**计算机组成原理 实验报告**

**姓名：宋玮 学号：PB20151793 实验日期：2022.3.30**

**一、实验题目：**

Lab02 寄存器堆与存储器及其应用

**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,fpgaol

**四、实验过程及结果：**

如下所示：

* **寄存器堆**

**1.verilog设计代码**

注：m，width，sum为三个参数，方便后续例化不同大小的寄存器堆

**下面是8×4 位的寄存器堆**

module register\_file#(

parameter m = 3, WIDTH = 4

)(

input clk,

input we,

input [m-1:0] wa,

input [m-1:0] ra0,ra1,

input [WIDTH-1:0] wd,

output [WIDTH-1:0] rd0,rd1

);

parameter sum = 8'b1 << m;

reg [WIDTH-1:0] regfile [0:sum-1];

assign rd0 = regfile[ra0],

rd1 = regfile[ra1];

always @(posedge clk)

begin

if (we) regfile[wa] <= wd;

end

endmodule

**2.仿真文件**

module rf\_sim();

reg clk;

reg we;

reg [2:0] ra0,ra1;

wire [3:0] rd0,rd1;

reg [2:0] wa;

reg [3:0] wd;

integer i;

register\_file rf(.clk(clk),.we(we),.rd0(rd0),.rd1(rd1),.ra0(ra0),.ra1(ra1),.wa(wa),.wd(wd));

initial clk = 0;

always #5 clk = ~clk;

initial we = 0;

always #5 we = ~we;

initial

begin

wa = 0; wd = 0; ra0 = 0; ra1 = 0;

for(i=1;i<8;i=i+1)

begin

#10; wa = wa+1; wd = wd + 1;

end

for(i=0;i<7;i=i+1)

begin

#10; ra0 = i; ra1 = 7-i;

end

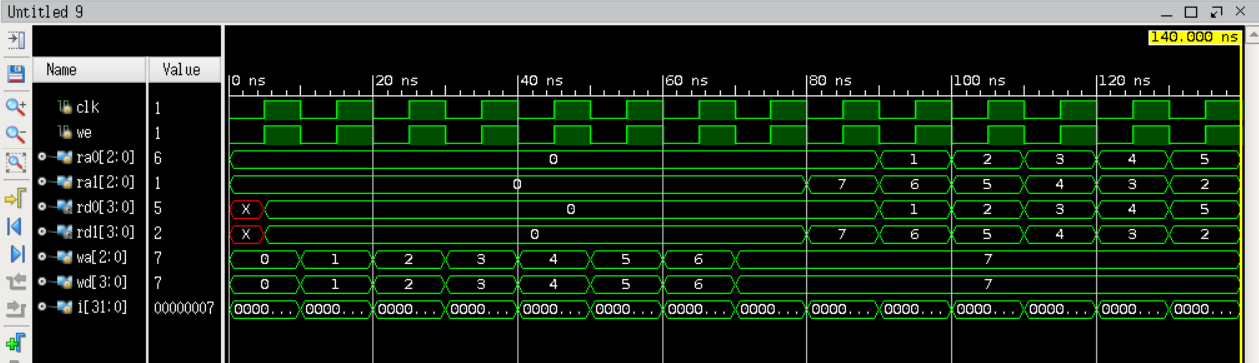
$finish;

