Lab4实验报告

521030910007 翁牧言

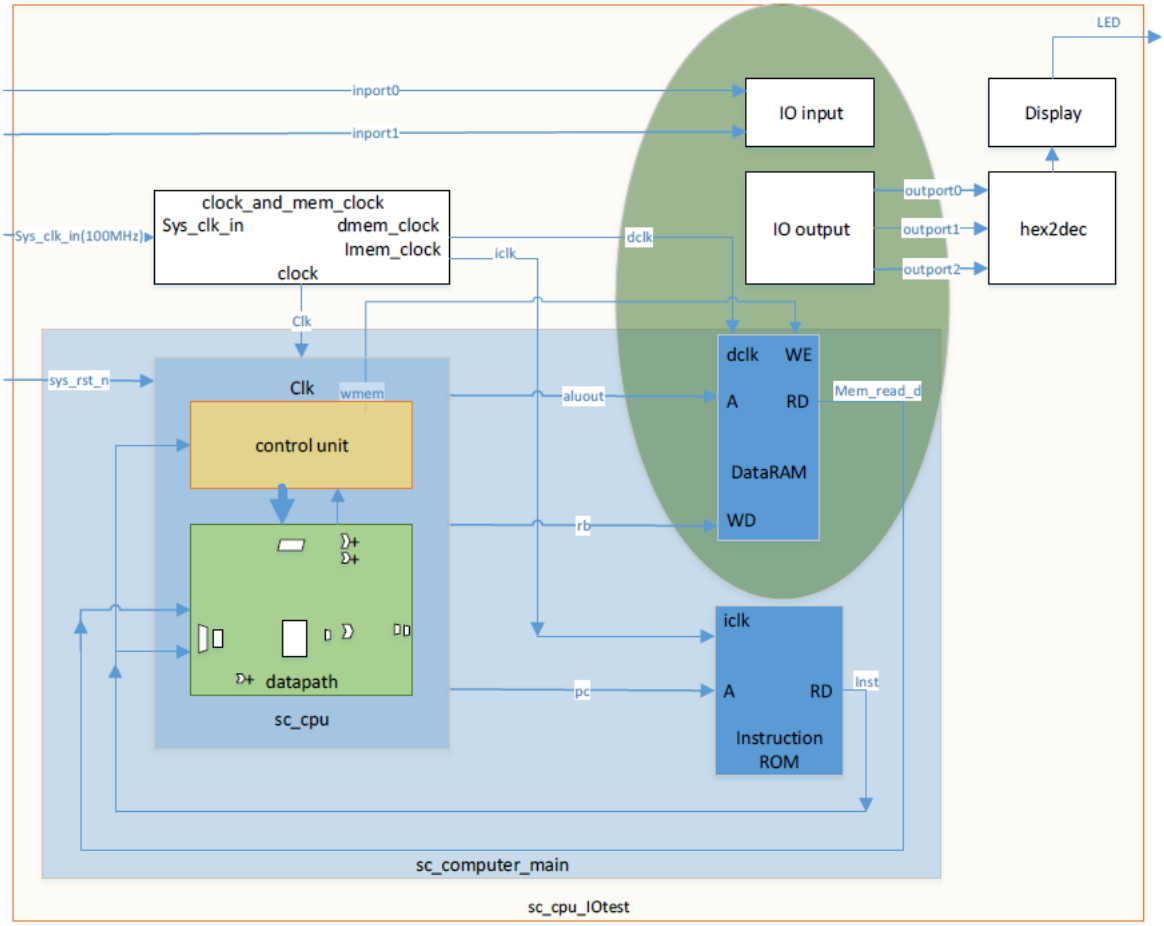
实验目的

1. 在lab3的基础上为CPU增加IO模块，实现输入并显示输出结果
2. 增加一条汉明距离的指令，并支持相关操作

实验思路与核心代码

实验文件树以及设计图如下


Design Sources (3)
sc_cpu_iotest (sc_cpu_iotest.v) (8)
clock_and_mem_clock : clock_and_mem_clock (clock_and_mem_clock.v)
in_port0 : in_port (in_port.v)
in_port1 : in_port (in_port.v)
sc_computer_main : sc_computer_main (sc_computer_main.v) (3)
CPU : sc_cpu (sc_cpu.v) (12)
ip : dff32 (dff32.v)
ImmGen : immext (immext.v)
regfile : regfile (regfile.v)
cu : sc_cu (sc_cu.v)
alu_b : mux2x32 (mux2x32.v)
al_unit : alu (alu.v)
pcplus4 : cla32 (cla32.v)
branch_adr : cla32 (cla32.v)
genjalrpc : cla32 (cla32.v)
nextpc : mux4x32 (mux4x32.v)
link : mux2x32 (mux2x32.v)
result : mux2x32 (mux2x32.v)
imem : sc_instmem (sc_instmem.v) (1)
irom : lpm_rom_irom (lpm_rom_irom.xci)
dmem : sc_datamem (sc_datamem.v) (4)
io_data_mux : mux2x32 (mux2x32.v)
dram : lpm_ram_dq_dram (lpm_ram_dq_dram.xci)
io_output : io_output (io_output.v)
io_input : io_input (io_input.v) (1)
dec54 : out_port_hex2dec (out_port_hex2dec.v)
dec32 : out_port_hex2dec (out_port_hex2dec.v)
dec10 : out_port_hex2dec (out_port_hex2dec.v)
display : display (display.v) (1)
Coefficient Files (2)
Constraints (1)
constrs_1 (1)
sc_computer_iotest.xdc



a. 实现IO

这次实验中需要对lab3进行一些修改主要包括以下几个地方

- **修改sc_datamem**

此处主要包括io_input和io_output模块的实例化

```
module sc_datamem (resetn,
    addr,datain,dataout,we,clock,dmem_clock,
    out_port0,out_port1,out_port2,in_port0,in_port1,);

   input  [31:0]  addr;
   input  [31:0]  datain;
   input [31:0] in_port0,in_port1;
   input          we, clock,dmem_clock,resetn;
   output [31:0]  dataout;
   output [31:0] out_port0,out_port1,out_port2;
   wire [31:0] io_read_data;

   wire write_enable;
   wire [31:0] dataout;
   wire [31:0] mem_dataout;
   wire write_data_enable;
   wire write_io_enable;

    //control signal for write dram or IO space
   assign         write_enable = we & ~clock;
```

```
    assign write_data_enable =write_enable & (~addr[7]);
    assign write_io_enable = write_enable & addr[7];

 //mux of mem_dataout and io_read_data

    mux2x32 io_data_mux(mem_dataout,io_read_data,addr[7],dataout);


