XADC 功能介绍和应用指南

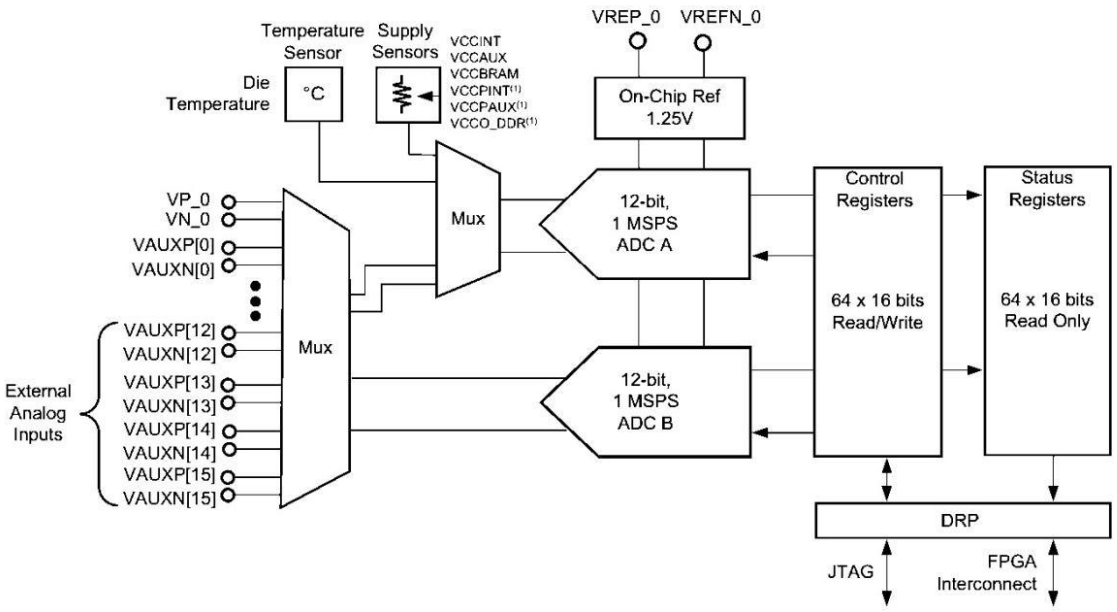
宋祈真　2019-01-25

一、XADC 功能介绍

XADC 是内嵌在 Xilinx 7 系列 FPGA 芯片和 Zynq7000 可编程 SOC 芯片中的的高速 AD 转换器模块。该模块内有两个 12 位 AD 转换器（确保 10 位可靠精度），采样速率达到每秒 1 兆次。XADC 可测量多达 17 个外部模拟通道的输入信号，同时还能对芯片的温度和供电电压进行监测和报警。XADC 能自动循环扫描所有的模拟输入通道，大大地减轻了主机的负担。特别是 Zynq7000 芯片内部集成了 2 个 ARM9 处理机芯片、FPGA 逻辑和 AD 转换器，丰富的资源构成完整的信号采集和处理系统，特别适合对精度要求不是太高但对成本又较敏感的测控仪器仪表等方面的应用，既可简化设备又可提高性价比，因此很受业界欢迎。

有关 XADC 的官方资料主要包含在 *Xilinx* 的“*7 Series FPGAs XADC Guide*（*UG480*）”和“*XADC Wizard v3.3 LogicCORE IP Product Guide*（*PG091*）”等技术文档中。这些资料内容丰富，但也比较庞杂，用户需要耗费大量时间才能消化理解。同时已有资料对应用实例的介绍又不太详尽。笔者主要对这两份资料的内容进行了梳理和精简，省略了仿真和 JTAG 方面的内容，重点放在介绍和理解 XADC 控制寄存器的操作方式。最后展示了在 Vivado 和 SDK 开发环境下的一个简单 XADC 应用实例。希望此文能帮助有兴趣的读者能在较短时间内能掌握和应用 XADC。

本文在短时间内完成，加上笔者水平有限，错误和缺漏难于避免，敬请原谅和批评指正。



UG480 图 1-1 XADC 原理框图

本文的参考资料主要来自 *Xilinx* 的 *UG480* 和 *PG091* 等技术文档，为便于读者对照原文查阅更详细的信息，本文引用的表格或插图仍采用原文的编序。

1. XADC 的属性介绍

- 内含两个 12 位、每秒 1 兆采样（1MSPS）的 AD 转换器 ADC A 和 ADC B。

- XADC 含片上温度传感器，可测试和监控芯片的温度，实现超温报警（OT）。默认的超温报警温度是 125° C，用户可以在发生过热报警时及时切断芯片供电以避免损失。用户还可以设定自己定义芯片工作温度范围，当芯片温度超出用户设定的界限时予以报警。

- XADC 内含 6 个芯片供电电压传感器，可对芯片供电电源监测和报警。（PL 是 FPGA 逻辑电路，PS 是 CPU 系统）被监测的电源电压如下：

  VCCINT: PL 内核电压（1.0V）
  VCCAU: PL 辅助电压（1.8V）
  VCCBRAM: PL BRAM 电压（1.0V）
  VCCPINT: PS 内核电压（1.0V）
  VCCPAUX: PS 辅助电压（1.8V）
  VCCO_DDR: DDR RAM 电压（1.5V）

  用户可以为这些电压设定上、下限，当电压超出用户设定的界限时报警。

- XADC 有一路专用的差分模拟输入通道 Vp/Vn，如果不使用 Vp/Vn 时，应该将 Vp/Vn 连接到 GNDADC 引脚。

- XADC 另有 16 路辅助模拟输入通道。这些输入通道与 Vp/Vn 不同之处是，他们的引脚是模拟/数字复用引脚。如果其中某个引脚被用做模拟输入，该引脚的数字 I/O 功能就不可再用。如果辅助模拟输入通道所在的 BANK 中混合有模拟和数字 I/O，该 I/O BANK 必须由满足所用数字 I/O 标准所需的电压供电。

- 辅助模拟输入通道既可以设置成单极性输入，也可以设置成双极性输入。但所有片上传感器（温度和电压）都采用单极性工作模式。

- XADC 可对 ADC 转换结果自动计算和输出平均值，以抑制信号和 PCB 板上的噪音。用户可从无平均值计算、16 个样本平均，64 个样本平均或 256 个样本平均等四个选项中选取。

- XADC 允许延长采样充电时间，将采样充电时间从 4 个 ADCCLK 延长至 10 个 ADCCLK。当信号源内阻或模拟通道多路器内阻过大影响采样电容获得的信号精度时，加长采样时间可以提高精度。

- XADC 内部含有 1.25V 的参考电压源。将 VREF_P0 接模拟地 GNDADC 就能启用片上参考电压。如果希望得到更高的精度，可以将 VREF_P0 连接外部精密电压参考 IC。

- XADC 可基于已知的参考电压源对传感器、ADC A 和 ADC B 的偏移和增益偏差进行校准计算，得到的校准系数存储在 XADC 的只读状态寄存器中。随后可以利用这些系数对 ADC 转换结果进行补偿。

- XADC 有 128 个 16 位寄存器（见图 1-1），其中前 64 个（DADDR[6:0]=00h～3Fh）为只读寄存器。用来存放片上传感器、专用模拟通道和辅助模拟通道的当前模数转换结果，同时存储芯片自本次上电以来测量到的最高和最低电压值，分别用 ADC A 和 ADC B 测量的电源传感器偏差、以及 ADC A 和 ADC B 的自身偏移和增益的校准系数。最后一个地址（3Fh）Flag 寄存器存储当前报警状态。

- 地址 40h 至 7Fh 是可读/写寄存器。其中前 3 个是 XADC 配置和控制寄存器、接着的 8 个是辅助模拟通道选择寄存器、再后面是 16 个报警阈值预设寄存器。用户通过配置这些寄存器来控制 XADC 的行为。剩下的其他寄存器用于工厂测试或未定义，用户不必关注。

