中国科学技术大学计算机学院
《计算机组成原理》报告

实验题目：寄存器堆与储存器及其应用

学生姓名：刘恒远

学生学号：PB20111642

完成日期：2022.3.31

计算机实验教学中心制

【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

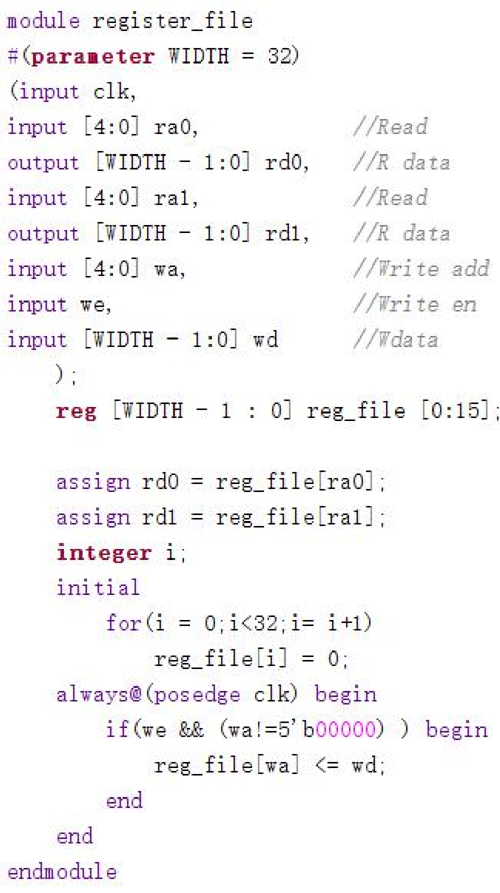
【实验环境】

Vivado，fpgaol.ustc.edu.cn（FPGAOL）

【实验过程】

题目一：

行为参数描述化寄存器堆，设计文件如图所示：

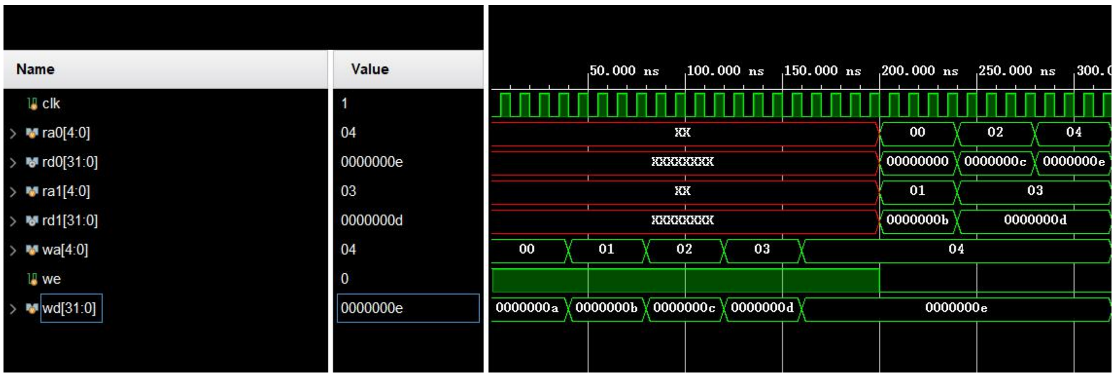


```verilog
module register_file
#(parameter WIDTH = 32)
(input clk,
input [4:0] ra0,            //Read
output [WIDTH - 1:0] rd0,   //R data
input [4:0] ra1,            //Read
output [WIDTH - 1:0] rd1,   //R data
input [4:0] wa,             //Write add
input we,                   //Write en
input [WIDTH - 1:0] wd      //Wdata
    );
    reg [WIDTH - 1 : 0] reg_file [0:15];

    assign rd0 = reg_file[ra0];
    assign rd1 = reg_file[ra1];
    integer i;
    initial
        for(i = 0;i<32;i= i+1)
            reg_file[i] = 0;
    always@(posedge clk) begin
        if(we && (wa!=5'b00000) ) begin
            reg_file[wa] <= wd;
        end
    end
endmodule
```