//  dram
lpm_ram_dq_dram dram (
  .clka(dmem_clock),    // input wire clka
  .wea(write_data_enable),      // input wire [0 : 0] wea
  .addra(addr[6:2]),  // input wire [4 : 0] addra
  .dina(datain),    // input wire [31 : 0] dina
  .ena(1'b1),
  .douta(mem_dataout)  // output wire [31 : 0] douta
);

//IO output , io_output ,add here
io_output io_output(
.resetn(resetn),
.addr(addr),
.datain(datain),
.write_io_enable(write_io_enable),
.io_clk(dmem_clock),
.out_port0(out_port0),
.out_port1(out_port1),
.out_port2(out_port2)
);



//IOinput , io_input ,add here
io_input io_input(
.addr(addr),
.io_clk(dmem_clock),
.io_read_data(io_read_data),
.in_port0(in_port0),
.in_port1(in_port1)
);



endmodule
```

- **进行sc_computer_main的链接**

根据设计图进行链接，需要注意dmem_clock和imem_clock的区别

```
//////////////////////////////////////////////////////////////////
//                                                              //
// School of SEIEE of SJTU                                    //
//                                                              //
```

```
//////////////////////////////////////////////////////////////

module sc_computer_main (
    resetn,clock,imem_clock,dmem_clock,pc,inst,aluout,memout,
    out_port0,out_port1,out_port2,in_port0,in_port1);

   input resetn,clock,imem_clock,dmem_clock;
    input [31:0] in_port0,in_port1;
   output [31:0] pc,inst,aluout,memout;
   output [31:0] out_port0,out_port1,out_port2;
   wire   [31:0] data;
   wire          wmem; // all these "wire"s are used to connect or interface the
cpu,dmem,imem and so on.



 //sc_cpu , CPU module. add here
    sc_cpu CPU(
   .clock(clock),
   .resetn(resetn),
   .inst(inst),
   .mem(memout),
   .pc(pc),
   .wmem(wmem),
   .aluout(aluout),
   .data(data)
   );



    sc_instmem  imem (
   .addr(pc),
   .inst(inst),
   .clock(clock),
   .imem_clock(imem_clock)
);


sc_datamem dmem(
.resetn(resetn),
.addr(aluout),
.datain(data),
.dataout(memout),
.we(wmem),
.clock(clock),
.dmem_clock(dmem_clock),
.out_port0(out_port0),
.out_port1(out_port1),
.out_port2(out_port2),
.in_port0(in_port0),
.in_port1(in_port1)
);


endmodule
```