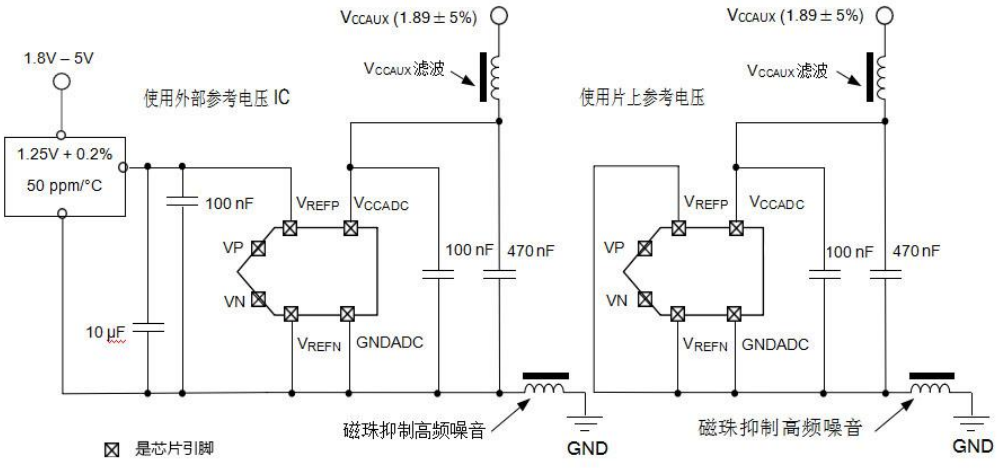
- 为了减轻 PS 处理器或 FPGA 中其他控制系统的负担，XADC 提供了一个叫作 Sequencer 的模拟通道自动扫描器。Sequencer 按预定的顺序自动地逐个接通用户选择的模拟输入通道，执行 ADC 转换并把结果存进 XADC 的只读状态寄存器。

- FPGA 或 Synq7000 中的 PL 逻辑可以通过 16 位宽的动态重配置接口（DRP）访问 XADC 的状态和控制寄存器，通过 JTAG 口也可以访问这些寄存器。但 Zynq7000 的 PS 是通过 AXI4-Lite 或 AXI4 Stream 接口访问 XADC 寄存器的。在 Vivado 开发环境中用户不需要关心 PS 和 XADC 互联的细节，只需在运行 XADC Wizard 时选择 AXI4-Lite 或 AXI4 Stream 就可以了，系统将自动建立两者的互连关系。

2. XADC 的引脚

所有 XADC 的专用引脚都在 BANK0 中，引脚名都以_0 后缀。*UG480* 图 1-2 展示了 XADC 的 2 种基本连接方法。XADC 由 VCCAUX（1.8V）供电，VCCAUX 上的低通滤波器用来滤除高频干扰以提高 ADC 性能。

图中左边的电路采用外部 1.25V 精密参考 IC 做参考电压，外部参考能提供更好的精度和热漂移性能。铁氧体磁珠用于隔离 GNDADC 和系统地 GND。*UG480* 图 1-2 右边电路采用片上自带的参考电压源。将 VREFP 引脚接到 GNDADC 引脚就能激活片上参考源。如果只需进行基本的片上温度和电源监测，片上参考已能提供良好性能。100 nF 退耦电容器应尽可能靠近芯片的 BGA 引脚，以减小寄生电感。



UG480 图 1-2 XADC 引脚

UG480 表 1-1 XADC 引脚

引 脚	用 途	说 明
V_{CCADC_0}	供电电压	这是 XADC 中 ADC 和其他模拟电路的模拟电源引脚。它可以连接到 1.8V VCCAUX 电源；但是，在混合信号系统中，电源最好应连接到独立的 1.8V 模拟电源（如果可用）。此引脚不应连接到 GND。即使 XADC 不用，也应将该管脚连接到 VCCAUX 上。
GNDADC_0	供电电压	这是 XADC 中 ADC 和其他模拟电路的接地参考引脚。如图 1-2 所示，它应通过隔离铁氧体磁珠与系统接地 GND 连接。在一个混合信号系统中，如果可能的话，此引脚应该连到在一个模拟的接地平面上，在这种情况下就可以不需要铁氧体磁珠。即使 XADC 不用，也应始终将此引脚连接到 GND。
V_{REFP_0}	参考电压输入	此引脚可连接到外部 1.25V 精确参考 IC（±0.2%或 12 位的±9 LSB），以获得 ADC 的最佳性能。应将其视为模拟信号，与 VREFN 信号一起提供 1.25V 差分电压。 如果将此引脚连接到 GNDADC（见图 1-2），芯片内参考源（12 位时为±1%或±41 LSB）被激活。如果没有提供外部引用，这个引脚应该应始终连接到 GNDADC。
V_{REFN_0}	参考电压输入	该引脚应与外部 1.25V 精确参考 IC（±0.2%）的地引脚相连，以获得最佳的 ADC 性能。应将其视为模拟信号，与 VREFP 信号一起提供差分 1.25V 电压。即使没有提供外部参考，此引脚也应始终连接到 GNDADC。
V_{P_0}	专用模拟输入	这是专用差分模拟输入通道（VP/VN）的正输入端。如果不使用，该引脚应连接到 GNDADC。
V_{N_0}	专用模拟输入	这是专用差分模拟输入通道（VP/VN）的负输入端。如果不使用，该引脚应连接到 GNDADC。
_AD0P_ to _AD15P_	辅助模拟输入/数字 IO	这些是多功能引脚，可以支持 16 个模拟的正输入端，也可以用作常规数字 I/O（见图 1-1）。当不被用作模拟输入时，这些引脚可以像其他任何数字 I/O 一样使用。
_AD0N_ to _AD15N_	辅助模拟输入/数字 IO	这些是多功能引脚，可以支持 16 个模拟的负输入端，也可以用作常规数字 I/O（见图 1-1）。当不被用作模拟输入时，这些管脚可以像其他任何数字 I/O 一样使用。

二、XADC 的状态和控制寄存器

XADC 的所有功能都是通过状态和控制寄存器实现的。了解这些寄存器的位定义功能可以帮助用户理解 Vivado 的 XADC Wizard 中的设计界面和掌握应用 XADC IP 的 API 函数。下面做详细介绍。

1. XADC 的状态寄存器

XADC 的前 64 个地址位置（DADDR[6:0]=00h 到 3Fh）是只读状态寄存器。用来存储片上传感器和外部模拟通道的模数转换结果。每个传感器和外部模拟输入通道都有一个唯一的通道地址，每个模拟通道的测量结果存储在通道选择寄存器（48h 和 49h）中指定地址的状态寄存器中（见 *UG480* 表 4-1）。

例如，ADC 多路复用器通道 0（温度传感器）的测量结果存储在通道地址 00h 的状态寄存器中。来自 ADC 多路复用器信道 1（VCCINT）的结果存储在地址 01h。

