中国科学技术大学计算机学院
《计算机组成原理实验报告》

学生姓名：林宸昊
学生学号：PB20000034
实验日期：2022.3.29

【实验题目】寄存器堆与存储器及其应用

【实验目的】

- 能够自行描述寄存器堆并进行仿真与应用；
- 能够例化存储器IP核并进行仿真与应用；
- 能够设计与实现FIFO队列结构。

【实验平台】Vivado、FPGAOL

【实验步骤】

【一、寄存器堆仿真】

- 设计文件

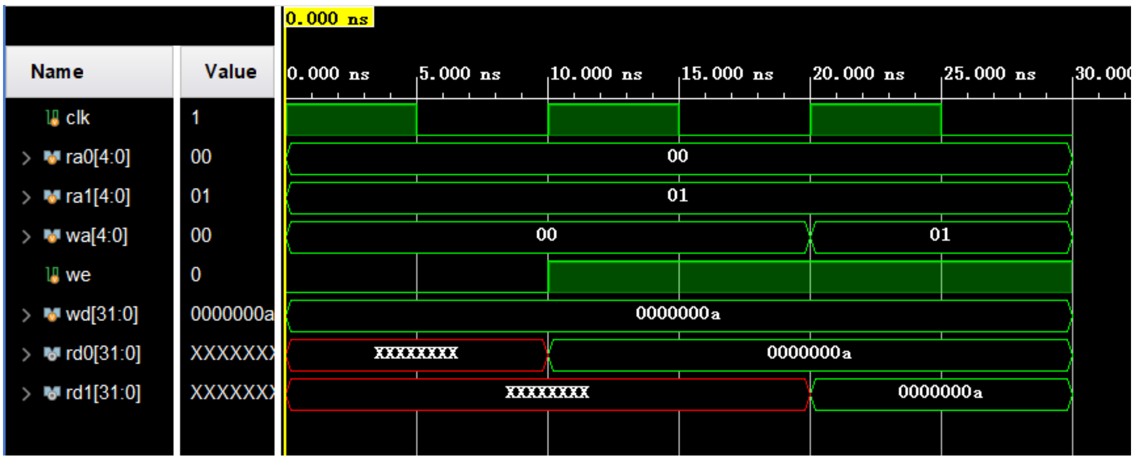
```
module RF1(
    input clk,
    input [4:0] ra0,
    input [4:0] ra1,
    input [4:0] wa,
    input we,
```

```
    input [31:0] wd,
    output [31:0] rd0,
    output [31:0] rd1
);
    reg [31:0] regfile[4:0];
    assign rd0 = regfile[ra0];
    assign rd1 = regfile[ra1];

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

- 仿真文件

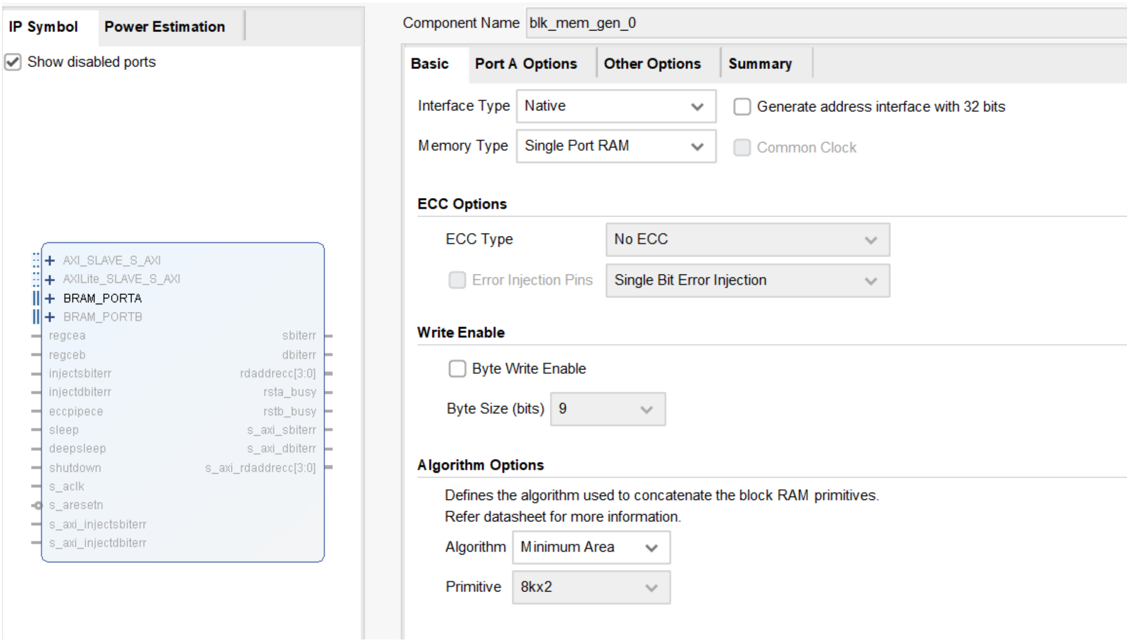
```
module RF_SIM(
    );
    reg clk;
    reg [4:0] ra0;
    reg [4:0] ra1;
    reg [4:0] wa;
