版本：1.2

版本更改说明：

1. 增加了自校准的描述。
2. 将差分时钟输入改为单端时钟输入，便于接入板载晶振时钟。
3. 根据最新 IP，调整了接口描述。
4. 增加了 SDK 函数的描述。

时间：2020 年 6 月 17 日

SeruTek TDC IP Core 简介

为了便于客户测试评估，瑟如电子现提供基于 zynq(7020 及以下)的评估版双通道 TDC IP 核。该 IP 核基于加密网表封装，并采用标准 AXI-lite、AXI-Stream 接口，可方便地在 Vivado 可视化设计环境中使用。

免费供评估的 IP 核是全功能 IP 核，hit 信号频率可高达 70MHz，IP 核内集成 FIFO 可供每通道最大速率测量 2048 个测试点（连续测量速率受限于 IP 核外的数据读出速率，通过使用 DMA 等技术可以轻易达到 30Msa/s 的连续测量速度），测时量程为 70 年。**评估版 IP 在每次上电后能正常工作约半小时，约 4 个小时后恢复正常工作，如一直上电，则保持该周期。**

该 IP 核绑定芯片 DeviceDNA，如需评估 IP 核及相关 SDK 程序，请致电瑟如电子并告知芯片型号及 DeviceDNA。

此外，瑟如电子也可提供大于 2 个通道或需基于更高等级器件（如 zynq7035, kintex 等）的 TDC IP 核，请致电上海瑟如电子了解详情。

目录

1 SeruTek TDC IP Core 功能及接口

1.1 IP 功能

1.1.1 两路时间戳

SeruTek TDC IP 采用时间戳的方式记录事件发生的时刻。输入的脉冲信号的上升沿代表事件的触发信号，触发信号使得该路 TDC 产生一个对应的时间戳。对同一路 TDC，将两个事件对应的时间戳的值相减就能得到两个事件之间的时差。两个通路之间的测量时差保持恒定（该值在外部校准时扣除）。因此，也可以采用多路 TDC，对不同来源的事件打上时间戳，作差后得到事件的时差。

本示例 IP 实现了 2 路 TDC 功能，一路可接收 start 信号，另一路可接收 stop 信号。IP core 输出 start 信号和 stop 信号的时间戳，可计算得到两者之间的时差。

1.1.2 自动片上校准

SeruTek TDC IP Core 具有片上自校准功能，由 IP 核内部产生校准所需的连续触发信号，采用码密度的方式，校准每个延迟单元的时延。得益于 SeruTek TDC IP Core 70MHz 的高速连续测量能力，能够采用高速校准信号，在数个毫秒内完

成一次自校准。

在温度变化范围较大的使用环境下，自校准功能能够极大的降低 TDC IP 的 INL 积分误差，从而保证 TDC 的测量准确度。用户只需根据片上温度传感器（XADC）获知温度变化，当超过一定阈值后，触发自校准功能即可。

