PCS-3225 Sistemas Digitais II — 20 de março de 2024 - v2

Somador de Ponto Flutuante IEEE 754

Aula 6 - 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
```

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

Utilizando o módulo `adder` disponível no edisciplinas, projete o módulo `fpa` para ler os endereços da ROM dois a dois, somá-los, e salvar o resultado no respectivo enderço da RAM indicado na tabela.

Defina o módulo a ser projetado como sendo:

```
module fpa(ram_addr, ram_out, done);
```

OBS: Note que a entrada `ram_addr` não pode ser usada diretamente pelo módulo para acessar a RAM, uma vez que inputs em Verilog não possuem registradores. Esta entrada existe para que o juiz possa acessar a RAM após o processo de soma e escrita for completo. Portanto, defina um wire para multiplexar a entrada do addr:

```
assign ram_addr_mux = (ram_RW ? addr_counter :  ram_addr);
```

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}	