PROJETO FINAL - ULA MIPS SIMPLIFICADA COM PROCESSO DE REDUNDÂNCIA

Universidade Federal de Santa Catarina (UFSC)
Ciências da Computação
Sistemas Digitais (INE5406)
Docentes: José Luís Güntzel e Ismael Seidel.
Discentes: Carlos Henrique Zanon, Heloisa Jonck Hammes, Leonardo Fonseca Franchini, Lucas Ryan Carneiro, Rebeca Ayumi Komatsu e Viccenzo Escarrone.

1 Redundância

2 Objetivos

3 ULA

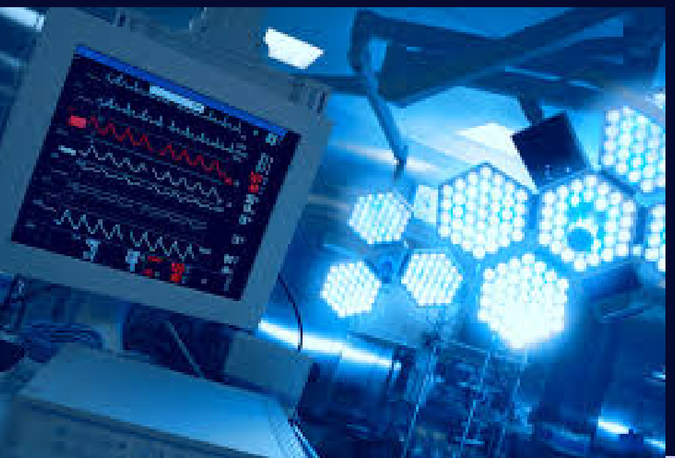
4 Processo de redundância

5 Bloco operativo

6 Bloco de controle

7 Testbench

8 Conclusão



Redundância?

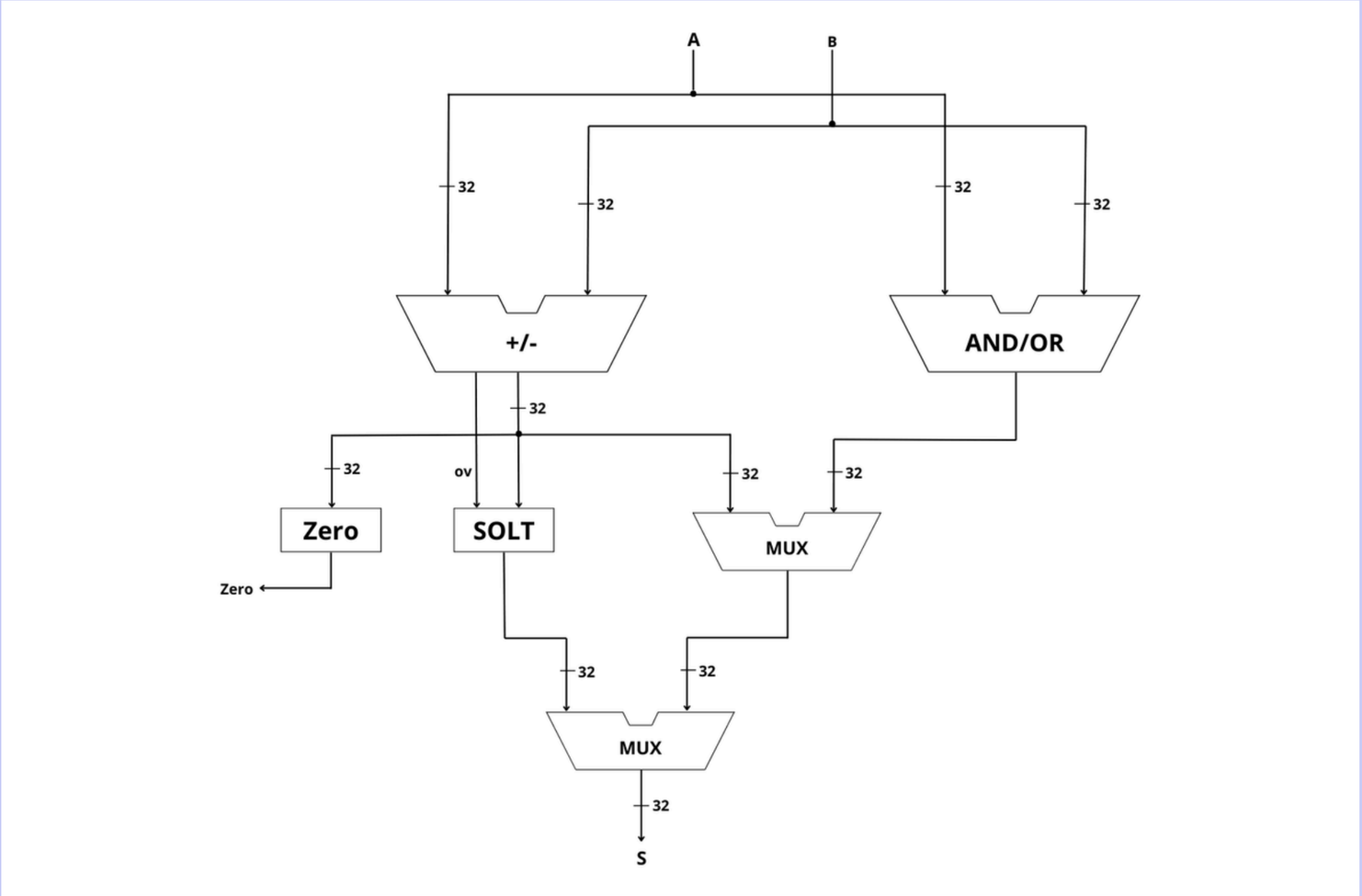
Procedimento que se baseia na repetição do processamento, com objetivo de assegurar uma maior confiabilidade do resultado fornecido.