仿真文件如图所示：

```
module sim1();
reg clk;
reg [4:0] ra0;
wire [31:0] rd0;
reg [4:0] ra1;
wire [31:0] rd1;
reg [4:0] wa;
reg we;
reg [31:0] wd;
register_file re(.clk(clk),.ra0(ra0),.rd0(rd0),.ra1(ra1),.rd1(rd1),.wa(wa),.we(we),.wd(wd));
initial clk = 0;
always #5 clk = ~clk;
initial begin
    we = 1; wd = 10; wa = 0; #40;
    we = 1; wd = 11; wa = 1; #40;
    we = 1; wd = 12; wa = 2; #40;
    we = 1; wd = 13; wa = 3; #40;
    we = 1; wd = 14; wa = 4; #40;
    we = 0; ra0= 0; ra1 = 1; #40;
    we = 0; ra0 = 2; ra1 = 3; #40;
    we = 0; ra0 = 4; #40;
    $stop;
end
endmodule
```

仿真图像如下：

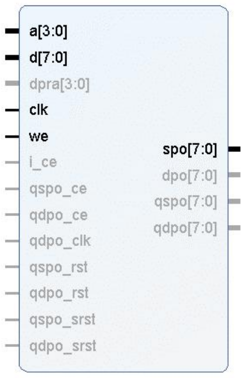


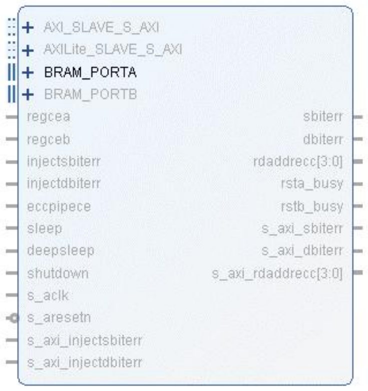
Name	Value
clk	1
ra0[4:0]	04
rd0[31:0]	0000000e
ra1[4:0]	03
rd1[31:0]	0000000d
wa[4:0]	04
we	0
wd[31:0]	0000000e

50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.0
XX 00 02 04
XXXXXXXX 00000000 0000000c 0000000e
XX 01 03
XXXXXXXX 0000000b 0000000d
00 01 02 03 04
0000000a 0000000b 0000000c 0000000d 0000000e

