PCS3225 - Sistemas Digitais II
Atividade Formativa 11 - Projeto do Processador PoliLEGv8 em VHDL
Parte 2 - Program Counter (PC) e Banco de Registradores do PoliLEGv8

Glauber de Bona, revisado por Edson Toshimi Midorikawa (2023) e Antonio Vieira da Silva Neto (2024)

Data do Arquivo: 08/11/2024; Prazo da AF11: 23/11/2024

O objetivo desta parte da Atividade Formativa 11 é exercitar a descrição de registradores em VHDL, que serão usados como componentes internos do PoliLEGv8.

O registrador PC (*Program Counter* e o banco de registradores.

Introdução

O registrador é um dos componentes básicos do processador. Ele é o elemento de memória mais próximo das unidades funcionais e também o mais rápido, pois está diretamente ligado a elas dentro do fluxo de dados. Todas as operações que ocorrem no processador PoliLEGv8, com exceção dos desvios incondicionais, envolvem ao menos um registrador. Todos os registradores do PoliLEGv8 possuem 64 bits de largura. Ademais, com exceção do registrador PC (*Program Counter*), os 32 outros registradores do PoliLEGv8 estão localizados em um **banco de registradores**.

Em sistemas digitais, um registrador nada mais é que um conjunto de *flip-flops*. Em VHDL, um *flip-flop* é descrito usando-se uma atribuição incompleta dentro de um bloco sequencial. Considere o exemplo de bloco sequencial a seguir:

```
ffdr: process(clock, reset)
begin
    if reset='1' then
        q <= '0';
        q_n <= '1';
    elsif clock='1' and clock'event then
        if en='1' then
            q <= d;
            q_n <= not d;
        end if;
    end if;
end process;
```

Nesse bloco, foi descrito um *flip-flop* tipo D com *reset* assíncrono. Note que a lista de sensibilidade do processo tem os sinais `clock` e `reset`, mas o `if` divide as ações em assíncronas (se houver *reset*) ou síncronas (sensível à borda de subida do `clock`). Porém, se não houver um *reset* nem uma borda de subida do *clock*, o comportamento não é especificado. Chamamos isso de **atribuição incompleta**, e a ação padrão do sintetizador será a de manter os sinais inalterados.

O *reset* é prioritário pois é avaliado antes.

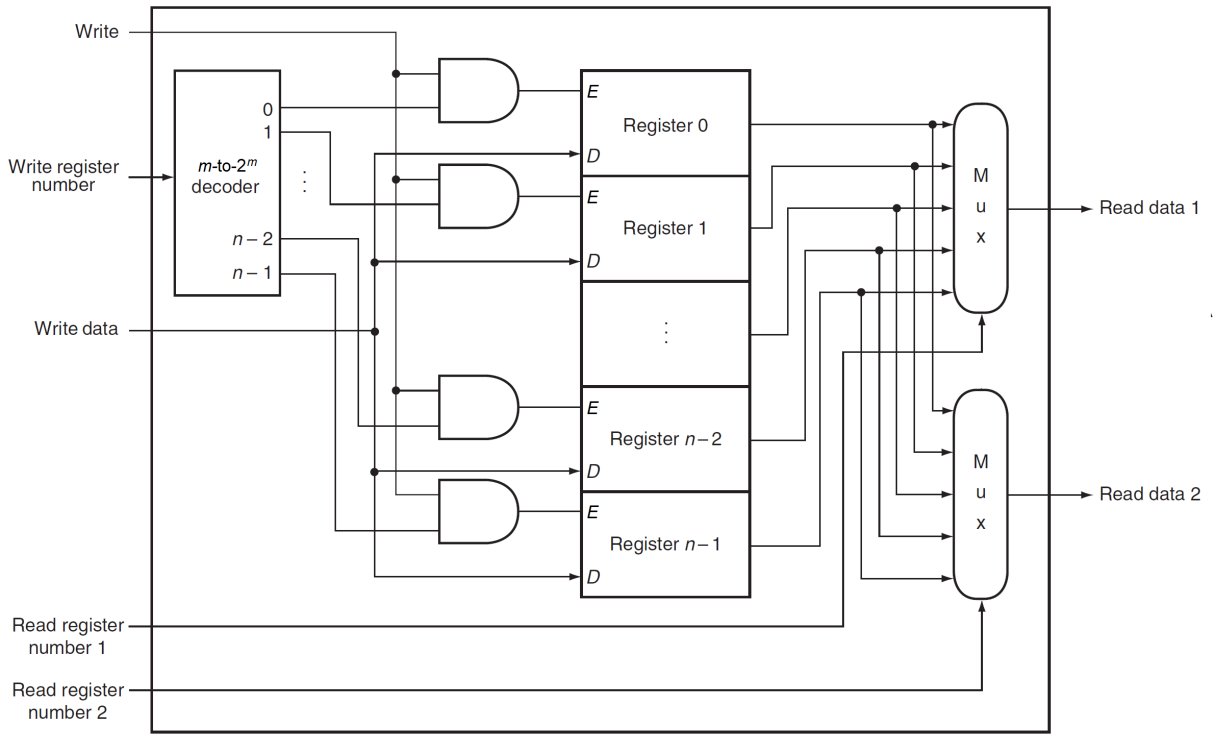
Para manter os sinais como estão, deve existir um elemento de me-

