International Journal on Emerging Technologies 11(2): 668-673(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Arithmetic and Logical Unit Design for Area Optimization for Microcontroller

Amrut Anilrao Purohit[1,2], Mohammed Riyaz Ahmed[2] and R. Venkata Siva Reddy[2]
[1]Research Scholar, VTU Belagavi (Karnataka), India.
[2]School of Electronics and Communication Engineering,
REVA University Bengaluru, (Karnataka), India.

(Corresponding author: Amrut Anilrao Purohit)
(Received 04 January 2020, Revised 02 March 2020, Accepted 03 March 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Arithmetic and Logic Unit (ALU) can be understood with basic knowledge of digital electronics and any engineer will go through the details only once. The advantage of knowing ALU in detail is two-folded: firstly, programming of the processing device can be efficient and secondly, can design a new ALU architecture as per the various constraints of the use cases. The miniaturization of digital circuits can be achieved by either reducing the size of transistor (Moore's law) or by optimizing the gate count of the circuit. The first has been explored extensively while the latter has been ignored which deals with the application of Boolean rules and requires sound knowledge of logic design. The ultimate outcome is to have an area optimized architecture/approach that optimizes the circuit at gate level. The design of ALU is for various processing devices varies with the device/system requirements. The area optimization places a significant role in the chip design. Here in this work, we have attempted to design an ALU which is area efficient while being loaded with additional functionality necessary for microcontrollers. One novel approach in our work is that the concept of hard-wired architecture (for multiplier and divider) is borrowed from Digital Signal Processors. Another aspect worth mentioning is the implementation of barrel shifter which will considerably reduce the execution time. The hardware design is made via Xilinx 9.1 ISE on Virtex 5 FPGA and verified using ISim. The results are very encouraging, and it seems that a thorough understanding and proper implementation of ALU will allow us to put other units in their place.

Keywords: ALU, Verilog, FPGA, Microcontroller, Area Optimization and Instruction set.

I. INTRODUCTION

The power of any embedded system is assessed based on how powerful the processor is, and the capability of processor is adjudicated by the ALU which invariably boils-down to the instruction set. ALU is a core part of a processor and its design will play a vital role in execution of every instruction [1].

The three primary parameters under consideration for ALU design are always power dissipation, speed of operation and chip area. As the designer tries to minimize one parameter, at least one of the other two parameters increases, thereby creating a challenge. Hence, optimizing all the three parameters in a proper balance is crucial. The circuit designers and researchers have tried hard to explore and exploit every avenue in the chip designing to achieve less power consumption, increased speed of operations and optimize area on the chip [2]. The advancement in technology and ever rising demand for hand-held devices have compelled chip designers to focus on area reduction/ optimization.

The miniaturization of the digital circuits can be achieved either by reducing the size of transistor or by minimizing the gate count of the design. The first approach tries to keep up with Moore's law and has reached sub-nanometer technology leading to advancements in CMOS VLSI [3]. The latter approach has less buyers due to the need of sound knowledge in logic design which demands more human efforts. It deals with application of Boolean rules to simplify the equations thereby reducing the complexity of the circuits. The very careful design and testing of ALU are crucial in the process of area optimization [4].

The Central Processing Unit (CPU) is the core of any embedded system [5]. The CPU's are categorized based on the processing devices as Application Specific Integrated Circuits (ASIC), Field Programmable Gate Arrays (FPGA), Microprocessor, Digital Signal Processor (DSP), Microcontrollers. Each of these processing devices are targeted for specific application. The major feature that distinguishes all the above specified processing devices is primarily it ALU. The ALU is generally an asynchronous logic circuit which acts as the core of any processing device. The ALU of the processing devices vary in its functionality, decoding logic, size of the instruction set based on the device it's going to be incorporated.

ASIC's are extremely fast as they are designed for a dedicated application and occupy the least amount of area on the chip. The design and fabrication of an ASIC consumes significant amount of time and man hours making them expensive at a smaller scale but would turn out to be cost effective at large scale. FPGA designs are slower than the other counterparts and occupies a significantly larger area on chip but dominate over the others due to their hardware reconfigurable architecture, and shorter design and testing time.