end

endmodule

**3.仿真结果**

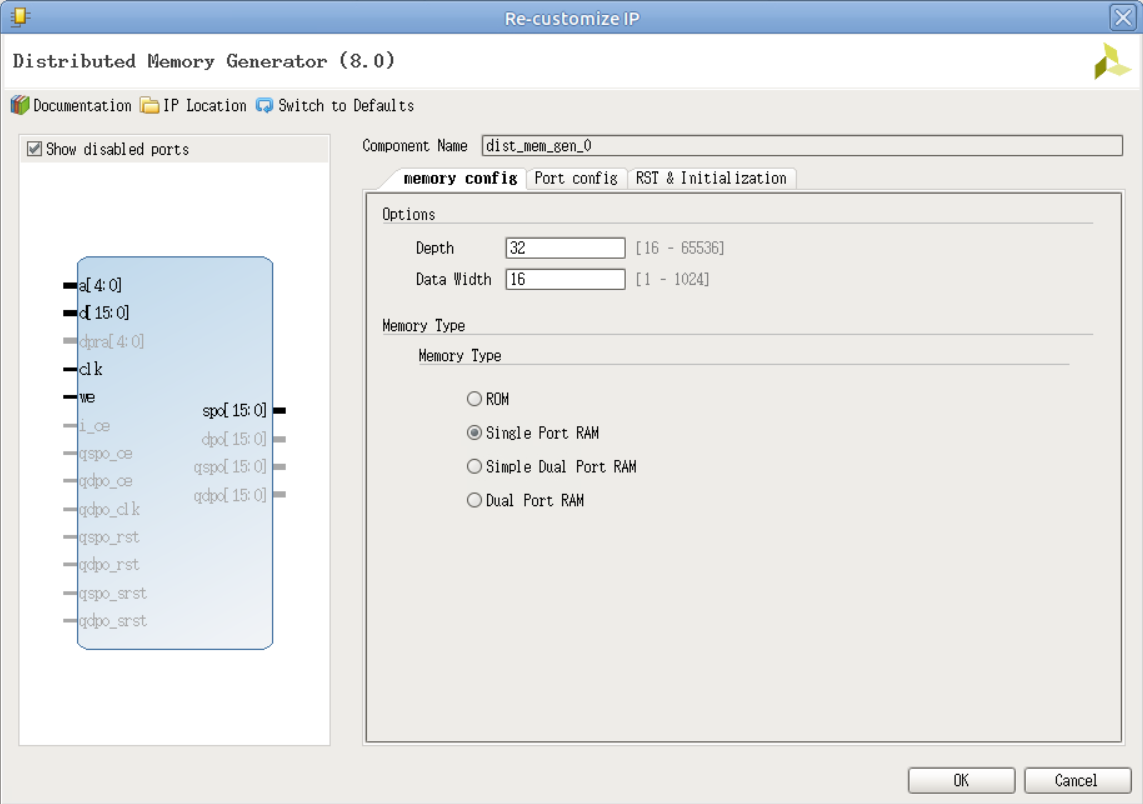
前面写入0~7，后面通过rd0，rd1输出0~7



* **存储器IP核**

**1.分布式存储器（Distributed Memory Generator）**

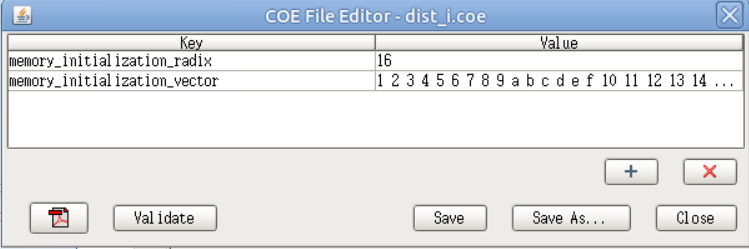
（1）在IP Catalog中生成分布式存储器ip核（dist\_mem\_gen\_0）



（2）载入coe文件：

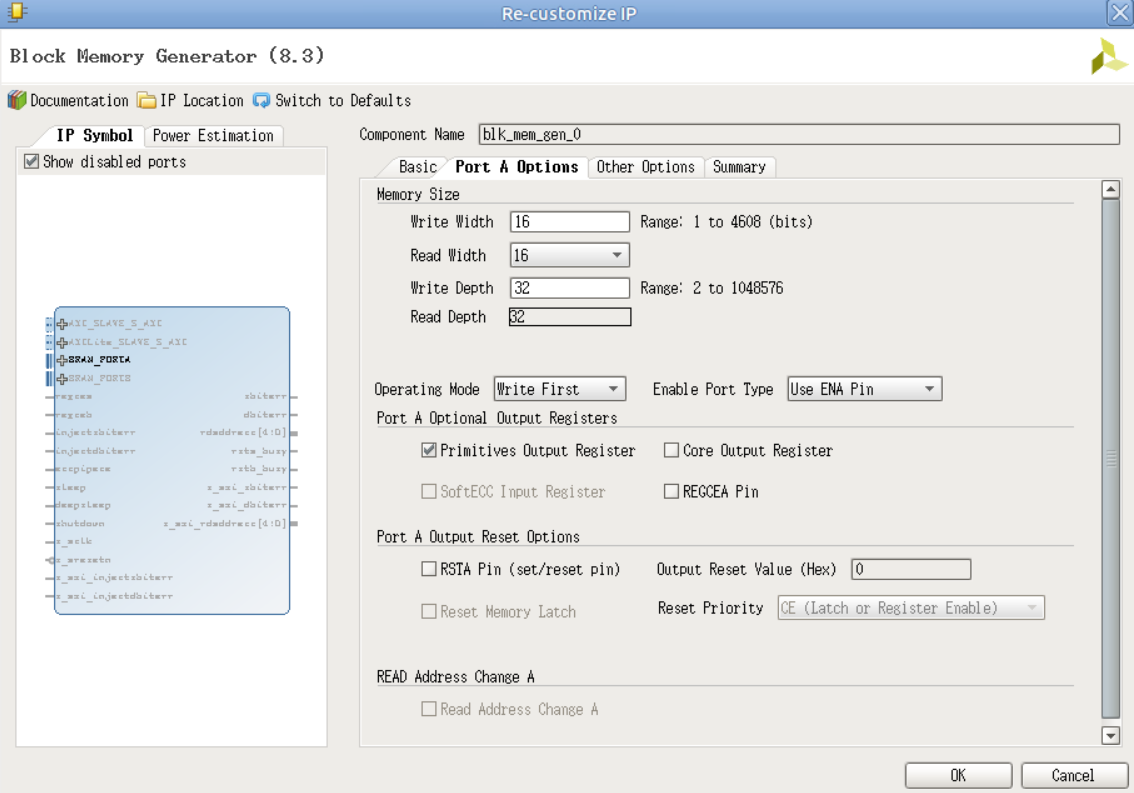
memory\_initialization\_radix = 16;