    reg we;
    reg [31:0] wd;
    wire [31:0] rd0;
    wire [31:0] rd1;
    RF1 RR(clk, ra0, ra1, wa, we, wd, rd0, rd1);
    initial clk = 1;
    always #5 clk = ~clk;
    initial
    begin
        ra0 = 0;
        ra1 = 1;
        wa = 0;
        wd = 10;
        we = 0; #10 we = 1;
        #10 wa = 1;
        #10 $finish;
    end
endmodule
```

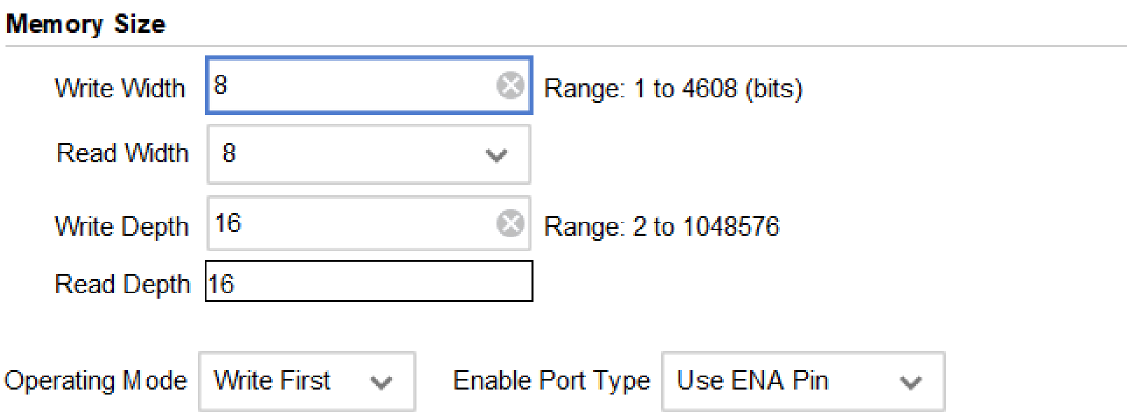
- 仿真结果



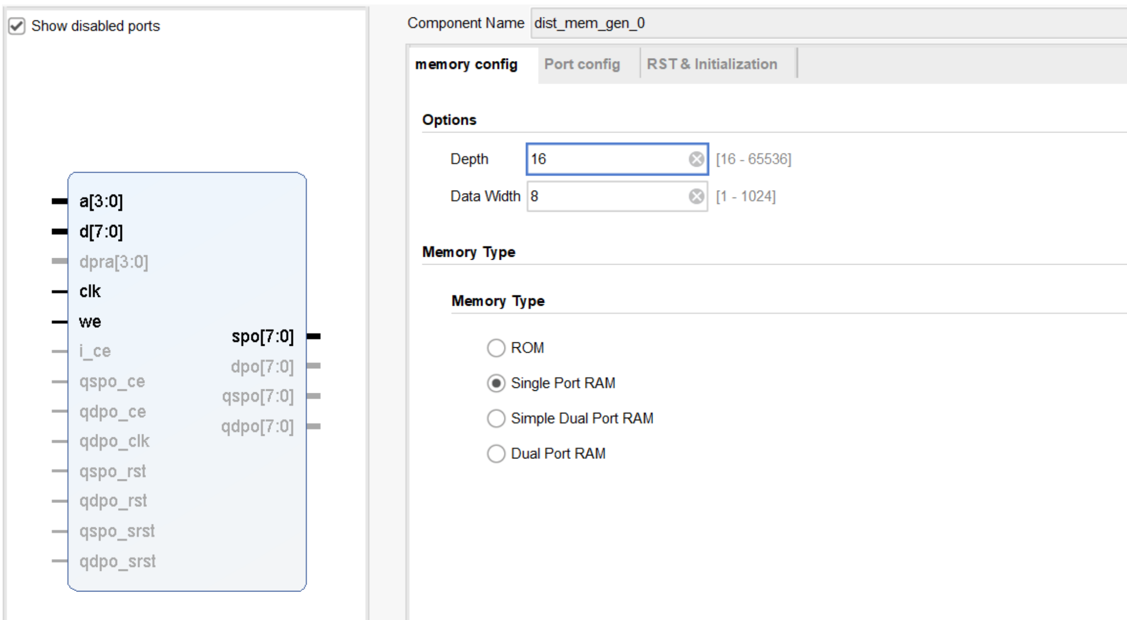
【二、IP例化存储器】

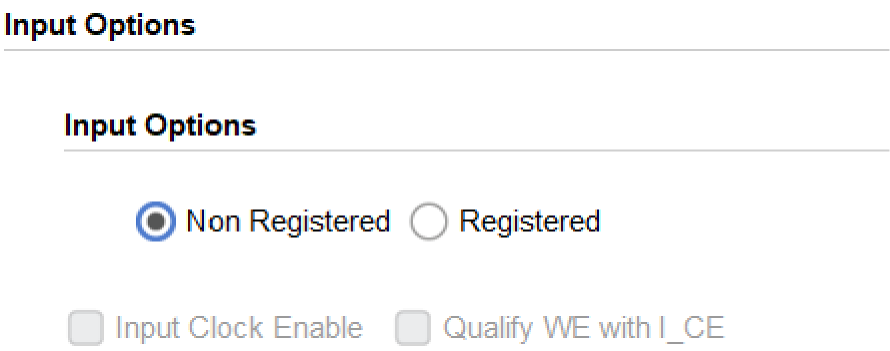
- 块式存储器例化（分布式类似）





Key	Value
memory_initialization_radix	16
memory_initialization_vector	00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

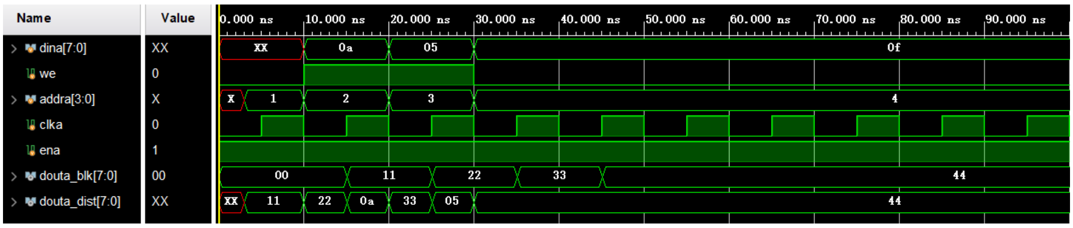




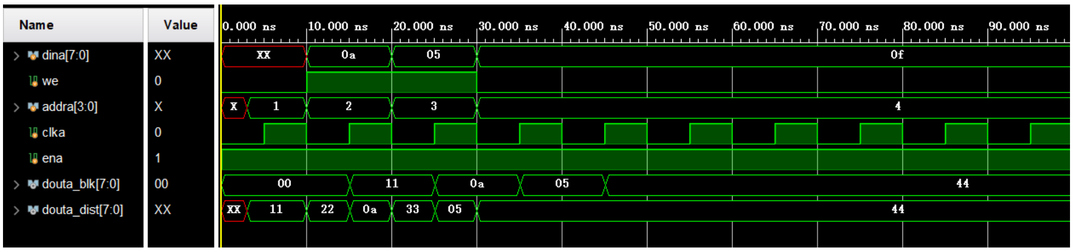
- 仿真文件

