Calculadora de 8 Bits

Calculadora

Nessa atividade, implementaremos uma calculadora de 8 bits que efetue operações de soma e subtração.

Especificações

Entradas:

- A, B: palavras binárias de 8 bits cada.
- OP: bit que indica se será feito soma ou subtração.

Saídas:

- S: palavra binária de 8 bits (pode ser o resultado de $A + B$ ou de $A - B$).
- OP: bit que indica se será feito soma ou subtração.
- Luz 1: acende se for uma soma.
- Luz 2: acende se for uma subtração.

Operação: Circuito combinacional que permite selecionar qual das duas operações será realizada.

Descrição estrutural: Assim como no projeto do somador/subtrator, esse sistema utilizará uma descrição estrutural com módulos reusáveis, como um somador completo de 1 bit.

Para esse somador, temos:

A soma é calculada como o XOR das três entradas A, B e C_{in}:

$$S = A \oplus B \oplus C_{\text{in}}$$

O *carry out* é gerado quando pelo menos duas das três entradas são 1:

$$C_{\text{out}} = (A \cdot B) + (A \cdot C_{\text{in}}) + (B \cdot C_{\text{in}})$$

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder is
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        Cin : in STD_LOGIC;
        Sum : out STD_LOGIC;
        Cout : out STD_LOGIC
    );
end full_adder;

architecture Behavioral of full_adder is
begin
    Sum <= A xor B xor Cin;
    Cout <= (A and B) or (B and Cin) or (A and Cin);
end Behavioral;
```

Entidade

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entidade da calculadora de 8 bits
entity calculator_8bit is
    Port (
        A : in STD_LOGIC_VECTOR(7 downto 0);     --
           Operando A
        B : in STD_LOGIC_VECTOR(7 downto 0);     --
           Operando B
        OP : in STD_LOGIC;                       -- 0
           para soma, 1 para subtra\c{c}\~{a}o
        S : out STD_LOGIC_VECTOR(7 downto 0);    --
           Resultado
        Luz1 : out STD_LOGIC;                    -- Luz
           para soma
```

```
        Luz2 : out STD_LOGIC                    -- Luz para subtra\c{c}\~{a}o
    );
end calculator_8bit;
```

Arquitetura

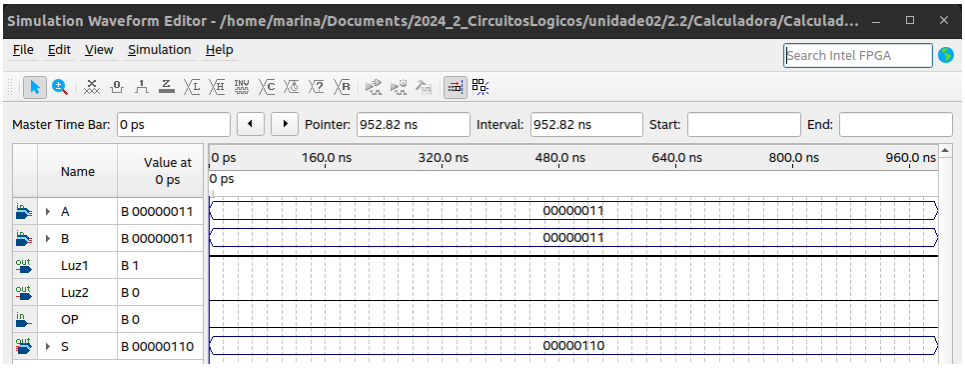
```
architecture Structural of calculator_8bit is

    signal B_inverted : STD_LOGIC_VECTOR(7 downto 0);
    signal carry : STD_LOGIC_VECTOR(7 downto 0);

begin
    -- Inverte B se OP = '1' (realizando o complemento de dois)
    B_inverted <= B xor (OP & OP & OP & OP & OP & OP & OP & OP);

    -- L\'ogica para definir as luzes de soma e subtra\c{c}\~{a}o
    Luz1 <= not OP;  -- Acende quando OP = '0' (soma)
    Luz2 <= OP;      -- Acende quando OP = '1' (subtra\c{c}\~{a}o)

    -- Inst\^ancias dos somadores completos
    FA0: entity work.full_adder port map (
        A => A(0),
        B => B_inverted(0),
        Cin => OP,
        Sum => S(0),
        Cout => carry(0)
    );

    FA1: entity work.full_adder port map (
        A => A(1),
        B => B_inverted(1),
        Cin => carry(0),
        Sum => S(1),
        Cout => carry(1)
    );