题目二：

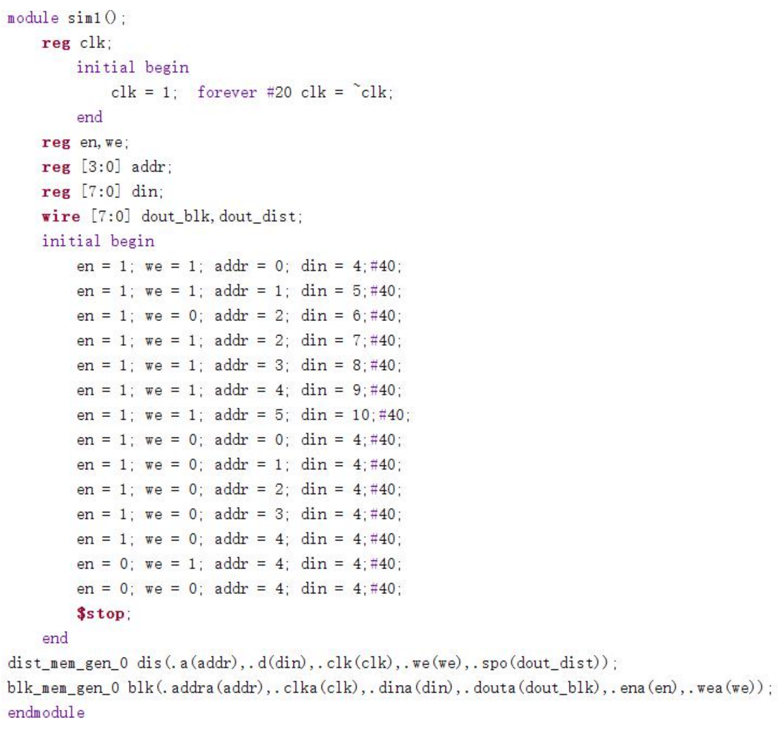
IP 例化分布式和块式 16x8 位单端口 RAM，例化模块如图所示：





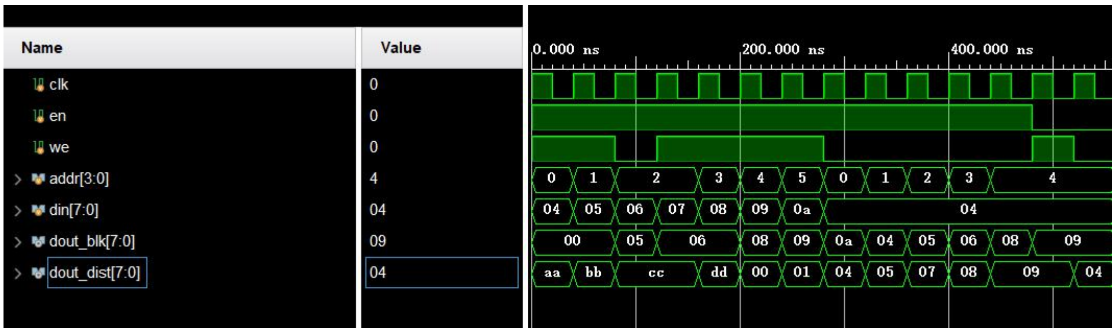
（分布式）　　　　　　　　　　　　　（块式）

仿真文件如图所示：



```
module sim1();
    reg clk;
        initial begin
            clk = 1;  forever #20 clk = ~clk;
        end
    reg en,we;
    reg [3:0] addr;
    reg [7:0] din;
    wire [7:0] dout_blk,dout_dist;
    initial begin
        en = 1; we = 1; addr = 0; din = 4;#40;
        en = 1; we = 1; addr = 1; din = 5;#40;
        en = 1; we = 0; addr = 2; din = 6;#40;
        en = 1; we = 1; addr = 2; din = 7;#40;
        en = 1; we = 1; addr = 3; din = 8;#40;
        en = 1; we = 1; addr = 4; din = 9;#40;
        en = 1; we = 1; addr = 5; din = 10;#40;
        en = 1; we = 0; addr = 0; din = 4;#40;
        en = 1; we = 0; addr = 1; din = 4;#40;
        en = 1; we = 0; addr = 2; din = 4;#40;
        en = 1; we = 0; addr = 3; din = 4;#40;
        en = 1; we = 0; addr = 4; din = 4;#40;
        en = 0; we = 1; addr = 4; din = 4;#40;
        en = 0; we = 0; addr = 4; din = 4;#40;
        $stop;
    end
dist_mem_gen_0 dis(.a(addr),.d(din),.clk(clk),.we(we),.spo(dout_dist));
blk_mem_gen_0 blk(.addra(addr),.clka(clk),.dina(din),.douta(dout_blk),.ena(en),.wea(we));
endmodule
```

仿真图像如下：

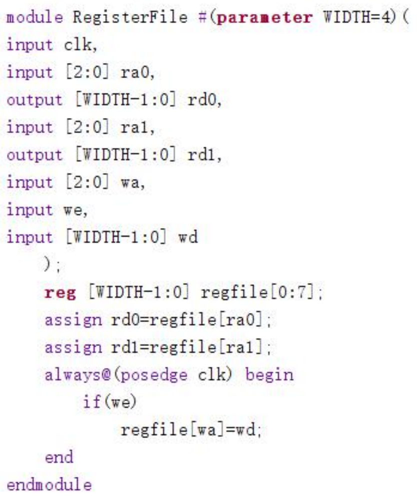


```
Name            Value
clk             0
en              0
we              0
addr[3:0]       4
din[7:0]        04
dout_blk[7:0]   09
dout_dist[7:0]  04
```

