RELATÓRIO

TRABALHO 01

Nome: Igor Fiedler de Bastos e Vinicius Domingos Gabriel

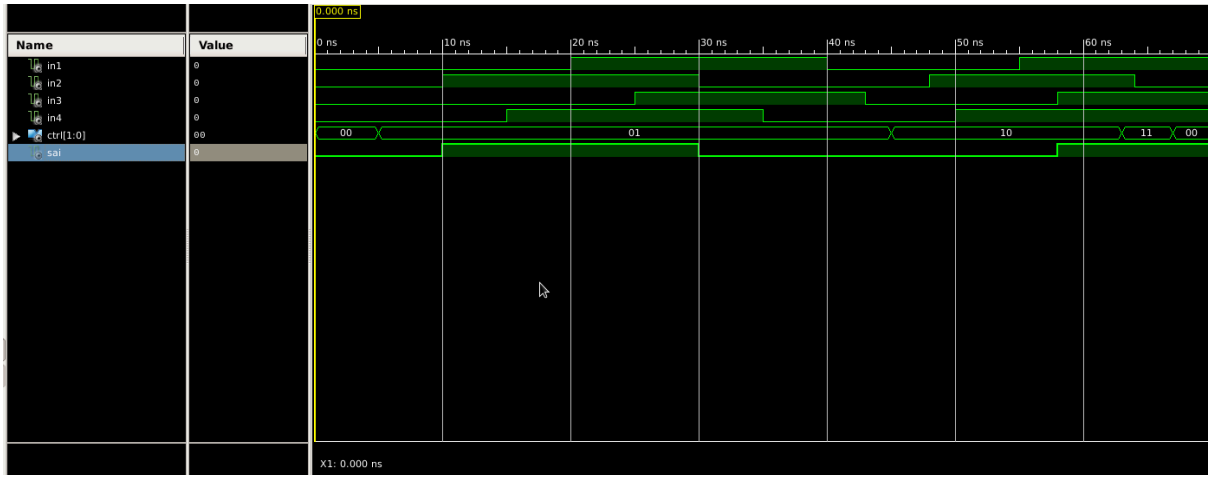
Araranguá, maio de 2025

Projeto 01

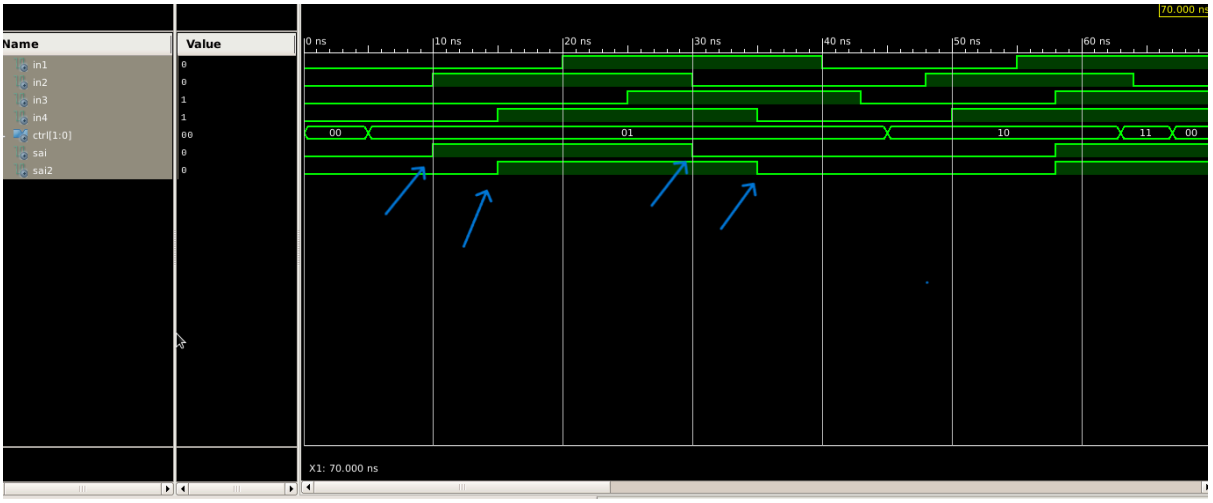
1. O código apresentado descreve um multiplexador 4:1, ou seja, um circuito com quatro entradas, uma saída e um sinal de controle de 2 bits. Esse sinal de controle determina qual das entradas será direcionada para a saída.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity prim_proc is
  port( in1, in2, in3, in4 : in std_logic;
        ctrl : in std_logic_vector (1 downto 0);
        sai : out std_logic);
end prim_proc;
architecture prim_proc of prim_proc is
begin
  process (in1, in2, in3, in4, ctrl)
  begin
    case ctrl is
      when "00"   => sai <= in1;
      when "01"   => sai <= in2;
      when "10"   => sai <= in3;
      when "11"   => sai <= in4;
      when others => null;
    end case;
   end process;
end prim_proc;
```

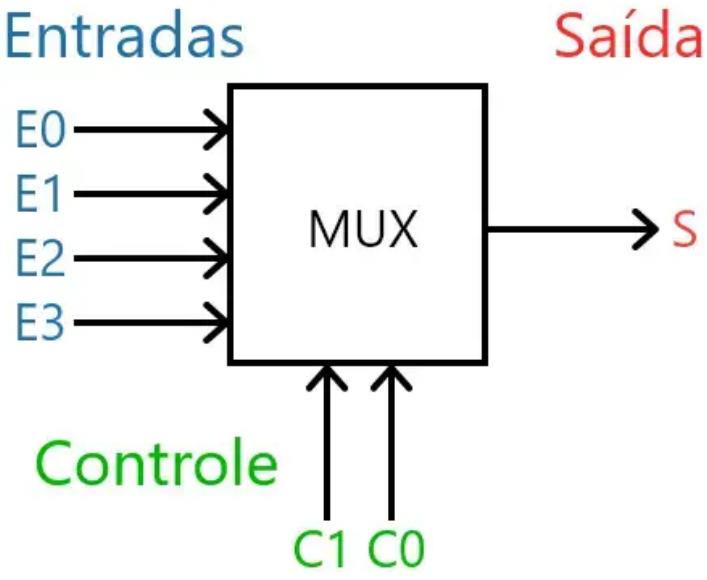
2. A partir da geração e simulação do testbench, foi possível verificar o funcionamento correto do multiplexador 4:1. A simulação demonstrou que, de acordo com o valor do sinal de controle `CTRL`, a saída `sai` assume o valor da entrada correspondente: `in1` quando `CTRL = "00"`, `in2` para `CTRL = "01"`, `in3` quando `CTRL = "10"` e `in4` para `CTRL = "11"`, conforme ilustrado na imagem abaixo. A simulação foi executada ao longo de 70 ns, com variações nos sinais de entrada e no controle, permitindo a observação do comportamento esperado da saída.



3. Foi criado um novo processo com uma segunda saída, denominada `sai2`, no qual o sinal `in2` foi intencionalmente removido da lista de sensibilidade. Ao comparar essa saída com a do processo original (com todos os sinais na lista), observamos que o processo modificado **não é executado quando apenas o `in2` muda de valor**. Isso compromete o funcionamento correto do multiplexador 4:1, pois a saída `sai2` permanece desatualizada até que outro sinal da lista de sensibilidade mude. Como consequência, ocorrem erros como atrasos ou valores incorretos na saída, evidenciando a importância de incluir todos os sinais relevantes na lista de sensibilidade para garantir o comportamento esperado do circuito. No nosso exemplo, da imagem abaixo, conseguimos verificar que a `sai2` deveria atualizar o valor de **`in2`** em 10ns porém a mudança ocorre apenas em 15ns quando se tem a atualização de **`in4`**.



4. Abaixo temos o esquemático do nosso circuito.



Projeto 02

Para atender ao objetivo do Projeto 2, foi desenvolvido um hardware responsável por realizar a transmissão serial de uma palavra de 8 bits. O circuito é estruturado em torno de uma máquina de estados finitos com os seguintes estados:

1. **REPOUSO**: estado inicial e de inatividade. A linha serial permanece em nível lógico alto ('1').

2. **START**: envia o bit de início da transmissão ('0').

3. **TRANSMITE**: transmite sequencialmente os 8 bits da palavra, do bit mais significativo (MSB) ao menos significativo (LSB).

4. **STOP**: envia o bit de parada ('0') e retorna ao estado de repouso.

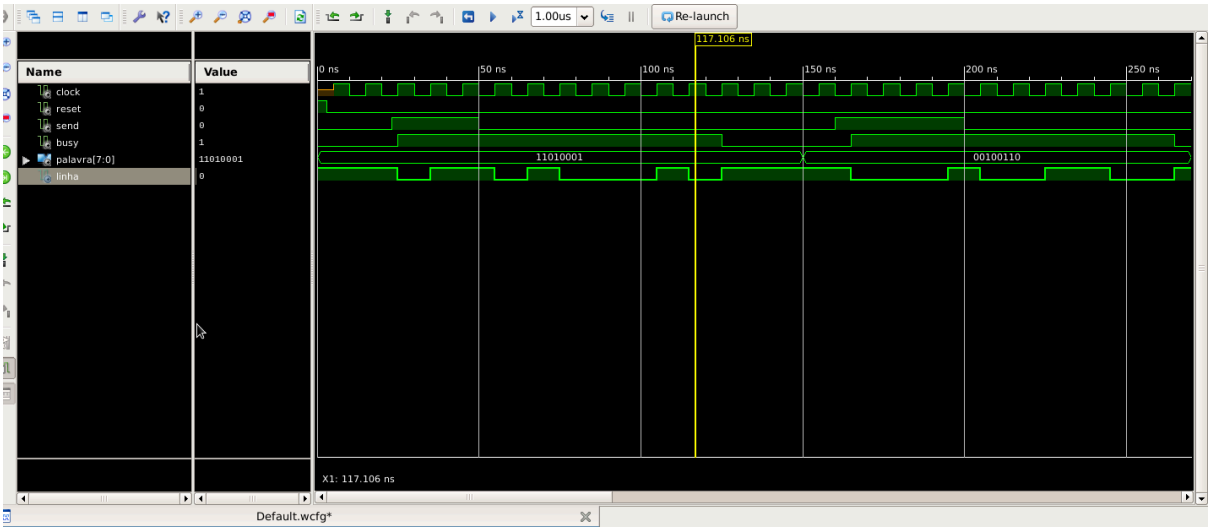
Foi implementado dois processos:

- Processo de controle sequencial (sensível a **clock** e **reset**):
  - Inicializa o sistema em **REPOUSO** após reset.
  - No estado **START**, carrega a palavra de 8 bits no registrador dado.
  - No estado **TRANSMITE**, incrementa o contador para percorrer os bits da palavra.
  - Zera o contador nos demais estados.

- Processo de controle combinacional (sensível a **estado_atual**, **send**, **contador**, dado):
  - Define a saída **linha_reg** com base no estado atual.
  - Controla a transição entre os estados de acordo com o sinal de envio (**send**) e o valor do contador.
  - Transmite os bits serialmente usando **linha_reg** <= **dado** (**7** - **contador**).

As saídas do são:

- **linha**: linha serial de saída, que reflete **linha_reg** e transmite os bits serialmente.
- **busy**: indica se o transmissor está ativo ('1' em qualquer estado exceto **REPOUSO**).


1.00us
Re-launch
117.106 ns
Name
Value
clock
1
reset
0
send
0
busy
1
palavra[7:0]
11010001
linha
0
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
11010001
00100110
X1: 117.106 ns
Default.wcfg*