```
module IP_sim(
    );
    reg [7:0] dina;
    reg we;
    reg [3:0] addra;
    reg clka;
    reg ena;
    wire [7:0] douta_blk;   //块式存储器输出
    wire [7:0] douta_dist;  //分布式存储器输出
    blk_mem_gen_0 blk1(.addra(addra),
                       .clka(clka),
                       .dina(dina),
                       .douta(douta_blk),
                       .ena(ena),
                       .wea(we));
    dist_mem_gen_0 dist1(.a(addra),
                         .d(dina),
                         .clk(clka),
                         .we(we),
                         .spo(douta_dist));
    initial clka = 0;
    always #5 clka = ~clka;
    initial
    begin
        ena = 1; we = 0;
        #10 we = 1; dina = 10;
        #10 dina = 5;
        #10 we = 0; dina = 15;
    end
    initial
    begin
        #3 addra = 1;
        #7 addra = 2;
        #10 addra = 3;
        #10 addra = 4;
    end
endmodule
```

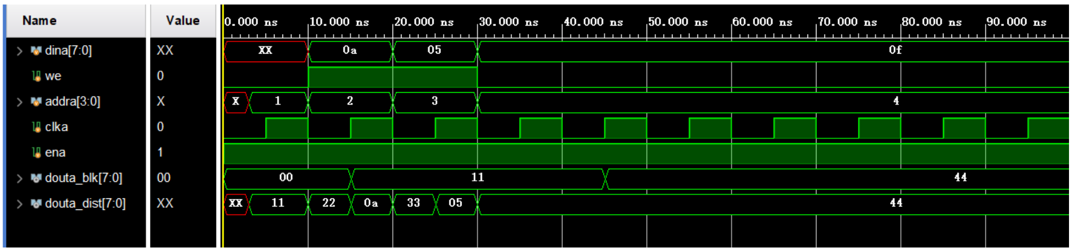
- 仿真结果
    - Read First



- Write First



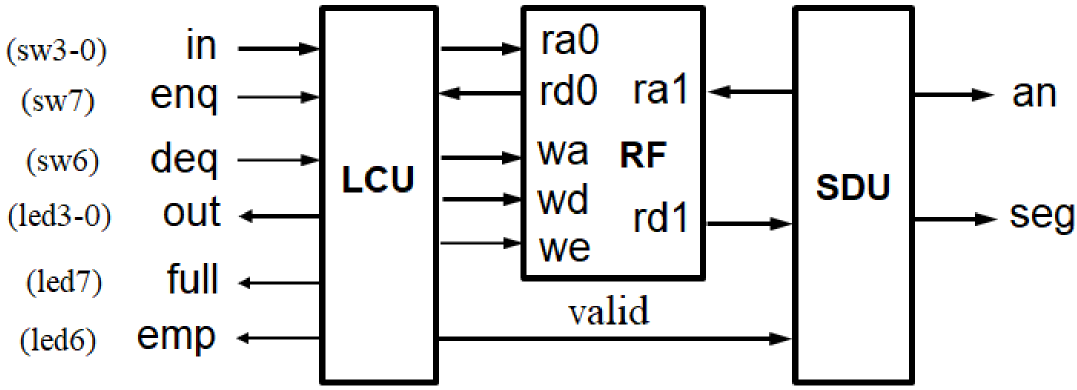
- No Change



- 可以发现，块式存储器拥有三种读写模式而分布式并没有；分布式存储器的读出与时钟信号无关，只要地址改变就会改变输出的值，属于异步，而块式是同步。

【三、FIFO队列实现】

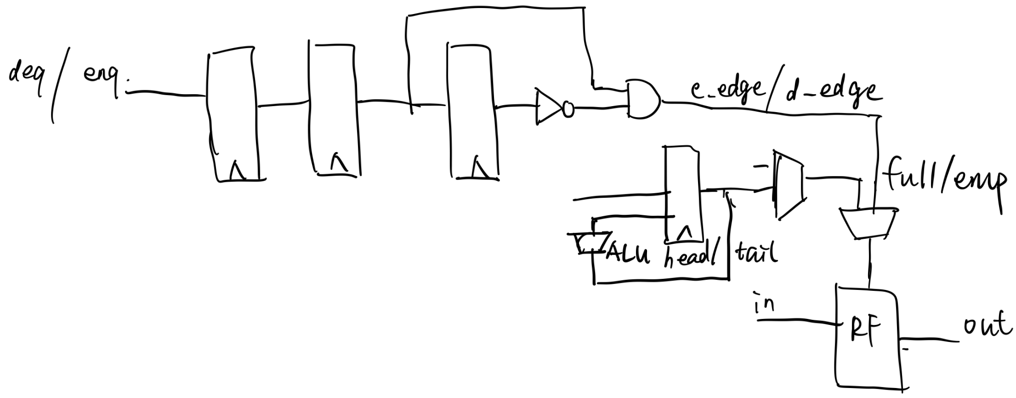
- 数据通路及控制器



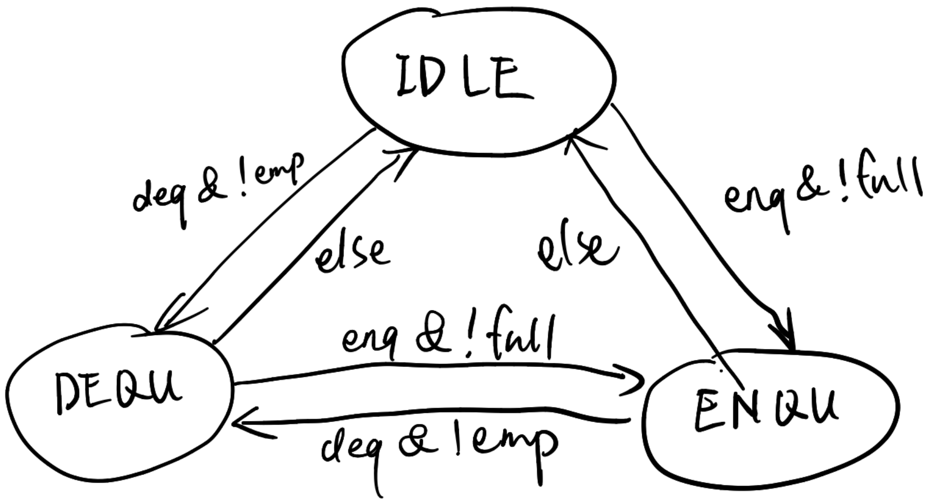
* 省略了clk（100MHz）和 rst（button）

其中RF已由上述步骤描述，SDU模块仅涉及hexplay的输出，主要设计LCU模块。

- LCU（大致数据通路）



- 状态图



- 设计文件