memory\_initialization\_vector = 1 2 3 4 5 6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20;



**2.块式存储器（Block Memory Generator）**

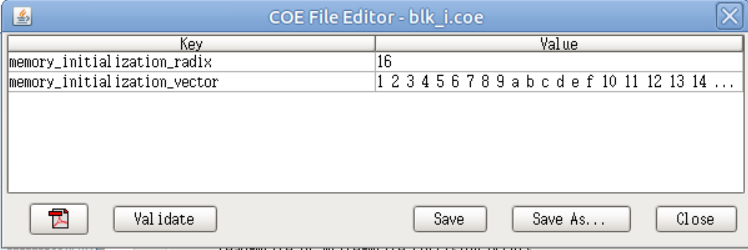
（1）在IP Catalog中生成块式存储器ip核（blk\_mem\_gen\_0）



（2）载入coe文件：

memory\_initialization\_radix = 16;

memory\_initialization\_vector = 1 2 3 4 5 6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20;



**3.仿真文件**

module dist();

reg clk,we,en;

reg [4:0] a;

reg [15:0] wd;

wire [15:0] distd, blkd;

dist\_mem\_gen\_0 distmem (

.a(a), // input wire [4 : 0] a

.d(wd), // input wire [15 : 0] d

.clk(clk), // input wire clk

.we(we), // input wire we

.spo(distd) // output wire [15 : 0] spo

);

blk\_mem\_gen\_0 blkmem (

.clka(clk), // input wire clka

.ena(en), // input wire ena

.wea(we), // input wire [0 : 0] wea

.addra(a), // input wire [4 : 0] addra

.dina(wd), // input wire [15 : 0] dina

.douta(blkd) // output wire [15 : 0] douta

);

initial clk = 0;

always #5 clk = ~clk;

initial

begin

en = 1; a = 1; wd = 0; we = 1;

#6 wd = 4;

#6 a = 2;

#8 wd = 5;

#10 wd = 6;

#1 a = 3;

#10 $finish;