2 Objetivos

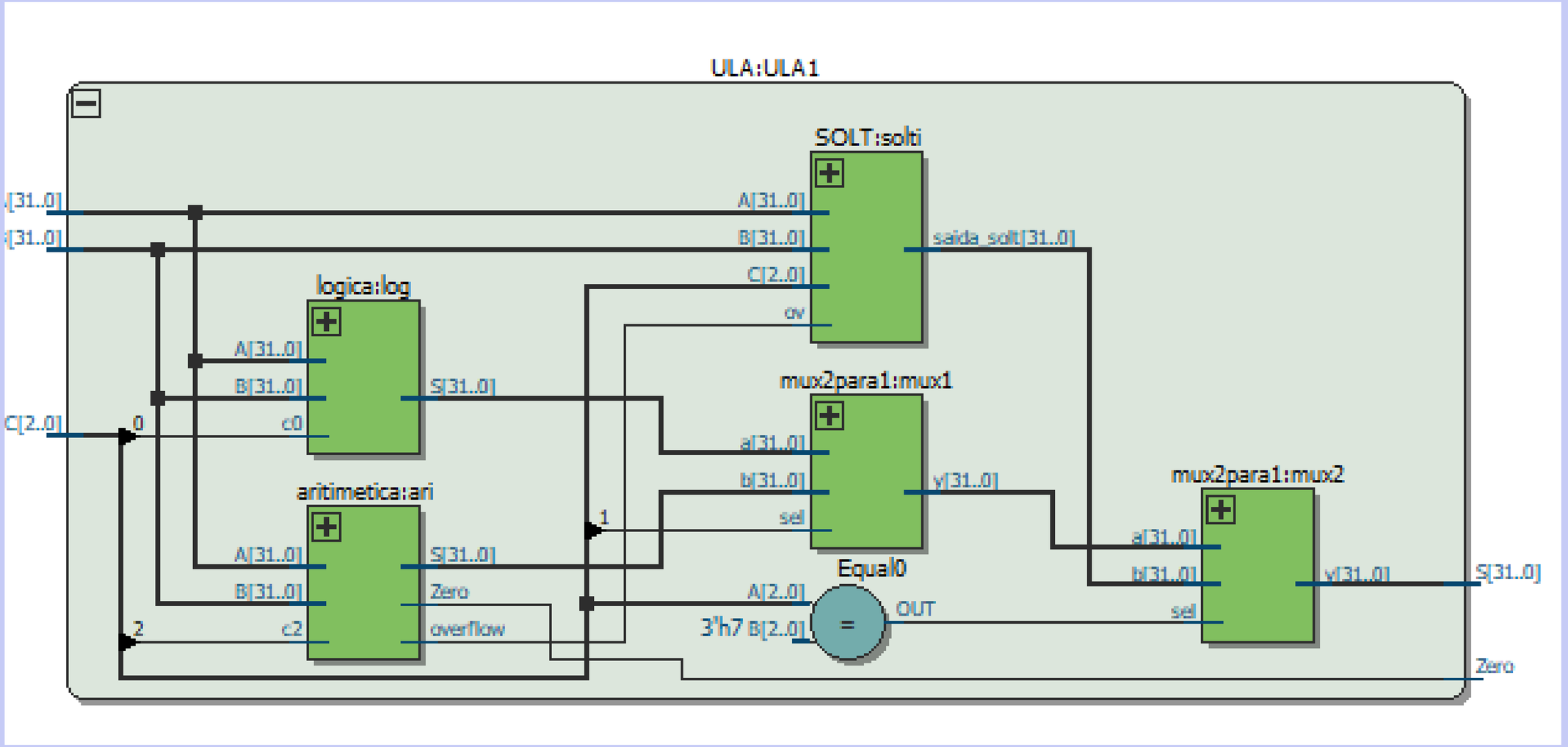
- **Aumento da Confiabilidade**
- **Tolerância a Falhas**
- **Detecção e Correção de Erros**

