数电实验报告

1. 题目要求

完成 XADC 对板上电位器产生的电压的采集。完成 1）或 2），选择完成 3）。
1）将采集结果通过 Led 灯显示，当电压为 1V 时，输出为 1111_1111_1111，当电压为 0.5V 时输出为 0111_1111_1111.
2）将输出显示到数码管上，当电压为 1V 时，显示 0FFF，当电压为 0V 时，显示 0000.
3）将二进制输出转换为相应电压值显示到数码管上（选作）。

2. 源代码

1) Xadc.v

```
module xadc_top(
    input clk,
    input rst_n,
    input vauxp1,vauxn1,
    output [11:0]led,
    output [6:0]a_to_g,
    output [3:0]an
    );

    wire [15:0]do_out;
    wire drdy_out;

    xadc_channel1 U0(
    .dclk_in(clk),
    .reset_in(rst_n),
    .vauxp1(vauxp1),
    .vauxn1(vauxn1),
    .do_out(do_out),
    .drdy_out(drdy_out)
    );

    xadc_result U3(
   .clk(clk),
   .rst_n(rst_n),
   .do_out(do_out),
   .drdy_out(drdy_out),
   .led(led)
    );

    wire [3:0]Y2;
    wire [3:0]Y1;
```

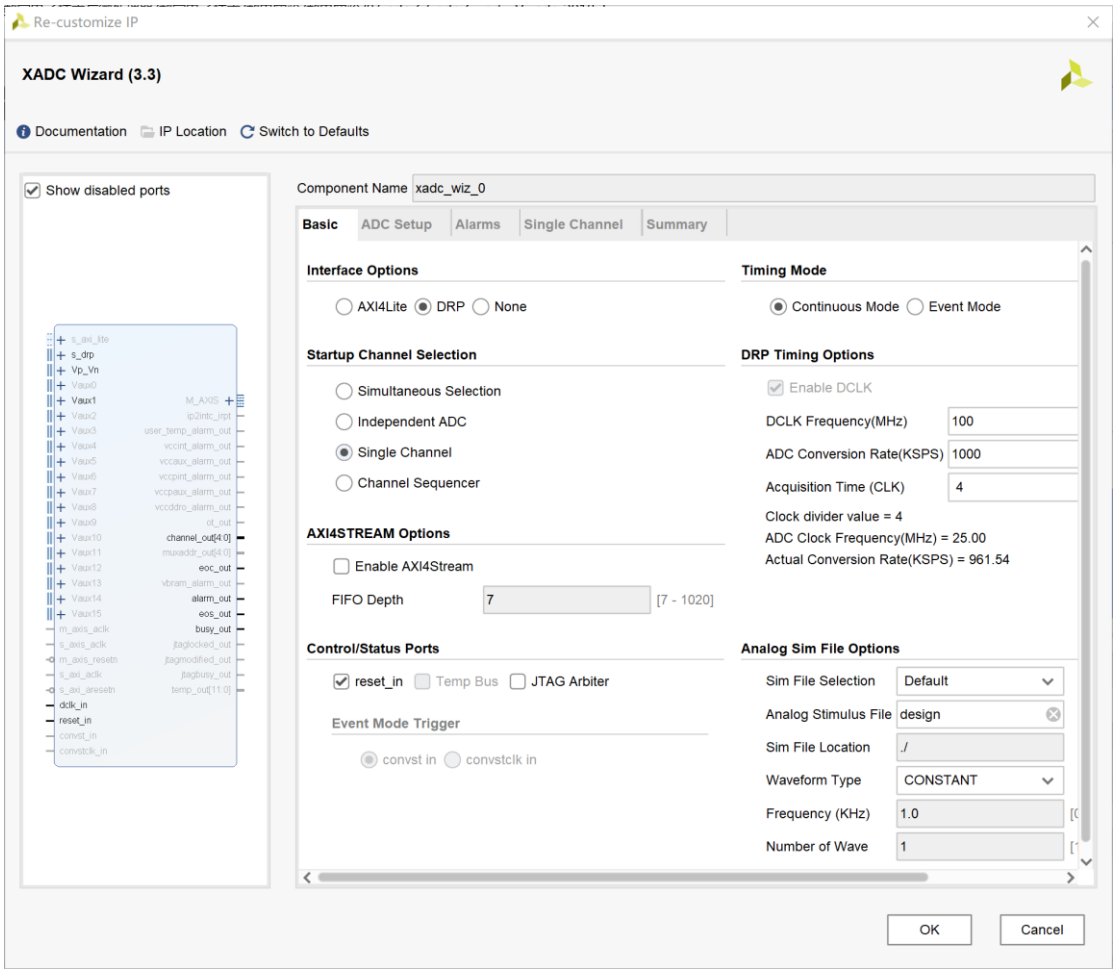
```
    wire [3:0]Y0;

    xadc_BCD U4(
    .led(led),
    .clk(clk),
    .rst_n(rst_n),
    .Y2(Y2),
    .Y1(Y1),
    .Y0(Y0)
    );

    my_x7seg_4bit U5(
    .clk(clk),
    .rst_n(rst_n),
    .x3(0),
    .x2(led[11:8]),
    .x1(led[7:4]),
    .x0(led[3:0]),

    .a_to_g0(a_to_g),
    .an(an)
    );

Endmodule
```

