计算机组成原理 实验报告

姓名：杨博涵　学号：PB20000328　实验日期：2022.3.24

一、实验题目：

Lab02 寄存器堆与FIFO

二、实验目的：

掌握寄存器堆（Register File）和存储器的功能、时序及其应用，熟练掌握数据通路和控制器的设计和描述方法。

三、实验平台：

Vivado

四、实验过程：

源码在最后给出。

1.寄存器堆

使用参数描述DWidth与Awidth，方便不同需求下的例化。

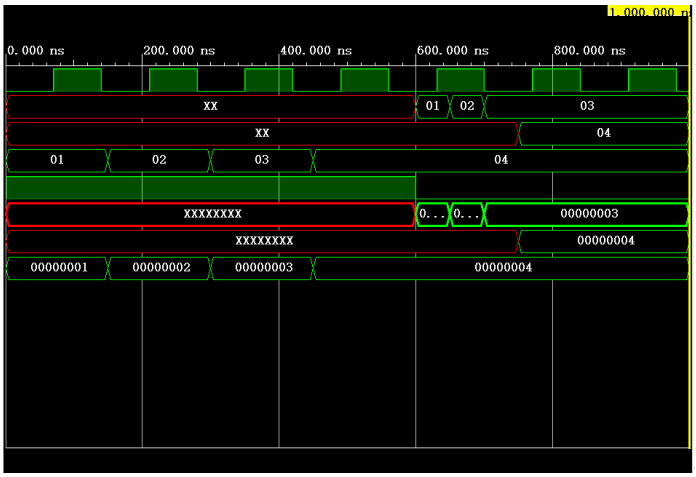
采用双端异步读和单端同步使能写的访问设计

源码：

```
module rf
#(parameter DWidth=32,parameter Awidth=5)
(
  input clk,
  input [Awidth-1:0] ra0,
  input [Awidth-1:0] ra1,
  input [Awidth-1:0] wa,
  input we,
  output [DWidth-1:0] rd0,
  output [DWidth-1:0] rd1,
  input [DWidth-1:0] wd
);

  reg [DWidth-1:0] RF[2**Awidth:0];

assign rd0 = RF[ra0], rd1 = RF[ra1];
```

```
always@(posedge clk)
begin
  if (we)  RF[wa]<=wd;
end

endmodule
```

