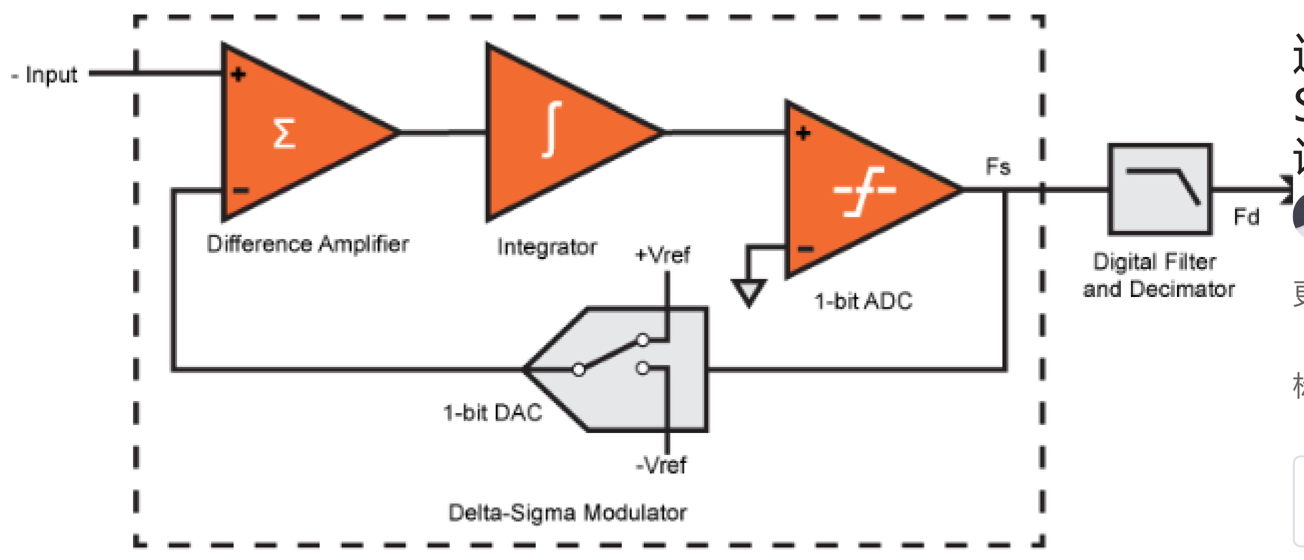


通过高速比较器和FPGA逻辑实现Sigma Delta ADC - 基于电赛综合训练板/小脚丫FPGA

子曰

更新 2021年03月29日

标签 FPGA 测试 ADC

469

基本信息 评论

内容介绍

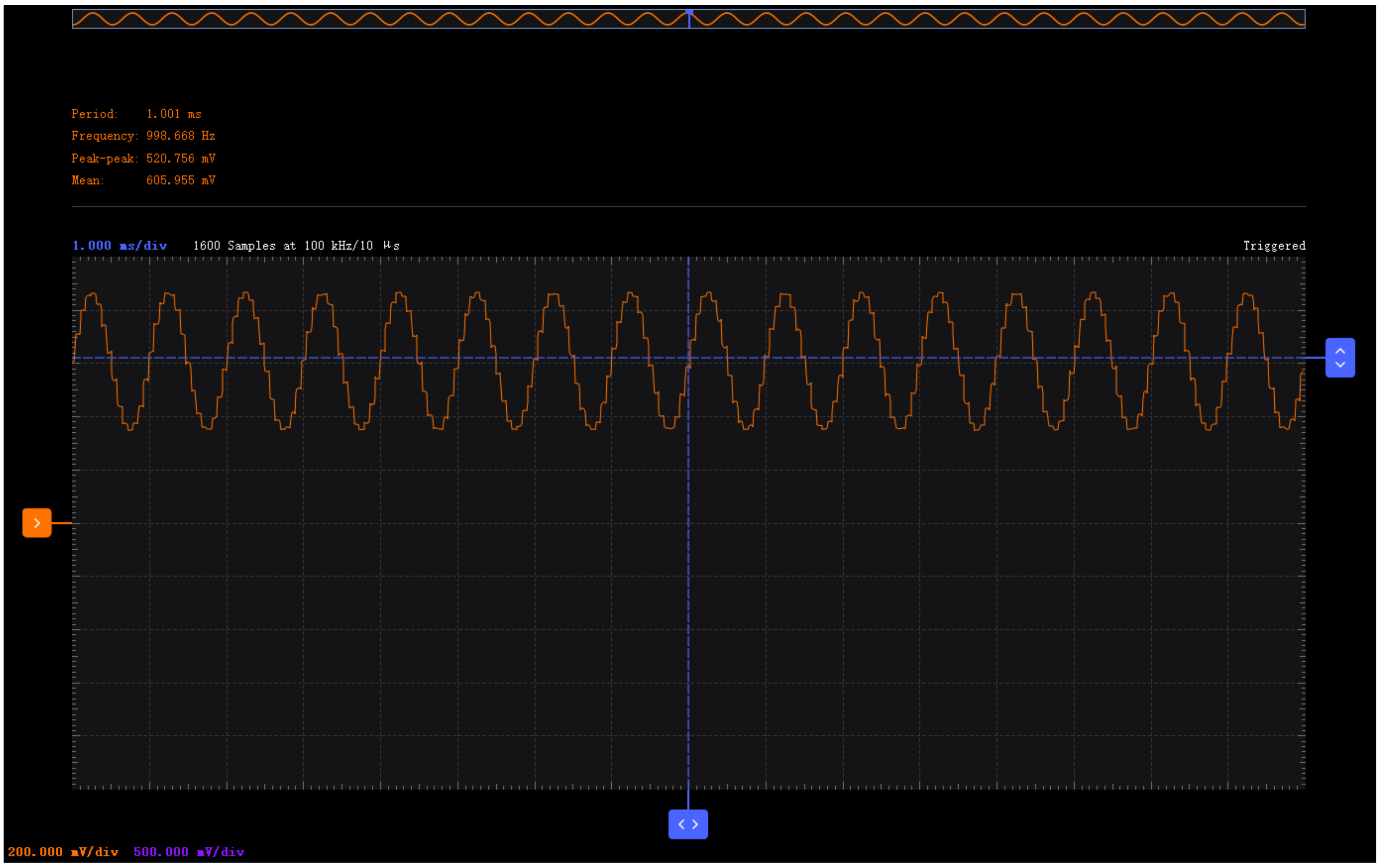
多数FPGA芯片上没有ADC的功能，而一些应用则需要用到ADC对一些模拟信号，比如直流电压等进行量化，有没有特别简单、低成本的实现方法呢？