3 ULA (Unidade Lógica Aritmética)

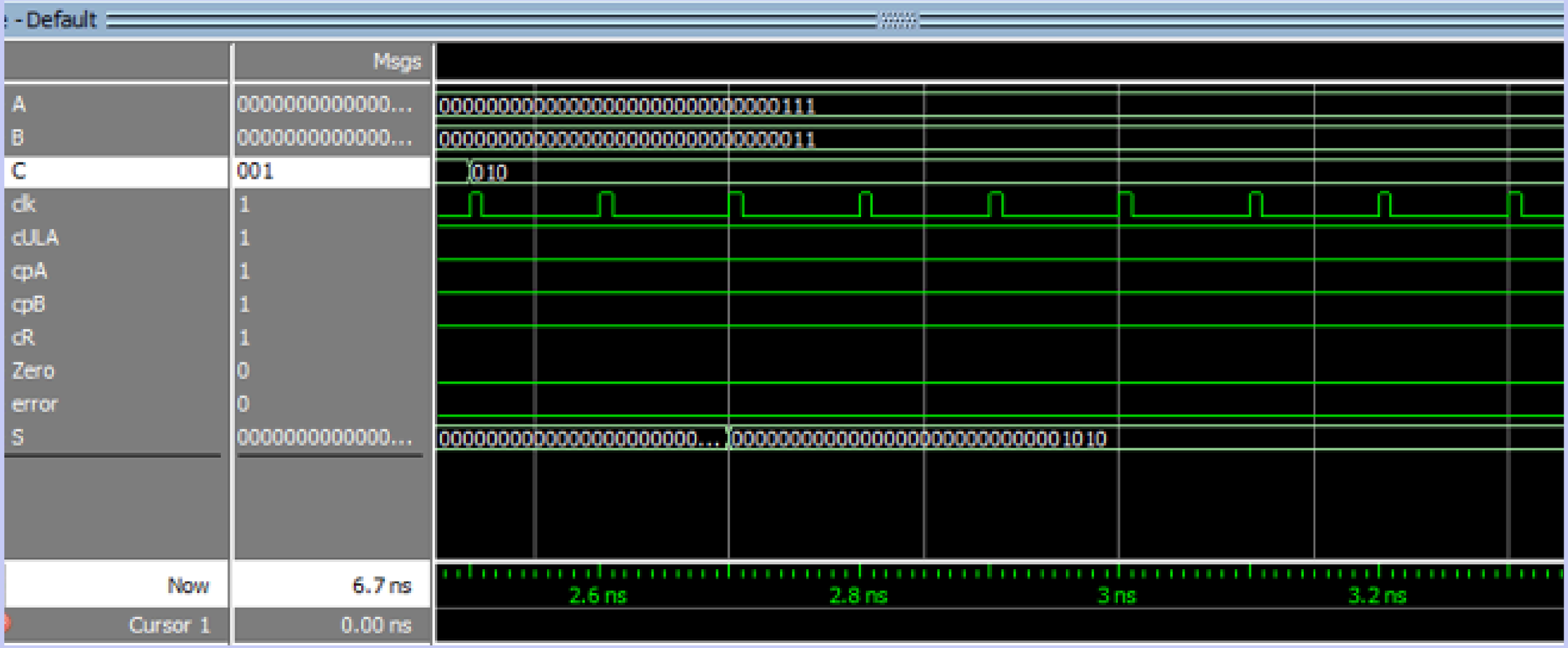
- Esquemático nível RT da ULA



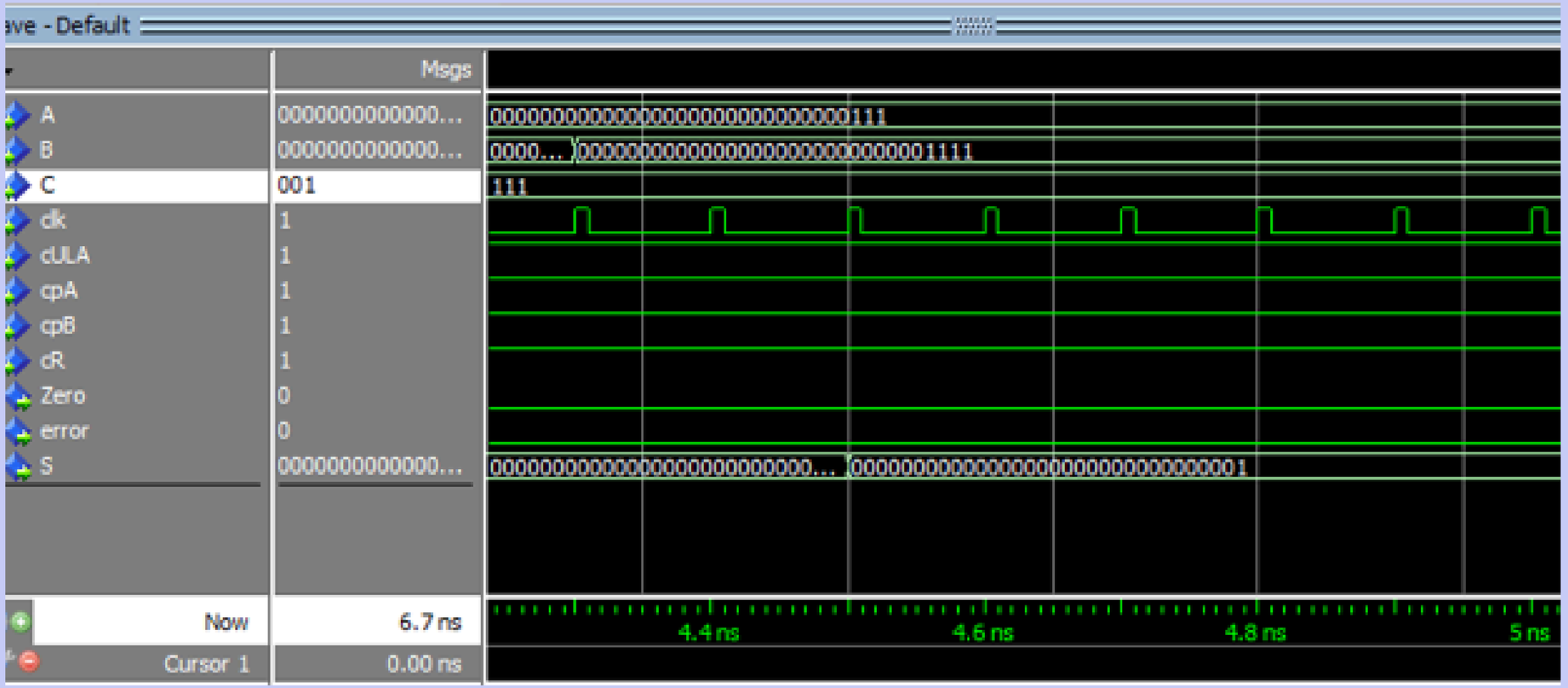
3 ULA (Unidade Lógica Aritmética)



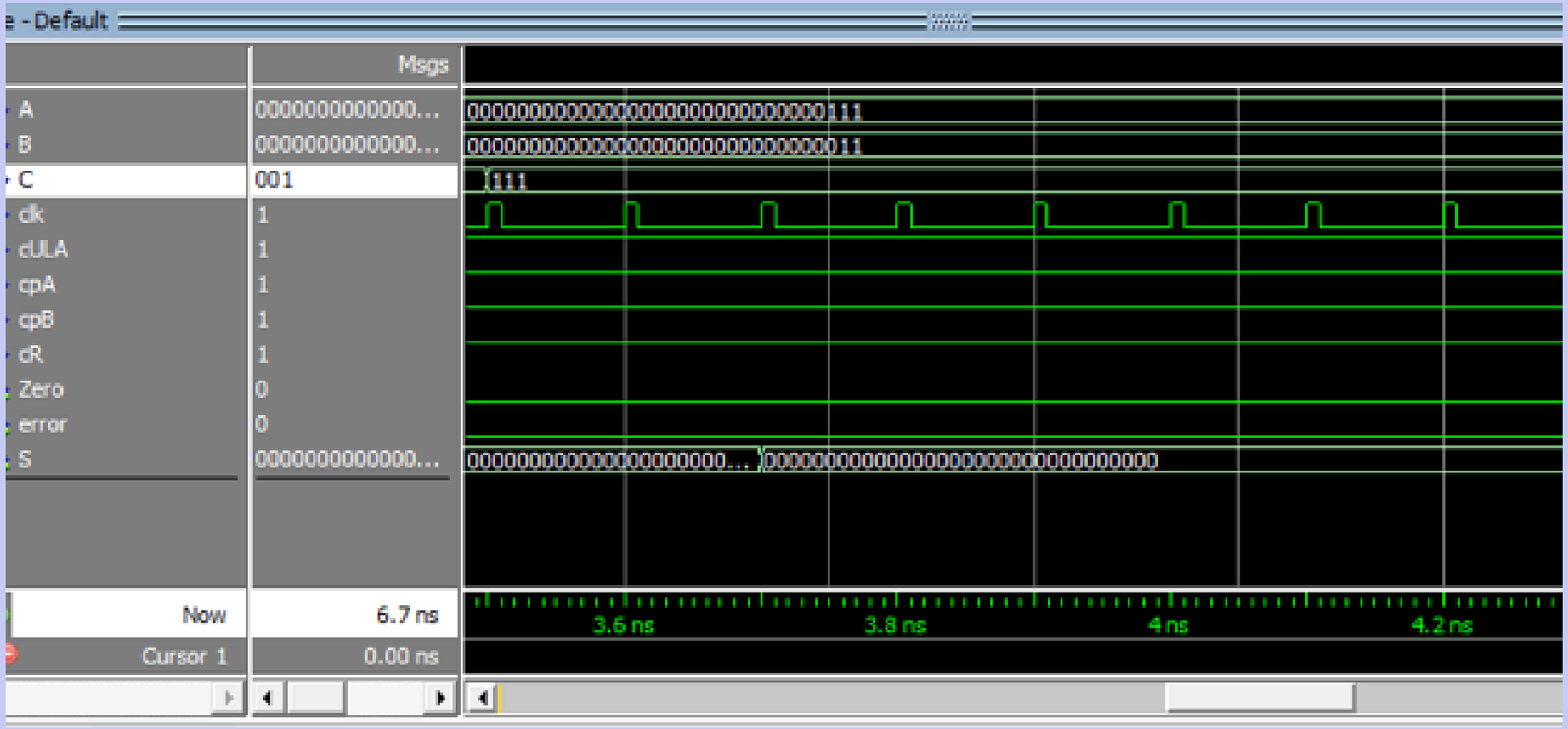
3 ULA (Unidade Lógica Aritmética)