状态寄存器还存储从设备加电或XADC的最后一次用户重置开始记录的芯片上传感器的最大和最小测量值。表 3-1 定义了状态寄存器。

UG480 表 3-1 状态寄存器（只读）

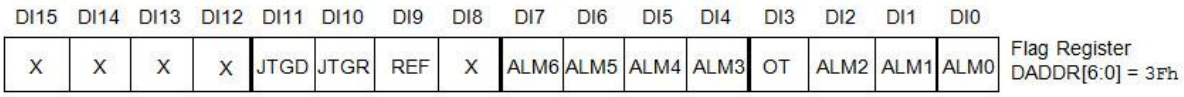
名	地址	说明
温度	00h	温度传感器测量结果存储在此位置。数据放在在 16 位寄存器的最高 12 位。
V_{CCINT}	01h	VCCINT 测量结果存储在该位置。数据放在在 16 位寄存器的高 12 位。
V_{CCAUX}	02h	VCCAUX 测量结果存储在该位置。数据放在在 16 位寄存器的高 12 位。
V_P/V_N	03h	专用模拟输入通道上的转换结果存储在此寄存器中。数据在 16 位寄存器中与最高位对齐的。
V_{REFP}	04h	参考输入 VREFP 上的转换结果存储在此寄存器中。数据在 16 位寄存器中与最高位对齐的。
V_{REFN}	05h	参考输入 VREFN 上的转换结果存储在此寄存器中。该通道在双极模式下进行测量。通过双极模式下的测量，可以测量到 0V（VREFN）左右的微小正负偏移。电源传感器也用于测量 VREFN，因此 1 LSB=3V/4096。数据在 16 位寄存器中是最高位对齐的。
V_{CCBRAM}	06h	VCCBRAM 电源监视器测量结果存储在该位置。数据在 16 位寄存器中是最高位对齐的。
未定义	07h	这些位置未使用，并且包含无效数据。
Supply A offset	08h	用 ADC A 计算的传感器偏移校准系数存储在此位置。
ADC A offset	09h	ADC A 的偏移校准系数存储在此位置。

ADC A gain	0Ah	ADC A 的增益误差校准系数存储在此位置。
未定义	0Bh to 0Ch	这些位置未使用，并且包含无效数据。
V_{CCPINT} (1)	0Dh	PS 电源 VCCPINT 上的转换结果存储在此寄存器中。数据在 16 位寄存器中是最高位对齐。测量 VCCPINT 时使用电源传感器。
V_{CCPAUX} (1)	0Eh	PS 电源 VCCPAUX 的转换结果存储在此寄存器中。数据在 16 位寄存器中是最高位对齐的。测量 VCCPAUX 时使用电源传感器。
V_{CCO_DDR} (1)	0Fh	PS 电源 VCCO_DDR 的转换结果存储在此寄存器中。数据在 16 位寄存器中是最高位对齐的。测量 VCCO_DDR 时使用电源传感器。
VAUXP[15:0]/ VAUXN[15:0]	10h to 1Fh	辅助模拟输入通道上的转换结果存储在此寄存器中。数据在 16 位寄存器中是最高位对齐的。
Max temp	20h	自通电或上次 XADC 复位后记录的最大温度测量值。
Max V_{CCINT}	21h	自通电或上次 xadc 重置后记录的最大 V_{CCINT} 测量值。
Max V_{CCAUX}	22h	自通电或上次 xadc 重置后记录的最大 V_{CCAUX} 测量值。
Max V_{CCBRAM}	23h	自通电或上次 XADC 重置后记录的最大 V_{CCBRAM} 测量值。
Min temp	24h	自通电或上次 XADC 复位后记录的最低温度测量值。
Min V_{CCINT}	25h	自上电或上次 XADC 复位后记录的最小 V_{CCINT} 测量值。
Min V_{CCAUX}	26h	自上电或上次 XADC 复位后记录的最小 V_{CCAUX} 测量值。
Min V_{CCBRAM}	27h	自上电或上次 XADC 复位后记录的最小 V_{CCBRAM} 测量值
V_{CCPINT} (1) max	28h	自上电或上次 XADC 复位后记录的最大 V_{CCPINT} 测量值
V_{CCPAUX} (1) max	29h	自上电或上次 XADC 复位后记录的最大 V_{CCPAUX} 测量值
V_{CCO_DDR} (1) max	2Ah	自上电或上次 XADC 复位后记录的最大 V_{CCO_DDR} 测量值
Unassigned	2Bh	未定义

V_{CCPINT} (1) min	2Ch	自上电或上次 XADC 复位后记录的最小 V_{CCPINT} 测量值
V_{CCPAUX} (1) min	2Dh	自上电或上次 XADC 复位后记录的最小 V_{CCPAUX} 测量值
V_{CCO_DDR} (1) min	2Eh	自上电或上次 XADC 复位后记录的最小 V_{CCO_DDR} 测量值
未定义	2Fh	未定义
Supply B offset	30h	用 ADC B 计算的电源传感器偏移校准系数存储在此位置。
ADC B offset	31h	ADC B 的偏移校准系数存储在此位置。
ADC B gain	32h	ADC B 的增益误差校准系数存储在该位置。
Undefined	33h to 3Eh	未使用
Flag	3Fh	此寄存器包含常规状态信息（请参见 Flag Register）

注 1：这些通道仅适用 Zynq-7000 AP SoC 器件。

3Fh 是 Flag 寄存器，见 *UG480* 图 3-2 和 *UG480* 表 3-2。



UG480 图 3-2 Flag 寄存器

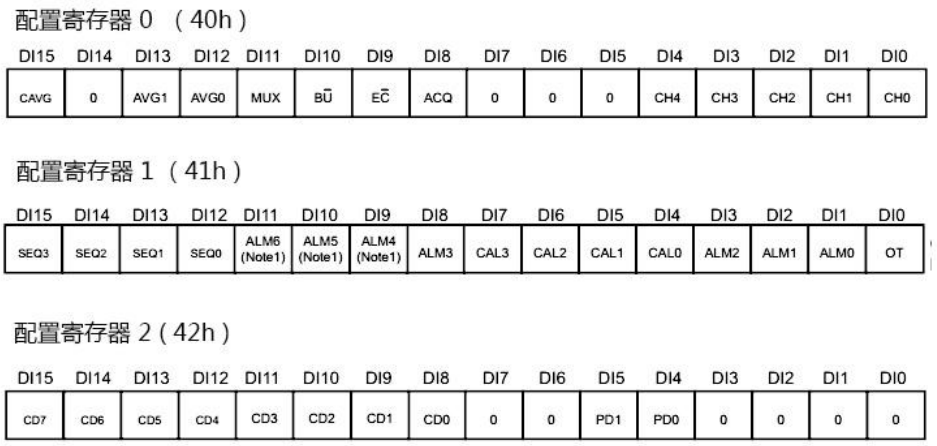
UG480 表 3-2 状态寄存器（只读）

位	名	说　明
DI7 to DI0	ALM6 to ALM0	[6 : 0]，这些位反映报警输出 ALM 的状态
DI3	OT	该位反映过热逻辑输出的状态
DI9	REF	当该位为逻辑 1 时，ADC 使用内部电压基准。当该位为逻辑 0 时，使用外部电压基准。
DI10	JTGR	逻辑 1 表示 JTAG_XADC 位流选项已用来将 JTAG 访问限制为只读。更多信息请参见 DRP JTAG 接口。
DI11	JTGD	逻辑 1 表示 JTAG_XADC 位流选项已用于禁用所有 JTAG 访问。更多信息请参见 DRP JTAG 接口。

2. XADC 的控制寄存器

⑴ 配置寄存器（40h 至 42h）

可读写寄存器中的前 3 个寄存器是配置寄存器。他们的位定义如 *UG480* 图 3-4 和 *UG48* 表 3-4 所示。



GU480 图 3-4

注意：寄存器中显示为 0 的位应始终保持设置为 0。

U480 表 3-4　配置寄存器 0（40h）的位定义

位	名	说　明
DI0 to DI3	CH0 to CH4	在单通道模式或使用片内辅助多路器模式下工作时，这些位用于选择 ADC 输入通道。通道分配见表 3-7。
D18	ACQ	单通道模式时将此位置 1 可使连续采样模式下的采样稳定时间增加 6 个 ADCCLK。
DI9	EC	置 1 将 ADC 置于事件驱动采样模式，置 0 将 ADC 置于连续采样模式。
DI10	BU	单通道模式下此位选择单极性或双极性操作模式。逻辑 1 将 ADC 置于双极性模式，逻辑 0 置于单极性模式。
DI11	MUX	此位置 1 可以将寄存器 0（40h）的 CH4:CH0 输出去控制一个片外的模拟多路器，只占用 FPGA 的 5 个 I/O 引脚。可节省 FPGA 的 I/O 资源。
DI12, DI13	AVG0, AVG1	用于设置单通道和自动序列模式下所选通道的样本平均计算取值
DI15	CAVG	此位用于禁用计算校准系数的平均值。默认情况下启用校准系数平均（逻辑 0）。禁用应将该位设置为逻辑 1。校准系数平均平均值固定为 16 个样本。