在要求转换速率不高的情况下，完全可以借助一颗高速比较器（成本只有几毛钱）来实现对模拟信号的量化，Lattice的官网上一篇文章就介绍了如何制作一个简易的Sigma Delta ADC，如果FPGA能够提供LVDS的接口，连外部的高速比较器都可以省掉。由于我们的小脚丫FPGA核心模块在设计的时候没有考虑到LVDS的应用场景，所以还是需要搭配一个高速的比较器来实现Lattice官网上推荐的简易Sigma Delta ADC的功能。

让小脚丫FPGA通过锁相环PLL运行于120MHz的主时钟（还可以更高，提速到240MHz、360MHz都应该没有问题），测试1KHz以内的模拟信号是没有问题的。

Lattice的官网上就可以下载到简易Sigma Delta ADC的Verilog源代码，可以非常方便地用在其它品牌、其它系列的FPGA上。

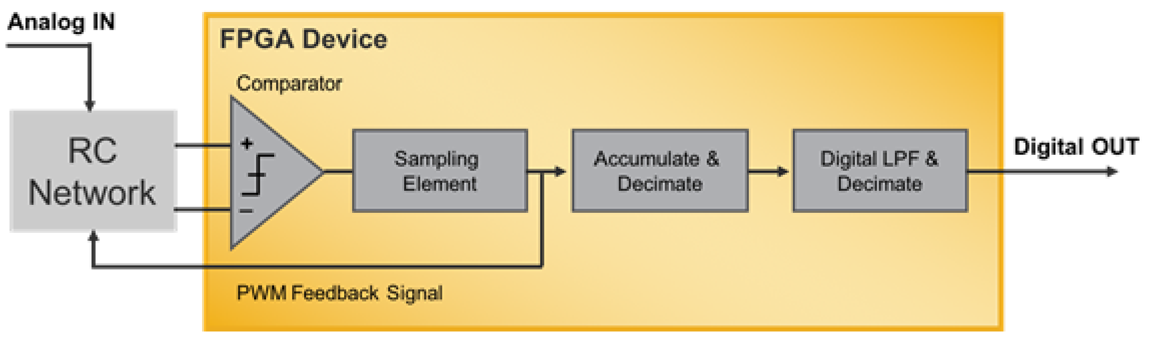
下面的截图就是采用120MHz的主时钟实现的对1KHz模拟信号的采样，并通过DDS/DAC输出的模拟信号波形。