```
module signal(  //取信号边沿
    input clk,button,
    output button_edge);
    reg b1, b2;
    always@(posedge clk)
    b1 <= button;
    always @ (posedge clk)
    b2 <= b1;
    assign button_edge = b1 & (~b2);
endmodule

module sync(    //二级同步模块（也可不使用）
    input clk,
    input s_src,
    output reg s_dst
    );
    reg meta;
    always @ (posedge clk)
    begin
        meta <= s_src;
        s_dst <= meta;
    end
```

```
endmodule

module FIFO(
    input clk, rst,
    input enq,
    input [3:0] in,
    input deq,
    output reg [2:0] an,
    output reg [3:0] seg,
    output reg [7:0] led
    );

    wire e_edge, d_edge;//信号边沿
    wire e_dst, d_dst;  //二级同步信号
    reg full = 0;       //队满标志
    reg emp = 1;        //队空标志
    signal edge1(clk, enq, e_edge);
    signal edge2(clk, deq, d_edge);
    sync sy1(clk, e_edge, e_dst);
    sync sy2(clk, d_edge, d_dst);
    reg [1:0] cs;       //当前状态
    reg [1:0] ns;       //接续状态
    parameter IDLE = 2'b00;//空闲状态
    parameter DEQU = 2'b01;//出队状态
    parameter ENQU = 2'b10;//入队状态

    reg [2:0] head = 0;     //队头
    reg [2:0] tail = 0;     //队尾
    reg [3:0] out;
    reg [7:0] valid = 0;    //判空标志数组
    reg [3:0] regfile[7:0]; //实际用于存储的寄存器堆

    always @ (*)
    begin
        case(cs)
            IDLE:
            begin
                if(e_dst && !full)      //入队按钮有效且非满
                    ns = ENQU;
                else if(d_dst && !emp)  //出队按钮有效且非空
                    ns = DEQU;
                else ns = IDLE;
            end
            DEQU:
            begin
                if(e_dst && !full)
                    ns = ENQU;
                else if(d_dst && !emp)
                    ns = DEQU;
                else ns = IDLE;
            end
            ENQU:
            begin
                if(e_dst && !full)
                    ns = ENQU;
                else if(d_dst && !emp)
                    ns = DEQU;
                else ns = IDLE;
```

