

中国科学技术大学计算机学院
《计算机组成原理》实验报告



实验题目：LabH5_流水线 CPU 设计

学生姓名：钟书锐

学生学号：PB19000362

完成日期：2021.6.9

计算机实验教学中心制

2020 年 09 月

【实验题目】

LabH5_流水线 CPU 设计

【实验目的】

- 理解 CPU 的结构和工作原理
- 掌握流水线 CPU 的设计和调试方法
- 熟练掌握数据通路和控制器的设计和描述方法

【实验环境】

硬件：

处理器：i7-10750H @ 2.60GHz 六核

显卡：RTX2060(6GB)

操作系统：

WINDOWS10 家庭中文版

软件：

Vivado

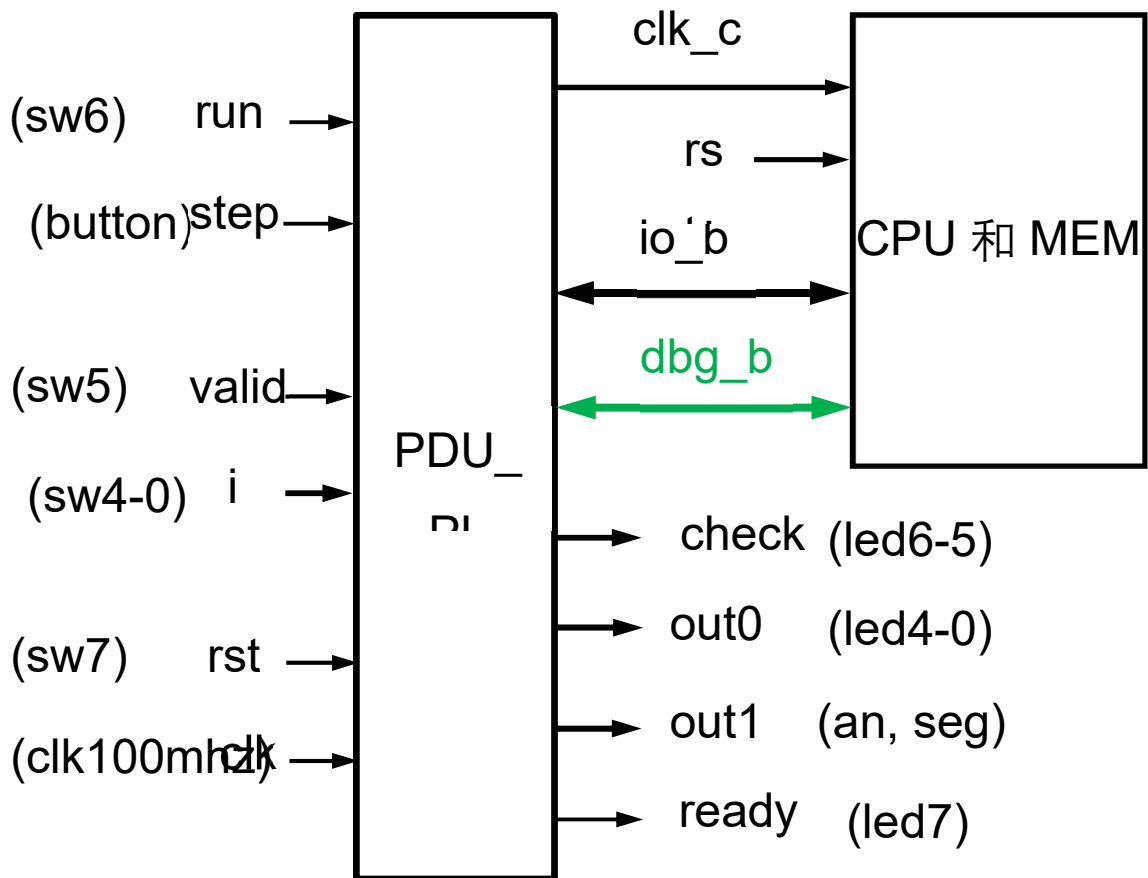
【实验内容】

设计实现 5 级流水线的 RISC-V CPU

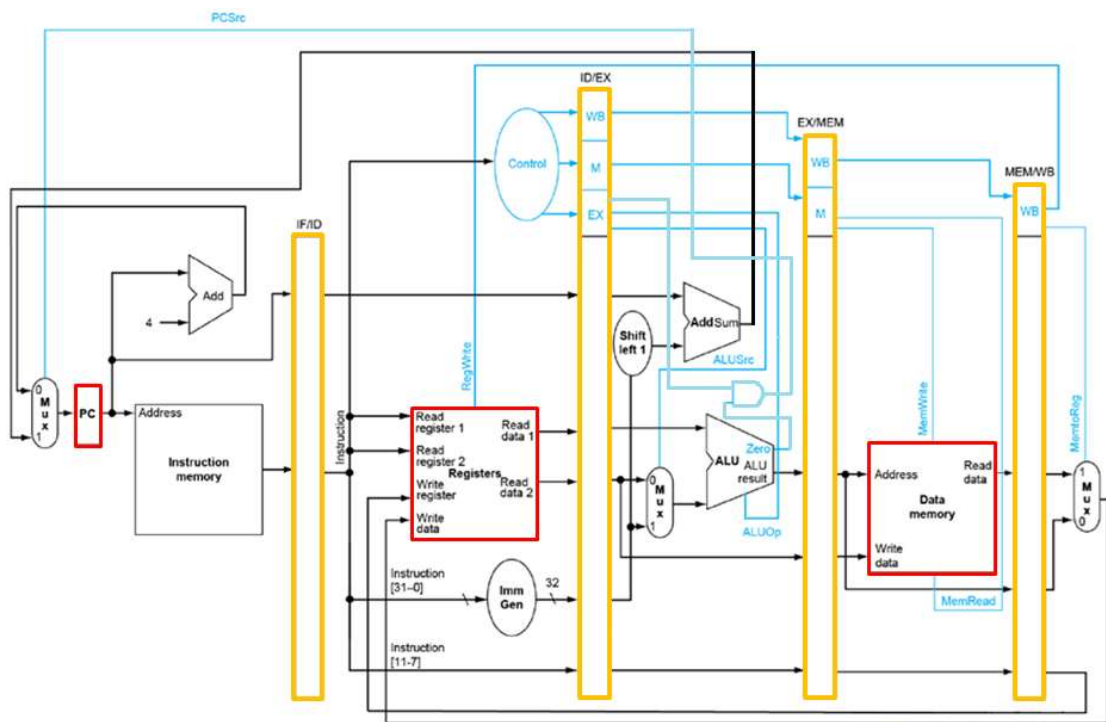
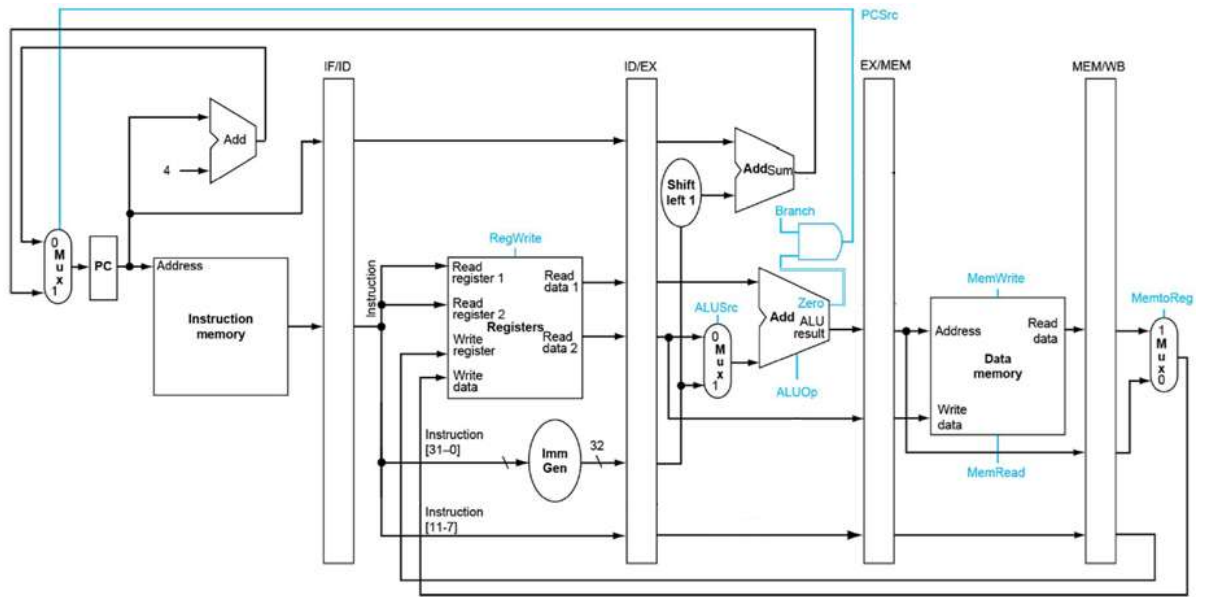
能够执行 6 条指令：add, addi, lw, sw, beq, jal

