ATLANTIS PRESS
Advances in Computer Science Research, volume 74
2nd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2017)

A novel design method of a simple ALU based on PROTEUS

Binbin Chen[a, *]
School of Automation, North China Electric Power University, Baoding 071000, China.
[a]zj_lh_cbb@163.com

Keywords: Design method, ALU, PROTEUS.

Abstract. Nowadays, computer has becoming necessary to everyone, so the designs of the computer's parts are vital, in which ALU matters. There may be a lot of tradition methods on designing an ALU by different tools now, but these methods are mostly complex and uneasy-understanding to most beginners. In this paper, we propose a new way to design a simple ALU base on PROTEUS. Briefly speaking, we split the whole part into several parts, and use a more understanding approach to present the total design structure. The results show that our method, using a simple and more pellucid approach, also has the traditional functions the normal ALU designed by others contains. And in the end, we put out a total conclusion on our design method.

1. Introduction

In this paper, we consider the ALU design method in an easy-understanding and simple way. Firstly, we need to learn about the ALU. The computer system includes the data input part, the storage part, the operation part and the output part. From the perspective of the composition of the computer system, computer system includes two parts: architecture - stored program (important: the Von Neumann) [1] and software part (including language support, resource management and application software, etc.). The Central Processing Unit (CPU, Central Processing Unit), one of the main devices of the computer, functions mainly to interpret computer instructions and to process data in computer software. The central processing Unit (CPU) is the core component of the arithmetic logic Unit, the arithmetic logic Unit, and all the central processing units. The computational and logic unit is a combinatorial logic circuit that can implement multiple sets of arithmetic and logical operations, called ALU.

2. Analysis

The basic operations of the arithmetic logic unit include addition, subtraction, multiplication, division, and other logical operations such as, or, non-, or otherwise, and shift, comparison, and transfer operations. When the computer runs, the operation and operation of the arithmetic logic unit is determined by the controller. The data processed by the arithmetic logic unit comes from memory;

The resulting data is usually sent back to storage or temporarily stored in an arithmetic logic unit. Data arithmetic is the processing object of data, so the data length and the computer data representation method have great influence on the performance of the arithmetic. How many operations and operational speeds can be performed by the operational arithmetic can signal the ability of the operator to be strong, even the ability of the computer itself. The basic operation of the arithmetic unit is addition. The sum of a number is equal to the simple transfer of this number. To sum up a number of code, add up to another number, or subtract the previous number from the last number. Subtract the two Numbers and compare them to size. The left and right displacement is the basic operation of the arithmetic unit. In the number of symbols, the symbol does not move and only moves the data bit, which is called arithmetic shift. If the data is moved along with all the bits of the symbol, it is called logical shift. The logical shift of the highest and lowest links of the data is called cyclic shift. The logic operation of the arithmetic unit can place two data in place with, or, different or, and the data of each of you. Multiplication and division are more complicated. Many computer calculators can do this directly. The multiplication operation is based on addition operation, and the multiplicative product is partially

accumulated by one or several decoding control of the multiplier. In addition to the rule, which is often based on multiplication, the number of factors multiplied by the divisor is approximately 1, and these factors are multiplied by the quotient. Computer programs that do not perform multiplication and division hardware can be multiplied and removed, but at a much slower rate. Some of the arithmetic machines can also carry out the maximum number of Numbers in a batch of Numbers, perform the same operation for a batch of data continuously, and take the square root and other complex operations.

3. Different Parts Designs of Alu

In our designs, this ALU consists of seven parts: clock module, instruction module, IR module, storage module, operation module, instruction judgment module and display module. These modules are connected directly or indirectly through the bus. We concentrate on the early six parts. [2]

3.1 Clock Module.

This module includes reset, clock and count. The reset button returns the instruction to the initial state, and the clock key generates the clock signal CP pulse for the entire ALU, and the counter (74LS161) coordinates the CP pulse to generate the count cycle.