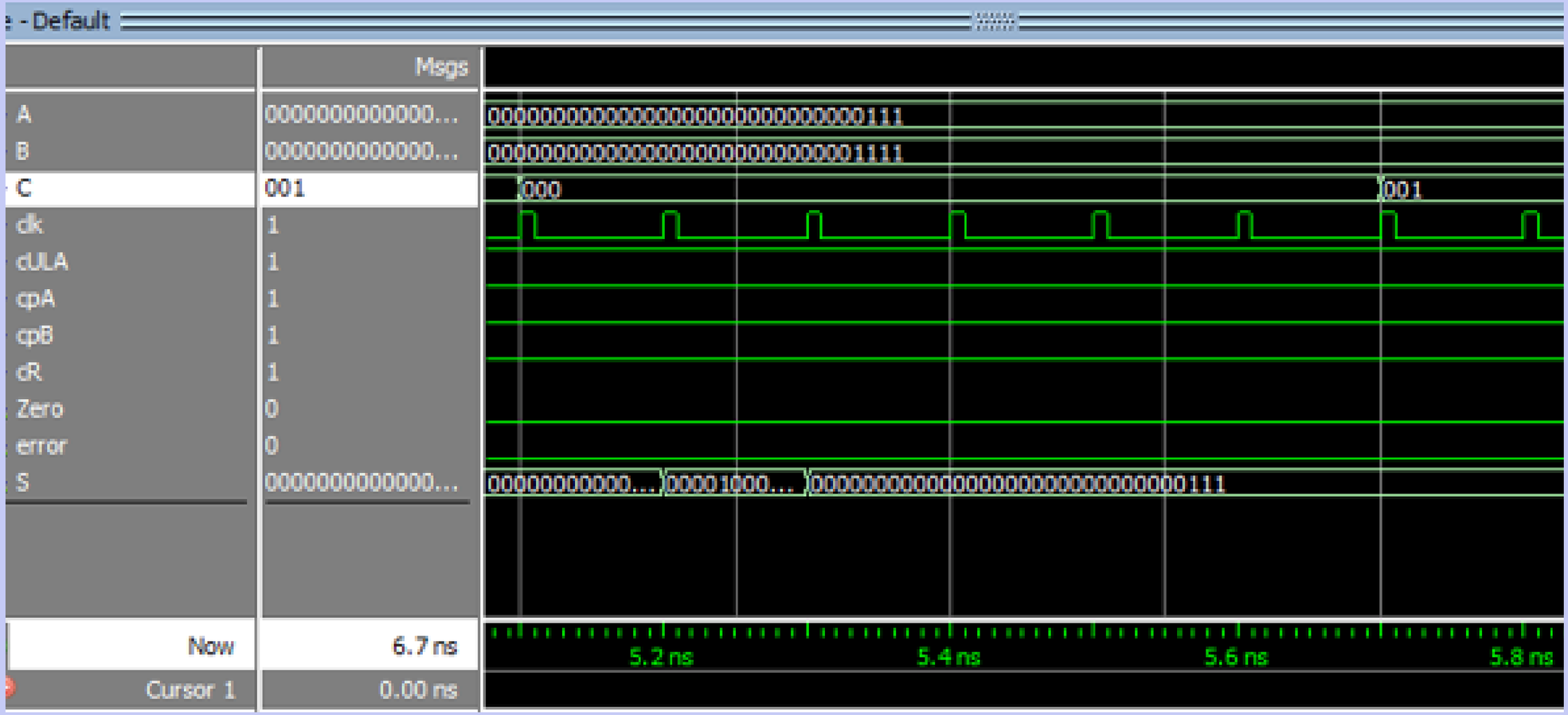
3 ULA (Unidade Lógica Aritmética)



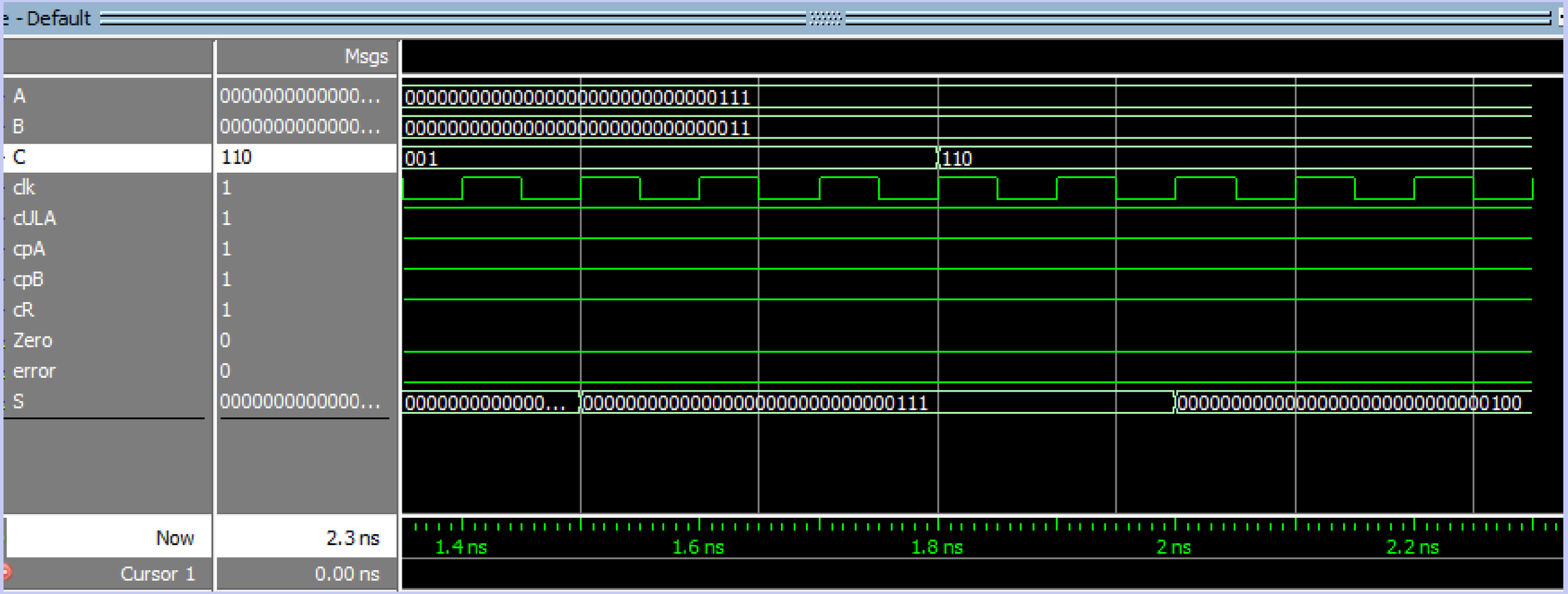
3 ULA (Unidade Lógica Aritmética)



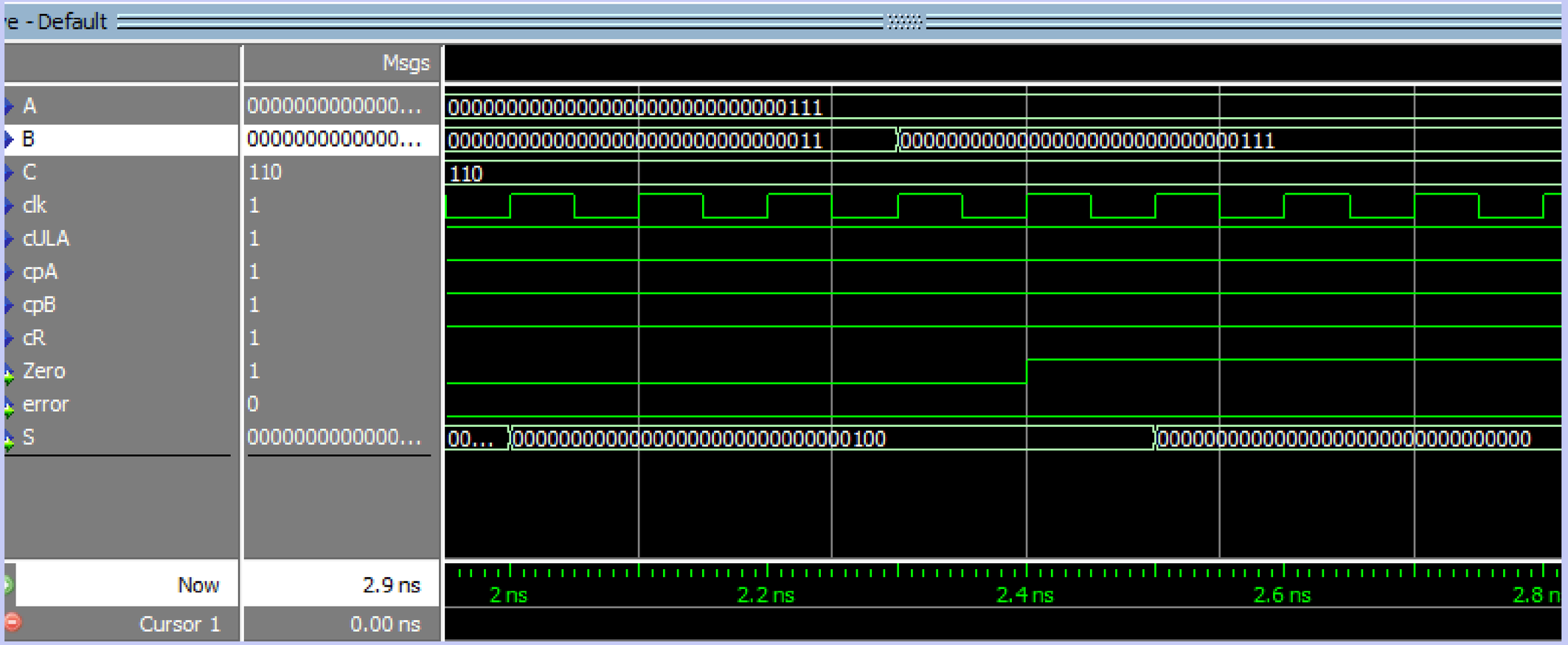
3 ULA (Unidade Lógica Aritmética)



