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

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 à 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 nivel 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 colocoado 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.

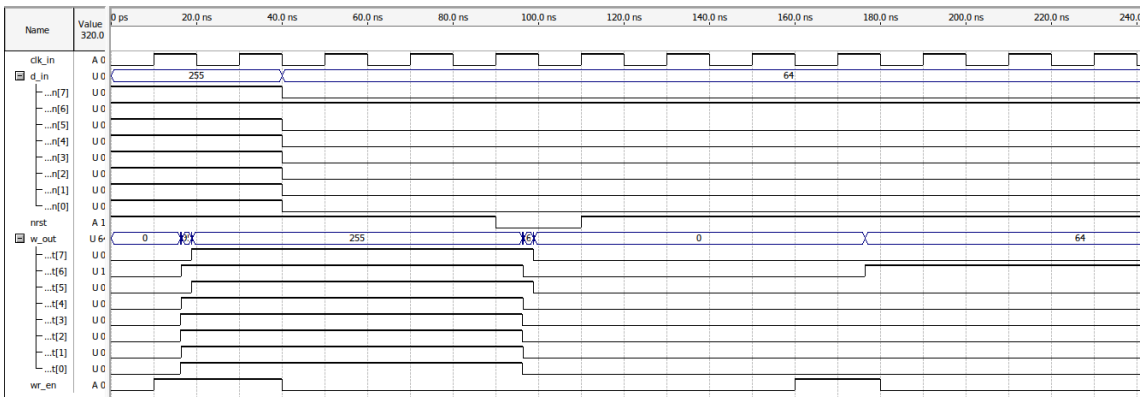

Figura 1: Simulação bloco w_reg

2 fsr_reg

O registrador FSR é um registrador semelhante ao implemntado 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	*Input*	Entrada de *reset* assíncrono.
clk_in	1 bit	*Input*	Entrada de *clock*.
abus_in	9 bit	*Input*	Entrada de enderençamento
dbus_in	8 bits	*Input*	Entrada de dados para escrita
wr_sel	1 bit	*Input*	Entrada de habilitação de escrita
rd_sel	1 bit	*Input*	Entrada de habilitação de leitura
dbus_out	8 bits	*Output*	Saída de dadas hailitada por rd_en
fsr_out	8 bits	*Output*	Saída de dadas dados sempre ativa

2.1 Implementação

O registrador fsr_reg foi implementado utilizando a linguagem VHDL.

O código na íntegra está abaixo:

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

ENTITY fsr_reg IS
    PORT (
        -- Inputs
        nrst : IN STD_LOGIC;                            -- Reset
        clk_in: IN STD_LOGIC;                           -- Clock
        abus_in: IN STD_LOGIC_VECTOR(8 DOWNTO 0);       -- Enderecamento
        dbus_in: IN STD_LOGIC_VECTOR(7 DOWNTO 0);       -- Dados
        wr_en : IN STD_LOGIC;                           -- Enable escrita
        rd_en : IN STD_LOGIC;                           -- Enable leitura

        -- Outputs
        dbus_out : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);    -- Dados
        fsr_out : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)      -- Registrador
    );
END ENTITY;

ARCHITECTURE fsr_reg OF fsr_reg IS
    SIGNAL mem_reg: STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
    PROCESS (nrst, clk_in, mem_reg, abus_in, dbus_in)
    BEGIN
        IF nrst = '0' THEN
            mem_reg <= "00000000";
        ELSIF abus_in(6 DOWNTO 0) = "0000100" THEN
            IF RISING_EDGE(clk_in) THEN
                IF wr_en = '1' THEN
                    mem_reg <= dbus_in;
                END IF;
            END IF;
        END IF;
    END PROCESS;

    dbus_out <= mem_reg WHEN rd_en = '1' ELSE "ZZZZZZZZ";
    fsr_out <= mem_reg;
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 seguites:

1. Escrita com enderaçamento 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 impendância;
     - frs_out = dbus_in;
4. *Reset* com leitura habilitada.
   - **Comportamento esperado:**
     - dbus_out = frs_out = “0b00000000”;

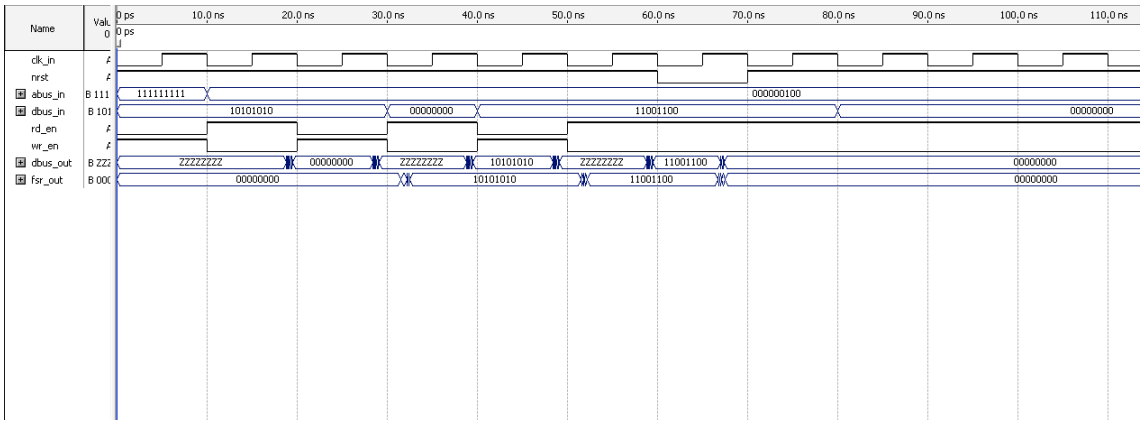

Figura 2: Simulação fsr_reg

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.