The Microprocessors find their application in the high-speed fixed-point arithmetic, logical and data transfer operations [6]. They are suitable for the more of a general-purpose application ranging from a personal computer to a super computer. They are designed generally using Von Neumann Architecture along with the complex instruction set (CISC) which makes the decoding logic bulky at the same time making it programmer friendly [7].

Digital Signal Processors operate on floating-point arithmetic and is supported by a Multiply and Accumulate (MAC) unit to perform faster multiplication and addition; and a Barrel Shifter to perform faster logical operation [8]. This increases the area occupied on the chip, hence increasing the cost. The DSP's

operate at lesser clock frequencies compared to Microprocessors, which reduces the power dissipation on the chip. The DSP's find their application in all multimedia applications due to the presence of MAC unit, Barrel shifter and a floating point ALU.

Microcontrollers are primarily system-on-chip (SoC), which works on fixed-point arithmetic at even lower clock frequencies compared to DSP's. The lesser external interface is required for the system to operate for a similar functionality when compared to Microprocessors. Microcontrollers operating at very low speeds that is suitable for the rate of change of physical parameters makes them handy to interface with the external world. The lower prices of microcontrollers add on to the demand for them in the industry. Both DSP's and Microcontrollers are designed using Harvard Architecture along with the reduced instruction set (RISC) leads to the smaller decoding logic thus reducing the chip area compared to microprocessors [9]. The multiply/divide operations are performed using microcode architecture in Microprocessor/ Microcontroller, whereas DSP's use hard-wired architecture [10]. Several architectures have been investigated previously with an intention to achieve trade-off between power and performance [1, 5]. Little has been done in the area optimization, which will lead to a win-win situation for both the cases: as lesser area requires lesser power and reduced number of gates will lower the complexity thereby increasing performance.

The existing architecture occupy more are on the chip or have lesser features or functionalities or instructions to work with. The fewer instructions makes the programs more complex and requires more ALU operations to perform the same task. Hence making the systems slower. The area optimization of ALU is necessary so as to reduce the chip area and eventually reduce the cost on the system. It not only reduces the chip area but also reduce the amount of power requirement as the number of gate operation in the circuit reduces.

The aim is to achieve lower area and decent performance with our proposed architecture. The work focuses on optimizing the circuit by reducing the gate count and designing a more powerful device. The power of any embedded system is assessed based on how powerful the processor is; and the capability of the processor is adjudicated by ALU, which invariably boils down to the instruction set. The very careful design and testing of the ALU are crucial in the process of area optimization. Verilog is used for the design and verification of the IP core. The Verilog designs are said to occupy lesser area if the number of Look-Up-Tables (LUT) and the number of slices is reduced in the design. Thus, conclude that the chip area can be reduced significantly by reducing the gate count of the design for the same operation.

The remainder of the paper is arranged as follows: Section II describes the proposed ALU architecture, along with experimental set-up. Section III provides details on the instruction set. Section IV presents the results along with the discussion. Finally, the paper concludes in section V. To reach to a broader audience, the paper also provides research gaps and future scope in this section.

II. PROPOSED ARCHITECTURE

The ALU operations are primarily divided into Arithmetic operations and Logical operations. The Arithmetic operations like Addition, Subtraction, Increment, Decrement and so on are performed by Arithmetic unit, while Logical unit performs AND, OR, XOR, complement, rotate, shift operations etc [11, 12]. The multiplexers and de-multiplexers play a vital role in selecting the respective results generated by the ALU [13]. The Microcontrollers need to handle the individual port lines which necessitate the need for bit handling instructions. Thus, an ALU of a Microcontroller needs to cater to both the bit-wise and the byte-wise Logical operations, arithmetic and data transfer operations. In spite of incorporating all the ALU operations, the challenges lie in optimizing the area occupied on the chip.

Adders form the core of any arithmetic unit with varied designs like Ripple Carry Adder (RCA), Carry Look Ahead Adder (CLA), Carry Save Adder (CSA), Carry Select Adder (CSeA) etc. The respective adders are selected based on the parameter to be minimized. If the focus is on reducing the delay, then CSeA would be a better choice but penalizes with a large chip area and more power-hungry. While CLA would be relatively slower than the CSeA but occupies significantly lesser area. The CSA would be a better choice as the number of bits needed to be added increases as in case of multipliers [14]. The RCA occupies the least possible area but is slower as the number of bits to be added increases.