3 ULA (Unidade Lógica Aritmética)

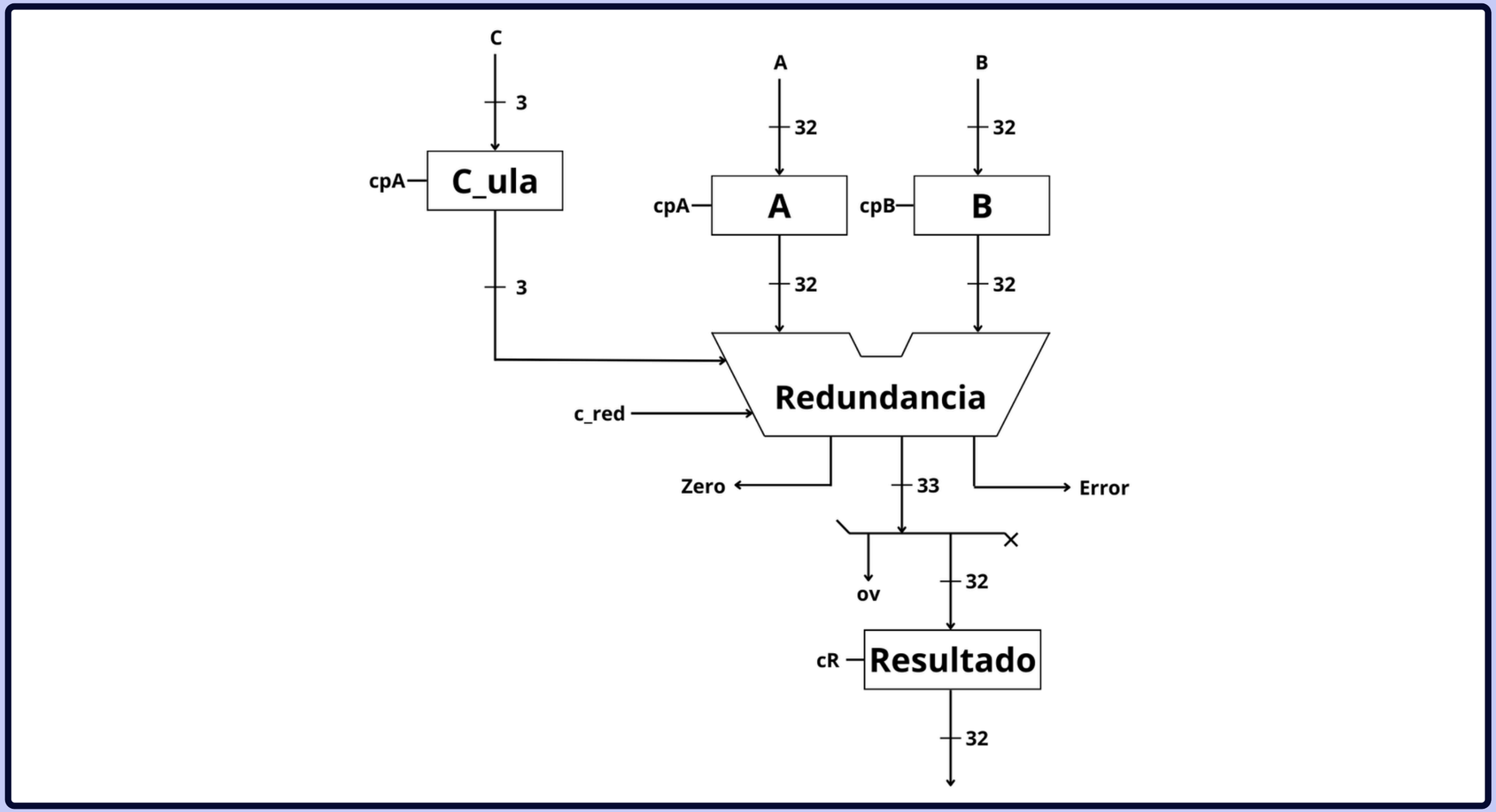


3 ULA (Unidade Lógica Aritmética)

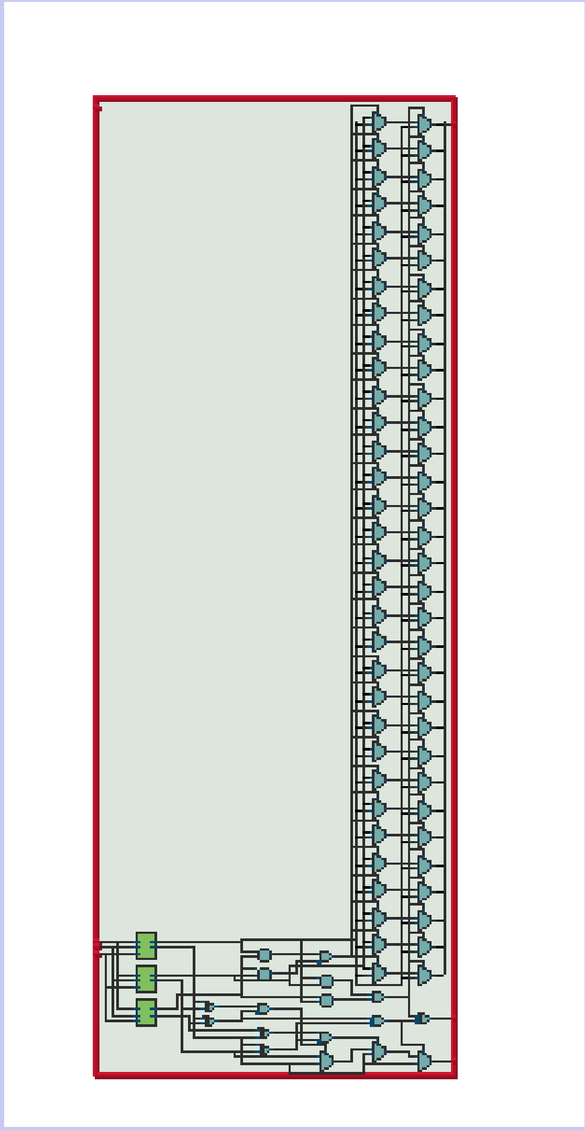


4 Processo de redundância

- Esquemático nível RT do processo de redundância



4 Processo de redundância



4 Processo de redundância

```
begin
   process(c_red)
   begin

      -- lógica de redundancia do Zero
      if (Zero1 = Zero2) and (Zero3 = Zero1) then
         error1 <= '0';
         Zero <= Zero1;
      elsif (Zero1 = Zero2) and (Zero3 /= Zero1) then
         Zero <= Zero1;
         error1 <= '1';
      elsif (Zero1 = Zero3) and (Zero2 /= Zero1) then
         Zero <= Zero1;
         error1 <= '1';
      else
         Zero <= Zero2;
         error1 <= '1';
      end if;
```