end

endmodule

**4.仿真结果**

**注意到：**

**分布式存储器为同步写，异步读；**

**块式存储器为同步写，并且在下一个周期读出（原因：块式存储器的内部结构，写入的数据在读出时要经过D触发器，导致慢一个周期，见图2）**

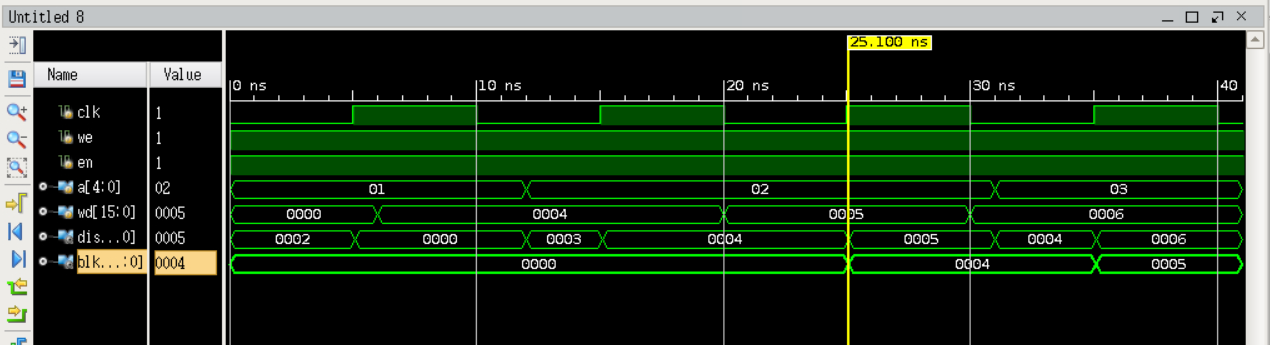


图1

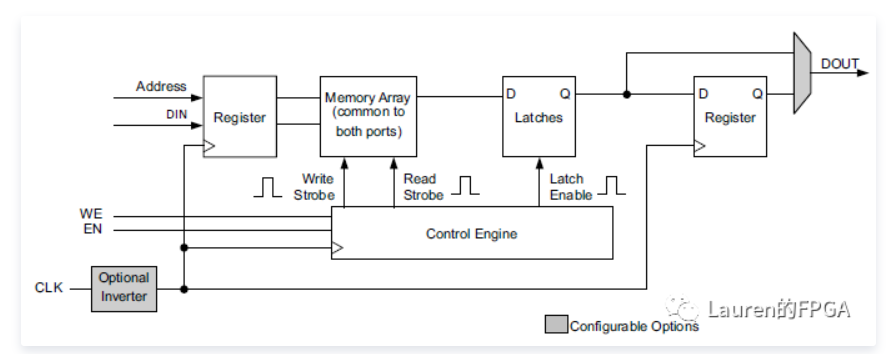
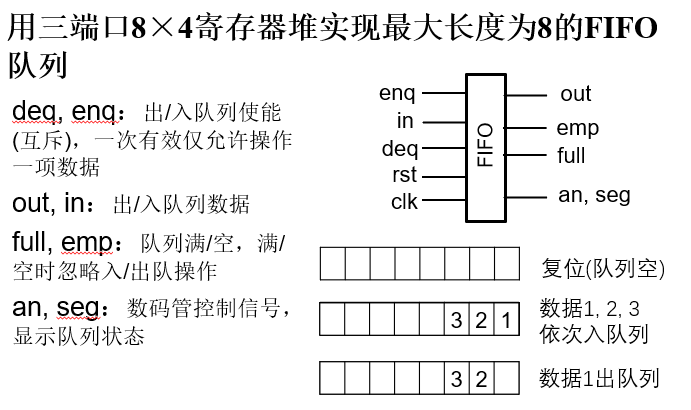


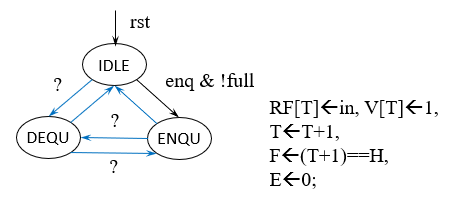
图2

* **FIFO队列**



**1.设计文件**

* 根据要求，设置了四个状态——rrst（复位状态），idle（闲置状态），enqu（入队状态），dequ（出队状态）；



* 也设置了队头指针（head）和队尾指针（tail），分别指向队头和队尾元素位置；
* 队列最大长度为8，含有元素数量由num表示，有效位由valid[]表示；

具体verilog代码如下：

module fifo(

input clk, rst,

input enq,

input [3:0] in,

input deq,

output [3:0] out,

output [2:0] an,

output [3:0] seg,

output full,

output empty

);

reg [2:0] head,tail;

reg we;

wire [2:0]ra0;

wire [3:0]rd0;

wire [3:0]rd1;

reg button\_r1,button\_r2,button\_r3,button\_r4;

wire enq\_edge,deq\_edge;

reg [2:0] led = 0;

wire t;

reg [31:0] cnt = 32'b0;

reg dequeue=0; //out

reg [4:0]num;

reg [7:0]valid;

reg [1:0]ns,cs;

parameter idle = 2'b00, enqu = 2'b01, dequ = 2'b10, rrst = 2'b11;