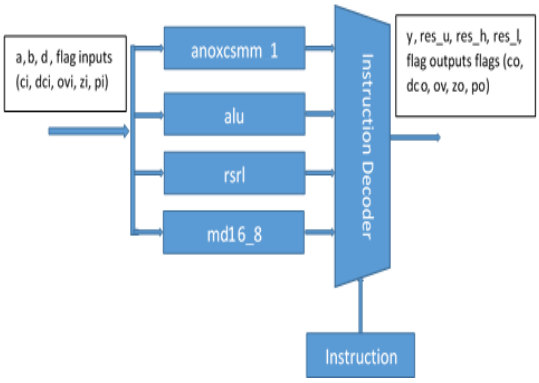


Fig. 1. A block diagram of proposed ALU architecture.

The proposed architecture focuses on the area optimization and hence RCA has been used in the entire design. The block diagram of the proposed ALU is shown in Fig.1.

The ALU is designed to be incorporated in a Microcontroller along with a barrel shifter to make logical operation yield quicker results [15]. The complete instruction set of the ALU has been designed. The pin diagram each of the modules/ units, namely byte ALU, Bit ALU, Barrel Shifter, ALU top module, and 8*8 bit multiplier and 16/8 bit divider module are given in Fig. 2.

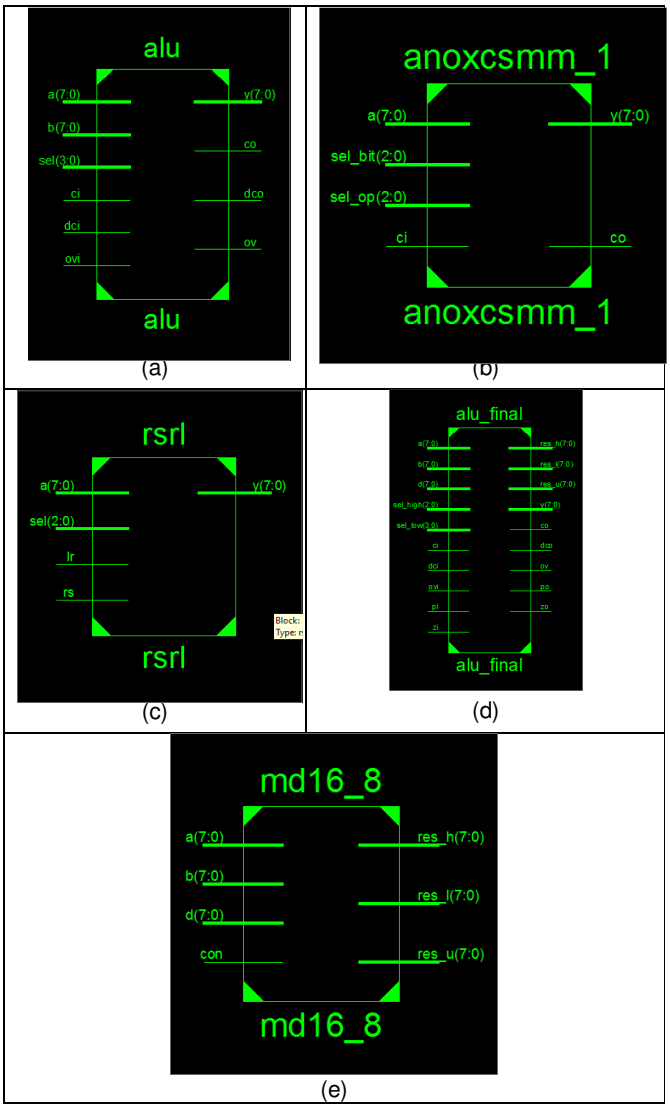


Fig. 2. Pin description of proposed ALU architecture (a) Arithmetic and Logical Unit (excluding multiplier and divider) (b) bit-handling unit (c) Barrel Shifter (d) ALU top module (e) 8*8 bit multiplier and 16/8 bit divider module.