UG480 表 3-5 配置寄存器 1（41h）的位定义

位	名	说 明
DI0	OT	此位设置为 1 时禁禁止过热报警。
DI1 to DI3, DI8	ALM0 to ALM3	这些位分别用于禁用温度、V_{CCINT}、V_{CCAUX} 和 V_{CCBRAM} 的单个报警输出。逻辑 1 禁用报警输出。
DI9 to DI11	ALM4 to ALM6	这些位分别用于禁用 V_{CCPINT}, V_{CCPAUX} 和 V_{CCO_DDR} 的单个报警输出。逻辑 1 禁用报警输出。
DI4 to DI7	CAL0 to CAL3	这些位可以将校准系数应用到 ADC 和片上电源传感器测量中。逻辑 1 启用校准，逻辑 0 禁用校准。位分配见表 3-10。
DI12 to DI15	SEQ0 to SEQ3	这些位设置自动通道定序器的操作模式。位分配见表 3-9。

UG480 表 3-4 配置寄存器 2（43h）的位定义

位	名	说 明
DI4, DI5	PD0, PD1	设置 PD1 = PD0=1 将使 XADC 功能块永久关闭电源。 设置 PD1 = 1，PD0 = 0 将使 ADC B 永久关电。
DI8 to DI15	CD0 to CD7	这些位用于选择 DRP 时钟（DCLK）和较低的 ADC 时钟（ADCCLK）之间的分频因子。位分配见表 3-12。

UG480 表 3-7 内部辅助模拟多路器选择

通道	CH4	CH3	CH2	CH1	CH0	说 明
0	0	0	0	0	0	片上温度
1	0	0	0	0	1	V_{CCINT}
2	0	0	0	1	0	V_{CCAUX}
3	0	0	0	1	1	V_P ， V_N （专用模拟通道）
4	0	0	1	0	0	V_{REFP} (1.25V) (1)
5	0	0	1	0	1	V_{REFN} (0V) (1)
6	0	0	1	1	0	V_{CCBRAM}

7	0	0	1	1	1	非法通道选择
8	0	1	0	0	0	执行一次 XADC 校准
9 - 12	...	...	...	...	...	非法通道选择
13	0	1	1	0	1	V_{CCPINT} (3)
14	0	1	1	1	0	V_{CCPAUX} (3)
15	0	1	1	1	1	V_{CCO_DDR} (3)
16	1	0	0	0	0	VAUXP[0], VAUXN[0] 辅助道 0
17	1	0	0	0	1	VAUXP[1], VAUXN[1] 辅助道 1
18 - 31	...	...	...	...	...	VAUXP[2:15], VAUXN[2:15] 辅助通道 2 to 15 (2)

注 1.这些通道选择选项用于 XADC 自检和校准操作。当选择这些通道时，电源传感器连接到 VREFP 或 VREFN。

注 2. Kintex®7 设备不支持辅助通道 6、7、13、14 和 15。某些 virtex-7、Artix-7 和 Zynq-7000 AP SOC 封装选项也可能不支持某些辅助模拟通道。用户应该查阅器件的封装文件。

注 3.这些通道仅在 Zynq-7000 AP SOC 器件中受支持。

UG480 表 3-8 平均值设定

AVG1	AVG0	功 能
0	0	不计算平均值
0	1	计算 16 个样本平均
1	0	计算 64 个样本平均
1	1	计算 256 个样本平均

UG480 表 3-9 自动序列器操作模式设定

SEQ3	SEQ2	SEQ1	SEQ0	自动定序器操作模式
0	0	0	0	Default mode
0	0	0	1	Single pass sequence
0	0	1	0	Continuous sequence mode
0	0	1	1	Single channel mode (sequencer off)
0	1	X	X	Simultaneous sampling mode
1	0	X	X	Independent ADC mode
1	1	X	X	Default mode

UG480 表 3-10　校准使能

名	说　明
CAL0	ADCs 偏移校准使能
CAL1	ADCs 偏移和增益校准使能
CAL2	电源传感器偏移校准使能
CAL3	电源传感器偏移和增益校准使能

*UG480*表 3-11　电源关闭选择

PD1	PD0	说　明
0	0	默认所有 XADC 功能都加电
0	1	无效位
1	0	ADC B 关电
1	1	XADC 关电

UG480 表 3-12 DCLK 分频因子选择

CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	分 频
0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	1	1	3
0	0	0	0	0	1	0	0	4
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

注 1：最小分频比为 2，例如，ADCCLK = DCLK/2。

⑵ 控制寄存器

为减轻 PS 处理器其他 FPGA 逻辑负担，XADC 中设计了自动通道定序器（Automatic Channel Sequencer）。自动通道定序器可以按预设的次序自动扫描用户选择的模拟通道，执行 ADC 转换并存储结果。定序器有多种操作模式，由 XADC 配置寄存器 1（41h）的 SEQ3 到 SEQ0 位加另外 8 个控制寄存器来配置。自动定序器支持的操作模式如表 *UG480* 表 3-9 所示。

UG480 表 3-9 自动定序器的操作模式

SEQ3	SEQ2	SEQ1	SEQ0	Function
0	0	0	0	Default mode
0	0	0	1	Single pass sequence
0	0	1	0	Continuous sequence mode
0	0	1	1	Single channel mode (sequencer off)
0	1	X	X	Simultaneous sampling mode
1	0	X	X	Independent ADC mode
1	1	X	X	Default mode

8 个控制寄存器（48h 至 4Fh），分别用来选择要监测的模拟通道，设置采样平均次数，设置模拟输入模式和采样稳定时间。这 8 个寄存器可以被看成四对 16 位寄存器。每对寄存器控制不同的定序器功能。每对寄存器（32 位）中的单个位用来启用或禁止相关模拟通道的通道选择、平均计算、模拟输入模式延长采样这四种功能。注意这 4 对寄存器中的道序位定义是完全相同的。

- ADC 通道选择寄存器（48h 和 49h）
- ADC 结果平均次数（4Ah 和 4BH）
- ADC 通道模拟输入模式（4Ch 和 4Dh）
- ADC 通道采样稳定时间（4Eh 和 4Fh）

下面分别介绍这 4 对控制寄存器。

① ADC 通道选择寄存器（48h 和 49h）

这 2 个 16 位寄存器用来启用或禁用相关模拟通道。这 2 个寄存器的位在表 4-1 和表 4-2 中定义。逻辑 1 为启用，逻辑 0 为禁止。自动序列扫描固定从寄存器 48h 的 LSB（位 0）开始，到寄存器 49h 的 MSB（第 15 位）结尾。

UG480 表 4-1：定序器辅助通道选择寄存器（48h）

顺序号 Zynq-7000	Bit	ADC 通道	说　明
1	0	8	XADC 校准通道
–	1	9	无效通道选择
	2	10	
	3	11	
	4	12	
2	5	13	V_{CCPINT}
3	6	14	V_{CCPAUX}
4	7	15	V_{CCO_DDR}
5	8	0	片上温度

6	9	1	V_{CCINT}
7	10	2	V_{CCAUX}
8	11	3	V_P，V_N 专用模拟输入通道
9	12	4	V_{REFP}
10	13	5	V_{REFN}
11	14	6	V_{CCBRAM}
–	15	7	无效通道选择

