Design of Two-digit Decimal Number Calculator Based on Verilog

Liu Bing Qing* and Zhou Qian† and Yao Ni Gua‡
Department of Information Science and Technology, Fudan University
220 Handan Road, Yangpu, Shanghai
Email: *17307130267@fudan.edu.cn, †18729283798@fudan.edu.cn,
‡18271836273@fudan.edu.cn

Abstract— Although existing calculators have fulfilled the basic requirements for addition and subtraction between one-digit numbers, there are still problems such as unreasonable displays, too simple calculation functions, and cumbersome design methods, which cannot provide users with a satisfying experience. To solve these problems, a scheme of a two-digit decimal number calculator based on the Verilog is designed. Different from the previous mechanical control methods, this design is relevant to programming languages in the Field Programmable Gate Array (FPGA) development platform of Quartus software and uses the Electronic Design Automation (EDA) tools for assembly, logic compilation, and simulation. In the design, by judging whether there are 0s in the upper bits to suppress the display of redundant zeros and adding a calculation function of numbers with the sign, the computing power and the displays of the results are greatly improved. Also, its calculation range with is expanded more digital tubes, which shows better test quality. The whole experiment was tested on the FPGA development board and simulated on the software, both demonstrated the feasibility and efficiency of the design scheme.

Keywords—EDA,Verilog,Calculator,Digital circuits

1 Introduction

In order to expand the functions of the calculator and optimize the display effect, the digital circuit form adopted by the traditional methods has been gradually eliminated, and the new EDA tools [1] have become the mainstream. This technology uses computers as tools and integrates the latest theories of databases, graphics, and computational [2] mathematics. . Figure 1 represents the typical interface of EDA tools.

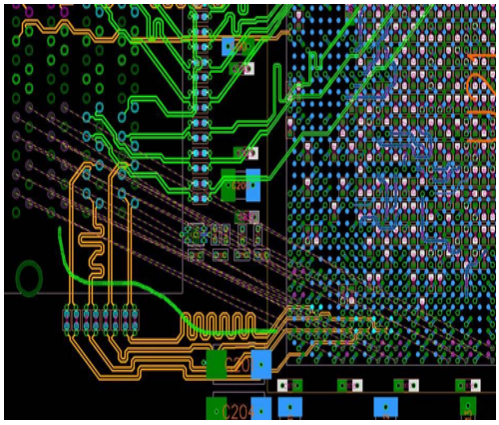

Fig. 1: EDA is used in PCB board design for fast routing and global optimization

Compared with traditional digital circuit design methods, using EDA to design has many extraordinary characteristics and advantages. It adopts a top-down design method. The primary simulation and error correction processes of the design are completed at a high level, which is helpful for simple detection of structural design errors, thereby avoiding waste of time and cost in work. Also, the workload of logic function simulation is significantly reduced and design efficiency is lifted evidently. Hardware description language is an essential part of EDA technology and a utterly significant software tool in EDA design and development. One of the advantages of using it for electronic system design is that it allows designers to focus on the realization of their functions without spending too much time and effort on process-related factors. Figure 3 demonstrates the overall circuit layout of a certain

project, making the advantage more distinct.