2) xadc_channel1.v

```
module xadc_channel1(
    input dclk_in,
    input reset_in,
    input vauxp1,vauxn1,
    output [15:0]do_out,
    output drdy_out
    );

    wire eoc_out;

xadc_wiz_0 U00 (
  .di_in(16'b0),              // input wire [15 : 0] di_in
  .daddr_in(7'h11),         // input wire [6 : 0] daddr_in
  .den_in(eoc_out),            // input wire den_in
  .dwe_in(1'b0),             // input wire dwe_in
  .drdy_out(drdy_out),        // output wire drdy_out
  .do_out(do_out),            // output wire [15 : 0] do_out
  .dclk_in(dclk_in),          // input wire dclk_in
```

```
  .reset_in(reset_in),         // input wire reset_in
  .vp_in(),                // input wire vp_in
  .vn_in(),                // input wire vn_in
  .vauxp1(vauxp1),             // input wire vauxp1
  .vauxn1(vauxn1),             // input wire vauxn1
  .channel_out(),  // output wire [4 : 0] channel_out
  .eoc_out(eoc_out),           // output wire eoc_out
  .alarm_out(),       // output wire alarm_out
  .eos_out(),           // output wire eos_out
  .busy_out()         // output wire busy_out
);
endmodule
```

3) IP 核



4) xadc_result.v

```
module xadc_result(
    input clk,
    input rst_n,

    input [15:0]do_out,
    input drdy_out,
```

```
    output reg [11:0]led
    );

    always@(posedge clk or negedge rst_n)
    begin
        if(rst_n==0)
        begin
            led<=12'b0;
        end
        else
        begin
            led<= do_out[15:4];
        end
    end

endmodule
```

5) xadc_BCD.v

```
module xadc_BCD(
    input [11:0]led,
    input clk,
    input rst_n,
    output reg [3:0]Y2,
    output reg [3:0]Y1,
    output reg [3:0]Y0
    );

    integer cnt=0;
    always@*
    begin
        if(rst_n==0)
        begin
            Y2=0;
            Y1=0;
            Y0=0;
        end
        else
        begin
            cnt=led;
            Y0=cnt%10;
            cnt=cnt/10;
            Y1=cnt%10;
            cnt=cnt/10;
            Y2=cnt%10;
```

```
        end
    end

endmodule
```

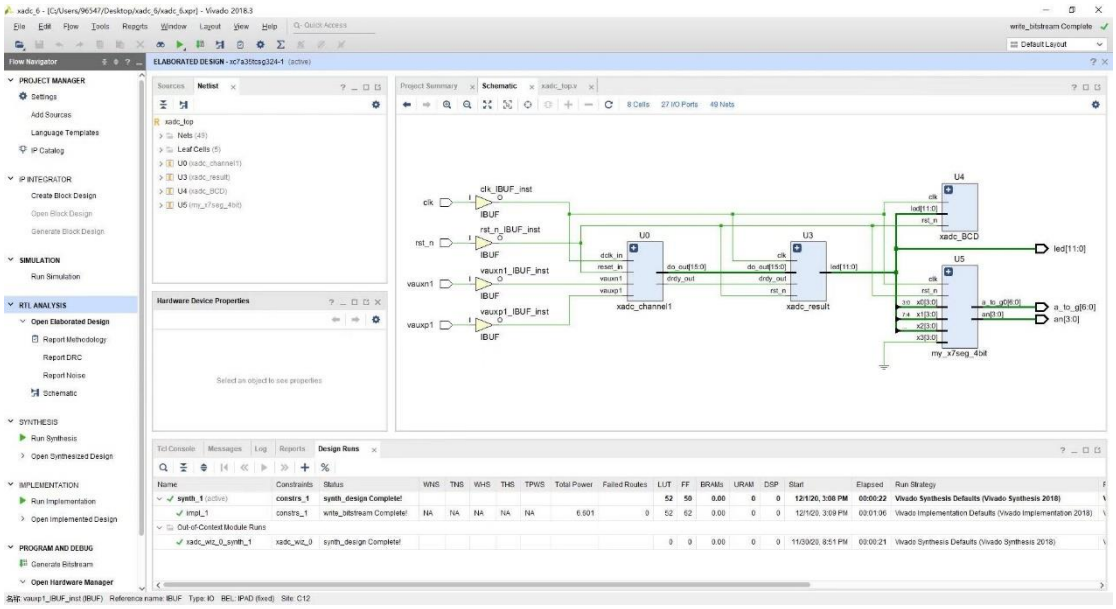
6) my_x7seg_4bit.v

```
module my_x7seg_4bit(
    input clk,
    input rst_n,

    input [3:0]x3,
    input [3:0]x2,
    input [3:0]x1,
    input [3:0]x0,

    output reg [6:0]a_to_g0,
    output reg [3:0]an
    );

    integer clkdiv;
    always@(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            clkdiv<=0;
        else if(clkdiv==500000)
            clkdiv<=0;
        else
            clkdiv<=clkdiv+1;
    end

    reg [1:0]bitcnt;
    always@(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            bitcnt<=0;
        else if(clkdiv==500000)
            bitcnt<=bitcnt+1;
    end

    always@*
    begin
        if(!rst_n)
            an=0;
        else
        begin
```