指令存储器和数据存储器均使用分布式存储器，容量均为 256x32 位，

寄存器堆和数据存储器均增加一个读端口用于调试



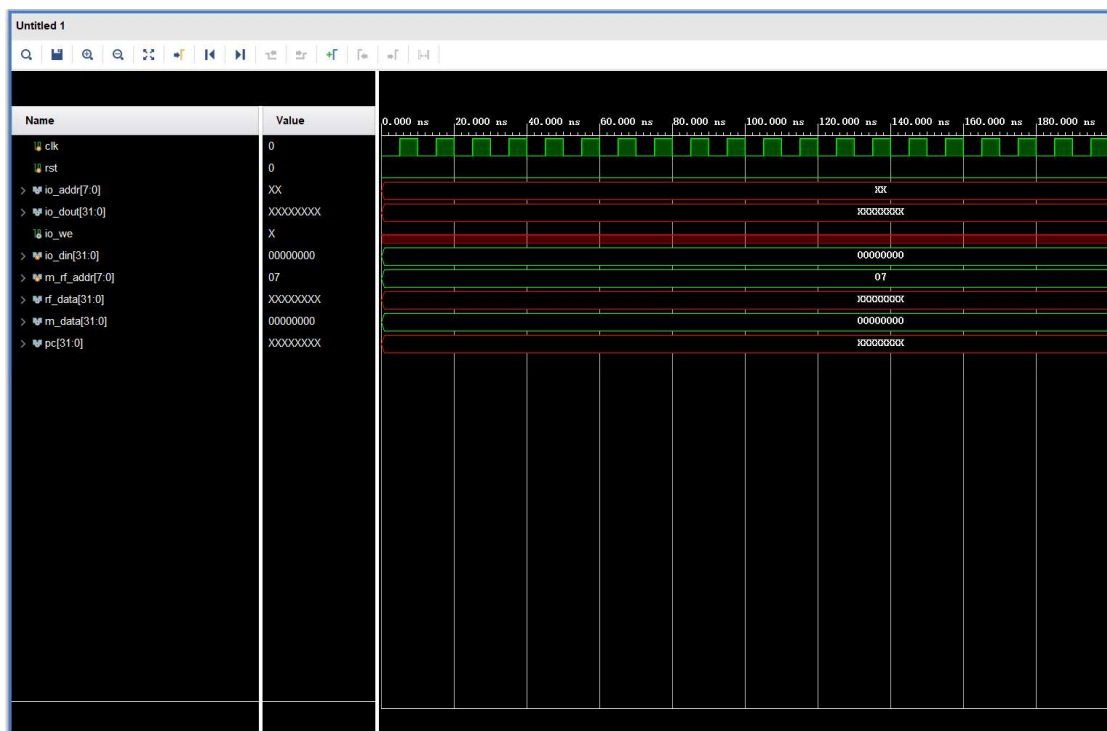
1.数据通路



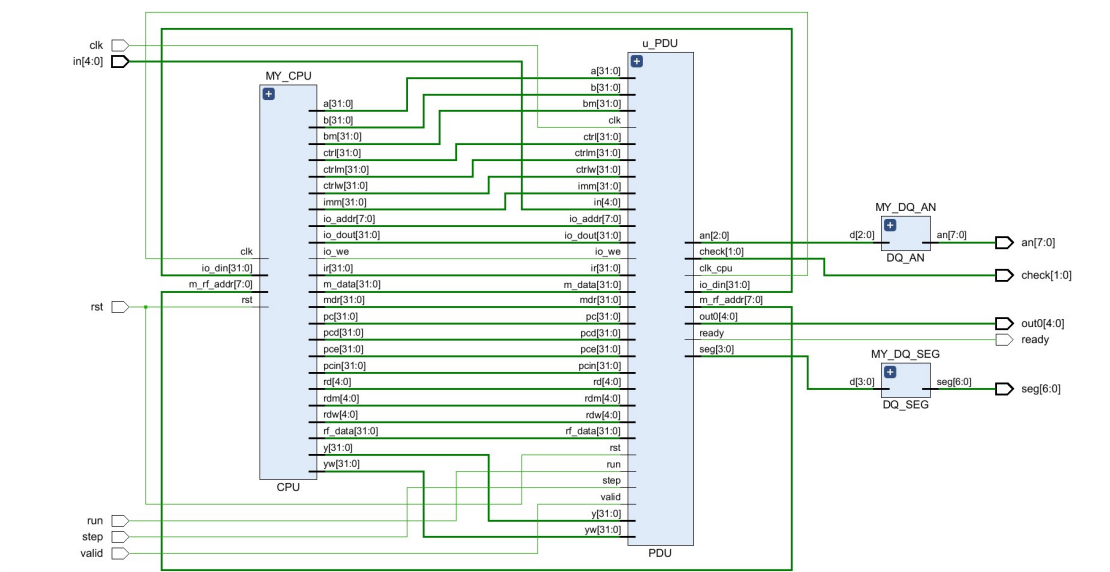
2.设计逻辑

- PDU (Processor Debug Unit)负责处理处理器调试
- CPU 负责运算
- TOP 负责调用 PDU 模块与 CPU 模块

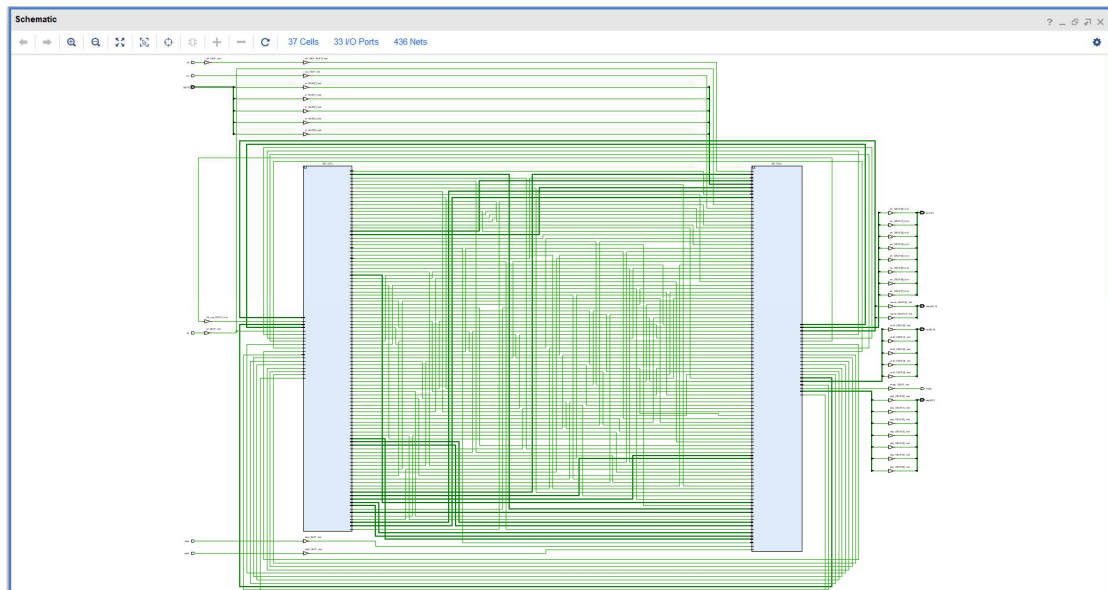
3.仿真测试



4.RTL 电路



5.综合电路



6.核心代码

```
CPU MY_CPU (  
  .clk           ( clk_cpu           ),  
  .rst           ( rst               ),  
  .io_din        ( io_din [31:0] ),  
  .m_rf_addr     ( m_rf_addr [7:0] ),  
  
  .io_addr       ( io_addr [7:0] ),  
  .io_dout       ( io_dout [31:0] ),  
  .io_we         ( io_we            ),  
  .rf_data       ( rf_data [31:0] ),  
  .m_data        ( m_data [31:0] ),  
  .pc            ( pc [31:0] ),  
  .pcd           ( pcd [31:0] ),  
  .ir            ( ir [31:0] ),  
  .pcin          ( pcin [31:0] ),  
  .pce           ( pce [31:0] ),  
  .a             ( a [31:0] ),  
  .b             ( b [31:0] ),  
  .imm           ( imm [31:0] ),  
  .rd            ( rd [4:0] ),  
  .ctrl          ( ctrl [31:0] ),  
  .y             ( y [31:0] ),  
  .bm           ( bm [31:0] ),  
  .rdm           ( rdm [4:0] ),  
  .ctrlm         ( ctrlm [31:0] ),  
  .yw           ( yw [31:0] ),  
  .mdr          ( mdr [31:0] ),  
  .rdw           ( rdw [4:0] ),  
  .ctrlw         ( ctrlw [31:0] )  
)
```

```

PDU u_PDU (
    .clk          ( clk          ),
    .rst          ( rst          ),