mória. Nesse caso, o elemento de memória que o sintetizador VHDL é forçado a usar para manter o valor do ciclo de *clock* anterior é exatamente um *flip-flop*. Caso fosse desejado, seria possível adicionar um caso complementar (`else`) contendo uma atribuição para o mesmo sinal, mas isso é desnecessário. Esse tipo de atribuição incompleta também funciona para outros sinais, e o sintetizador encarregar-se-á de colocar o número de *flip-flops* adequado para manter o valor de um vetor ou sinal inalterado.

```
q<=q;
```

Veja em `https://balbertini.github.io/vhdl_sequential-pt_BR.html` a postagem do Prof. Bruno Albertini (PCS/Poli-USP) sobre descrição de Circuitos Sequenciais em VHDL.

Um **banco de registradores** é composto por um conjunto de registradores que podem ser lidos e escritos. Para isso, é necessário fornecer um número de registrador para o acesso ser realizado. Em um banco de registradores com `n` registradores, os números dos registradores variam de `0` a `n-1`. A figura a seguir ilustra um diagrama lógico geral de um banco de registradores que permite acessar simultaneamente até três registradores, sendo dois deles para leitura e um para escrita.



A leitura de um registrador específico é feita de modo assíncrono e pode ser realizada somente com a definição do número de registrador. Por exemplo, no banco de registradores da figura anterior, fornece-se o código numérico de um registrador em qualquer das entradas `Read register number x`, com `x = 1, 2`, e obtém-se o dado

correspondente na respectiva saída de dados `Read data x`.

A escrita em um registrador do banco, por sua vez, requer três entradas: (i.) o código numérico do registrador, (ii.) o dado a ser escrito e (iii.) um sinal de controle para definir a operação de escrita. O código numérico do registrador é definido na entrada `Write register number`, ao passo que o dado a ser escrito deve ser fornecido na entrada `Write data`. Quando ocorrer uma borda de subida do sinal de `Clock` e o sinal de controle de escrita `Write` for igual a `1`, a escrita do dado `Write data` é efetuada no registrador `Write register number`. Por outro lado, se o sinal de controle `Write` for igual a `0`, nenhuma operação de escrita em registrador é efetuada (independentemente dos valores de `Write register number` e `Write datas`).

Atividades

AF11-P2E1 Implemente um componente em VHDL correspondente a um registrador parametrizável que respeite a seguinte entidade:

Atividade Formativa 11 Parte 2 - Exercício 1

```
entity reg is
    generic(wordSize: natural :=4);
    port(
        clock  : in  bit;                                  --! entrada de clock
        reset  : in  bit;                                  --! clear assincrono
        enable : in  bit;                                  --! write enable (carga paralela)
        d      : in  bit_vector(wordSize-1 downto 0);      --! entrada
        q      : out bit_vector(wordSize-1 downto 0)       --! saida
    );
end entity reg;
```

O registrador é parametrizável por meio do parâmetro `wordSize`, que determina sua largura em bits. Ele recebe um sinal de *clock* periódico e é sensível a borda de subida desse sinal. O sinal `reset` é assíncrono e força todos os bits do registrador para zero. Quando o sinal `load` é alto, a escrita é considerada habilitada e o valor na entrada `d` é gravada nos *flip-flops* quando ocorrer uma borda de subida do *clock*, refletindo-se, então, na saída `q`. Contrariamente, quando o sinal `load` é baixo, nada acontece e a saída permanece inalterada. A saída `q` é assíncrona e mostra o conteúdo atual do registrador.

