Ann. For. Res. 65(1): 9707-9721, 2022
ISSN: 18448135, 20652445

ANNALS OF FOREST RESEARCH
https://www.e-afr.org/

INTEGRATION OF CO-ALU AND MULTIPLIER WITH 8-BIT CONTROLLER USING VHDL AND RECONFIGURABLE HARDWARE

Tanaji M Dudhane*
Research Scholar, Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Chennai-600 119, India.

T. Ravi
Research Supervisor, Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Chennai-600 119, India.
***Correspondence: tmdudhane@yahoo.com**

Abstract: The 8-bit microcontroller with their growing demand in different areas of applications as medical, consumer appliances, smart metering, ATMs, vending machines, with small modifications in the hardware architecture, they are performing efficiently in terms of operation throughput and power consumption. This paper presents the integration of 16-bit co-operative ALU and 8-bit multiplier to 8-bit controller using VHDL. The goal of this research was to improve the computability of the 8-bit controller by creating a VHDL model with C-ALU and multiplier. This research presents a novel implementation of hybrid 8-bit controller design, which can process mixed size data operations efficiently by utilizing multiple ALUs in single data path. For the implementation of 8-bit controller with C-ALU and multiplier, the selected hardware platform is the SPARTAN-7 device family XC7K160T-3FBG676E FPGA. Maximum clock frequency achieved on the selected device is 31.15 MHz rather than 20 MHz of the original controller. The average performance 71% is improved in multi-byte operational cases.
Keywords: ATMS; Co-Operative ALU; VHDL; ALU; FPGA

1. Introduction

Exponential growth of enriched integrated circuits along with cutting edge embedded devices simplifies the implementation of many real world applications. Today, almost every field of application incorporates most tiny embedded devices. These embedded systems mainly perform data capturing, data processing and controlling activity. Embedded system manufacturer always tries to improve overall system performance by means of reducing overall processing time, minimizing required power along with improving overall throughput.The accomplishment of successful optimal digital circuit is either by optimizing the number of gate-count for the circuit under design or by improving the area of transistors over the silicon wafer with latest silicon technology.

The per transistor area optimization over the silicon wafer now approach as to sub-nanometre semiconductor technology due to improvements in CMOS VLSI technology over the decades following the Moore's law predictions. Although FPGA designs are more time consuming and

utilise larger semiconductor area as compared to other processing device technology, still it dominates over its counter devices due to its reconfigurable architecture along with significant less implementation time.

Microprocessors replaced many counter devices due to their strong design of data path and ALU architecture, capable to perform variety of fixed-point arithmetic as well as many data transfer operations at high speed.

There are different modifications in the instruction length from 32 bits to 14 bit. So, there is change in the integration of different modules of different bit size to the controller. Integrated modules are ALU of different word length, multiplier or even MAC unit. The architectural modifications lead to improve the execution speed using FPGA's, which is providing a complete solution for different cores, processors as well as controllers. There is huge development in the area like VLSI, embedded systems as well as in the features of microprocessors and microcontrollers specifically in terms of its hardware design. Modern CPUs work with complex ALUs for implementation of instructions (Florin M. Mihai et al. 2009). Word width affects the speed and improvement in power consumption also (R. Uma 2012). The 8-Bit controller has a strong position in current market even there is development of different controllers from 8-Bit to 16 or 32 bits, because of its reliability, low cost and low power consumption.

Asynchronous processor design mostly follows the top-down design approach dividing into tiny modules so that it can be verified and analysed appropriately (Amrut A.Purohit et al. 2020) and designed a complete ALU using repeated structural VHDL code as this was not successful to synthesize the code because of the limitations of the old technology for FPGA. The hardware and software cooperative design to implement 8-Bit RISC processor over a reusable FPGA platform elaborated (A. J Salim et al. 2012). This implementation provides more upgrading and operating clock flexibility which saves the development time and performance. The detailed implementation of 8-Bit processor basic modules which are designed and tested separately such as ALU, control module and a memory module as major building blocks with VHDL and implemented over Xilinx-Spartan-III board utilizing around 132 slices (E Ayeh.et al. 2008).The implementation of Arithmetic Logic Unit in which all modules were designed and coded using VHDL. (Shika Khurana et al.2012).The 32-Bit RISC Processor using VHDL which with few additives were applied and examined on Xilinx FPGA (Wael M et al. 2008).