    .run          ( run          ),
    .step         ( step         ),
    .valid        ( valid        ),
    .in           ( in           [4:0] ),
    .io_addr      ( io_addr      [7:0] ),
    .io_dout      ( io_dout      [31:0] ),
    .io_we        ( io_we        ),
    .rf_data      ( rf_data      [31:0] ),
    .m_data       ( m_data       [31:0] ),
    .pcin         ( pcin         [31:0] ),
    .pc           ( pc           [31:0] ),
    .pcd          ( pcd          [31:0] ),
    .pce          ( pce          [31:0] ),
    .ir           ( ir           [31:0] ),
    .imm          ( imm          [31:0] ),
    .mdr          ( mdr          [31:0] ),
    .a            ( a            [31:0] ),
    .b            ( b            [31:0] ),
    .y            ( y            [31:0] ),
    .bm           ( bm           [31:0] ),
    .yw           ( yw           [31:0] ),
    .rd           ( rd           [4:0] ),
    .rdm          ( rdm          [4:0] ),
    .rdw          ( rdw          [4:0] ),
    .ctrl         ( ctrl         [31:0] ),
    .ctrlm        ( ctrlm        [31:0] ),
    .ctrlw        ( ctrlw        [31:0] ),

```

```

FORWARDING MY_FORWARDING (
    .rs1          ( rs1 ),
    .rs2          ( rs2 ),
    .wbm          ( ctrlm[18] ),
    .wbw          ( ctrlw[18] ),
    .rdm          ( rdm ),
    .rdw          ( rdw ),

    .a_fwd        ( a_fwd ),
    .b_fwd        ( b_fwd )
);

```

```