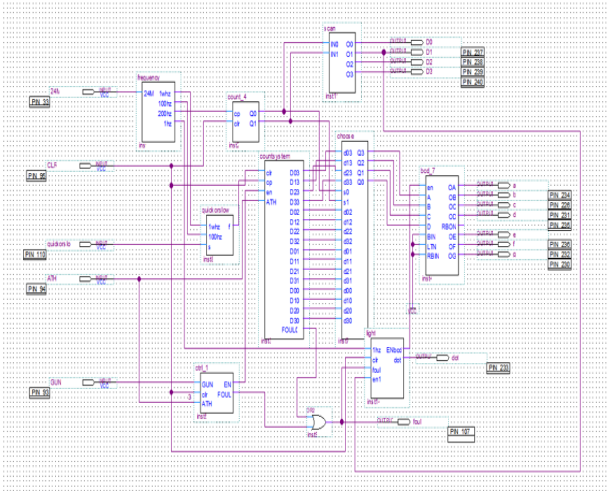

Fig. 2: Circuit top-level block is designed in Quartus

Besides, its supporting hardware integration is high enough and the hardware has small sizes and consumes relatively low energy. It leads to not only low design costs, short design cycle, good design portability and high efficiency, but also convenience for teamwork.Regardless of the design process or the design results, it is worth trying to use the EDA mainstream development software Quartus to compile the program design. The experimental results also show that the solution is feasible. Furthermore, it is well implemented in the compilation environment of Quartus and the hardware condition of cyclone board.

2 Design Method

2.1 Verilog HDL

Verilog generally refers to Verilog HDL. Verilog HDL is a Hardware Description Language (HDL), which describes the structure and behavior of digital system hardware in text form. It can be used to represent logical circuit diagrams, logical expressions, and the logical functions performed by digital logic systems.

Verilog was first designed to be a hardware description language with a basic syntax [3] similar to C. This is because C language has been widely used in many fields at the beginning of Verilog invention, and many language elements of C language have been well-received. Being similar to C makes it easier for circuit designers to learn and accept, although there are still many differences between Verilog and C.

As a hardware description language different from ordinary computer programming languages, it has some unique language elements, such as vector nets and registers, non-blocking assignments in the process, and so on. In general, designers with a C language will be able to grasp the Verilog hardware description language quickly.

2.2 Design Scheme Overview

The design is to complete a calculator that can implement addition, subtraction, and multiplication between two signed two-digit decimal numbers. The input data range is from 0 to 99, and the range of calculation results is from -9081 to 9081. The realization of the calculator function is to input the first data by pressing the keys, select the corresponding calculation symbol, then enter the second data, and press the equal sign to get the corresponding result.

It is found that most calculators in the current markets can only do a single-digit calculation, and they cannot process the addition and subtraction of negative numbers. This scheme is greatly improved and optimized on this basis. There is a total of five modules in this design plan, which respectively are input data module, calculation module, display module, transcoding module and control module. Based on the modular design idea, the top-down hierarchical design method is used for the project. The input signals include a key signal, a symbol signal, an equal signal, a zero signal and a negation signal. The output signal is the output data, which is the result of the calculation.

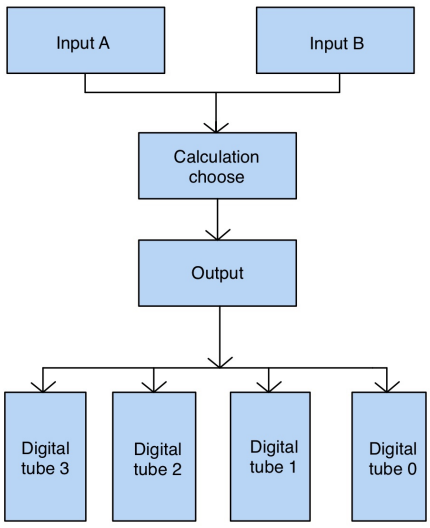

Fig. 3: Mind mapping of the design explains how i achieve the purpose of the calculator

2.2.1 Input Module Design

The one-digit calculator used a four-digit switch on the development board to indicate the number of the input data. For example, the levels of four switches, which are 1001, indicate that the data is 9. However, it is more difficult in reality to use them to show numbers because users do not have the time to convert what exactly the binary numbers of the input ones are, so i decided to increase the value of the input data by pressing keys. In order to avoid the trouble of converting two decimal numbers into two BCD codes, the design of the solution was to use two keys when inputting one data. One controlled the tens of the input data, and the other controlled the units. Each time the user clicks, the corresponding digit of the data is increased by one. What is more, it is more surprising that when the input data is relatively large, the number of key presses will be much less. Suppose that the input data is 78, instead of pressing the same key for 78 times, uses only need to press 15 times to adjust to the required number, saving pretty much time.