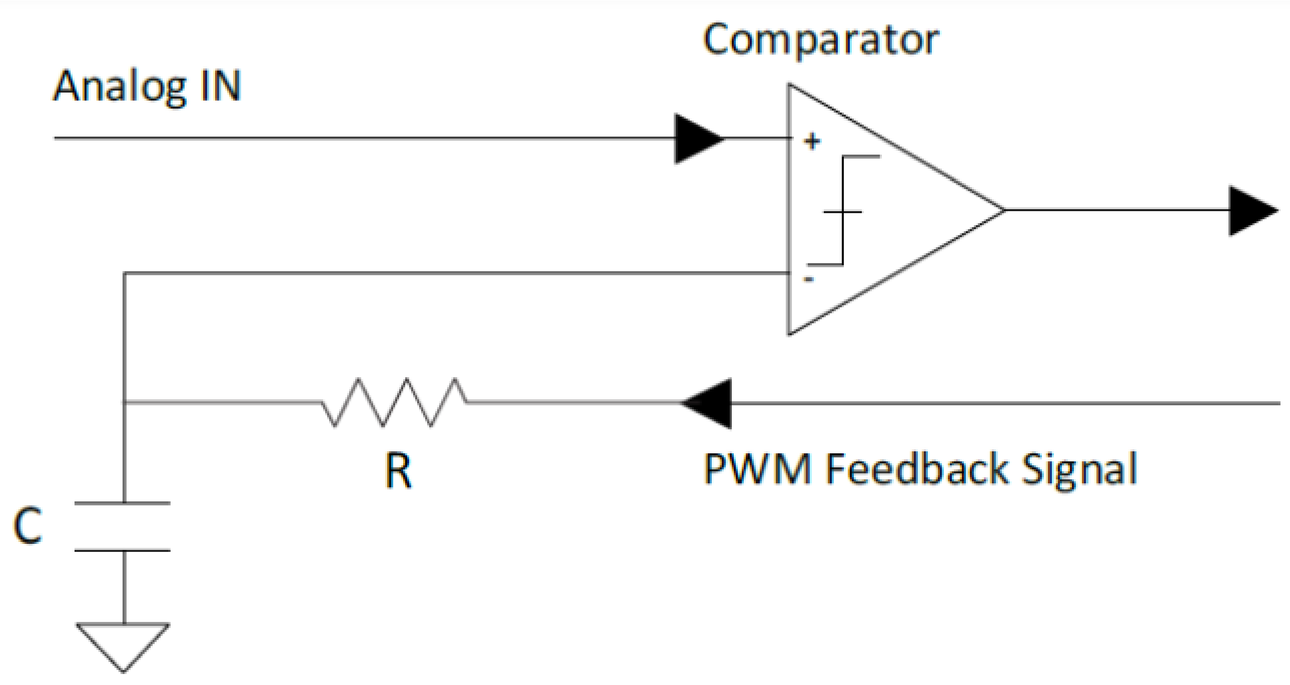
```
Period:    1.001 ms
Frequency: 998.668 Hz
Peak-peak: 520.756 mV
Mean:      605.955 mV
```

```
1.000 ms/div    1600 Samples at 100 kHz/10 µs                Triggered
200.000 mV/div  500.000 mV/div
```