The 8-Bit RISC processor with 16-Bit instructions using Field Programmable Gate Arrays Spartan 3E tool using Verilog hardware description language implemented (Ramandeep Kaur et al. 2014). The 8-Bit processor using RISC architecture with 144 instructions implemented using Harvard architecture. (Jikku Jeemon et al. 2015). The 20 bit RISC system using VHDL over FPGA implemented (Deepak Kumar et al. 2013). The modified power efficient 8-Bit processor circuitry compatible with 16-Bit wide RISC processor architecture performing MAC operation is coded in VHDL (A.J Salim et al. 2013) and implemented over the FPGA. The design and implementation of the arithmetic and logical processing unit for SIMD processor is carried out (Mohammed F. Tolba et al. 2016) and this ALU design is based on Approximated Precision Shader and Look-Up Table (LUT) Multiplier. The battery operated IoT applications as well as running over the energy

harvesting modules, hence making energy efficient of high importance and as the size of the applications become more complex, these devices required more processing power. The outcome of this was the power efficient design implementation for the generalized applications which was achieved by reducing hardware complexity of the processor core (Sergio F. Johann. et al. 2016).The unavailability of coprocessor accommodation over the chip, which is now possible due to the semiconductor technology (Raju Bansal et al. 2017) and analyzed the implementation by comparing it with the performance of design utilized power and occupied chip area.

A new FPGA based floating point processor aimed at speeding up the translation work in a free CNC machining method. In the CNC system design movement planning and translation is done on an FPGA based processor, which removes the complex computational tasks from the processor host (J. Dong et al. 2017). The challenges include optimization for processing speed, occupied chip area, required power and energy of a digital design (Kunal Jadhav et al. 2015), even there are few challenges faced by the designers for design optimization without sacrificing each parameter. The 16-Bit Vedic Multiplier based on digital design implemented through VHDL language and compared the implementation details with the Array multiplier (G. Challa Ram et al. 2016).Complex multiplier design and implemented it with the quad real multiplier solution (K. Deergha et al. 2016) and mainly discussed complex multiplier implementation over the FPGA development system. The design of Harvard Architecture for the microcontroller module wise, starting by the ALU following by the register, RAM, ROM and finished with the control unit in (Patricio Bulic et al.2010). The present research paper has been initiated by investigating the original implementation of PIC16F84 microcontroller using VHDL coding over the FPGA hardware. A similar work can be found in (A.J. Salim et al. 2013), modified 8-bit RISC processor base architecture to work as compatible with 16-bit RISC processor improving overall performance using the MAC operation with low power consumption. The designed extended address, data lines, also re-modified base, 8-Bit core architecture perform equally to 16-Bit processor unit. This modified ASIP based new instruction set based on low end 8-Bit microcontroller helps to perform complex data processing such as DSP operations over the low-end RISC processors. (Khoi Nguyen et al. 2014). Hardwired control approach based 16-Bit RISC processor using hierarchical approach so as to combine basic units using structural programming. (Pravin S. Mane et al. 2006).

The implemented an online data acquisition and treatment system using Tracking Numerical Treatment (TNT) boards for the most ambitious future experiments carried out in European country (Mark G Arnold 2006). The Octa Lynx 8-Bit RISC microcontroller over FPGA chip with external memory and instruction decoder pipelined (Ryszard Gal et al. 2011); hence overall improvement in timing is achieved. The basic 8-Bit asynchronous microprocessor designed and the designed circuit was realized using a general FPGA development board (Tatsuya Suto et al. 2010). Larger number of multiplications utilized in digital signal processing application with the error free implementation of multiplier architecture incorporating pipelined correction with power efficient maximum device utilization (Patricio Bullic et al. 2010). The author's implemented

processor based on Harvard's specific RISC instruction set Architecture which performs complex instructions in minimum steps (Antinio H. Zavala et al. 2011).

The part of internal data memory of size 68 bytes (RAM file) has been named as Register1 address through the address multiplexer with the input of FSR register and internal data bus. Thus, data exchanged between Register1 and internal data memory is pointed by the FSR register content. Address multiplexer is used to give the addresses of instructions saved in the RAM file. Instruction decoder and control module is responsible for taking input from instruction register as an address of instruction, decodes and generates ready to execute synchronized sequence of triggering signals for the required multiple internal modules to perform completely execution of instructions. Power on reset (POR) is enabled whenever sufficient voltage is available over the MCLR pin. At the time of reset applied through MCLR pin, the content of FSR register which is indirect data memory address now points to very first location from Register1. Similarly, the content of ALU and the bank select-Bit for the data memory are resets to zero. ALU unit performs 8-Bit operations on 8-Bit operands. Most of the instructions store the results into a destination 'W' register called as a 'working register' similar to the concept of 'Accumulator' register from microprocessors and basic microcontrollers. A timer, TMR0, is used as 8-Bit timer/counter with programmable pre-scaling of 8 bit. The real word digital interaction is achieved through two independent input output ports namely PORTA with 5 bits as RA4-RA0 and PORTB having 8 bits as RB7 – RB0, as physical pins. All 8-Bit instructions are single cycle, except program branching instructions, which are of two cycles. It supports all the basic addressing modes with 4 special purpose interrupt sources serviced as external RB0/INT pin, overflow of timer TMR0, Interrupt on-change port B <7:4> and Data EEPROM write complete.