*UG480*表 4-2：定序器辅助通道选择寄存器 (49h)

顺序号 Zynq-7000	Bit	ADC 通道	说　明
12	0	16	VAUXP[0], VAUXN[0] – 辅助通道 0
13	1	17	VAUXP[1], VAUXN[1] – 辅助通道 1
14	2	18	VAUXP[2], VAUXN[2] – 辅助通道 2
15	3	19	VAUXP[3], VAUXN[3] – 辅助通道 3
16	4	20	VAUXP[4], VAUXN[4] – 辅助通道 4
17	5	21	VAUXP[5], VAUXN[5] – 辅助通道 5
18	6	22	VAUXP[6], VAUXN[6] – 辅助通道 6
19	7	23	VAUXP[7], VAUXN[7] – 辅助通道 7
20	8	24	VAUXP[8], VAUXN[8] – 辅助通道 8
21	9	25	VAUXP[9], VAUXN[9] – 辅助通道 9
22	10	26	VAUXP[10], VAUXN[10] – 辅助通道 10
23	11	27	VAUXP[11], VAUXN[11] – 辅助通道 11
24	12	28	VAUXP[12], VAUXN[2] – 辅助通道 12

25	13	29	VAUXP[13], VAUXN[13] – 辅助通道 13
26	14	30	VAUXP[14], VAUXN[14] – 辅助通道 14
27	15	31	VAUXP[15], VAUXN[15] – 辅助通道 15

② ADC 通道平均值寄存器（4Ah 和 4Bh）

平均值寄存器（4Ah 和 4BH）用来启用或禁止每个通道的采样数据平均计算处理。这 2 个寄存器的道序位定义和表 4-1 和表 4-2 完全相同。XADC 配置寄存器 0（40h）中的 AVG1 和 AVG0 位用来选择平均值 16、64 或 256。参见 *UG480* 表 3-8。

如果自动序列中的某个通道未启用计算平均，其输出状态寄存器在每次通过一次序列扫描时都会更新。如果一个通道已启用平均，其输出状态寄存器仅在平均完成后才更新。

③ ADC 模拟信号输入模式寄存器（4Ch 和 4Dh）

在自动序列模式时，这 2 个寄存器用于将 ADC 通道配置为单极性或双极性输入。这 2 个寄存器的道序的位定义和表 4-1 和表 4-2 相同。但只有专用输入通道(VP 和 VN)和辅助模拟输入(Vauxp[15:0]和 Vauxn[15:0]）才可以配置。所有片内传感器通道都采用单极性输入。将寄存器中的一个位设置为逻辑 1 将启用相关通道的双极性输入模式，设置成逻辑 0（默认）启用单极性输入模式。

④ ADC 通道采样稳定时间寄存器（4Eh 和 4Fh）

连续采样模式下外部通道的默认采样稳定时间为 4 个 ADCCLK 周期。当信号源内阻较大或使用内阻较大的模拟信号多路器的情况下延长采样稳定时间可以提高采集信号的精度。这 2 个寄存器的道序的位定义和表 4-1 和表 4-2 相同。通过设置寄存器 4Eh 和 4Fh 中相应的位可以使相关信道的采样时间延长到 10 个 ADCCLK 周期。

⑤ 报警寄存器（50h 至 5Fh）

这些寄存器用于为内部监控通道：温度、VCCINT、VCCAUX 和 VCCBRAM 设定报警阈值。Zynq-7000 的 PS 供电电压 VCCPINT， VCCPAUX 和 VCCO_DDR 的报警阈值也使用这些寄存器进行设置。在 Vivado 的 XADC Wizard 中用户报警的阈值直接在对话框中设置。

三、XADC 自动通道定序器的操作模式

下面我们对 *UG480* 表 3-9 中所列的自动定序器的操作模式一一介绍。

1. 默认模式（Default Mode ）

这是最基本的操作模式，能监测温度传感器和所有片内电压传感器，此模式不需要用户对 XADC 寄存器进行配置。XADC 在上电和 FPGA 下载编程后自动进入默认模式。

任何其他模式时只需向配置寄存器 1（41h）的[SEQ3:SEQ0]写 0000 就可进入默认模式。在这种操作模式下，XADC 自动监测片上传感器并将结果存储在状态寄存器中。该模式下的两个 ADC 执行自动校准，并输出 16 个样本的平均值。此模式下 XADC 的其他 8 个控制寄存器中的设置值都无影响。

默认模式除 OT（过热）以外的所有报警输出（ALM[7:0]）都被禁止，但 ADC 校准被自动启用。

UG480 表 4-3：默认模式下的采样顺序

采样顺序（Zynq-7000）	通道	地　址	说　明
1	校准	08h	ADC A 和 ADC B 校准
2	V_{CCPINT}	0Dh	V_{CCPINT} 传感器
3	V_{CCPAUX}	0Eh	V_{CCPAUX} 传感器
4	V_{CCO_DDR}	0Fh	V_{CCO_DDR} 传感器
5	Temp	00h	温度传感器
6	V_{CCINT}	01h	V_{CCINT} 传感器
7	V_{CCAUX}	02h	V_{CCAUX} 传感器
8	V_{CCBRAM}	06h	V_{CCBRAM} 传感器

2. 单次通过模式（Single Pass Mode ）

当配置寄存器 1（41h）的[SEQ3:SEQ0]被设置为 0001 时，就立即启动单次通过模式。控制寄存器 48h 至 4Fh 用于定义用户要求的的行为功能。在这种模式下，XADC 按通道选择寄存器（48h 和 49h）定义的采样顺序扫描一次后就停止 ADC 转换。当执行序列扫描时，（48h 和 49h）寄存器中选择的通道都被采样和转换一次。表 4-1 和表 4-2 中列出的所有通道均可按序接通。

单次扫描完成后，XADC 默认进入单通道（Single Channel）模式。此时 XADC 将对配置寄存器 0（40h）中位 CH5 到 CH0 选择的通道采样和转换。如果要再次启动单次通过，就必须再次对配置寄存器 1（41h）的[SEQ3:SEQ0]写 0001。

3. 连续循环扫描模式（Continuous Sequence Mode ）

[SEQ3:SEQ0]被设置为 0010 时启动连续循环扫描模式。其操作过程与单次通过模式相同，但 ADC 采通道扫描会自动循环执行。用户可以在此模式运行期间时通过 DRP 重新配置通道选择寄存器（48h 和 49h）。但在对这 2 个寄存器执行写操作之前，必须先禁止连续循环扫描模式。建议在重置通道选择寄存器（48h 和 49h）前将 0000 写进寄存器（41h）的[SEQ3:SEQ0]，将 XADC 置于默认模式。

在对 SEQ3 至 SEQ0 执行写操作时，XADC 会自动复位，但状态寄存器的当前内容不会被清除。但是对 SEQ3 至 SEQ0 的写操作重新启动自动定序器扫描时会清除所有通道的平均参数设置。

4. 同步采样模式（Simultaneous Sampling Mode）

此模式使自动定序器同时操作 2 个 ADC 模块（ADC A 和 ADC B），对 8 对辅助模拟输入通道进行同步采样和转换，如表 4-4 所示。这对需要保持被采样的两个信号之间准确相位关系的应用非常有用。

辅助模拟通道 0 至 7 分配给转换器 ADC A，并命名为 A 通道。

辅助模拟信道 8 至 15 分配给转换器 ADC B，并命名为 B 通道。

在同步采样模式时，总是一对 A、B 通道同步采样和转换。表 4-4 显示使用通道寄存器 49h 时 A 通道和 B 通道配对情况。其他控制寄存器如定义平均次数、模拟输入模式和稳定时间都可用于此模式。

