林庭寫
111550006
2024年3月21日

Lab 1

ALU & register_file implementation

1.Architecture Diagrams

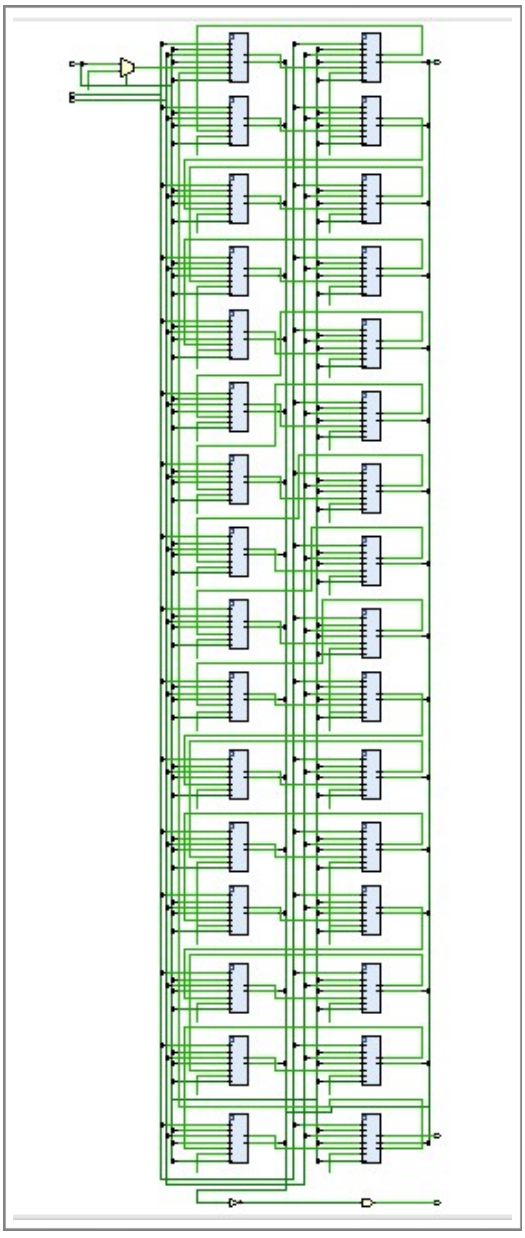

Figure 1. Architecture Diagram

2. Answer the Following Questions

a. How the overflow is calculated?

[reference]

Simply assign overflow = carry_in ^ carry_out. We can get the result by analyzing two possible conditions:

1. Two negative numbers are added.
2. Two positive numbers are added.

Hence, considering the carry_in in MSB ALU, which is the operands, and applying the two scenarios, we can know that overflow = 1 when:

1. If carry_in == 1 and two positive numbers are added.
2. If carry_in == 0 and two negative numbers are added.

b. Why ALU control signal of SUB is "0110" and NOR is "1100"?

By the given hint we know ALU Control = {A_invert, B_invert, operation}, then we know that 10 (ADD) and 00 (AND) are the operation inputs.

For SUB, input b needs to be inverted since we use the 2's complement calculation, which means $a - b = a + b^{C}$.

For NOR, we use Demorgan's low: $(a' \cap b') = (a \cap b)'$

c. (2 cont'd) If you assign different signal to these operation, what problems you may encountered?

The provided way of assigning operation code is well-designed, if I change the code I may need redundant bit to control the inversion of input.

d. True or false: Because the register file is both read and written on the same clock cycle, any MIPS data path using edge-triggered writes must have more than one copy of the register file. Explain the answer.

True. The register file is both read and written on the same clock cycle, which means if there were only one copy of the register file, it would be impossible to read the current values from the register file and simultaneously write new values to it in a single clock cycle without encountering issues.

3.Experimental Result

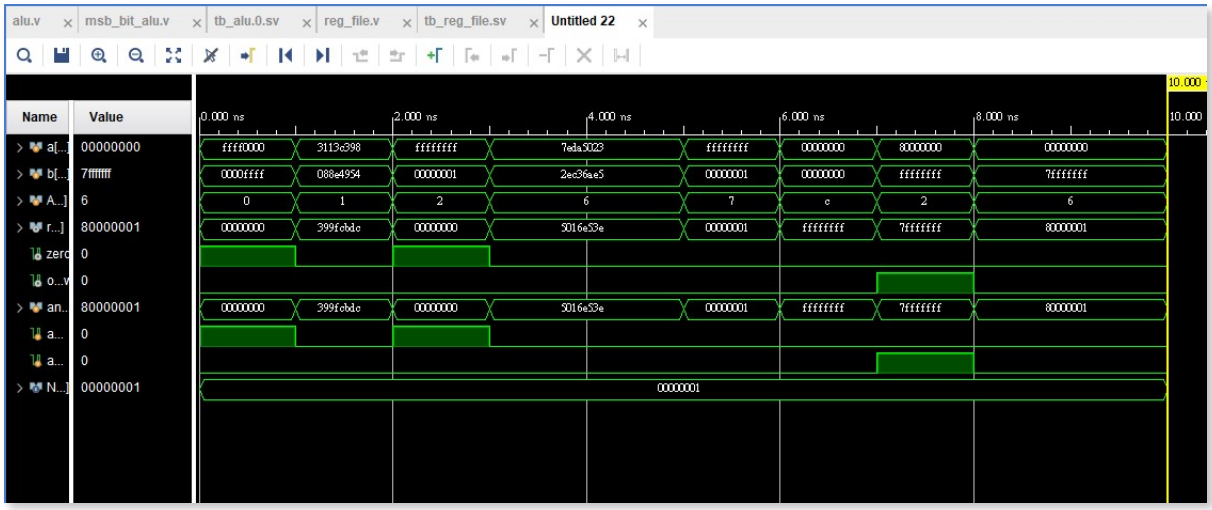

Figure 2. Waveform Screen Shot

Test case						
-2147483648	-1	ADD	0x7fffffff	0	1	// try an overflow case
0x00000000	0x7fffffff	SUB	0x80000001	0	0	// try an small SUB big case

Table 1. Additional Test Case

4.Problems Encountered & Solution

I have tried using assign rather than always by rewrite the code as:

assign result = (operation == 2'b00) ? (ai & bi) :

(operation == 2'b01) ? (ai | bi) :

(operation == 2'b10) ? (sum) :

(operation == 2'b11) ? (less) : 0 ;

This and the always version both works; however, use always is more readable.

5.Feedback

I think this Lab is really helpful and the guidelines are nice.