Juni Khyat ISSN: 2278-4632
(UGC Care Group I Listed Journal) Vol-10 Issue-5 No. 3 May 2020

IMPLEMENTATION OF 32-BIT INTERLOCK COLLAPSING ALU

P. Nagaraju,
Assistant Professor, Department of ECE, Narayana Engineering College, Gudur, AP, 524101
Email :pnaga5151@gmail.com,

P.Susmitha, M.Pavani, M.Sreekanth, B.Bharath Kumar
UG Student, Department of ECE, Narayana Engineering College, Gudur, AP, 524101
psusmithareddysr@gmail.com

Abstract: 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 serial instructions. They are executed in serial like any serial machine. It takes more than one cycle to execute multiple instructions causing performance degradation of machine. In addition there is hardware underutilization as a result of serial execution in parallel machine. This limitation is over come to implemented a special kind of device called "Interlock collapsing ALU". The Interlock Collapsing ALU, unlike conventional 2-1ALU's is a 3-1 ALU. The implemented device executes the interlocked instructions in a single instruction cycle, unlike other parallel machines, resulting in high performance. The resulting implementation demonstrates that the implemented 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 observed in simulation.

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

1. INTRODUCTION

BAKGROUND

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. 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.

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.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. The ICALU is implemented in VHDL. Its functionality is verified through simulation.

2. LITERATURE REVIEW

This chapter reviews some references from previous projects, journals, articles and books. All this information were collected from different sources such as internet, products, manuals etc. The information gathered in this chapter is related to the background study of this project.

1. J. Phillips, S. Vassiliadis, "High-Performance 3-1 Interlock Collapsing ALU's," IEEE Transactions on Computers, vol. 43, no. 3, pp. 257-268, Mar., 1994

A high-performance 3-1 interlock collapsing ALU, i.e., 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. The resulting implementation demonstrates that the proposed 3-1 interlock collapsing ALU can be

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

designed to outperform existing schemes for interlock collapsing ALU's by a factor of at least two. Finally, it is suggested that the proposed device can be used in the implementation of multiple instruction issuing machines, allowing the issuance and execution of interlocks in parallel and in a single machine cycle with no cycle time increases.

2. S. Vassiliadis, J. Phillips, B. Blaner, "Interlock Collapsing ALUs", IEEE Transactions on Computers.

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 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. It is suggested that the device can maintain full architectural compatibility.

3. R. D. Acosta, J. Kjelstrup, H. C. Torng, "An instruction issuing approach to enhancing performance in multiple functional unit processors", IEEE Trans. Comput., vol. 35, pp. 815-828, Sept. 1986.

Processors with multiple functional units, such as CRAY-1, Cyber 205, and FPS 164, have been used for high-end scientific computation tasks. Much effort has been put into increasing the throughput of such systems. One critical consideration in their design is the identification and implementation of a suitable instruction issuing scheme. Existing approaches do not issue enough instructions per machine cycle to fully utilize the functional units and realize the high-performance level achievable with these powerful execution resources.

4. N. P. 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.

A methodology for quickly estimating machine performance is developed. A first-order estimate is based on the average degree of machine parallelism. A second-order model corrects for the effects of nonuniformities in instruction-level and machine parallelism and is shown to be accurate to within 15% for three widely different machine pipelines: the CRAY-1, the MultiTitan, and a dual-issue superscalar machine.

5. N. Malik, R. J. Eickemeyer, S. Vassiliadis, "Instruction-level parallelism for execution interlock collapsing", Comput. Architecture News, vol. 20, no. 4, pp. 38-43, Sept. 1992.

An innovation technique has been developed that permits the collapsing of execution interlocks between integer ALU operations as well as between address generation operations, allowing parallel execution of two instructions, having true dependencies, in a single cycle. Given that the proposed scheme has been shown not to increase the machine cycle time, it potentially provides an attractive means for increasing the instruction—level parallelism. Preliminary results show that within the basic blocks, the geometric mean of the speedup from this new design technique is up to 10% in the integer SPEC Benchmarks. The geometric mean of the speedup including floating point benchmarks is up to 6%. The results also suggest that depending on the application environment this new design may be used as an alternative to the relatively more expensive out of order instruction issue approach.

3. IMPLEMENTATION OF 2-1 ICALU

3.1 The Interlock Collapsing ALU Unit

In this chapter all the designed components are puttogether to implement the ICALU. Also, ALU1 is createdusing the designed components. Finally, the Interlockcollapsing ALU unit is implemented which consists of bothALU1 and ICALU. The chapter also estimates the relative delay.

3.2 Implementation of 3-1 ICALU

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. The block diagram is shown in Fig1.

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

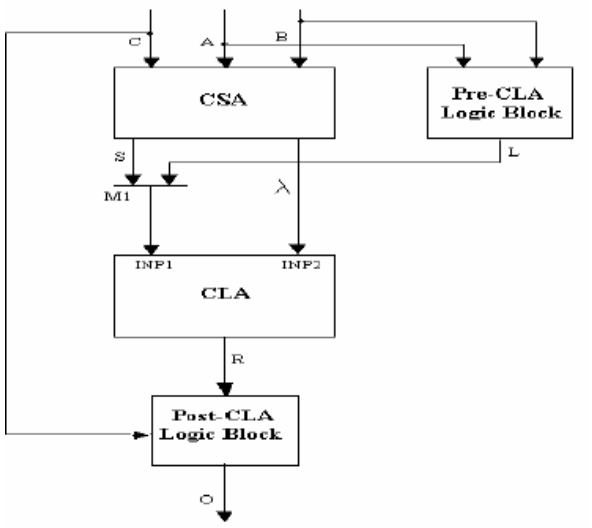

Fig1: Implementation of 3-1 I Calu

3.3 ALU1 model

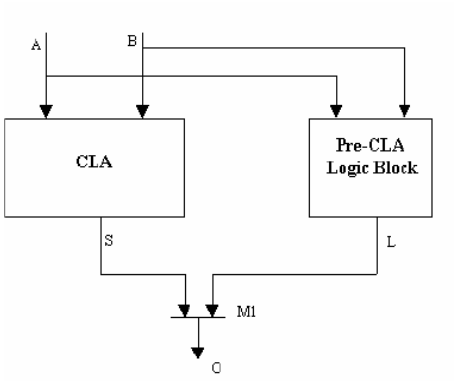

Fig2: 3-1 Model of ALU1

The control signals for multiplexer are K12 and K13 and are set as follows :
I) CATEGORY 1 (ARITHMETIC) :
K12 = 1 and, K13 = 0 ;
Output of ALU1 = O = A ± B.
II) CATEGORY 2 (LOGICAL) :
K12 = 0 and, K13 = 1 ;
Output of ALU1 = O = A LOP

The values of control signals are summarized in Table 3 :

Table 1 : Output table for ALU

CATEGORY	K12	K13	O
1	1	0	A ± B
2	0	1	A LOP B

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

3.3.1 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.

3.4 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.

3.5 Estimation Of Relative Delay Between ALU1 And ICALU

In this section the relative delay between the ALU1 in Fig and the ICALU in Fig isestimated. 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. 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 Fig1 (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 the ALU's. Hence they can be eliminated.
2) The extra stages in the ICALU path are the CSA and the Post-CLA Logic Stage.
3) 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.

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) CSA :
To estimate this consider 3.13 and 3.14 which represent the input-output transformations of the CSA sum and carry respectively. Both are in parallel.
SUM = Si = Ai V Bi V Ci, 3.13
λi+1 = K2 Ai Bi + K1 Bi Ci + K1 Ai Ci + K3 Ci+1. 3.14
3.13 and 3.14 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) LOGIC DELAY OF POST-CLA LOGIC BLOCK :
Similarly, (3.9) (shown below) can be implemented in one gate delay by the AO gate.
Li = Lli KPRE1 + Lri KPRE1 + LliLri KPRE2 + LliLri KPRE3 (3.9)
Thus total relative gate delay of the ICALU over the ALU1 = Logic delay due to CSA stage + Logic delay due to Post-CLA
Logic Stage =1+1 = 2.

3.6 Determination of Instruction Cycle Lengths of a Machine With And Without ICALU

The average instruction length is calculated to find out the speed of the machine. The instruction cycle length varies for each instruction. Hence an average instruction length has to be calculated. It is sufficient to take the average of only frequently executed instructions. The following discussion shows how the instruction lengths can be calculated for a given instruction.

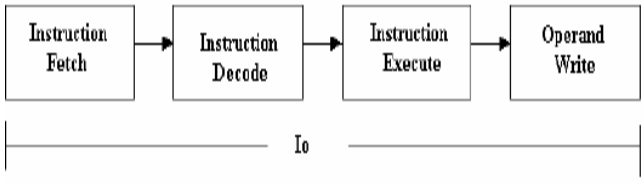

Fig 3.6: Phases of Instruction execution process

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

Fig 3.6 represents the instruction path of serial machine. instructions given as I0, or the basic instruction cycle time been discussed in Chapter 1.
The time to execute an individual stage have

3.6.1 Without ICALU

For a parallel machine there are two such paths in parallel. Fig 4.4 shows instruction execution (considering non-interlocked case) in a parallel machine with respect to time.
Fig 3.6 shows the instruction cycle of a parallel machine for a two-operand instruction pair shown below. The upper cycle in the figure represents execution of instruction 1. The instruction time is the same as the basic instruction cycle time, I0. Execution of Instruction 2 is shown in the lower half. It starts a memory write cycle after the first instruction, because memory cannot be accessed simultaneously. It shifts to the right by 1MW. The x's in figure represents an idle cycle.
ADD R1, R2 / Executed by ALU1 /
ADD R3, R4 / Executed by ALU2 /
The ID2 is smaller than ID1 by one memory access because we already have R2, fetched by Instruction 1. This compensates for the delay in start of execution of Instruction 2 and thus the execution cycles of both the instructions start at the same time. After the EX cycle is complete, Instruction 2 has to wait for 1MW for Instruction 1 to complete its memory access.
Instruction 2 takes a further 1MW to complete its cycle. Thus from the figure it can clearly be seen that the instruction time of a parallel machine is lengthened by 1MW.

3.6.2 With ICALU

The instruction cycle in figure is for the pair given below :
ADD R1, R2
ADD R1, R3
The operation is almost similar to that of an ordinary parallel machine except that there is no memory access for ALU1. Hence the memory access starts once the ICALU completes it's execution which is two additional logic or gate delays more than the 2-1ALU. Hence its instruction cycle time increases to I0 + 2 D (D– Unit gate delay or the delay of one gate). MW can be treated as three gate delays for CMOS memories. Substituting this value average instruction length can be calculated.

4. PERFORMANCE ANALYSIS PERFORMANCE ANLAYSIS

In this chapter the performance of a Non-ICALU and that of a parallel machine with the ICALU is compared. Table 5.1 shows the average instruction lengths of a machine with ICALU and a Non-ICALU parallel machine for the interlocked and Non-interlocked categories. The average instruction lengths were calculated by taking the average of instructions lengths obtained for all possible interlocked and non-interlocked pairs (See Appendix B). The average instruction length is the time taken to execute an instruction pair, that is two consecutive instructions.

Table 2 : Average Instruction Lengths for machines with and without ICALU

CATEGORY	AVERAGE INSTRUCTION LENGTH (NON – ICALU)	AVERAGE INSTRUCTION LENGTH (WITH ICALU)
Non–interlocked	$I_{PAVE1} = I_0 + 3.5$	**$I_{ICAVE1} = I_0 + 4.17$**
Interlocked	**$I_{PAVE1} = 2I_0$**	**$I_{ICAVE2} = I_0 + 2.63$**

Using the values in the table, the total execution time for eachmachine can be calculated, for a given number of instructions.

1) Comparison Of Total Execution Time

The total execution time of a parallel machine is given as :

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

TNI NNI + TI NI
Where,
TNI = Time taken to execute a Non-Interlocked pair.
NNI = Number of Non-Interlocked pairs.
TI = Time taken to execute an Interlocked pair.
NI = Number of Interlocked pairs.
Further,
N = 2 (NNI + NI)
NNI = ((100 – X) / 100) N / 2, and
NI = (X / 100) N / 2.
Where,
N = Total number of instructions to be executed.
X = Percentage of interlocked pairs.
Now, (5.1) can be rewritten as :
TNI [((100 – X) / 100) N / 2] + T I [(X / 100) N / 2]

Now, consider the following for a program :
a) N = 100,
b) X = 50 %
c) I0 = 25
Logic Delays, typically The execution times for the machines are :

I) NON-ICALU MACHINE
From Table 5.1 :
TNI = IPAVE1 = I0 + 3.5. (5.1)
TI = IPAVE2 = 2I0. (5.1a)
Substituting in (5.1a), we get,
T1 = (I0 + 3.5) 25 + (2I0) 25 = 1962.5 Logic Delays.

II) MACHINE WITH ICALU
Again from Table 5.1 :
TNI = IICAVE1 = I0 + 4.17.
TI = IICAVE2 = I0 + 2.63.
Substituting in (5.1a), we get,
Total execution time for 50 pairs of instructions,
T2 = (I0 + 4.17) 25 + (I0 + 2.63) 25
= 1419.78 Logic Delays.
The machine with ICALU takes fewer logic delays than the Non- ICALU machine.

Chart 5.1 is a plot of (5.1a) with N constant (100) and varying X between 0 and 100 percent. It can be seen that theperformance of the Non-ICALU machine degrades, where as the performance of the machine with ICALU is almost constant as X increases. This is because the Non-ICALU has to execute more and more instructions in serial. In the next section Percentage Speed Ratio is calculated.

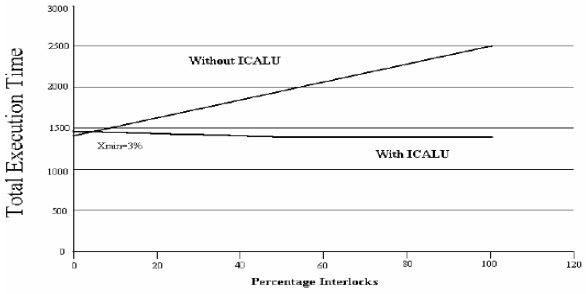

Fig 3 Percentage Interlocks Vs Total Execution Time

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

2) Percentage Speed Ratio

Percentage Speed Ratio of Machine 2 over Machine 1 is defined as :

[(T1 - T2) / T1] x 100 (5.2)

Percentage Speed Ratio reflects the time saved by one machine over the other. Using (5.1a) in (5.2), we get,

[(TNI1 – T NI2) (100 – X) + (T I1 – T I2) X] / [TNI1 (100 – X) + T I1 X] (5.2a)

Hence,

Percentage Speed Ratio of machine with ICALU over the Non-ICALU machine for the previous case (that is X = 50%) ≈ 28

Similarly, for (say) X = 75% :

Percentage Speed Ratio ≈ 37.

Thus the Percentage Speed Ratio increases as X increases. Chart 5.2 shows variation of Percentage Speed Ratio with interlock percentage (X). It can be seen clearly how Percentage Speed Ratio increases as interlock percentage (X) increases.

From chart we can see that at X ≈ 3%, the gain of the machine with ICALU is zero. Below thispoint the gain is negative, that is the machine withICALU is slower than the machine Non-ICALUmachine. This point can also be obtained by settingPercentage Speed Ratio to zero in (10.2a).

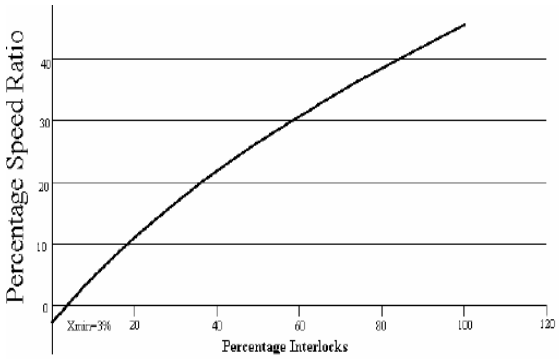

Fig :4 (Percentage Interlock Vs. Percentage Speed Ratio.)

5.TESTING PROCEDURES TESTING

The ICUNIT has two outputs, result of ALU1 and that of ALU2.The testing of the ICUNIT was done by categories. They are as follows :

1) CATEGORY 1 (ARITHMETIC FOLLOWED BY ARITHMETIC) :

Since there are three operands, the four sub categories are :

i) All positive numbers.
ii) Two positive numbers.
iii) One positive number.
iv) None positive.

2) CATEGORY 2 (LOGICAL FOLLOWED BY ARITHMETIC) :

The sub categories are :

i) Logical AND followed by Arithmetic.
ii) Logical OR followed by Arithmetic.
iii) Logical XOR followed by Arithmetic.
iv) Logical NAND followed by Arithmetic.
v) Logical NOR followe1d by Arithmetic.
vi) Logical XNOR followed by Arithmetic.

3) CATEGORY 3 (ARITHMETIC FOLLOWED BY LOGICAL) :

The sub categories are :

i) Arithmetic followed by Logical AND.
ii) Arithmetic followed by Logical OR.

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

iii) Arithmetic followed by Logical XOR.
iv) Arithmetic followed by Logical NAND.
v) Arithmetic followed by Logical NOR.
vi) Arithmetic followed by Logical XNOR.

4) Category 4 (LOGICAL FOLLOWED BY LOGICAL) :

Category 2 and 3 cover all possible categories here. Hence onlyone subcategory is considered (say) :
Logical AND followed by Logical AND.

6. SIMULATION RESULTS

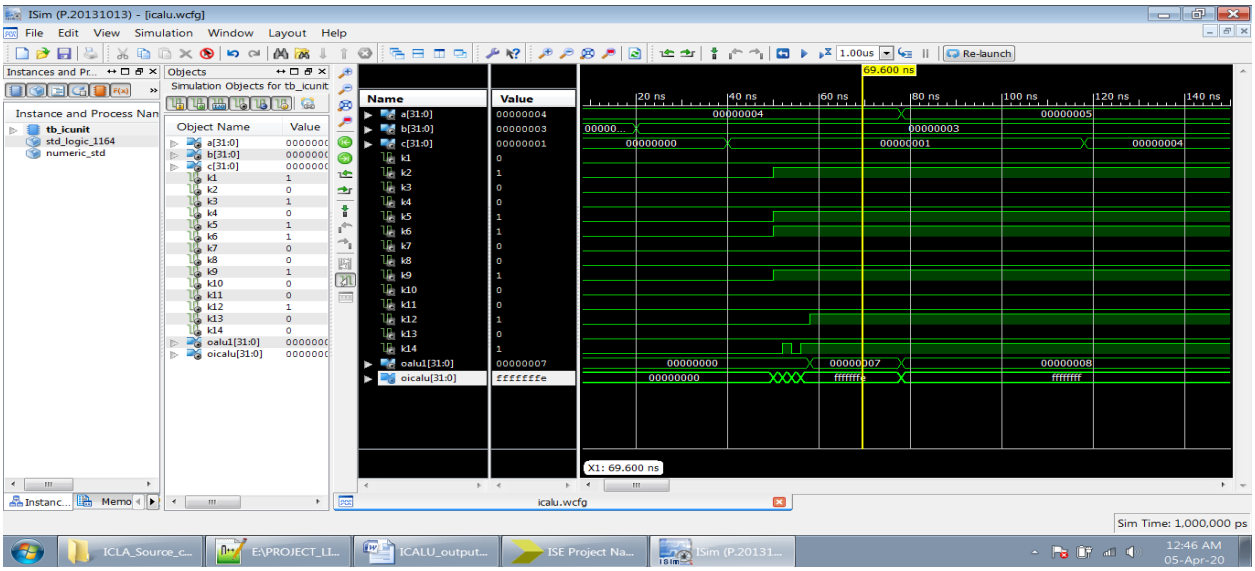

Fig1: Output of ALU1 &ICALU by using control signals

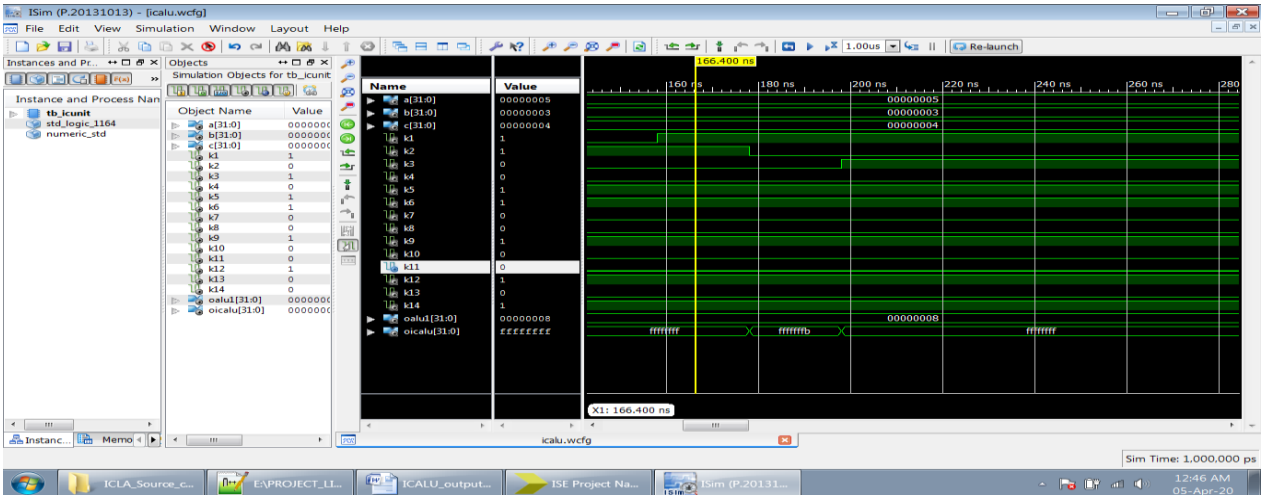

Fig2: Output of ALU1 & ICALU by using control signals

7.CONCLUSION

The objective of the thesis, execution of interlocked instructions in one instruction cycle. This was achieved byICALU successfully designed and implemented usingVHDL. Its functionality was verified through simulation.TheICALU can be implemented in just 2 logicdelays more than that of a conventional 2-1 ALU.

ISSN: 2278-4632
Vol-10 Issue-5 No. 3 May 2020

Theperformance of an ordinary (Non-ICALU) parallel machineand the machine with the ICALU incorporated in it, wascompared.

The following is concluded from the performance analysis:

The Percentage Speed Ratio of the machine with theICALU over the Non-ICALU machine depends only onthe amount of interlocked instructions in the code andnot on the total number of instructions. The Percentage Speed Ratio increases as the number ofinterlocked instructions increase. This is due to thedegradation in performance of Non-ICALU machines.Assuming an average of (50-75)% interlocks in a givencode, the Percentage Speed Ratio obtained is between(23-37)%, which implies that the ICALU, whenincorporated in a parallel machine saves up to a third ofthe total execution time of the Non-ICALU machine.