分布式 RAM 的写操作是同步的，读操作是异步的；
块式 RAM 的写操作是同步的，读操作是同步的。

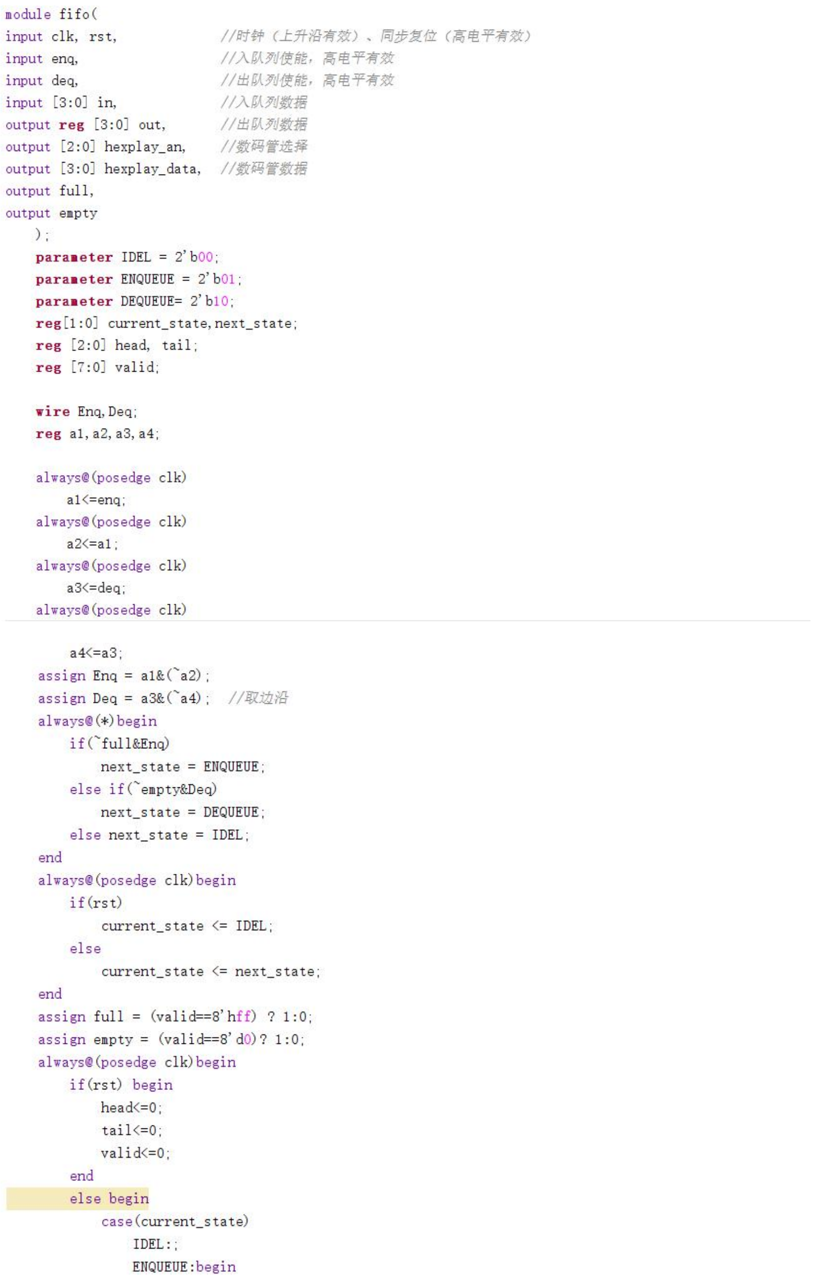
题目三：

例化一个寄存器堆，设计文件如图所示：



```
module RegisterFile #(parameter WIDTH=4)(
input clk,
input [2:0] ra0,
output [WIDTH-1:0] rd0,
input [2:0] ra1,
output [WIDTH-1:0] rd1,
input [2:0] wa,
input we,
input [WIDTH-1:0] wd
    );
    reg [WIDTH-1:0] regfile[0:7];
    assign rd0=regfile[ra0];
    assign rd1=regfile[ra1];
    always@(posedge clk) begin
        if(we)
            regfile[wa]=wd;
    end
endmodule
```

