Designing a 4-bit ALU to perform 4 different operations

1st Kazi Md. Al-Wakil
Computer Science and Engineering
BRAC University
19301051

2nd Sababa Rahman Meem
Computer Science and Engineering
BRAC University
19101569

3rd Md. Galib Hasan Mim
Computer Science and Engineering
BRAC University
19301094

4th Masiat Hasin Rodoshi
Computer Science and Engineering
BRAC University
19201089

5th Konika Islam
Computer Science and Engineering
BRAC University
17201007

***Abstract*—A 4-bit ALU (Arithmetic Logic Unit) that is capable of performing four different arithmetic operations has been designed. The ALU has been implemented using Verilog HDL in Quartus. It takes two 4-bit binary numbers and an opcode as input. Based on the opcode, it performs different bitwise arithmetic operations. Such as, XNOR, NAND, ADD, SUB operations. The outputs are also a 4-bit binary number and Some flags used to identify the nature of the number. The flags are, ZF (Zero flag, Logical High when all the bits are zero), SF (Sign flag, when the MSB bit is 1, the flag is logical high) and CF (Carry Flag, occurs when there is a carry bit in the output). The correctness of these operations has been tested by using a timing diagram. The software used in this project is Quartus —— 8.1.**

I. Introduction

An ALU (Arithmetic Logic Unit) is a part of the Central Processing Unit that carries out arithmetic and logical operations. The assigned task was to design one such ALU that performs four different arithmetic and logical operations which are specified as XNOR, SUB, NAND and ADD. It has been implemented in a software named Quartus using Verilog which is a HDL (Hardware Specification Language) used to model electronic systems. This ALU takes two 4-bit binary numbers and an opcode as input. Each opcode refers to a distinct operation. For example, in our project specification, opcode 001 refers to XNOR operation. The ALU performs the specified operation on the binary numbers in a bit wise manner using states. For every opcode, the next state from the initial reset state is a specific state where the operation identified by the mentioned opcode is carried out. Within that state, it is possible to travel to more inner states to accomplish the bit wise operation. As long as the specific opcode is given as input, that operation is carried out and when it is over, the next state becomes the reset state. During this time, some flags aside from the result are also updated. These flags are 0 of logical low by default but become 1 or logical high as soon as the result meets specific conditions. The zero flag, sign flag and carry flag are shown as output along with the result of the operation which is a 4-bit binary number. The correctness of the system has been verified by using a timing diagram and testing every operation with a set of inputs.

In the following segment, the state diagram of the system and the timing diagram used for verification have been presented. Detailed description of the workflow has also been provided.