表 4-4：同步采样模式下定序器寄存器（49h）的位定义

顺序号	Bit	ADC 通道	说 明
1	0	16，24	辅助通道 0 和 8
2	1	17，25	辅助通道 1 和 9
3	2	18，26	辅助通道 2 和 10
4	3	19，27	辅助通道 3 和 11
5	4	20，28	辅助通道 4 和 12
6	5	21，29	辅助通道 5 和 13(1)
7	6	22，30	辅助通道 6 和 14(1)
8	7	23，31	辅助通道 7 和 15(1)
x	8	x	未定义
x	9	x	未定义
x	10	x	未定义
x	11	x	未定义
x	12	x	未定义
x	13	x	未定义
x	14	x	未定义
x	15	x	未定义

注 1.不支持 Kintex-7 器件

同步采样模式时，可以为 2 个通道分别设置平均次数（4Ah 和 4Bh）和模拟输入模式（4Ch 和 4Dh）。因此允许把 A 通道配置为单极性输入，B 通道配置为双极性输入。位序的定义和表 4-1 和表 4-2 一样。

但采样稳定时间的设置仅适用于通道对。因此，如果将寄存器 4Eh 中的位 0 设置为 1，结果将两个辅助通道 0 和 8 的稳定时间都设置为 10 个 ADCCLK 周期。

通过设置序列器寄存器 48h 中适当的位可以监控芯片上的温度和电源。但是同步采样模式时，当一个内部传感器通道正在被转换时，另一个 ADC（如 ADC B）不会对内部传感器进行 ADC 转换。

同步采样模式下不能执行自动校准，XADC 只能在默认模式或其他定序器模式时才能进行校准。但是 XADC 在上电时总是执行 ADC 自动校准，这对于大多数应用来说已足够了。

同步采样模式的定时与其他 XADC 模式相同。因为两个 ADC 在锁步模式下运行，所以 ADC 的两个状态寄存器是同时更新的。连续和事件驱动转换的定时模式都可以用于此模式。

5. 独立 ADC 模式 （Independent ADC Mode）

在独立 ADC 模式时，ADC A 用来实现固定的“监控模式”，除了自动启用报警功能外其他与默认模式相同。在此模式下，报警输出处于启用状态，所以必须正确配置报警阈值。与默认模式一样，平均值固定为 16 个样本。

ADC B 只能用于外部模拟输入通道。所以此模式可以将第二个 ADC 用于用户应用程序，同时又维持对 FPGA 的电源电压和温度的监控。

只有专用模拟通道（VP 和 VN）和辅助模拟输入通道才可以分配给 ADC B。而内部通道（传感器）自动分配给 ADC A，ADC A 自动监测这些通道并根据用户定义的报警阈值报警。

与同步采样模式一样，此模式无法执行 ADC B 的自动校准，但 ADC A 时能自动校准。如果要对 ADC B 执行校准并维持片上传感器监控，应先进入默认模式，等待 XADC 执行自动校准后（EOS 至少变高一次）再返回到独立 ADC 模式。

独立 ADC 模式的通道选择也是用控制寄存器 48h 和 49h（见表 4-5 和表 4-6）来定义的。其他控制寄存器如定义平均次数、模拟输入模式和采样稳定时间等功能都保持不变。

UG480 表 4-5 独立 ADC 顺序器模式的寄存器 48h 位定义

顺序号	Bit	ADC 通道	说　明
–	0	–	未定义
	1		未定义
	2		未定义
	3		未定义
	4		未定义
	5		未定义
	6		未定义
	7		未定义
	8		未定义
	9		未定义
	10		未定义
1	11	3	V_P/V_N
	12		未定义

-	13		未定义
	14		未定义
	15		未定义

注 1. 不支持 Kintex-7 器件

UG480 表 4-6:独立 ADC 模式的位定义（49h）

顺序号	Bit	ADC 通道	说　明
2	0	16	辅助通道 0
3	1	17	辅助通道 1
4	2	18	辅助通道 2
5	3	19	辅助通道 3
6	4	20	辅助通道 4
7	5	21	辅助通道 5
8	6	22	辅助通道 6(1)
9	7	23	辅助通道 7(1)
10	8	24	辅助通道 8
11	9	25	辅助通道 9
12	10	26	辅助通道 10
13	11	27	辅助通道 11
14	12	28	辅助通道 12
15	13	29	辅助通道 13(1)
16	14	30	辅助通道 14(1)
17	15	31	辅助通道 15(1)

注 1. 不支持 Kintex-7 器件

四、Vivado 开发环境下使用 XADC Wizard 建立 XADC 工程项目

1. Vivado 和 SDK 开发工具简介

利用 Vivado 开发环境和 SDK 调试工具创建和调试 Zynq7000 应用项目方法如下。首先在 Vivado 开发环境下调用 *Xilinx* 提供的 IP 或者用户自己建立的 IP 构建实现用户功能的硬件平台，接着为建好的硬件平台生成顶层文件（wrapper），执行综合和实现，生成 bit 流文件。然后将硬件平台的设计和 bit 流文件传送给 SDK，并打开 SDK 调试工具。

然后在 SDK 中建立一个软件项目，该项目与 Vivado 输出的设计的硬件平台自动关联并生成一个板级支持包（bsp）。板级支持包中包含了硬件平台中所有 IP 的全部信息，如与 IP 相关的硬件设备的 ID、基地址、每个 IP 的变量和 API 函数的声明和定义。Vivado 提供并封装了能执行 IP 所有功能的 API 函数，用户只需在应用程序中为要调用的 IP 创建一个实例，执行实例的初始化后就能轻松调用这些 API 函数，执行所希望的功能。

Zynq7000 中的 PS（处理器）部分和 PL（FPGA 逻辑）部分的连接是通过 AXI 总线实现的。AXI 是一种高性能、高带宽、低延迟的片内总线，用以替代以前的 AHB 和 APB 总线。Zynq7000 中采用的是 AXI4 总线。AXI4 总线的架构虽然很复杂，但在 Vivado 中 PS 和 IP 之间的通信是由系统自动连接的，并把 AXI 时序所有细节都封装起来，对用户完全透明。用户不必关心细节，只需关注自己的应用。所以 Vivado 简化了开发过程，大大提高了工作效率。

2. XADC 项目举例

如图 6-1 所示，创建新项目名 XADC_Test，生成原理框图。首先添加“ZYNQ7 Processing System”IP 。然后在 ZYNQ7 的 Peripheral I/O Pins 配置项中将 MIO48 和 MIO49 分配给 UART1（因为笔者使用的 ZYNQ 开发板引出的串口是 UART1），波特率选择 115200。在 DDR Configuration 配置项中指定合适的 DDR3 芯片型号。然后添加“XADC Wizard” IP。双击 xadc_wiz_0 模块开始配置 XADC 功能，见图 6-1 和图 6-2。