- **完成sc_cpu_iotest的链接**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Company: SJTU
// Engineer:
//
// Create Date: 2021/10/29 14:57:06
// Design Name: CYQ
// Module Name: sc_cpu_iotest
// Project Name: 32BIT RISC-V CPU IO TEST
// Target Devices: ARTIX-7, xc7a50tcsg324-1
// Tool Versions: VIVADO 2020.2
// Description:DISPLAY THE SUM OF 2 5BIT-INPUT DATA AT THE LEDS AND DIGITRONS.
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//


module sc_cpu_iotest(
    input sys_rst_n,  //active low global reset
    input sys_clk_in, //100MHz clock
    input [4:0] sw_pin,//left 8 sw_pin
    input [4:0] dip_pin,//right 8 dip_pin
    output [7:0] seg_data_0_pin, //output DP1,G1,F1,E1,D1,C1,B1,A1, left
    output [7:0] seg_data_1_pin,  //output DP0,G0,F0,E0,D0,C0,B0,A0,  right
    output [7:0] seg_cs_pin,
//DN1_K4,DN1_K3,DN1_K2,DN1_K0,DN0_K4,DN0_K3,DN0_K2,DN0_K1 ,left to right
    output [0:15] led_pin //left to right
     );

    wire [3:0] HEX4b5,HEX4b4,HEX4b3,HEX4b2,HEX4b1,HEX4b0;
    wire clock, imem_clock, dmem_clock;
    wire [31:0] inport0, inport1;
    wire[31:0] out_port0, out_port1, out_port2;
    wire [31:0] pc,inst,aluout,memout;


  //clock_and_mem_clock unit, generate clock, imem_clock, dmem_clock  , add here
clock_and_mem_clock clock_and_mem_clock(
  .sys_clk_in(sys_clk_in),
  .clock_out(clock),
  .imem_clock(imem_clock),
  .dmem_clock(dmem_clock)
  );
```

```
  //extend in_port0 to 32bit,  in_port0 = {27'b0,sw_pin}; add here
in_port in_port0(
.sw(sw_pin),
.out(inport0)
);



  //extend in_port1 to 32bit,  in_port1 = {27'b0,dip_pin}; add here
 in_port in_port1(
 .sw(dip_pin),
 .out(inport1)
 );



  //sc_computer_main unit  , add here
sc_computer_main sc_computer_main(
.resetn(sys_rst_n),
.clock(clock),
.imem_clock(imem_clock),
.dmem_clock(dmem_clock),
.pc(pc),
.inst(inst),
.aluout(aluout),
.memout(memout),
.out_port0(out_port0),
.out_port1(out_port1),
.out_port2(out_port2),
.in_port0(inport0),
.in_port1(inport1)
);




 //led sign
    assign led_pin[10:15] = out_port2[5:0];
    assign led_pin[0:4] = out_port0[4:0];
    assign led_pin[5:9] = out_port1[4:0];

// out_port_hex2dec unit , outport 0 data(=in_port0) hex to decimal
   out_port_hex2dec dec54(out_port0,HEX4b5,HEX4b4);  // input 0

// out_port_hex2dec unit ,outport1 data(=in_port1) hex to decimal
   out_port_hex2dec dec32(out_port1,HEX4b3,HEX4b2); //input 1

// out_port_hex2dec unit ,sum of in_port0 and in_port1 hex to decimal
   out_port_hex2dec dec10(out_port2,HEX4b1,HEX4b0);  //output

//display unit, seven segment decode and digitron drive
display display(
```

```
  .clk(sys_clk_in),
  .reset(sys_rst_n),
  .s({{8'b0},HEX4b5,HEX4b4,HEX4b3,HEX4b2,HEX4b1,HEX4b0}),
  .seg0(seg_data_0_pin),
  .seg1(seg_data_1_pin),
  .ans(seg_cs_pin)
    );

endmodule
```

b. 实现汉明距离

首先我们要定义该计算的控制信号

```
 wire i_ham = r_type&(func3==3'b111)&(func7 == 7'b0100000);
```

然后我们需要将aluc和wreg信号中加入该信号