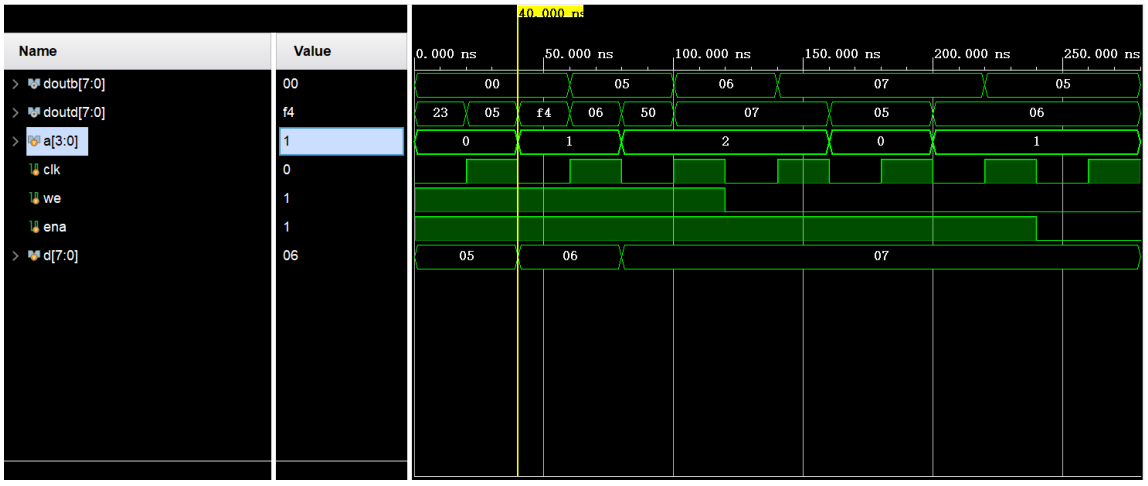
仿真结果：



2. RAM 存储器

直接例化 IP 核即可，分布式 RAM 是异步读，而块式是同步读，其余一样，下图中 doutb 被 registered。

仿真结果：



3.FIFO 设计

先讨论 moore 型控制器设计。采用 reg counter 记录存入数据数，用来判断空仓和满仓的特殊状态，并进行约束处理。

reg state 记录读写状态（IDLE，ENA，DEA），根据状态产生 we 和 re 信号。

Head 和 Tail 记录位置。

输入的 deq 和 enq 必须取边沿否则多个 clk 都会重复处理。

SDU 模块采用独立的降频时钟域和控制器，对数码管进行驱动。

源码：

```
`timescale 1ns / 1ps
`define EMP  4'b0
`define FULL 4'b1000
`define MIDDLE_in  1'b1
`define MIDDLE_out 3'b111
`define ENA 2'b01
`define DEA 2'b00
`define IDLE 2'b10


module FIFO(
  input clk, rst,    //时钟（上升沿有效）、同步复位（高电平有效）
  input enq,           //入队列使能，高电平有效
  input [3:0] in,      //入队列数据
  input deq,           //出队列使能，高电平有效
  output [3:0] out,  //出队列数据
  output [2:0] an,   //数码管选择
  output [3:0] seg, //数码管数据
  output full,emp
    );
  wire clk_seg,we;
  wire [2:0] ra1,ra0,wa,head,tail;
  wire [3:0] rd1,rd0,wd,cnt;
```

```
    rf #(.DWidth(4),.Awidth(3))
regfile(.clk(clk),.ra0(ra0),.ra1(ra1),.wa(wa),.we(we),.rd0(rd0),.rd1(rd
1),.wd(wd));
    clk_div divCLK(rst,clk,clk_seg);
    SDU sdu(rd1,head,tail,cnt,ra1,clk_seg,an,seg);
    LCU
lcu(.clk(clk),.rst(rst),.enq(enq),.in(in),.deq(deq),.out(out),.full(ful
l),.emp(emp),.we(we),.rd0(rd0),.ra0(ra0),.wd(wd),.wa(wa),.head(head),.t
ail(tail),.counter(cnt));

    endmodule



    module clk_div(  //时间太长无法仿真
      input rst,
      input clk,
      output clk_seg
    );
      reg [19:0] counter;
      reg CLK;

    always@(posedge clk)
    begin
      if(rst==1) begin counter<=0; CLK<=0; end
      else if(counter==125000) begin counter<=0; CLK<=~CLK; end
      else counter<=counter+1;
    end

    assign clk_seg=CLK;
    endmodule
```

```
module SDU(
  input [3:0] d,
  input [2:0] head,tail,
  input [3:0] cnt,
  output [2:0] a,
  input clk,
  output [2:0] an,
  output [3:0] seg
);
  reg [2:0] counter;

always@(posedge clk)
begin
  if(cnt==0) begin counter<=0; end
  else if(counter+tail+1'b1==head) begin counter<=0; end
  else begin counter<=counter+1; end
end

assign a=counter+tail+1'b1;
assign an=cnt==0?0:counter;
assign seg=cnt==0?0:d;

endmodule


module LCU(
  input clk, rst,//时钟（上升沿有效）、同步复位（高电平有效）
  input enq,           //入队列使能，高电平有效
  input [3:0] in,      //入队列数据
  input deq,           //出队列使能，高电平有效
```