//去抖动

always@(posedge clk)

button\_r1 <= enq;

always@(posedge clk)

button\_r2 <= button\_r1;

assign enq\_edge = button\_r1 & (~button\_r2);

always@(posedge clk)

button\_r3 <= deq;

always@(posedge clk)

button\_r4 <= button\_r3;

assign deq\_edge = button\_r3 & (~button\_r4);

register\_file r\_f(.clk(clk),.we(we),.ra0(ra0),.rd0(rd0),.ra1(an),.rd1(rd1),.wa(tail),.wd(in));

assign empty = (num == 0) ? 1 : 0;

assign full = (num == 8) ? 1 : 0;

//描述ns

always@(\*)

begin

if(rst) ns = rrst;

else

begin

case(cs)

rrst:

begin

if(enq\_edge && !full) ns = enqu;

else if(deq\_edge && !empty) ns = dequ;

else ns = rrst;

end

default:

begin

if(enq\_edge && !full) ns = enqu;

else if(deq\_edge && !empty) ns = dequ;

else ns = idle;

end

endcase

end

end

//描述cs

always@(posedge clk)

begin

cs <= ns;

end

always@(posedge clk)

begin

case(ns)

enqu:

begin

num <= num + 1;

tail <= tail + 3'b1;

we <= 1;

valid[tail+3'b1] <= 1'b1;

end

dequ:

begin

num <= num - 1;

head <= head + 3'b1;

we <= 0;

dequeue <= 1;

valid[head] <= 1'b0;

end

rrst:

begin

head <= 0;

tail <= 3'b111;

num <= 0;

we <= 0;

dequeue <= 0;

valid <= 8'b0;

end

idle:

begin

we <= 0;

end

endcase

end

assign ra0 = head ? head-1 : 3'b111;

assign out = dequeue ? rd0 : 0;

//时钟分时复用

always@(posedge clk)

begin

if(cnt >= 400000)

cnt <= 32'b0;

else

cnt <= cnt + 32'b1;

end

assign t = (cnt == 32'b1);

always@(posedge clk)

begin

if(t)

begin

if( valid[led+1] == 1'b0 ) led <= head;

else led <= led + 1;

end

end

//描述输出

assign an = led;

assign seg = empty ? 15 : rd1; //ra1:an

endmodule

**2.仿真文件**

module fifo\_sim();

reg clk, rst;

reg enq,deq;

reg [3:0] in;

wire [3:0] out,seg;

wire [2:0] an;

wire full,empty;

integer i;

initial clk = 0;

always #5 clk = ~clk;

fifo fi\_fo(.clk(clk),.rst(rst),.enq(enq),.deq(deq),.in(in),.out(out),.seg(seg),.an(an),.full(full),.empty(empty));

initial

begin

rst = 1; enq = 0; deq = 0; in = 0;

#12 rst = 0;

for (i=0; i<24; i=i+1)

begin

#12

enq = ~enq;

in = in + 2;

end

for(i=0; i<24; i=i+1)

begin

#12

deq = ~deq ;

end

#16 $finish;