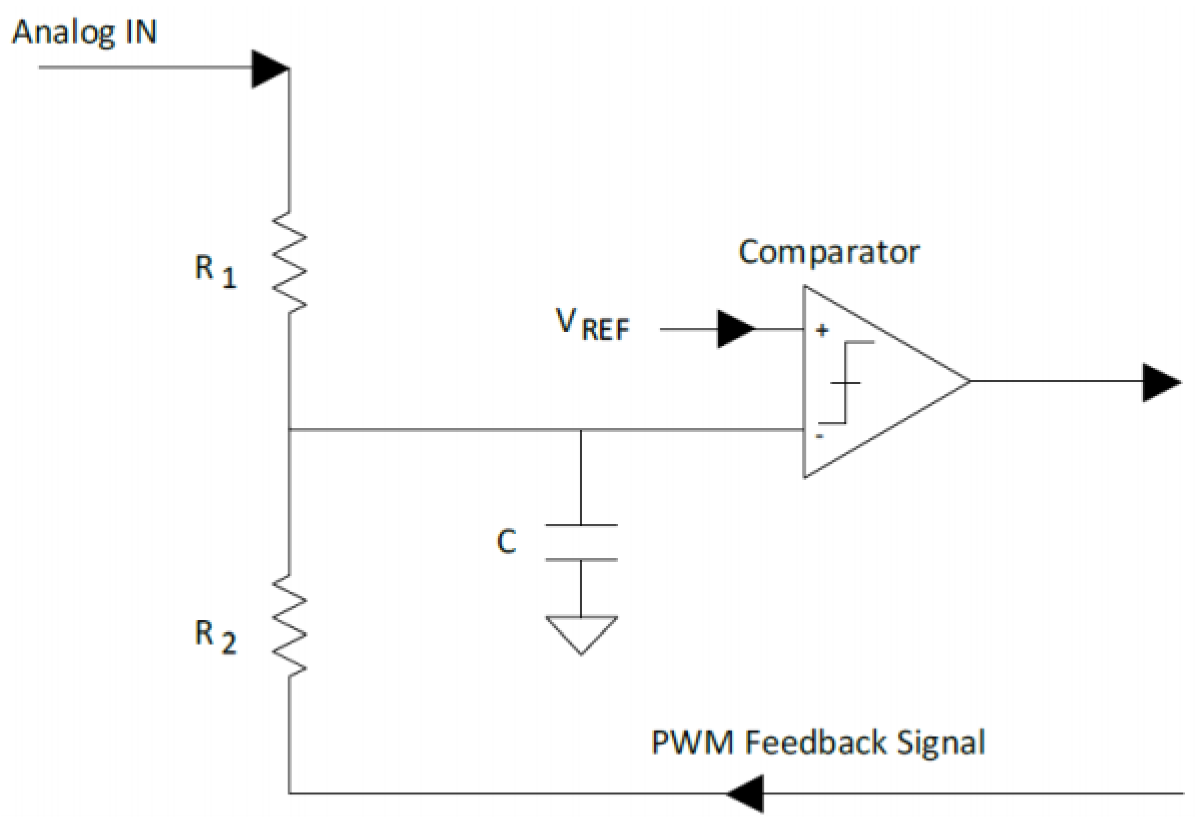
工作原理



简易Sigma Delta ADC的工作原理



直接连接 - 被测模拟信号的幅度范围为0-3.3V



通过电阻分压网络输入，并在比较器+端提供参考电压，则被采集模拟信号的电压变化范围可以扩展

Operation Frequency	Output Sample Rate	Input Frequency (Hz)	8-Bit SSD SNR	8-Bit SSD ENOB1	10-Bit SSD SNR	10-Bit SSD ENOB*	Operation Frequency
62.5 MHz	7.63 KHz	50	47.0	7.12	54.9	8.60	62.5 MHz
62.5 MHz	7.63 KHz	1000	46.7	7.18	52.8	8.25	62.5 MHz
62.5 MHz	7.63 KHz	3800	42.5	6.74	53.1	8.53	62.5 MHz

Note: ENOB = Equivalent Number of Bits of resolution.

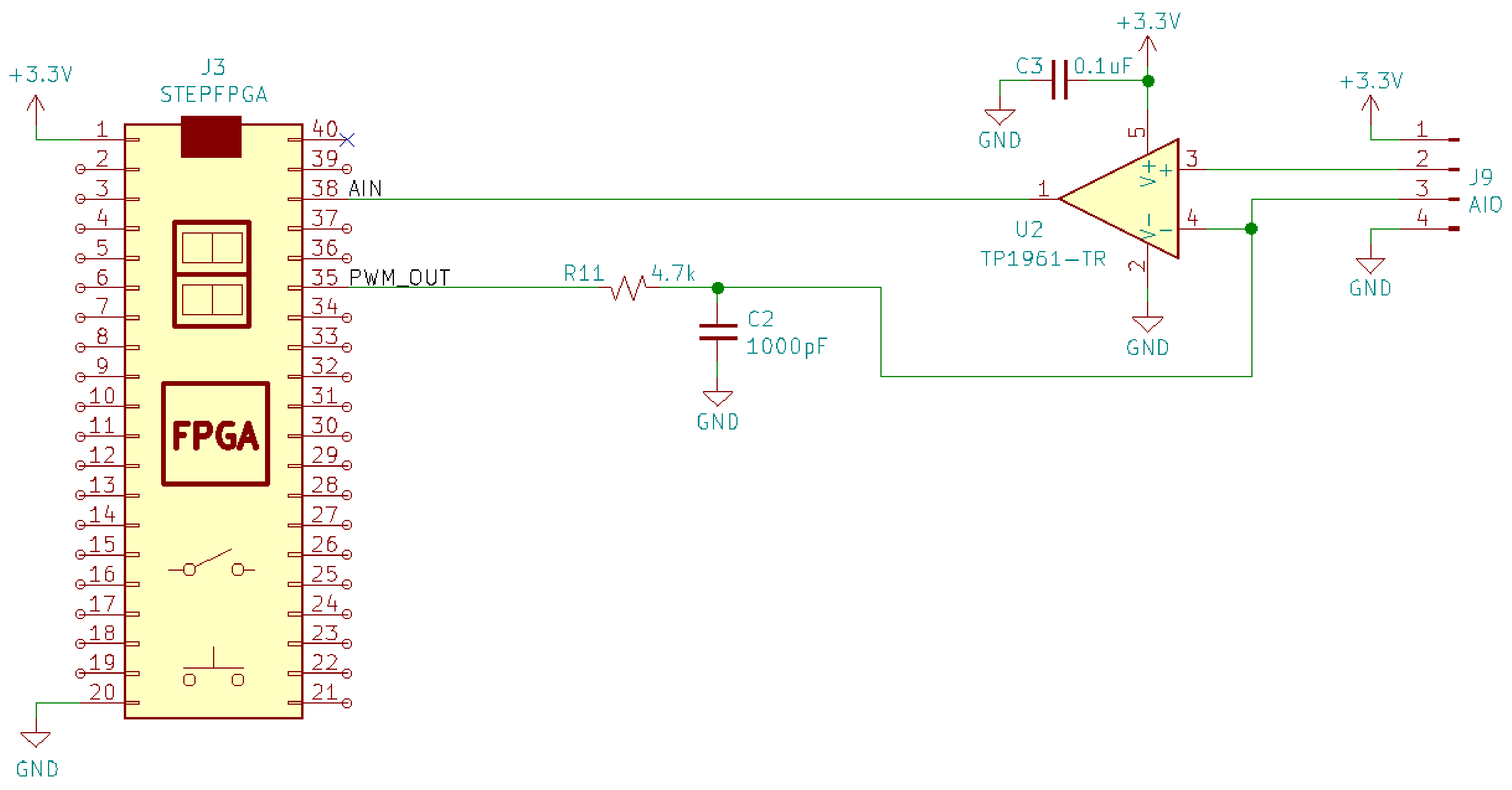
简易Sigma Delta ADC的性能与逻辑电路的工作频率

Table 7.1. Performance and Resource Utilization[1]

