Lab 2 report

PB22051022王嘉宁

实验目的与内容

> 通过 Verilog 硬件描述语言实现一个可用的单周期 CPU。复习 Verilog 语法，实现 CPU 的功能部件。

逻辑设计

> 任务 1：寄存器堆设计

```
module Q1(
    input                       [ 0 : 0]        clk,

    input                       [ 4 : 0]        rf_ra0,
    input                       [ 4 : 0]        rf_ra1,
    input                       [ 4 : 0]        rf_wa,
    input                       [ 0 : 0]        rf_we,
    input                       [31 : 0]        rf_wd,

    output          reg         [31 : 0]        rf_rd0,
    output          reg         [31 : 0]        rf_rd1
);

    reg [31 : 0] reg_file [0 : 31];

    // 用于初始化寄存器
    integer i;
    initial begin
        for (i = 0; i < 32; i = i + 1)
            reg_file[i] = 0;
    end

    always @(posedge clk) begin
        if(rf_we && rf_wa != 5'd0) begin
            reg_file[rf_wa] <= rf_wd;
        end
        else    reg_file[rf_wa] <= 32'd0;
        end
    always @(*) begin
        rf_rd0 = reg_file[rf_ra0];
        rf_rd1 = reg_file[rf_ra1];
    end
endmodule
```

> 任务 2：ALU 设计

比较器

```
module Comp_4(
    input [ 31 : 0] a, b,
    output [ 0 : 0] ul,
    output [ 0 : 0] sl
);

wire [ 31 : 0]A;
wire x;
assign A = a - b;
assign sl = ((a[31] ~^ b[31]) & A[31]) | ((a[31] ^ b[31]) & a[31]);
assign ul = (a < b) ? 1'b1 : 1'b0;

endmodule
```