```
 assign aluc[3] =i_sub|i_sra|i_srai|i_beq|i_bne |i_ham  ;
   assign aluc[2]
=i_and|i_or|i_xor|i_srl|i_sra|i_andi|i_ori|i_xori|i_srli|i_srai  |i_ham ;
   assign aluc[1] =i_and|i_or|i_andi|i_ori|i_lui |i_ham  ;
   assign aluc[0] =i_and|i_sll|i_srl|i_sra|i_andi|i_slli|i_srli|i_srai |i_ham  ;

assign wreg = i_add | i_sub | i_and | i_or   | i_xor  |
                i_sll | i_srl | i_sra | i_addi | i_andi |
                i_ori | i_xori | i_srli | i_slli | i_srai | i_lw | i_jalr |
i_lui  | i_jal | i_ham;
```

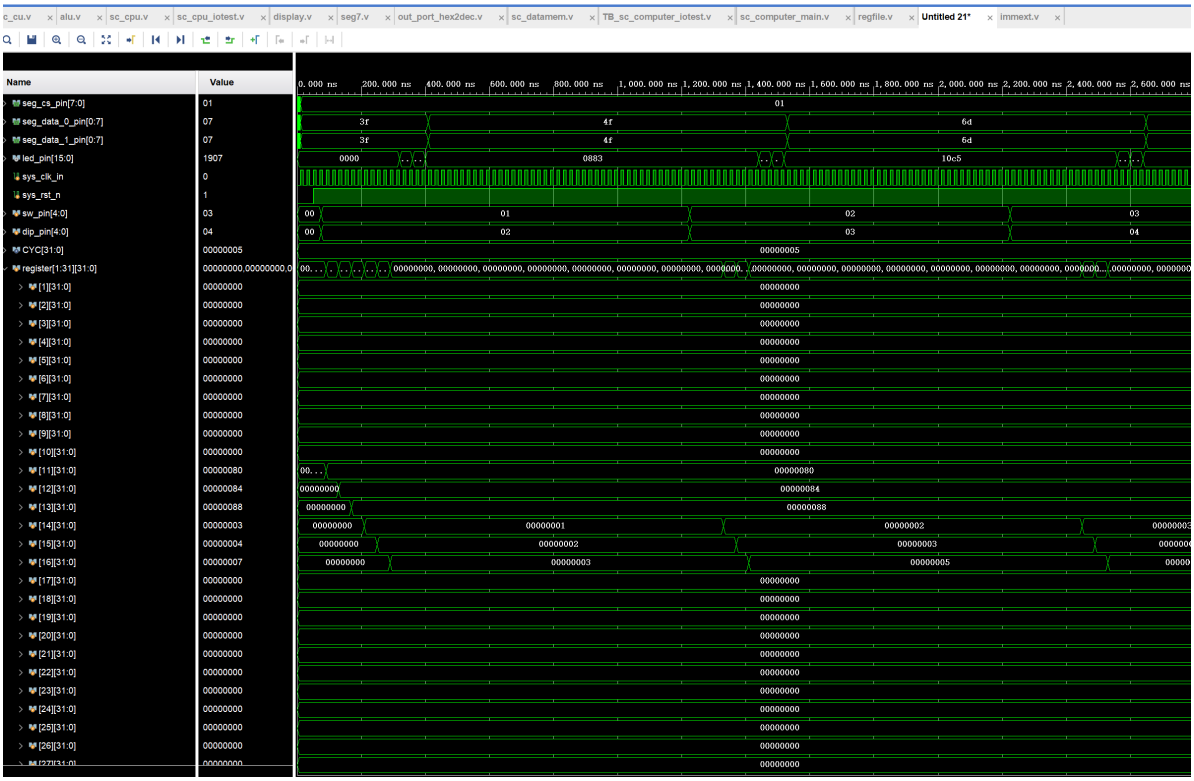
最后我们需要在alu中完成计算汉明距离的函数