The Figure 1 shows the block diagram of architecture of PIC16F84 Microcontroller. It has 20 pins small microcontroller with 35 single word single cycled instructions running at 20 MHZ. It has 1k of program memory with 68 bytes of RAM as well as EEPROM memory. It also includes 15 Special Function hardware Registers (SFRs) and 8-level deep hardware stack.

It is now a challenging task to find the best suitable controller out of plenty eligible devices for the specific application. In real world embedded application, custom developed embedded system has to perform a fixed repetitive task and process the generated data to drive or shape the output according to the system requirement.

As per the need of society for different appliances using controllers and as per the development in this field of controllers, there is a growing need to improve the speed of execution for facilitating the execution of the number of operations. Hence controller is modified.

This paper is organized as follows: Section 2 describes the proposed design of the enhanced controller. Section 3 consists of the instruction executed by the proposed controller and its detailed design. Section 4 describes the result. The section 5 present the discussion and section 6 includes conclusion.

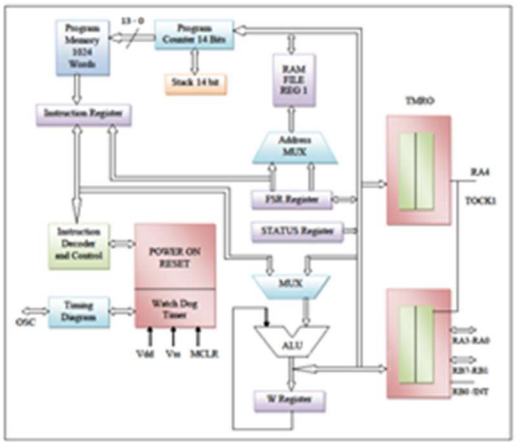

Figure 1.16F84 Basic Block of Controller

2. Proposed Design

The main aim of the present paper is to design a high performance 8-Bit controller with 16-Bit arithmetic and logical unit and 8-Bit multiplier using FPGA. This work incorporates innovative architectural expansion of the pre-existing well-known PIC microcontroller to extend unattended functionality covering a variety of applications. The main purpose of this paper is to design and implement, 8-Bit controller core scaled up with 8-Bit multiplier and 16-Bit C-ALU for speed improvement of instruction execution, and to enhance capabilities of 8-Bit controller. FPGA hardware platform is used for realization of the prototype as it is more cost-effective solution as compared to the other counterparts. The reconfigurable structure of FPGA programmed with Xilinx 14.7 ISE supports to simulate and analyze all the implementation done in this work to achieve improvement in speed which is challenging, with execution of number of 16-Bit instructions with 8-Bit controller.

2.1 Block Diagram of 16-Bit Co-operative ALU

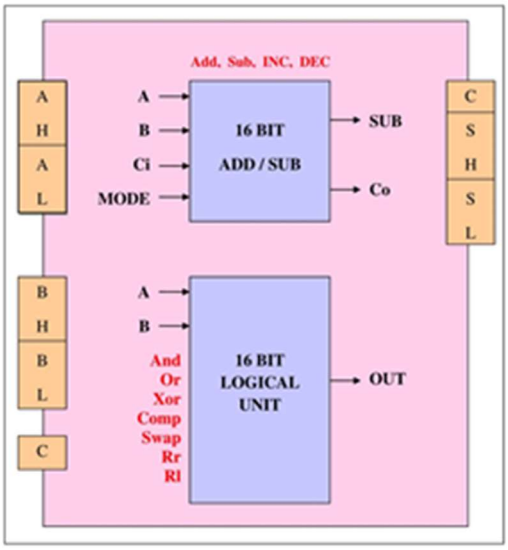

Figure 2.Block Diagram of 16-Bit CALU

Figure 2, shows that the block diagram of 16-Bit CALU is performing four,16-Bit arithmetic instructions as addword, subword, incword, decword and 7 instructions of 16 bits of logical and rotate instructions as andword, xorword, compword, swapword, rrword, rlword. Total 11 instructions of 16-Bit size are executed with 8-Bit controller, keeping its original functionality. For the implementation and design, three registers of 16-Bit SFRs are designed.

