
**Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Departamento de Engenharia de Computação**

Relatório: Trabalho Prático 1

Multiplexador de Endereçamento e ULA

Professores: Antônio Hamilton Magalhães

Bruno Luiz Dias Alves de Castro
Rafael Ramos de Andrade

Belo Horizonte
Campus Coração Eucarístico

9 de novembro de 2024

Conteúdo

1	Introdução	3
1.1	Objetivos	3
1.2	Simulação via Quartus II	3
1.2.1	Bloco w_reg	3
2	fsr_reg	3
2.1	Implementação	4
2.2	Simulação	4
3	Conclusão	5

1 Introdução

Durante as aulas da disciplina de Sistemas Reconfiguráveis, fomos introduzidos à linguagem VHDL. VHDL (**V**HSIC **H**ardware **D**escription **L**anguage) é uma linguagem de descrição de hardware. Com ela, podemos montar circuitos lógicos de maneira totalmente textual, o que garante à linguagem uma grande vantagem ante às soluções visuais.

1.1 Objetivos

1.2 Simulação via Quartus II

Nessa etapa realizamos testes no software Quartus II da altera.

1.2.1 Bloco w_reg

Nesta imagem é realizado 3 testes para verificar a funcionalidade do registrador, nos primeiros 60ns é alterado os bits da entrada de dados (d_in) para nível lógico alto, o bit de reset (nrst) que é ativo em baixa, é desativado, ou seja, nível lógico alto e o bit de ativação (wr_en) é colocado em nível lógico alto após 10ns. Assim é possível verificar a mudança na saída (w_out) com um tempo de delay de 6ns. No segundo teste a partir de 60ns até 140ns é resetado os bits da memória do registrador colocando reset em nível lógico zero, o resultado é propagada para a saída após o tempo de delay de aproximadamente 6ns. No terceiro teste foi verificados se o bit de ativação de escrita está funcionando corretamente, portanto com o bit 6 da saída em nível lógico alto esse valor será escrito apenas no tempo 160ns quando é colocado a porta de ativação do registrador em nível lógico alto e o registrador é escrito.

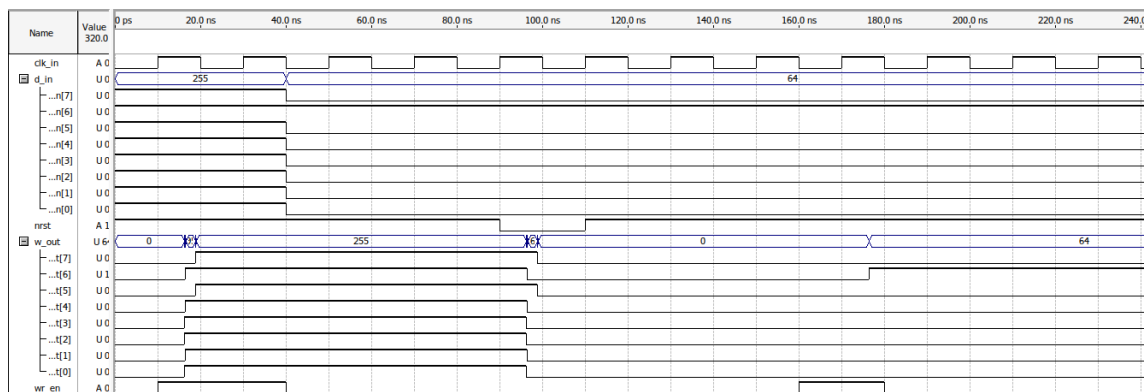


Figura 1: Simulação bloco w_reg

2 fsr_reg

O registrador FSR é um registrador semelhante ao implementado anteriormente. A principal diferença entre os dois está na presença de um sistema de endereçamento, e de duas entradas binárias independentes para habilitação da escrita e da leitura. Os requisitos são descritos na tabela abaixo.

Nome	Tamanho	Tipo	Descrição
nrst	1 bit	<i>Input</i>	Entrada de <i>reset</i> assíncrono.
clk_in	1 bit	<i>Input</i>	Entrada de <i>clock</i> .
abus_in	9 bit	<i>Input</i>	Entrada de endereçamento
dbus_in	8 bits	<i>Input</i>	Entrada de dados para escrita
wr_sel	1 bit	<i>Input</i>	Entrada de habilitação de escrita
rd_sel	1 bit	<i>Input</i>	Entrada de habilitação de leitura
dbus_out	8 bits	<i>Output</i>	Saída de dados habilitada por rd_en
fsr_out	8 bits	<i>Output</i>	Saída de dados sempre ativa

2.1 Implementação

O registrador `fsr_reg` foi implementado utilizando a linguagem VHDL.

