© Universiti Tun Hussein Onn Malaysia Publisher's Office

JEVA

Journal homepage: http://publisher.uthm.edu.my/ojs/index.php/jeva
e-ISSN : 2716-6074

Journal of Electronic Voltage and Application

Design of Multiplication and Division Operation for 16 Bit Arithmetic Logic Unit (ALU)

Muhammad Ikmal Mohd Taib[1], Muhammad Najmi Zikry Nazri[1], Fatin Syazana Khairol Amali[1], Muhamad Amirul Mohd Yusof @Md Jusoh[1], Abd Kadir Mahamad[1*], Sharifah Saon[1]

[1]Faculty of Electrical and Electronic Engineering,
Universiti Tun Hussein Onn, Parit Raja, Batu Pahat, 86400, Johor, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/jeva.2020.01.02.006
Received 01 October 2020; Accepted 21 December 2020; Available online 28 December 2020

Abstract: Arithmetic Logic Unit (ALU) is one of the most crucial components of an embedded system and has been used in many devices such as cell phones, calculator, computers, and many other applications. It performs all the arithmetic and logical operations such as addition, subtraction, logical AND and OR. An ALU is a multi-functional circuit that conditionally performs one of several possible functions of two operands A and B depending on control inputs. It is nevertheless the main performer of any computing device. The objective of this project is to design ALU 16-bit using VHDL. The Altera Quartus II software is used as the tool to create the designed operation of multiplication and division. The simulation results show that the proposed ALU design successfully perform the operation of multiplication and division of 16 bits operation.

Keywords: ALU, VHDL, 16 bit, Altera Quartus II

1. Introduction

In computing, an arithmetic and logic unit (ALU) is a digital circuit that performs integer arithmetic and logical operations [1]. The Arithmetic Logic Unit (ALU) is a basic block of the central processing unit of a computer, and even the simplest microprocessors that performs the work of maintaining the timers. The processors accommodate very powerful and very complex ALUs inside modern Central Processing Unit (CPUs) and Graphics Processing Units (GPUs), as the single component that may contain a few ALUs [2].

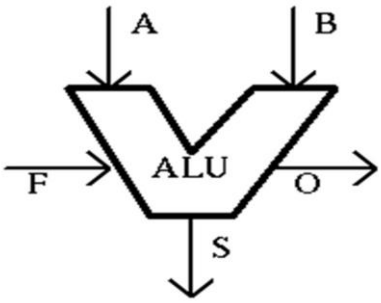

Fig. 1 - Arithmetic logic unit

*Corresponding author: kadir@uthm.edu.my

Fig. 1 is an ALU that has two inputs called operands, and a code which is call as OPCODE. The specified code will operatize performed on the operands. ALU also has a result output which is the result of the operation on the operands [3].

Arithmetic shifts can be useful as efficient ways of performing multiplication or division of signed integers by powers of two. In multiplying 2n, shifting left by n bits on signed binary number or unsigned binary number will affect the output. While, in dividing it by 2n, this requires to shifting left by n bits on signed binary number or unsigned binary number [4].

In performing division and multiplication of integers by power of two, logical shift is used. The process is in shifting left by k bits on a binary number is equivalent to multiplying it by 2k. The same goes to shifting right by k bits on a binary number is equivalent to dividing it by 2k. For example, consider the binary number is 0001 0111 [2][5].

For multiplication, it is an algorithm where N bit is multiplicand by N bit multiplier. In ALU 16 bit, a multiplier can multiply an 8-bit number with another 8-bit number, and this will result in a 16-bit product. Here, the column bypass technique is used where this will reduce power consumption. In calculating each bit in a product, the compressor module is used in which this will help reduce the power. There are two methods to obtain power reduction which are column bypass or compressor.

2. Design of 16-bit ALU

This project proposes the design operation of multiplication and division using ALU 16 bit. This ALU 16 bit consist of two operations which are multiplication and division operation. This operation acts as calculator to calculate the value using multiplication or division depending on the control input. This design can be operated with single operation in one time. The inputs given to an ALU are the data to be operated which is called as operands, and the ALU's output is the result of the performed operation.