```
        end
    endcase
end

//LCU & RF
always @ (posedge clk or posedge rst)
begin
    if(rst)
        cs <= IDLE;
    else
        cs <= ns;
end

always @ (posedge clk or posedge rst)
begin
    if(rst)
    begin
        valid <= 0;
        full <= 0;
        emp <= 1;
    end
    else
        case(cs)
            ENQU:
            begin
                regfile[tail] <= in;    //输入
                valid[tail] <= 1;       //将当前位置标记为非空
                tail <= (tail + 1);     //队尾进一
                full <= ((tail + 1 == 8)||(tail + 1 == head));
                emp <= 0;
            end
            DEQU:
            begin
                out <= regfile[head];   //输出
                valid[head] <= 0;       //将当前位置标记为空
                head <= head + 1;       //队头进一
                emp <= ( (head + 1 == 8) || (head + 1 == tail));
                full <= 0;
            end
        endcase
    end
//SDU 利用高频闪烁达到同时显示效果
always @ (posedge clk)
begin
    led[7] <= full;
    led[6] <= emp;
    led[3:0] <= out;
end

reg [9:0] cnt;

always @ (posedge clk)
begin
    if(cnt >= 10'd800)
        cnt <= 0;
    else
        cnt <= cnt + 1;
    if(valid == 0)
```

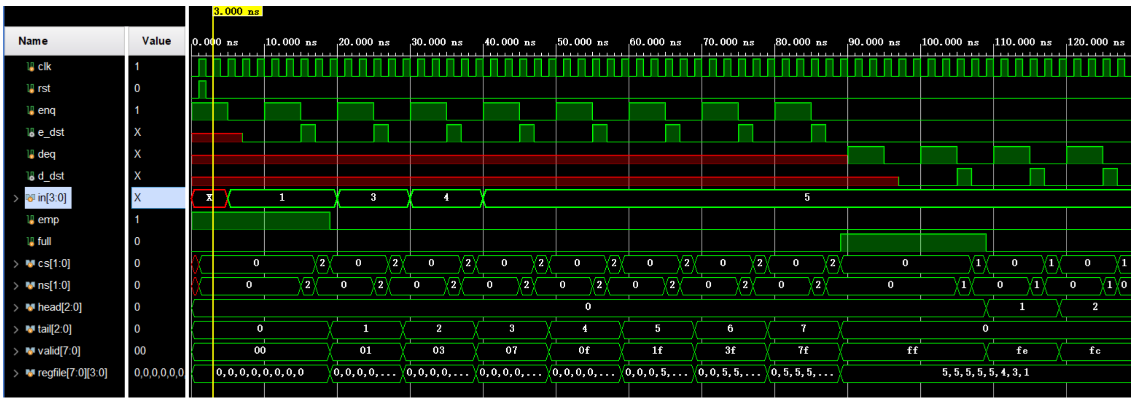
```
            begin
                an <= 0;
                seg <= 0;
            end
        else if(cnt <= 10'd100)
        begin
            if(valid[0])
            begin
                an <= 0;
                seg <= regfile[0];
            end
        end
        else if(cnt <= 10'd200)
        begin
            if(valid[1])
            begin
                an <= 1;
                seg <= regfile[1];
            end
        end
        else if(cnt <= 10'd300)
        begin
            if(valid[2])
            begin
                an <= 2;
                seg <= regfile[2];
            end
        end
        else if(cnt <= 10'd400)
        begin
            if(valid[3])
            begin
                an <= 3;
                seg <= regfile[3];
            end
        end
        else if(cnt <= 10'd500)
        begin
            if(valid[4])
            begin
                an <= 4;
                seg <= regfile[4];
            end
        end
        else if(cnt <= 10'd600)
        begin
            if(valid[5])
            begin
                an <= 5;
                seg <= regfile[5];
            end
        end
        else if(cnt <= 10'd700)
        begin
            if(valid[6])
            begin
                an <= 6;
                seg <= regfile[6];
```

```
            end
        end
        else
        begin
            if(valid[7])
            begin
                an <= 7;
                seg <= regfile[7];
            end
        end
    end
endmodule
```

- 仿真文件