```

```
    FA2: entity work.full_adder port map (
        A => A(2),
        B => B_inverted(2),
        Cin => carry(1),
        Sum => S(2),
        Cout => carry(2)
    );

    FA3: entity work.full_adder port map (
        A => A(3),
        B => B_inverted(3),
        Cin => carry(2),
        Sum => S(3),
        Cout => carry(3)
    );

    FA4: entity work.full_adder port map (
        A => A(4),
        B => B_inverted(4),
        Cin => carry(3),
        Sum => S(4),
        Cout => carry(4)
    );

    FA5: entity work.full_adder port map (
        A => A(5),
        B => B_inverted(5),
        Cin => carry(4),
        Sum => S(5),
        Cout => carry(5)
    );

    FA6: entity work.full_adder port map (
        A => A(6),
        B => B_inverted(6),
        Cin => carry(5),
        Sum => S(6),
        Cout => carry(6)
    );

    FA7: entity work.full_adder port map (
        A => A(7),
        B => B_inverted(7),
```

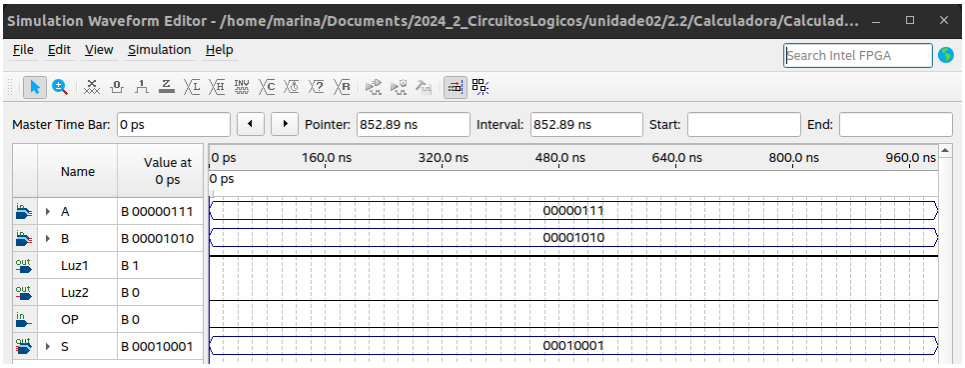
```
74          Cin => carry(6),
75          Sum => S(7),
76          Cout => open    -- N\~{a}o precisamos usar o
                carry final
77      );
78
79  end Structural;
```

Plano de Simulações e Testes

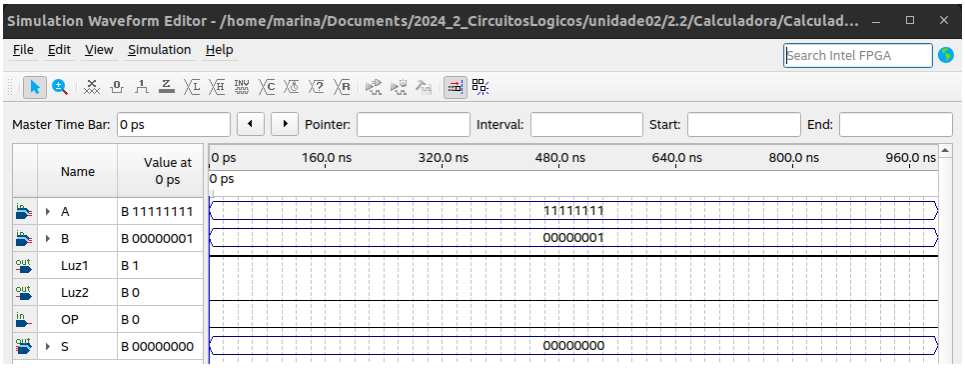
- **Caso 1:** Soma de dois positivos.



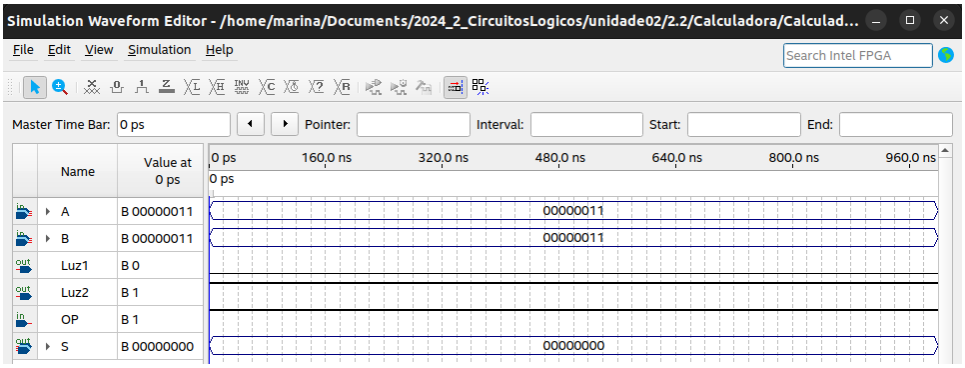
- **Caso 2:** Soma com *carry*.