2.1 Flow Chart

The flow chart for the proposed project is shown in Fig. 2. Once the input data is given, the operation chooses either multiply or division. 16-bit operation for multiplication starts from “0000” to “0111” and for division, it starts from “1000” to “1111”. Then output result of the operation for the collected data is displayed.

Meanwhile, Table 1 shows the data of the operation. “0000” to “0111” represent multiplication operation and “1000” to “1111” represent division operation. The value for the input can be any value in binary to perform the operation.

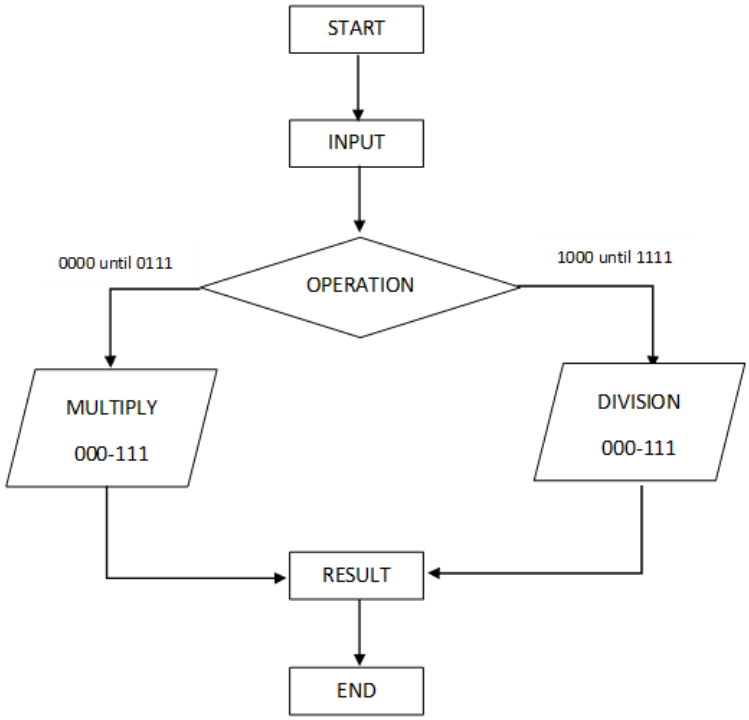

Fig. 2 - Flow of Proposed Project

Table 1 - The Truth Table of operation

16 bits	Operation	
0000	A X 000	Multiplication
0001	A X 001	
0010	A X 010	
0011	A X 011	
0100	A X 100	
0101	A X 101	
0110	A X 110	
0111	A X 111	
1000	A ÷ 000	Division
1001	A ÷ 001	
1010	A ÷ 010	
1011	A ÷ 011	
1100	A ÷ 100	
1101	A ÷ 101	
1110	A ÷ 110	
1111	A ÷ 111	

2.2 Quartus II Software

The Altera Quartus II software is used as the tool to create the design of operation for multiplication and division using ALU 16 bit. This software is used to create the code to perform the operation. VHDL supports to constructing the coding to perform the output result of operation. The circuit shown in the RTL Viewer is based on the coding that is constructed in VHDL. All the data collected from the source are recorded in the event log and displayed as the timing diagram.

The VHDL coding for the ALU 16-bit calculator is shown in Fig. 3. On the module part, it declares the input and output for the program. The input has two parameters; input A and ALU selection. The input A has 8-bit and the Alu_Sel has 4-bit that contain 16-bit ALU. Next for the output, it has 2 values; ALU 12-bit output and carryout. After that, it needs to be declared as ‘’reg’’ because the order of inputs to a function dictates how it should be wired up when it is called.

In this project, it is needed to declare ‘reg’ for use the always block. For wire, it has nine bit that is from zero to eight (wire [8:0] tmp). For the ALU_Output, it is assigned to equal with the ALU_result to synchronize it with ALU operation. This project also declares the always @(*) block. This is because, assign inside of an always block, it must be declared as a reg. A reg signal is typically the output of a flip flop, a latch, or combinational logic that appears in an always @(*) block.

