lab6_多周期 mips-cpu 设计

姓名：姜庆彩

学号：PB15051087

- 实验要求

  设计 CPU，完成以下程序代码的执行，其功能是起始数为 3 和 3 的斐波拉契数列的计算。只计算 20 个数。

- 实验说明

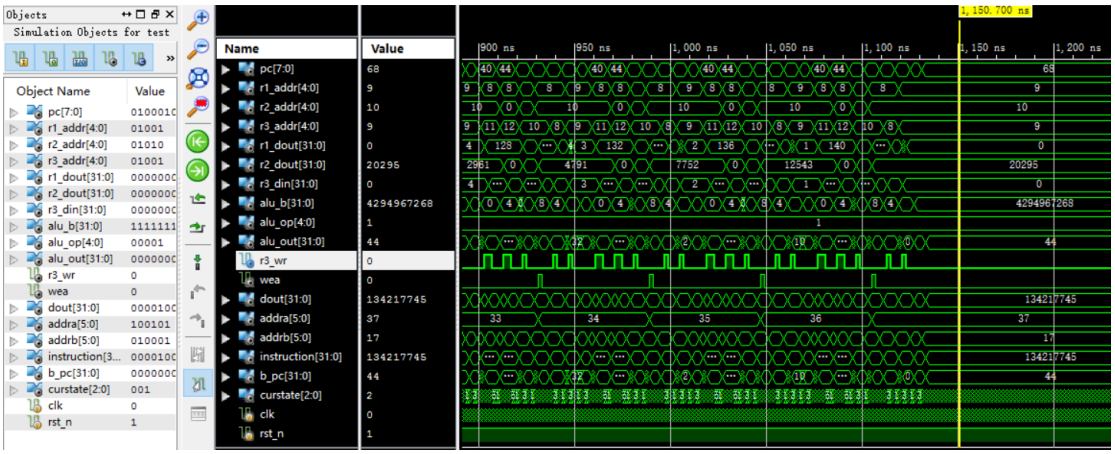
  实验设计中可以不使用给定的数据通路和状态机，但仅允许使用一个存储器。

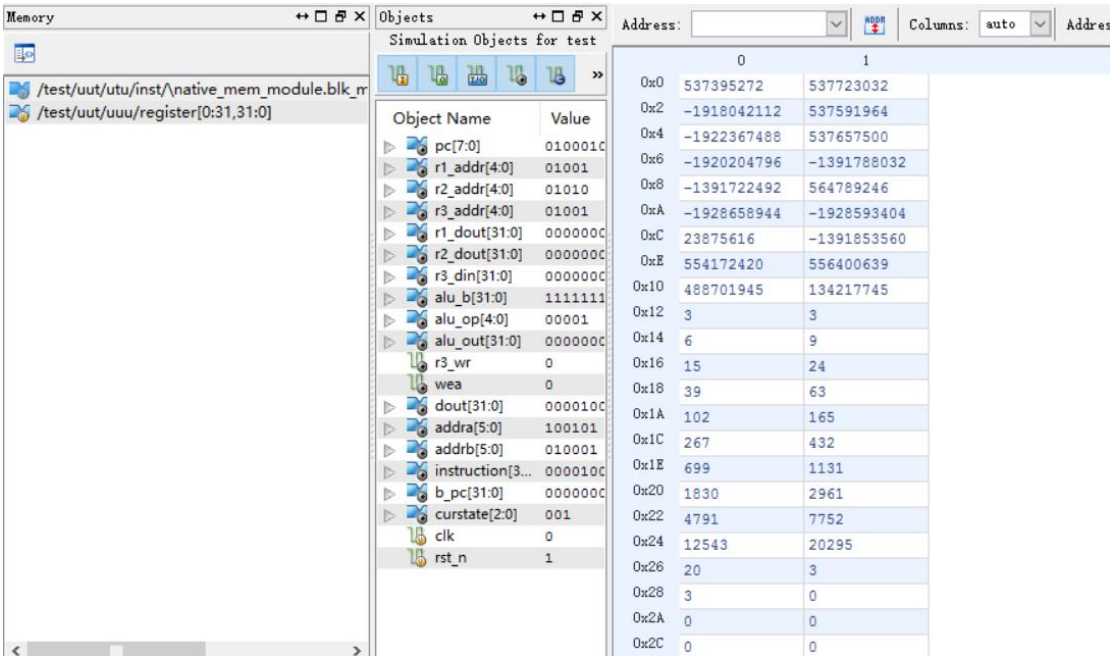
  对指令/数据存储器的附加要求：

  使用异步存储器，最高评分为√ √

  使用同步存储器，最高评分为√ √ √，使用同步存储器时，需要对数据通路和状态机进行适当修改。

- 实验结果：





- 这次实验的实现主要是单周期的基础上加了一个状态机，执行多周期的状态切换，其余与前一个实验十分类似。
- 源代码：

```
module top(
input clk,
input rst_n,
output signed [7:0] pc,
output [4:0] r1_addr,
output [4:0] r2_addr,
output [4:0] r3_addr,
output [31:0] r1_dout,
output [31:0] r2_dout,
output [31:0] r3_din,
output [31:0] alu_b,
output [4:0] alu_op,
output [31:0] alu_out,
output  r3_wr,
output wea,
output [31:0] dout,
output [5:0] addra,
output [5:0] addrb,
output [31:0] instruction,
output [31:0] b_pc,
output [2:0] curstate
    );
wire [31:0] alu_a;
REG_FILE uuu(
.clk(clk),
.rst_n(rst_n),
.r1_addr(r1_addr),
.r2_addr(r2_addr),
.r3_addr(r3_addr),
.r3_din(r3_din),
.r3_wr(r3_wr),
.r1_dout(r1_dout),
.r2_dout(r2_dout)
);
alu ttt(
.alu_a(alu_a),
.alu_b(alu_b),
.alu_op(alu_op),
.alu_out(alu_out)
);
control utt(
.clk(clk),
.rst_n(rst_n),
.dout(dout),
```

```
    .r1_dout(r1_dout),
    .r2_dout(r2_dout),
    .alu_out(alu_out),
    .instruction(instruction),
    .pc(pc),
    .r1_addr(r1_addr),
    .r2_addr(r2_addr),
    .r3_addr(r3_addr),
    .r3_din(r3_din),
    .wea(wea),
    .addrb(addrb),
    .addra(addra),
    .alu_a(alu_a),
    .alu_b(alu_b),
    .alu_op(alu_op),
    .r3_wr(r3_wr),
    .b_pc(b_pc),
    .curstate(curstate)
    );
    id_memory utu(
    .clka(clk),
    .wea(wea),
    .addra(addra),
    .dina(r2_dout),
    .clkb(clk),
    .addrb(addrb),
    .doutb(dout)
    );
endmodule
```

```
module REG_FILE(
input  clk,
input  rst_n,
input [4:0] r1_addr,
input [4:0] r2_addr,
input [4:0] r3_addr,
input [31:0] r3_din,
input  r3_wr,
output reg [31:0] r1_dout,
output reg [31:0]   r2_dout
   );
reg [31:0] register[0:31]; integer k,i;
always@(posedge clk or negedge rst_n)
begin
 if(~rst_n)    //初始化
 begin
   for(k=0;k<32;k=k+1)
     register[k]=32'b0;
 end
 else if(r3_wr)  begin
   i=r3_addr;
   register[i]=r3_din;
   i=r1_addr;
   r1_dout=register[i];
   i=r2_addr;
   r2_dout=register[i];
 end
 else
 begin
   i=r1_addr;
   r1_dout=register[i];
   i=r2_addr;
   r2_dout=register[i];

end



endmodule
```