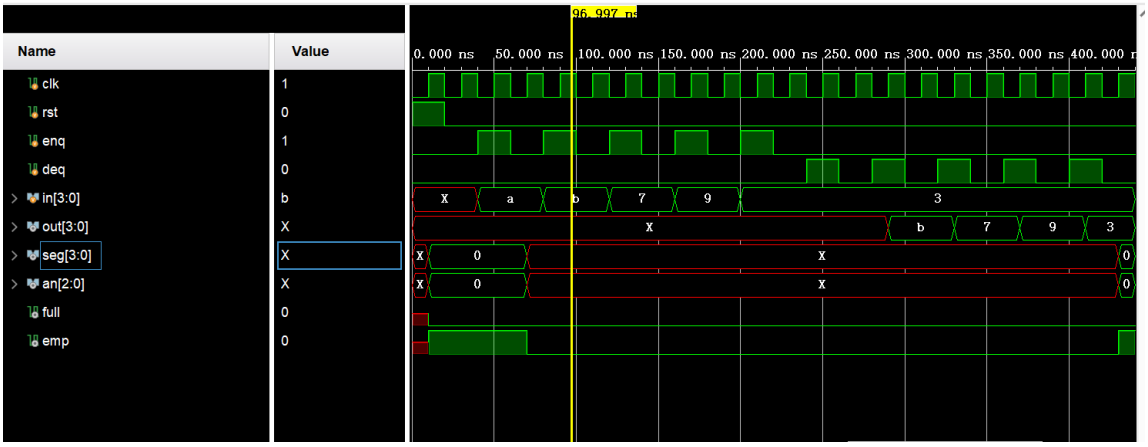
```
  output reg [3:0] out,  //出队列数据
  output reg full,emp,
  output reg we,
  input [3:0] rd0,
  output [2:0] ra0,
  output [3:0] wd,
  output [2:0] wa,
  output reg [2:0] head,tail,
  output reg [3:0] counter
);

  reg [2:0] next_head,next_tail;
  reg [3:0] outreg,next_counter;
  reg [1:0] state,next_state;
//reg we_b,re_b;
  wire enq_eff,deq_eff;
  reg re;
  wire[3:0] next_outreg;

signal_edge edger1(clk,deq,deq_eff);
signal_edge edger2(clk,enq,enq_eff);
//signal_edge edger3(clk,we_b,we);
//signal_edge edger4(clk,re_b,re);

assign wd=in;
assign wa=tail+1'b1; //小时序问题
assign next_outreg=rd0;
assign ra0=deq_eff==1 ? head : head+1'b1; //fix

always@(*)
begin
  if(enq_eff==1)
```

```
      begin
            if(counter==`FULL) begin next_counter=`FULL;
next_state=`IDLE; next_head=head; next_tail=tail; end
            else begin next_counter=counter+1; next_tail=tail-1;
next_state=`ENA; next_head=head;end
      end
      else if(deq_eff==1)
      begin
           if(counter==`EMP) begin next_counter=`EMP; next_state=`IDLE;
next_head=head; next_tail=tail;end
            else begin next_counter=counter-1; next_head=head-1;
next_state=`DEA; next_tail=tail;end
      end
     else begin next_state=`IDLE; next_counter=counter; next_head=head;
next_tail=tail; end
    end

    always@(posedge clk)
    begin
      if(rst==1)
      begin
            state<=`DEA;
            counter<=`EMP;
            outreg<=0;
            head<=0;
            tail<=0;
      end

      else
      begin
            state<=next_state;
            counter<=next_counter;
```

```
        head<=next_head;
        if(re==1'b1) out<=next_outreg;
        tail<=next_tail;
  end
end

always@(*)
begin
  if(counter==`FULL) full=1'b1;
  else full=1'b0;
  if(counter==`EMP)  emp=1'b1;
  else emp=1'b0;

  if(state==`ENA)
        begin
              we=1'b1;
              re=1'b0;
        end
  else if(state==`DEA)   //errors
        begin
              re=1'b1;
              we=1'b0;
        end
  else
        begin
              re=1'b0;
              we=1'b0;
        end

end
endmodule
```

```
module signal_edge   //取边沿对于比较短（clk 左右）的信号可能致命
(
  input clk,
  input button,
  output button_edge
);

reg button_r1,button_r2;
always@(posedge clk)
  button_r1<=button;
always@(posedge clk)
  button_r2<=button_r1;

assign button_edge=button_r1&(~button_r2);

endmodule



//第一次上板
//1. 未弹出时就已经在 led 上显示，可能接口 reg 没有调好       fixed
//2. 输入时总是先由上一个值，感觉接口 reg 没做好
//
```

仿真结果：



在实验中由于控制器十分复杂，出现了多次 bug 并进行修复：

1. 仿真中出现大量不定信号————没有复位就开始仿真，而且 latch 复位容易忽略。
2. IDLE 一开始没有引入，导致多周期重复工作。
3. 未弹出时就已经在 led 上显示，可能接口 reg 没有调好 和 输入时总是先由上一个值，感觉接口 reg 没做好————是时序出现问题，因为数据通路需要花 2 个 clk 才能写入，而 wa 与 ra0 都是一个 clk 就变，所以出现时序问题，修改为 wa=tail+1'b1 和 ra0=deq_eff==1 ? head : head+1'b1;即可（fake pipelined，**不是单周期的**）
4. Clk 级时序一开始出现混乱