Simulação da ULA em VHDL - OAC

Luan José de Almeida Cardoso 14/0150161

1 Problema

O objetivo desde projeto é a implementação de uma ula em VHDL, implementando principalmente as funções :

- Soma
- Subtração
- AND
- OR
- XOR
- Shift Logico a Esquerda (SLL)
- Shift Lógico a Direita (SRL)
- Shift Aritmético a Direita (SRA)
- Condições de Menor (SLT)
- Condição de Menor sem sinal (SLTU)
- Condição de Maior ou igual (SGE)
- Condição de Maior ou igual sem sinal (SGEU)
- Condição de Igualdade (SEQ)
- Condição de diferença (SNE)

2 Implementação

A implementação do sistema com a linguagem se torna bem mais simples, foi criado então o código, que inclui a interface e o testbench do sistema apresentado na Secção 5. Além da biblioteca *numeric_std* também foi utilizada a *std_logic_unsigned* para poder realizar as operações de soma e subtração com vetores de forma direta. Dentre as funções implementadas podemos citar algumas diferenças no código e em sua lógica de implementação:

- Shifts: Com o auxilio da biblioteca *numeric_std* a diferenciação dentre o shift logico ou aritmético se dá pelo tipo de entra para um modelo com sinal teremos um shift aritmético, já mudando-a para sem sinal(unsigned) realizamos uma operação logica.
- Para os modelos de condicionamento (Comparação), temos também que tem relevância de sinal ou não, a diferença entre eles está principalmente no ”range”da análise para por exemplo o **SGEU** teríamos uma maior gama de análises modulares do *0* até seu bit mais significativo.

3 Resultados

Os resultados Gerais podem ser visualizados na Figura 2, onde temos a saída principal do University Waveform, criada pelo Quartus.

Alguns dos testes ainda não foram realizados dentro dela, por isso serão apresentados mais a frente.

Na Figura 1 temos um Zoom da simulação anterior oque nos permite visualizar de forma mais explicita alguns detalhes, podemos ver que o sistema se comporta de forma coerente para todas as ações requisitadas. Neste exemplo temos principalmente valores positivos, que apenas demonstram uma simulação de teste inicial.

De forma adicional ao sistema como não foi explicitado quais seriam as saídas para um opcode que não estivesse dentro do range, Z é assumido como 0, como pode ser observado Na Figura 3.

O segundo teste para o sistema foi aplicar uma soma geral negativa apresentada na Figura 4.

Como sistema continuou reagindo bem as entradas solicitadas foi feito um ultimo teste, tentando aplicar uma entrada de overflow principalmente a operação de soma do sistema -1 + -1, que deveria sair do range de 32bits definidos. O resultado encontrado está na Figura 5.

Neste momento foi observado um pequeno erro dentro do código, que envolve principalmente as operações de Soma e subtração para a saída *zero* apresentado no Código 5.1.