```
    module alu(
    input signed [31:0] alu_a,
    input signed [31:0] alu_b,
    input [4:0]  alu_op,
    output reg [31:0] alu_out
        );
        always@(*)
        begin
        case(alu_op)
        A_ADD:alu_out=alu_a+alu_b;
        A_SUB:alu_out=alu_a-alu_b;
        A_AND:alu_out=alu_a&alu_b;
        A_OR:alu_out=alu_a|alu_b;
        A_XOR:alu_out=alu_a^alu_b;
        A_NOR:alu_out=~(alu_a|alu_b);
        default: alu_out=32'h0;
        endcase
        end

endmodule
```

```
    module control(
    input clk,
    input rst_n,
    input [31:0] dout,
    input [31:0] r1_dout,
    input [31:0] r2_dout,
    input [31:0] alu_out,
    output reg [31:0] instruction,
    output reg signed [7:0] pc,
    output reg [4:0] r1_addr,
    output reg [4:0] r2_addr,
    output reg [4:0] r3_addr,
    output reg [31:0] r3_din,
    output reg wea,
    output reg [5:0] addrb,
    output reg [5:0] addra,
    output reg [31:0] alu_a,
    output reg [31:0] alu_b,
    output reg [4:0] alu_op,
    output reg r3_wr,
    output reg signed [31:0] b_pc,
    output reg [2:0] curstate
    );
    parameter init=3'b00, fi=3'b01,id=3'b10, ex=3'b11,
me=3'b100, wb=3'b101;
    reg [2:0] nextstate;
    reg flag;
    reg signed [31:0] judge;
    reg signed [31:0] offset;
    reg signed [31:0] b_offset;
    reg [7:0] next_pc;
    reg [27:0] j_pc;
    always@(posedge clk or negedge rst_n)
    begin
      if(~rst_n)
         curstate<=init;
      else
        curstate<=nextstate;
    end
    always@(*)
    begin
      case(curstate)
      init:nextstate=fi;
      fi:nextstate=id;
```

```
      id:
      begin
         if(instruction[31:26]==6'b000010)
nextstate=fi;
         else nextstate=ex;
      end
      ex:
      begin

   if(instruction[31:26]==6'b001000||instruction[31:26
]==6'b000000) //add or addi
            nextstate=wb;
         else
if(instruction[31:26]==6'b100011||instruction[31:26]==
6'b101011)//lw or sw
            nextstate=me;
         else if(instruction[31:26]==6'b000111)  //bgtz
            nextstate=fi;
      end
      me:
      begin
         if(instruction[31:26]==6'b100011) //lw
            nextstate=wb;
         else if(instruction[31:26]==6'b101011) //sw
            nextstate=fi;
      end
      wb: nextstate=fi;
      endcase
   end
   always@(*)
   begin
     if(~rst_n) next_pc=0;
