

# Lab 03 存储器 RAM

PB16051448 赵敏帆

## 实验目的

- 学习使用 ISE/Vivado 的 IP 核
- 学习使用 Xilinx FPGA 内的 RAM 资源
- 从 RAM 中读取数据作为斐波那契数列初值，进行斐波那契运算并将结果存在 Regfile 和 RAM 中

## 实验平台

Vivado 2016.4

## 实验过程及结果

### 编写代码

#### RAM 构建

利用 Vivado 自带的 IP 核，创建一个 64\*32bit 的简单双端口 RAM，并用 `.coe` 文件对其初始化。

`.coe` 文件如下：

```
MEMORY_INITIALIZATION_RADIX=10;
MEMORY_INITIALIZATION_VECTOR=
2,2;
```

#### Regfile 搭建

```
module regfile(
    input clk,
    input rst_n,
    input [5:0]rAddr,
    input [5:0]wAddr,
    input [31:0]wDin,
    output [31:0]rDout,
    input wEna
);
    reg [31:0]regfiles[0:63];
    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 部件

```

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

```

module control(
    input clk,
    input rst_n,
    input [31:0]read_data,
    output reg [31:0]out1,
    output reg [31:0]out2,
    output reg [5:0]read,
    output reg wena,
    output reg [5:0]write,
    output reg [5:0]read_ram,
    input wire [31:0]read_data_ram,
    output reg wen_ram
);
    reg [1:0]count;
    reg [5:0]ram_control;
    always @(posedge clk or negedge rst_n)
        if(!rst_n)

```

```

begin
    count<=0;
    read<=0;
    wena<=0;
    out1<=0;
    out2<=0;
    ram_control<=0;
    read_ram<=0;
    wen_ram<=0;
end
else if(read<63)
begin
    if(ram_control==0)begin
        read_ram<=0;
        wena<=0;
        ram_control<=ram_control+1;
    end
    else if (ram_control==1)begin
        ram_control<=ram_control+1;
        wena<=0;
    end
    else if (ram_control==2)begin
        ram_control<=ram_control+1;
        out1<=read_data_ram;
        out2<=0;
        write<=0;
        wena<=0;
    end
    else if (ram_control==3)begin
        ram_control<=ram_control+1;
        wena<=1;
    end
    else if(ram_control==4)begin
        read_ram<=1;
        wena<=0;
        ram_control<=ram_control+1;
    end
    else if (ram_control==5)begin
        ram_control<=ram_control+1;
        wena<=0;
    end
    else if (ram_control==6)begin
        ram_control<=ram_control+1;
        wena<=0;
        out1<=read_data_ram;
        out2<=0;
        write<=1;
    end
    else if (ram_control==7)begin
        ram_control<=ram_control+1;
        wena<=1;
    end

else

```

```

        case(count)
2'h0:
begin
out1<=read_data;
read<=read+1;
count<=count+1;
wena<=0;
wen_ram<=0;
end
2'h1:
begin
out2<=read_data;
write<=read+1;
count<=count+1;
wena<=0;
wen_ram<=0;
end
2'h2:
begin
wena<=1;
count<=0;
wen_ram<=1;
end
        endcase
    end
endmodule

```

由于从 RAM 读取数据有两个周期的延迟，所以使用一个状态机来从 RAM 中读取初始的斐波那契数列值并存入 Regfile 中。然后使用状态机对其进行斐波那契数列计算。

## top 模块

```

module top(
    input clk,
    input rst_n
);
wire [5:0]read;
wire [5:0]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 [5:0]read_ram;
wire wen_ram;
    control control1(
        .clk (clk),
        .rst_n (rst_n),
        .read_data (read_data),
        .out1 (out1),
        .out2 (out2),

```

```

        .read (read),
        .wena (wen),
        .write (write),
        .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),
        .wDin (Din),
        .rDout (read_data),
        .wEna (wen)
    );
    ALU alu(
        .alu_a (out1),
        .alu_b (out2),
        .alu_op (5'h1),
        .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