2.2 Block Diagram of proposed Controller

Figure 3 shows the proposed complete block diagram of enhanced data path of PIC16F84A controller. The 8-Bit controller is integrated with 16-Bit C-ALU for the implementation of 16-Bit instructions. Instruction size is modified to 15 bit from 14 bit so as to accommodate 11 add-on instructions of 16-Bit data operations as well as for operation of multiply instruction. Internal RAM size of 68 bytes is sufficient for storing 8-Bit as well as 16-Bit data.

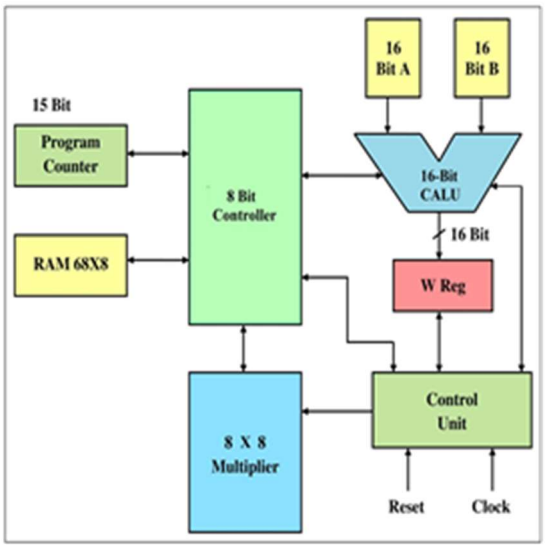

Figure 3.Proposed Block Diagram of Enhanced Controller

This modified 8-Bit controller design follows general three stage pipelined architecture as Fetch, Decode and Execute. In instruction fetch stage from program memory, 15 bit instruction is fetched and stored into 15 bit instruction register. Subsequently Decode unit decodes this 15 bit instruction fetched from the instruction registers and provides the synchronized sequential operation pulses to the respective module present in the data path. Finally the Execute Unit completely processed the sequence of operations that were decoded by the Decoder unit through 8-Bit or 16-Bit ALU. The Control Unit synchronizes search module to follow the sequence of flow of execution of an instruction in several combinational units that verifies the completion signal and activates the enable signal. The structural modification is done over the existing controller data path, to process more data bits per instruction cycle as compared to the conventional 8-Bit data processing approach. Although, C-ALU required more space as compared to existing 8-Bit ALU space, it improves the throughput of the existing 8-Bit ALU by performing the data intensive arithmetic and logical operations. With the integration of 16-Bit C-ALU and 8-Bit multiplier, the new design enhances the capability of original architecture of 8-Bit controller.

3. Detailed Diagram of Proposed Controller

Figure 4 shows the detailed diagram of proposed controller. The diagram below consists of Arithmetic unit, Decoder unit, Logic unit, Rotate and Shift unit, Bit Set Clear unit for 8-Bit controller and 16-Bit C-ALU, 8-Bit Multiplier and 16-Bit Special Function Registers. The enhanced controller here is called as the hybrid controller because it executes 8-Bit as well as 16-Bit add-on instructions. The length of instruction size is changed to 15 bit from 14 bit, in addition of MSB bit which is an extra bit.

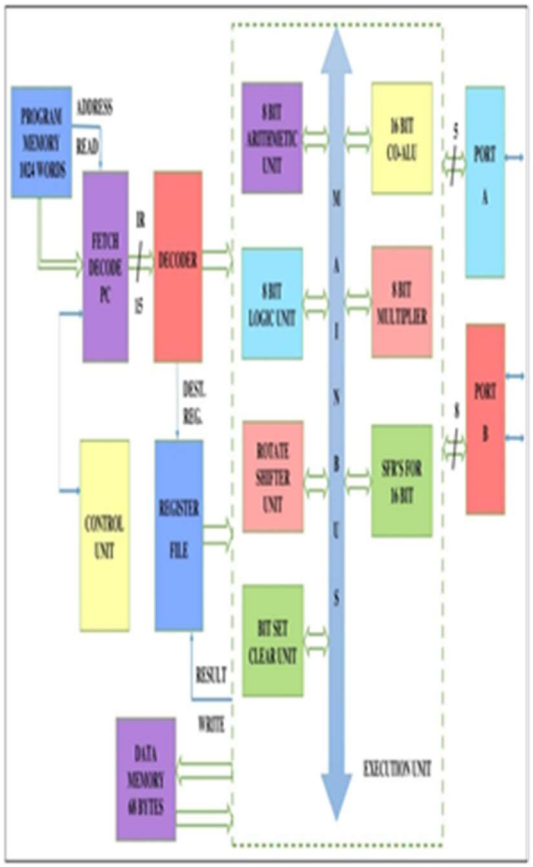

Figure 4.Detailed Diagram of Proposed Controller

3.1 Proposed Instruction Set

Proposed C-ALU integrated instruction set includes four arithmetic instructions with 16-Bit operand such as addword, subword, incword and decword. It also includes seven instructions with 16-Bit operand for logical operation as andword, orword, xorword, compword, swapword, and two rotate operations as rrword and rlword. These 11 instructions of 16-Bit operand are executed with 8-Bit controller with the help of C-ALU, maintaining the controller's original functionality. Thus, to carry out the operation through C-ALU a number of 16-Bit SFRs have been designed.