1.1.3 TDC 配置

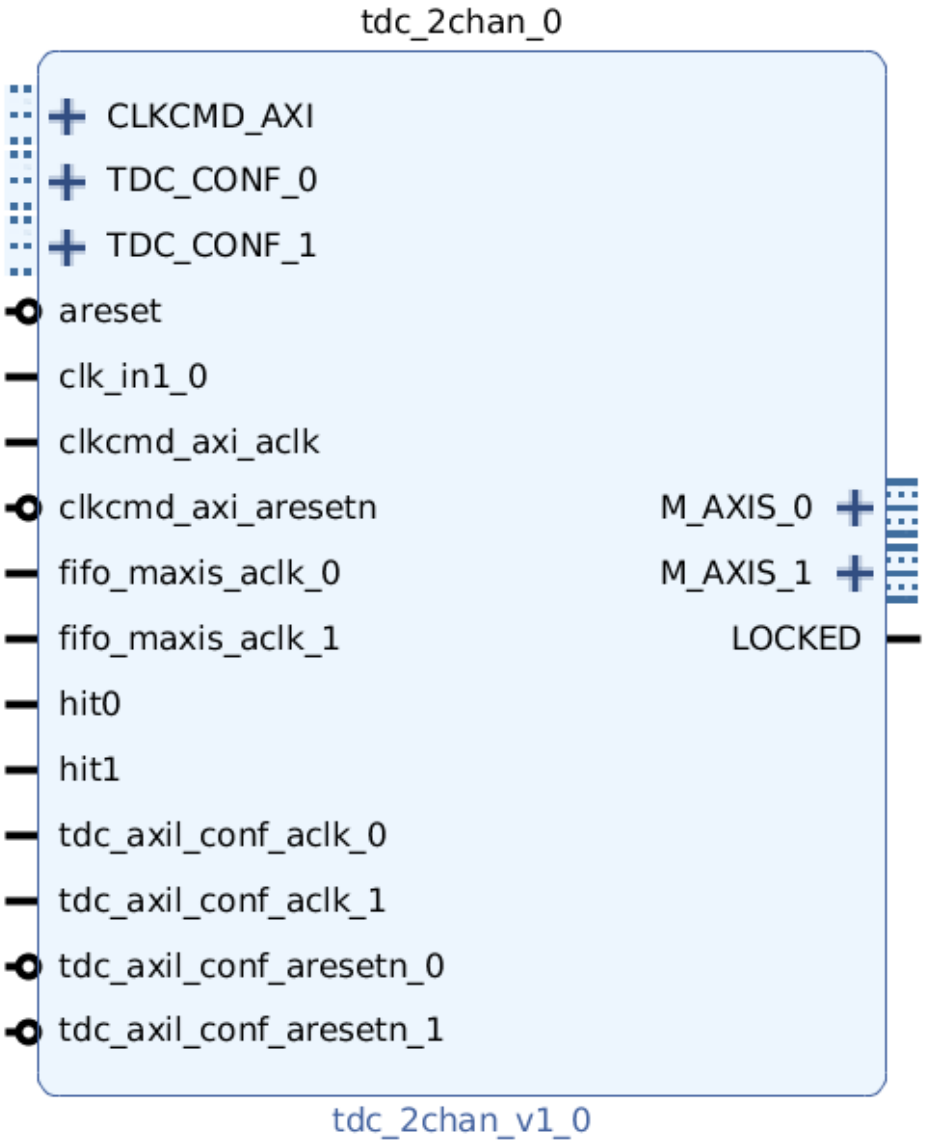
本 IP core 具有三个标准 AXI-lite 配置接口，可以非常方便的与 Zynq 或 Microblaze 连接，配置功能都可以在 SDK 中以函数形式调用。

1.1.4 时间戳数据输出

测量得到的时间戳通过标准 AXI-Stream 接口输出，Data 宽度为 96bit。可通过多种 AXIS-FIFO 接口接入 Zynq 的 AXI-GP 接口，示例程序中给出了参考实现。在 SDK 程序中提供了译码函数，能够打印时间戳以及时间戳之间的差值。

1.2 IP 接口

IP 的接口如下图所示



1.2.1 脉冲输入接口

本 IP 包含两路 TDC，因此 hit0、hit1 分别为两路 TDC 的脉冲输入。

1.2.2 配置接口

CLKCMD_AXI、TDC_CONF_0, TDC_CONF_1 是三个配置接口，为标准的 AXI-lite 格式，并且有对应的 clock,及 aresetn 接口。建议与系统的 AXI 总线连接。