Device	Tool/Coding Language	Speed Grade	Utilization (LUTs)	fMAX (MHz)	I/Os	Architecture Resources
iCE40 UltraPlus[5]	Verilog-LSE	NA	99	>150	13	NA
	Verilog-Syn	NA	101	>150	13	NA
MachXO2™ [2]	Verilog-LSE	–6	49	>150	13	NA
	Verilog-Syn	–6	61	>150	13	NA
	VHDL-LSE	–6	49	>150	13	NA
	VHDL-Syn	–6	61	>150	13	NA
MachXO[3]	Verilog-LSE	–5	46	>150	13	NA
	Verilog-Syn	–5	48	>150	13	NA
	VHDL-LSE	–5	46	>150	13	NA
	VHDL-Syn	–5	48	>150	13	NA
LatticeXP2™ [4]	Verilog-Syn	–5	62	>150	13	NA
	VHDL-Syn	–5	62	>150	13	NA

在不同的FPGA平台上消耗的逻辑资源

以下就是我们的电赛综合训练板上简易Sigma Delta ADC部分的电路连接



参考文章：

- The basics of sigma delta analog-to-digital converters
- TEARING INTO DELTA SIGMA ADC’ S

关键代码：

顶层调用代码：

```
wire [7:0] sd_adc_out; // sigma delta adc data output

wire sample_rdy;     // flag for adc conversion

ADC_top my_adc(.clk_in(clk_hs),.rstn(1'b1),.digital_out(sd_adc_out), .analog_cmp(comp_in),.analog_out(ad_pwm),.sample_rdy(

assign dac_data = sd_adc_out;
assign dac_clk = clk_hs; //120MHz generated by PLL
```

Sigma Delta ADC顶层程序

```
//   =================================================================
//   >>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<<<
//   ------------------------------------------------------------------
//   Copyright (c) 2014 by Lattice Semiconductor Corporation
//   ALL RIGHTS RESERVED
//   ------------------------------------------------------------------
//
//   Permission:
//
//      Lattice SG Pte. Ltd. grants permission to use this code
//      pursuant to the terms of the Lattice Reference Design License Agreement.
//
//
//   Disclaimer:
//
//      This VHDL or Verilog source code is intended as a design reference
//      which illustrates how these types of functions can be implemented.
//      It is the user's responsibility to verify their design for
//      consistency and functionality through the use of formal
//      verification methods.  Lattice provides no warranty
//      regarding the use or functionality of this code.
//
//   --------------------------------------------------------------------
//
//                  Lattice SG Pte. Ltd.
//                  101 Thomson Road, United Square #07-02
//                  Singapore 307591
//
//
//                  TEL: 1-800-Lattice (USA and Canada)
//                       +65-6631-2000 (Singapore)
//                       +1-503-268-8001 (other locations)
//
//                  web: http://www.latticesemi.com/
//                  email: techsupport@latticesemi.com
//
//   --------------------------------------------------------------------
//
//  Project:     ADC_lvds
//  File:        adc_top.v
//  Title:       ADC Top Level
//  Description: Top level of Analog to Digital Convertor
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
// $Log: RD#rd1066_simple_sigma_delta_adc#rd1066#source#verilog#adc_top.v,v $
// Revision 1.1  2015-02-05 00:00:56-08  mbevinam
// Updated RD Placed in RD Folder. Previous versions are in RD_Dimensions Archive folder.
//
// Revision   Date
// 1.0        10/12/2009 Initial Revision
//
// --------------------------------------------------------------------



//*********************************************************************
//
```

```
// ADC Top Level Module
//
//***********************************************************************



module ADC_top (
    clk_in,
    rstn,
    digital_out,
    analog_cmp,
    analog_out,
    sample_rdy);

parameter
ADC_WIDTH  =  8,                // ADC Convertor Bit Precision
ACCUM_BITS  =  10,              // 2^ACCUM_BITS is decimation rate of accumulator
LPF_DEPTH_BITS  =  3,           // 2^LPF_DEPTH_BITS is decimation rate of averager
INPUT_TOPOLOGY  =  1;           // 0: DIRECT: Analog input directly connected to + input of comparitor
                                // 1: NETWORK:Analog input connected through R divider to - input of comp.

//input ports
input     clk_in;               // 62.5Mhz on Control Demo board
input     rstn;
input     analog_cmp;           // from LVDS buffer or external comparitor

//output ports
output   analog_out;            // feedback to RC network
output   sample_rdy;
output  [7:0]  digital_out;     // connected to LED field on control demo bd.


//***********************************************************************
//
// Internal Wire & Reg Signals
//
//***********************************************************************
wire                              clk;
wire                              analog_out_i;
wire                              sample_rdy_i;
wire  [ADC_WIDTH-1:0]          digital_out_i;
wire  [ADC_WIDTH-1:0]          digital_out_abs;



assign  clk  =  clk_in;


//************************************************************************
//
//  SSD ADC using onboard LVDS buffer or external comparitor
//
//************************************************************************
sigmadelta_adc  #(
    .ADC_WIDTH(ADC_WIDTH),
    .ACCUM_BITS(ACCUM_BITS),
    .LPF_DEPTH_BITS(LPF_DEPTH_BITS)
    )
SSD_ADC(
    .clk(clk),
     .rstn(rstn),
     .analog_cmp(analog_cmp),
     .digital_out(digital_out_i),
     .analog_out(analog_out_i),
     .sample_rdy(sample_rdy_i)
     );

assign digital_out_abs = INPUT_TOPOLOGY ? ~digital_out_i : digital_out_i;

//***********************************************************************
//
// output assignments
//
//***********************************************************************


assign digital_out    = ~digital_out_abs;     // invert bits for LED display
assign analog_out     =  analog_out_i;
assign sample_rdy     =  sample_rdy_i;

endmodule
```