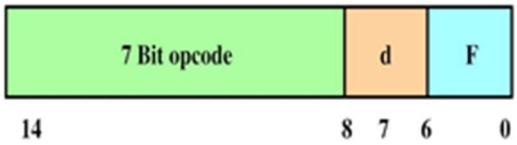

Figure 5.Proposed Format of Instruction Set

Table 1. Summary of Instructions for Proposed Controller

8-Bit Instruction Executed				16-Bit Instruction Executed		8 bit Multiplier	Total Instructions
Arithmetic	Logical	Rotate/ Shift	Bit-set Clear	Arithmetic	Logic		
8	7	6	2	4	7	1	35

Figure 5 shows proposed 15 bit instruction format for the modified controller. The 7 lower bits denoted by 'F' in instruction format were used to address the file register, the total 128 register file locations were addressed by these bits. The single bit d at 7th position decides the destination for the result of corresponding operation. If this bit is '1' then the result is stored in used file register else it stores the result in working register. The default result storage location is file register. The instructions are categorized according to the destined 8-Bit ALU or 16-Bit ALU, byte oriented, literal and control operations and CALL, GOTO instructions separately.

The CALU is intentionally designed to process 16-bit data operations and execute to 16-Bit instructions. This format is modified because, original controller having 14 bits (13-0) for IR register, is newly designed instruction register having 15 bits as (14-0). Without affecting the original functionality of the instruction set, there was no space for new instructions. So, one more-Bit is added in the MSB position. For 16-Bit operations, MSB-Bit is '1' and for 8-Bit operations, MSB-Bit is '0'.

Table 1.summaries the total number of instructions implemented in each instruction group of the controller. These are also classified according to the operand size as 8-Bit or 16-Bit.

The pre-existing set of 8-Bit instructions is implemented with minor change as this research deals with hybrid data types.

Table 2 below shows the 16-Bit instructions executed by co-operative arithmetic and logic unit. All the instructions are executed in single cycle by giving 16-Bit address, provided that data is available in data memory.

The MSB is kept '1' for the execution of 16-Bit instructions and '0' is for the execution of 8-Bit instructions. Of all the instructions, 'swapword', is different than others. It swaps the byte in register A and as in case of 8-Bit 'SWAP' instruction, swapping is with lower 4 bits i.e. nibble to the upper 4 bits. Due to 'addword' and 'subword', 16-Bit instructions, changes the flags as carry (C), digital carry (DC) and zero flag (Z). In remaining cases, after the execution of instructions, only carry and zero flag is affected.

Table 2. 16-Bit Instructions execution for C-ALU

Sr. No.	Mnemonic	Description	Cycles	15-Bit Opcode	Affected Flags

Table

1	addword	16-Bit addition of two words	1	100 0111 0001 0000	C,DC,Z
2	subword	16-Bit subtraction of two words	1	100 0010 0001 0000	C,DC,Z
3	andword	Logical AND operation	1	100 0101 0001 0000	Z
4	orword	Logical OR operation	1	100 0100 0001 0000	Z
5	xorword	Logical XOR operation	1	100 0110 0001 0000	Z
6	incword	Increment register A	1	100 1010 0001 0000	Z
7	decword	Decrement register A	1	100 0011 0001 0000	Z
8	swapword	Swap byte in reg. A	1	100 1110 0001 0000	-
9	rlword	Rotate left f through carry	1	100 1101 0001 0000	C
10	roword	Rotate right f through carry	1	100 1100 0001 0000	C
11	compword	Complement of A	1	100 1001 0001 0000	Z

3.Elaborates the detailed comparison of the hardware module presented in the data path of 64-Bit architecture and proposed controller. The implemented controller follows 8-Bit Harvard architecture in which data and program memory are having separate buses for accessing data and address whereas the 64-Bit controller follows MICA i.e. Modular Integrated Concept Architecture.