This project has two operations; multiplication and division operation. Input A gave the input that and the ALU_Sel, performed the operation of multiplication and division based on the selection that are have in ALU 16-bit selector. For both operations, there exist eight operations for multiplication and eight operations for division, and there is a need to choose from 0000 to 1111 in ALU 16-bit operation. For example, in the case of multiplication by 7, this refers to the ALU_Result = A*3'b111 that contains in 0111 ALU 16-bit operation. For division operation, it is needed to choose the ALU-16 bit operation from 1000 to 1111. To prevent from error, this program needs to declare default: ALU_Result = 8'b00000000. This is important to show for any error for the output.

```
module muldiv_calculator(
          input [7:0] A,  // ALU 8-bit Inputs
          input [3:0] ALU_Sel,// ALU Selection
          output [7:0] ALU_Out, // ALU 8-bit Output
          output CarryOut // Carry Out Flag
    );
              reg [7:0] ALU_Result;
              wire [8:0] tmp;
              assign ALU_Out = ALU_Result; // ALU out
              assign CarryOut = tmp[8]; // Carryout flag
              always @(*)
              begin
                  case(ALU_Sel)

 //Multiplication operation
4'b0000: // multiply 0
ALU_Result = A*3'b000 ;
4'b0001: // multiply 1
ALU_Result = A*3'b001 ;
4'b0010: // multiply 2
ALU_Result = A*3'b010;
4'b0011: // multiply 3
ALU_Result = A*3'b011;
4'b0100: // multiply 4
ALU_Result = A*3'b100;
4'b0101: // multiply 5
ALU_Result = A*3'b101;
4'b0110: // multiply 6
ALU_Result = A*3'b110;
4'b0111: // multiply 7
ALU_Result = A*3'b111;

//Division operation
4'b1000: //  divide 0
ALU_Result = A/3'b000;
4'b1001: //  divide 1
ALU_Result = A/3'b001;
4'b1010: //   divide 2
ALU_Result = A/3'b010;
4'b1011: //   divide 3
ALU_Result = A/3'b011;
4'b1100: //   divide 4
ALU_Result = A/3'b100;
4'b1101: //   divide 5
ALU_Result = A/3'b101;
4'b1110: //   divide 6
ALU_Result = A/3'b110;
4'b1111: //   divide 7
ALU_Result = A/3'b111;
default: ALU_Result = 8'b00000000;
        endcase
        end

endmodule
```

Fig. 3 - Coding for 16-bit ALU operation

2.3 Register Transfer Level

Fig. 4 shows the Register Transfer Level (RTL) diagram for this program. The component in this RTL diagram is 4 division operation, 4 add operation and 12 multipliers. For multiplication operation, the 4 add operation has been used to perform it.

From Fig 5, it can be seen there exist 2 inputs that involve in this program that are 8-bit for A input add ALU_Sel that contains ALU 16-bit operation for multiplication and division. This ALU operated on 16-bit input. The ALU is connected to each multiplier to perform their operation. The input A is then divided for addition and division operation to perform the input that has been called. The Add and Div. operation are shown in Fig. 6.

Because the output for this program has 12-bit, the 12 multiplexers have been used in this program to perform that 12-bit. These multiplexers are then combined to perform the output for ALU 16-bit. Fig. 7 shows the output of the ALU.

