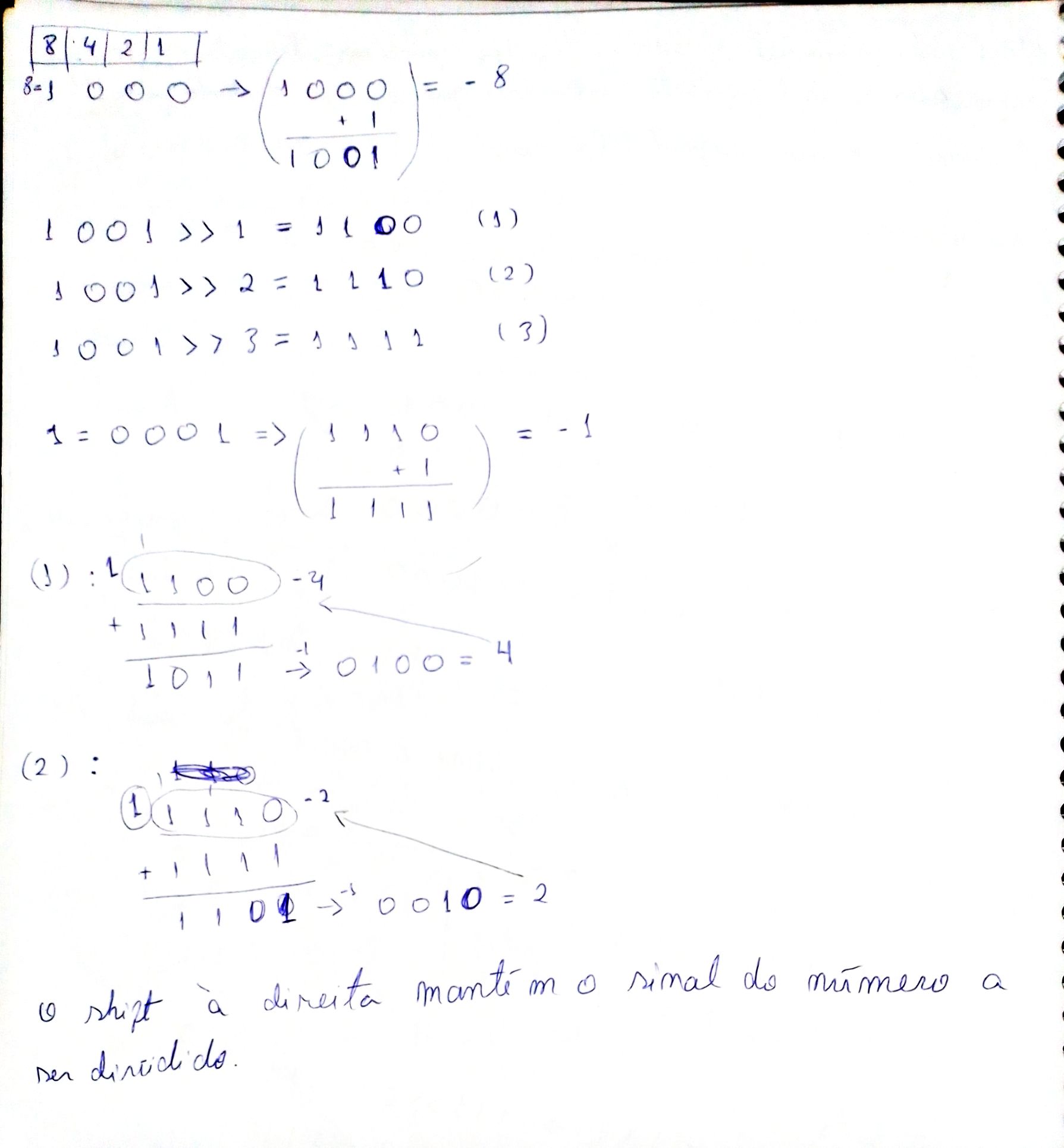
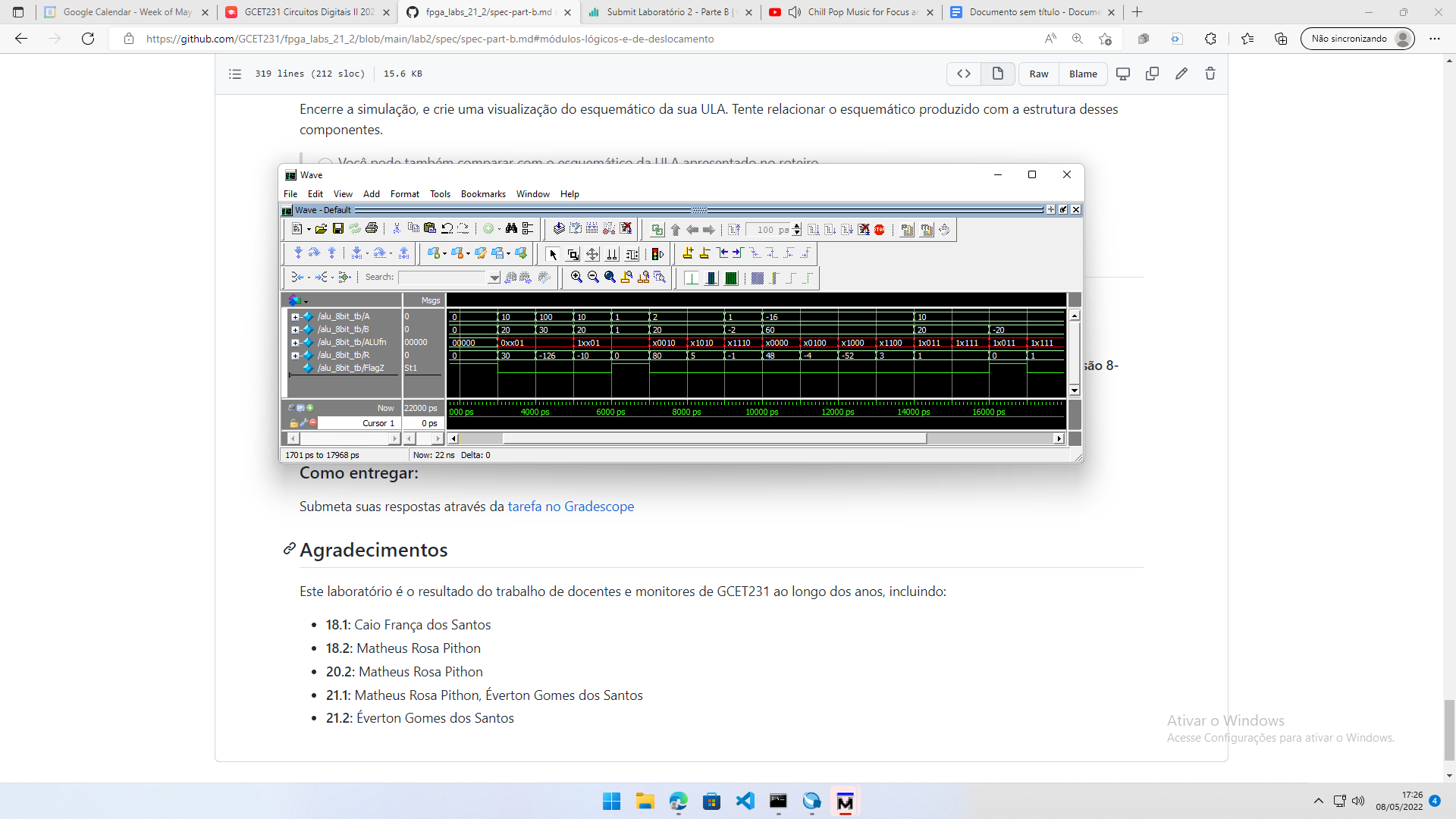
Você está usando operadores Verilog para os os três tipos de operação de deslocamento. Observe que o tipo de dado de IN é declarado como signed. Por que?