设计 FIFO，设计文件如图所示：

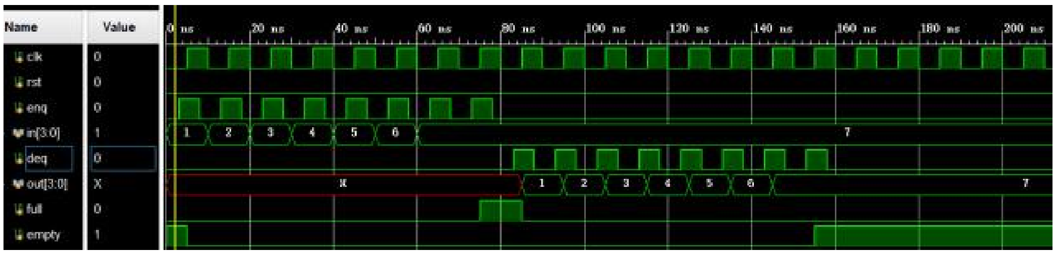


```
module fifo(
input clk, rst,              //时钟（上升沿有效）、同步复位（高电平有效）
input enq,                   //入队列使能，高电平有效
input deq,                   //出队列使能，高电平有效
input [3:0] in,              //入队列数据
output reg [3:0] out,        //出队列数据
output [2:0] hexplay_an,     //数码管选择
output [3:0] hexplay_data,   //数码管数据
output full,
output empty
    );
    parameter IDEL = 2'b00;
    parameter ENQUEUE = 2'b01;
    parameter DEQUEUE= 2'b10;
    reg[1:0] current_state,next_state;
    reg [2:0] head, tail;
    reg [7:0] valid;

    wire Enq,Deq;
    reg a1,a2,a3,a4;

    always@(posedge clk)
        a1<=enq;
    always@(posedge clk)
        a2<=a1;
    always@(posedge clk)
        a3<=deq;
    always@(posedge clk)

        a4<=a3;
    assign Enq = a1&(~a2);
    assign Deq = a3&(~a4);  //取边沿
    always@(*)begin
        if(~full&Enq)
            next_state = ENQUEUE;
        else if(~empty&Deq)
            next_state = DEQUEUE;
        else next_state = IDEL;
    end
    always@(posedge clk)begin
        if(rst)
            current_state <= IDEL;
        else
            current_state <= next_state;
    end
    assign full = (valid==8'hff) ? 1:0;
    assign empty = (valid==8'd0)? 1:0;
    always@(posedge clk)begin
        if(rst) begin
            head<=0;
            tail<=0;
            valid<=0;
        end
        else begin
            case(current_state)
                IDEL:;
                ENQUEUE:begin
```