1.2.3 时钟输入

Clk_in1_0 是时钟输入，预设为 50MHz 的单端输入（可修改为其它格式，如有需求请与瑟如电子联系）。

1.2.4 内部复位信号

Arest： （应为 arestn，低电平触发）

1.2.5 数据输出接口

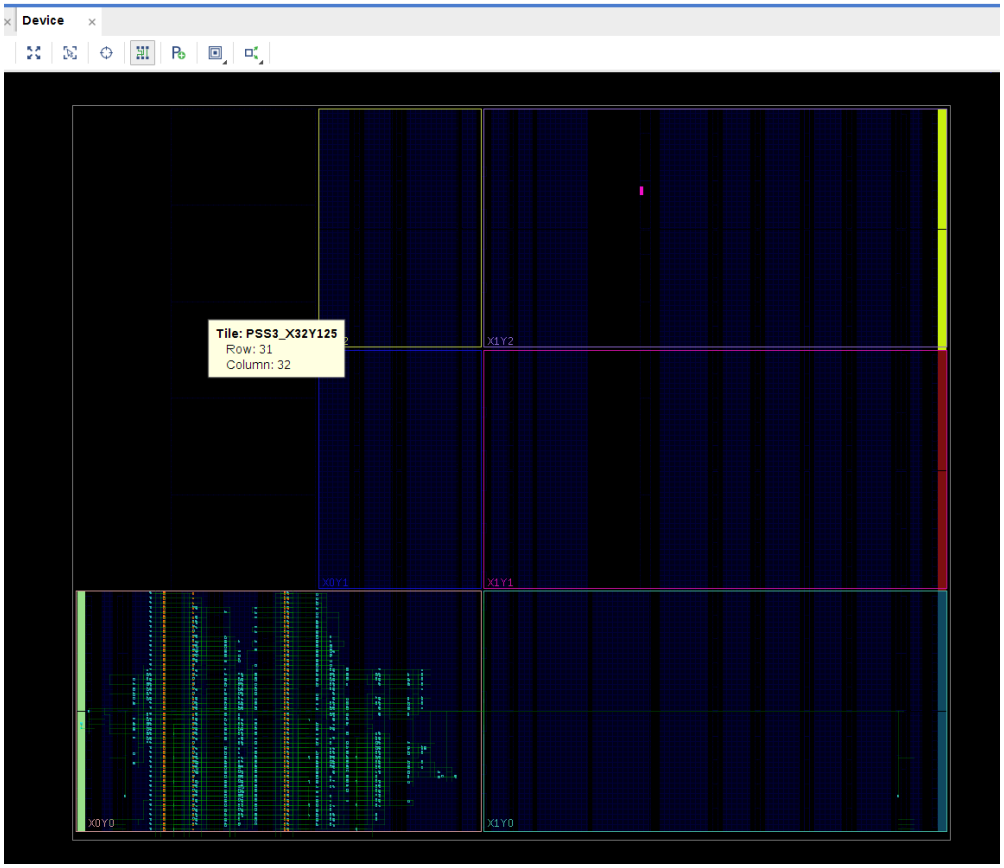
M_AXIS_0，M_AXIS_1 分别为时间戳数据的输出接口。其对应的时钟信号分别为 fifo_maxi_aclk_0、fifo_maxi_aclk_1