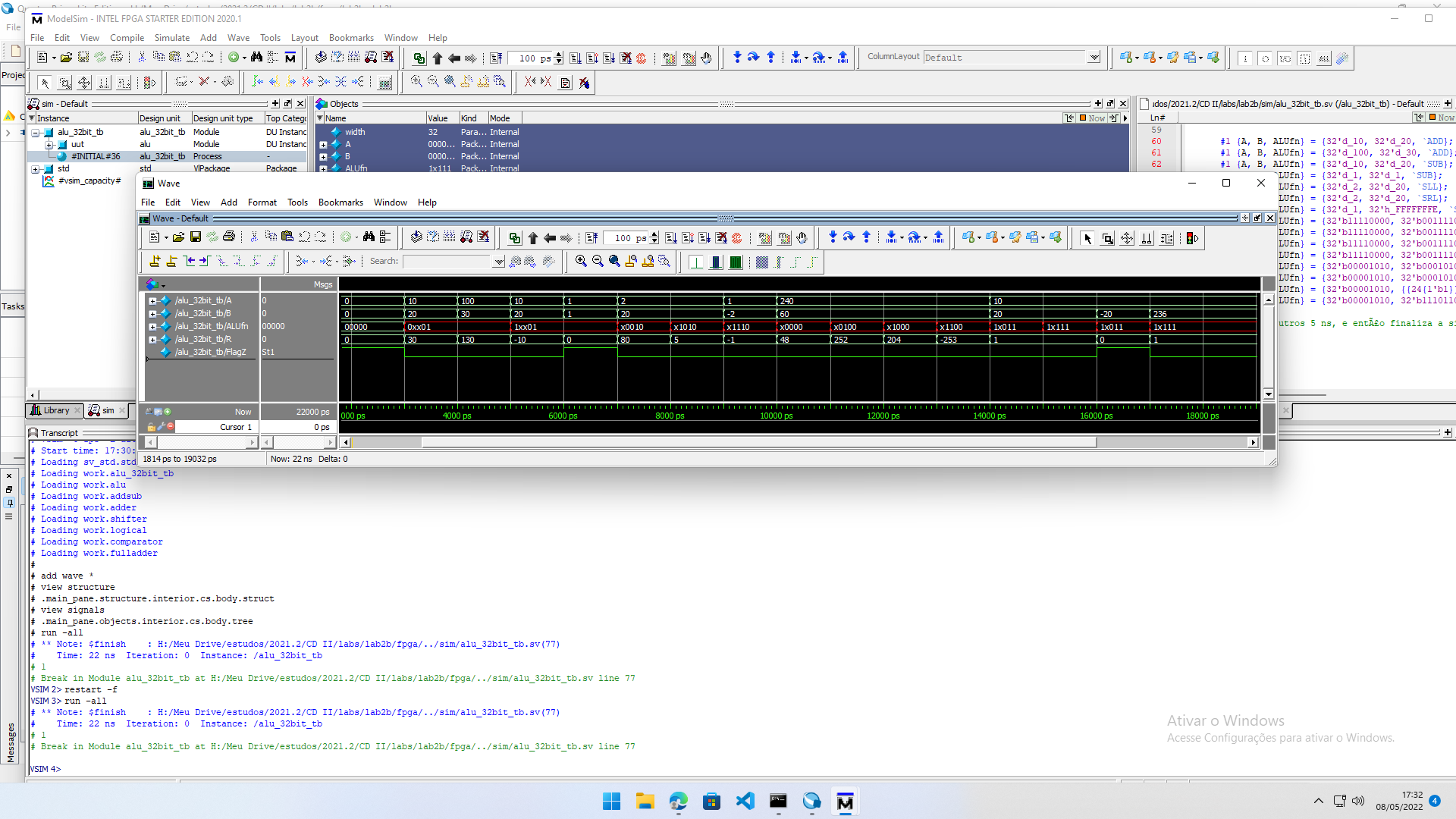
*Considere as operações de deslocamento à direita em um número de 4 bits abaixo:*