Table 3.Comparison of Features of Proposed Controller with 64-Bit Architecture

Parameter	**64 Bit Architecture**	**Proposed Controller**
Architecture	MICA	Harvard
Instructions	33	35
Instruction Reg.	32	15
Data Memory	64 byte	68 byte

Data Bus Width	64 bit	8 bit / 16 bit
Address Bus Width	32 bit	8 bit

The 64-Bit controller uses 32 bits address bus which can allow access to memory to 4GB, the data bus width is 64-Bit, which is used to exchange 64-bit data between memory, processing modules and internal registers. There are 33 different instructions present with 32-Bit instruction length. Thus, the size of the instruction register and the decoder module is 32-Bit wide.

In this proposed 8-Bit hybrid controller, designed with mixed 8-Bit and 16-Bit address bus, can allow access to memory up-to 256 bytes, which is very less as compared to 64-Bit controller. The data bus width 8-Bit and 16-Bit is used to exchange 8-Bit information between memory, 8-Bit data processing units, and internal 8-Bit registers, whereas, 16-Bit data bus is used to exchange data between word size register with 16-Bit C-ALU. In this controller, 35 different instructions have been presented with 15 bit instruction length. Thus, the size of the instruction register and the decoder module is 15 bit wide.

4. Results

This research paper uses latest tools to design, coding as well as implementing over the hardware platform. The PIC16F84 RISC controller design is intended to demonstrate a small and easy to use 8-Bit controller written in VHDL. The results are simulated for enhanced 8-Bit controller. Implementation of RISC core with module wise coding of VHDL using FPGA is carried out with Xilinx software 14.7 ISE. For the implementation of 8-Bit controller with C-ALU and multiplier, the selected hardware platform is the Spartan-7 device family XC7K160T-3FBG676E FPGA. The implemented controller shows performance improvement in terms of execution of a number of instructions and time required for execution, power improvement to perform 16-Bit operation. It also shows a good result in terms of execution time performance.

4.1 16-Bit Operations Performed by C-ALU

Figure 6.shows simulation output of 11 different 16-Bit operations performed with C-ALU. Each instruction is executed in less than 20 ns. The two operands are loaded in a [15:0] and b [15:0] register, the result register sum [15:0] is a 16-Bit result register which includes 16-Bit result with carry/borrow flag. For swap, rotate, compliment, increment and decrement operations are performed over the single operand a [15:0].

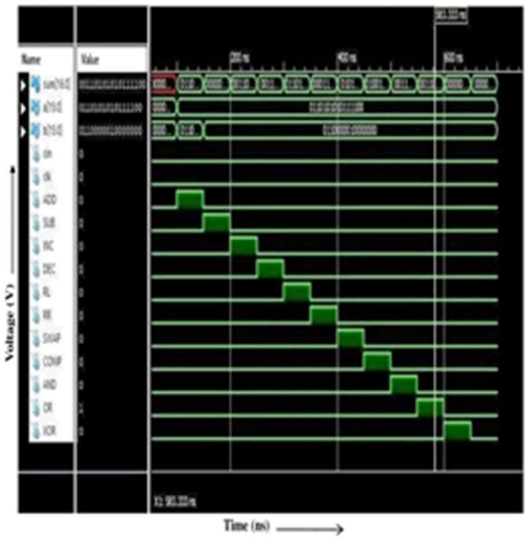

Figure 6.Simulation Output of 16-Bit C-ALU with 11, 16-Bit Instructions

4.2 Simulation Output of 16-Bit C-ALU and Multiplier

Figure 7, shows simulation output of 16-Bit C-ALU and 8-Bit multiplier. The operands are stored in two 16-Bit register namely a [15:0] and b [15:0], the output is available in sum [15:0] register. Operations are executed through the 8-Bit multiplier circuit. Multiply instruction completely executes in less than 28 ns whereas C-ALU operations are performed in less than 20ns.

Figure 7.Simulation Outputs of 16-Bit C-ALU and 8-Bit Multiplier

Table 4.Performance Analysis of Proposed 8-Bit Controller with C-ALU and Multiplier

Parameter Summarization		**16-Bit C-ALU**	**8-Bit Multiplier**	**8-Bit Controller With C-ALU and Multiplier**
No. of Slice Registers	Used	125	25	300
	Available	202800	202800	202800
	% Util.	0.062	0.012	0.14
	Used	53	24	99

Slice Logic Distribution	Available	25350	25350	25350
	% Util.	0.21	0.094	0.39
	Used	62	36	102
Bonded IoB	Available	400	400	400
	% Util.	15.5	9	25.5

5. Discussion

Table 4. Elaborate performance analysis of proposed 8-Bit controller with C-ALU and multiplier. The number of slice LUTs, slice and I/O blocks are required for implementing each individual design as well as combined design.

