Lab 04 时序与状态机

PB16051448 赵敏帆

实验目的

- 掌握阻塞与非阻塞赋值的特性
- 掌握状态机的使用
- 利用前三次实验的结果实现 ram 中数据的读出/写入。

实验平台

Vivado 2016.4

实验过程及结果

代码编写

ram 构建

使用 Vivado 自带的 IP，构建一个双端口（一个读端口和一个写端口），256*64bit 的 ram，并使用 `.coe` 文件对其赋初值。

`.coe` 文件如下：

```
MEMORY_INITIALIZATION_RADIX=10;
MEMORY_INITIALIZATION_VECTOR=
10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,6,-1;
```

regfile 寄存器文件

```
module regfile(
input clk,
input rst_n,
input [7:0]rAddr,
input [7:0]wAddr,
input [31:0]wDin,
output [31:0]rDout,
input wEna
    );
    reg [31:0]regfiles[0:255];
    reg [5:0] i;
    always@( negedge rst_n or posedge clk)
    if(!rst_n)begin
    regfiles[0]<=0;
    regfiles[1]<=0;
    end
    else if(wEna)
        regfiles[wAddr]<=wDin;
    assign rDout=regfiles[rAddr];
endmodule
```

ALU 模块

```
module ALU(
    input signed  [31:0] alu_a,
    input signed  [31:0] alu_b,
    input  [4:0] alu_op,
    output reg [31:0] alu_out
    );
        parameter     A_NOP= 5'h00;
        parameter    A_ADD= 5'h01;
        parameter    A_SUB= 5'h02;
        parameter    A_AND= 5'h03;
        parameter    A_OR = 5'h04;
        parameter    A_XOR= 5'h05;
        parameter    A_NOR= 5'h06;
          always@(*)
           case(alu_op)
            A_NOP: alu_out = 0;
            A_ADD: alu_out = alu_a+alu_b;
            A_SUB: alu_out = alu_a-alu_b;
            A_AND: alu_out = alu_a &alu_b;
            A_OR:  alu_out = alu_a|alu_b;
            A_XOR: alu_out = alu_a^alu_b;
            A_NOR: alu_out = ~(alu_a|alu_b);
            default: alu_out=0;
          endcase
endmodule
```

Control 模块

Control 模块的设计是本次实验的重点，通过 `count` 来进行计数，达到状态的转换，并在适当的状态进行读取和写入操作。整个过程可以分成五个状态。

```
module control(
input clk,
input rst_n,
output reg [31:0]out1,
output reg [31:0]out2,
output reg wena,
output reg [7:0]write,
output reg [7:0]read_ram,

output reg [7:0]write_reg,
output reg [4:0]alu_op,
input wire [31:0]read_data_ram,
output reg wen_ram
    );
    reg [2:0]count;
    reg [5:0]ram_control;
    reg [7:0]read_op;
    always @(posedge clk or negedge rst_n)
    if(!rst_n)
        begin
            read_op<=100;
            read_ram<=0;
           count<=0;
            write<=200;
            write_reg<=0;
            wena<=0;
            wen_ram<=0;
            alu_op<=0;
        end
   else
   begin
            case(count)
    3'h0:
    begin
    read_ram<=read_ram+1;
    count<=count+1;
    wena<=1;
    wen_ram<=0;
    end
    3'h1:
    begin
    read_ram<=read_op;
    ram_control<=read_ram;
    read_op<=read_op+1;
    count<=count+1;
    wena<=1;
    write_reg<=write_reg+1;
    wen_ram<=0;
    end
    3'h2:
    begin
    out1<=read_data_ram;
    write_reg<=write_reg+1;
    count<=count+1;
    wena<=1;
    wen_ram<=0;
    end
    3'h3:
    begin
    out2<=read_data_ram;
    wen_ram<=0;
    write<=write+1;

    count<=count+1;
    wena<=1;
    end
    3'h4:
    begin
    alu_op<=read_data_ram;
    write_reg<=0;
    read_ram<=ram_control+1;
    wen_ram<=1;
    wena<=0;
    count<=0;
    end
            endcase
    end
endmodule
```