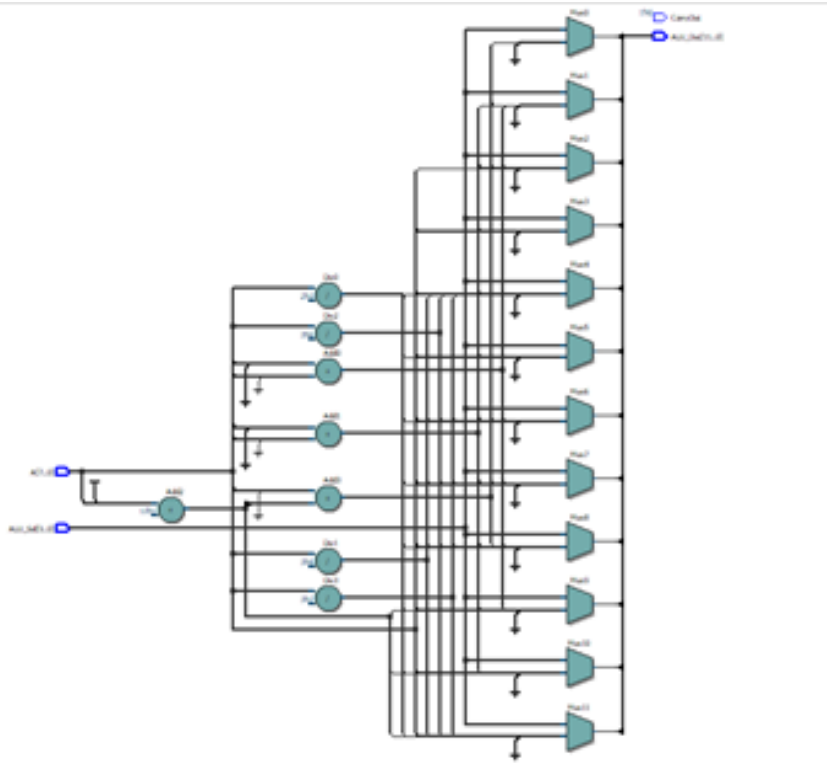

Fig. 4 - Register Transfer Level (RTL) diagram

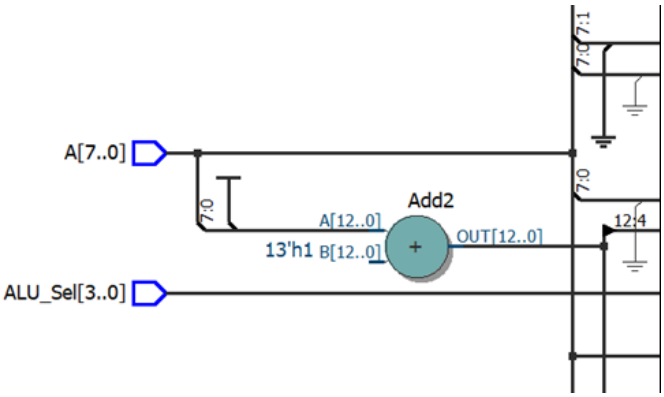

Fig 5 - Input A and ALU_Sel

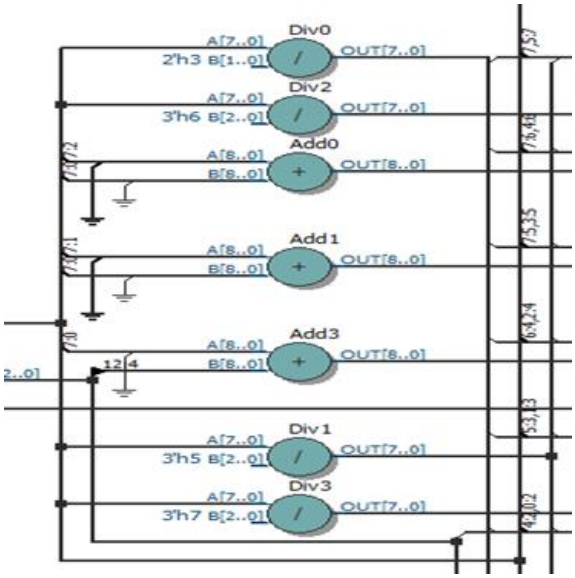

Fig. 6 - Add and Div. has been used to perform the input value

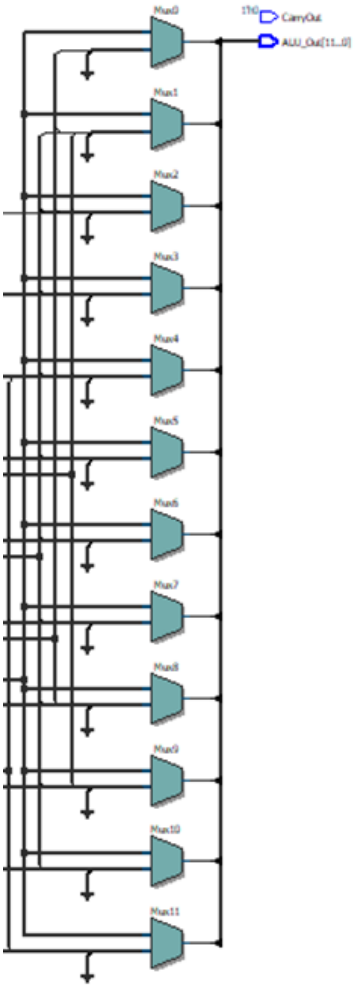

Fig. 7 - Output of ALU

3. Result of Experiment

3.1 Timing Diagram of Multiplication operation

Fig. 8 shows the simulation result of binary number when the input is 1011 and the operation ALU-16 bit that is chosen is 0001. The result is 1011 in binary.

Fig. 8 - The result on Quartus II when A=1011 and Alu_Sel=0001

Fig. 9 shows the simulation result of binary number when the input is 11011 and the operation ALU-16 bit that is chosen is 0100. The result is 1101100 in binary.