**Figura 1**. Resposta para a pergunta na seção "Módulos Lógicos e de Deslocamento”.



**Figura 2**. Resultados da simulação final para a versão 8-bits.



**Figura 3.** Resultados da simulação final para a versão 32-bits.

Código do módulo *ula*:

module alu #(parameter N=32) (

input wire [N-1:0] A, B,

output wire [N-1:0] R,

input wire [4:0] ALUfn,

output wire FlagZ

);

wire FlagN, FlagV, FlagC;

wire subtract, bool1, bool0, shft, math;

wire compResult;

//separar ALUfn em bits nomeados

assign {subtract, bool1, bool0, shft, math} = ALUfn[4:0];

//Resultados dos três componentes da ALU

wire [N-1:0] addsubResult, shiftResult, logicalResult;

addsub #(N) AS(A, B, subtract, addsubResult, FlagN, FlagC, FlagV);

shifter #(N) S(B, A[$clog2(N) -1:0], ~bool1&~bool0, bool1&~bool0, shiftResult);

logical #(N) L(A, B, {bool1,bool0}, logicalResult);

comparator C(FlagN, FlagV, FlagC, bool0, compResult);

//Multiplexador de 4 entradas para selecionar a saída

assign R = (~shft & math) ? addsubResult :

(shft & ~math) ? shiftResult :

(~shft & ~math) ? logicalResult : {{(N-1){1'b0}}, compResult};

// Utilizar o operador de resolução

assign FlagZ = ~(|R);

endmodule

Código do módulo *comparator*:

module comparator (

input wire FlagN, FlagV, FlagC, bool0,

output wire comparison

);

assign comparison = bool0 & ~FlagC | ~bool0 & FlagN & ~FlagV | ~bool0 & ~FlagN & FlagV;

endmodule

Código do módulo *logical:*

module logical #(parameter N=32)(

input wire [N-1:0] A, B,

input wire [1:0] op,

output wire [N-1:0] R

);

assign R = (op == 2'b00) ? A&B :