```
module fifo_sim(
    );
    reg clk, rst;
    reg enq;
    reg [3:0] in;
    reg deq;
    wire [2:0] an;
    wire [3:0] seg;
    wire [7:0] led;
    FIFO fifo(clk, rst, enq, in, deq, an, seg, led);
    initial clk = 0;
    always #1 clk = ~clk;
    initial begin
    rst = 0;#1 rst = 1;#1 rst = 0;
    end
    initial
    begin
    enq = 1;#5 enq = 0;in = 1;#5 enq = 1;#5 enq = 0;#5 enq = 1;in = 3;#5 enq = 0;#5 enq = 1;in = 4;#5 enq = 0;#5 enq = 1;in = 5;#5 enq = 0;
    #5 enq = 1;in = 5;#5 enq = 0;#5 enq = 1;in = 5;#5 enq = 0;#5 enq = 1;in = 5;#5 enq = 0;#5 enq = 1;in = 5;#5 enq = 0;
    #5 deq = 1;#5 deq = 0;#5 deq = 1;#5 deq = 0;#5 deq = 1;#5 deq = 0;#5 deq = 1;#5 deq = 0;#5 $finish;
    end
endmodule
```

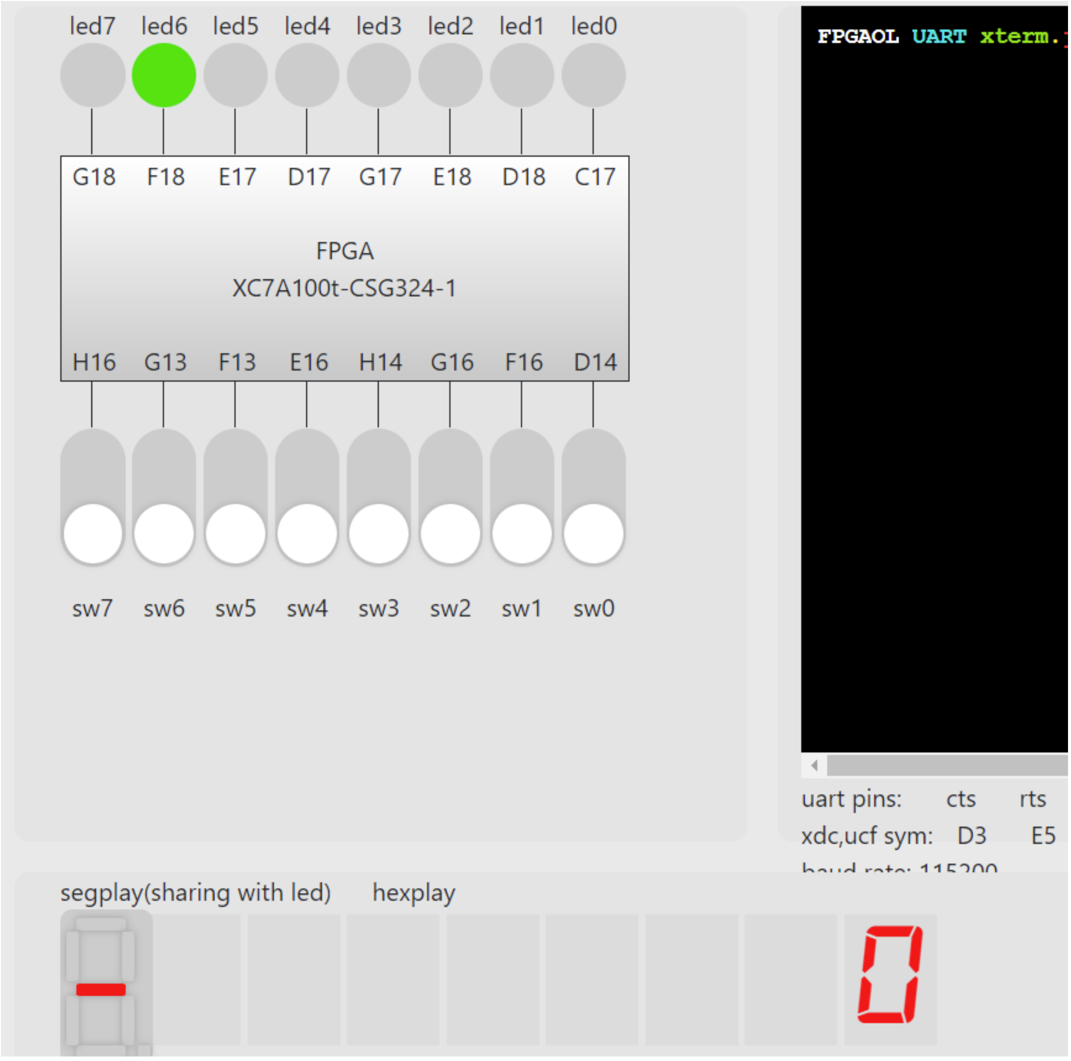
- 仿真结果



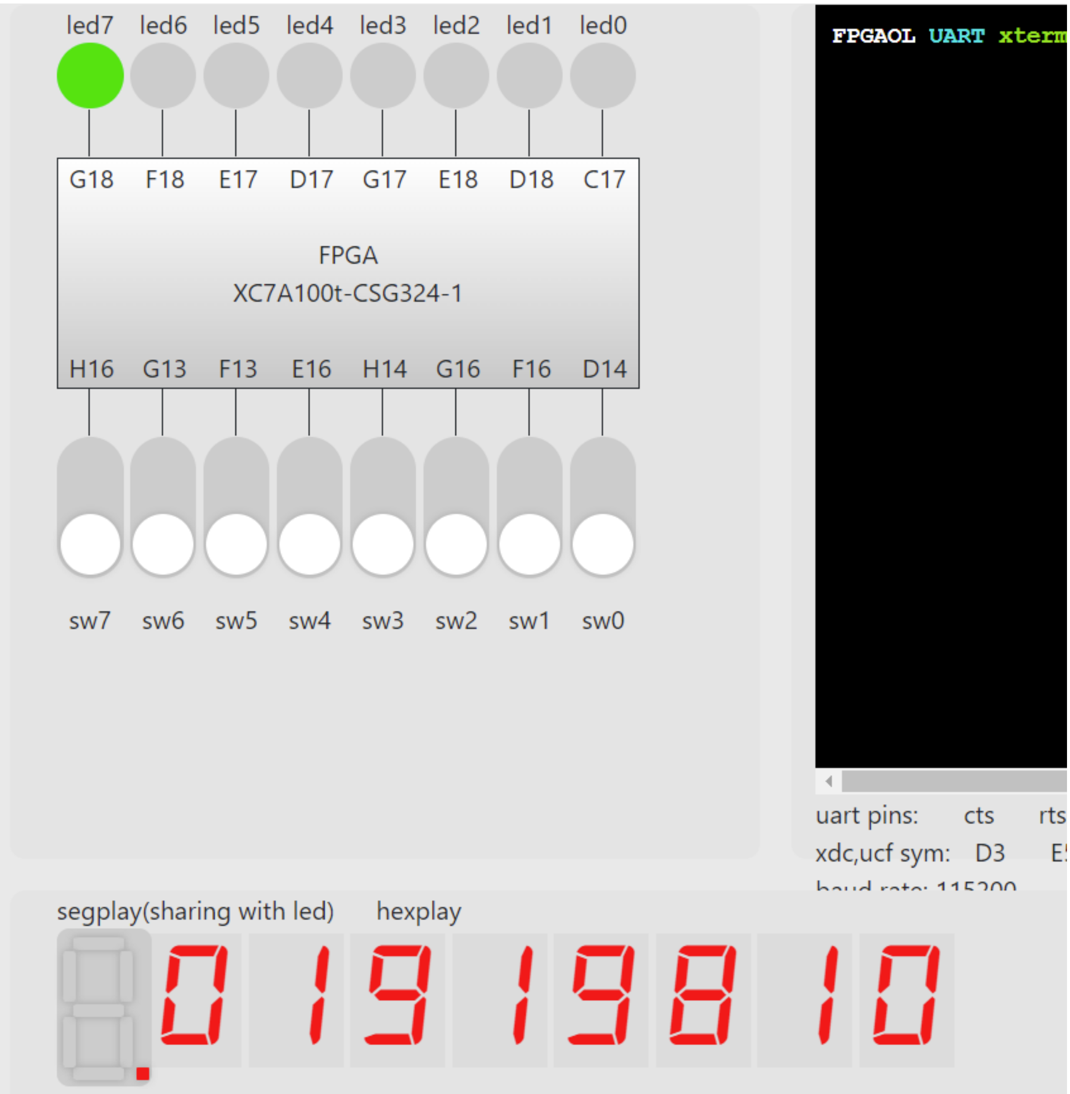
- 约束文件