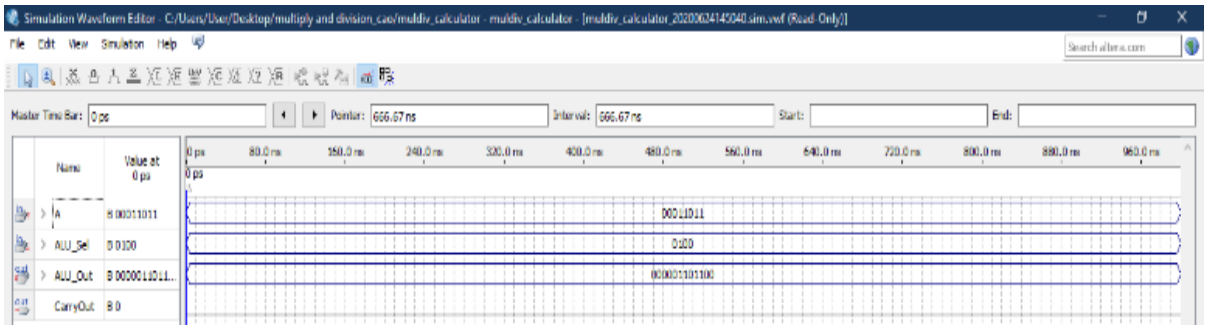

Fig. 9 - The result on Quartus II when A=11011 and Alu_Sel=0100

Fig. 10 shows the simulation result of binary number when the input is 1011011 and the operation ALU-16 bit that is chosen is 0110. The result is 1000100010 in binary.

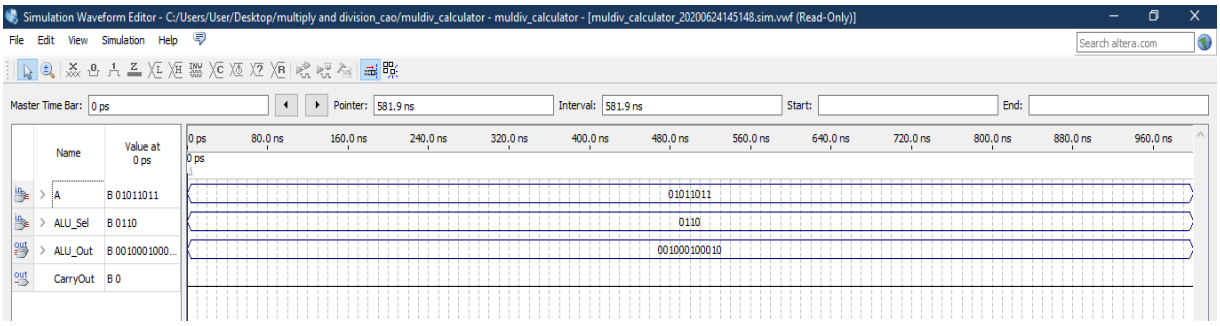

Fig. 10 - The result on Quartus II when A=1011011 and Alu_Sel=0110

All the results shown using simulation in Quartus II are same as the calculation made in Microsoft Excel as in Table 2. Table 2 shows the result obtain for multiplication operation with the input of A and multiplication with the output which is ALU results.