Sigma Delta ADC主程序

```
//   ==================================================================
//   >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<<<
//   ------------------------------------------------------------------
//   Copyright (c) 2014 by Lattice Semiconductor Corporation
//   ALL RIGHTS RESERVED
//   ------------------------------------------------------------------
//
//   Permission:
//
//      Lattice SG Pte. Ltd. grants permission to use this code
//      pursuant to the terms of the Lattice Reference Design License Agreement.
//
//
//   Disclaimer:
//
//      This VHDL or Verilog source code is intended as a design reference
//      which illustrates how these types of functions can be implemented.
//      It is the user's responsibility to verify their design for
//      consistency and functionality through the use of formal
//      verification methods.  Lattice provides no warranty
//      regarding the use or functionality of this code.
//
//   --------------------------------------------------------------------
//
//                  Lattice SG Pte. Ltd.
//                  101 Thomson Road, United Square #07-02
//                  Singapore 307591
//
//
//                  TEL: 1-800-Lattice (USA and Canada)
//                       +65-6631-2000 (Singapore)
//                       +1-503-268-8001 (other locations)
//
//                  web: http://www.latticesemi.com/
//                  email: techsupport@latticesemi.com
//
//   --------------------------------------------------------------------
// --------------------------------------------------------------------
//
//  Project:     Simple Sigma Delta (SSD)
//  File:        sigmadelta_adc.v
//  Title:       SSD Top Level
//  Description: Top level of SSD Analog to Digital Convertor
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
// $Log: rd1066#source#verilog#sigmadelta_adc.v,v $
// Revision 1.1  2015-02-05 00:00:52-08  mbevinam
// Updated RD Placed in RD Folder. Previous versions are in RD_Dimensions Archive folder.
//
// Revision   Date
// 1.0        10/12/2009 Initial Revision
//
// --------------------------------------------------------------------



//*********************************************************************
```

```
//
// SSD Top Level Module
//
//***************************************************************************



module sigmadelta_adc (
    clk,
    rstn,
    digital_out,
    analog_cmp,
    analog_out,
    sample_rdy);

parameter
ADC_WIDTH  =  8,                    // ADC Convertor Bit Precision
ACCUM_BITS  =  10,                  // 2^ACCUM_BITS is decimation rate of accumulator
LPF_DEPTH_BITS  =  3;               // 2^LPF_DEPTH_BITS is decimation rate of averager

//input ports
input     clk;                                  // sample rate clock
input     rstn;                                 // async reset, asserted low
input     analog_cmp ;                          // input from LVDS buffer (comparitor)

//output ports
output    analog_out;                           // feedback to comparitor input RC circuit
output    sample_rdy;                           // digital_out is ready
output  [ADC_WIDTH-1:0]     digital_out;        // digital output word of ADC


//***************************************************************************
//
// Internal Wire & Reg Signals
//
//***************************************************************************
reg                             delta;              // captured comparitor output
reg  [ACCUM_BITS-1:0]       sigma;              // running accumulator value
reg  [ADC_WIDTH-1:0]            accum;              // latched accumulator value
reg  [ACCUM_BITS-1:0]       counter;            // decimation counter for accumulator
reg                             rollover;           // decimation counter terminal count
reg                             accum_rdy;          // latched accumulator value 'ready'




//***************************************************************************
//
//  SSD 'Analog' Input - PWM
//
// External Comparator Generates High/Low Value
//
//***************************************************************************

always @  (posedge clk)
begin
    delta  <=  analog_cmp;              // capture comparitor output
end

assign analog_out  =  delta;          // feedback to comparitor LPF
```

```
//***************************************************************************
//
//  Accumulator Stage
//
// Adds PWM positive pulses over accumulator period
//
//***************************************************************************

always @  (posedge clk or negedge rstn)
begin
    if( ~rstn )
    begin
        sigma       <= 0;
        accum       <= 0;
        accum_rdy   <= 0;
    end else begin
        if (rollover) begin
            // latch top ADC_WIDTH bits of sigma accumulator (drop LSBs)
            accum <= sigma[ACCUM_BITS-1:ACCUM_BITS-ADC_WIDTH];
            sigma <= delta;             // reset accumulator, prime with current delta value
        end else begin
            if (&sigma != 1'b1)             // if not saturated
                sigma <= sigma + delta; // accumulate
        end
        accum_rdy <= rollover;      // latch 'rdy' (to align with accum)
    end
end



//***************************************************************************
//
//  Box filter Average
//
// Acts as simple decimating Low-Pass Filter
//
//***************************************************************************

box_ave #(
    .ADC_WIDTH(ADC_WIDTH),
    .LPF_DEPTH_BITS(LPF_DEPTH_BITS))
box_ave (
    .clk(clk),
    .rstn(rstn),
    .sample(accum_rdy),
    .raw_data_in(accum),
    .ave_data_out(digital_out),
    .data_out_valid(sample_rdy)
);

//***************************************************************************
//
// Sample Control - Accumulator Timing
//
//***************************************************************************

always @(posedge clk or negedge rstn)
begin
    if( ~rstn ) begin
        counter <= 0;
        rollover <= 0;
        end
    else  begin
        counter <= counter + 1;            // running count
        rollover <= &counter;              // assert 'rollover' when counter is all 1's
        end
end

endmodule
```

