PCS-3225 Sistemas Digitais II — 3 de abril de 2024 - v1

Multiplicador de Ponto Flutuante IEEE 754

Aula 8 - Exercício

Bruno Albertini, Edson Gomi e Ricardo Lera

Considere os seguintes módulos ROM e RAM (disponíveis também no edisciplinas):

ROM de 8 palavras:

```
module rom_8(
  input [2:0] addr,
  input OE,
  output reg [31:0] out
);
  reg [31:0] data[7:0];
  initial
  begin
    data[0] = 32'h0986ab68;
    data[1] = 32'h10385ba9;
    data[2] = 32'b0;
    data[3] = 32'b0;
    data[4] = 32'b0;
    data[5] = 32'b0;
    data[6] = 32'b0;
    data[7] = 32'b0;
  end
  always @(addr, OE)
    if (OE==1'b1)
      out=data[addr];
    else
      out=32'bz;
endmodule
```

RAM de 4 palavras:

```
module ram_4(
  input [31:0] in,
  input [1:0] addr,
  input RW, OE,
  output reg [31:0] out
);
  reg [31:0] data[3:0];
  always @(in, addr, RW, OE)
  begin
    if(RW==1'b0 & OE==1'b1)
      out=data[addr];
    else
      out=32'bz;
    if(RW==1'b1)
      data[addr]=in;
  end
endmodule
```

1 Parte I

Escreva no bloco Initial do módulo `rom_8` a representação em ponto flutuante dos valores listados na tabela ao lado para os endereços 0x2 a 0x7.

Salve o arquivo como `rom_8_completo.v` e envie no e-disciplinas.

ROM addr	ROM valores	RAM addr
0x0	fornecido	0x0
0x1	fornecido	
0x2	1	0x1
0x3	0.01_2	
0x4	3_{10}	0x2
0x5	10_{10}	
0x6	0.3125_{10}	0x3
0x7	0.875_{10}	

2 Parte II

Utilizando o módulo `fp_mult` disponível no ediscipliinas, projete o módulo `fpm` para ler os endereços da ROM dois a dois, multiplicá-los, e salvar o resultado no respectivo enderço da RAM indicado na tabela anterior.

Utilize o seguinte módulo como base:

```
1  module mfp(
2    input  [1:0]  ram_addr_juiz,
3    output [31:0] ram_out_juiz,
4    output reg    done
5  );
6    // Variaveis da ROM e RAM
7    reg  [2:0]  rom_addr;
8    reg  [1:0]  ram_addr;
9    wire [31:0] rom_out, ram_in;
10   reg         ram_RW, rom_OE, ram_OE;
11
12   // Variaveis do multiplicador
13   reg  [31:0] mul_a, mul_b;
14   reg         mul_en, mul_rst, mul_clk=0;
15   wire        mul_done;
16   wire [31:0] mul_z;
17
18   // Clock para o multiplicador
19   always
20     #1 mul_clk = ~mul_clk;
21
22   // Multiplexador para o juiz
23   wire [1:0] ram_addr_mux;
24   assign ram_addr_mux = (ram_RW ? ram_addr : ram_addr_juiz);
25
26
27   /* Sua resposta aqui */
28
29
30 endmodule
```