(op == 2'b01) ? A|B :

(op == 2'b10) ? A^B :

(op == 2'b11) ? ~(A|B) : {N{1'b1}};

endmodule

Código do módulo *shifter:*

module shifter #(parameter N=32)(

input wire signed [N-1:0] IN,

input wire [$clog2(N)-1:0] shamt,

input wire left, logical,

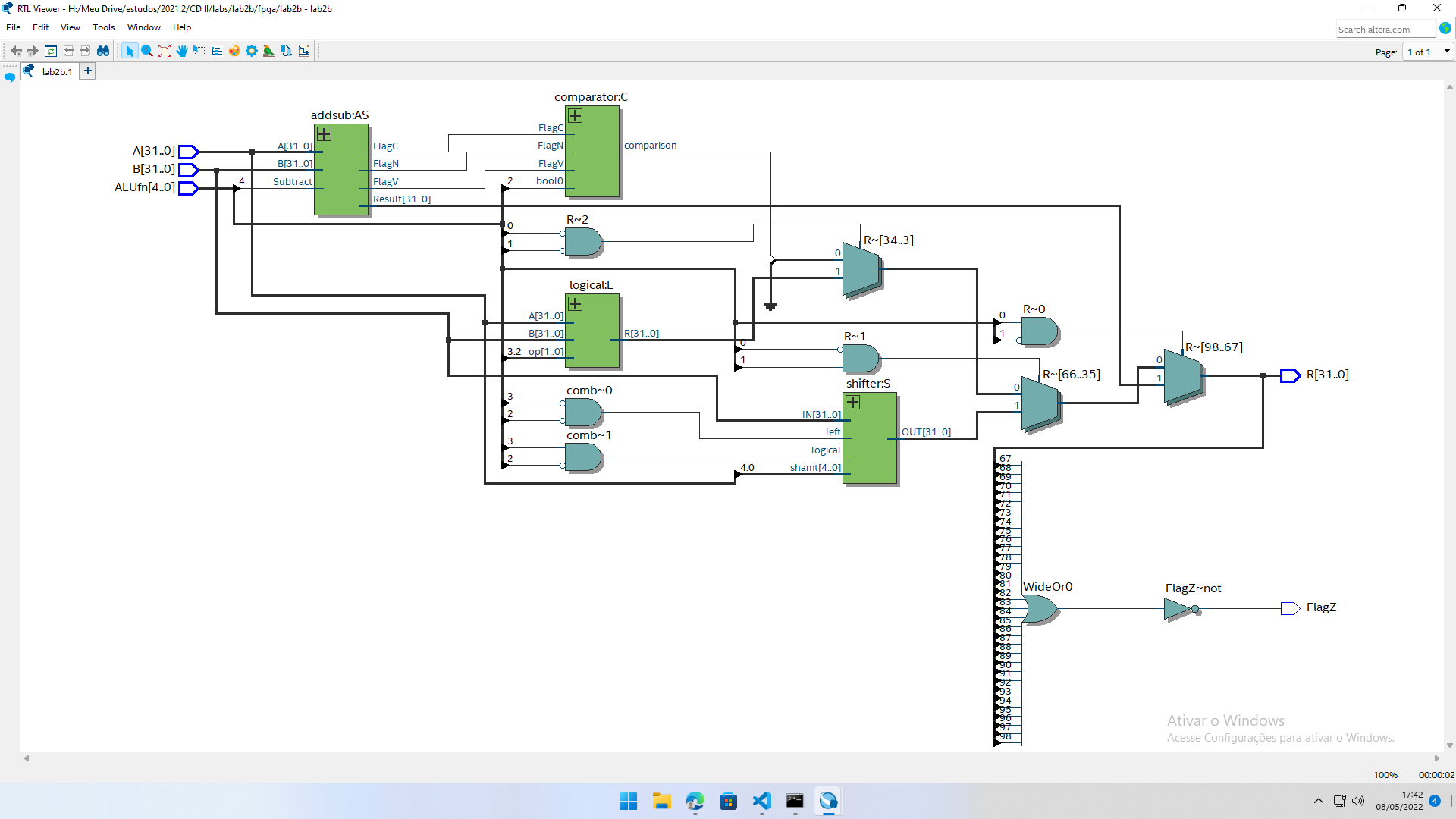
output wire signed [N-1:0] OUT

);

assign OUT = left ? (IN << shamt) :

(logical ? IN >> shamt : IN >>> shamt);

endmodule



**Figura 4.** Esquemático do circuito para o projeto final (32-bits).