Tanto a entrada `d` quando a saída `q` são paralelas e contém `wordSize` bits.

Dica: Este registrador serve como referência para a implementação do *Program Counter* (PC) e para os elementos que integram o banco de registradores do processador PoliLEGv8. É altamente recomendável utilizar a descrição criada no exercício **AF11-P2E1** para essas finalidades.

AF11-P2E1 Implemente um componente em VHDL correspondente ao banco de registradores do PoliLEGv8, que possui 32 registradores com 64 bits de largura cada. O banco de registradores deve respeitar a seguinte entidade:

Atividade Formativa 11 Parte 2 - Exercício 2

```
entity regfile is
    port(
        clock    : in  bit;                    --! entrada de clock
```

```
        reset    : in  bit;                      --! entrada de reset
        regWrite : in  bit;                      --! entrada de carga do registrador wr
        rr1      : in  bit_vector(4 downto 0);   --! entrada define registrador 1
        rr2      : in  bit_vector(4 downto 0);   --! entrada define registrador 2
        wr       : in  bit_vector(4 downto 0);   --! entrada define registrador de escrita
        d        : in  bit_vector(63 downto 0);  --! entrada de dado para carga paralela
        q1       : out bit_vector(63 downto 0);  --! saida do registrador rr1
        q2       : out bit_vector(63 downto 0)   --! saida do registrador rr2
    );
end entity regfile;
```

Por padrão, os registradores são numerados de `0` até `31`. As entradas `rr1`, `rr2`, `wr` são de 5 bits e definem os registradores `1` e `2` a serem lidos e o registrador a ser escrito, respectivamente. Todos os registradores podem ser usados para leitura e escrita de dados. O último registrador (`X31`) não deve aceitar escritas (não há efeito algum em escrever nele) e sempre devolve zero quando lido.

No PoliLEGv8, o registrador `X31` é o `XZR`.

É opcional, mas fortemente aconselhável, usar o registrador da atividade anterior como componente. O restante das caraterísticas do registrador permanecem as mesmas: todos os registradores do banco são sensíveis à borda de subida do *clock* e o *reset* é assíncrono.

O *reset* vale para todos os registradores do banco.

A entrada de dados `d` possui largura de 64 bits e, se o sinal de controle `regWrite` for alto, na borda de subida do *clock* somente o registrador apontado pela entrada `wr` será escrito. Exemplo: se `wr=01010` o registrador `X10` (ou o décimo primeiro registrador) amostrará a entrada `d` para seus *flip-flops* internos. Caso o `regWrite` seja baixo na borda de subida do *clock*, a escrita não acontece.

Escrita no banco

A leitura é assíncrona, e o banco possui duas saídas de leitura. O registrador apontado pela entrada `rr1` coloca o seu valor na saída `q1` e o registrador apontado pela entrada `rr2` coloca seu valor na saída `q2`. Por exemplo: se `rr1=00011` e `rr2=10001`, o registrador `X3` (ou o quarto registrador) coloca o seu valor na saída `q1`. ao passo que o registrador `X17` (ou o 18º registrador) coloca o seu valor na saída `q2`.

Leituras no banco

É **obrigatório** o uso da forma de **descrição estrutural** no projeto do Banco de Registradores, com a composição dos elementos internos (registradores, multiplexadores, decodificador e lógica de habilitação).

Dica: Consulte a postagem do Prof. Bruno Albertini em `https://balbertini.github.io/vhdl_generate-pt_BR.html` sobre descrição de circuitos em VHDL com a composição repetitiva de diversas instâncias do mesmo componente (usando o comando `for-generate`).

Instruções para os Grupos

Como boa prática de projeto de Engenharia, façam seus *testbenches* e utilize o GHDL/GtkWave ou *EDA Playground* (selecionando o GHDL como simulador) para validar suas soluções.

Use as técnicas apresentadas no módulo de **Verificação de Projetos** para acrescentar formas de verificação de funcionamento para

validar o projetos das diversas atividades do projeto.

Atenção: Para as atividades do projeto, está proibido o uso da biblioteca `ieee.std_logic_1164`.