ALU

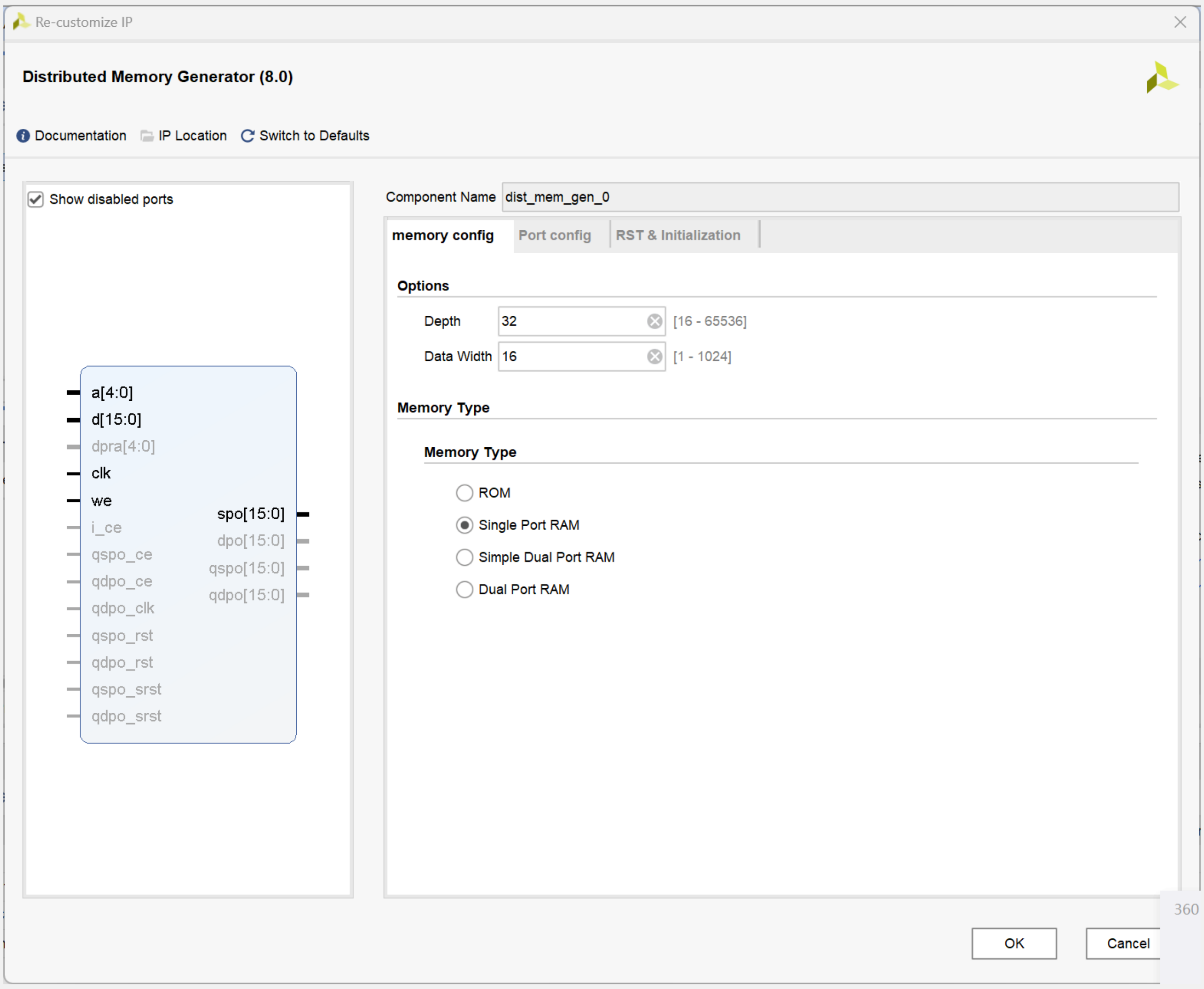
```
`define     ADD     5'b00000
`define     SUB     5'B00010
`define     SLT     5'B00100
`define     SLTU    5'B00101
`define     AND     5'B01001
`define     OR      5'B01010
`define     XOR     5'B01011
`define     SLL     5'B01110
`define     SRL     5'B01111
`define     SRA     5'B10000
`define     SRC0    5'B10001
`define     SRC1    5'B10010

module ALU(
    input   [31 : 0] alu_src0,
    input   [31 : 0] alu_src1,
    input   [ 4 : 0] alu_op,
    output  reg [31 : 0] alu_res
);

wire [31:0] add_out;
wire [31:0] sub_out;
wire [0 :0] slt_out;
wire [0 :0] sltu_out;

Comp_4 comp(
    .a(alu_src0),
    .b(alu_src1),
    .ul(sltu_out),
    .sl(slt_out)
);

always @(*) begin
    case(alu_op)
        `ADD :
            alu_res = alu_src0 + alu_src1;
        `SUB :
            alu_res = alu_src0 - alu_src1;
        `SLT :
            alu_res = {31'd0 , slt_out};
        `SLTU:
            alu_res = {31'd0 , sltu_out};
        `AND :
            alu_res = alu_src0 & alu_src1;
        `OR  :
            alu_res = alu_src0 | alu_src1;
        `XOR :
            alu_res = alu_src0 ^ alu_src1;
        `SLL :
            alu_res = alu_src0 << alu_src1[4:0];
        `SRL :
            alu_res = alu_src0 >> alu_src1[4:0];
        `SRA :
            alu_res = $signed(alu_src0) >>> alu_src1[4:0];
        `SRC0:
            alu_res = alu_src0;
        `SRC1:
            alu_res = alu_src1;
        default:
            alu_res = 32'H0;
    endcase
end
```