数字低通滤波器模块，做平滑滤波

```
//   ==============================================================
//   >>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<<<
//   --------------------------------------------------------------
//   Copyright (c) 2014 by Lattice Semiconductor Corporation
//   ALL RIGHTS RESERVED
//   --------------------------------------------------------------
//
//   Permission:
//
//      Lattice SG Pte. Ltd. grants permission to use this code
//      pursuant to the terms of the Lattice Reference Design License Agreement.
//
//
//   Disclaimer:
//
//      This VHDL or Verilog source code is intended as a design reference
//      which illustrates how these types of functions can be implemented.
//      It is the user's responsibility to verify their design for
//      consistency and functionality through the use of formal
//      verification methods.  Lattice provides no warranty
//      regarding the use or functionality of this code.
//
//   ----------------------------------------------------------------
//
//                  Lattice SG Pte. Ltd.
//                  101 Thomson Road, United Square #07-02
//                  Singapore 307591
//
//
//                  TEL: 1-800-Lattice (USA and Canada)
//                       +65-6631-2000 (Singapore)
//                       +1-503-268-8001 (other locations)
//
//                  web: http://www.latticesemi.com/
//                  email: techsupport@latticesemi.com
//
//   ----------------------------------------------------------------
// ----------------------------------------------------------------
//
//  Project:     ADC_lvds
//  File:        box_ave.v
//  Title:       Box Filter Average
//  Description: Returns average of last N samples, with /N decimation
//
// ----------------------------------------------------------------
//
// Revision History :
// ----------------------------------------------------------------
// $Log: RD#rd1066_simple_sigma_delta_adc#rd1066#source#verilog#box_ave.v,v $
// Revision 1.1  2015-02-05 00:00:54-08  mbevinam
// Updated RD Placed in RD Folder. Previous versions are in RD_Dimensions Archive folder.
//
// Revision   Date
// 1.0       10/16/2009 S. Hossner   Initial Revision
//
// ----------------------------------------------------------------



//*********************************************************************
```

```
//
// 'Box' Average
//
//  Standard Mean Average Calculation
//   Can be modeled as FIR Low-Pass Filter where
//   all coefficients are equal to '1'.
//
//*************************************************************************



module box_ave (
    clk,
    rstn,
    sample,
    raw_data_in,
    ave_data_out,
    data_out_valid);

parameter
ADC_WIDTH  = 8,                   // ADC Convertor Bit Precision
LPF_DEPTH_BITS  = 4;              // 2^LPF_DEPTH_BITS is decimation rate of averager

//input ports
input     clk;                                    // sample rate clock
input     rstn;                                   // async reset, asserted low
input     sample;                                 // raw_data_in is good on rising edge,
input     [ADC_WIDTH-1:0]    raw_data_in;        // raw_data input

//output ports
output  [ADC_WIDTH-1:0]     ave_data_out;         // ave data output
output  data_out_valid;                           // ave_data_out is valid, single pulse

reg  [ADC_WIDTH-1:0]    ave_data_out;
//*************************************************************************
//
// Internal Wire & Reg Signals
//
//*************************************************************************
reg  [ADC_WIDTH+LPF_DEPTH_BITS-1:0]        accum;              // accumulator
reg  [LPF_DEPTH_BITS-1:0]                  count;              // decimation count
reg  [ADC_WIDTH-1:0]                    raw_data_d1;      // pipeline register

reg  sample_d1,  sample_d2;                               // pipeline registers
reg  result_valid;                                         // accumulator result 'valid'
wire  accumulate;                                          // sample rising edge detected
wire  latch_result;                                        // latch accumulator result

//*************************************************************************
//
//  Rising Edge Detection and data alignment pipelines
//
//*************************************************************************
always @(posedge clk or negedge rstn)
begin
    if( ~rstn )  begin
        sample_d1  <=  0;
        sample_d2  <=  0;
        raw_data_d1  <=  0;
        result_valid  <=  0;
    end else begin
        sample_d1 <= sample;                 // capture 'sample' input
        sample_d2 <= sample_d1;              // delay for edge detection
        raw_data_d1 <= raw_data_in;          // pipeline
        result_valid <= latch_result;        // pipeline for alignment with result
    end
end

assign      accumulate = sample_d1 && !sample_d2;        // 'sample' rising_edge detect
assign      latch_result = accumulate && (count == 0);  // latch accum. per decimation count

//***************************************************************************
//
// Accumulator Depth counter
//
//***************************************************************************
always @(posedge clk or negedge rstn)
begin
    if( ~rstn ) begin
        count <= 0;
    end else begin
        if (accumulate)   count <= count + 1;           // incr. count per each sample
    end
end


//***************************************************************************
//
// Accumulator
//
//***************************************************************************
always @(posedge clk or negedge rstn)
begin
    if( ~rstn ) begin
        accum <= 0;
    end else begin
        if (accumulate)
            if(count == 0)                      // reset accumulator
                accum <= raw_data_d1;            // prime with first value
            else
                accum <= accum + raw_data_d1;    // accumulate
    end
end

//***************************************************************************
//
// Latch Result
//
// ave = (summation of 'n' samples)/'n'  is right shift when 'n' is power of two
//
//***************************************************************************
always @(posedge clk or negedge rstn)
begin
    if( ~rstn ) begin
        ave_data_out <= 0;
    end else if (latch_result) begin                // at end of decimation period...
        ave_data_out <= accum >> LPF_DEPTH_BITS;     // ... save accumulator/n result
    end
end
```

```
assign data_out_valid = result_valid;        // output assignment

endmodule
```

