Relatório para o trabalho V

Projeto e Simulação de uma ULA RISCV em VHDL

Aluno: Cristiano Tolentino Santos

Matrícula: 21102850

Professor: Ricardo Pezzuol Jacobi

Turma: T02 (2024.2 - 35T23)

Objetivo:

O objetivo deste trabalho é o projeto e a simulação de uma Unidade Lógica e Aritmética (ULA) RISC-V de 32 bits em VHDL. Responsável por executar operações aritméticas, lógicas e de comparação, essenciais para o funcionamento de um processador. As operações implementadas incluem soma, subtração, AND, OR, XOR, deslocamentos lógicos e aritméticos, além de comparações com e sem sinal. A simulação foi realizada no ambiente EDA playground, utilizando o Aldec Riviera Pro.

Diferença entre as comparações com e sem sinal

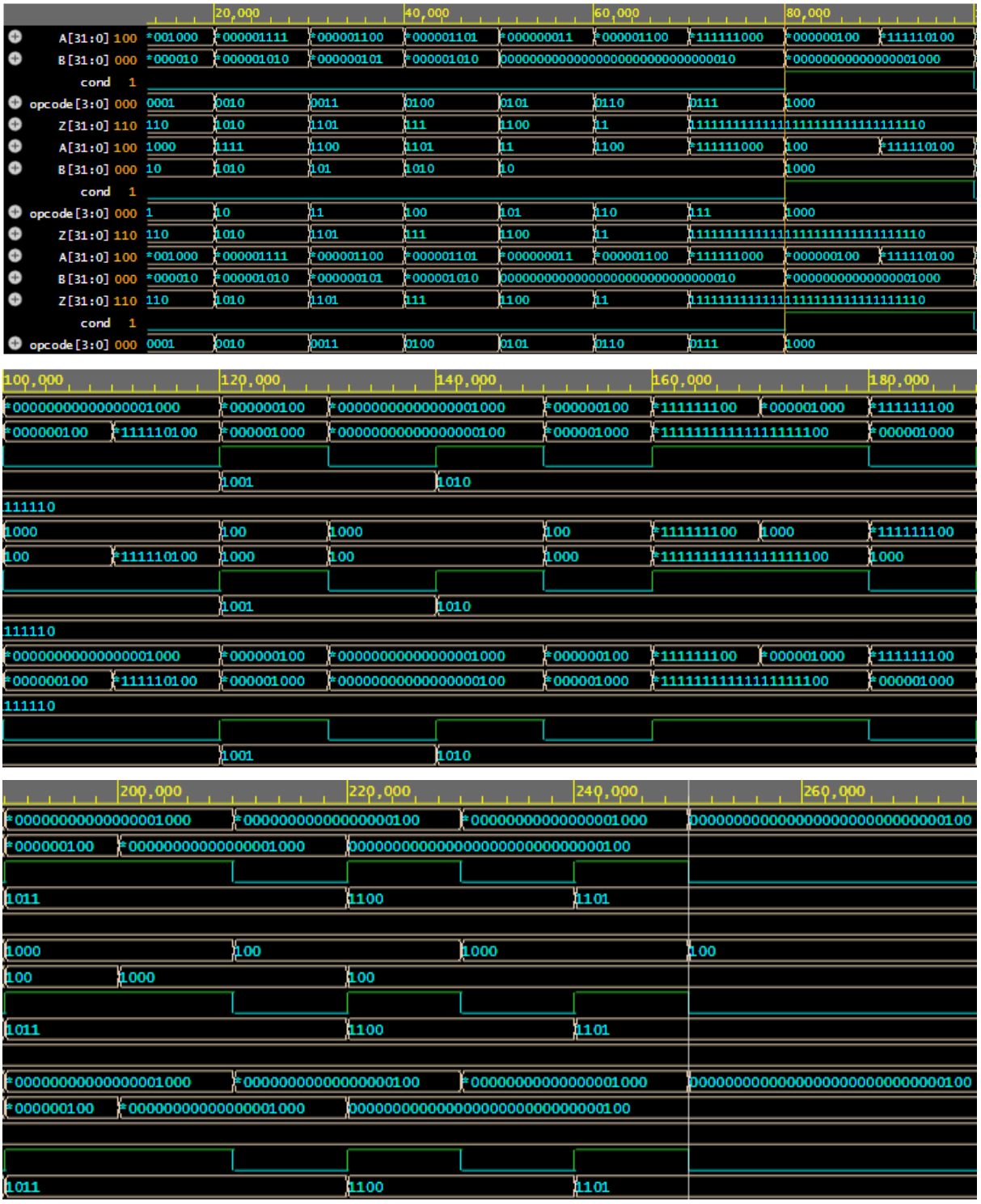
Nas comparações com sinal, o bit mais significativo (MSB) indica o sinal do número: MSB = 0: Número positivo e MSB = 1: Número negativo. O VHDL usa o MSB para determinar a ordem entre dois números. Se os MSBs forem diferentes, o número com MSB = 0 é maior. Caso contrário, a comparação é feita considerando o valor absoluto, levando em conta a estrutura de complemento de dois.