endmodule

> 任务 3：在线计算器

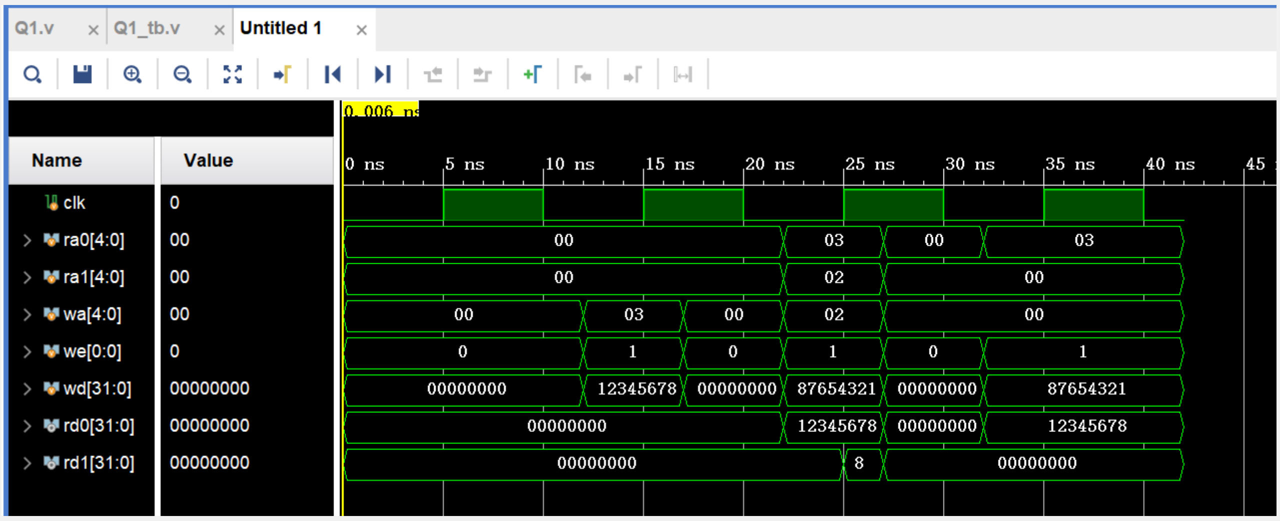
```
module TOP (
    input                   [ 0 : 0]            clk,
    input                   [ 0 : 0]            rst,

    input                   [ 0 : 0]            enable,
    input                   [ 4 : 0]            in,
    input                   [ 1 : 0]            ctrl,

    output                  [ 3 : 0]            seg_data,
    output                  [ 2 : 0]            seg_an
);
```

```
reg op_en,src0_en,src1_en,res_en;
wire [31:0] alu_res;
wire [31:0] outputdata;
wire [31:0] alu_src0;
wire [31:0] alu_src1;
wire [31:0] alu_op;

always @(posedge clk)begin
    if(rst) begin
        op_en <= 1'b0;
        src0_en <= 1'b0;
        src1_en <= 1'b0;
        res_en <= 1'b0;
    end
    else begin
            case(ctrl)
                2'b00:  begin
                    if(enable) begin
                            op_en <= 1'b1;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                    else begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                end
                2'b01:  begin
                    if(enable) begin
                            op_en <= 1'b0;
                            src0_en <= 1'b1;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                    else begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                end
                2'b10:  begin
                    if(enable) begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b1;
                            res_en <= 1'b0;
                    end
                    else begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                end
                2'b11:  begin
                    if(enable) begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b1;
                    end
                    else begin
                            op_en <= 1'b0;
                            src0_en <= 1'b0;
                            src1_en <= 1'b0;
                            res_en <= 1'b0;
                    end
                end
            endcase
    end
end

regfile reg0(
    .clk(clk),
    .rst(rst),
    .rf_we(src0_en),
    .rf_wd({27'd0,in}),
    .rf_rd(alu_src0)
);

regfile reg1(
    .clk(clk),
    .rst(rst),
    .rf_we(src1_en),
    .rf_wd({27'd0,in}),
    .rf_rd(alu_src1)
);

regfile reg2(
    .clk(clk),
    .rst(rst),
    .rf_we(op_en),
    .rf_wd({27'd0,in}),
    .rf_rd(alu_op)
);

ALU alu(
    .alu_src0(alu_src0),
    .alu_src1(alu_src1),
    .alu_op(alu_op[4:0]),
    .alu_res(alu_res)
);

regfile reg3(
    .clk(clk),
    .rst(rst),
    .rf_we(res_en),
    .rf_wd(alu_res),
    .rf_rd(outputdata)
);

Segment segment(
    .clk(clk),
    .rst(rst),
    .output_data(outputdata),
    .seg_data(seg_data),
    .seg_an(seg_an)
);

endmodule
```

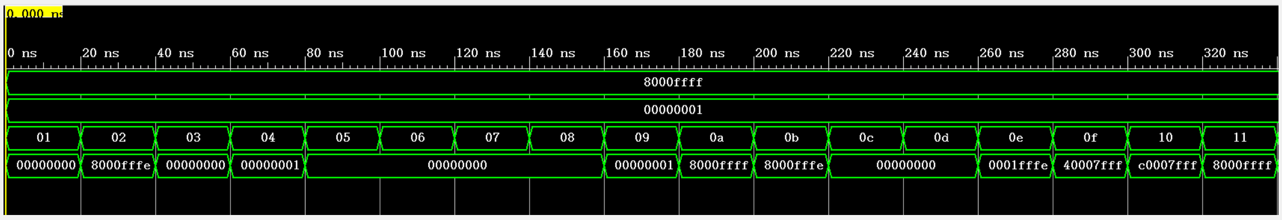
> 任务 4：初始化存储器
>
> 
> Re-customize IP
>
> **Distributed Memory Generator (8.0)**
>
> Documentation IP Location Switch to Defaults
>
> Show disabled ports
>
> Component Name: dist_mem_gen_0
>
> memory config | Port config | RST & Initialization
>
> **Options**
>
> Depth: 32 [16 - 65536]
>
> Data Width: 16 [1 - 1024]
>
> **Memory Type**
>
> **Memory Type**
>
> ROM
> [✓ Single Port RAM]
> Simple Dual Port RAM
> Dual Port RAM
>
>