III. INSTRUCTION SET

There are 8 bit-handling instructions designed as described below:

MOV ci, bit: Transfers the content of bit location identified by the Op-code (lower bits) in input 'a' to carry flag (represented by co output in the waveform).

XOR ci, bit: Performs XOR operation between the content of bit location identified by the Op-code (lower bits) in 'a' input with carry flag (represented by ci input in the waveform) and store it in the carry flag (represented by co output in the waveform**).**

AND ci, bit: Performs AND operation between the content of the bit location identified by the Op-code (lower bits) in 'a' input with carry flag (represented by ci input in the waveform) and store it in the carry flag (represented by co output in the waveform).

OR ci, bit: Performs OR operation between the content of the bit location identified by the Op-code (lower bits) in 'a' input with carry flag (represented by ci input in the waveform) and store it in the carry flag (represented by co output in the waveform).

MOV bit, ci: Transfers the carry flag (represented by ci input in the waveform) to the bit location identified by the Op-code (lower bits) in output y.

INV bit: Compliments the content of the bit location identified by the Op-code (lower bits) on input 'a' and stores in the same bit position on output y.

CLR bit: Clears the content of the bit location identified by the Op-code (lower bits) on input a and stores in the same bit position on output y.

SET bit: Sets the content of the bit location identified by Op-code (lower bits) on input 'a' and stores in the same bit position on output y. The details of all the instructions and the Op-Codes is given in Table 1.

Table 1: Table of Instruction Set.

		Op-Code (Lower bits)							
Op-Code (Higher bits)	**Operations**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	Arithmetic/Logical	NOP	CLR MEM	INC MEM	DEC MEM	ADD MEM,A	ADC MEM,A	SBB MEM,A	SB MEM,A
0001	Arithmetic/Logical	CPL MEM	XCHG A, MEM	MOV A,MEM	MOV MEM,A	RLC MEM	RRC MEM	MUL	DIV
0010	Arithmetic/Logical	CMP	CLR	INC	DEC	ADD	ADC	SBB	SB
0011	Arithmetic/Logical	CPL	XOR	AND	OR	RLC	RRC	DAA	DAS
0100	Rotate left by no. bits	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
0101	shift left by no. bits	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
0110	Rotate right by no. bits	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
0111	shift right by no. bits	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1000	Mov ci,bit	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1001	Xor bitwise	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1010	And bitwise	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1011	Or bitwise	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1100	Mov bit,ci	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1101	Inv bitwise	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1110	Clr bit (bit position)	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7
1111	Set bit (bit position)	Bit-0	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6	Bit-7

IV. RESULTS

The entire ALU is divided into 5 major blocks:

anoxcsmm_1: This block operates upon all the bit-wise operations carried out by the ALU. These operations are carried out when the Op-Code (higher bits) range from 1000 to 1111. The related output waveforms are shown in Fig. 3 (a) (b)

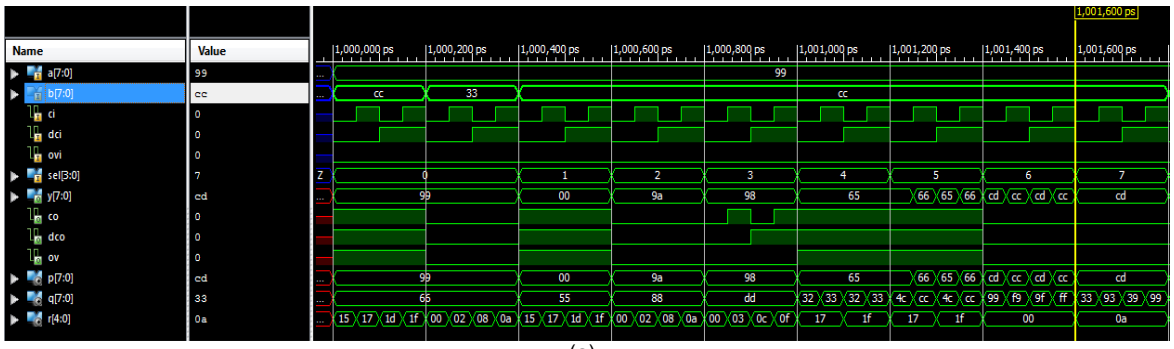
alu: This block operates upon 8-bit data arithmetic and logical operations. These operations are carried out when the Op-Code (higher bits) range from 0000 to 0011. The results of operation wrt 0000 and 0001 would be stored in the memory locations whereas 0010 and 0011 would be stored in accumulator. The results of multiplication and division would be stored in exclusive memory locations meant for multiplication and division registers. The NOP instruction, data transfer and exchange instructions are part of this Op-code range. The RLC and RRC instructions perform left and right rotation of the data through carry flag. The waveforms are shown in Fig. 3 (b).

