2022 Digital IC Design

Homework 1: Arithmetic logic unit

1. Introduction:

Arithmetic logic unit is a combinational digital circuit that performs different operations according to different control signal lines. In this homework, you are required to design a 1-bit ALU circuit which can perform the basic calculation operations. The 1-bit ALU module is then used to constitute an 8-bit ALU circuit. The operation of the ALU is listed in Table I, and the specifications and function of the 1-bit ALU and the 8-bit ALU are detailed in the following sections.

Table I. Operation of the ALU.

Operation	Ainvert	Binvert	*op*	Description
Add	0	0	10	Operand1 + Operand2
Sub	0	1	10	Operand1 – Operand2
And	0	0	00	Operand1 & Operand2
Or	0	0	01	Operand1 | Operand2
Nand	1	1	01	~(Operand1 & Operand2)
Nor	1	1	00	~(Operand1 | Operand2)
SLT	0	1	11	(Operand1 < Operand2)? 1 : 0

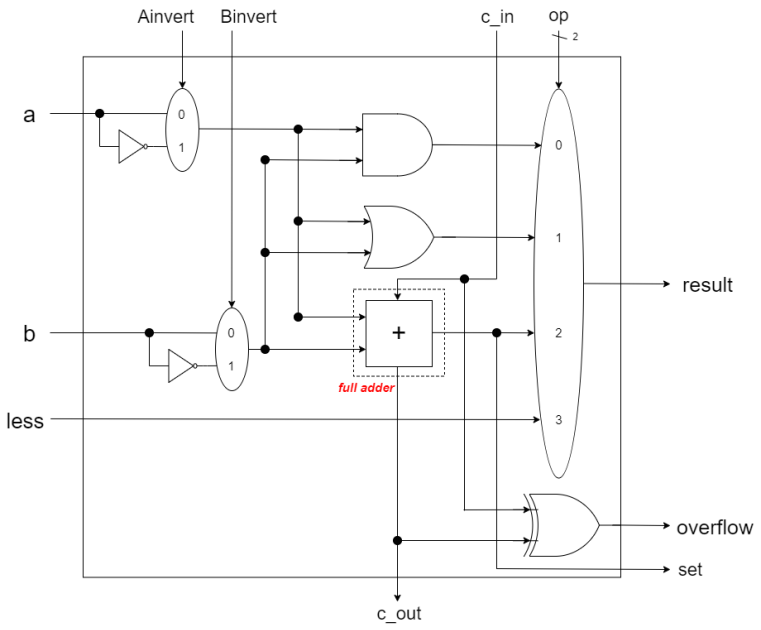

Fig. 1. The logic diagram of 1-bit ALU.

1.1. 1-bit ALU

The logic diagram of the 1-bit ALU for this homework is shown in Fig. 1, and its specifications of I/O interface is listed in Table II. When the invert signal is high, the corresponding operand has to be converted to 1's complement before operation. The operation is determined according to the *op* signal. The SLT(Set Less Than) operation is implemented with subtraction operation. If the subtraction result is minus, the *set* port should output 1. The *less* signal will be passed to the *result* port. The *overflow* signal should be determined according to the subtraction result.

Table II. I/O interface of the 1-bit ALU.

Signal Name	I/O	width	Description
a	I	1	Input operand 1.
b	I	1	Input operand 2.
less	I	1	Result of SLT operation. The *less* port of the 1-bit ALU for LSB will be connected to the *set* port of the 1-bit ALU for MSB.
Ainvert	I	1	Complementation signal of operand 1.
Binvert	I	1	Complementation signal of operand 2.
c_in	I	1	Carry in.
op	I	2	Operation code.
result	O	1	Operation result.
c_out	O	1	Carry out.
set	O	1	Result of SLT operation. The *less* port of the 1-bit ALU for LSB will be connected to the *set* port of the 1-bit ALU for MSB.
overflow	O	1	Calculation overflow signal.

1.2. 8-bit ALU

The 8-bit ALU can be constructed by cascading eight 1-bit ALU modules. The architecture of the 8-bit ALU is illustrated in Fig. 2, and its specifications of I/O interface is listed in Table III. The SLT operation is implemented with subtraction operation. If the subtraction result is minus, the *set* port of the 1-bit ALU for MSB should output 1. The *less* port of the 1-bit ALU for LSB will then take the *set* signal from MSB and pass it to *result*. The subtraction operation may cause overflow. To

ensure that the SLT operation is correct, an overflow detection circuit should be implemented. The *set* signal from MSB will be adjusted according to the *overflow* signal before it is transmitted to the *less* port of the 1-bit ALU for LSB. The *less* port of the 1-bit ALUs besides from LSB should be hardwired to 0.