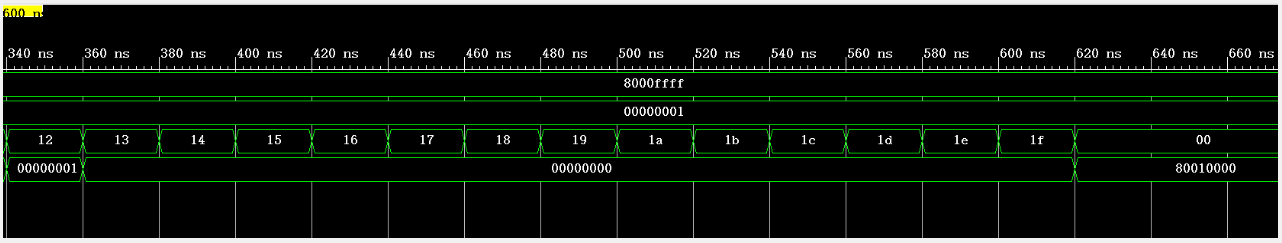
仿真结果分析

任务 1：寄存器堆设计



任务 2：ALU 设计

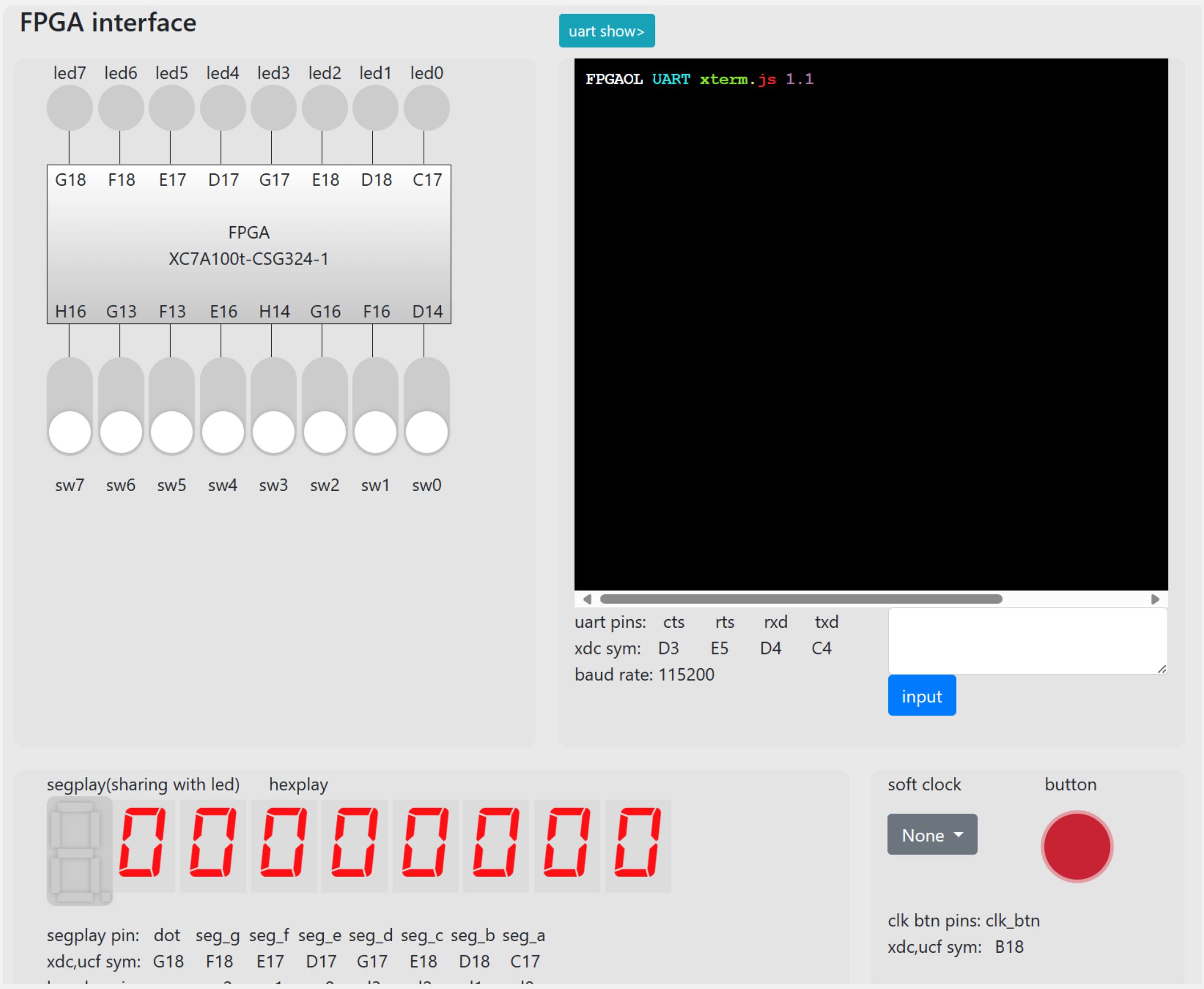

0.000 ns
0 ns 20 ns 40 ns 60 ns 80 ns 100 ns 120 ns 140 ns 160 ns 180 ns 200 ns 220 ns 240 ns 260 ns 280 ns 300 ns 320 ns
8000ffff
00000001
01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
00000000 8000fffe 00000000 00000001 00000000 00000001 8000ffff 8000fffe 00000000 0001fffe 40007fff c0007fff 8000ffff