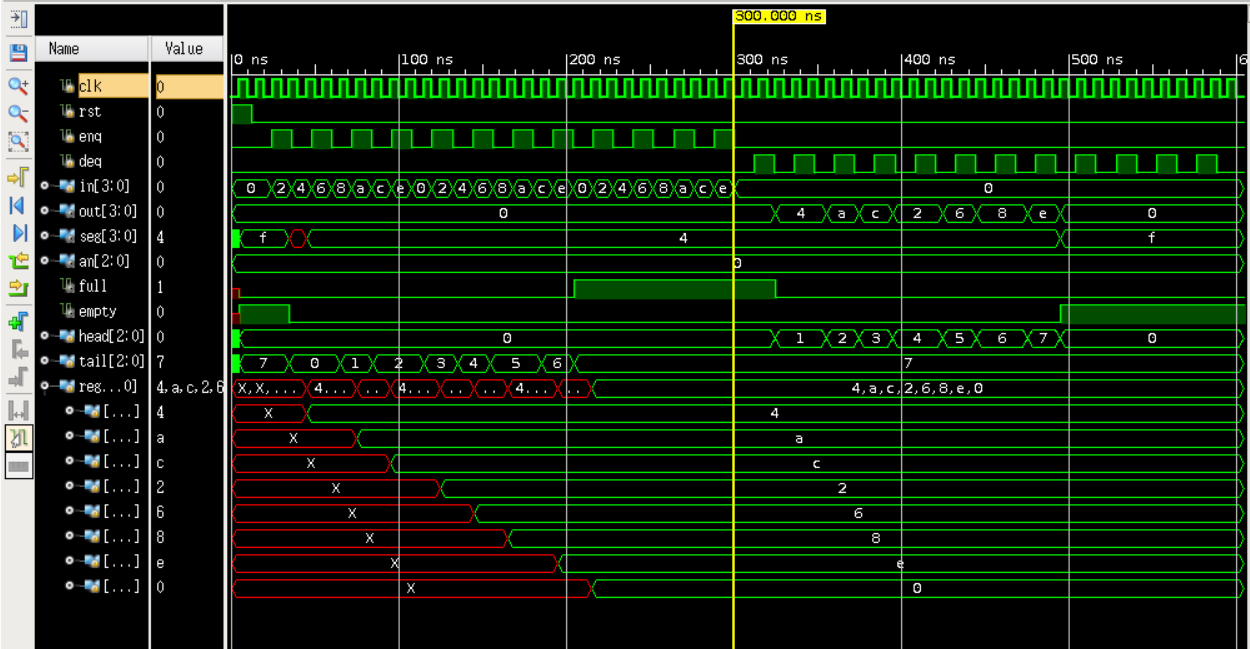
end

endmodule

**3.仿真结果**

首先，4,a,c,2,6,8,e,0入队，之后队满，无法再入队，并且full变为高电平；

之后，4,a,c,2,6,8,e,0依次出队，直至队空，显示f（false），并且empty变为高电平



**4.约束文件**

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33} [get\_ports { clk }];

set\_property -dict { PACKAGE\_PIN F13 IOSTANDARD LVCMOS33} [get\_ports { rst }];

set\_property -dict { PACKAGE\_PIN H16 IOSTANDARD LVCMOS33} [get\_ports { enq }];

set\_property -dict { PACKAGE\_PIN G13 IOSTANDARD LVCMOS33} [get\_ports { deq }];

set\_property -dict { PACKAGE\_PIN C17 IOSTANDARD LVCMOS33} [get\_ports { out[0] }];

set\_property -dict { PACKAGE\_PIN D18 IOSTANDARD LVCMOS33} [get\_ports { out[1] }];

set\_property -dict { PACKAGE\_PIN E18 IOSTANDARD LVCMOS33} [get\_ports { out[2] }];

set\_property -dict { PACKAGE\_PIN G17 IOSTANDARD LVCMOS33} [get\_ports { out[3] }];

set\_property -dict { PACKAGE\_PIN G18 IOSTANDARD LVCMOS33} [get\_ports { full }];

set\_property -dict { PACKAGE\_PIN F18 IOSTANDARD LVCMOS33} [get\_ports { empty }];

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 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] }];

set\_property -dict { PACKAGE\_PIN A14 IOSTANDARD LVCMOS33} [get\_ports { seg[0] }];

set\_property -dict { PACKAGE\_PIN A13 IOSTANDARD LVCMOS33} [get\_ports { seg[1] }];

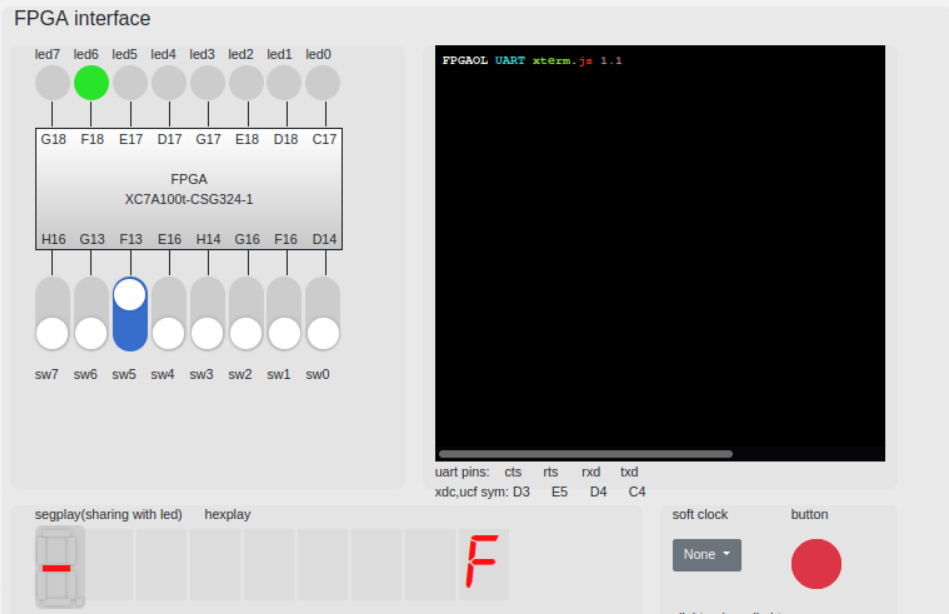
set\_property -dict { PACKAGE\_PIN A16 IOSTANDARD LVCMOS33} [get\_ports { seg[2] }];