II. Operation

A. State Diagram and Timing Diagram

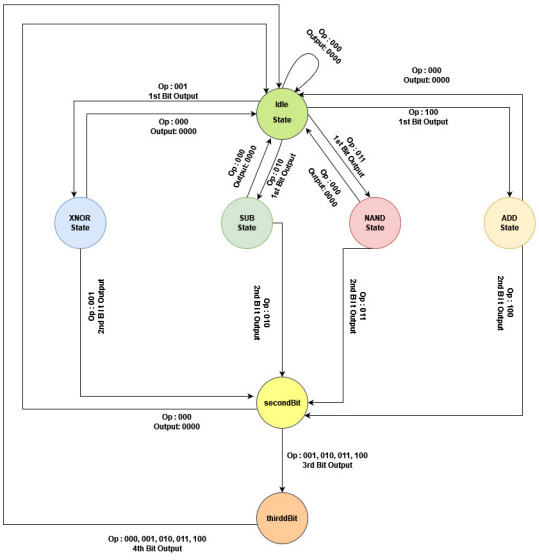

Fig. 1. 4-bit ALU State Diagram

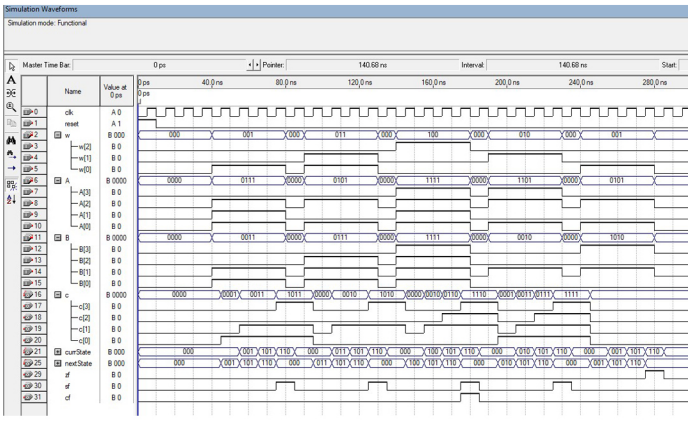

Fig. 2. Timing Diagram

B. Timing Diagram Discussion

In the timing diagram, we can see five different operations to show results of 4 different opcodes and 3 flags. The opcodes are: 000 : Idle State, 001 : XNOR, 010 : SUB, 011 : NAND, 100 : ADD

The flags are: SF : Sign Flag (High when Output MSB bit is 1), ZF: Zero Flag (High when Output is Zero), CF: Carry Flag(High when output carry/borrow is 1)

First Case: OP code : 001. So, XNOR operation will be performed. Input A: 0111 Input B: 0011

TABLE I
XNOR TRUTH TABLE

B	A	Output
0	0	1
0	1	0
1	0	0
1	1	1

From the truth table we can see, only when the both inputs are same, only then the output is 1 otherwise, the output is 0. We can verify this from the above timing diagram too. For the given input, output is 1011. The timing diagram matches the output from the truth table. The output is giving its result by doing bitwise operation. Also, as MSB of the output is '1', sign flag, SF is '1'. Lastly, after performing the operation, the output retains its value until a new opcode is in the input.

Second Case: OP code : 011. So, NAND operation will be performed. Input A: 0101 Input B: 0111

From the timing diagram we can see that the output is 1010. Only when both inputs are 1, then the output is '0'. Else, the output is '1' . Also, as MSB of the output is '1', sign flag, SF is '1'.

TABLE II
NAND TRUTH TABLE

B	A	Output
0	0	1
0	1	1
1	0	1
1	1	0

Third Case: OP code : 100. So, an ADD operation will be performed. Input A: 1111 Input B: 1111

From the timing diagram we can see that the output is 1110 and the carry flag, CF is '1'. The result matches with the truth table. Also, as MSB of the output is '1', sign flag, SF is '1'.

TABLE III
ADD TRUTH TABLE

B	A	Output
0	0	0
0	1	1
1	0	1
1	1	0(Carry 1)

Fourth Case: OP code : 010. So, SUB operation will be performed. Input A: 1101 (-5) Input B: 0010 (2) Input A - Input B = -5 - 2 = -7 Output should be : 1111 (-7)

From the timing diagram we can see that the output is 1111 where MSB 1 defines that the output is a negative number and the value is the rest of the 3 bits (111). The value we got from the timing diagram matches with the actual result. Also, as MSB of the output is '1', sign flag, SF is '1'.

TABLE IV
SUB TRUTH TABLE

B	A	Output
0	0	0
0	1	1(Borrow 1)
1	0	1
1	1	0

Fifth Case: OP code : 001. So, XNOR operation will be performed. Input A: 0101 Input B: 1010

According to the timing diagram, the output should be zero as no input pair is either '1,1' or '0,0'. So, the output is '0'. We get zero from our timing diagram for this case. As well as the Zero flag, ZF is also '1' because all the bits of output are Zero.

TABLE V
XNOR TRUTH TABLE

B	A	Output
0	0	1
0	1	0
1	0	0
1	1	1

III. Conclusion

In conclusion, we can say that the 4-bit ALU (Arithmetic Logic Unit) system can operate on 4 different opcodes, producing correct results and also assign flag values where the respective condition is True.