rsrl: This block performs rotate and shift operations left or right by the number of bits specified in the Op-Code (lower bits). These operations are carried out when the Op-Code (higher bits) range from 0100 to 0111. The barrel shifter is designed in this block. The waveforms are shown in Fig. 3 (d).

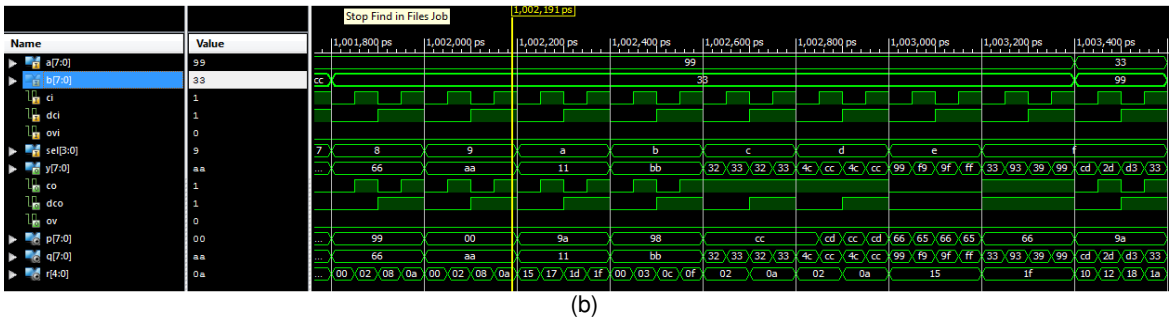
md16_8: This block performs 8-bit * 8-bit multiplication of a and b inputs to yield 16-bit result on res_h and res_l output lines and division of 16-bit (a and b inputs put together) by 8-bit number on d input to yield 16-bit quotient on res_u and res_h and 8-bit reminder on res_l output lines. The waveforms are shown in Fig. 3 (e).

alu_final: This block integrates anoxcsmm_1, alu, rsrl, and md16_8 blocks. The design summary of the ALU top module is shown in Fig. 4.

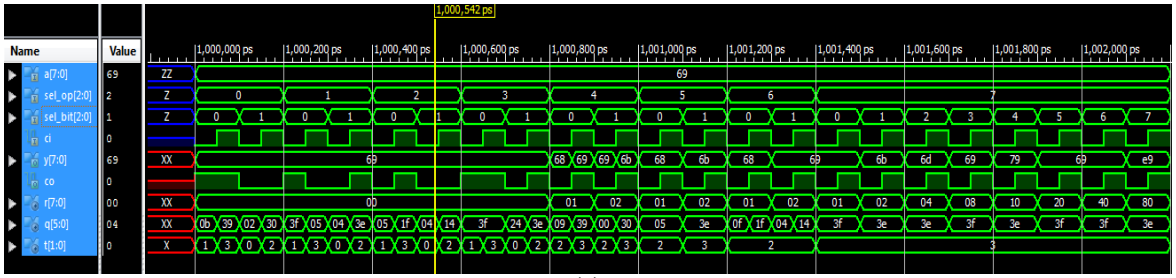
The obtained results of the proposed design are summarized in Table 2.