Table 2 - Result of multiplication by using Microsoft Excel

Input		Output
A	Operation (Multiplication)	ALU Result
1011	1	1011
11011	100	1101100
1011011	110	1000100010

3.2 Timing Diagram Division operation

Fig. 11 shows the simulation result of binary number when the input is 11110 and the operation ALU-16 bit that have been chosen which is 1010. The result is 1111 in binary.

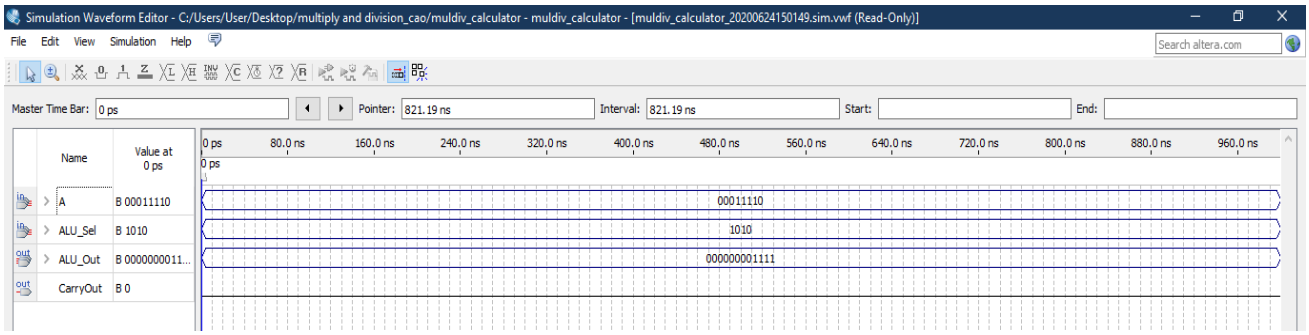

Fig. 11 - The result on Quartus II when A=11110 and Alu_Sel=1010

Fig. 12 shows the simulation result of binary number when the input is 1010 and the operation ALU-16 bit that been chosen is 1110. The result is 0001 in binary.

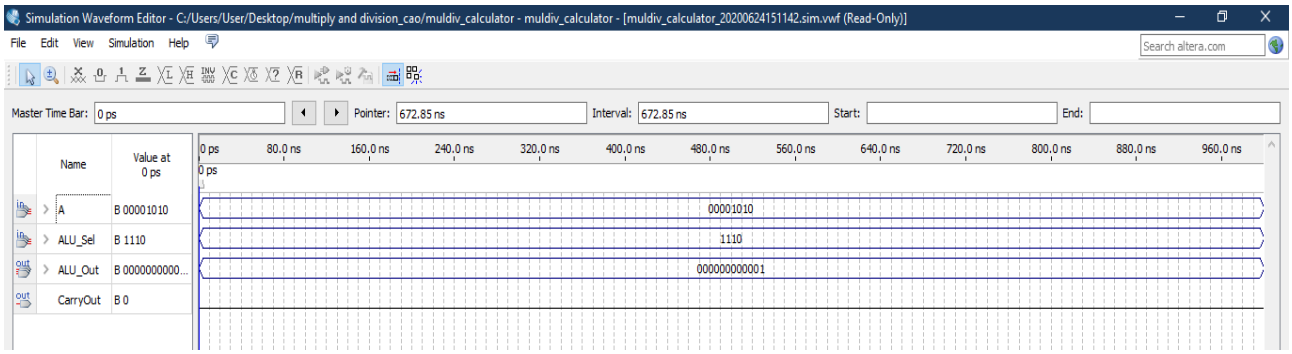

Fig. 12 - The result on Quartus II when A=1010 and Alu_Sel=1110

Fig. 13 shows the simulation result of binary number when the input is 111011 and the operation ALU-16 bit that been chosen is 1010. The result is 11101 in binary.

