-CLA Logic Block

Control Signal	Description
K_{AND}	AND Inputs A & B
K_{OR}	OR Inputs A & B
K_{XOR}	XOR Inputs A & B
K_{INV}	Inverts above operations.

V. DESCRIPTION AND WORKING

The ICALU is basically a 3-1 ALU. It has 3-1 CSA (Carry Save Adder) in addition to the 2-1 CLA to achieve the desired 3-1 arithmetic operation. The ICALU also has an extra logic when compared to the 2-1 ALU. The ICALU is implemented in the parallel machine by replacing ALU2 with the ICALU[8]. The operation of the parallel machine employing the ICALU is explained as

A. Non-Interlocked

The operation is the same as that for the parallel machine when the sequence is non-interlocked.

ADD R1, R2; [R1] [R1] ← [R2]
ADD R4, R3; [R4] [R4] ← [R3]

Instruction 2 required the result of instruction 1 (stored in R2). Instruction 2 can be executed only after instruction 1 has been executed. Thus, instruction 2 is said to be dependent on instruction 1. The dependency prevents the simultaneous execution of the instructions. of a non-interlocked instruction pair. Instruction 2 does not require that instruction 1 be executed before it (instruction 2) is executed. Thus, they can be executed simultaneously. Before moving on to the ICALU, consider the data flow of a parallel machine, which can execute two instructions concurrently.

B. Interlocked

Consider the interlocked sequence of expression 6 and 7. ALU1 executes the first instruction as usual. The ICALU collapses the two instructions into a single 3-operand instruction as shown in expression 6.

ADD R2, R1; [R2] ← [R2] + [R1] (6)
ADD R3, R2; [R3] ← [R3] + [R2] (7)

Instructions are said to be interlocked if an instruction depends on a previous instruction for its data so that they cannot be executed simultaneously.

ADD R3, R2, R1; [R3] ← [R3] + [R2] + [R1] (8)

Multi-operand execution units are required, since there are two interlocked sequential instructions. The first instruction is executed by a traditional ALU. Since the second instruction may be dependent on the first, the second ALU must be capable of performing the collapsed instruction of both the instructions, in parallel to the first ALU.

VI. RESULT

The figure shown in the following pages depict the result of the RTL schematics and the various categories of interlocked instructions simulation waves.

A. RTL Schematic

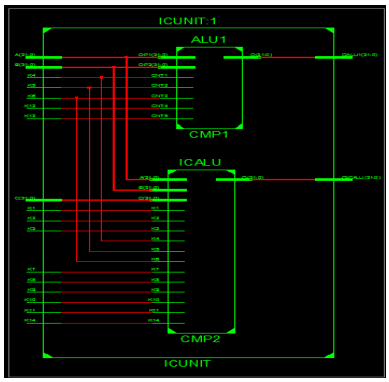


Figure 5: RTL Schematic for ICUNIT

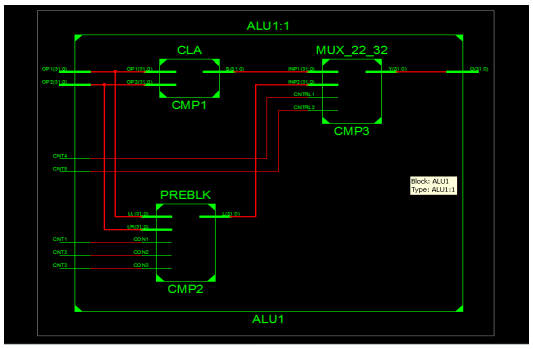


Figure 6: RTL Schematic of ALU

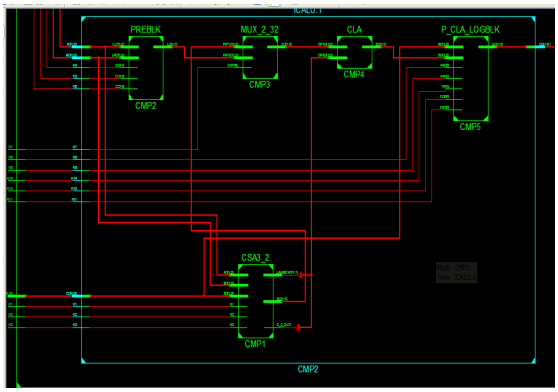


Figure 7: RTL Schematic of ICALU

B) Simulation Results

In simulation result A, B and C represents the three inputs and K1, K2, K3, ..., K14 represents the different control signals. Their valves are shown in each cycle. OALU and OICALU are the outputs of ALU and ICALU respectively. in this OALU performs operation on A and B, whereas OICALU perform operation on the three operands A, B and C.