HAZARD MY_HAZARD (
    .PCSrc        ( PCSrc        ),
    .rs1          ( ir[19:15]    ),
    .rs2          ( ir[24:20]    ),
    .opcode       ( opcode[6:0]  ),
    .rd           ( rd [4:0]     ),
    .m_rd         ( ctrl[13]     ),
    .wbm          ( ctrlm[18]    ),
    .wbw          ( ctrlw[18]    ),

    .fstall       ( fstall       ),
    .dstall       ( dstall       ),
    .dflush       ( dflush       ),
    .eflush       ( eflush       )
);

```

```

module FORWARDING(
    input [4:0] rs1,
    input [4:0] rs2,
    input wbm,
    input wbw,
    input [4:0] rdm,
    input [4:0] rdw,
    output reg [1:0] a_fwd,
    output reg [1:0] b_fwd
);

always @(*)
begin
    if (wbm && rdm && rs1 == rdm)
    begin
        a_fwd = 2'b01;
    end
    else if (wbw && rdw && rs1 == rdw)
    begin
        a_fwd = 2'b10;
    end
    else
    begin
        a_fwd = 2'b00;
    end

    if (wbm && rdm && rs2 == rdm)
    begin
        b_fwd = 2'b01;
    end
    else if (wbw && rdw && rs2 == rdw)
    begin
        b_fwd = 2'b10;
    end
    else
    begin

```



```

add #(32) add_4(pc, 32'h4, pc_4);
mux2 #(32) mux_pc(pc_4, pc_target, PCSrc, pcin);

```

```

program_counter MY_program_counter (
    .clk          ( clk          ),
    .rst          ( rst          ),
    .en           ( ~fstall      ),
    .in           ( pcin         ),
    .out          ( PC           )
);

```

```

mem_ins mem_ins(pc[9:2], ins);