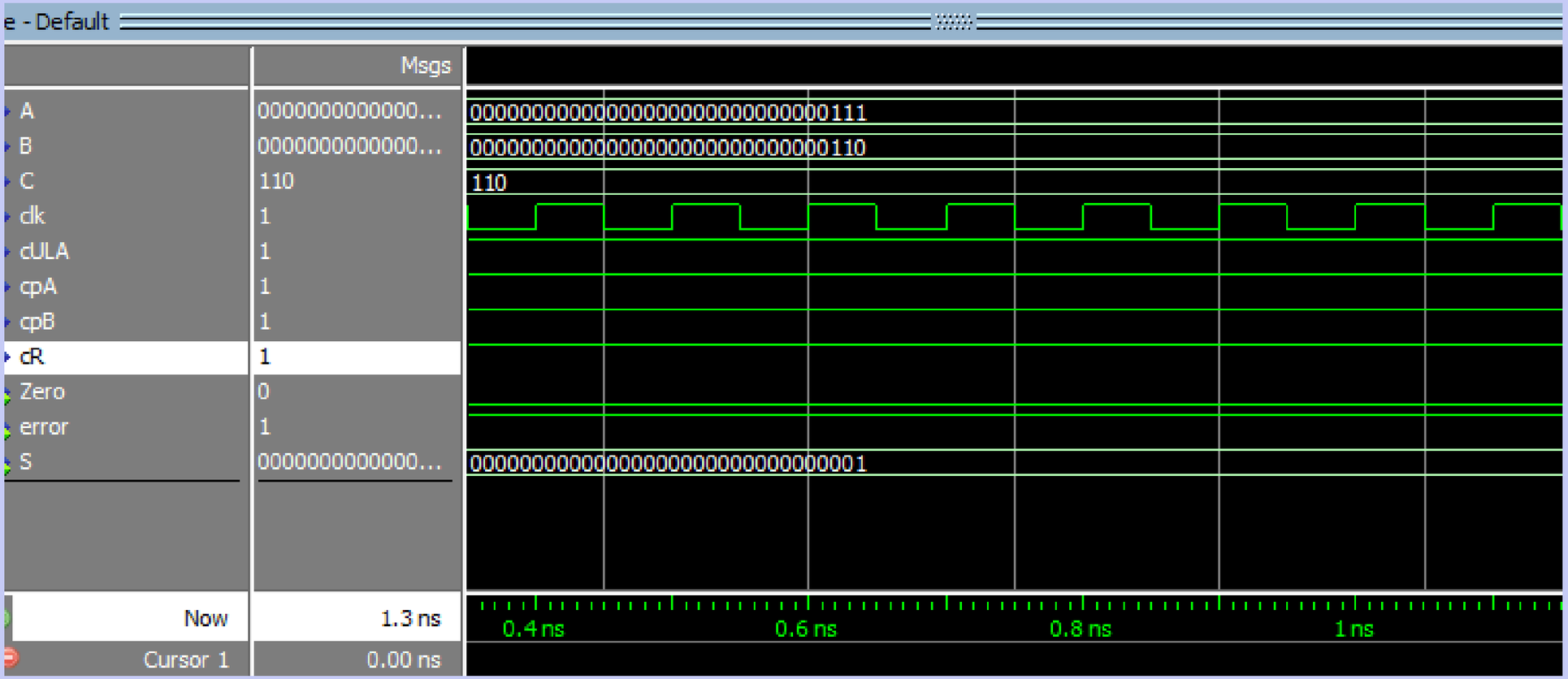
```
      -- lógica de redundancia do S
      if (resultado1 = resultado2) and (resultado3 = resultado1) and (c_red = '1') then
         S <= resultado1;
         error2 <= '0';
      elsif (resultado1 = resultado3) and (resultado3 /= resultado2) and (c_red='1') then
         S <= resultado1;
         error2 <= '1';
      elsif (resultado2 = resultado3) and (resultado2 /= resultado1) and (c_red='1') then
         S <= resultado2;
         error2 <= '1';
      else
         S <= resultado2;
         error2 <= '1';
      end if;
   end process;
```

4 Processo de redundância

```
signal tempor: std logic vector(N-1 downto 0) := "00100000000000000000000000000000";
```

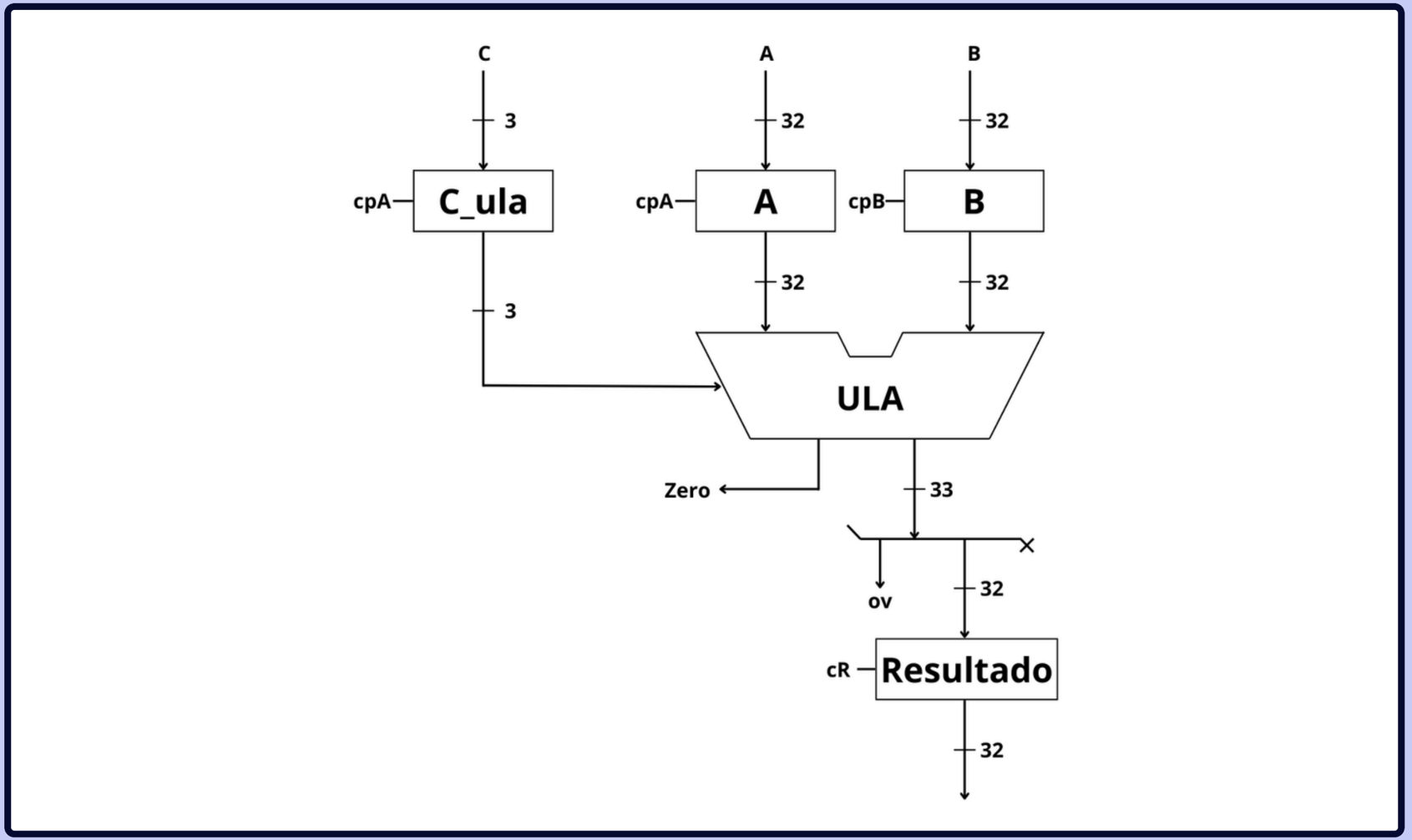
```
    -- lógica de redundancia do S

    if (resultado1 = tempor) and (resultado3 = resultado1) and (c_red = '1') then
        S <= resultado1;
        error2 <= '0';
    elsif (resultado1 = resultado3) and (resultado3 /= tempor) and (c_red='1') then
        S <= resultado1;
        error2 <= '1';
    elsif (tempor = resultado3) and (tempor /= resultado1) and (c_red='1') then
        S <= tempor;
        error2 <= '1';
    else
        S <= tempor;
        error2 <= '1';
    end if;
end process;
```

