# 数电实验报告

## 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）。

## 源代码

### 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

### 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

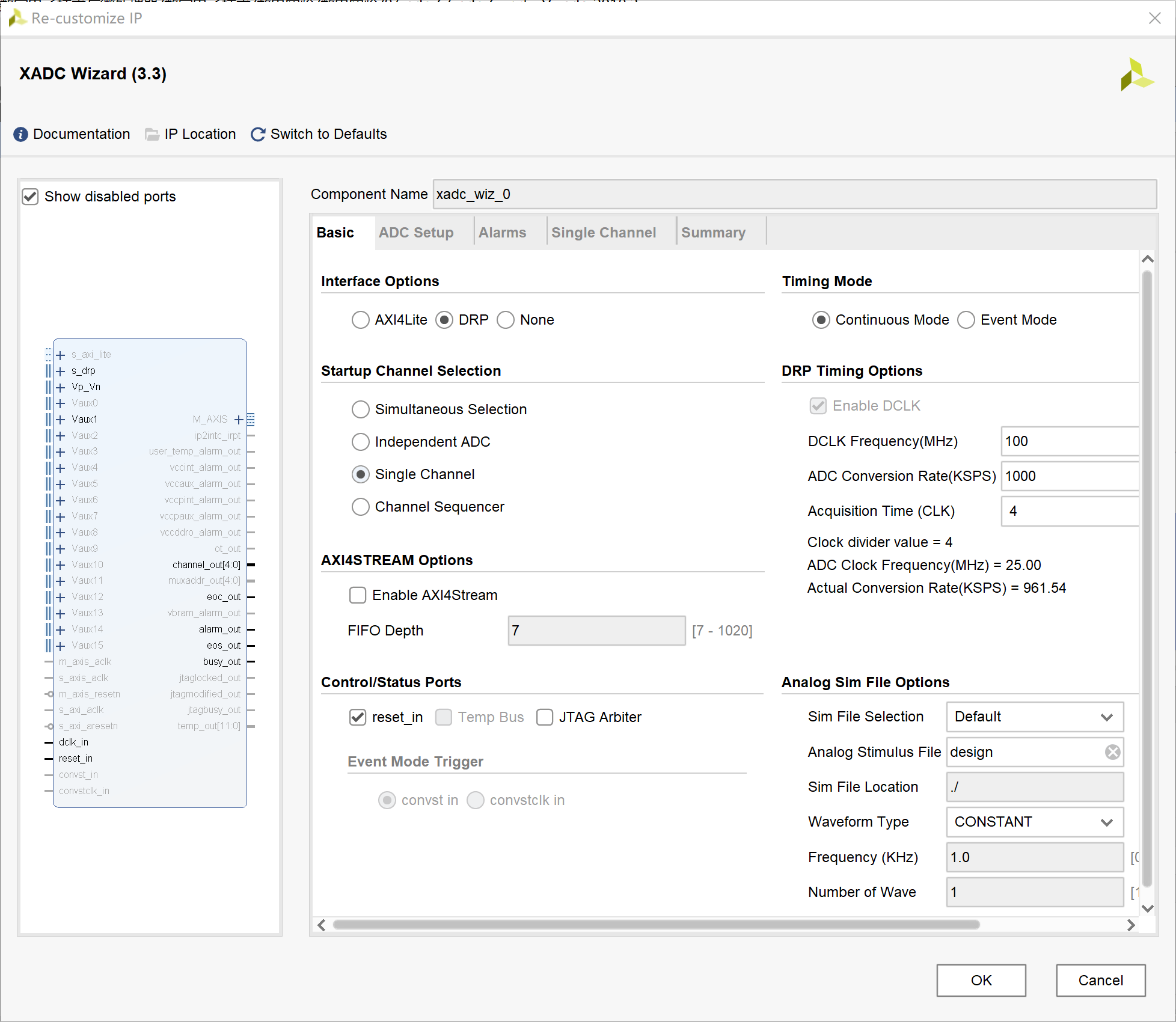
.eos\_out(), // output wire eos\_out

.busy\_out() // output wire busy\_out

);

endmodule

### IP核



### 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

### 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

### 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;

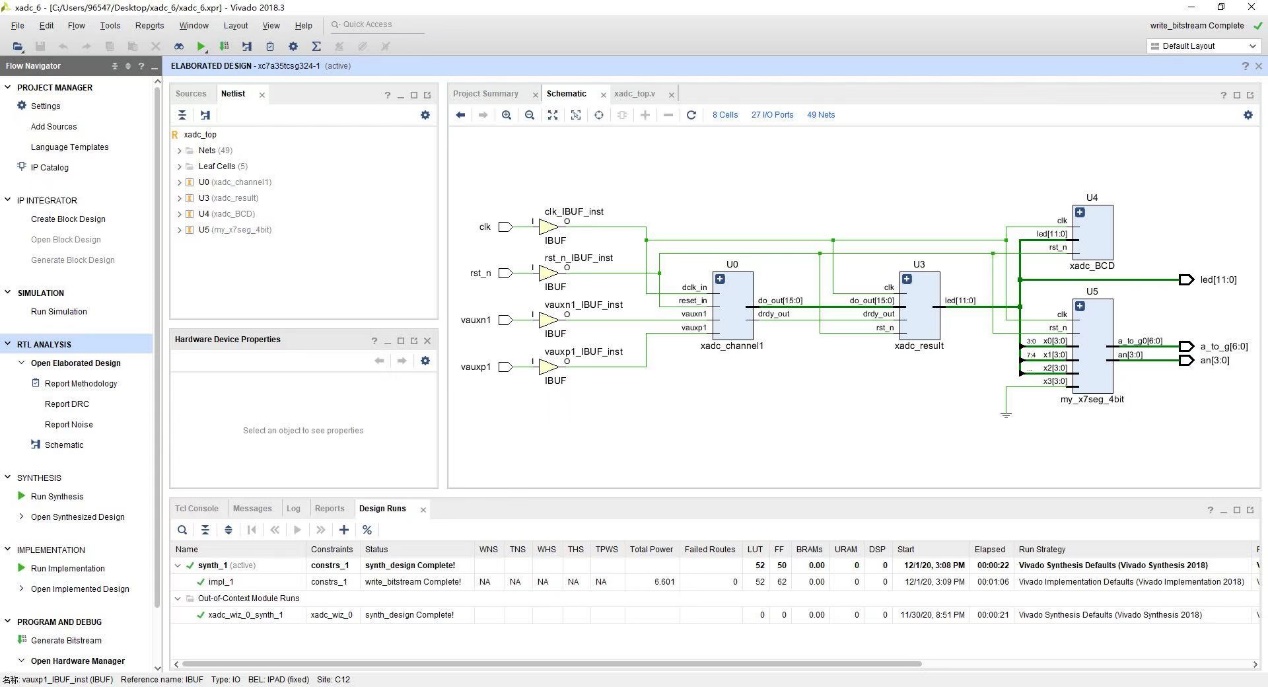
default:a\_to\_g0<=7'b1111110;

endcase

end

endmodule

## RTL分析



## 开发板结果