```

```

IF_ID MY_IF_ID (
    .clk          ( clk          ),
    .rst          ( rst | dflush ),
    .en           ( ~dstall      ),
    .pc_4_d_i     ( pc_4        ),
    .pcd_i        ( pc_4_d      ),
    .ir_i         ( pc          ),

    .pc_4_d_o     ( pcd         ),
    .pcd_o        ( ins         ),
    .ir_o         ( ir          )
);

```

```

shift_left #(32) shifter(imm, offset);
add #(32) add_pc(pce, offset, pc_target);
mux3 #(32) mux_a(a, y, wd, a_fwd, alu_a);
mux3 #(32) mux_b(b, y, wd, b_fwd, alu_i);
mux2 #(32) mux_alu(alu_i, imm, ctrl[4], alu_b);
alu #(32) alu(alu_a, alu_b, ctrl[3:0], ALUresult, Zero);

EX_MEM EX_MEM(clk, rst, 1'b1, pc_4_e, pc_4_m, ALUresult, y, alu_i, bm, rd, rdm, ctrl, ctrlm);

mem_data mem_data(y[9:2], bm, m_rf_addr, clk, ctrlm[12], data, m_data);
mux2 #(32) mux_io(data, io_din, y[10], mdr_i);

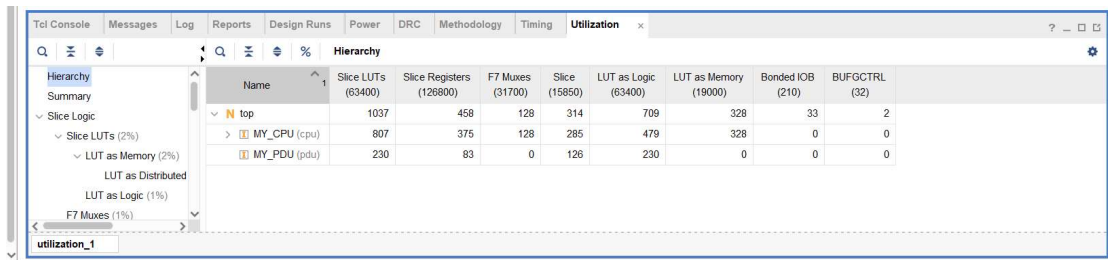
MEM_WB MEM_WB(clk, rst, 1'b1, pc_4_m, pc_4_w, y, yw, mdr_i, mdr, rdm, rdw, ctrlm, ctrlw);

mux3 #(32) mux_wd(yw, mdr, pc, ctrlw[17:16], wd);

endmodule

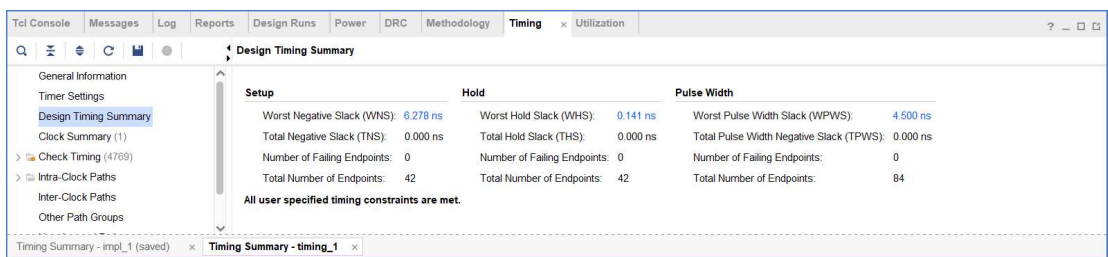
```

6.资源分析



Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	Slice (15850)	LUT as Logic (63400)	LUT as Memory (19000)	Bonded IOB (210)	BUFGCTRL (32)
top	1037	458	128	314	709	328	33	2
MY_CPU (cpu)	807	375	128	285	479	328	0	0
MY_PDU (pdu)	230	83	0	126	230	0	0	0

7.时间性能分析



Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.278 ns	Worst Hold Slack (WHS): 0.141 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 42	Total Number of Endpoints: 42	Total Number of Endpoints: 84

All user specified timing constraints are met.

8.结果分析

在 FPGA 开发板上能成功运行给定的 coe 文件读取输入，产生输出。

【实验总结】

深入了解 riscv cpu 流水线设计