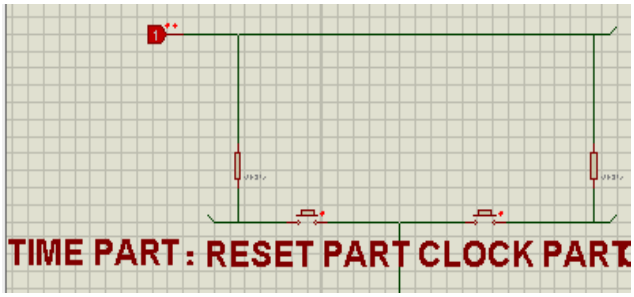

Fig. 1 Design map on clock module

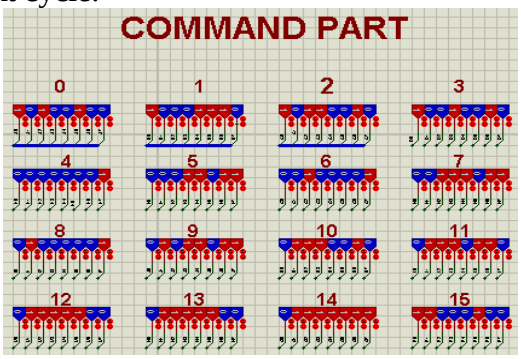

Fig. 2 Design map on instruction module

3.2 Instruction Module.

This module is used to edit the command statement to perform the corresponding functions. The maximum number of commands that this ALU can do is 16-bit, not super, but not all. The data is eventually uploaded to the instruction bus.

3.3 Ir Module.

At the core of this section is a register that is used to save the program's instructions from 8 to 8, which is called the program register because of the special purpose of this register. The input port of IR reads data from the instruction bus. The export of IR transfers data to the data bus to supply subsequent storage modules.

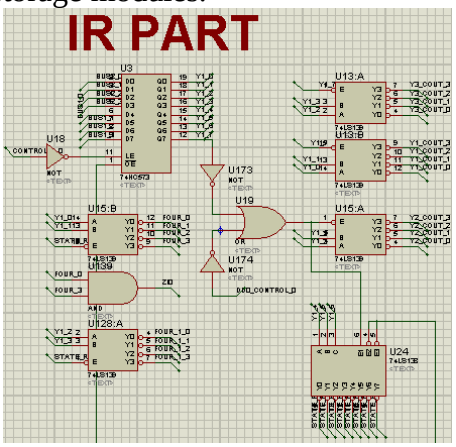

Fig. 3 Design map on IR module

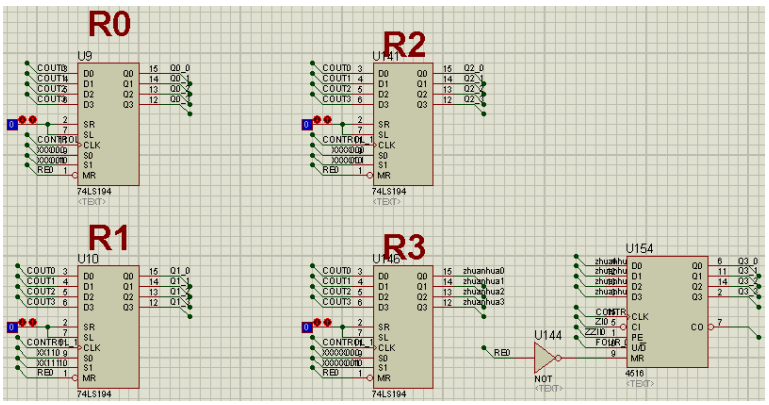

Fig. 4 Design map on storage module

3.4 Storage Module.

Storage module is composed of four registers, in order to realize the function of the displacement and the deceleration since, there are three in the four registers is 74 ls194, in order to realize the shift, and there's a 4516, in order to realize from the decrement. The data input port of the register data bus can read data from this. Its export is performed by logic transformation, access operation module, and arithmetic or logic operation.