When the complement signal is valid, the most significant digit of the seven-digit digital tube is only lit in the middle, which means that the number becomes negative and the g tube is red, as is shown in Figure 3. When the operator in need is selected and the calculation symbol signal is valid, only the least significant digit zero of the four-digit digital tube lights up, waiting for the second data to be input. The second data input process is the same as the first data operation. When the equal sign signal is valid, the calculation result can be output. If the set signal [4] is required during the process, the digital tube will display zero.

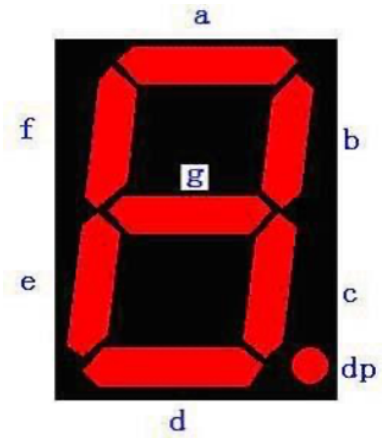

Fig. 4: Structure of the digital tube is the reference for the codes

When compiling a language, two input data need to be represented by two variables. Because the design takes into account the problem of positive and negative signs, the sign bit is also included in the input data, and a signed variable is added. The data with the added sign bit is also a new variable, which needs to be defined in the module.

In previous designs, many solutions chose to use keys as a carrier for inputting operator signals, and keys had to specify an intermediate variable to store the change in level. In programming languages, in order to reduce intermediate variables, switches were eventually selected as the input symbol to control the addition, subtraction, and multiplication symbols, making the program more concise and compiled faster and reducing the warnings accordingly.

2.2.2 Calculation Module Design

There were not many factors to consider in the calculation module. For the corresponding symbols were already provided in the language, addition could be implemented directly in the programming language, and subtraction was to invert the original number and add one to the inverted number, which was quite simple. The code for multiplication was complicated and needed to be completed using related rules. At the beginning of the multiplication of two four-digit binary numbers, if the last bit of the second number is 0, the result is shifted directly to the left [5] by one bit. If the second bit of the data is 1, the current result is equal to the first data adding the current result and second data is moved one bit to the left. This process is repeated eight times and the result of the multiplication is obtained. The codes of the multiplication module are as follows:

```
always @ (M1, M2, CLR, R, P)
begin
    R = M2;
    P = 1'b0;
    if (CLR)
    P = 0;
    else begin
            repeat(8)
            begin
                P = P <<1;
                if(R[7])
                P = P + M1;
                R = R <<1;
            end
        end
end
```

2.2.3 Transcoding Module Design

Because the sign bit calculation is added to this design, when the negation signal is valid, the module needs to negate and added one to the current binary data. When the data is input, units and tens are input separately. The BCD code is in need of being converted into a binary number to be calculated in the program. The final calculation result is binary, which requires to be converted into BCD code for display, and Figure 4 represents the process. In consideration of convenience, take the traditional one-digit decimal number calculator as an example, its results are divided into three types: negative one-digit numbers, positive one-digit ones, and positive two-digit ones.

When negative, the sign bit is 1, and the intermediate variable is used to invert the result, after which it is added one to get the units. When one-digit, the penultimate digit is 0, and the last digit shows that unit number. When two-digit, a method called shift and add three is time to use. If the calculator only needs to shift eight digits, so repeat 8. The codes judged whether every four digits are greater than 4, and if they are, three are added to get each bit, and units and tens are got. For a two-digit decimal number calculator, I repeated the above steps for 16 times.