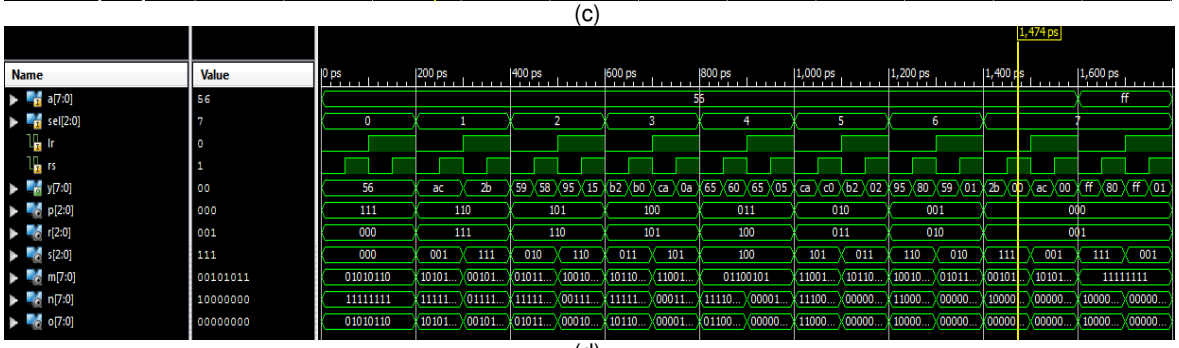
(a)



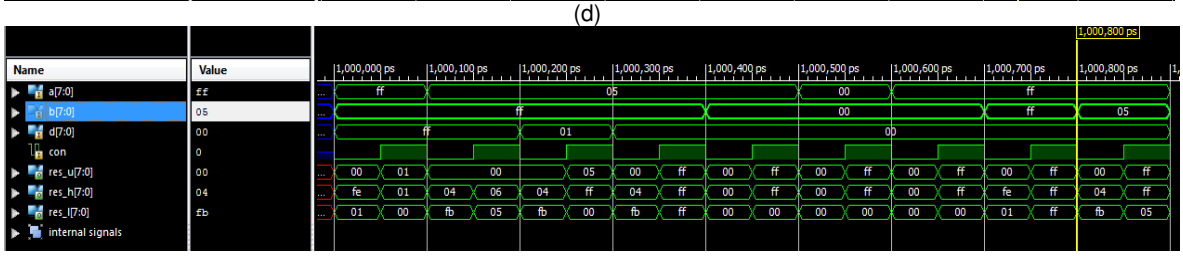
(b)



(c)



(d)



(e) 8*8 bit multiplier and 16/8 bit divider module.

Fig. 3. Waveforms, (a) and (b) bytewise arithmetic and logical operations, (c) bitwise operations, (d) barrel shifter, (e) 8*8 bit multiplier and 16/8 bit divider module.

alu_final Project Status (12/30/2019 - 20:05:52)			
Project File:	alu_final.xise	**Parser Errors:**	No Errors
Module Name:	alu_final	**Implementation State:**	Placed and Routed
Target Device:	xc5vlx30-3ff324	• **Errors:**	No Errors
Product Version:	ISE 14.2	• **Warnings:**	291 Warnings (291 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice LUTs	426	19,200	2%	
Number used as logic	426	19,200	2%	
Number using O6 output only	338			
Number using O5 and O6	88			
Number of occupied Slices	190	4,800	3%	
Number of LUT Flip Flop pairs used	426			
Number with an unused Flip Flop	426	426	100%	
Number with an unused LUT	0	426	0%	
Number of fully used LUT-FF pairs	0	426	0%	
Number of slice register sites lost to control set restrictions	0	19,200	0%	
Number of bonded IOBs	73	220	33%	
Average Fanout of Non-Clock Nets	4.16			

Fig. 4. Design Summary of proposed ALU, showing the number of LUTs, slices.

Table 2: Design Summary.

Module	No. LUT	No. Slices
anoxcsmm_1	13	7
alu	66	30
rsrl	22	7
md16_8	300	133
Alu_final	426	190

V. CONCLUSION

An asynchronous ALU has been designed with 4 modules namely: bit handling ALU block, byte handling ALU, Barrel shifter block, and hard-wired Multiplier and Divider block. The proposed architecture is area efficient and is optimized at gate level. The novel architecture incorporating hard-wired multiplier and divider along with barrel shifter have shown good results. Thus, the chip area was reduced significantly by reducing the gate count of the design for same operations.