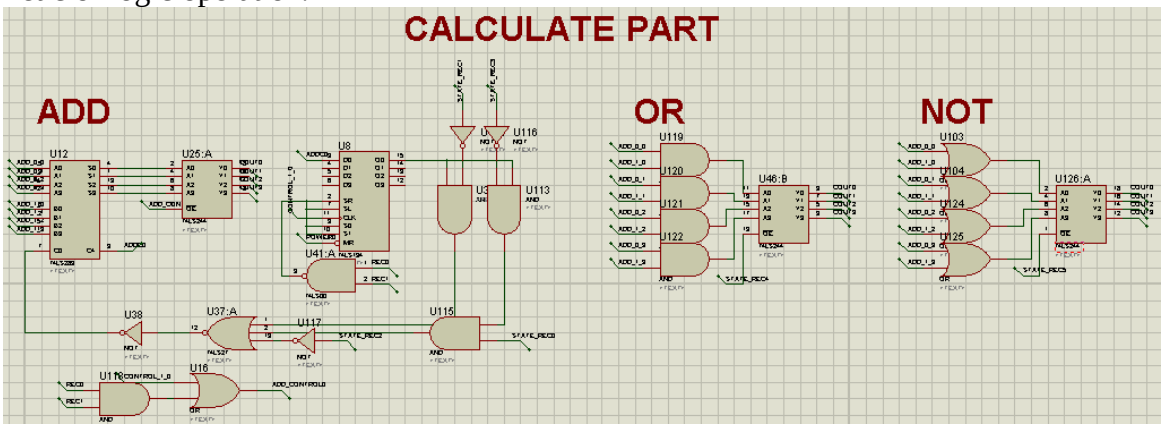

Fig. 5 Design map on operation module

3.5 Operation Module.

There are four operations in this module: the addition and subtraction of arithmetic, logic and or, the addition and subtraction are divided into the carry and the carry. Regardless of what kind of operation, the data input port, all access storage module, the output and the output back storage module, which registers specific back depends on the instruction of writing, this section in part 3.

3.6 Instruction Judgment Module.

The essence of this part is to determine the sequence of all these modules using the relationship of various combinatorial logic, and control each part of ALU so that various functions can be realized accurately.

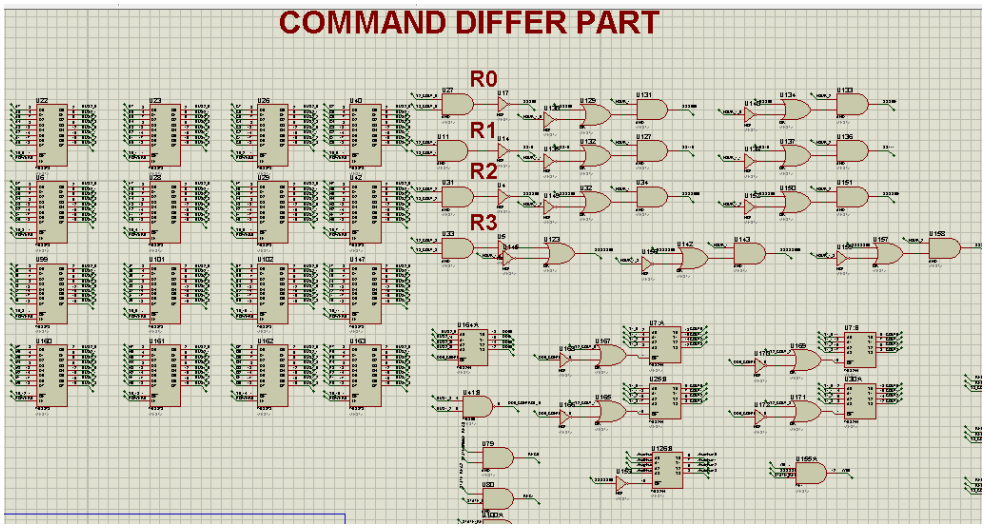

Fig. 6 Design map on instruction judgment module

4. Operation Instrument

This ALU has the following nine instructions in form and format:

1) 10ddxxxx: MOV *dd*, *xxxx*: This instruction is the register that is stored as a number of instructions to store the data *XXXX* to dd.