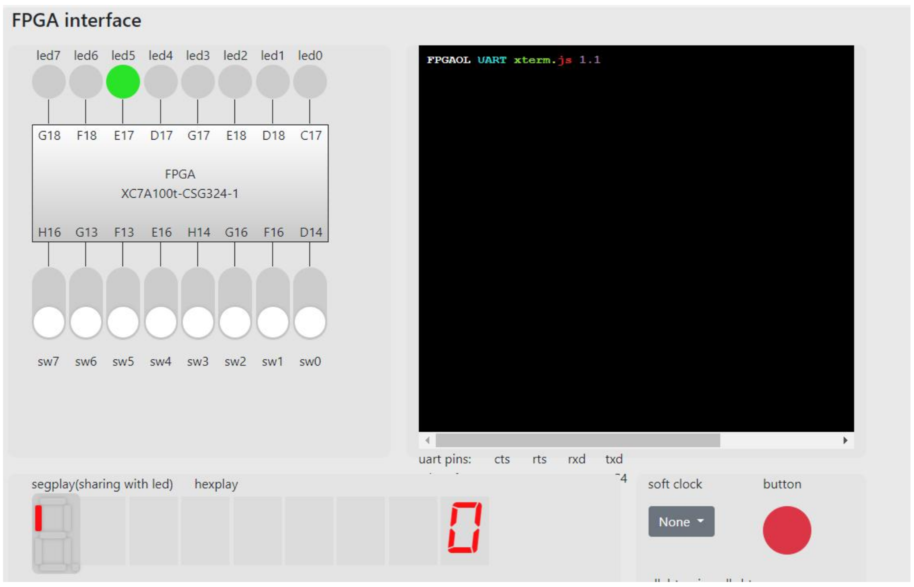
```
                    tail <= tail + 1;
                    valid [tail] <= 1;
                end
                DEQUEUE:begin
                    head <= head + 1;
                    valid[head] <= 0;
                end
            endcase;
        end
    end
    reg [4:0]cnt;
    reg [2:0] next_an,current_an;
    wire [3:0] seg;
    always@(posedge clk)
        cnt<=cnt+1;
    always@(posedge clk) begin
        if(cnt == 0)
            next_an<=next_an+1;
    end
    always@(posedge clk) begin
        if(rst)
            current_an<=0;
        else if(valid[next_an]==1)
            current_an<=next_an;
        else
            current_an<=current_an;
    end
    wire WE;
    assign WE = Enq&(~full);
```

```
    wire [3:0] Out;
    always@(posedge clk) begin
        if(rst)
            out<=0;
        else if(~empty&Deq)
                out<=Out;
            else
                out<=out;
    end
    RegisterFile regfile(.clk(clk),.ra0(head),.rd0(Out),.ra1(current_an),.rd1(seg),.wa(tail),.we(WE),.wd(in));
    assign hexplay_an = empty?head:current_an;
    assign hexplay_data = empty?0:seg;
endmodule
```