As a part of future enhancement, we can focus on integrating the proposed ALU on to a microcontroller and further extend it to ASIC to reduce the delay. Various additions to the architecture such as dual-core ALU design, is possible. We can also include various communication protocols such as Serial Peripheral interface, Universal Asynchronous Receive and Transmit (UART), and Inter IC Communication (I2C).

VI. FUTURE SCOPE

The future scope of this work is to optimize the area of the chip using various modelling style and look into which styles suits the most in the ALU design and further optimize the area occupied on the chip retaining all the features or instructions deigned in the proposed ALU, thereby increasing the performance of the circuit.

ACKNOWLEDGMENT

The authors would like to thank Rukmini Education Charitable Trust for providing the necessary facilities and support to carryout this work.

Conflict of Interest. The authors declare no conflict of interest associated with this work.

REFERENCES

[1]. Chen, J., Vasudevan, D., Popovici, E., & Schellekens, M. (2011). Design of a Low Power, Sub-Threshold, Asynchronous Arithmetic Logic Unit Using a Bidirectional Adder. In *2011 14th Euromicro Conference on Digital System Design*, 301-308. IEEE.

[2]. Kulkarni, R., & Kulkarni, S. Y. (2014). Energy efficient implementation of 16-Bit ALU using block enabled clock gating technique. In *2014 Annual IEEE India Conference (INDICON)*, 1-6. IEEE.

[3]. Syamala, Y., & Tilak, A. V. N. (2011). Reversible arithmetic logic unit. In *2011 3rd International Conference on Electronics Computer Technology*, *5*, 207-211. IEEE.

[4]. Adamec, F., & Fryza, T. (2009). Design and Optimization of ColdFire CPU Arithmetic Logical Unit. In *2009 MIXDES-16th International Conference Mixed Design of Integrated Circuits & Systems,* 699-702. IEEE.

[5]. TM, A., & Selvakumarraja, S. (2015). Design of Low power four function 8-bit ALU for nano based systems. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, 1583-1587. IEEE.

[6]. Hinsu, N., & Suryavanshi, D. (2014). A prototype design for microprocessor based on Verilog HDL. In *International Conference for Convergence for Technology-2014*, 1-5. IEEE.

[7]. Šilc, J., Ungerer, T., & Robic, B. (2000). A survey of new research directions in microprocessors. *Microprocessors and Microsystems*, *24*(4), 175-190.

[8]. Eyre, J., & Bier, J. (2000). The evolution of DSP processors. *IEEE Signal Processing Magazine*, *17*(2), 43-51.

[9]. Palekar, S., & Narkhede, N. (2016). 32-Bit RISC processor with floating point unit for DSP applications. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2062-2066. IEEE.

[10]. Bhagat, S. M., & Bhandari, S. U. (2018). Design and Analysis of 16-bit RISC processor. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1-4. IEEE.

[11]. Gupta, N., Shrivastava, S., Patidar, N., Katiyal, S., & Choudhary, K. K. (2012). Design of one bit arithmetic logic unit (ALU) in QCA. *Int. J. Comput. Appl. Eng. Sci.*, *2*(3), 281-285.

[12]. Muduli, G., Pradhan, B., Jena, M. R., & Nath, S. (2014). Design of an Efficient Low Power 4-bit arithmatic Logic Unit (ALU) using VHDL. *International Transaction of Electrical and Computer Engineers System*, *2*, 144-148.

[13]. Yadav, P., Kumar, G., & Gupta, S. (2014). Design and implementation of 4-bit arithmetic and logic unit chip with the constraint of power consumption. *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, *9*(3), 36-43.

[14]. Kathirvelu, M. (2019). Design and Implementation of Optimized Area and PDP Multiplier for High Speed Digital Circuit Applications. *International Journal of Recent Technology and Engineering (IJRTE),* 1081-1085.

[15]. Trivedi, P., & Tripathi, R. P. (2015). Design & analysis of 16 bit RISC processor using low power pipelining. In *International Conference on Computing, Communication & Automation*, 1294-1297 IEEE.

How to cite this article: Purohit, A. A., Ahmed, M. R., Reddy, R. V. S. (2020). Arithmetic and Logical Unit Design for Area Optimization for Microcontroller. *International Journal on Emerging Technologies*, *11*(2): 668–673.