2) 0000ssdd: ADD ss, *dd*: The instruction is a no-carry addition instruction, and the number of the registers represented by the *ss* is not added to the number in the register represented by *dd*, and the result is stored back to the register of the *ss* representative.

3) 0001ssdd: ADC ss, *dd*: The instruction is an adjoin command with a carry, and the number of registers represented by the *ss* is added to the number in the register represented by *dd*, and the result is stored back to the register of the *ss* representative.

4) 0010ssdd: RED ss, *dd*: The instruction is a subtraction instruction without a carry, and the number of the registers represented by the *ss* is less than the number of the registers represented in *dd*, and the result is stored back to the register of the *ss* representative.

5) 0011ssdd: REC *ss*, *dd*: The instruction is a subtraction instruction with a carry, and the number of registers represented by the ss is reduced with the number of digits in the register represented by *dd*, and the result is stored back to the register of the *ss* representative.

6) 0100ssdd: AND ss, *dd*: The instruction is the logical "and" (&) operation instruction, and the number of the registers represented by the *ss* represents the number in the register represented by *dd*, and the result is stored back to the register of the *ss* representative.

7) 0101ssdd: OR *ss*, *dd*: The instruction is the logic "or" (|) operation instruction, and the number of the registers represented by the *ss* is represented in the register represented by *dd*, and the result is stored back to the register of the *ss* representative.

8) 0111ssdd: DTC *ss*, *dd*: The instructions is a data transformation, and the *ss* to represent four basic on the number of registers in the change, including the added, the decrement, left and right (the original data has changed), of which only shift R0, R1, R2, shift the vacant position to R0, R3 only since the decrement.

9) 1100xxxx: JUMP xxxx: This directive is a jump instruction that terminates the existing instructions and jumps to the instructions represented by *XXXX*, which can be executed or not executed.

5. Summary

This ALU is built in PROTEUS platform, the structure is more complex, but the content is relatively comprehensive. The product has basically possessed several functions of general ALU: arithmetic operation, logic operation, data shift, data counting, and instruction jump. According to the standard of instruction, after many times of testing and commissioning, test results broadly in line with the expected results, but we also found some inevitable problems. When debugging the ALU, we found the following bugs. [3]After carefully checking the simulation circuit, not timing has not been found, may be the design itself. It is inevitable, so briefly explained in the field, and remind reader's attention to avoid these points:

1) the register R0, R1 and R2 have only the function of shift and storage, and R3 only has the function of self - decreasing and storage, and cannot mix the two together instructions, otherwise the result will not be correct;

2) When the jump instruction is executed, it is necessary to wait for a button again, which is the instruction after the jump. Do not think that the jump is finished after the first time.

3) After the execution of the register of R3 count instruction, if there are other operations, the result can't back into the R3, otherwise the result error, but beyond that of other conditions, R0, R1, R2 back into position can be arbitrary changes.

4) The addition and subtraction of carry bits must be done without the addition and subtraction of the carry, otherwise the result is wrong.

References

[1]. Yuyu Sun. Research on the design of Von Neumann computer design [J]. Journal of Changchun University, 2001, (2003): 35-37.

[2]. Wang Min, Zhang Shuping. A model of the ALU of logic operation [J]. China electronics education, 1999, (01): 55-57.

[3]. Yao Yunxia. Application of simulation software Proteus in the course teaching of computer components [J]. Agricultural network information, 2011, (11): 134-135.