REFERENCES

1. J. Phillips, S. Vassiliadis, "High-Performance 3-1 Interlock Collapsing ALU's," IEEE Transactions on Computers, vol. 43, no. 3, pp. 257-268, Mar., 1994
2. S. Vassiliadis, J. Phillips, B. Blaner, "Interlock Collapsing ALUs", IEEE Transactions on Computers.
3. R. D. Acosta, J. Kjelstrup, H. C. Torng, "An instruction issuing approach to enhancing performance in multiple functional unit processors", *IEEE Trans. Comput.*, vol. 35, pp. 815-828, Sept. 1986.
4. N. P. 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.
5. N. Malik, R. J. Eickemeyer, S. Vassiliadis, "Instruction-level parallelism for execution interlock collapsing", *Comput. Architecture News*, vol. 20, no. 4, pp. 38-43, Sept. 1992.
6. D. W. Ruck, S. K. Rogers, M. Kabrinsky, M. E. Oxley, and B. W. Sutter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function,"IEEE Trans. Neural Networks, vol. 1, no. 4, pp. 296-298, Dec. 1990.
7. H. Ling, "High speed binary adder,"IBM J. Res. Develop., vol. 25, no. 3, pp. 156-166, May 1981.
8. M. J. Flynn and S. Waser,Introduction to Arithmetic for Digital Systems Designers. CBS College Publishing, 1982, pp. 215-222.
9. R. M. Keller, “Lookahead Processors,” **Computing Surveys,**Vol. 7, No. 4, pp. 514-537, December 1973.
10. R. M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units,"IBM J. Res. Develop., pp. 25-33, Jan. 1967.
11. R D Acosta , J Kjelstrup , H C Torng, An instruction issuing approach to enhancing performance in multiple functinal unit processors, IEEE Transactions on Computers, v.35 n.9, p.815-828, Sept. 1986
12. JAIN R.P . Digital Electronics , Printice hall
13. The Low Carb VHDL Tutorial ,Bryan Mealy 2004