top 模块

```



module top(
    input clk,
    input rst_n
    );
    wire [7:0]read;
    wire [7:0]write;
    wire [31:0]out1;
    wire [31:0]out2;
    wire [31:0]Din;
    wire [31:0]read_data;
    wire wen;
    wire [31:0]read_data_ram;
    wire [7:0]read_ram;
    wire [7:0]write_reg;
    wire wen_ram;
    wire [4:0]alu_op;
     control control1(
     .clk (clk),
     .rst_n (rst_n),
     .out1 (out1),
     .out2 (out2),
      .wena (wen),
     .write (write),
     .write_reg (write_reg),
     .alu_op (alu_op),
     .read_ram (read_ram),
     .read_data_ram (read_data_ram),
     .wen_ram (wen_ram)
     );
     regfile  regfile1(
           .clk (clk),
           .rst_n (rst_n),

           .rAddr (read),
            .wAddr (write_reg),
            .wDin (read_data_ram),
            .rDout (read_data),
            .wEna (wen)
            );
      ALU alu(
      .alu_a (out1),
      .alu_b (out2),
      .alu_op (alu_op),
      .alu_out (Din)
      );
      blk_mem_gen_0 ram1(
       .clka (clk),
         .ena  (wen_ram),
         .wea  (wen_ram),
         .addra (write),
         .dina  (Din),
         .clkb  (clk),
         .enb (1),
         .addrb (read_ram),
         .doutb (read_data_ram)
      );
endmodule
```

仿真

仿真代码

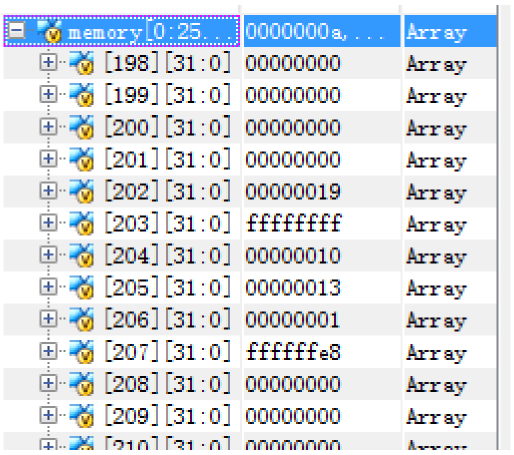
```
module test(

    );
    reg clk;
        reg rst_n;

        top test(
        .clk (clk),
        .rst_n (rst_n)
        );
        always #10 clk=~clk;
        initial begin
        clk=0;
        rst_n=0;
        #10
        rst_n=1;
        end
endmodule
```

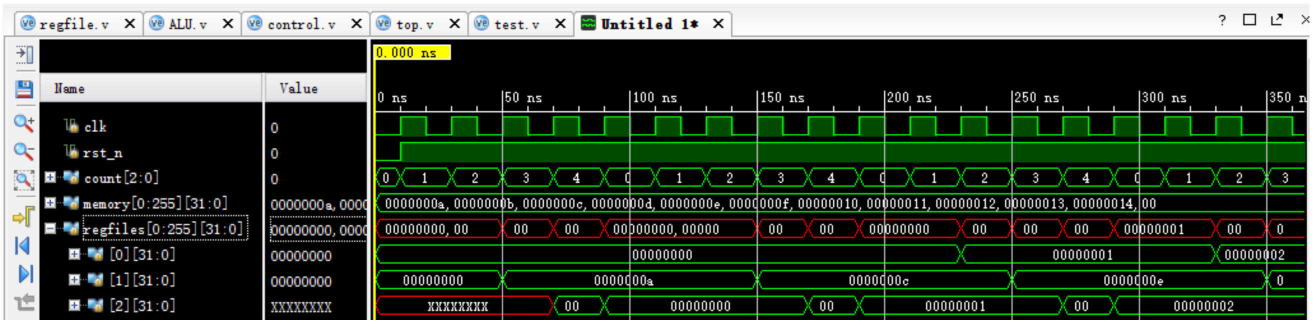
仿真结果

RAM 地址 200 附近的内容：



```
memory[0:25...   0000000a,...   Array
  [198][31:0]    00000000       Array
  [199][31:0]    00000000       Array
  [200][31:0]    00000000       Array
  [201][31:0]    00000000       Array
  [202][31:0]    00000019       Array
  [203][31:0]    ffffffff       Array
  [204][31:0]    00000010       Array
  [205][31:0]    00000013       Array
  [206][31:0]    00000001       Array
  [207][31:0]    ffffffe8       Array
  [208][31:0]    00000000       Array
  [209][31:0]    00000000       Array
  [210][31:0]    00000000       Array
```

周期及其他



```
Name                        Value
clk                         0
rst_n                       0
count[2:0]                  0
memory[0:255][31:0]         0000000a, 0000
regfiles[0:255][31:0]       00000000, 0000
  [0][31:0]                 00000000
  [1][31:0]                 00000000
  [2][31:0]                 XXXXXXXX
```

心得体会

- RAM 在使用时要注意读取数据和写入数据的时序关系，RAM 从拿到地址到读出地址要经过两个周期。