I) Arithmetic Followed by Arithmetic Operation

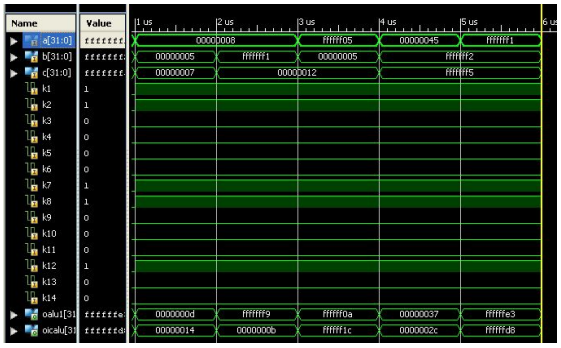


Figure 8: Arithmetic followed by Arithmetic Operation

II) Arithmetic Followed by Logical Operation

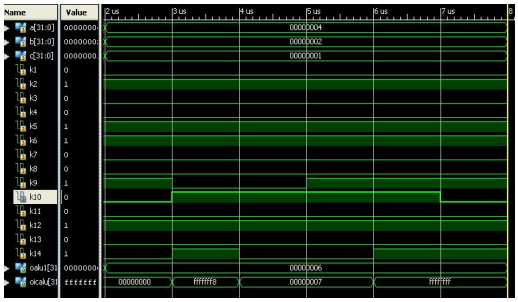


Figure 9: Arithmetic followed by Logical Operation

III) Logical Followed by Arithmetic Oeration

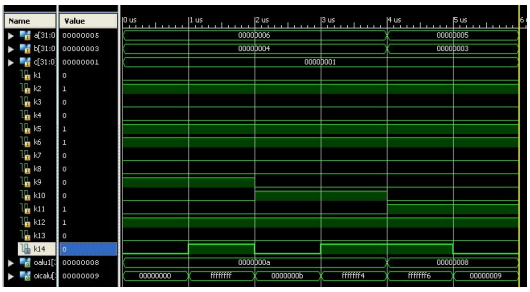


Figure 10: Logical followed by Arithmetic Operation

IV) Logical Followed by Logical Oeration

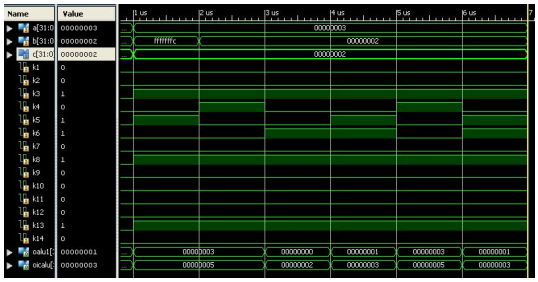


Figure 10: Logical followed by Logical Operation

C) Comparison of Execution Delay

Table 2: comparing table for outputs

COMPARISION	OALU	OICALU
Delay	**12.824ns**	**9.730ns**

VII. CONCLUSION AND FUTURE SCOPE

The objective of the thesis, execution of interlocked instructions in one instruction cycle. This was achieved by ICALU successfully designed and implemented using VHDL in Xilinx. Its functionally was verified through simulation. The ICALU can be implemented in just 3 logic delays more than of a conventional 2-1 ALU. The performance of an ordinary non-ICALU i.e., parallel machine and machine with the ICALU incorporated in it, was compared.