O código na íntegra está abaixo:

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_unsigned.all;
4  USE ieee.numeric_std.all;
5
6  ENTITY fsr_reg IS
7      PORT (
8          -- Inputs
9          nrst : IN STD_LOGIC;                -- Reset
10         clk_in: IN STD_LOGIC;                -- Clock
11         abus_in: IN STD_LOGIC_VECTOR(8 DOWNTO 0); -- Endereçamento
12         dbus_in: IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- Dados
13         wr_en : IN STD_LOGIC;                -- Enable escrita
14         rd_en : IN STD_LOGIC;                -- Enable leitura
15
16         -- Outputs
17         dbus_out : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- Dados
18         fsr_out : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- Registrador
19     );
20 END ENTITY;
21
22 ARCHITECTURE fsr_reg OF fsr_reg IS
23     SIGNAL mem_reg: STD_LOGIC_VECTOR(7 DOWNTO 0);
24 BEGIN
25     PROCESS (nrst, clk_in, mem_reg, abus_in, dbus_in)
26     BEGIN
27         IF nrst = '0' THEN
28             mem_reg <= "00000000";
29         ELSIF abus_in(6 DOWNTO 0) = "0000100" THEN
30             IF RISING_EDGE(clk_in) THEN
31                 IF wr_en = '1' THEN
32                     mem_reg <= dbus_in;
33                 END IF;
34             END IF;
35         END IF;
36     END PROCESS;
37
38     dbus_out <= mem_reg WHEN rd_en = '1' ELSE "ZZZZZZZZ";
39     fsr_out <= mem_reg;
40 END fsr_reg;

```

Listing 1: Código VHDL `fsr_reg`

2.2 Simulação

Para testar nosso código VHDL e certificar-nos de que nosso circuito funciona de maneira esperada, simulamos alguns casos de testes utilizando o software Quatus II.

Os testes realizados foram os seguintes:

1. Escrita com endereçamento incorreto (diferente de XX0000100).

- **Comportamento esperado:**

- dbus_out em alta impedância;
- fsr_out sem alteração;

2. Leitura habilitada e escrita desabilitada.

- **Comportamento esperado:**

- dbus_out = frs_out = último valor escrito;

3. Leitura desabilitada e escrita habilitada.

- **Comportamento esperado:**

- dbus_out em alta impedância;
- frs_out = dbus_in;

4. *Reset* com leitura habilitada.

- **Comportamento esperado:**

- dbus_out = frs_out = “0b00000000”;

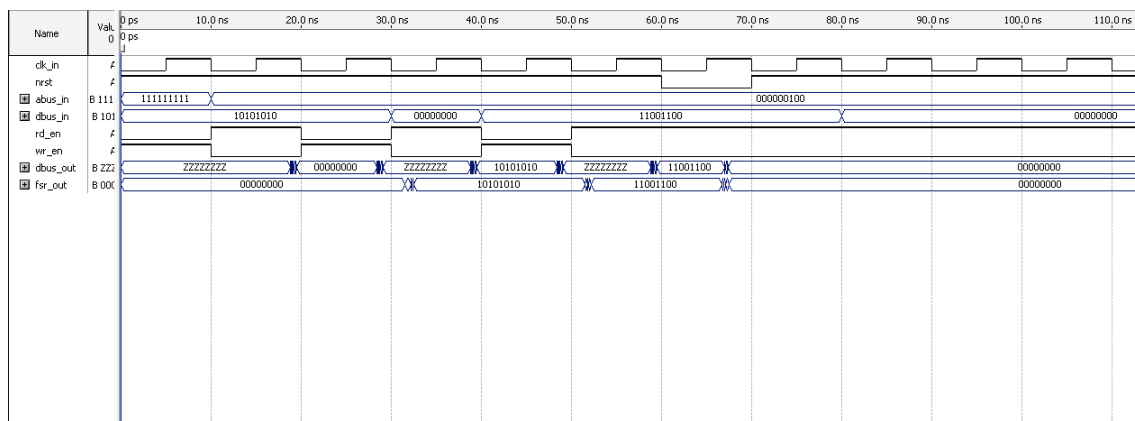


Figura 2: Simulação fsr_reg

3 Conclusão

Com estes dois projetos simples, tivemos um excelente primeiro contado com a linguagem VHDL, bem como à programação concorrente e desenvolvimento de circuitos FPGA. Os dois circuitos implementados (Multiplexador de Endereçamento e Unidade Lógica Aritimética) são blocos de construção chave para a maior parte dos circuitos complexos, e serão de suma importância não só para os demais trabalhos práticos que realizaremos ao longo do semestre, mas para nosso desenvolvimento acadêmico e profissional.