The 16-bit C-ALU implementation over the FPGA occupies only 125 slice registers out of two lakh of total slice registers which is more compact and efficient as compared to the 16-Bit ALU of 16-Bit controller. The 8-Bit multiplier design occupies 25 slice registers out of two lakh of total slice registers. Whereas, the combined design occupies the 300 slice registers space over the FPGA, which is smaller and efficient as compared to the 16-Bit controller.

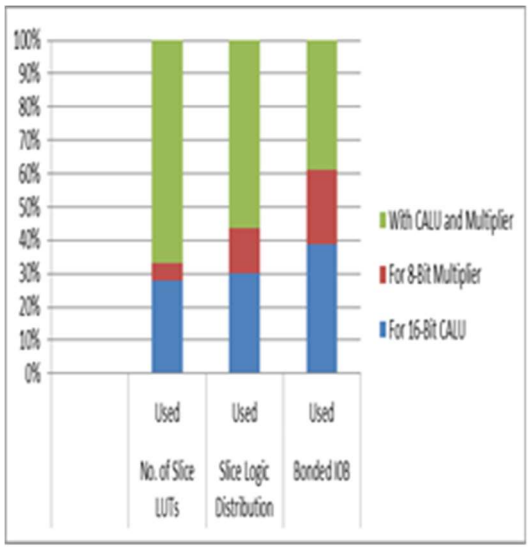

Figure 8.Graph of the Device Utilization

Figure 8 shows graph of device utilization for proposed 8-Bit controller with CALU and multiplier. The 16-bit CALU implementation over the FPGA occupies only 135 LUTs out of one lack of LUTs which is more compact and efficient as compared to the 16-Bit ALU of 16-Bit controller. The 8-Bit multiplier design occupies 155 LUTs out of one lack of LUTs. Whereas, the combined design occupies the 300 LUTs space over the FPGA, it is smaller and efficient as compared to the 16-Bit controller. The proposed hybrid 8-Bitcontroller implemented over the Spartan-7 Family device family XC7K160T-3FBG676E based FPGA board.

6. Conclusions

This paper presents the hybrid controller enhanced data path architecture, which is an improvement over the traditional 8-Bit controller in terms of operations and instruction execution speed. The implemented controller follows 8-Bit Harvard architecture. The instruction length is

modified from 14 bit to 15 bit instruction format, so as to incorporate 16-Bit Cooperative ALU inside the 8-Bit controller. For modifying these instruction formats, only one ZERO bit is introduced into MSB position of pre-existing 14 bit instruction format. While, newly introduced 15-Bit instruction format includes ONE as a MSB Bit. Thus, the instruction decoder now categorized the instruction into 14 bit or 15 bit according to status of MSB bit of instruction fetch from the instruction register.

The implementation of RISC controller with module wise coding of VHDL using FPGA is carried out with Xilinx software 14.7 ISE. Thus, the designed 15 bit instruction operation is verified using Xilinx 14.7 simulation tool. There is percentage saving in clock cycles as, 6 instructions are required for execution of 16-Bit operation on 8-Bit core but single instruction is required with enhanced controller. Maximum clock frequency achieved on the selected device is 31.15 MHz rather than 20 MHz of the original core. The average performance 71% is improved in multi-byte operational cases.

Compliance with ethical standards

Funding: No funding received.

Data Availability Statement:

The data used to support the findings of this study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

A. J. Salim., Nur Raihana Samsudin., SaniIrwan Md Salim and Yewguan Soo. 2012. Multiply-Accumulate Instruction Set Extension in a Soft-core RISC Processor", IEEE-ICSE, Proceedings, pp. 512-516.

A. J. Salim., S. I. M. Salim., N. R. Samsudin., and Y. Soo. 2013. Instruction Set Extension through Partial Customization of Low-End RISC Processor. Australian Journal of Basic and Applied Sciences. 7(6):678-687.

Amrut A. Purohit., Mohamed Riyaz Ahmed., R. Venkata S. Reddy. 2020. Design of Area Optimized Arithmetic and Logical Unit for Microcontroller. IEEE VLSI Device Circuit and System, Kolkata, (VLSI DCS), pp.335-339.

Antinio H. Zavala., Jorge Avante R., Quetzal coatl Duarte R.., J. David Valencia P. 2011. RISC-Based Architecture for Computer Hardware Introduction. IEEE 2011 3rd International Conference on Computer Research and Development. pp.17-21.

Deepak Kumar., K. Anusudha. 2013. RISC System Design in Xilinx. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. 2(4): 1406-1411.

E.Ayeh., K. Agbedanu., Y. Morita., O. Adamo., and P. Guturu. 2008. FPGA Implementation of an 8-bit Simple Processor. 2008. IEEE Region 5 Conference, pp. 1-6.