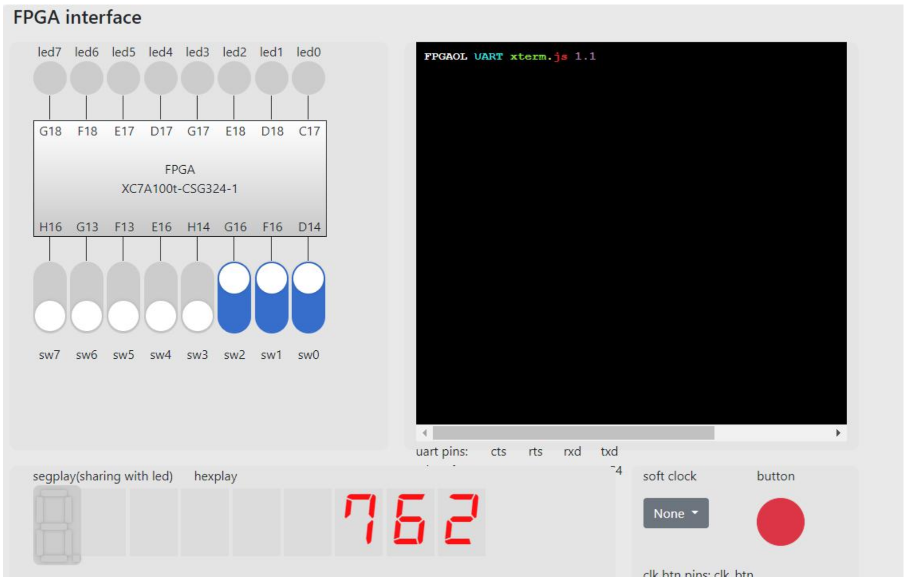
仿真图像如下：



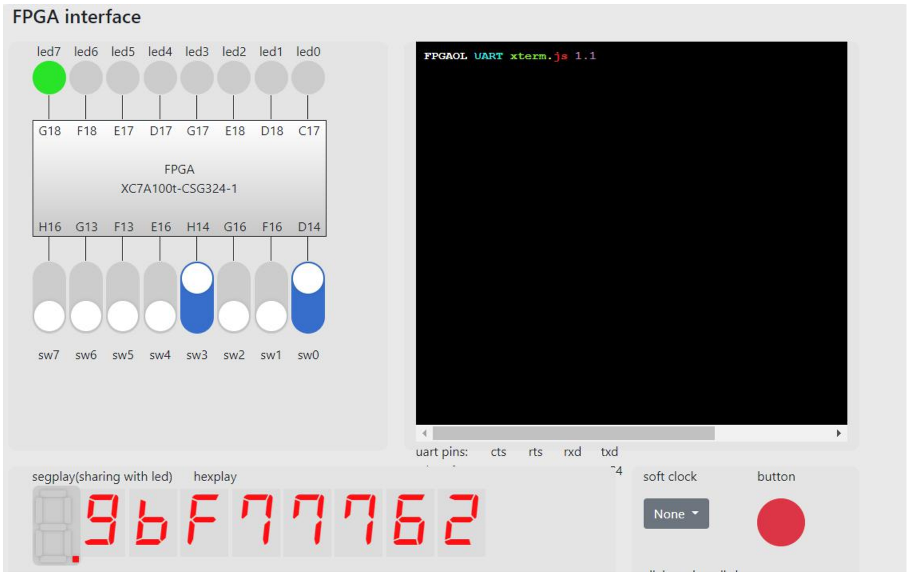
烧写至 FPGA，结果如下：


FPGA interface
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
uart pins: cts rts rxd txd
segplay(sharing with led) hexplay
0
soft clock
None
button

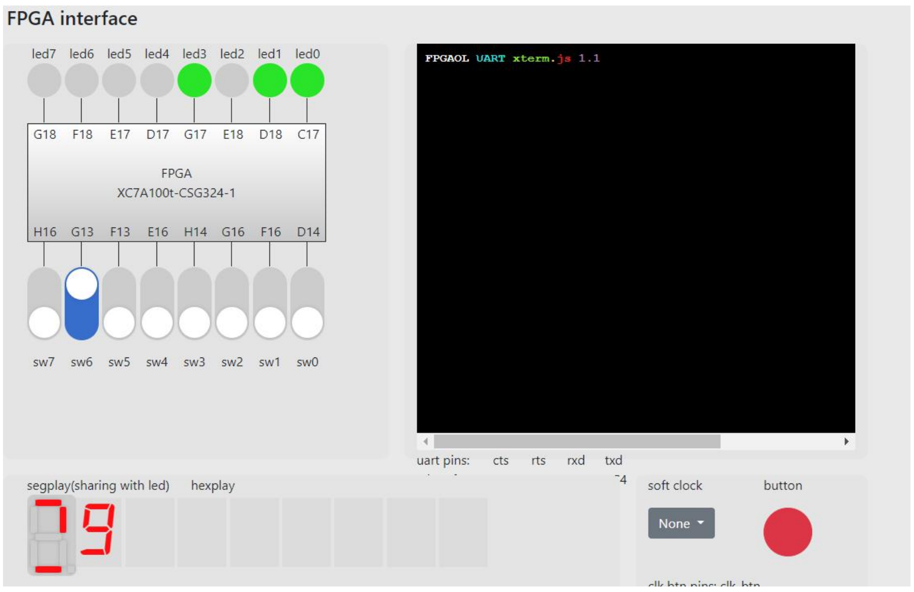
```
FPGAOL UART xterm.js 1.1
```


FPGA interface
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
uart pins: cts rts rxd txd
segplay(sharing with led) hexplay
762
soft clock
None
button
clk btn pins: clk btn

```
FPGAOL UART xterm.js 1.1
```


FPGA interface
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
uart pins: cts rts rxd txd
segplay(sharing with led) hexplay
9bF77762
soft clock
None
button

```
FPGAOL UART xterm.js 1.1
```


FPGA interface
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
uart pins: cts rts rxd txd
segplay(sharing with led) hexplay
soft clock
None
button

```
FPGAOL UART xterm.js 1.1
```

功能正常

【总结与反思】

PPT 不详细，建议详细一些