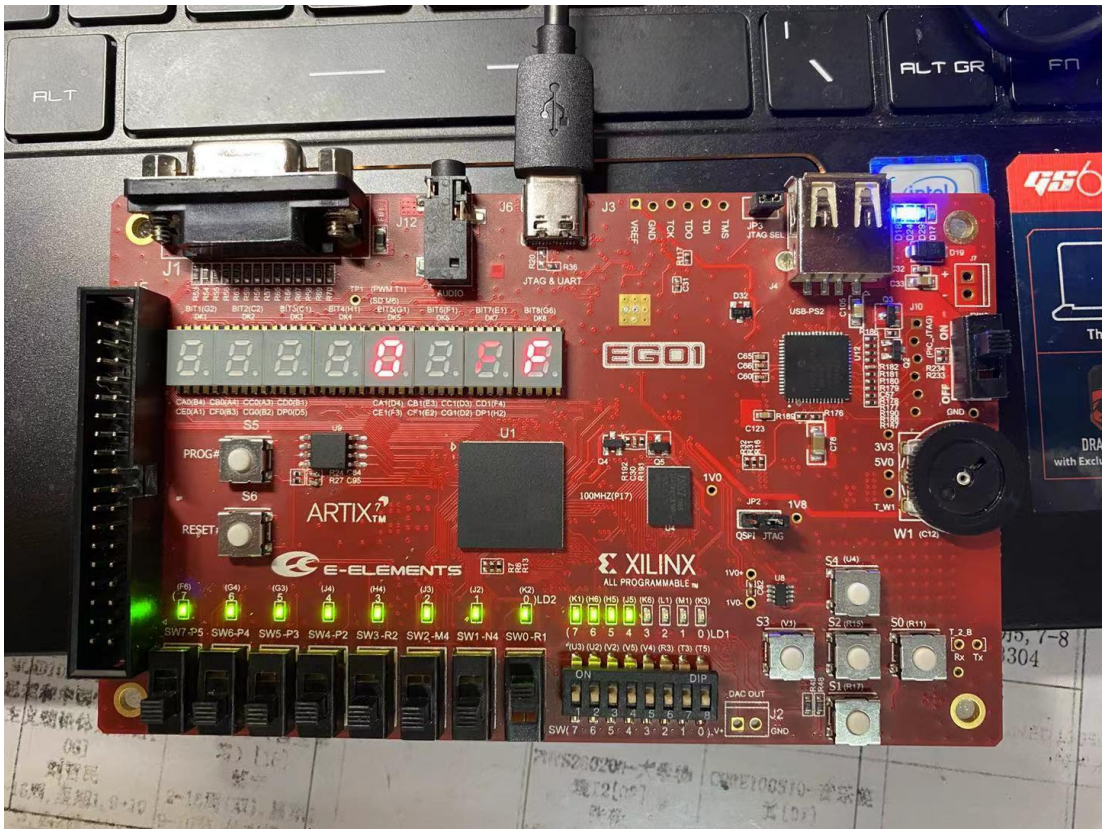
```
        an=4'd0;
        an[bitcnt]=1;
    end
end

reg[3:0]digit0;
always@(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        digit0<=0;
    else
        case(bitcnt)
            2'd0:digit0<=x0[3:0];
            2'd1:digit0<=x1[3:0];
            2'd2:digit0<=x2[3:0];
            2'd3:digit0<=x3[3:0];
        endcase
end

always@*
begin
if(!rst_n)
begin
    a_to_g0<=7'b1111111;
end
else
    case(digit0)
    0:a_to_g0<=7'b1111110;
    1:a_to_g0<=7'b0110000;
    2:a_to_g0<=7'b1101101;
    3:a_to_g0<=7'b1111001;
    4:a_to_g0<=7'b0110011;
    5:a_to_g0<=7'b1011011;
    6:a_to_g0<=7'b1011111;
    7:a_to_g0<=7'b1110000;
    8:a_to_g0<=7'b1111111;
    9:a_to_g0<=7'b1111011;
    10:a_to_g0<=7'b1110111;
    11:a_to_g0<=7'b1111111;
    12:a_to_g0<=7'b1001110;
    13:a_to_g0<=7'b1111110;
    14:a_to_g0<=7'b1001111;
    15:a_to_g0<=7'b1000111;
    default:a_to_g0<=7'b1111110;
```