4 Processo de redundância



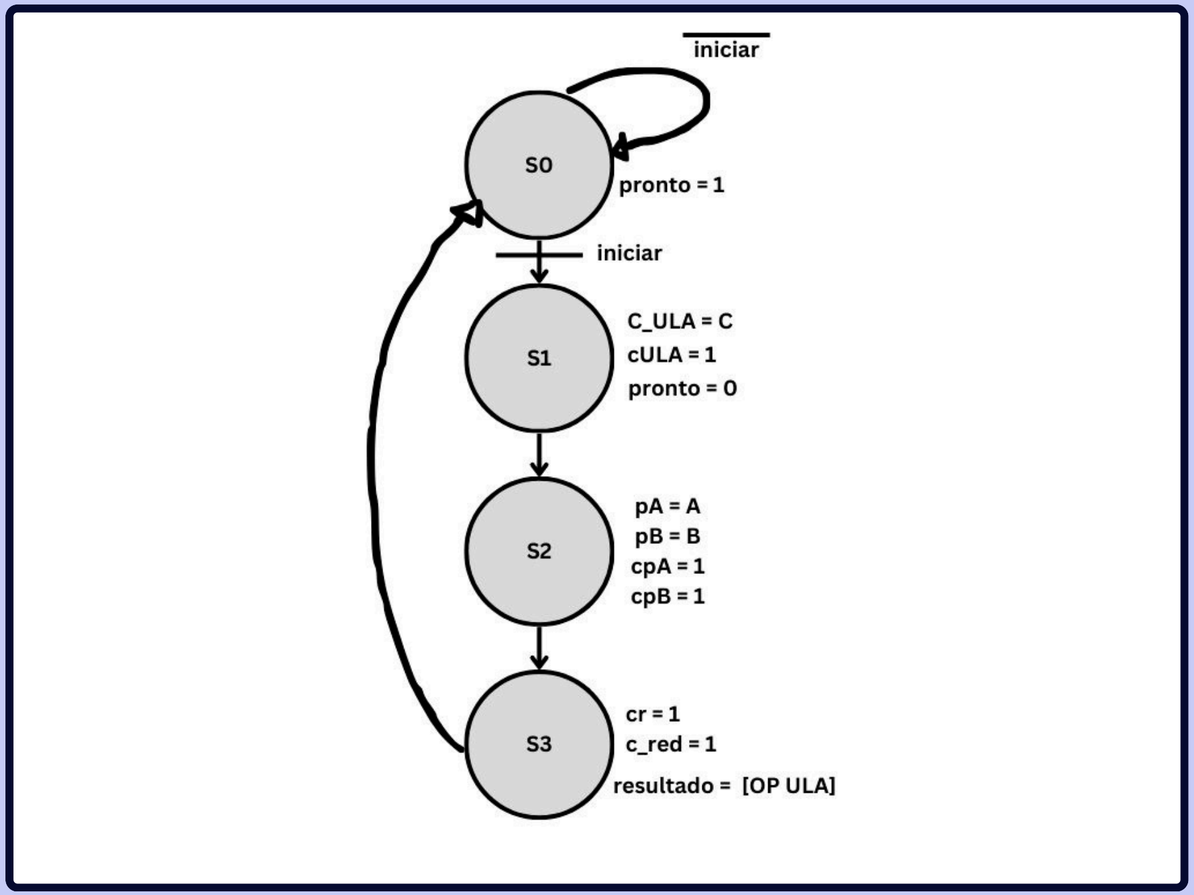
5 Bloco Operativo

- Esquemático nível RT do Bloco Operativo:



6 Bloco de Controle

- Esquemático do Bloco de Controle:



7 Testbench

- **Golden Model:**

```python
import random


def calculo_ula(c_ula, operando_a, operando_b, bit_width):
    max_val = 2 ** bit_width
    mask = max_val - 1
    zero = 0

    if c_ula == '0':  # AND
        ula_value = operando_a & operando_b
        print(operando_a)
        print(operando_b)
        print(ula_value)

    elif c_ula == '1':  # OR
        ula_value = operando_a | operando_b

    elif c_ula == '2':  # SOMA
        ula_value = (operando_a + operando_b) & mask
        if ula_value == 0:
            zero = 1

    elif c_ula == '6':  # SUBTRAÇÃO
        ula_value = (operando_a - operando_b) & mask
        if ula_value == 0:
            zero = 1

    elif c_ula == '7':  # SET ON LESS THAN
        ula_value = 1 if operando_a < operando_b else 0
    else:
        ula_value = 0

    return {'resultado': ula_value, 'sinal_zero': zero}

```

```python
def gerar_estimulos(filename, n_valores=50, c_ula_width=3, bit_width=32, result_width=32):
    with open(filename, 'w') as f:
        for n in range(n_valores):
            valores = ['0', '1', '2', '6', '7']
            c_ula = random.choice(valores)
            operando_a = random.randint(0, 2 ** bit_width - 1)
            operando_b = random.randint(0, 2 ** bit_width - 1)

            ula_value_zero = calculo_ula(c_ula, operando_a, operando_b, bit_width)
            resultado = ula_value_zero['resultado']
            zero = ula_value_zero['sinal_zero']
            erro = 0

            c_ula_bin = f'{int(c_ula):0{c_ula_width}b}"
            operando_a_bin = f'{operando_a:0{bit_width}b}'
            operando_b_bin = f'{operando_b:0{bit_width}b}'
            resultado_bin = f'{resultado:0{result_width}b}'
            print(operando_a_bin)
            print(operando_b_bin)
            print(resultado_bin)

            f.write(f'{operando_a_bin} {operando_b_bin} {c_ula_bin} {resultado_bin} {zero} {erro}\n')


# Gerar o arquivo de estímulos
gerar_estimulos('estimulos.dat', n_valores=50, c_ula_width=3, bit_width=32, result_width=32)