No entanto para as comparações sem sinal, o VHDL trata todos os bits igualmente, sem considerar o MSB. A comparação é feita diretamente entre os valores binários.

Detecção de overflow para operações de ADD e SUB

Para a detecção de overflow em adição é possível verificar a mudança no bit de sinal (MSB) após a soma de forma que, se A e B possuem o mesmo sinal e o resultado tem sinal oposto, então ocorreu um overflow. E para a subtração deve-se verificar quando A é absolutamente maior que B, mas o resultado da subtração é negativo, ou quando B é absolutamente maior que A, mas o resultado da subtração é positivo.

Formas de onda dos sinais



Código

Testbench:

```
Unset
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ula_tb is
end ula_tb;

architecture behavior of ula_tb is
    signal opcode : std_logic_vector(3 downto 0);
    signal A : std_logic_vector(31 downto 0);
    signal B : std_logic_vector(31 downto 0);
    signal Z : std_logic_vector(31 downto 0);
    signal cond : std_logic;
begin
    base: entity work.ulaRV
        generic map (input_size => 32)
        port map (
            A => A,
            B => B,
            opcode => opcode,
            Z => Z,
            cond => cond
        );

    tests: process
    begin
--------------------------------------------------------------------------------

        A <= "00000000000000000000000000000011";
        B <= "00000000000000000000000000000101";
        opcode <= "0000";
        wait for 10 ns;
        report "Teste 1 - Z recebe a soma das entradas A, B: " &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", Z: " & integer'image(to_integer(signed(Z)));

--------------------------------------------------------------------------------

        A <= "00000000000000000000000000001000";
```

```
        B <= "00000000000000000000000000000010";
        opcode <= "0001";
        wait for 10 ns;
        report "Teste 2 - Z recebe A - B:" &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", Z: " & integer'image(to_integer(signed(Z)));

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001111";
        B <= "00000000000000000000000000001010";
        opcode <= "0010";
        wait for 10 ns;
        report "Teste 3 - Z recebe a operacao logica A and B, bit a bit:" &
               "A: " & to_string(std_logic_vector(A)) &
                ", B: " & to_string(std_logic_vector(B)) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001100";
        B <= "00000000000000000000000000000101";
        opcode <= "0011";
        wait for 10 ns;
        report "Teste 4 - Z recebe a operacao logica A or B, bit a bit:" &
               "A: " & to_string(std_logic_vector(A)) &
                ", B: " & to_string(std_logic_vector(B)) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001101";
        B <= "00000000000000000000000000001010";
        opcode <= "0100";
        wait for 10 ns;
        report "Teste 5 - Z recebe a operacao logica A xor B, bit a bit:" &
               "A: " & to_string(std_logic_vector(A)) &
                ", B: " & to_string(std_logic_vector(B)) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000000011";
```

```
        B <= "00000000000000000000000000000010";
        opcode <= "0101";
        wait for 10 ns;
        report "Teste 6 - Z recebe a entrada A deslocada B bits a esquerda:"
&
                "A: " & to_string(std_logic_vector(A)) &
                ", B: " & integer'image(to_integer(unsigned(B))) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));


--------------------------------------------------------------------------

        A <= "00000000000000000000000000001100";
        B <= "00000000000000000000000000000010";
        opcode <= "0110";
        wait for 10 ns;
        report "Teste 7 - Z recebe a entrada A deslocada B bits a direita
sem sinal:" &
               "A: " & to_string(std_logic_vector(unsigned(A))) &
                ", B: " & integer'image(to_integer(unsigned(B))) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));

--------------------------------------------------------------------------

        A <= "11111111111111111111111111111000";
        B <= "00000000000000000000000000000010";
        opcode <= "0111";
        wait for 10 ns;
        report "Teste 8 - Z recebe a entrada A deslocada B bits a direita
com sinal:" &
               "A: " & to_string(std_logic_vector((signed(A)))) &
                ", B: " & integer'image(to_integer(unsigned(B))) &
                ", opcode: " & to_string(std_logic_vector(opcode)) &
                ", Z: " & to_string(std_logic_vector(Z));

--------------------------------------------------------------------------

        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000001000";
        opcode <= "1000";
        wait for 10 ns;
        report "Teste 9 - cond = 1 se A < B, com sinal:" &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", cond: " & std_logic'image(cond);
```

```
        A <= "11111111111111111111111111110100";
        B <= "00000000000000000000000000001000";
        opcode <= "1000";
        wait for 10 ns;
        report "Teste 9 - cond = 1 se A < B, com sinal:" &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1000";
        wait for 10 ns;
        report "Teste 9 - cond = 1 se A < B, com sinal:" &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "11111111111111111111111111110100";
        opcode <= "1000";
        wait for 10 ns;
        report "Teste 9 - cond = 1 se A < B, com sinal:" &
               "A: " & integer'image(to_integer(signed(A))) &
               ", B: " & integer'image(to_integer(signed(B))) &
               ", opcode: " & to_string(std_logic_vector(opcode)) &
               ", cond: " & std_logic'image(cond);

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000001000";
        opcode <= "1001";
        wait for 10 ns;
          report "Teste 10 - cond = 1 se A < B, sem sinal:" &
          "A: " & integer'image(to_integer(unsigned(A))) &
          ", B: " & integer'image(to_integer(unsigned(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1001";
        wait for 10 ns;
          report "Teste 10 - cond = 1 se A < B, sem sinal:" &
          "A: " & integer'image(to_integer(unsigned(A))) &
          ", B: " & integer'image(to_integer(unsigned(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
```

```
-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1010";
        wait for 10 ns;
          report "Teste 11 - cond = 1 se A >= B, com sinal:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000001000";
        opcode <= "1010";
        wait for 10 ns;
          report "Teste 11 - cond = 1 se A >= B, com sinal:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "11111111111111111111111111111100";
        B <= "11111111111111111111111111111100";
        opcode <= "1010";
        wait for 10 ns;
          report "Teste 11 - cond = 1 se A >= B, com sinal:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "11111111111111111111111111111100";
        opcode <= "1010";
        wait for 10 ns;
          report "Teste 11 - cond = 1 se A >= B, com sinal:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "11111111111111111111111111111100";
        B <= "00000000000000000000000000001000";
        opcode <= "1010";
        wait for 10 ns;
          report "Teste 11 - cond = 1 se A >= B, com sinal:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
```

```
-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1011";
        wait for 10 ns;
          report "Teste 12 - cond = 1 se A >= B, sem sinal:" &
          "A: " & integer'image(to_integer(unsigned(A))) &
          ", B: " & integer'image(to_integer(unsigned(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000001000";
        opcode <= "1011";
        wait for 10 ns;
          report "Teste 12 - cond = 1 se A >= B, sem sinal:" &
          "A: " & integer'image(to_integer(unsigned(A))) &
          ", B: " & integer'image(to_integer(unsigned(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000001000";
        opcode <= "1011";
        wait for 10 ns;
          report "Teste 12 - cond = 1 se A >= B, sem sinal:" &
          "A: " & integer'image(to_integer(unsigned(A))) &
          ", B: " & integer'image(to_integer(unsigned(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000000100";
        opcode <= "1100";
        wait for 10 ns;
          report "Teste 13 - cond = 1 se A == B:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1100";
        wait for 10 ns;
          report "Teste 13 - cond = 1 se A == B:" &
          "A: " & integer'image(to_integer(signed(A))) &
```

```
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);

-------------------------------------------------------------------------------

        A <= "00000000000000000000000000001000";
        B <= "00000000000000000000000000000100";
        opcode <= "1101";
        wait for 10 ns;
          report "Teste 14 - cond = 1 se A != B:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);
        A <= "00000000000000000000000000000100";
        B <= "00000000000000000000000000000100";
        opcode <= "1101";
        wait for 10 ns;
          report "Teste 14 - cond = 1 se A != B:" &
          "A: " & integer'image(to_integer(signed(A))) &
          ", B: " & integer'image(to_integer(signed(B))) &
          ", opcode: " & to_string(std_logic_vector(opcode)) &
          ", cond: " & std_logic'image(cond);

-------------------------------------------------------------------------------

        wait;
    end process;
end behavior;
```