100's	10's	1's	Binary	Operation
			1010 0010	162
		1	010 0010	<< #1
		10	10 0010	<< #2
		101	0 0010	<< #3
		1000		add 3
	1	0000	0010	<< #4
	10	0000	010	<< #5
	100	0000	10	<< #6
	1000	0001	0	<< #7
	1011			add 3
1	0110	0010		<< #8
	1	6	2	

Fig. 5: Schematic diagram of binary codes to BCD codes shows that the method is to shift the number and add three to it

If the final number is to be displayed on the FPGA development board, converting the BCD code into a seven-segment code is indispensable. This is the transcoding process that is necessary to be considered throughout the design.

2.2.4 Display Module Design

The main problem of the display module is dynamic scanning. The program needed to scan the four digital tubes seen in Figure 5 continuously, which meant that they would be lit in turn. There was not only one bit at a time, but also the data result of that bit, which met the demand of dynamic scanning. During the scanning process, the binary number of the computer result was converted into BCD codes and then converted into seven-segment codes before it was displayed. This has been described in the transcoding module design above. When writing a program, a clock signal cannot be ignored. Every time the rising edge of the clock signal came, the bright digital tube of the four was replaced. Display modules are almost common parts of such FPGA programs, and the method is essentially the same thing.

Fig. 6: Display of four-digit digital tubes

2.2.5 Control Module Design

In the control module, the program Design of controls whether the first number or the second number was input when the key was pressed. It means that the operator is required to enter. Otherwise, it is the first data. It also controls whether addition, subtraction, or multiplication is performed. If one of the addition and subtraction signals is valid, the control program performs that operation. At the same time, whether the data is cleared or not is commanded by the controller. Before entering the first data and before entering the second data, the nixie tube should show zero and only in the least significant position. Controlling the display of excessive zeros is also an important function of the control module, that is, determining whether the high level is zero. If it is, the digital tube was dimmed. If it is not, it continues to determine the next high-level data, which is more in line with the user habits. It can display four digits when calculating, but when the calculation result is less than

four digits, the previous digits are prevented from being displayed. For such improvements, adding some 'always' drivers in the program was enough.

2.2.6 Experimental Optimization Design

Real problems were founded after downloading the program to the FPGA development board for experiments. Due to the hardware limitations of the development board, the buttons and switches had the corresponding jitter. This was especially obvious when entering data. When a key was pressed, the data increased by more than 1, but a random number, because the key jumped slightly up and down during the pressing process, which caused the key signal to be valid for a couple of times. Although the switch also had a problem of instability, since not involved in the display field, it was not necessary to take optimization measures.

The switch used for the normal button was a mechanical elastic switch. When the mechanical contact is opened or closed, due to the elastic effect of the mechanical contact, a key switch will not turn on steadily immediately [6] when it is closed. Therefore, there was a series of jitters at the moment of closing and opening. The measure to prevent this phenomenon was to depress the keys.

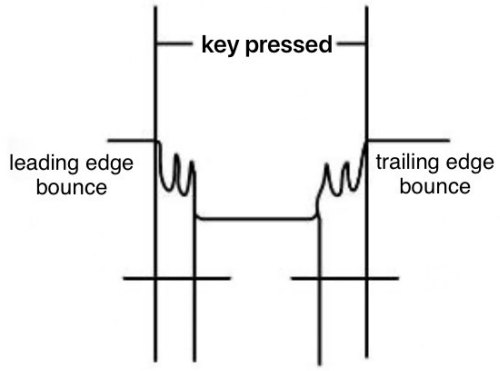

Fig. 7: Schematic of button jitter

Because software debounce is more convenient and does not require additional hardware overhead, it is generally used. The software debounce method commonly used in single-chip microcomputers was actually very simple. After the single-chip microcomputer obtains the information that the port is low, it is not immediately determined that the button is pressed, but it is tested again after a delay of 10ms or longer. If the port is still low, it means that the key was indeed pressed. This avoids the jitter time when the key is pressed, and when the port is detected to be high after the key is released, it is delayed for another 50ms. Eliminate the jitter on the back edge, and then process the keys. Nevertheless, in general, the back edge of the key release is usually not processed. The practice has proved that it can meet the daily requirements.

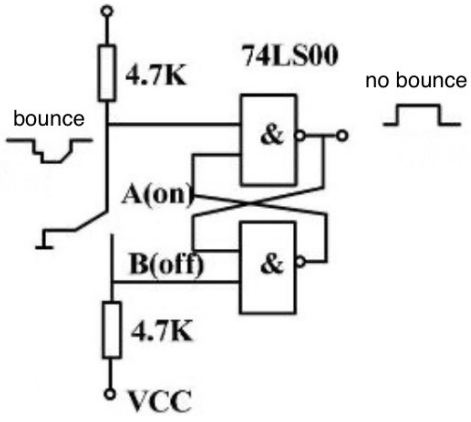

Fig. 8: Hardware debounce circuit

In the input module, the measure taken in this solution was the de-jittering program, which introduced an intermediate variable called count. This signal will increase by one as long as the button is shaken. It was only effective when the button was pressed, and the variable was equal to a certain number, after which the count variable changed to zero. This improved the situation of button shake to a certain extent.

3 Results

3.1 Operating Steps

The results of the design were shown in the steps.

Step1: Download the file obtained after the pins are assigned to the FPGA board. After the download is successful, one zero is displayed on the board. Increase the single-digit value by controlling the ones button at this time. To change the ten digits, press the ten-digit button, and input the first data 46.

Step2: Choose to dial the next operation symbol. For example, dial the plus switch.

Step3: Then enter the second data the same way as the first data. Suppose that the data is 67, and then press a sign button, it becomes -67. The negative sign is displayed at the highest position.

Step4: Press the equal switch to show the sign to get the final result -21. The negative sign is in the highest position, 21 is respectively at the penultimate position and the penultimate position.

Step5: Press the clear signal to reset, the display result is 0, in the lowest position. A new round of calculations can be performed.

3.2 Demonstration

The experimental results perfectly met the expectations, and the correct calculation results were obtained.

3.3 Defect

The problem of key jitter was better overcome when inputting data and inputting operator signals.But this calculator cannot proceed continuous addition, subtraction, and multiplication. However, since the calculator does not have similar functions in real life and does not need to use similar functions, it was not designed accordingly in this experiment.

4 Conclusion

This experiment can implement addition, subtraction, and multiplication between two signed two-digit decimal numbers. Taking into account the actual reading habits, the high zero suppression is not displayed during the design display process. The design process is slightly complicated, but the design goals are well accomplished. And most of the calculators on the market have been improved and functions have been expanded. In the design of the experiment, it should be noted that a highly important idea in the hardware description is the design of the modules. Sub-modules make the code clearer and more convenient to call, improve the efficiency of the code and the ability to correct errors, and save the time of designers. The rapid development of EDA tools has brought great convenience to the design of digital circuits.

Acknowledgments

We acknowledge the effort from the authors of the Verilog Learning. These resources make this project in the wild possible. This design was supported by the Programmable Tools Center of Fudan University.

References

[1] Zhou, X. & Hao, X. & Liu, S. & Wang, J. & Xu, J. & Hu, J. (2014). Design of Washing Machine Controller Based on VHDL. 194.10.1117/12.2780289.

[2] Alex, H. & Ilya, B. & Hg, F.. (2015). Digital Circuit Design Guide. Proceedings of EDA, IEEE, EDA System Foundation. 1097-1105.

[3] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2019). MobileNets: Efficient Compiling Languages for FPGA, arXiv:1704.04861 [cs.CV].

[4] Wang, L., Xiong, Y. & Wang, Z., Qiao, Y. & Lin, D., Tang, X. & Gool, L. V. (2016). Temporal Segment Networks: Towards Good Practices for FPGA designs. Computer Tools – ECCV 2016 Lecture Notes in Computer Science, 19–32. doi: 10.1087/978-3-319-46484-8_2.

[5] Feichtenhofer, C. & Pinz, A. & Zisserman, A. (2016). Experiments on FPGA Board, 2016 IEEE Conference on EDA tools (CVPR).