1.2.6 时钟锁定指示

LOCKED 是内部时钟锁定指示功能，可接入 LED，也可接入 GPIO，读取当前时钟的锁定状态。

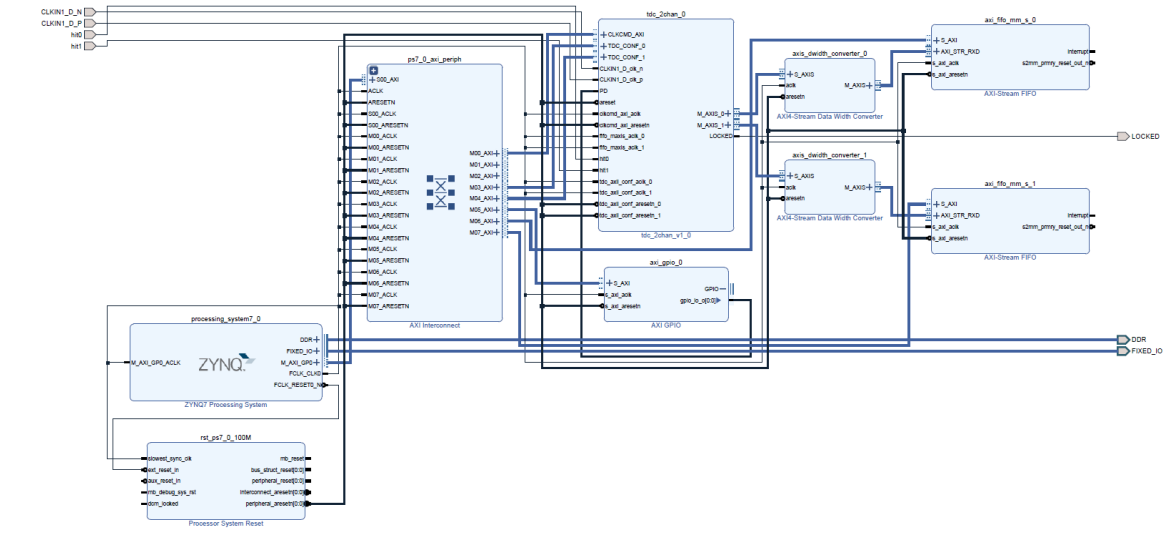
1.2.7 IP 资源占用

采用 Out of context 的方式综合、实现 IP 后得到如下的 Device 图



2 基于黑金 Zynq AX7020 开发板的示例

2.1 配置接口连接



示例程序的 Block Design 如上图所示。IP core 的三个配置接口通过自动连

接助手，通过 AXI interconnect 连接到 Zynq PS 的 GP0 端口。其对应的 CLK 为 Zynq PS 端引出的 PL CLOCK0，频率为 100M， resetn 为 Proceesor System reset 输出的 reset 信号。

2.2 时钟输入接口

TDC IP 默认输入时钟为 50MHz 的单端输入，在示例程序中，该时钟是由黑金 AX7020 开发板的 J11 扩展口引入的，此种方式时钟信号容易受到外界干扰，信号的质量较差。黑金的很多 Zynq 开发板上都集成了 50MHz 的晶振，也可以作为 TDC IP 的输入时钟，只需修改 clk_in1 的管脚约束即可。

高速差分时钟作为 TDC 的输入能够得到更低的抖动。如果新设计 FPGA 板卡，请联系上海瑟如电子咨询相关技术问题。

此外需要注意的是，时钟信号（或晶振输出信号）的各项性能指标将很大程度上影响 TDC IP 的测量精度及测量准确度。测时量程越大，对晶振频率的准确度、频率稳定度要求越高。如果使用环境温度变化较大，要选用低温漂的晶振，以降低输入温度变化对测时结果的影响。

2.3 时间戳数据输出接口

TDC IP 输入的数据格式为 AXI-Stream，data 长度为 12 个字节。需要加入 axis width converter 转换为 data 长度为 4 个字节的 AXI-Stream，并输入到 AXIS fifo 中去，Zynq ps 可通过该 IP 的 AXI-lite （或 AXI）接口读取时间戳。需要注意的是 axis width converter 和 axis fifo IP 中的 tlast 需要设置为 on。AXI 相关函