```
  function [31:0] ham;
    input [31:0] a,b;
    integer i;
    begin
        ham=32'b0;
        for(i=0;i<32;i=i+1'b1)
            begin
                if((a[k])^(b[i])==1'b1)
                begin
                ham=ham+1'b1;
            end
        end
    end
  endfunction
```

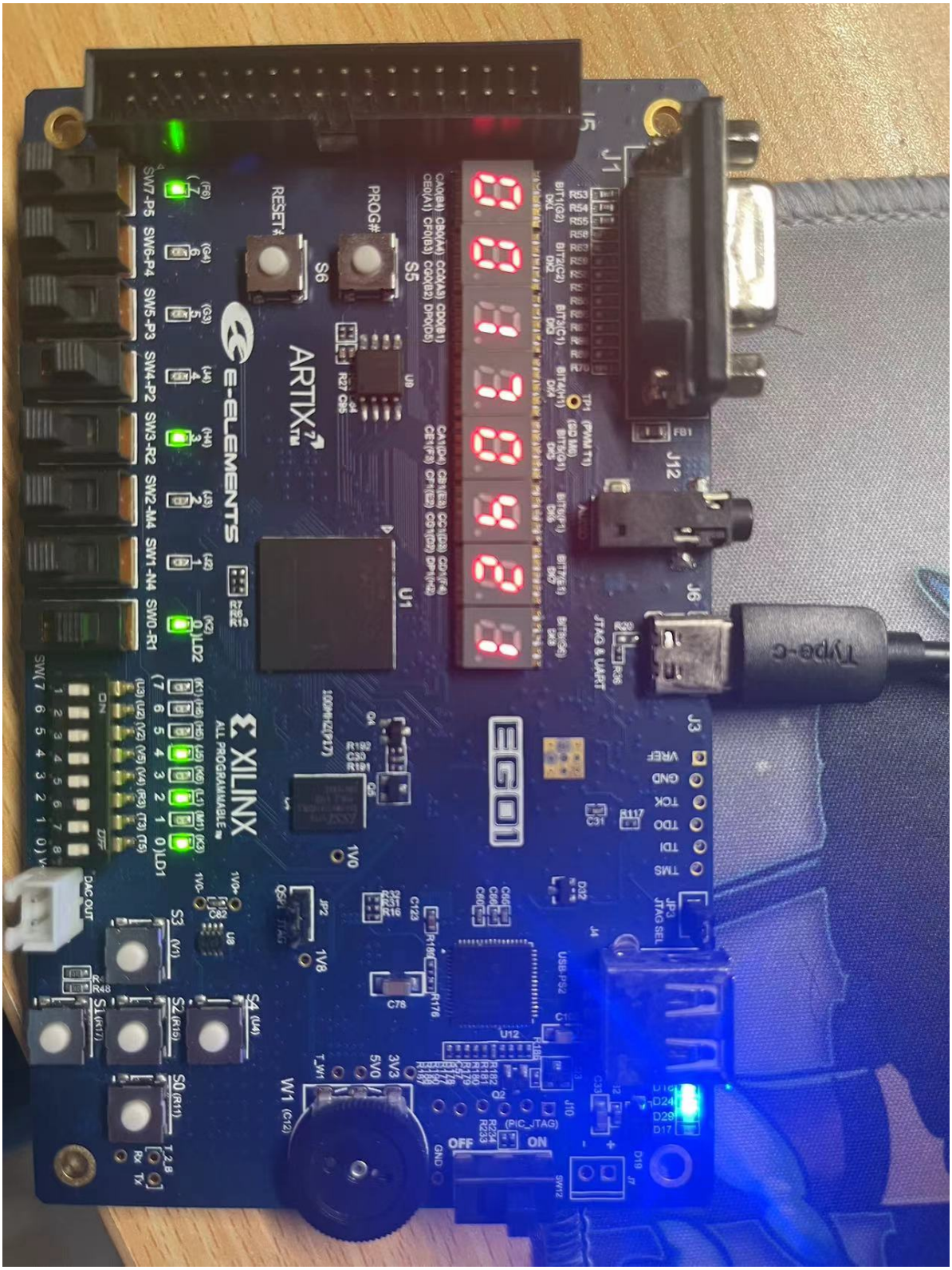
结果展示

指令为加法时：

仿真结果

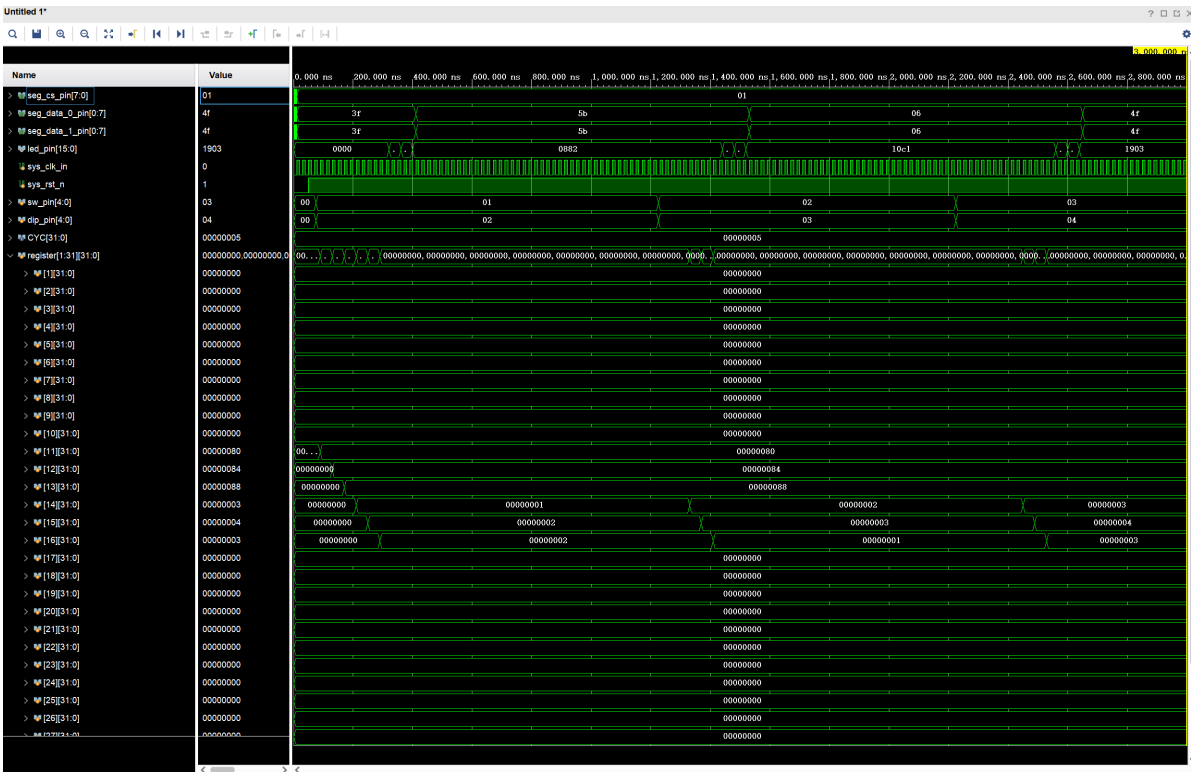

c_cu.v
alu.v
sc_cpu.v
sc_cpu_iotest.v
display.v
seg7.v
out_port_hex2dec.v
sc_datamem.v
TB_sc_computer_iotest.v
sc_computer_main.v
regfile.v
Untitled 21*
immext.v
Name
Value
seg_cs_pin[7:0]
seg_data_0_pin[0:7]