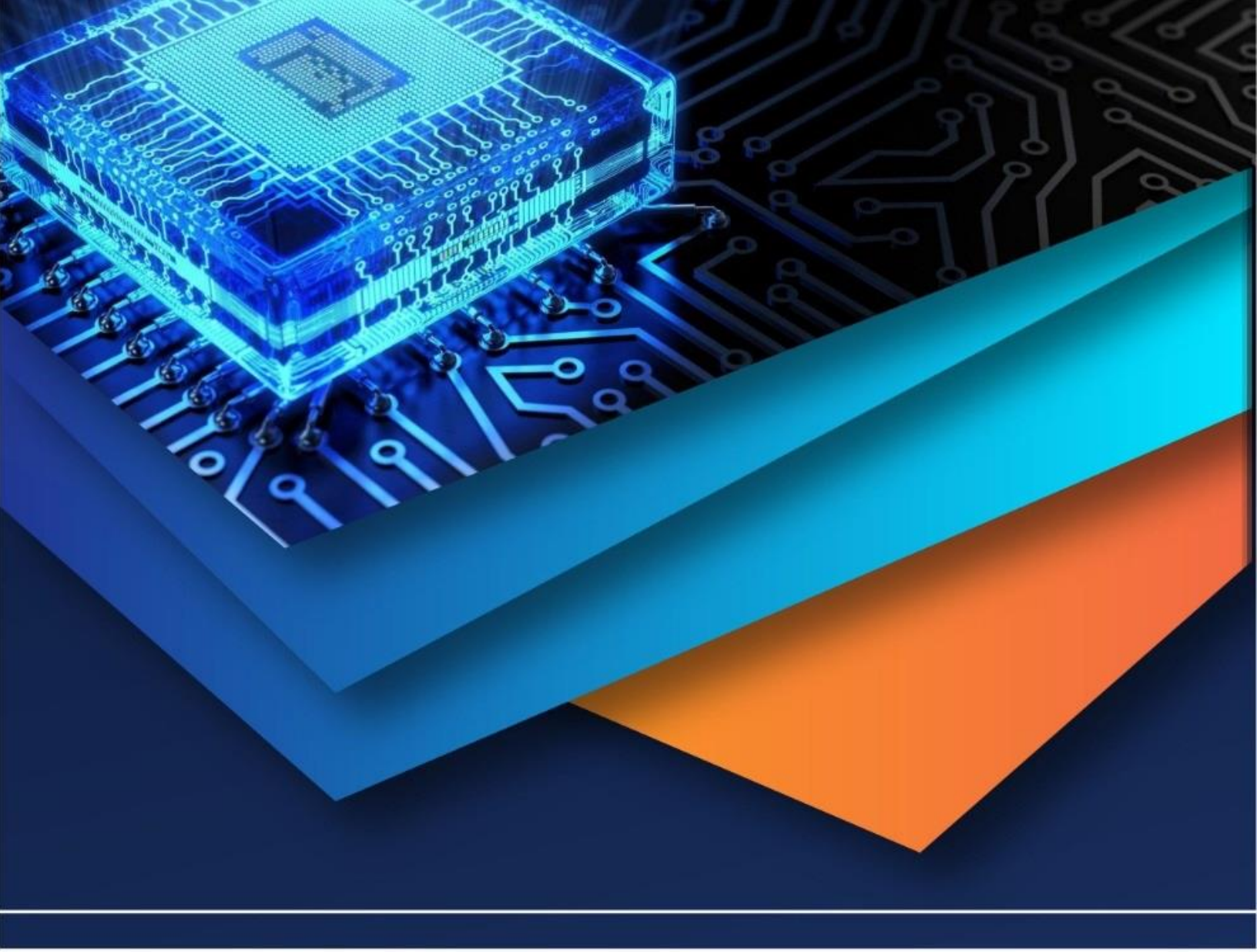
This work has been designed for 32-bit word size and results is evaluated for parameters like delay. This work can be further extended for higher number of bits. New architecture can be designed in order to reduce the delay of the circuit. Steps may be taken to optimize the other parameters like area, power, frequency, number of gate clocks, etc.

REFERENCES

[1] P. M. Kogge, The Architecture of Pipelined Computers, New York: McGraw-Hill, 1981.
[2] S. Vassiliadis, J. Phillips and B. Blaner, ICU design considerations, pp. 22, Oct. 1991.
[3] S. Vassiliadis and J. Phillips, Interlock collapsing SCISM ALU design, pp. 31, Oct. 1991. R. M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units", IBM J. Res. Developp., pp. 25-33, Jan. 1967.
[4] R. D. Acosta, J. Kjelstrup and H. C. Torng, "An instruction issuing approach to enhancing performance in multiple functional unit processors", IEEE Trans.Comput., vol. C-35, pp. 815-828, Sept. 1986.
[5] H. S. Warren, "Instruction scheduling for the IBM RISC system/6000 processor", IBM J. Res. Develop., vol. 34, no. 1, pp. 85-92, Jan. 1990.
[6] N. F. Jouppi, "The nonuniform distribution of instruction-level and machine parallelism and its effect on performance", IEEE Trans. Comput., vol. 38, no. 12, pp. 1645-1658, Dec. 1989.
[7] W. A. Wulf, "The WM computer architecture", Comput. Architecture News, vol. 16, no. 1, pp. 70-84, Mar. 1988.
[8] N. Malik, R. J. Eickenmeyer and S. Vassiliadis, "Interlock collapsing ALU for increased instruction-level parallelism", Conf. Proc. MICRO 25, pp. 149-157, 1992-Dec.

Dr. R. Sravanthi, Ph.D. Associate professor in the department of Electronics and communication Engineering in PBR-Visvodaya Institute of Technology and science, Kavali-A. P:524201, pbrvits.hodece@visvodyata.ac.in

T. BeaulahMery presently pursuing M.Tech -VLSID IV-Sem in department of Electronics and communication Engineering in PBR-Visvodaya Institute of Technology and science Kavali-A.P:524201tbeaulahsony97@gmail.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY