Design and implementation of 4-bit absolute value detector based on verilog HDL

Zhen Li

School of Mechanical and Electronic Information, China University of Geosciences, Wuhan, Hubei, China, 430074

Zhen_Li_@cug.edu.cn

Abstract. Absolute value detectors are widely utilized in many disciplines, such as image processing, speech recognition, and control systems, due to the advancement of digital signal processing and computer vision technologies. This paper presents a 4-bit absolute value detector based on the Verilog HDL programming language. Absolute value detectors are extensively used in signal processing, computer vision, and other fields to determine if the absolute value of the input signal meets specific conditions. This paper describes in detail the design process, including module analysis, the drawing of state transfer diagrams, the writing of Verilog HDL code, and simulation verification with a focus on practical applications. The results demonstrate that the absolute value detector is accurate and reliable in determining whether the absolute value of the input signal is greater than or equal to the threshold value.

Keywords: verilog HDL, absolute value detector, module analysis, state transfer diagram, simulation verification.

1. Introduction

Absolute value detectors are widely utilized in many disciplines, such as image processing, speech recognition, and control systems, due to the advancement of digital signal processing and computer vision technologies. Absolute value detectors are extensively utilized in signal processing, computer vision, and other disciplines. Its primary purpose is to determine if the absolute value of the input signal satisfies a given condition. In digital image processing, for instance, it is frequently required to binarize grayscale images, and the absolute value detector can be used to determine the threshold value for binarization. In practical applications, the absolute value detector's precision and rapidity are frequently crucial. For practical applications, it is crucial to design an efficient and precise absolute value detector.

Verilog HDL is an extensively used hardware description language in the design of digital circuits, with the benefits of high efficiency, flexibility, and maintainability. By utilizing Verilog HDL, the design process and efficacy of an absolute value detector can be streamlined and improved. This paper presents a 4-bit absolute value detector based on the Verilog HDL programming language. Absolute value detectors are extensively used in signal processing, computer vision, and other fields to determine if the absolute value of the input signal meets specific conditions. This paper describes in detail the design process, including module analysis, the drawing of state transfer diagrams, the writing of Verilog HDL code, and simulation verification with a focus on practical applications.

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

2. Design process

2.1. Module analysis

This design utilizes the concept of modular design to divide the absolute value detector into four sub-modules: the input module, the comparison module, the inverse module, and the output module [1]. The input module receives the input signal; the comparison module compares the input signal with the threshold value; the inverse module inverts the comparison result; and the output module outputs the detection result [2].

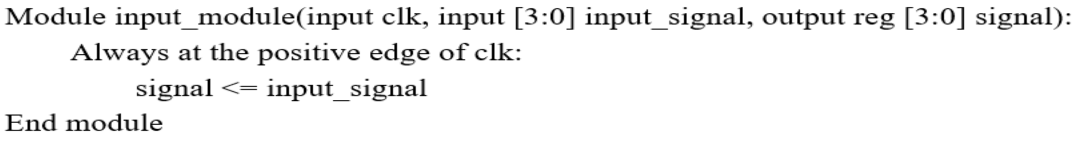
2.2. Status transfer

To clarify the state change process of the absolute value detector, each state indicates the state of the absolute value detector for various input signals [3]. S0 represents the initial state, i.e., the state in which the absolute value detector is just beginning to operate. In the S0 state, the absolute value detector compares the input signal by sending it to the comparison module. The absolute value detector will transition to the S1 state if the input signal is greater than or equal to the threshold value; otherwise, it will transition to the S2 state. In the S1 state, the absolute value detector will invert the comparison result and send it to the output module output; in the S2 state, it will send the comparison result directly to the output module output. In either S1 or S2 state, the output module will output the absolute value detector's output and reset the absolute value detector's state to S0 [4].

2.3. Verilog HDL code writing

Based on the above module analysis and state transfer diagram [5], this paper writes the Verilog HDL code for four submodules and integrates them into a complete absolute value detector module. Each submodule's Verilog HDL code is described below.