```

## 仿真文件

```

module test(

);
reg clk;
reg rst_n;

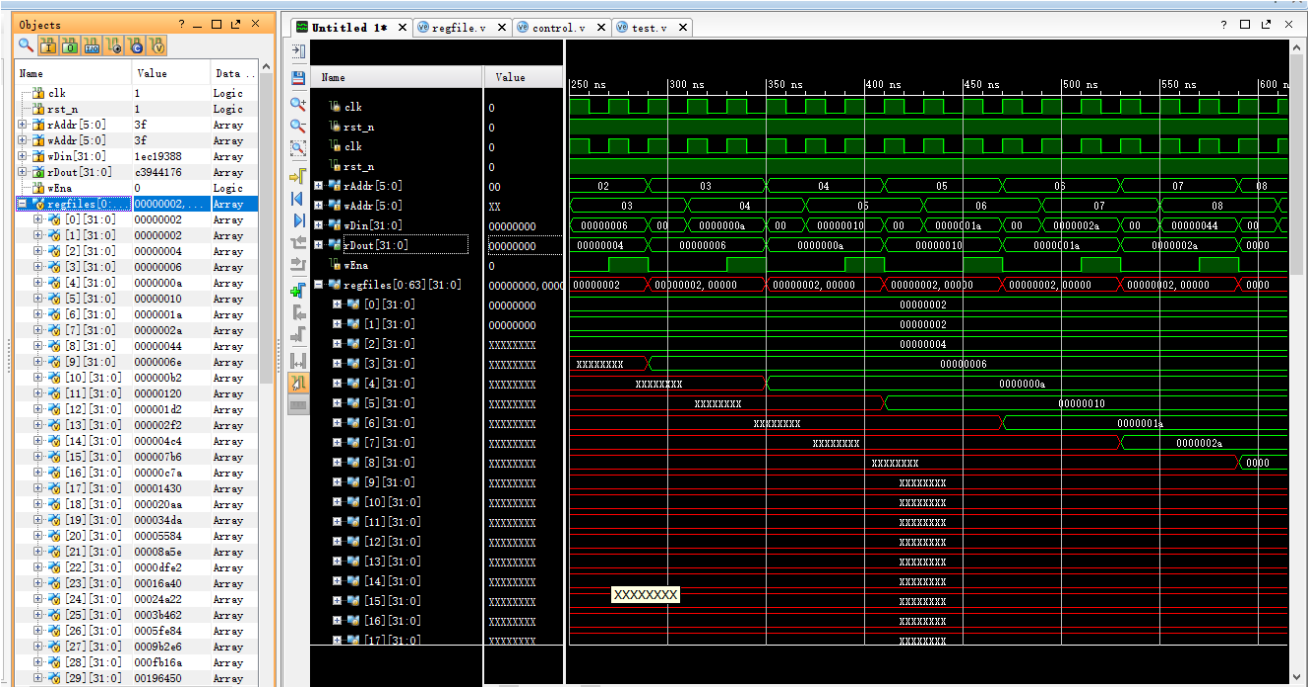
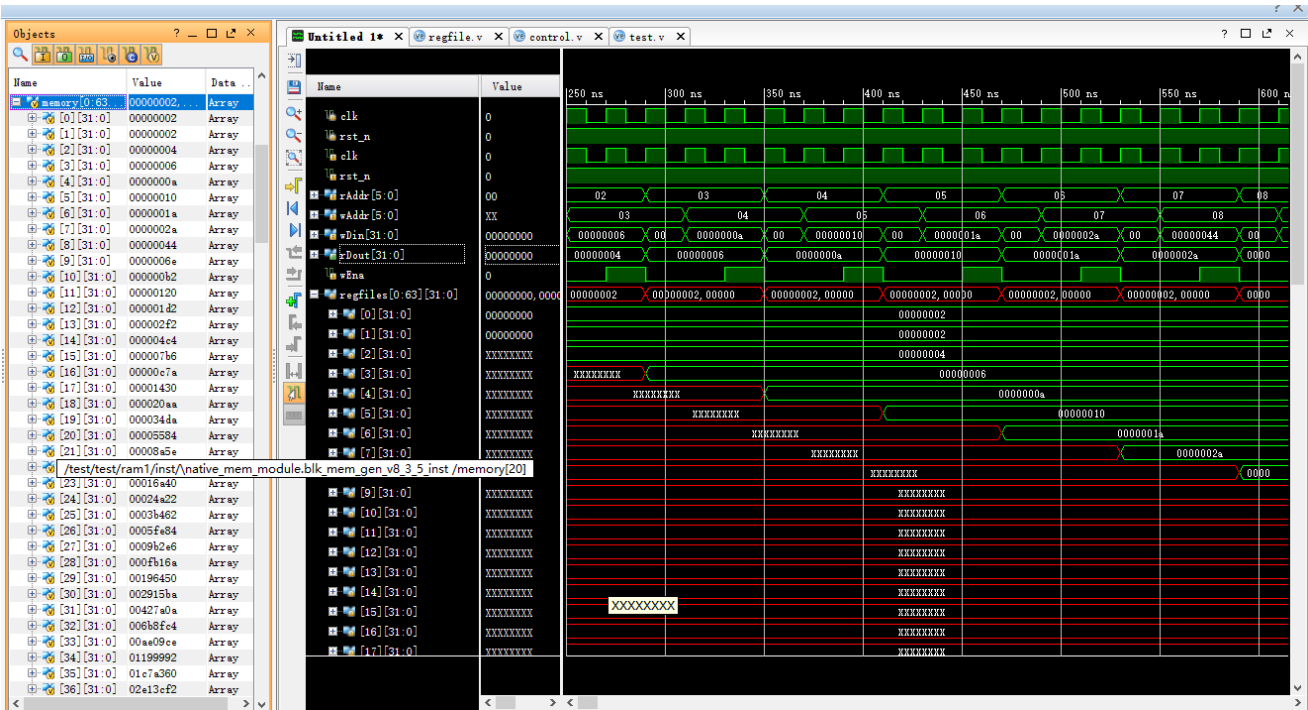
top test(
    .clk (clk),
    .rst_n (rst_n)
);
always #10 clk=~clk;
initial begin
    clk=0;
    rst_n=0;

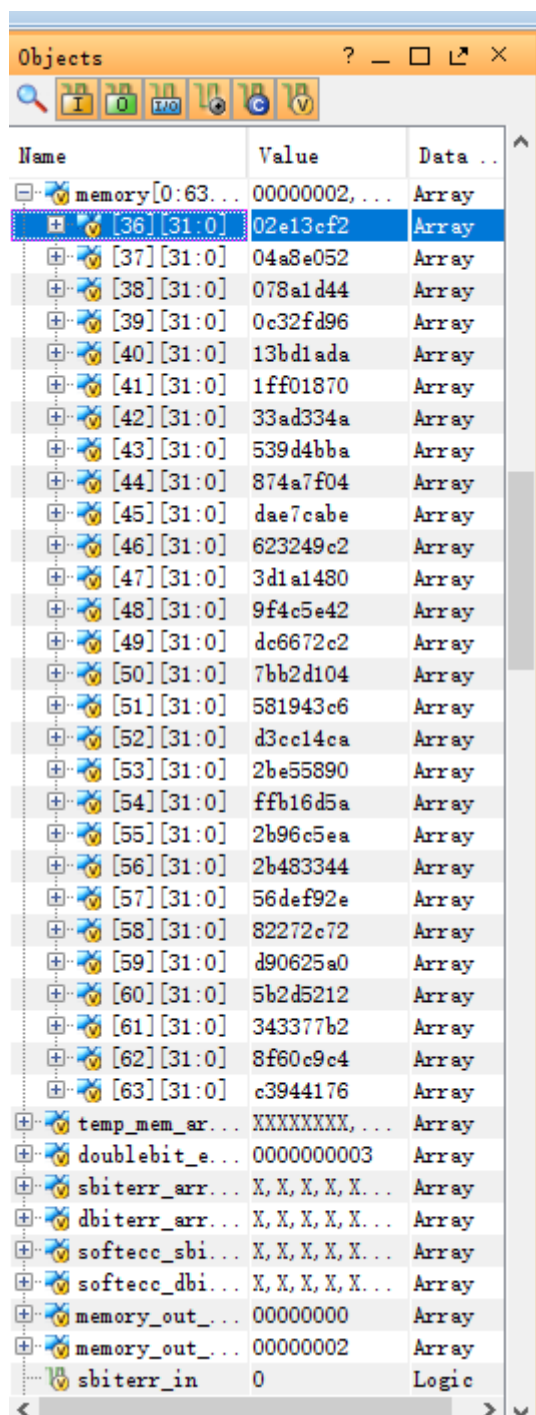
#10

