**Lecture6 Assignments 1&2**

**6.1 Read and understand the Verilog code, write brief function description for each module.**

(1)adder : Performs a 32-bit addition of inputs a and b, assigning the result to y = a + b.

(2)alu : 32-bit Arithmetic Logic Unit, using case-alucont to control. AND 00 , OR 01 , add 10 , signed comparison 11. The zero flag is set if the result is zero.

(3)sl2 : Shifts the input a left by 2 bits, and set the 2 least sig bits as 00. This is a combinational circuit used for multiplication by 4 or address adjustment.

(4)signext : Extends a 16-bit signed input (a[15:0]) to 32 bits by replicating its MSB (sign bit) to upper 16 bits.

(5)flopr : Sequential Circuits -- A parameterized D flip-flop with synchronous operation and asynchronous reset. On the rising edge of clk, if reset is high, q is cleared to 0; otherwise, q takes the value of d. This is a sequential circuit.

(6)mux2 : A parameterized 2-to-1 multiplexer. Selects between d0 (when s is 0) and d1 (when s is 1) to assign to y. This is a combinational circuit.

(7)flopenr : A parameterized D flip-flop with asynchronous reset and an enable signal. On the rising edge of clk, if reset is high, q is cleared to 0; if en is high, q takes the value of d. This is a sequential circuit with write control.

(8)flopenrc : A parameterized D flip-flop with asynchronous reset, enable, and synchronous clear, with additional clear functionality. On the rising edge of clk, if reset is high, q is cleared to 0; if clear is high, q is cleared to 0; if en is high, q takes the value of d.

**6.2 ALU design and simulation**

**Assignment-1: 8-bit ALU Design**

Write Verilog code for an 8-bit Arithmetic and Logic Unit, and verify your design.

**Assignment-2: 8-bit ALU with an accumulator**

The accumulator is an edge triggered register with asynchronous reset. Write Verilog code for this 8-bit ALU with an accumulator register.

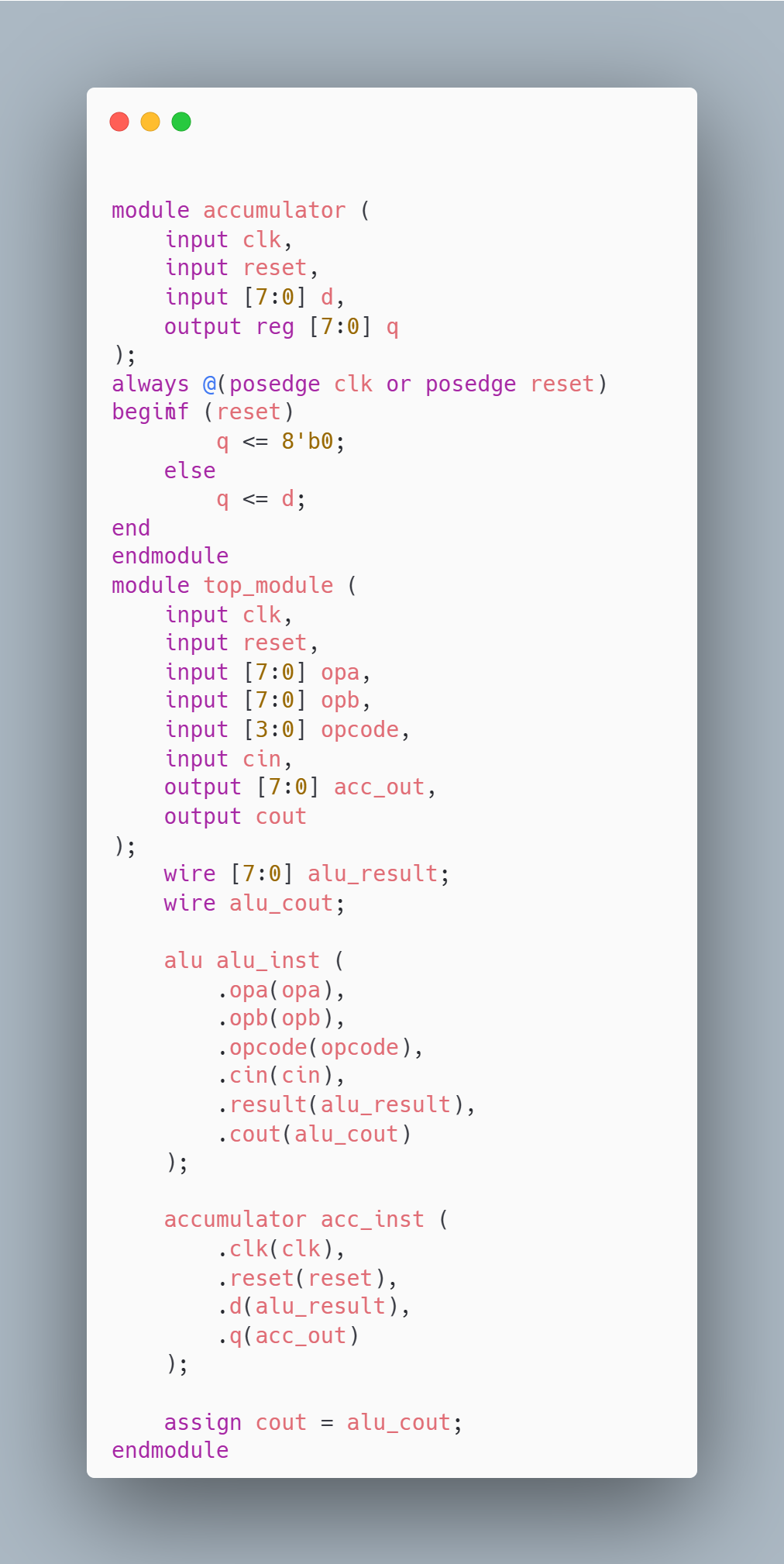
1.The first part of the assignment implements the code for the entire module

I use the module alu to finish the combinational part.

图片包含 文本

AI 生成的内容可能不正确。

The sequential part is below.



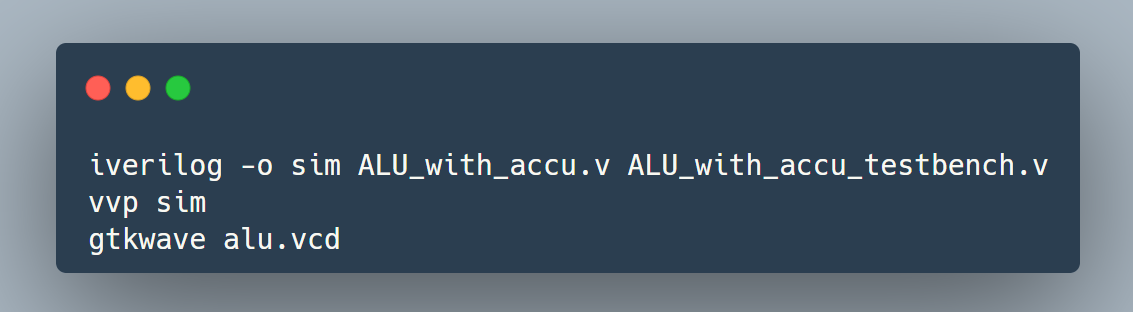
2. The second part is to test my code.

文本

AI 生成的内容可能不正确。

This testbench completes the inspection of the module and implements addition based on the example given in the question. The following is the waveform file and the measured waveform we obtained. Here we use the VSCode plug-in iVerilog and WaveTrace, which can more conveniently display the waveform.

I output this calculated value in the integrated terminal



– the result is 8hA0