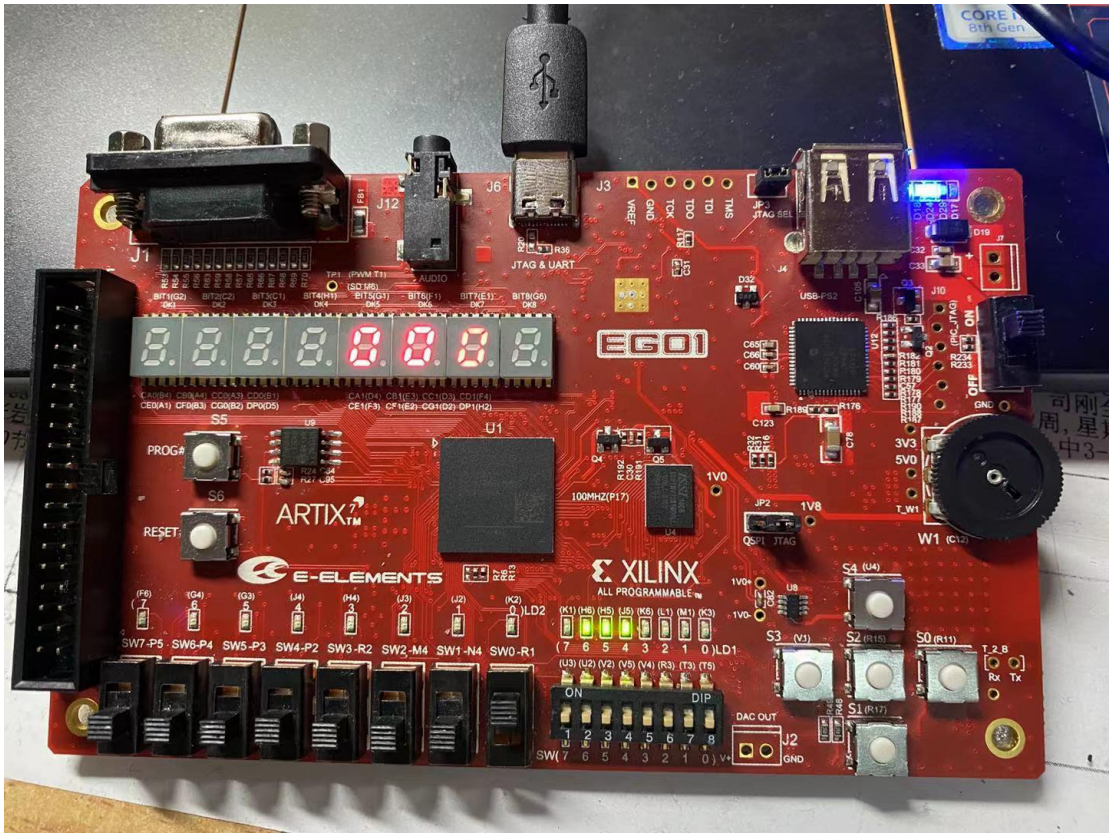
```
            endcase
        end

endmodule
```

3. RTL 分析



4. 开发板结果




EGO1
E-ELEMENTS
XILINX
ALL PROGRAMMABLE
ARTIX
JTAG & UART
AUDIO
USB-PS2
100MHZ(P17)
SW7-P5 SW6-P4 SW5-P3 SW4-P2 SW3-R2 SW2-M4 SW1-N4 SW0-R1
PROG
RESET
DAC OUT
QSPI JTAG
3V3
5V0
1V0
1V8
GND