```

```
rst_n=1;
end
endmodule
```

## 仿真结果





Name	Value	Data ..
memory[0:63...	00000002,...	Array
[36] [31:0]	02e13cf2	Array
[37] [31:0]	04a8e052	Array
[38] [31:0]	078a1d44	Array
[39] [31:0]	0c32fd96	Array
[40] [31:0]	13bd1ada	Array
[41] [31:0]	1ff01870	Array
[42] [31:0]	33ad334a	Array
[43] [31:0]	539d4bba	Array
[44] [31:0]	874a7f04	Array
[45] [31:0]	dae7cabe	Array
[46] [31:0]	623249c2	Array
[47] [31:0]	3d1a1480	Array
[48] [31:0]	9f4c5e42	Array
[49] [31:0]	dc6672c2	Array
[50] [31:0]	7bb2d104	Array
[51] [31:0]	581943c6	Array
[52] [31:0]	d3cc14ca	Array
[53] [31:0]	2be55890	Array
[54] [31:0]	ffb16d5a	Array
[55] [31:0]	2b96c5ea	Array
[56] [31:0]	2b483344	Array
[57] [31:0]	56def92e	Array
[58] [31:0]	82272c72	Array
[59] [31:0]	d90625a0	Array
[60] [31:0]	5b2d5212	Array
[61] [31:0]	343377b2	Array
[62] [31:0]	8f60c9c4	Array
[63] [31:0]	c3944176	Array
temp_mem_ar...	XXXXXXXX,...	Array
doublebit_e...	0000000003	Array
sbiterr_arr...	X, X, X, X, X...	Array
dbiterr_arr...	X, X, X, X, X...	Array
softccc_sbi...	X, X, X, X, X...	Array
softccc_dbi...	X, X, X, X, X...	Array
memory_out_...	00000000	Array
memory_out_...	00000002	Array
sbiterr_in	0	Logic

## 心得体会

- 使用状态机可以比较容易地理清时序关系
- Vivado 可能会出现 BUG，仿真出已经删除的线，这时候重启可以解决这个 BUG。