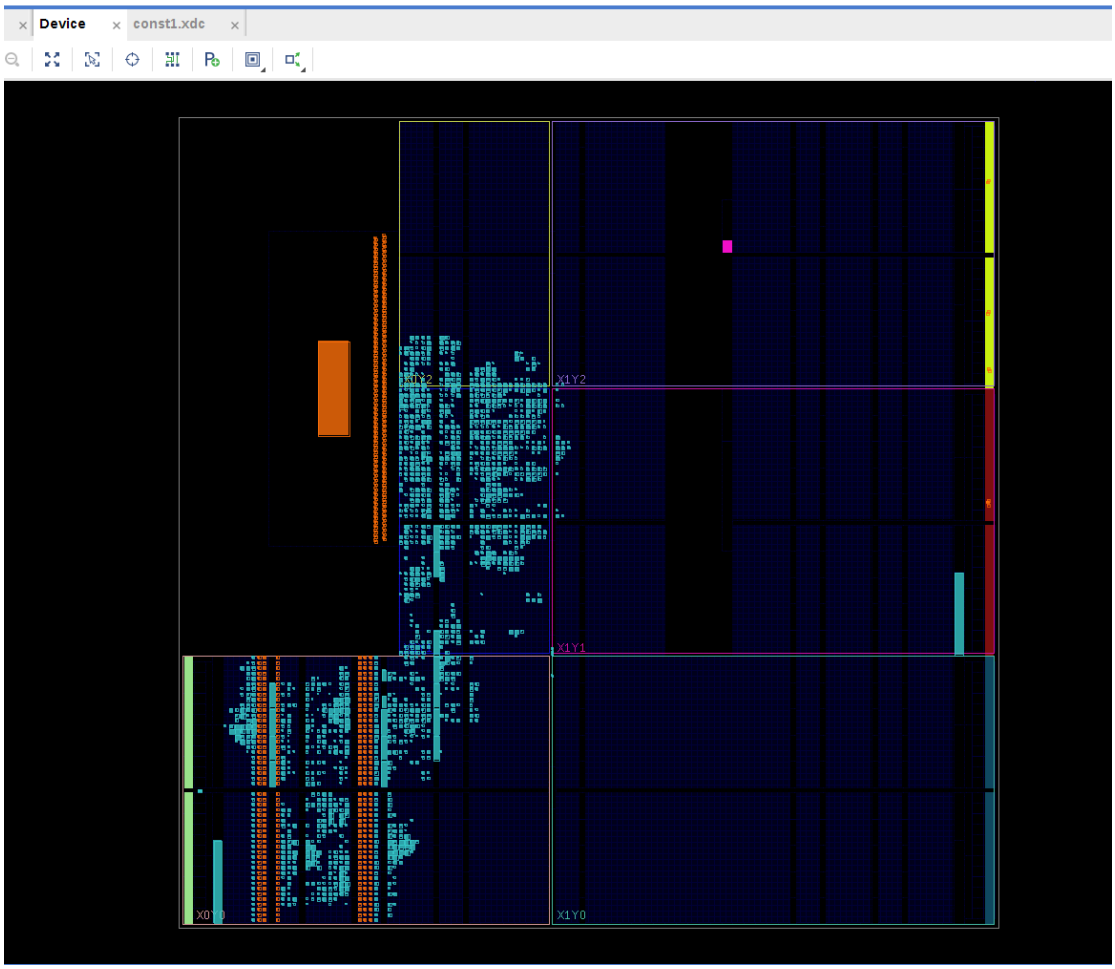
数将在 SDK 软件包中提供。

2.4 开发工具及资源使用

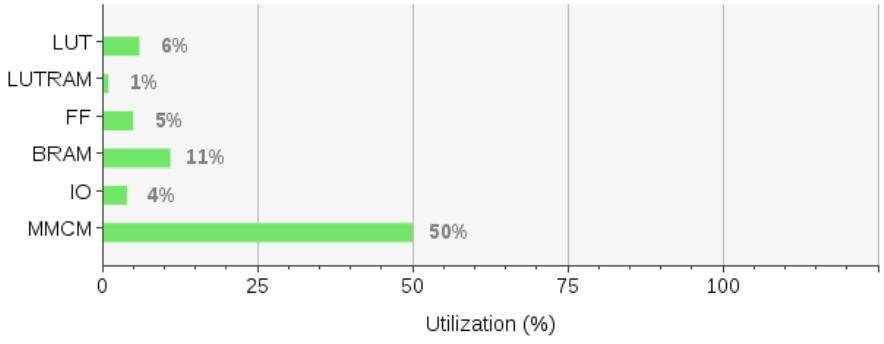
本示例开发工具为 Vivado 2019.1（可根据客户要求适配不同的版本）。Implementation 时需要设置一些 timing 约束。本示例所用约束示意图如下图：

```
1
2  set_property IOSTANDARD LVCMOS25 [get_ports clk_50M]
3  set_property PACKAGE_PIN U14 [get_ports clk_50M]
4
5  set_property PACKAGE_PIN F16 [get_ports hit0_0]
6  set_property PACKAGE_PIN K14 [get_ports hit1_0]
7  set_property IOSTANDARD LVCMOS25 [get_ports hit0_0]
8  set_property IOSTANDARD LVCMOS25 [get_ports hit1_0]
9
0  set_property IOSTANDARD LVCMOS25 [get_ports LOCKED_0]
1  set_property PACKAGE_PIN M15 [get_ports LOCKED_0]
2
3
4  set_clock_groups -asynchronous -group [get_clocks *fpga*
5  set_false_path -from [get_pins {design_1_i/tdc_2chan_0/i
6  set_max_delay -from [get_pins {design_1_i/tdc_2chan_0/in
7  set_false_path -from [get_pins {design_1_i/tdc_2chan_0/i
8  set_max_delay -from [get_pins {design_1_i/tdc_2chan_0/in
9
```