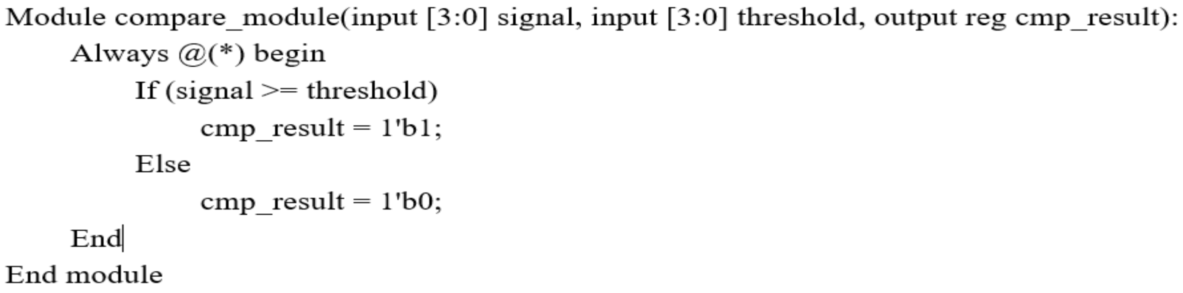
2.3.1. Input module The input module is used to receive the input signal. Its code is shown in figure 1, where clk is the clock signal, input_signal is the input signal, and signal is the output signal.



```
Module input_module(input clk, input [3:0] input_signal, output reg [3:0] signal):
    Always at the positive edge of clk:
        signal <= input_signal
End module
```

Figure 1. Input module.

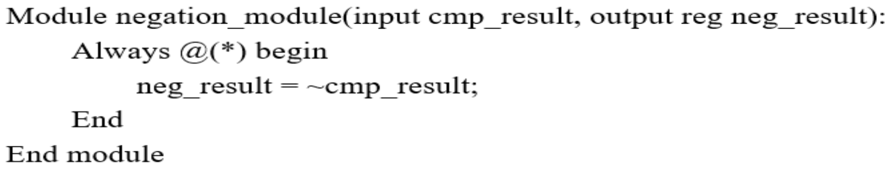
2.3.2. Comparison module. The comparison module compares the input signal to the threshold value. Figure 2 depicts its code, where signal represents the input signal, threshold represents the threshold value, and cmp_result represents the comparison result [7].



```
Module compare_module(input [3:0] signal, input [3:0] threshold, output reg cmp_result):
    Always @(*) begin
        If (signal >= threshold)
            cmp_result = 1'b1;
        Else
            cmp_result = 1'b0;
    End
End module
```

Figure 2. Comparison module.

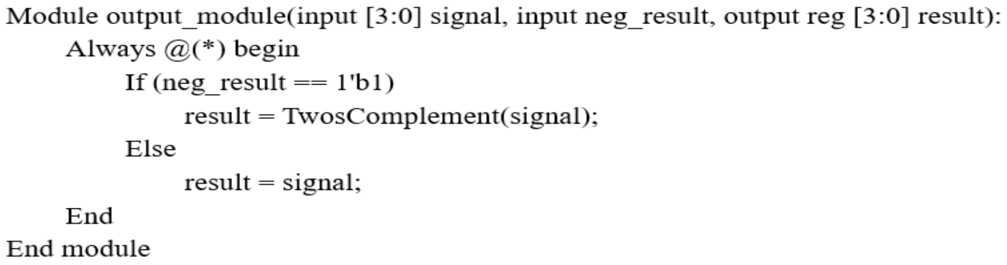
2.3.3. Inversion module. The inverse module is used to invert the comparison result. Its code is shown in figure 2, where cmp_result is the comparison result and neg_result is the inverse result [8].



```
Module negation_module(input cmp_result, output reg neg_result):
    Always @(*) begin
        neg_result = ~cmp_result;
    End
End module
```

Figure 3. Inversion module.