4 Imagens

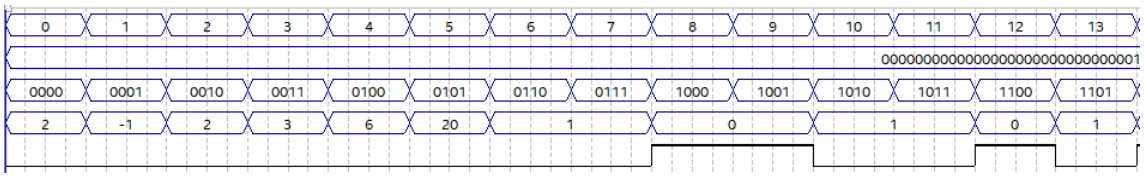

Figura 1: Saída Inicial Positiva

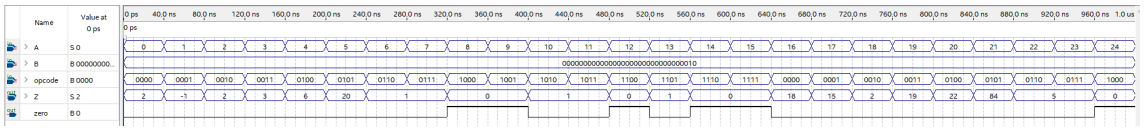

Figura 2: Saída Geral

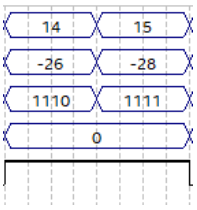

Figura 3: Itens fora do Opcode

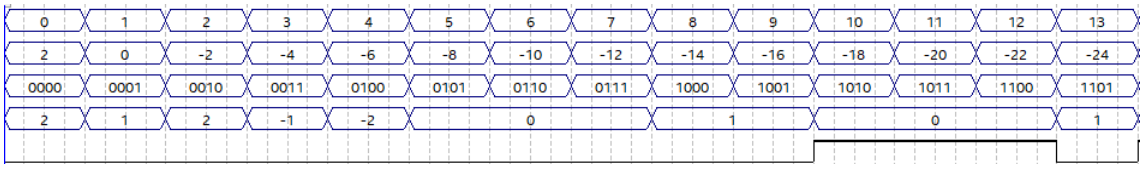

Figura 4: Respostas a entrada Negativa

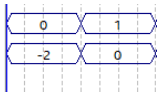

Figura 5: Respostas ao Overflow

5 Código

5.1 Code fix

```
when "0000" =>
  Z <= A + B;
  if A+B = 0 then
    zero <= '1';
  else
    zero <= '0';
  end if;
when "0001" =>
  Z <= A - B;
  if A-B = 0 then
    zero <= '1';
  else
    zero <= '0';
  end if;
```

5.2 Full Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity ularv is
        port (
                opcode : in std_logic_vector(3 downto 0);
                A,B    : in std_logic_vector(31 downto 0);
                Z              : out std_logic_vector( 31 downto 0);
                zero   : out std_logic
                );
end ularv;


architecture ula_arc OF ularv is

begin

        process (A,B,opcode)
        begin

        if A = 0 and B = 0 then
```

```
        zero <= '1';
else
        zero <= '0';
end if;

case opcode is

when "0000" =>
        Z <= A + B;
        if A+B = 0 then
                zero <= '1';
        else
                zero <= '0';
        end if;
when "0001" =>
        Z <= A - B;
        if A-B = 0 then
                zero <= '1';
        else
                zero <= '0';
        end if;

when "0010" =>
        Z <= A AND B;
        if A = 0 or B = 0 then
                zero <= '0';
        end if;

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))));
        if std_logic_vector(shift_left(unsigned(A), to_integer(unsigned(B)))) = 0 then
                zero <= '0';
        end if;

when "0110" =>
        Z <= std_logic_vector(shift_right(unsigned(A),to_integer(unsigned(B))));
        if std_logic_vector(shift_right(unsigned(A),to_integer(unsigned(B)))) = 0 then
                zero <= '0';
        end if;

when "0111" =>
        Z <= std_logic_vector(shift_right( signed(A),to_integer(unsigned(B))));
        if std_logic_vector(shift_right( signed(A),to_integer(unsigned(B)))) = 0 then
                zero <= '0';
        end if;

when "1000" =>
        if A < B then
                Z <= "00000000000000000000000000000001";
                zero <= '0';
        else
                Z <= "00000000000000000000000000000000";
                zero <='1';
        end if;
```

```
        when "1001" =>
                if unsigned(A) < unsigned(B) then
                        Z <= "00000000000000000000000000000001";
                        zero <= '0';
                else
                        Z <= "00000000000000000000000000000000";
                        zero <='1';
                end if;
        when "1010" =>
                if A >= B then
                        Z <= "00000000000000000000000000000001";
                        zero <= '0';
                else
                        Z <= "00000000000000000000000000000000";
                        zero <='1';
                end if;
        when "1011" =>
                if  unsigned(A) >= unsigned(B) then
                        Z <= "00000000000000000000000000000001";
                        zero <= '0';
                else
                        Z <= "00000000000000000000000000000000";
                        zero <= '1';
                end if;
        when "1100" =>
                if A = B then
                        Z <= "00000000000000000000000000000001";
                        zero <= '0';
                else
                        Z <= "00000000000000000000000000000000";
                        zero <='1';
                end if;
        when "1101" =>
                if A /= B then
                        Z <= "00000000000000000000000000000001";
                        zero <= '0';
                else
                        Z <= "00000000000000000000000000000000";
                        zero <='1';
                end if;
        when others =>
                        Z <= "00000000000000000000000000000000";
                        zero <='1';
        end case;

        end process;

end ula_arc;
```