软件 & 硬件

元器件

TP1961-TR

超高速（7ns）比较器, 可工作于+3V/+5V

软件

Verilog

Verilog HDL是一种硬件描述语言，用于设计和归档电子系统。

工具

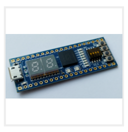
Diamond

Lattice的FPGA开发环境，支持Windows/Linux操作系统，支持器件XO、XO2、XO3系列FPGA器件

平台

2021年全国大学生电子设计竞赛综合训练平台

专为电赛技能培训设计，帮助信号源、仪器仪表、控制以及信号处理类题目的训练，基于全FPGA或MCU+FPGA混合架构，板上有高速ADC、高速DAC、高速比较...



STEP-MXO2小脚丫：基于Lattice XO2-4000HC的FPGA学习模块，板载编程器

STEP-MXO2选用了Lattice公司的MXO2-4000HC产品，在板卡的背面集成了编程器，只需要一根USB数据线就能够完成FPGA的编程和下载，提供了丰富的板卡外...

附件

FPGA-RD-02047-1-6-Simple-Sigma-Delta-ADC.pdf

Lattice发布的简单Sigma Delta ADC的应用指南 PDF

团队介绍

苏州硬禾信息科技有限公司

团队成员

子曰

猜你喜欢

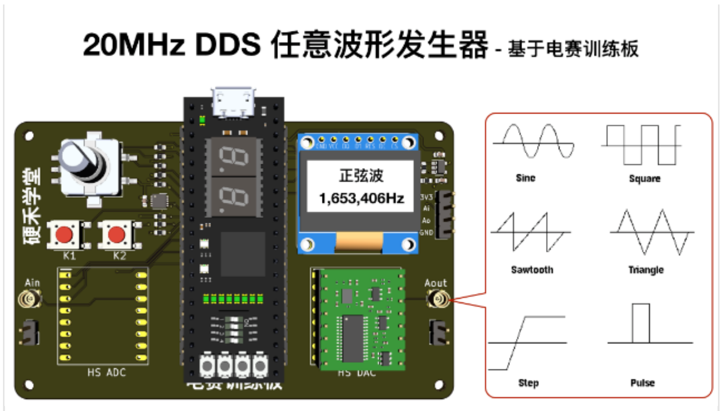
全国大学生电子设计竞赛训练平台

- 核心模块：FPGA + MCU
- 信息显示：128 * 64 点阵SPI OLED
- 控制输入：旋转编码器/按键
- 信号采集：10bit/50Msps 高速ADC
- 信号生成：10bit/120Msps 高速DAC
- 频率测量：高速比较器
- 控制输出：PWM
- 传感器：三轴姿态感知

2021年全国大学生电子设计竞赛综合训练平台

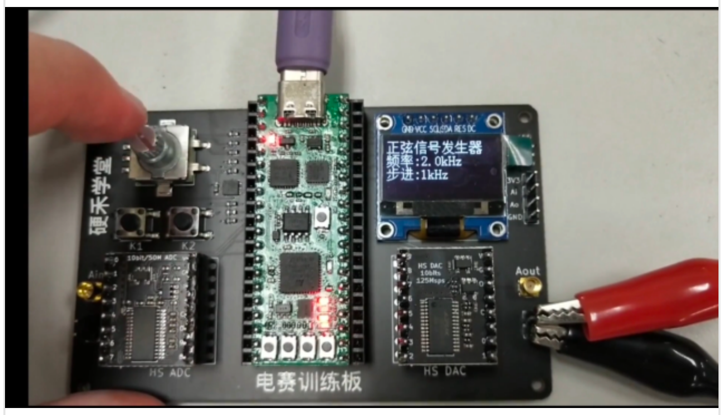
专为电赛技能培训设计，帮助信号源、仪器仪表、控制以及信号处理类题目的训练，基于全FPGA或MCU+FPGA混合架构，板上有高速ADC、高速

进度　　100%



DDS信号发生器 - 基于电赛综合训练板/小脚丫FPGA

在电赛综合训练板上实现高速DDS信号发生器的功能，1个旋转编码器和2个按键控制参数的输入，通过



硬禾学堂STM32+FPGA核心板+电赛综合训练板完成的DDS正弦信号发生器

本作品采用硬禾学堂的STM32G032 + ICE40UP5K FPGA核心板+电赛综合训练板套装，以单片机为控制