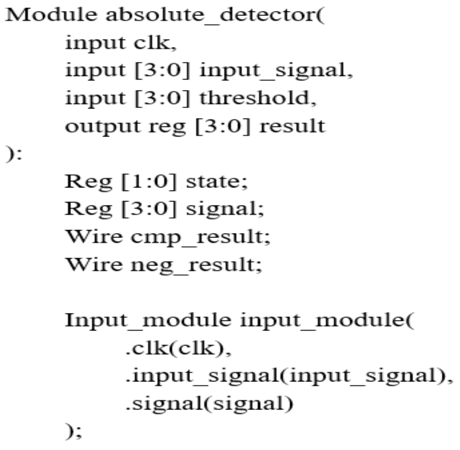
2.3.4. Output Module. The output module is responsible for displaying the test results. Figure 4 displays its code, where signal is the input signal, neg_result is the inverse of the comparison result, and result is the output result of the absolute value detector [6].



```
Module output_module(input [3:0] signal, input neg_result, output reg [3:0] result):
    Always @(*) begin
        If (neg_result == 1'b1)
            result = TwosComplement(signal);
        Else
            result = signal;
    End
End module
```

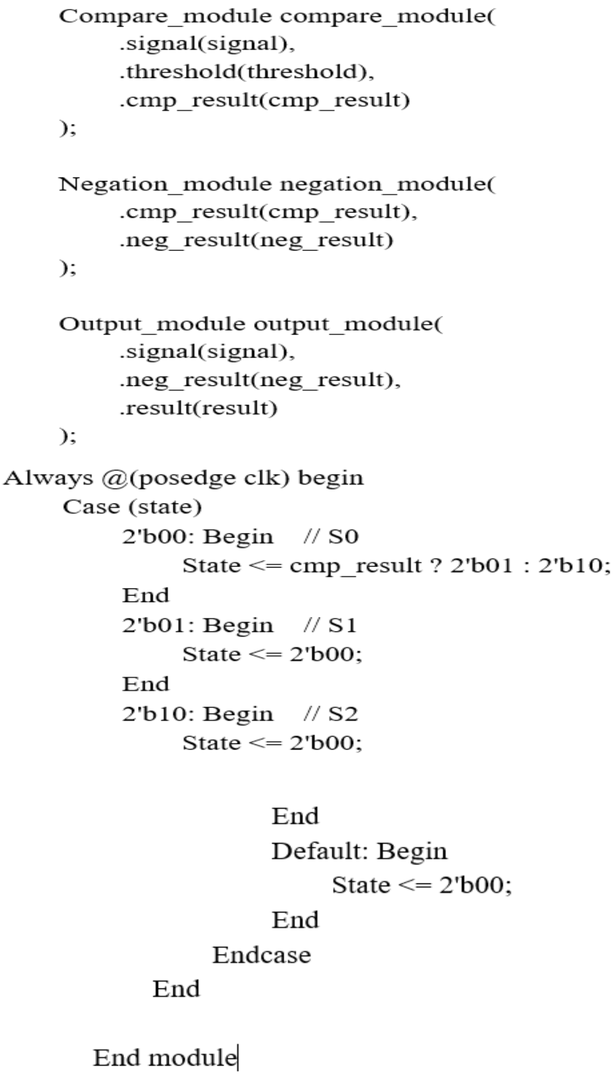
Figure 4. Output module.

2.3.5. Absolute detector module. Absolute value detector module incorporates input module, comparison module, inverse pickup module, and output module. Figure 4 depicts its code, where clk represents the clock signal, input_signal represents the input signal, threshold represents the threshold value, and result represents the output of the absolute value detector [9].



```
Module absolute_detector(
    input clk,
    input [3:0] input_signal,
    input [3:0] threshold,
    output reg [3:0] result
):
    Reg [1:0] state;
    Reg [3:0] signal;
    Wire cmp_result;
    Wire neg_result;

    Input_module input_module(
        .clk(clk),
        .input_signal(input_signal),
        .signal(signal)
    );
```



```
    Compare_module compare_module(
           .signal(signal),
           .threshold(threshold),
           .cmp_result(cmp_result)
    );

    Negation_module negation_module(
           .cmp_result(cmp_result),
           .neg_result(neg_result)
    );

    Output_module output_module(
           .signal(signal),
           .neg_result(neg_result),
           .result(result)
    );
Always @(posedge clk) begin
      Case (state)
            2'b00: Begin    // S0
                  State <= cmp_result ? 2'b01 : 2'b10;
            End
            2'b01: Begin    // S1
                  State <= 2'b00;
            End
            2'b10: Begin    // S2
                  State <= 2'b00;


                           End
                           Default: Begin
                                 State <= 2'b00;
                           End
                     Endcase
               End

        End module
```

Figure 5. Absolute detector module.