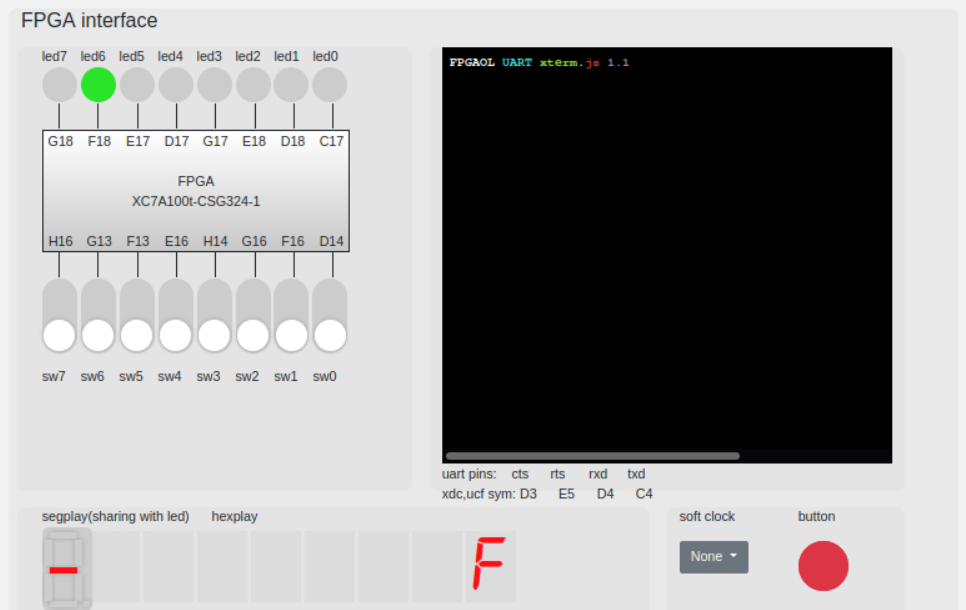
set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVCMOS33} [get\_ports { seg[3] }];

**5.fpga ol测试bit文件**

**（1）首先，复位，sw5为复位键**

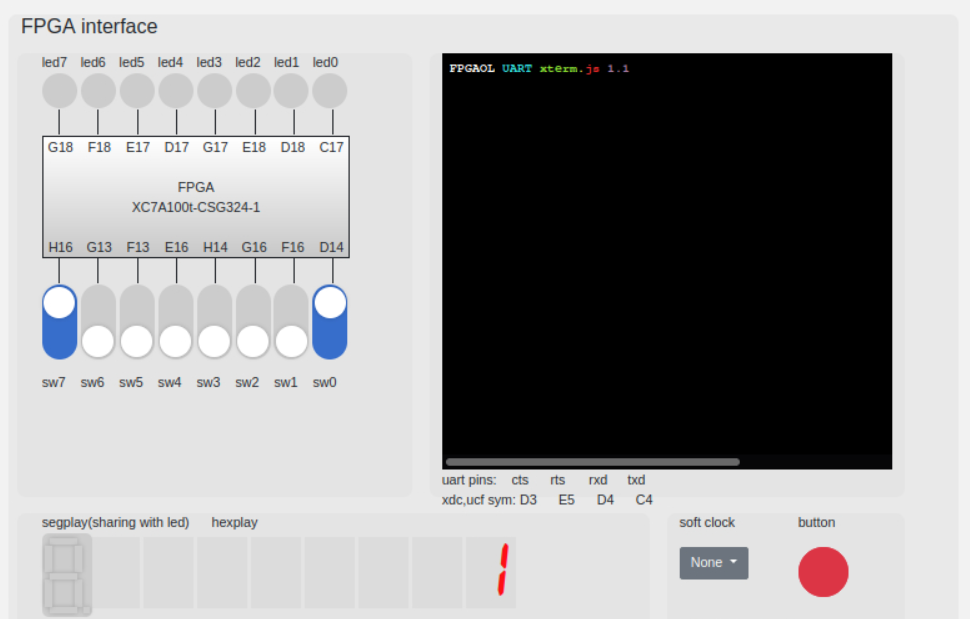
队空，led6（empty）亮起，并且数码管显示f（false），表示此时队内没有元素

