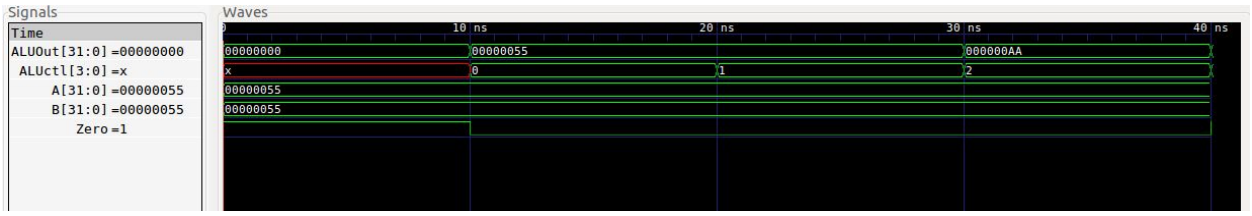
Summary:

I changed the value of 32 bit A and B operands, and see the difference. One interesting thing I notice is: when the A and B have identical value, the ALUout will only access the value once, if A and B have different value, then in the 20ns on the waves, there's a "gap", if we looking the time wave line, in MP-I2, we can clear see the value access is occurred at 10 ns and 20 ns, which means ALU have two input and it is from A and B. if we looking the MP-I, then we can see it only happening at 10 ns, but not 20 ns. I think one explanation is the gtkewave will only consider the value change then it will show the "gap", but if the value will not change, then there will show only one change, which is on the 10 ns, but actually the value access happened twice. But if it is not like this way, then this ALU will automatically ignore the same value access so it can reduce the cycle per instruction.



Details:

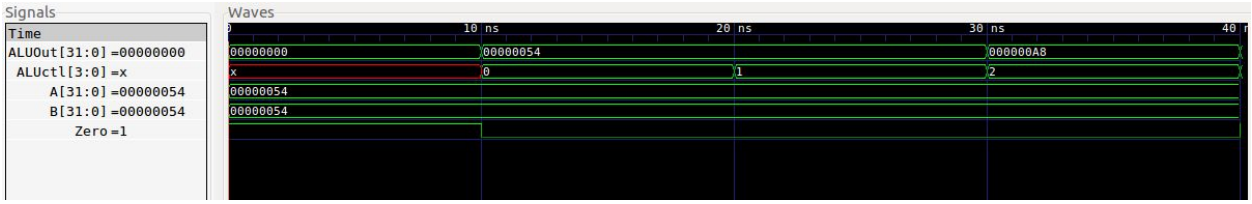
1st change: change A and B with the same value, A:55 -> A:54, B:55->B:54. Except the change of ALUOut and A, B value, nothing is changed much:

Result: A: value changed from 55 to 54

B: value changed from 55 to 54

A == B.

ALUOut: value changed to A and B value(54) in 10 ns, no change at point 20 ns, 10 ns - 30 ns value changed from 55 to 54. 30 ns - 40 ns, value changed from AA to A8 (which is sum of A and B).



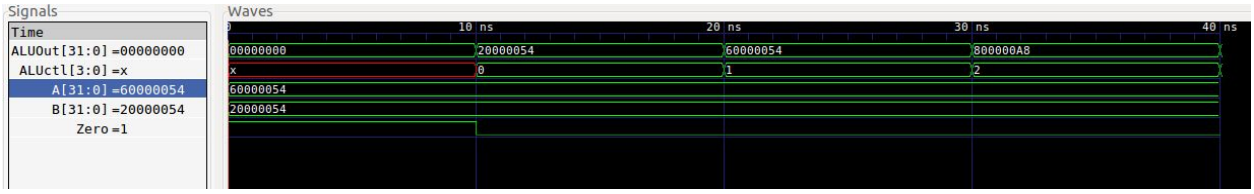
2nd change: change A and B with the same value, A:55 -> A:20000054, B:55->B:60000054. Except the change of ALUOut and A, B value, nothing is changed much:

Result: A: value changed from 55 to 20000054

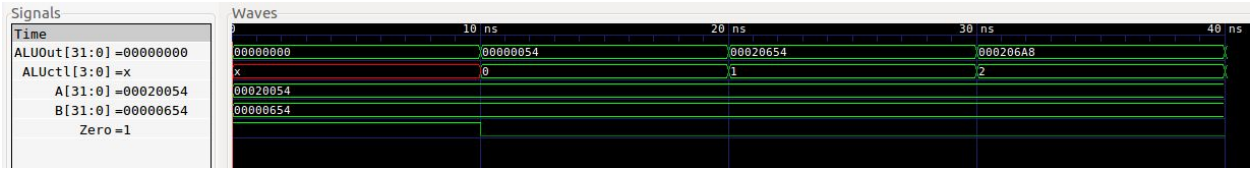
B: value changed from 55 to 60000054

A =\= B.

ALUOut: value changed to A value(20000054) in 10 ns, then changed to 60000054(value of B)at 20 ns, at 30 ns value changed to 800000A8 (which is sum of A and B).



3rd change: change A and B with the same value, A:55 -> A:54, B:55->B:00020654. Except the change of ALUOut and A, B value, nothing is changed much:
Result: A: value changed from 55 to 54
B: value changed from 55 to 00020654
A =\= B.
ALUOut: value changed to A value in 10 ns, then it changed to 00020654(value of B)at 20 ns, and at 30 ns value changed to 000206A8 (which is sum of A and B).



Attach: Verilog code of MIPS ALU:

```
module MIPSALU (ALUctl, A, B, ALUOut, Zero);
 input [3:0] ALUctl;
 input [31:0] A,B;
 output reg [31:0] ALUOut;
 output Zero;
 assign Zero = (ALUOut==0); //Zero is true if ALUOut is 0
 always @(ALUctl, A, B) begin //reevaluate if these change
 case (ALUctl)
 0: ALUOut <= A & B;
 1: ALUOut <= A | B;
 2: ALUOut <= A + B;
 6: ALUOut <= A - B;
 7: ALUOut <= A < B ? 1 : 0;
 12: ALUOut <= ~(A | B); // result is nor
 default: ALUOut <= 0;
 endcase
 end
endmodule
```