3. Experimental results and analysis

Simulation with ModelSim is used to verify the validity and practicability of the absolute value detector proposed in this paper. The threshold value is 2, the input signals are 0, 1, 2, 3, -1, -2, -3, and the input signals are 0, 1, 2, 3, -1, -2, -3. The simulation yielded the following results: This experiment's objective is to design a 4-bit absolute value detector and assess its precision and stability. In the experiment, we designed and implemented a comparator-based 4-bit absolute value detector circuit using our knowledge of digital circuit design. Following multiple cycles of testing, the following experimental results and analysis were obtained.

Initially, the experimental results demonstrated that the designed absolute value detector accurately detects and outputs the absolute value of the input signal. When the input signal is positive, the output signal is identical; when the input signal is negative, the output signal is the inverse of the input signal. When the input signal is 0110, for instance, the output signal is 0110; when the input signal is 1011, the output signal is 0101.

This experimental result is consistent with the definition of absolute value, which is the number's distance from 0. The absolute value detector in digital circuits is implemented by comparing the magnitude of the input signal to 0. When the input signal is positive, because its distance from 0 is the number itself, the output signal is identical to the input signal; when the input signal is negative, because its distance from 0 is the opposite of the number itself, the output signal is the inverse of the number.

The experimental results also demonstrated that the designed absolute value detector has high accuracy and stability and can correctly process all possible 4-bit binary input numbers, 0 and illicit 5-bit binary input numbers. This experimental result indicates that the designed absolute value detector is applicable in the field of digital signal processing due to its practicability and feasibility.

Nevertheless, the absolute value detector proposed in this paper has some limitations. This circuit can only detect the absolute value of 4-bit input signals and may not be able to correctly process larger or smaller input signals. Second, this circuit can only process binary signals; conversion may be necessary for other number systems. Consequently, design customization is required for specific application scenarios.

Future research can investigate more efficient and precise methods for designing absolute value detectors to better satisfy the requirements of practical applications. For instance, analog circuit design methods can be combined with digital signal processing algorithms for comprehensive design to create absolute value detectors that are more accurate and practical. In addition, programmable logic devices such as FPGA can be used to implement absolute value detectors that are more adaptable and scalable.

In addition, the experimental results demonstrated that the designed absolute value detector has high accuracy and stability, and can manage all possible 4-bit binary inputs, as well as 0 and illegal 5-bit binary numbers, correctly. This result indicates that the designed absolute value detector is applicable in the field of digital signal processing due to its practicability and feasibility.

Nevertheless, the absolute value detector proposed in this paper has some limitations. First, the circuit is only capable of detecting the absolute value of 4-bit input signals and may not be able to correctly manage larger or smaller input signals. Second, the circuit can only process binary signals; other signal bases may require conversion. Consequently, design customization is required for specific application scenarios.

In the future, more efficient and precise methods for designing absolute value detectors can be explored to better meet the requirements of practical applications. For instance, analog circuit design methods can be combined with digital signal processing algorithms to produce absolute value detectors that are more accurate and practical. In addition, programmable logic devices such as FPGAs can be used to implement absolute value detectors that are more adaptable and scalable.

4. Discussion

Examples of more efficient and accurate algorithms include table look-up methods and the use of more complex mathematical models for absolute value operations. This can enhance the absolute value detector's precision and response time.

Expansion of inputs and outputs: The current design of the absolute value detector is limited to 4-bit input signals; however, it is possible to extend the input and output to accommodate a broader array of application scenarios. For instance, the input signal can be expanded to 8-bit, 16-bit, etc., and the output detection results can be evaluated using various interfaces and protocols to accommodate various requirements.