Design:

Unset

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ulaRV is
generic (input_size : natural := 32);
    port (
        opcode : in std_logic_vector(3 downto 0);
        A : in std_logic_vector(input_size-1 downto 0);
```

```
        B : in std_logic_vector(input_size-1 downto 0);
        Z : out std_logic_vector(input_size-1 downto 0);
        cond   : out std_logic);
end ulaRV;

architecture behavior of ulaRV is
begin
    process (A, B, opcode)
    begin
        case opcode is
            when "0000" => Z <= std_logic_vector(signed(A) + signed(B));
            when "0001" => Z <= std_logic_vector(signed(A) - signed(B));
            when "0010" => Z <= A and B;
            when "0011" => Z <= A or B;
            when "0100" => Z <= A xor B;
            when "0101" => Z <= std_logic_vector(shift_left(unsigned(A),
to_integer(unsigned(B))));
            when "0110" => Z <= std_logic_vector(shift_right(unsigned(A),
to_integer(unsigned(B))));
            when "0111" => Z <= std_logic_vector(shift_right(signed(A),
to_integer(unsigned(B))));
            when "1000" => cond <= '1' when signed(A) < signed(B) else '0';
            when "1001" => cond <= '1' when unsigned(A) < unsigned(B) else
'0';
            when "1010" => cond <= '1' when signed(A) >= signed(B) else '0';
            when "1011" => cond <= '1' when unsigned(A) >= unsigned(B) else
'0';
            when "1100" => cond <= '1' when A = B else '0';
            when "1101" => cond <= '1' when A /= B else '0';
            when others =>
                Z <= (others => '0');
                cond <= '0';
        end case;
    end process;
end behavior;
```