600 ns
340 ns 360 ns 380 ns 400 ns 420 ns 440 ns 460 ns 480 ns 500 ns 520 ns 540 ns 560 ns 580 ns 600 ns 620 ns 640 ns 660 ns
8000ffff
00000001
12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 00
00000001 00000000 80010000

测试结果与分析

任务 3：在线计算器

1 + 2 = 3



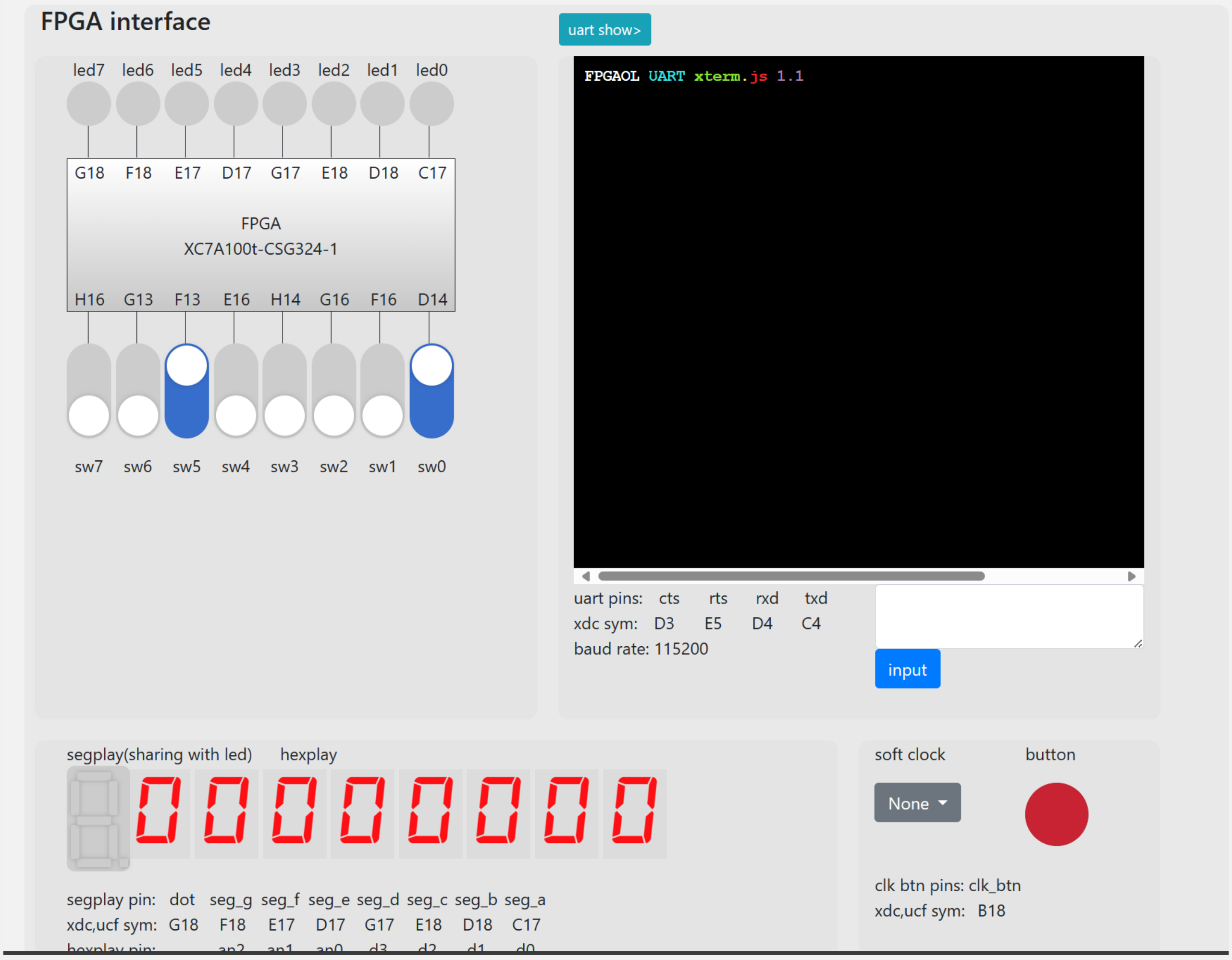
FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
input
segplay(sharing with led)
hexplay
soft clock
button
None