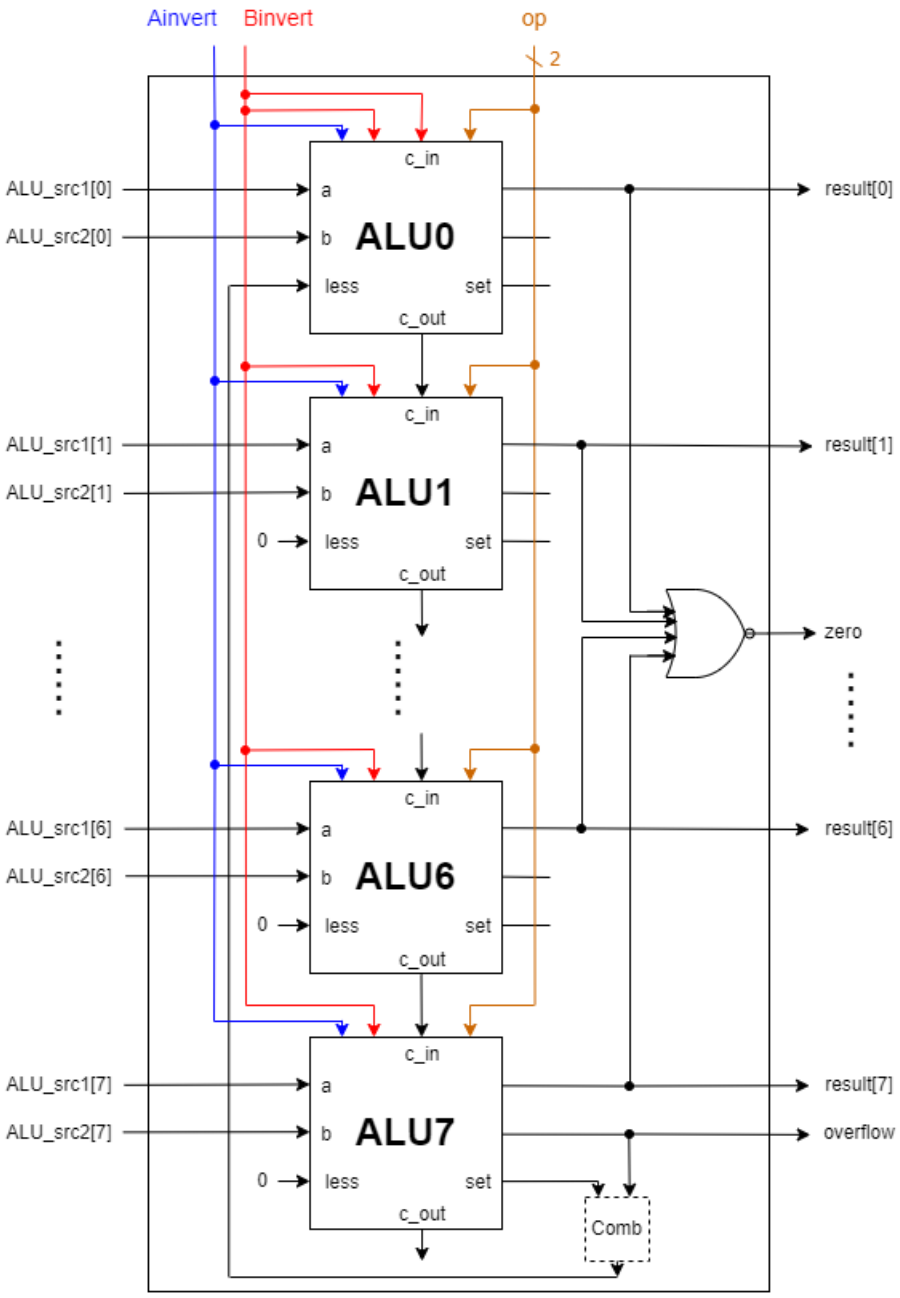

Fig. 2. The architecture of 8-bit ALU.

Table III. I/O interface of the 8-bit ALU.

Signal Name	I/O	width	Description
ALU_src1	I	8	Input operand 1.

ALU_src2	I	8	Input operand 2.
Ainvert	I	1	Complementation signal of operand 1.
Binvert	I	1	Complementation signal of operand 2.
op	I	2	Operation code.
result	O	8	Operation result.
zero	O	1	Zero signal. When the result is equal to 0, this signal has to be pulled up. Otherwise, it has to be pulled down.
overflow	O	1	Calculation overflow signal.