All the results shown using simulation in Quartus II are the same as the calculation made in Microsoft Excel as illustrated in Table 3.

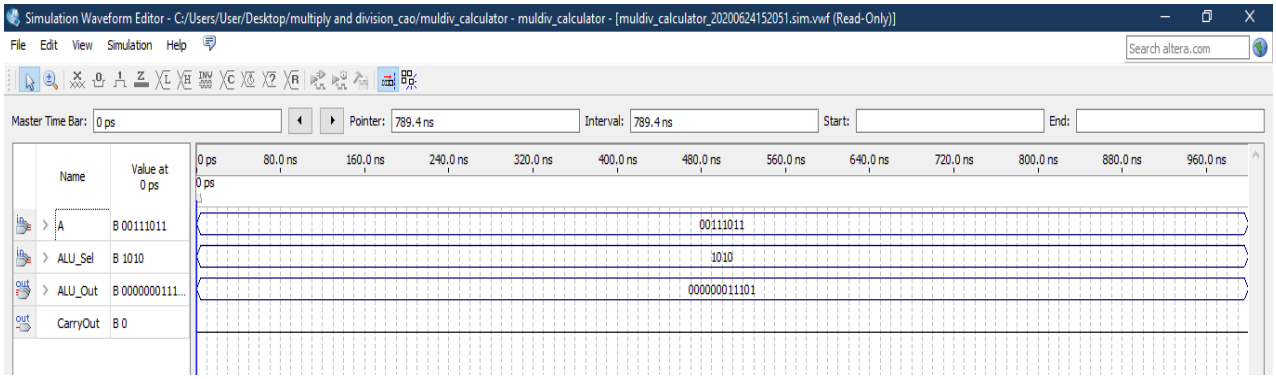

Fig. 13 - The result on Quartus II when A=111011 and Alu_Sel=1010

Table 3 - Result of division by using Microsoft Excel

Input		Output
A	Operation (Division)	ALU Result
11110	10	1111
1010	110	1
111011	10	11101

4. Conclusion

To sum up, Arithmetic Logic Unit (ALU) perform an arithmetic and logical operation where this project uses division and multiplication. Arithmetic Logic Unit (ALU) 16-Bit Division and Multiplication can be designed using Altera Quartus II software. Operation of division and multiplication is a kind of calculator that can calculate for both input and output of 16-bit operation. Using Altera Quartus II software code can be create based on the project objective. Simulation results prove that this design was successfully performed as expected.

Acknowledgement

This work was partially supported by the Faculty of Electrical and Electronic Engineering (FKEE), and Research Management Center (RMC), Universiti Tun Hussein Onn Malaysia.

References

[1] Zandbergen, P. (2015). Arithmetic Logic Unit (ALU): Definition, Design & Function. Lessons and Online Courses, Available:https://study.com/academy/lesson/arithmetic-logic-unit-alu-definition-design function.html

[2] Bharath, R. M. (2016). Design, Analysis, and Synthesis of a 16 bit Arithmetic Logic Unit using Reversible Logic Gates.Available:http://search.proquest.com/openview/09e15f3998b446b987d6e42859e3dc45/1?pqorigsite=gscholar&cbl=18750&diss=y

[3] IAY0340 Labs. (2020). Arithmetic Logic Unit (ALU). Available: http://ati.ttu.ee/IAY0340/labs/Tutorials/SystemC/ALU.html

[4] Thapliyal, H. Srinivas, M.B (2005). Novel design and reversible logic synthesis of multiplexer based full adder and multipliers. 48th Midwest Symposium on Circuits and Systems, 1593-1596

[5] Rakhi, N. Neeraj, K.S. (2018). Resource Utilization Optimization with Design Alternatives in FPGA based Arithmetic Logic Unit Architectures. Procedia Computer Science, 132, 843-848