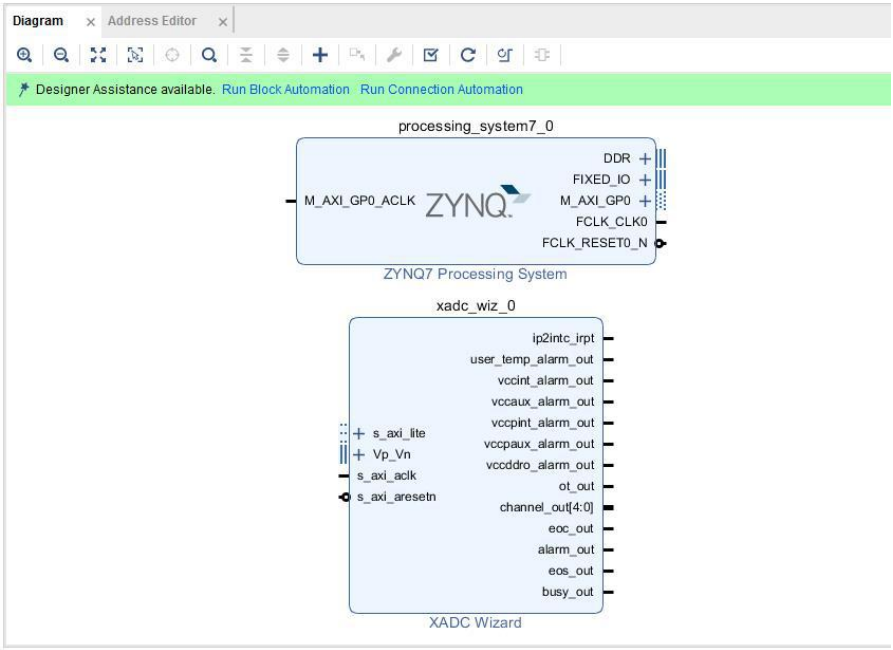


图 6-1 添加 ZYNQ7 和 XADC IP

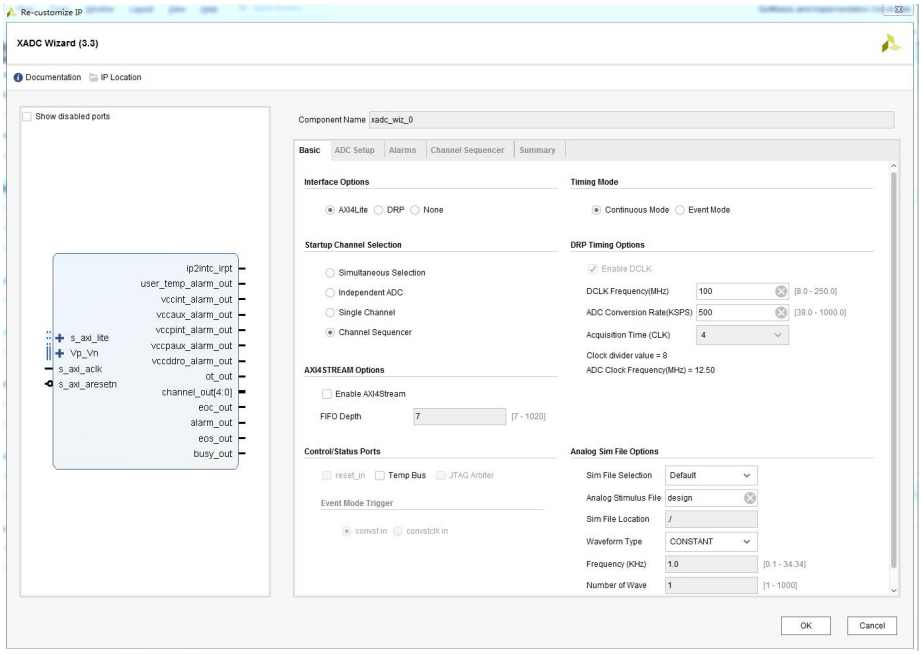


图 6-2 XADC 的配置页面

1. Basic 选项页

首先进入 XADC Wzard 的 Basic 选项页。有以下选项区：

⑴ Interface Option 选项区

此区域选择 XADC 的接口类型。可以选择 AXI4Lite，DRP 或 None，DRP 是默认选择。DRP 端口是 XADC 的 FPGA 逻辑接口。FPGA 使用此端口读写 XADC 控制寄存器。PS 通过 AXI4Lite 接口访问 XADC 寄存器。

⑵ Startup Channel Selection 选项区

- Simultaneous Selection，此模式允许同时监测两路外部模拟输入。
- Independent ADC，此模式允许用 ADC A 监测片内电压和温度，ADC B 监测外部模拟输入。
- Single Channel，此模式只能选择一个模拟输入通道。如果选择校准（Calibration）通道，仿真测试台不支持验证此项选择的仿真激励和数据比较。
- Channel Sequencer，此模式启用自动扫描所有的模拟通道。用户可以在 Channel Sequencer 选项页面中选择要监测的模拟通道。

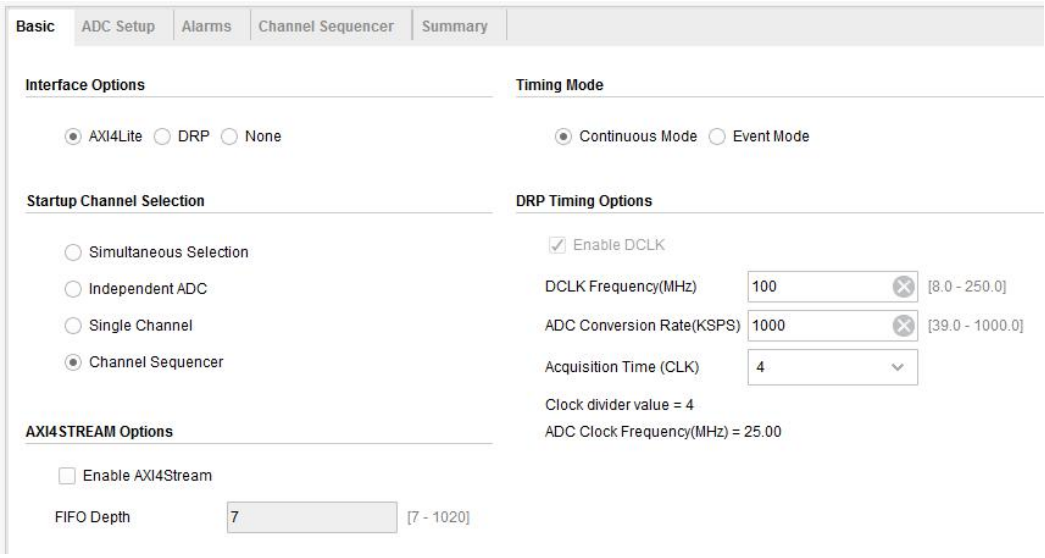


图 6-3 Basi 选项页

(3) AXI4-Stream Option，此选件用来启用或禁用 AXI4-Stream 接口。

- Enable AXI4-Stream 勾选框用来启用或禁用 AXI4-Stream 接口。默认情况下禁用。

- FIFO Depth 可以将 FIFO 深度配置为 7 到 1020。这里的 FIFO 深度是指在 ALMOSTFULL 变高之前可以写入 FIFO 的数据的最大深度。

(4) Timing Mode，XADC 可以在两种定时模式下工作：

- Continuous Mode，此模式下，XADC 对选定的模拟通道连续采样和 AD 转换。

- Event Mode，此模式需要一个外部触发器事件 CONVST 或 CONVSTCLK 启动所选通道上的 AD 转换。Event Mode 不适用片内传感器模拟通道。

(5) DRP Timing Option

XADC 时钟（ADCCLK）来自动态重组端口（DRP）的 DCLK 时钟。XADC 最高支持 250 兆赫的 DRP 时钟。XADC 也可以在没有 DCLK 的情况下操作。有关 DRP 详细信息参阅 7 series of the Series FPAS XADC user guide（UG480）。

ADCCLK 时钟应该在 4-26 MHz 范围内。为了获得这么低的时钟频率，XADC 内部设有一个分频器，Vivado IDE 允许指定外部 DCLK 频率和要求的 AD 转换速率（最大 1Msps）。根据给定的 DCLK 时钟，XADC Wizard 根据 DCLK 和 ADC 转换速率计算出合适的时钟分频值，并显示最终的 ADC 时钟频率和实际的 ADC 转换速率。

(6) Analog Sim Option，本选项可为仿真激励文件指定相对或绝对路径和更新名称，本文不详述。读者可自行参考 XADC Wizard V3.3 中的相关介绍。

