**UNIVERSIDADE DE SÃO PAULO**

ESCOLA DE ENGENHARIA DE SÃO CARLOS

**SEL0384 – Laboratório de Sistemas Digitais I**

Prof. Dr. Maximiliam Luppe

Bárbara Fernandes Madera - nº: 11915032

Johnny Caselato Guimarães - nº: 11915481

**PRÁTICA Nº10**

**Aprendizado baseado em problema (PBL)**

**PBL02 - Circuitos Sequenciais - Banco de Registradores**

**SÃO CARLOS**

**2023**

**1. Introdução**

O presente relatório documenta a implementação de um Banco de Registradores, utilizando a linguagem de descrição de hardware VHDL, com número de registradores e tamanho do barramento de dados parametrizáveis. Este projeto faz parte da construção de um modelo simplificado de processador RISC-V.

**2. Equipamentos Utilizados**

* Kit Mercurio® IV

**3. Descrição do Problema**

A seguinte prática consiste na implementação do Contador de Programa, dos Registradores Auxiliares e do Banco de Registradores. Estes últimos são utilizados para armazenar dados temporários, sendo classificados como auxiliares (A, B, IR) e de dados (Banco de Registradores).

**4. Banco de Registradores: Definição e Características**

* Registradores Auxiliares: A, B, IR.
* Banco de Registradores: R0 a RM-1 (M = 2^SIZE).

**5. Desenvolvimento**

a. **Registrador de WIDTH bits:**

* Entradas: CLK e EN.
* Componente independente utilizado na organização da arquitetura.

b. **Banco de Registradores:**

* 1 Entrada de dados (WD3).
* 2 Saídas de dados (RD1, RD2).
* Seletor de registrador para escrita (A3).
* 2 Seletores de registrador para leitura (A1, A2).
* Clock (CLK).
* Habilitação de escrita (WE3).
* Utilização de Multiplexadores e decodificadores para seleção

**6. Conceitos de Banco de Registradores em Arquiteturas de 32 bits**

Em arquiteturas de 32 bits, os bancos de registradores desempenham um papel importante no funcionamento eficiente do processador. Neste sentido, um banco de registradores é um conjunto de registradores de dados que são utilizados para armazenar informações temporárias durante a execução de programas os quais permitem o acesso direto a dados fundamentais para otimizar o desempenho e eficiência operacional. Esses registros, os locais mais rápidos de armazenamento em computadores, possibilitam operações de leitura e escrita extremamente rápidas, essenciais para o funcionamento global do sistema.

Além disso, os bancos de registradores facilitam a execução eficiente de operações aritméticas e lógicas, promovendo um processamento mais eficaz de instruções. Eles desempenham um papel na manutenção do estado da CPU, incluindo contadores de programa, ponteiros de pilha e outros registros de controle.

Dessa forma, a capacidade de acesso simultâneo a diferentes conjuntos de dados possibilita a execução paralela de instruções, melhorando o desempenho em tarefas multithread. Os registradores, por sua vez , projetados para leitura e escrita rápidas, são essenciais para garantir um fluxo contínuo de informações durante a execução de programas.

Com isso, dado que as instruções de máquina frequentemente envolvem operações com registradores, o uso de bancos de registradores promove portabilidade e padrões o que maximiza a eficiência do processamento em arquiteturas de 32 bits facilitando, assim, a uniformidade entre diferentes processadores e sistemas.

Portanto, os bancos de registradores são elementos fundamentais que otimizam o desempenho global dos processadores em arquiteturas de 32 bits, garantindo a execução eficaz de programas em ambientes computacionais modernos.

**7. Procedimento Experimental**

Implementação do registrador genérico de WIDTH bits. b. Construção do Banco de Registradores baseado na arquitetura RISC-V. c. Estruturação de pastas e projeto no Quartus Lite. Incorporação do código do Banco de Registradores ao projeto principal. Configuração do projeto para o Kit Mercurio® IV (WIDTH = 4 e SIZE = 2). f. Conexão de entradas e saídas conforme especificado. g. Execução do projeto no Kit Mercurio® IV.

**8. implementação e Resultados**

1. Código da entidade *top level* “MercurioIV\_RegBank”:

**-- Projeto Banco de Registradores**

**-- Autores:**

**-- Bárbara Fernandes Madera - nº: 11915032**

**-- Johnny Caselato Guimarães - nº: 11915481**

**-- Professor: Maximiliam Luppe**

**-- Definição da entidade MercurioIV\_RegBank**

**entity MercurioIV\_RegBank is**

**port (**

**SW : in bit\_vector(3 downto 0); -- Entrada dos interruptores**

**SA : in bit\_vector(7 downto 0); -- Entrada A**

**SB : in bit\_vector(7 downto 0); -- Entrada B**

**KEY : in bit\_vector(11 downto 0); -- Entrada da chave**

**DISP0\_D : out bit\_vector(7 downto 0); -- Saída para o Display 0**

**DISP1\_D : out bit\_vector(7 downto 0) -- Saída para o Display 1**

**);**

**end MercurioIV\_RegBank;**

**-- Arquitetura para MercurioIV\_RegBank**

**architecture top of MercurioIV\_RegBank is**

**-- Sinais utilizados dentro da arquitetura**

**signal r1\_out : bit\_vector(3 downto 0);**

**signal r2\_out : bit\_vector(3 downto 0);**

**begin**

**-- Instância do módulo 'regbank' da biblioteca 'work'**

**regbank\_0: work.regbank**

**generic map (wd => 4, sz => 4) -- Parâmetros genéricos para o módulo regbank**

**port map (**

**A1 => SB(3 downto 0), A2 => SB(7 downto 4), A3 => SW(3 downto 0), -- Entradas**

**CLK => KEY(8), -- Entrada de clock**

**WE3 => KEY(5), -- Habilitar escrita para SA**

**WD3 => SA(3 downto 0), -- Dados de escrita para SA**

**RD1 => r1\_out, -- Dados lidos do regbank para o Display 0**

**RD2 => r2\_out -- Dados lidos do regbank para o Display 1**

**);**

**-- Instância do módulo 'MercurioIV\_decod' para o Display 0**

**disp\_0: work.MercurioIV\_decod**

**port map (hexa => r1\_out, segments => DISP0\_D(6 downto 0));**

**-- Instância do módulo 'MercurioIV\_decod' para o Display 1**

**disp\_1: work.MercurioIV\_decod**

**port map (hexa => r2\_out, segments => DISP1\_D(6 downto 0));**

**end top;**

1. Código da entidade “regbank”:

**-- Projeto Banco de Registradores**

**-- Autores:**

**-- Bárbara Fernandes Madera - nº: 11915032**

**-- Johnny Caselato Guimarães - nº: 11915481**

**-- Professor: Maximiliam Luppe**

**-- Definição da entidade 'regbank'**

**entity regbank is**

**generic (**

**wd : unsigned integer := 2; -- Tamanho dos registradores**

**sz : unsigned integer := 2; -- Número de registradores**

**);**

**port (**

**A1, A2, A3 : in bit\_vector(sz-1 downto 0); -- Entradas dos registradores**

**CLK, WE3 : in bit; -- Sinais de clock e write enable**

**WD3 : in bit\_vector(wd-1 downto 0); -- Dados de escrita para registrador 3**

**RD1, RD2 : out bit\_vector(wd-1 downto 0) -- Dados de leitura dos registradores 1 e 2**

**);**

**end regbank;**

**-- Arquitetura RTL para 'regbank'**

**architecture rtl of regbank is**

**begin**

**-- Geração dos registradores**

**gera\_regs : for r in 0 to sz-1 generate**

**-- Instância do módulo 'reg' da biblioteca 'work' para cada registrador**

**reg\_n : work.reg**

**generic map(n => wd) -- Configuração do tamanho do registrador**

**port map(**

**clk\_r => CLK, -- Sinal de clock para o registrador**

**en => WE3, -- Sinal de write enable para o registrador**

**d => WD3, -- Dados de entrada para o registrador**

**q => -- Dados de saída do registrador**

**);**

**end generate gera\_regs;**

**end rtl;**

1. Código da entidade “reg”:

**-- Projeto Banco de Registradores**

**-- Autores:**

**-- Bárbara Fernandes Madera - nº: 11915032**

**-- Johnny Caselato Guimarães - nº: 11915481**

**-- Professor: Maximiliam Luppe**

**-- Definição da entidade 'reg'**

**entity reg is**

**generic (**

**n : integer := 2 -- Parâmetro genérico para determinar o tamanho dos vetores**

**);**

**port (**

**clk\_r, en : in bit; -- Sinais de entrada para o clock e enable**

**d : in bit\_vector(n-1 downto 0); -- Vetor de dados de entrada**

**q : out bit\_vector(n-1 downto 0) -- Vetor de dados de saída**

**);**

**end reg;**

**-- Arquitetura RTL para 'reg'**

**architecture rtl of reg is**

**begin**

**-- Processo para verificar a borda de subida do clock e a condição de enable**

**check\_clock : process (clk\_r)**

**begin**

**if (clk\_r'event and clk\_r = '1' and en = '1') then**

**q <= d; -- Quando há borda de subida do clock e enable é ativo, atualiza o vetor de saída (q) com os dados de entrada (d)**

**end if;**

**end process;**

**end rtl;**

**9. Conclusão**

Apesar da implementação não ter sido completa, a lógica construída se mostrava promissora, e, com o suporte do professor em sala, foi possível compreender as limitações que a plataforma e o VHDL impõem ao projeto.

O banco de registradores, para este caso, necessitaria da implementação de um componente de conversão binário-decimal e um decodificador, não para a sua saída (como já foi feito a fim de demonstrar os resultados nos displays de 7 segmentos), mas para a lógica de seleção de escrita e leitura dos dados.