1.3. Full Adder

In the architecture of 1-bit ALU, a full adder is necessary. A full adder can be constructed with two half adders and an OR gate. Fig. 3 shows the logic diagram of the half adder and the full adder. In this homework, the module of full adder is already given, you can use it to accomplish your design.

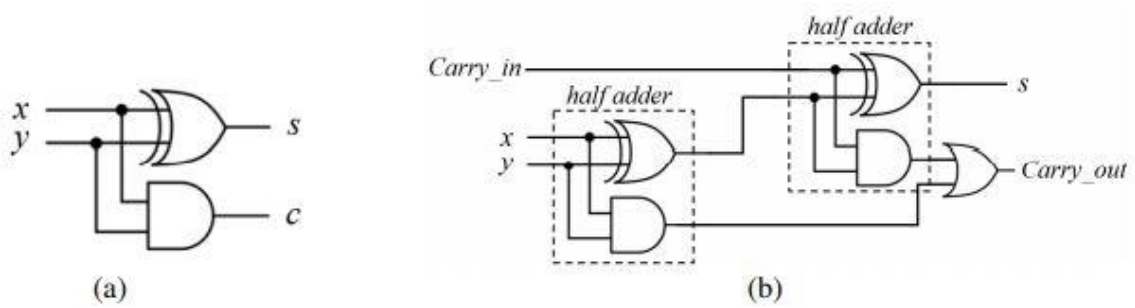

Fig. 3. Logic diagram of (a) an half-adder and (b) a full adder.

2. File Description:

File Name	Description
HA.v	The module of half-adder.
FA.v	The module of full adder.
ALU_1bit.v	The module of 1-bit ALU.
ALU_8bit.v	The module of 8-bit ALU.
ALU_tb.v	The testbench file. The content in this file is **not allowed** to be modified.
test_data_1bit_ALU.dat	Test data for 1-bit ALU verification.

test_data_8bit_ALU.dat	Test data for 8-bit ALU verification.
golden_data_1bit_ALU.dat	Golden data for 1-bit ALU verification.
golden_data_8bit_ALU.dat	Golden data for 8-bit ALU verification.

3. Scoring:

3.1 1-bit ALU operation **[40%]**

All of the result should be generated correctly, and you will get the following message in ModelSim simulation.

```
# ------------Stage 1 : 1-bit ALU Simulation-------------
#
# --And Operation--
#
# Pass!
#
# --Or Operation--
#
# Pass!
#
# --Nand Operation--
#
# Pass!
#
# --Nor Operation--
#
# Pass!
#
# --Add Operation--
#
# Pass!
#
# --Sub Operation--
#
# Pass!
#
# --Slt Operation--
#
# Pass!
#
```

Fig. 4. Simulation result for 1-bit ALU operations.

3.2 8-bit ALU bitwise operation **[30%]**

All of the result of bitwise operations, which are And, Or, Nand, and Nor, should be generated correctly, and you will get the following message in ModelSim simulation. Please construct the 8-bit ALU circuit with your 1-bit ALU modules. Otherwise, you can just get half of the points.

```
------------Stage 2 : 8-bit ALU bitwise operation Simulation-------------

--And Operation--

Pass!

--Or Operation--

Pass!

--Nand Operation--

Pass!

--Nor Operation--

Pass!
```

Fig. 5. Simulation result for 8-bit ALU bitwise operations.

3.3 8-bit ALU arithmetic operation **[30%]**

All of the result of arithmetic operations, which are Add, Sub, and SLT, should be generated correctly, and you will get the following message in ModelSim simulation. Please construct the 8-bit ALU circuit with your 1-bit ALU modules. Otherwise, you can just get half of the points.

```
------------Stage 3 : 8-bit ALU arithmetic operation Simulation-------------
--Add Operation--
Pass!
--Sub Operation--
Pass!
--Slt Operation--
Pass!
```

Fig. 6. Simulation result for 8-bit ALU arithmetic operations.

4. Submission:

4.1 Submitted files

You should classify your files into two directories and compress them to .zip format. The naming rule is HW1_studentID_name.zip. If your file is not named according to the naming rule, you will lose five points.

	RTL category
*.v	All of your Verilog RTL code
	Documentary category
*.pdf	The report file of your design (in pdf).

4.2 Report file

Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible.

4.3 Note

Please submit your .zip file to folder HW1 in moodle.

Deadline: 2022/3/14 23:55

If you have any problem, please contact TA by email

p76091187@gs.ncku.edu.tw

lt2es.93039@gmail.com