具体约束文件随 IP 核一起提供，其中部分时钟名可能需要根据客户的具体实现调整(如 AXI 的时钟信号名称)。本示例综合实现后的 device 如下图，可据此结合 utilization summary 一起评估所需资源。



Resource	Utilization	Available	Utilization %
LUT	3340	53200	6.28
LUTRAM	247	17400	1.42
FF	5717	106400	5.37
BRAM	15	140	10.71
IO	5	125	4.00
MMCM	2	4	50.00



3 SDK 函数介绍

3.1 功能介绍

为了便于客户的二次开发，我司免费提供基于上述示例工程配套的 SDK 示例程序及所需的函数库，共包含 3 个文件：库文件 libserutek_lib.a，头文件 serutek.h，及 bare-metal 应用： helloTDC.c。

在库文件中封装了诸如初始化、自校准、启动测量、停止测量以及译码函数等功能，可通过头文件 sertek.h 中列出的相应函数定义调用。而在示例应用 helloTDC.c 中，则采用了这些函数实现了 TDC 初始化（包含自校准），测量、数据读出、译码打印等一系列功能。

Serutek.h 文件内容如下：

```
#ifndef SERUTEK_TDC_H          /* prevent circular inclusions */
#define SERUTEK_TDC_H


#include "xil_types.h"
#include <stdio.h>
#include "xil_printf.h"
#include "xllfifo.h"
#include "xstatus.h"
struct mTdc
{
    u8   id;
```

```
    s64    corase_ns;
    s32    fine_ps;
};

/*************Clibrate TDC  should be called when temperature has been changed**************************************************/
int calibrate_tdc(u32 tdc_addr, u32 cmd_addr, XLlFifo *InstancePtr, u16 fifo_id);
int init_tdc(u32 tdc0_addr, u32 tdc1_addr, u32 cmd_addr, XLlFifo *InstancePtr0, XLlFifo *InstancePtr1);
void startMeasure2chan(u32 tdc_addr0, u32 tdc_addr1, u32 cmd_addr);
void startMeasure1chan(u32 tdc_addr, u32 cmd_addr);
void stopMeasure1chan(u32 tdc_addr);
void parseRaw2tdc(u8* bufPtr, struct mTdc *val, u16 tdc_id);
s32 TdcSubstract(struct mTdc tdc1, struct mTdc tdc2);
s64 TdcSubstract64(struct mTdc tdc1, struct mTdc tdc2);
#endif
```

其中 mTdc 结构体定义了测量值的格式，一个测量值由 3 个数据组成：id **表示 TDC 的通道号，本 IP 是双通道 TDC，通道号为 0 或 1。**corase_ns **表示粗计数值，单位为 ns。**fine_ps **表示精计数的值，单位为 ps。**

calibrate_tdc： **自校准函数，输入参数分别为：**tdc **寄存器地址，命令寄存**

器地址，该通道 TDC 对应的指向 AXIS-FIFO 实例的指针，以及 tdc 的通道号。

init_tdc：TDC 初始化函数，包括了 TDC 鉴权、初始化等功能，输入参数为：tdc0 寄存器地址，tdc1 寄存器地址，命令寄存器地址，TDC0 对应的指向 AXIS-FIFO 实例的指针，TDC1 对应的指向 AXIS-FIFO 实例的指针。