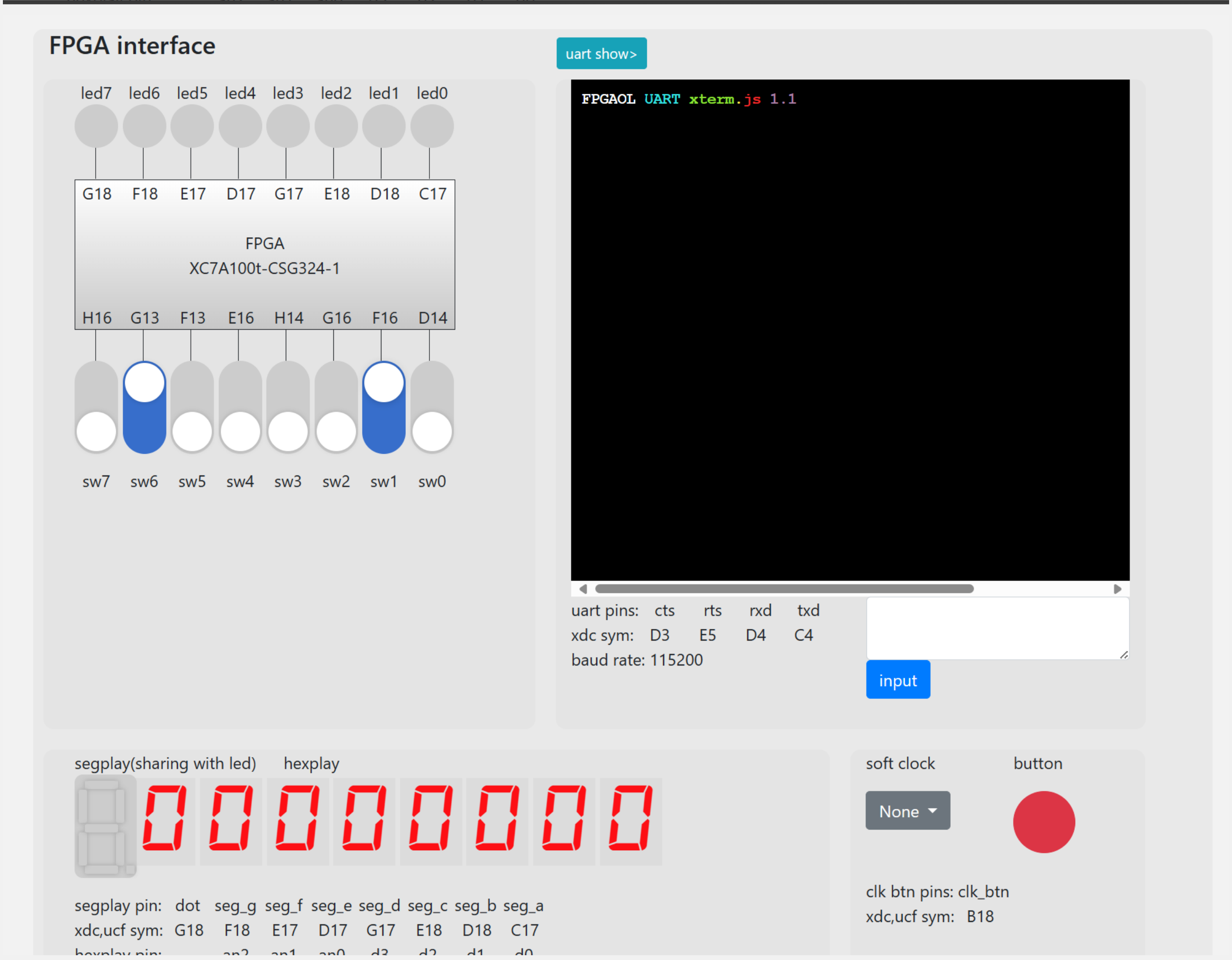
```
FPGAOL UART xterm.js 1.1
```

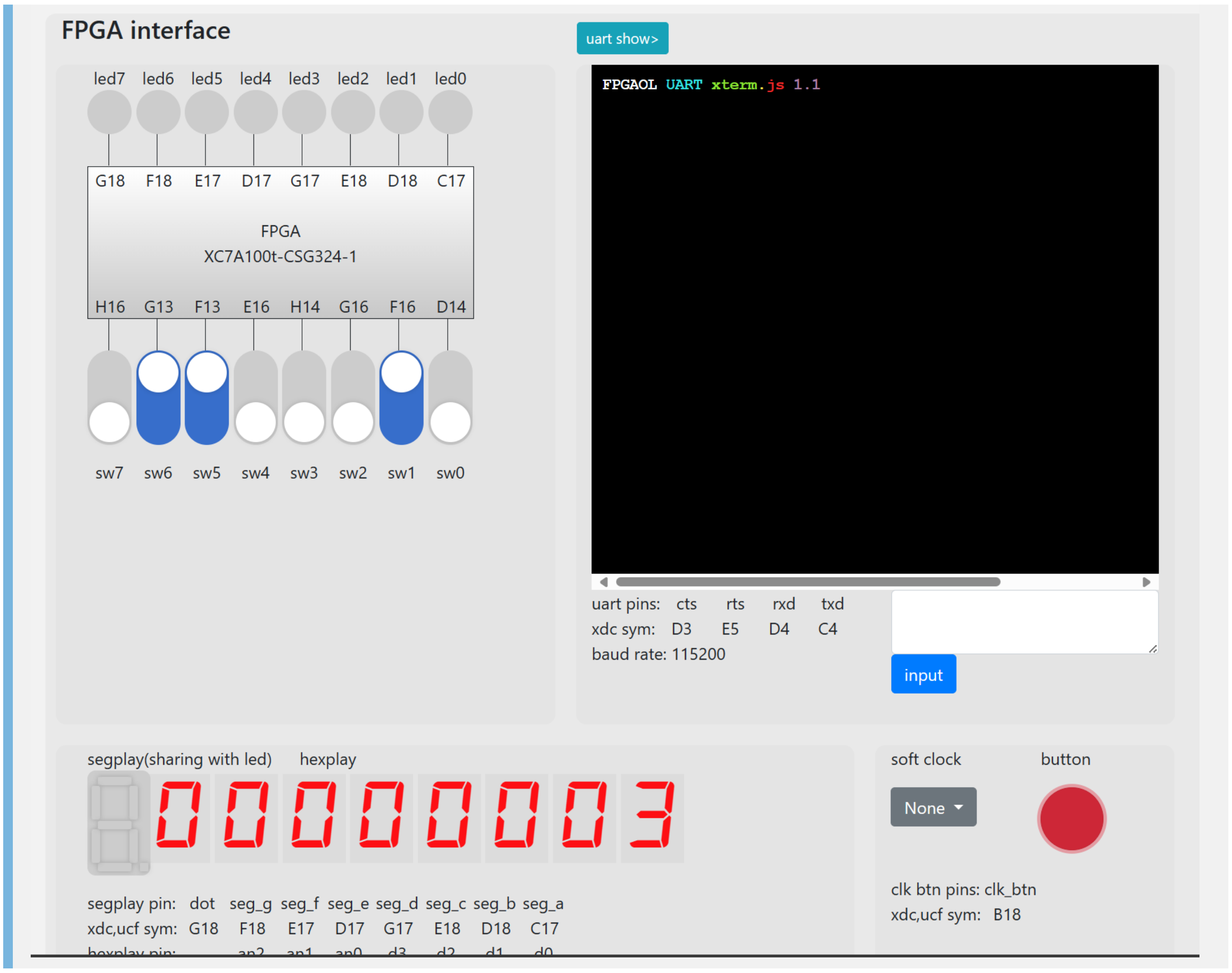
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

clk btn pins: clk_btn
xdc,ucf sym: B18


FPGA interface
uart show>
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 sym: D3 E5 D4 C4
baud rate: 115200
input
segplay(sharing with led) hexplay
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17


FPGA interface
uart show>
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 sym: D3 E5 D4 C4
baud rate: 115200
input
segplay(sharing with led) hexplay
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17


FPGA interface
uart show>
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
input
segplay(sharing with led)
hexplay
soft clock
button
None