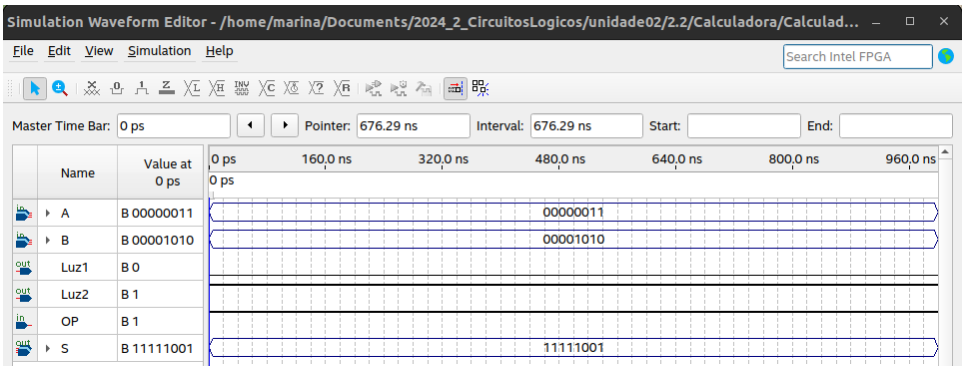
- **Caso 3:** Soma com *overflow*.



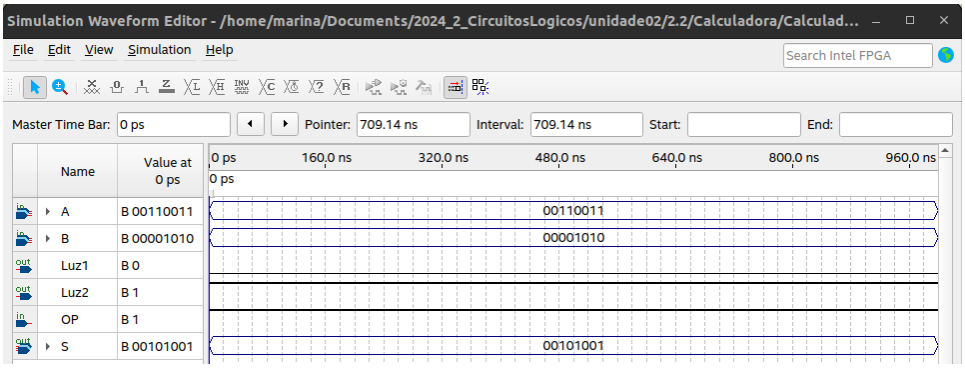
- **Caso 4:** Subtração com resultado não negativo.



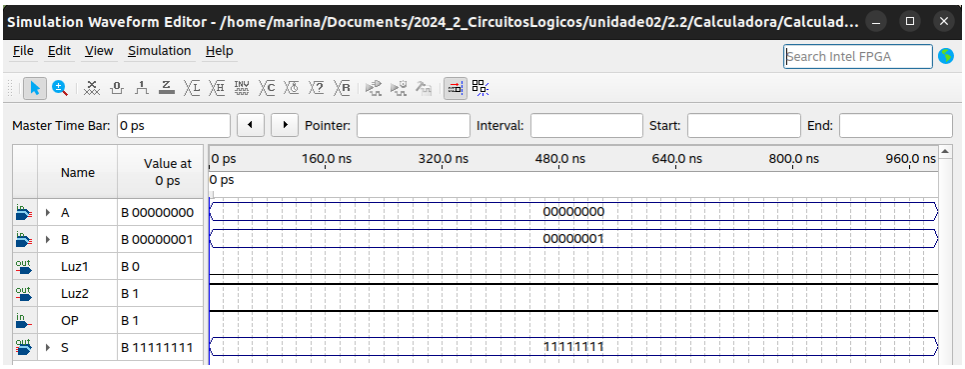
- **Caso 5:** Subtração com resultado negativo.



- **Caso 6:** Subtração com *carry*.



- **Caso 7:** Subtração com underflow.



Conclusão

Com os resultados dos testes, podemos ver que o sistema funciona conforme o esperado.