```
# FPGAOL LED (signle-digit-SEGPLAY)

set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { led[7] }];


## FPGAOL SWITCH

set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { in[0] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { in[1] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { in[2] }];
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { in[3] }];
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports {  rst}];
set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports {  }];
set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { deq }];
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { enq }];

## FPGAOL HEXPLAY

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { seg[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { seg[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { seg[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { seg[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { an[0] }];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports { an[1] }];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports { an[2] }];
```

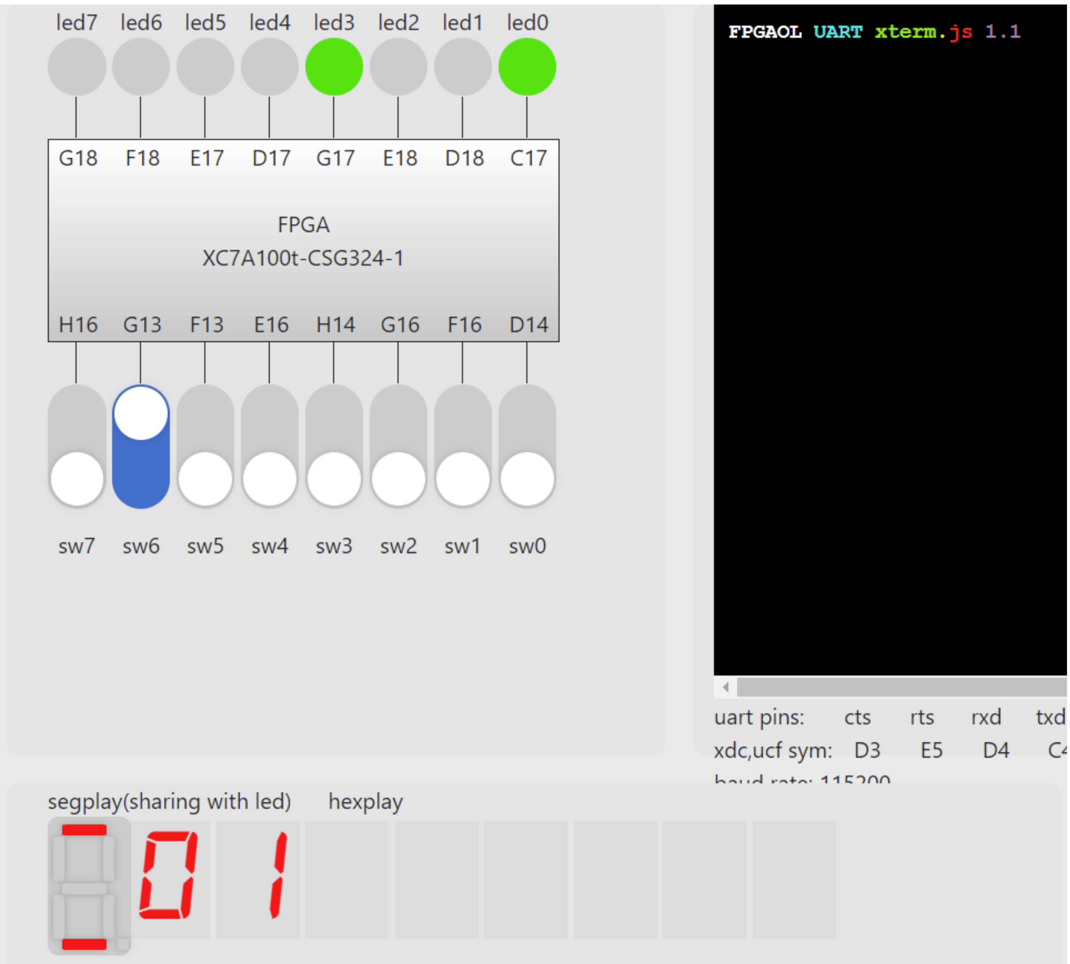
- 烧写结果
    - 初始状态

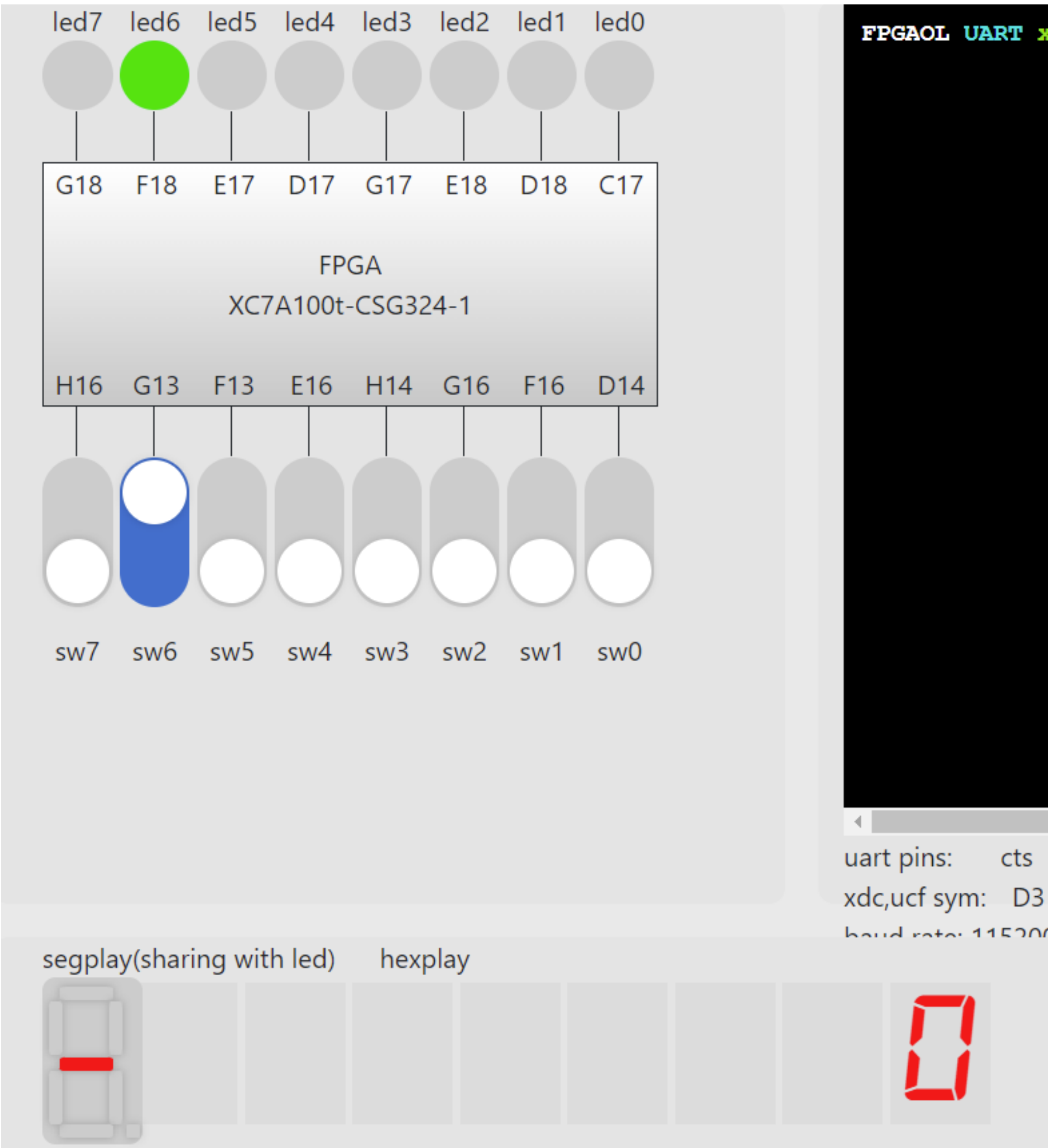

led7
led6
led5
led4
led3
led2
led1
led0
G18
F18
E17
D17
G17
E18
D18
C17
FPGA
XC7A100t-CSG324-1
H16
G13
F13
E16
H14
G16
F16
D14
sw7
sw6
sw5
sw4
sw3
sw2
sw1
sw0
FPGAOL UART xterm.
uart pins: cts rts
xdc,ucf sym: D3 E5
segplay(sharing with led)
hexplay

- 写入至队满


led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
FPGAOL UART xterm
uart pins: cts rts
xdc,ucf sym: D3
segplay(sharing with led) hexplay
0 19 1981 0

- 输出至队空


led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
FPGAOL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc,ucf sym: D3 E5 D4
segplay(sharing with led)
hexplay



【总结与思考】

- 实验总结
  - 事实上上学期数电实验已使用过IP核，但仅例化了分布式，较为简单。本次实验通过例化块式与分布式存储器了解二者的本质区别所在，并能够改变读写模式进行写入或输出；
  - 学会设计FIFO队列，其实本质还是复习数电实验（包括时分复用），难度中等（对没有基础而言），毕竟PPT其实已经给出了较为详细的说明。
- 实验建议
  - 如果实验文档能更详细一点就好了；
  - 要是实验文档能更详细一点就好了；
  - 真的实验文档能更详细一点就好了。