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

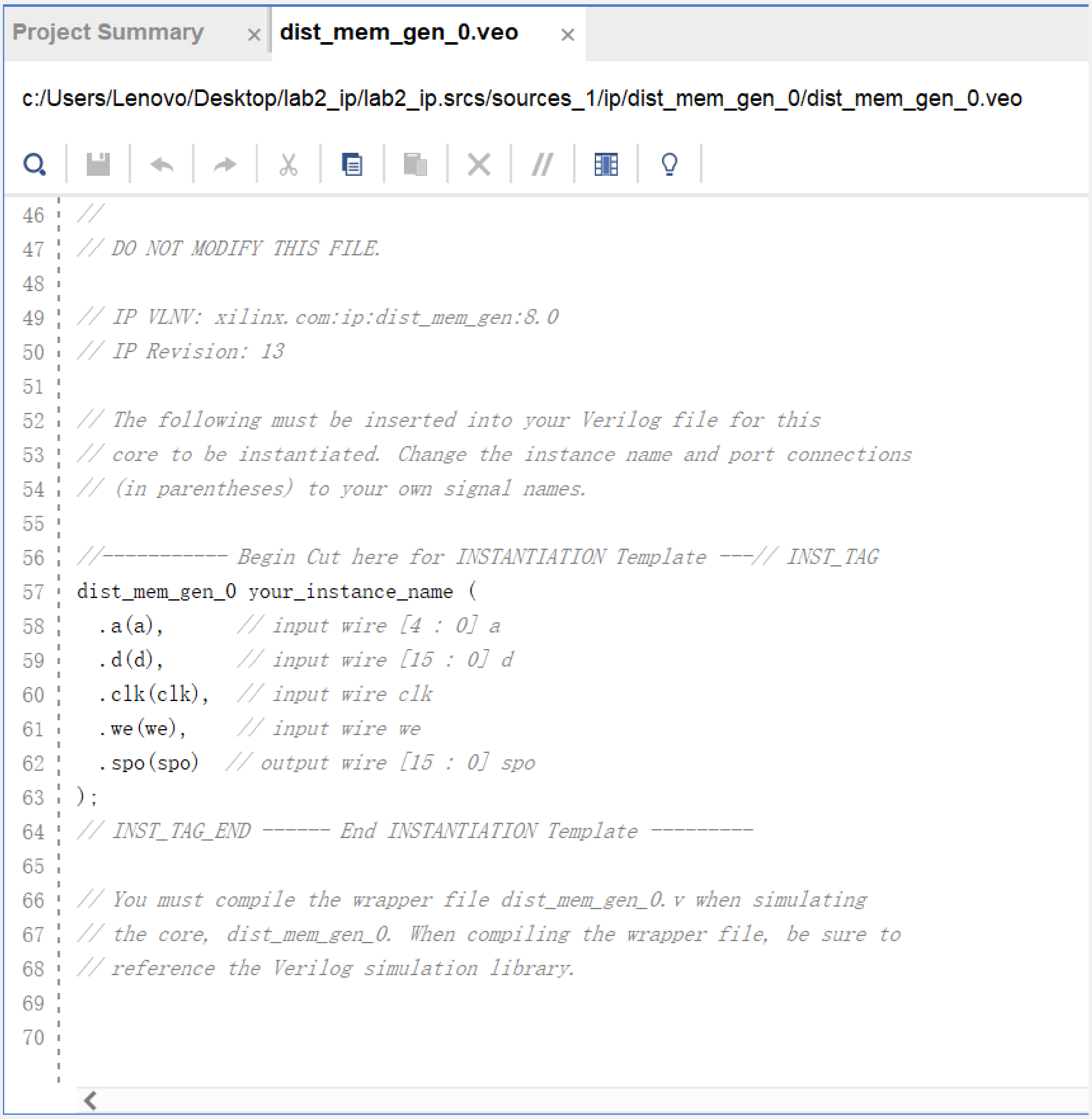
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

00000003

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0

clk btn pins: clk_btn
xdc,ucf sym: B18

任务 4：初始化存储器



c:/Users/Lenovo/Desktop/lab2_ip/lab2_ip.srcs/sources_1/ip/dist_mem_gen_0/dist_mem_gen_0.veo

```
//
// DO NOT MODIFY THIS FILE.

// IP VLNV: xilinx.com:ip:dist_mem_gen:8.0
// IP Revision: 13

// The following must be inserted into your Verilog file for this
// core to be instantiated. Change the instance name and port connections
// (in parentheses) to your own signal names.

//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG
dist_mem_gen_0 your_instance_name (
  .a(a),      // input wire [4 : 0] a
  .d(d),      // input wire [15 : 0] d
  .clk(clk),  // input wire clk
  .we(we),    // input wire we
  .spo(spo)  // output wire [15 : 0] spo
);
// INST_TAG_END ------ End INSTANTIATION Template ---------

// You must compile the wrapper file dist_mem_gen_0.v when simulating
// the core, dist_mem_gen_0. When compiling the wrapper file, be sure to
// reference the Verilog simulation library.
```

总结

复习实现了ALU和寄存器等功能，学习了ip核。