Florin M., Mihai C., Constantin S., Dan F. 2009. Implementation of a 32-bit RISC Microcontroller-FPGA prototype. 4th International Conference on Inter disciplinary in Education ICIE'09. Lithuania. pp. 193-199.

G. Challa Ram., D. Sudha Rani., Y. Rama Lakshmanna., K. Bala Sindhuri. 2016. Area Efficient Modified Vedic Multiplier, IEEE, International Conference on Circuit, Power and Computing Technologies, pp.1-5.

Jikku Jeemon. 2015. Low Power Pipelined 8-bit RISC Processor Design and Implementation on FPGA. International Conference on Control, Instrumentation, Communication and Computational Technologies, pp. 476-481.

Jingchuan Dong., Taiyong Wang., Bo Li., Zhe Liu., Zhigiang Yu. 2017. An FPGA-Based Low-Cost VLIW Floating-Point Processor for CNC Applications. Microprocessors and Microsystems, pp.1-24.

Julio Enrique Rodriguez Prieto., Edwar Jacinto Gomez., Fernando Martinez Santa. 2017. Implementation of an 8-Bit Softcore Microcontroller on Xilinx Spartan FPGA Family. Indian Journal of Science and Technology. 10(22):1-7.

K.Deergha Rao., Ch. Gangadhar., Praveen K. Korrai. 2016. FPGA Implementation of Complex Multiplier Using Minimum Delay Vedic Real Multiplier Architecture. IEEE, International Conference on Electrical, Computer and Electronics Engineering, pp.580-584.

Khoi-Nguyen LE-HUU., Anh-Vu., Thanh VU., Quoc-Minh., Vy LUU. 2014. A Proposed RISC Instruction Set Architecture for the MAC Unit of 32-bit VLIW DSP processer Core. International Conference on Computing, Management and Telecommunication. pp. 170-175.

Kunal Jadhav., Aditya Vibhute., Shyam Iyer., R. Dhanabal. 2015. Novel Vedic Mathematics Based ALU Using Application Specific Reversibility. IEEE, International Conference on Intelligent Systems and Control. pp. 1-5.

Mark G. Arnold. 2006. A RISC Processor with Redundant LNS Instructions. Proceedings of the 9th EUROMICRO Conference on Digital System Design, IEEE, pp. 1-8.

Mohammed F. Tolba., Ahmed H. Madian., Ahmed G. Radwan.2016. FPGA Realization of ALU for Mobile GPU. IEEE, International Conference on Advances in Computational Tools for Engineering Applications, pp.16-20.

Patricio Bulic., Zdenka Babic., Aleksej Avramovic. 2010. A Simple Pipelined Logarithmic Multiplier. IEEE, pp. 235-240.

Pravin S. Mane., Indra Gupta., M. K. Vasantha. 2006. Implementation of RISC Processor on FPGA. IEEE: Industrial Technology, 2006. ICIT 2006. IEEE International Conference, pp. 2096-2100.

R.Uma. 2012. Design and Performance Analysis of 8-bit RISC Processor Using Xilinx Tool. International Journal of Engineering Research and Applications, 2(2), pp. 53-58.

Rajul Bansal., Abhijit Karmakar. 2017. Efficient Integration of Coprocessor in LEON3 Processor Pipeline for System-On-Chip Design. Microprocessors and Microsystems, pp.56-75.

Ramandeep Kaur., Anuj. 2014. 8 Bit RISC Processor Using Verilog HDL. Journal of Engineering Research and Applications. 4 (3): 417-422.

Ryszard Gal., Adam Golda., Maciej Frankiewicz., Andrzej Kos. 2011. FPGA Implementation of 8-bit RISC Microcontroller for Embedded Systems. 18th International Conference on Mixed Design of Integrated Circuits and Systems, June 16-18, pp. 323-328.

Sergio F. Johann., Matheus T. Moreia., Leandro S. Hech., Noy L.V. Calazans. 2016. A Processor for IOT Applications: An Assessment of Design Space and Trade-Offs. Microprocessors and Microsystems, pp.1-9.

Shikha Khurana., Kanika kaur. 2012. Implementation of ALU using FPGA. International Journal of Emerging Trends and Technology in Computer Science (IJETTCS). 1(2), pp. 146-149.

Tatsuya Suto., Kenji Ichijo., Yoshio Yoshioka. 2010. Design and Evaluation of Brust mode Asynchronous 8-bit Microprocessor Using Standard FPGA Development System. 25th International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 172-179.

Wael M EI-Medany., Khalid A. AI- Kooheji. 2008. Design and Implementation of a 32-bit RISC Processor on Xilinx FPGA. pp.1- 4.