seg_data_1_pin[0:7]
led_pin[15:0]
sys_clk_in
sys_rst_n
sw_pin[4:0]
dip_pin[4:0]
CYC[31:0]
register[1:31][31:0]
0. 000 ns
200. 000 ns
400. 000 ns
600. 000 ns
800. 000 ns
1, 000. 000 ns
1, 200. 000 ns
1, 400. 000 ns
1, 600. 000 ns
1, 800. 000 ns
2, 000. 000 ns
2, 200. 000 ns
2, 400. 000 ns
2, 600. 000 ns

板级验证

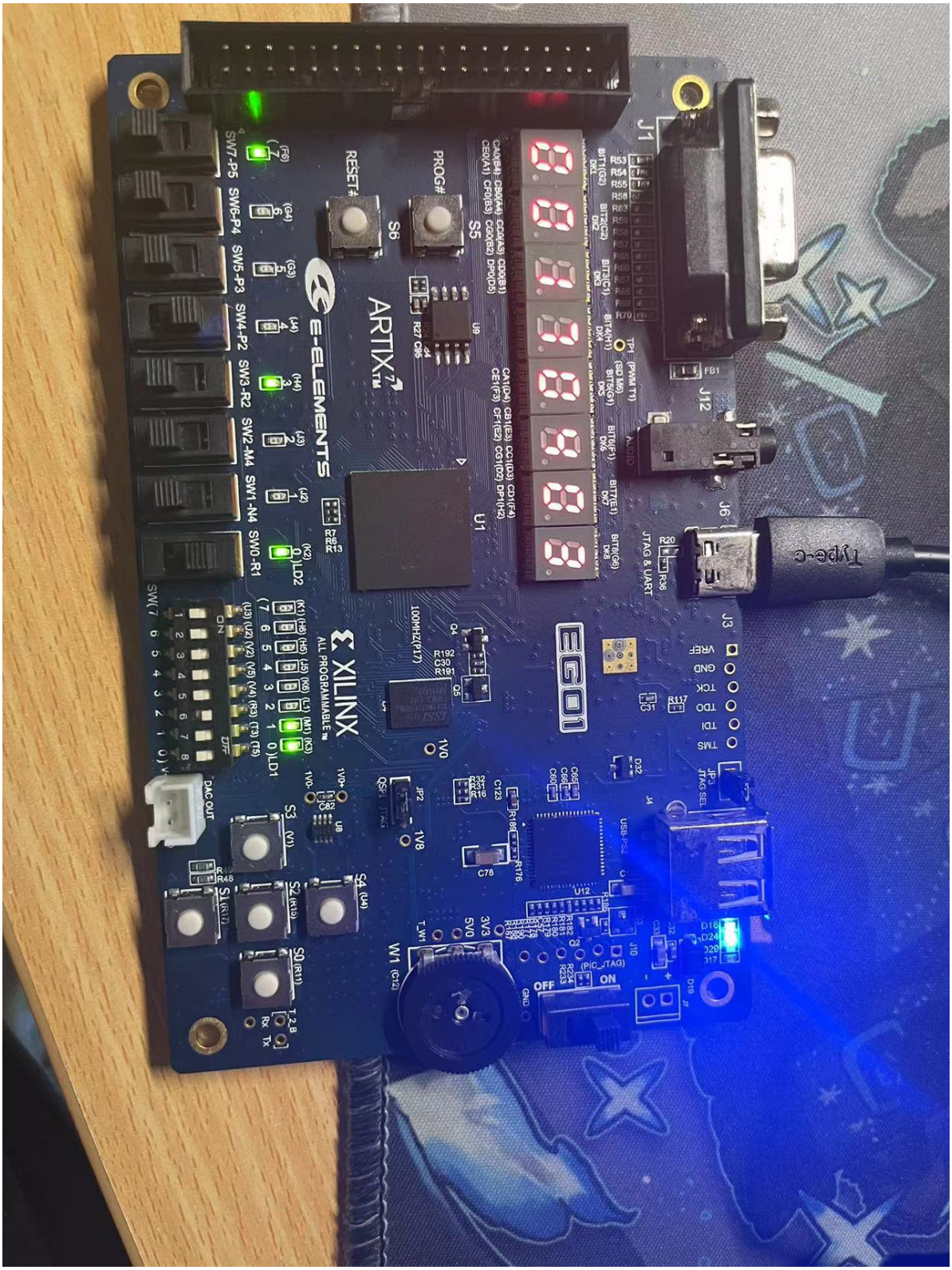


指令为求汉明距离时

仿真结果


Untitled 1*
Name
Value
seg_cs_pin[7:0]
seg_data_0_pin[0:7]
seg_data_1_pin[0:7]
led_pin[15:0]
sys_clk_in
sys_rst_n
sw_pin[4:0]
dip_pin[4:0]
CYC[31:0]
register[1:31][31:0]
01
4f
4f
1903
0
1
03
04
00000005
00000000.00000000.0
0. 000 ns
200. 000 ns
400. 000 ns
600. 000 ns
800. 000 ns
1, 000. 000 ns
1, 200. 000 ns
1, 400. 000 ns
1, 600. 000 ns
1, 800. 000 ns
2, 000. 000 ns
2, 200. 000 ns
2, 400. 000 ns
2, 600. 000 ns
2, 800. 000 ns
3f
5b
06
4f
0000
0882
10c1
1903
00
01
02
03
04
00000000
00000080
00000084
00000088
00000001
00000002
00000003

板级验证



总结

到了Lab4，我们已经可以实现输入输出的交互以及自定义指令。

一些心得：

1. 在这次实验中发现了一些之前代码中的bug，这提示我们在检查时要采用更多的测试
2. 这次的xdc文件生成比特流时，会产生报错，而我们需要根据报错信息加入以下代码

```
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets
{sc_computer_main/CPU/regfile/dram_i_298_0}];
```

注意get_nets中的内容会由报错信息给出