      b_pc=alu_out;
     j_pc=instruction[25:0]<<2;
   end
   always@(*)
   begin
     if(~rst_n)
     begin
      r3_wr=0;
      r1_addr=0;
      r2_addr=0;
      judge=0;
     end
```

```
      else
      begin
          if(instruction[15]==1)

   offset={16'b11111_1111_1111_1111,instruction[15:0]}
;
          else
offset={16'b0000_0000_0000_0000,instruction[15:0]};
       case(curstate)
       fi:
       begin
          if(instruction[31:26]==6'b000010)
pc=j_pc[7:0];
         else if(instruction[31:26]==6'b000111&&judge>0)
pc=b_pc[7:0];
                addrb=pc>>2;
          wea=0;
          r3_wr=0;
       end
       id:
       begin
          instruction=dout;
          if(instruction[31:26]==6'b001000)        begin
             alu_a=r1_dout;
             alu_b=offset;
             r1_addr=instruction[25:21];
             alu_op=5'h01;
             r3_wr=0;
          end
          else if(instruction[31:26]==6'b000000)    //add
          begin
             r1_addr=instruction[25:21];
             r2_addr=instruction[20:16];
             r3_addr=instruction[15:11];
             alu_a=r1_dout;
             alu_op=5'h01;
             r3_wr=0;
          end
          else if(instruction[31:26]==6'b100011)   //lw
          begin
             r1_addr=instruction[25:21];

             alu_op=5'h01;
             r3_addr=instruction[20:16];
```

```
            end
            else if(instruction[31:26]==6'b101011)   //sw
            begin
               r1_addr=instruction[25:21];
               r2_addr=instruction[20:16];
               alu_op=5'h01;
               r3_wr=0;
            end
            else if(instruction[31:26]==6'b000111) //bgtz
            begin
               r1_addr=instruction[25:21];
               judge=r1_dout;

               alu_a=next_pc;
               alu_b=b_offset;
            end
         end
         ex:
         begin

      if(instruction[31:26]==6'b001000||instruction[31:26
]==6'b000000)  //add or addi
            begin
               r3_din=alu_out;
               r3_wr=1;
            end

               r3_wr=1;
            end
            else if(instruction[31:26]==6'b101011) //sw

         me:
         begin
            wea=0;
            r3_din=dout;
         end
         wb:r3_wr=0;
         endcase
        end
      end

      endmodule
```