```

7 Testbench

- **Arquitetura:**

```
architecture tb of testbench is

    constant N: integer := 32;

    signal clk: std_logic := '0';
    signal iniciar: std_logic := '0';
    signal reset: std_logic := '0';
    signal A: std_logic_vector(N - 1 downto 0):= (others => '0');
    signal B: std_logic_vector(N - 1 downto 0):= (others => '0');
    signal C: std_logic_vector(2 downto 0):= (others => '0');
    signal Zero: std_logic := '0';
    signal erro: std_logic := '0';
    signal S: std_logic_vector(N-1 downto 0):= (others => '0');
    signal pronto: std_logic := '0';
    signal finished: std_logic := '0';

    constant passo : TIME := 20 ns;
```

```
stim: process
 file arquivo_de_estimulos : text open read_mode is "../../estimulos.da
 variable linha_de_estimulos: line;
 variable espaco: character;
 variable valor_de_entrada_1: bit_vector(N-1 downto 0);
 variable valor_de_entrada_2: bit_vector(N-1 downto 0);
 variable valor_de_entrada_3: bit_vector(2 downto 0);
 variable valor_de_saida_1: bit_vector(N-1 downto 0);
 variable valor_de_saida_2: bit;
 variable valor_de_saida_3: bit;
```

- **Geração do clock:**

```
begin

    clk_process : process
    begin
        while finished /= '1' loop
            clk <= '0';
            wait for passo / 2;
            clk <= '1';
            wait for passo / 2;
        end loop;
        clk <= '0';
    end process;
```

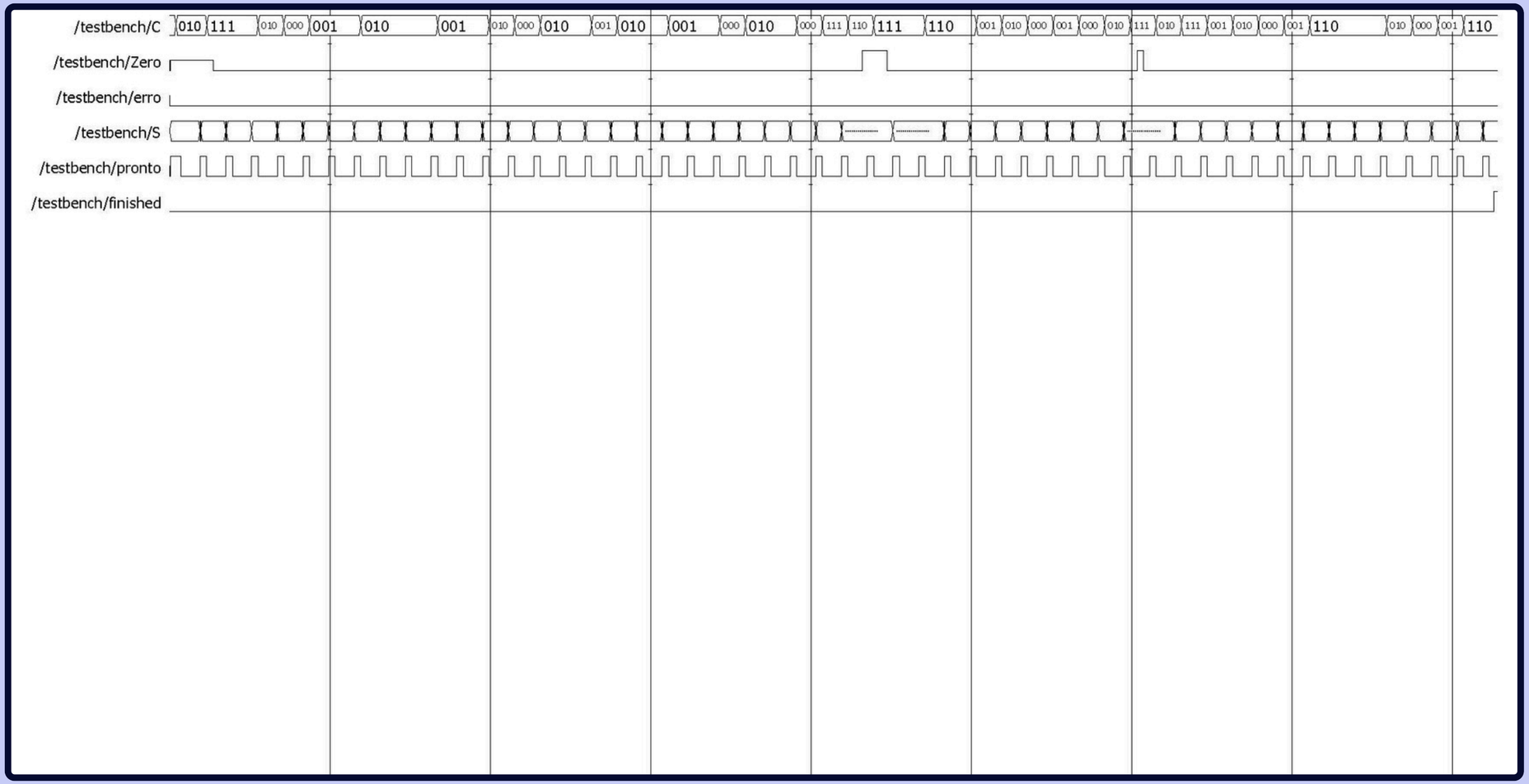
- **Instanciação DUV:**

```
DUV: entity work.ula_mips
     port map(
          clk => clk,
          iniciar => iniciar,
          reset => reset,
          A => A,
          B => B,
          C => C,
          S => S,
          Zero => Zero,
          erro => erro,
          pronto => pronto
     );
```

```
begin

    reset <= '1';
    wait for passo;
    reset <= '0';
    iniciar <= '1';

    while not endfile(arquivo_de_estimulos) loop

            readline(arquivo_de_estimulos, linha_de_estimulos);
            read(linha_de_estimulos, valor_de_entrada_1);
            A <= to_stdlogicvector(valor_de_entrada_1);
            read(linha_de_estimulos, espaco);
            read(linha_de_estimulos, valor_de_entrada_2);
            B <= to_stdlogicvector(valor_de_entrada_2);
            read(linha_de_estimulos, espaco);
            read(linha_de_estimulos, valor_de_entrada_3);
            C <= to_stdlogicvector(valor_de_entrada_3);
            read(linha_de_estimulos, espaco);
            read(linha_de_estimulos, valor_de_saida_1);
            read(linha_de_estimulos, espaco);
            read(linha_de_estimulos, valor_de_saida_2);
            read(linha_de_estimulos, espaco);
            read(linha_de_estimulos, valor_de_saida_3);

            wait until rising_edge(clk);
            wait until pronto = '1';
            wait for passo;

            assert (S = to_stdlogicvector(valor_de_saida_1))
                report "Falha na simulação: S errado."
                severity error;

            assert (Zero = to_stdulogic(valor_de_saida_2))
                report "Falha na simulação: Zero errado."
                severity error;

            if erro = '1' then
                report "Falha na simulação: 'Erro' sinalizado."
                severity error;
            end if;
    end loop;
```

7 Testbench

- **Resultados:**



8 Conclusão

- **Resultados finais e considerações**
-