Arquitetura e Organização de Computadores
Turma C - 2024/02

Projeto das Memórias de Dados e Instruções

Objetivo: projetar, simular e sintetizar as memórias de dados e instruções do RISC-V.

Resumo

O RISC-V *uniciclo* utiliza duas memórias internas para armazenar programa e dados. Neste trabalho deverá ser desenvolvido um modelo VHDL para cada memória. A memória de dados deve ser uma memória tipo RAM, com processo de leitura e escrita. No caso da memória de instruções, os sinais de escrita não são necessários, pois é uma memória de apenas leitura (ROM), que deve ser iniciada com instruções lidas de um arquivo texto.

Descrição

O bloco de memória ROM tem a seguinte interface:

- um barramento de endereço (supor 11 bits de endereço)
- um barramento de dados de saída (32 bits)

A ROM é uma memória de palavras de 32 bits. Cada endereço localiza uma palavra. Desta forma, a ROM tem 2048 palavras de 32 bits, endereçadas através de 11 bits de endereço. No caso do RISC-V, cada endereço da ROM armazena uma instrução de 32 bits.

O bloco de memória RAM tem a seguinte interface:

- um barramento de endereço (supor 13 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*)
- sinal de indicação de acesso à byte (*byte_en*)
- sinal de indicação de acesso à byte com ou sem sinal (*sgn_en*)

A RAM é uma memória com entradas e saídas de 32 bits de dados mas internamente é constituída por um vetor de *bytes*, sendo endereçada a *byte*. Portanto, apesar da RAM ter 13 bits de endereço e a ROM ter 11 bits de endereço, ambas tem a mesma capacidade: 8196 bytes cada. O acesso a *byte* é indicado pelo sinal *byte_en* = 1, enquanto que o acesso a *word* é indicado por *byte_en* = 0. A leitura e escrita de *bytes* e *words* pode ser feita de forma similar ao simulador. Lembrar que os endereços de *words* devem ser múltiplos de 4. Por fim, a leitura de *bytes* com ou sem sinal é indicada pelo sinal *sgn_en*. Se *sgn_en* = 1, acesso com sinal (para *lb*), se *sgn_en* = 0, acesso sem sinal (para *lbu*).

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

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

```
entity ram_rv is
  port (
    clck    : in  std_logic;
    we      : in  std_logic;
    byte_en : in  std_logic;
    sgn_en  : in  std_logic;
    address : in  std_logic_vector;
    datain  : in  std_logic_vector;
    dataout : out std_logic_vector
  );
end entity ram_rv;

entity rom_rv is
  port (
    address : in  std_logic_vector;
    dataout : out std_logic_vector
  );
end entity rom_rv;
```

   Neste caso, os parâmetros devem ser obtidos a partir das propriedades dos sinais conectados às portas, como indicado abaixo. O sinal *mem* é definido como um vetor de palavras

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

2. No caso da memória de instruções, não existe processo de escrita, apenas a leitura das instruções. A carga das instruções 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:
   a. 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* a versão do VHDL é indicada na entrada "Compile Options" e, por padrão, é 2008.

3. *Testbench* para verificação das memórias:
   a. Memória de dados: escrever e ler uma sequência de valores, para os diferentes tamanhos de palavra:
      i. escrever / ler palavras
      ii. ler *bytes* das palavras com / sem sinal
      iii. a memória de dados pode ser iniciada em zero com o comando
         `(others => (others => '0')`

b. Memória de instruções: carregar um arquivo com instruções para a ROM e exibí-los.
c. A visualização das saídas pode ser feita com a janela de formas de onda do ModelSim / EdaPlayground.
d. O *clock* da memória é gerado dentro do *testbench*.
Uma forma simples de realizar a geração do clock é com um *processo*:

```
clk_gen: process
begin
  for i in 0 to <#ciclos> loop
    clk <= '0';
    wait for clk_period / 2;
    clk <= '1';
    wait for clk_period / 2;
  end loop;
  wait;
end process;
```

e. 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 1 ns;
end loop;
```

Entrega:

- Arquivo ".zip" tendo o número de matrícula como nome, incluindo os arquivos VHDL, o arquivo com as instruções lidas para a ROM.