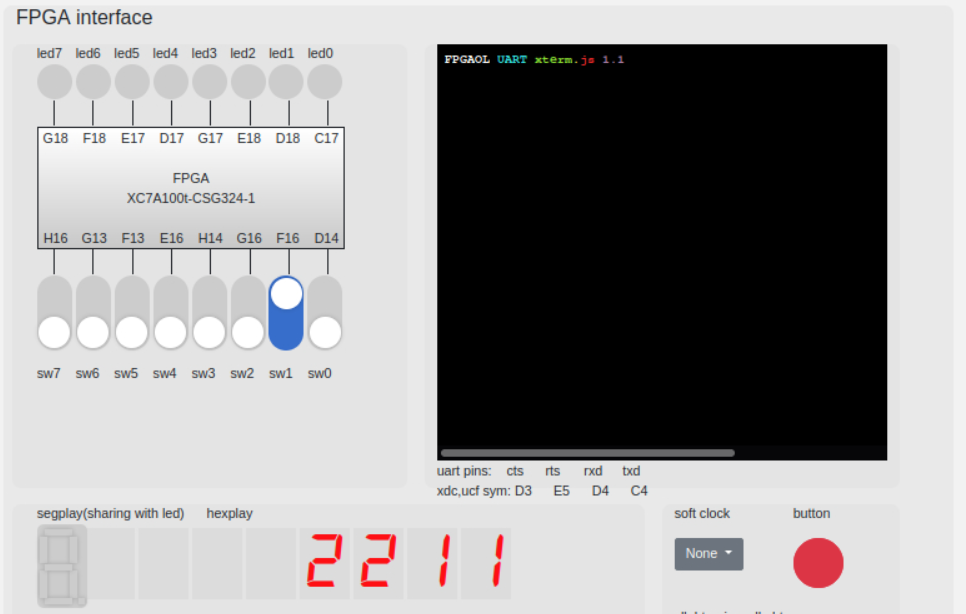


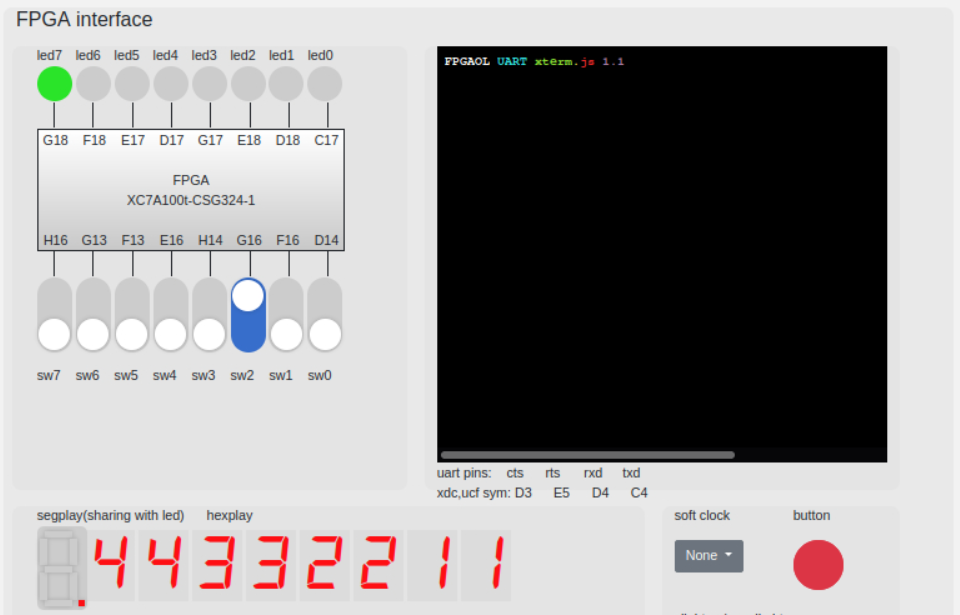


**（2）入队1,1,2,2,3,3,4,4，sw7位入队按钮**

最后队满，led7（full）亮起







**（3）1,1,2,2,3,3,4,4依次出队，sw6为出队按钮**

最后led6（empty）亮起，并且数码管显示f（false），表示此时队内没有元素