startMeasure2chan： 同时使能开启双通道。

startMeasure1chan： 使能单通道测量。

stopMeasure1chan： 关闭单通道。

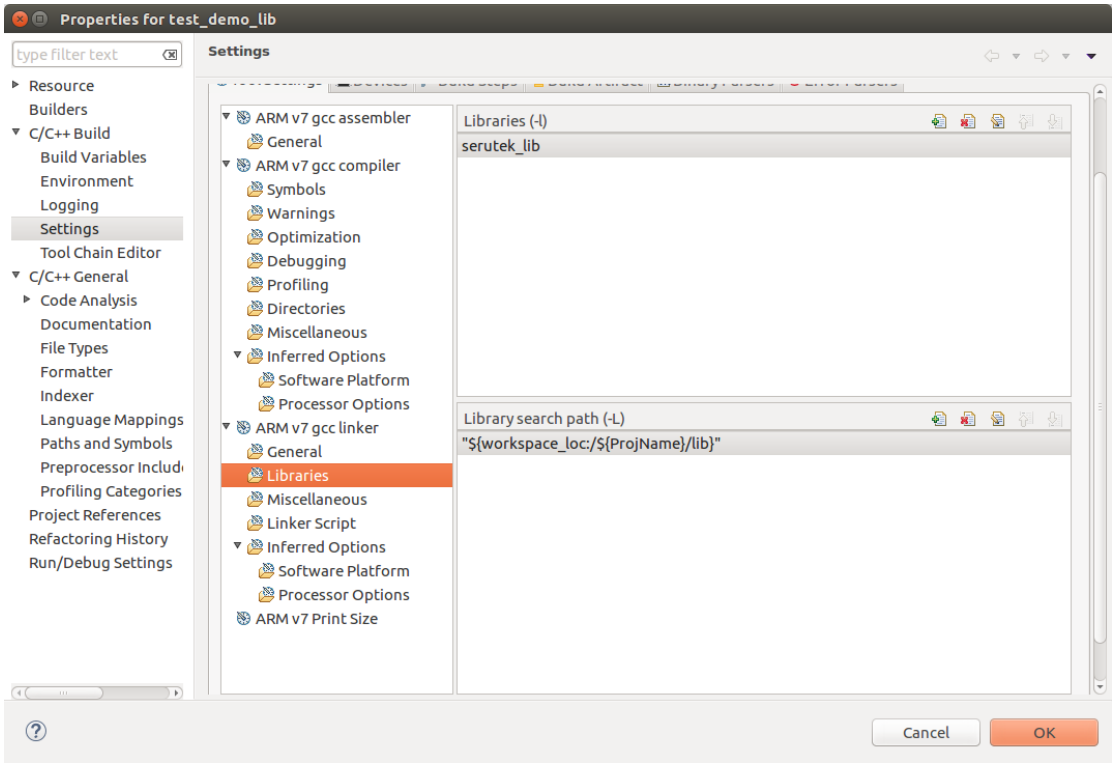
parseRaw2tdc(u8* bufPtr, struct mTdc *val, u16 tdc_id)： 译码函数。bufPtr 为缓存指针，存放从 AXIS-FIFO 读到的单条 TDC 测量值原始记录。 Val 是指向存放译码结果结构体的指针。tdc_id 是通道号。

s32 **TdcSubstract**(struct mTdc tdc1, struct mTdc tdc2)： 计算两个测量值之间的时差，返回数据为 32 位 signed integer，单位为 ps。能够正确表示的时间差最大约为 4.2 毫秒。

s64 **TdcSubstract64**(struct mTdc tdc1, struct mTdc tdc2): 计算两个测量值之间的时差，返回数据为 64 位 signed integer，单位为 ps。能够正确表示的时间差最大约为 213 天。

3.2 函数库使用方法

由 vivado 导出 bitstream 后，创建新的 SDK 功能，并新建 helloworld 示例程序。删除 helloworld.c 文件，将头文件 serutek.h 及 helloTDC.c 添加到 src 目录。在工程下新建文件夹如 lib，复制库文件 libserutek_lib.a 至此目录。并右击工程，选择 c/c++ building settings: 如下图设置 linker 的 libraries 和 library path:



接下来就可以正常编译了。