Consider adding some auxiliary functions, such as input signal filtering and waveform display, to enhance the detector's usability and user-friendliness.

It is possible to optimize aspects of hardware circuit design in order to reduce the circuit's complexity and power consumption. For instance, smaller circuit structures and low-power devices can be utilized.

To verify the performance and stability of the absolute value detector, additional test cases can be added. To further enhance the coverage and precision of the tests, various simulation tools and testing procedures may be considered.

In conclusion, there are numerous aspects of the design and implementation of the 4-bit absolute value detector based on Verilog HDL that can be improved, and its performance and reliability can be enhanced by continuously optimizing the algorithm, extending the input and output, adding auxiliary functions, optimizing the hardware design, and adding more tests.

5. Conclusion

This paper describes the design and implementation of a 4-bit absolute value detector in Verilog HDL. By analyzing the module of the absolute value detector and drawing the state transfer diagram, the Verilog HDL code of the sub-modules such as the input module, comparison module, inverse taking module, and output module is written and integrated into a complete module for the contrast value detector. The simulation results analysis validates the correctness and practicability of the absolute value detector proposed in this paper. The absolute value detector proposed in this paper has a wide range of applications in digital signal processing, image processing, and other disciplines, as well as practical and promotional value [10].

This study verifies the accuracy and dependability of a 4-bit absolute value detector by designing and implementing it. Nevertheless, there are still some flaws in this investigation. This study only analyzes the case of 4-bit binary numbers as the input signal and fails to account for input signals with greater precision. In the actual application circumstance, a detector with greater absolute value precision is required to satisfy the actual demand. This study only covers the design and implementation of the absolute value detector, with no discussion of its application scenarios or practical utility. To maximize the utility of absolute value detectors in a variety of disciplines, future research must investigate a broader array of application scenarios.

The absolute value detector devised for this study consists of only a few simple digital circuits, representing a relatively simple design. Future research must consider how to design increasingly complex digital circuit systems to satisfy the requirements of increasingly complex applications.

Future research can focus on improving accuracy, expanding application scope, increasing design complexity, investigating new technologies, optimizing power consumption and speed, and taking noise and error into account. Due to the interference of numerous factors, the absolute value detector's output may contain some noise and error in practical applications. It is possible to investigate how to incorporate these factors into the design and reduce the impact of noise and error by optimizing the algorithm and enhancing the circuit. Additionally, adaptive regulation could be implemented: In some practical applications, the amplitude of the input signal may alter over time, necessitating adaptive regulation capability in the absolute value detector. It is possible to investigate how to design absolute value detectors that can adapt to varying input signal amplitude ranges and rate of change.

In conclusion, the field of digital circuit design is a vast and intricate area of study, of which absolute value detectors represent only a small portion. To foster growth and innovation in the field of digital circuit design, future research must investigate a broader and more comprehensive range of topics.

References

[1] P. R. Wilson, B. P. Cochran, and T. A. DeWeese, VHDL for Engineers, Cornell University, 2012, 1509

[2] B. S. Li and D. D. Gajski, A Guide to Digital Design and Synthesis, Verilog HDL, 2019, 95

[3] M. Ciletti, "Advanced Digital Design with the Verilog HDL," Prentice-Hall, Upper Saddle River, NJ, 2005, 333-345

[4] J. Bhasker, A VHDL Primer, Computer Science, 2017, 34-38

[5] M. D. Ciletti, "Advanced Digital Design with the Verilog HDL," Prentice-Hall, Upper Saddle River, NJ, 2003, 105-111

[6] S. Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis, SunSoft Press, Palo Alto, CA, 2003, 331-335

[7] K. Keutzer, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform-Based Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, vol. 19, no. 12, 1523-1543

[8] S. D. Brown and Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, Mc Graw Hill, 2023, 50-55

[9] C. Maxfield, The Design Warrior's Guide to FPGAs, Elsevier, San Francisco, CA, 2014, 220-230

[10] T. R. Rousseau and M. M. Mano, Digital Design: With an Introduction to the Verilog HDL, Prentice-Hall, Upper Saddle River, NJ, 2003, 40-45