Arquitetura e Organização de Computadores
Turma C - 2021/01

Projeto da Memória de Dados e Instruções

Objetivo: projetar, simular e sintetizar a memória de dados e instruções do RISC-V.

Resumo

O RISC-V multiciclo utiliza uma memória interna para armazenar programa e dados. Neste trabalho deverá ser desenvolvido um modelo de memória em VHDL para simulação. O tamanho da memória deverá ser compatível com a configuração do RARS: 4096 palavras de 32 bits.

Descrição

O bloco de memória tem a seguinte interface:

- um barramento de endereço (supor 12 bits de endereço)
- um barramento de dados de entrada (32 bits)
- um barramento de dados de saída (32 bits)
- sinal de habilitação de escrita (*wren write-enable*)

Com relação ao modelo VHDL, é sugerida a seguinte abordagem:

1. Utilizar uma descrição reutilizável. Isso pode ser obtido definindo-se a interface da seguinte maneira:

```
entity mem_rv is
  port (
    clock   : in  std_logic;
    wren    : in  std_logic;
    address : in  std_logic_vector;
    datain  : in  std_logic_vector;
    dataout : out std_logic_vector
  );
end entity mem_rv;
```

   Neste caso, os parâmetros devem ser obtidos a partir das propriedades dos sinais conectados às portas (

```
architecture RTL of mem_rv is
Type mem_type is array (0 to (2**address'length)-1) of std_logic_vector(datain'range);
signal mem : ram_type;
signal read_address : std_logic_vector(address'range);
```

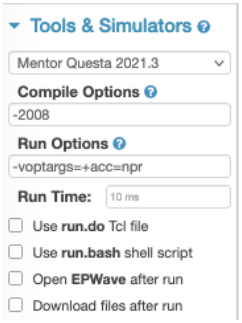
2. Permitir leitura depois da escrita, ou seja, se um endereço for escrito e lido ao mesmo tempo, o dado escrito é retornado pela leitura.

   Isso pode ser realizado atrasando-se um delta ciclo a leitura, utilizando-se o sinal interno read_address:

```
Write: process(clock) begin
        …
        if we = '1' then
              mem(to_integer(unsigned(address))) <= datain;
        end if;
        read_address <= address;
        …
       end process;

       dataout <= mem(to_integer(unsigned(read_address)));
```

3. A carga de instruções e dados deve ser feita a partir de um arquivo texto, contendo em cada linha uma palavra de 32 bits em hexadecimal, pode ser realizada conforme indicado no link abaixo:
   - https://vhdlwhiz.com/initialize-ram-from-file/
     obs: note que é necessário utilizar o padrão VHDL-2008. No ModelSim, para indicar a utilização desse padrão deve-se clicar com o botão direito do mouse nos arquivos e selecionar a opção PROPERTIES, aba VHDL. Selecionar 1076-2008.
     No EdaPlayground, selecionando-se Mentor Questa como simulador, basta indicar "-2008" (sem aspas) nas opções de compilação.



   - Para gerar os arquivos, usar a opção *Dump Memory* do RARS, no formato texto hexadecimal
   - O laço para leitura de arquivo pode ser implementado com o comando while:
     - `while not endfile(code_file) loop`

4. *Testbench* para verificação da memória:
   - Carregar um arquivo com instruções e dados para a memória e exibí-los. Os dados devem ser escritos, de acordo com o modelo compacto, a partir do endereço 2048 (0x2000 – endereço de *byte*).
   - A visualização das saídas pode ser feita com a janela de formas de onda ou imprimindo-se o conteúdo lido da memória na console, com auxílio do comando `to_hstring(signal)`.
   - A geração de estímulos pode ser feita com o auxílio do comando *for loop*:

```
for i in 0 to 255 loop
  address <= std_logic_vector(to_unsigned(i,8));
  datain  <= std_logic_vector(to_unsigned(i,30)) & "00";
  wait for 10 ps;
end loop;
```

5. Entregar apenas o código implementado incluindo *testbench*.