2.ADC Setup 选项页

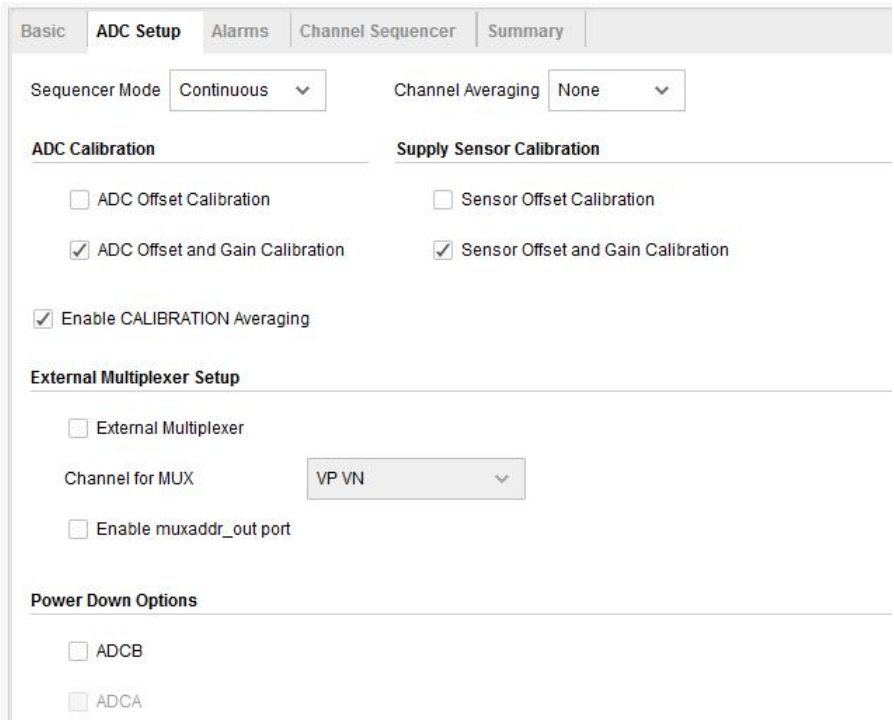


图 6-4 ADC Setup 选项页

⑴ Sequencer Mode，此选项的下拉菜单有 3 个选择

- Continuous
- Default
- One Pass

如果在 Basic 选项页中选择了 Channel Sequencer、 Simultaneous Sampling 或 Independent ADC 模式，此选项就会显示出来，用户可以选择所需的定序器模式。

⑵ Channel Averaging，下拉菜单允许选择所需的平均值。可用选项包括无、16、64 和 256。

⑶ ADC Calibration 和 Supply Sensor Calibration 选区中的复选框用来选择 ADC 校准和/或电源传感器校准的类型。

- Enable CALIBRATION Averaging 复选框在 XADC 向导中是默认启用的。用户可以禁用此选项。

⑷ External Multiplexer Setup，XADC 允许用户在 FPGA I/O 资源紧缺或将辅助模拟 I/O 用做其他接口时更有价值的情况下使用芯片外部的模拟多路器。此时应钩选“External Multiplexer”复选框来启用使用此功能。此外还需要指定用哪个 XADC 模拟通道做外部多路器的输入。右侧的下拉菜单用来选择此频道，可供选择的通道有 VPVN，VAUX0 至 VAUX15。

- Enable muxaddr_out port 选择框为采用 DRP 端口的外部 MUX 模式提供 muxaddr_out 使能。

⑸ Power Down Options，转换器 ADC B 和 ADC A 在不使用时可以断电。可以关闭 ADCB，只使用 ADCA。但只有当 ADC B 已经断电时，才能关闭 ADCA。此选项可用来节省用电，在断电时，ADC 不会生成任何控制和状态信号。

3. Alarm 选项业

Alarms 选项用来启用片上传感器的报警输出。用户可以为芯片上温度和电源指定报警阈值的上限和下限。如果测量值超出这些限制，报警逻辑的输出将激活。当测量值落在在这些限值之内时，报警将被被清除。Vivado IDE 中的默认限值是电源名义值的±5%。

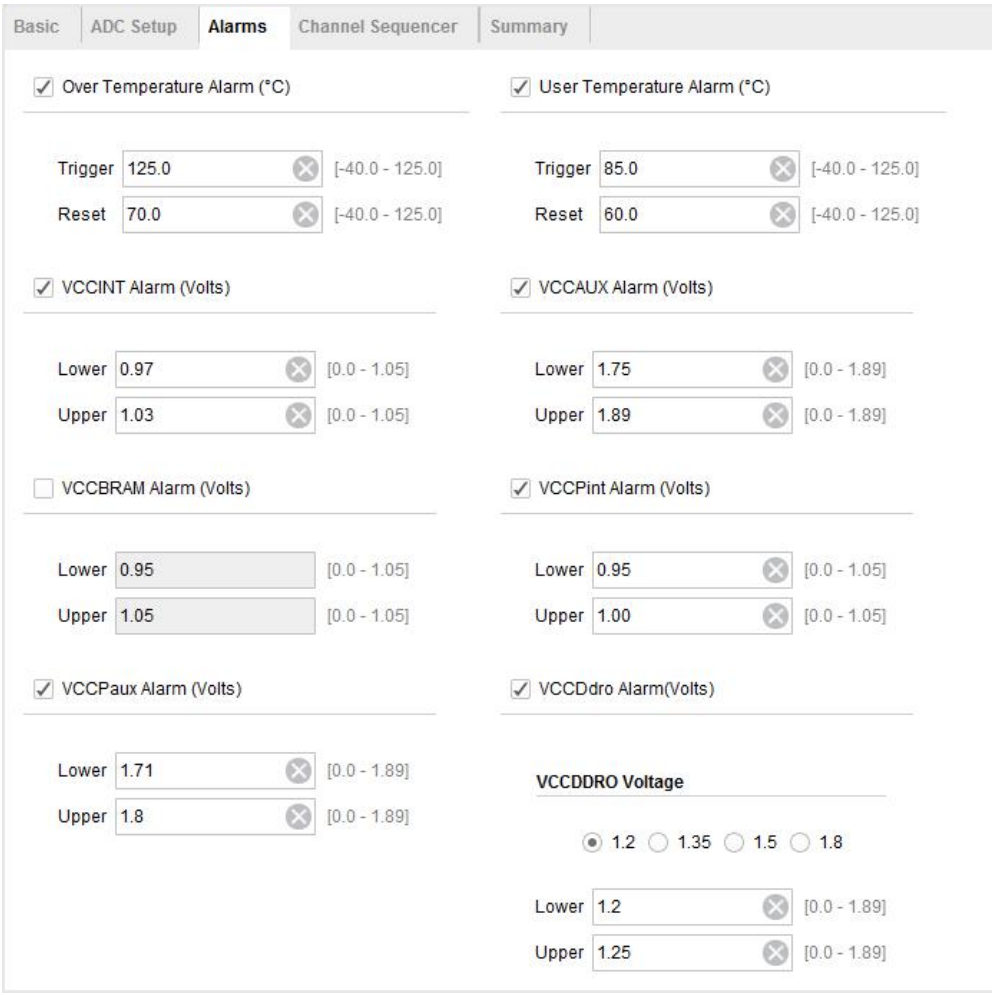


图 6-5 报警选项页

4. Channel Sequencer 选项页

(1) 在Basic选项页中，如果XADC配置成Channel Sequencer，Simultaneous sampling或 Independent ADC 模式时，此选项卡用于配置 XADC 通道序列寄存器。所有通道都列在 Wizard 的通道序列器表格中，如图 4-9 所示。

此页面列出了可供选择的所有通道，同时可为所选通道启用平均值，为外部通道启用双极性输入模式，并增加所选通道的采样时间。

在Simultaneous sampling mode时，选择通道va uxp[0]/va uxn[0]将自动选择通道va uxp[8]/va uxn[8]。同样，选择通道 vauxp[1]/vauxn[1]将选择通道 vauxp[9]/vauxn[9]等。

在 Independent ADC mode 时，则只列出外部通道（不显示内部传感器通道）供用户选择。

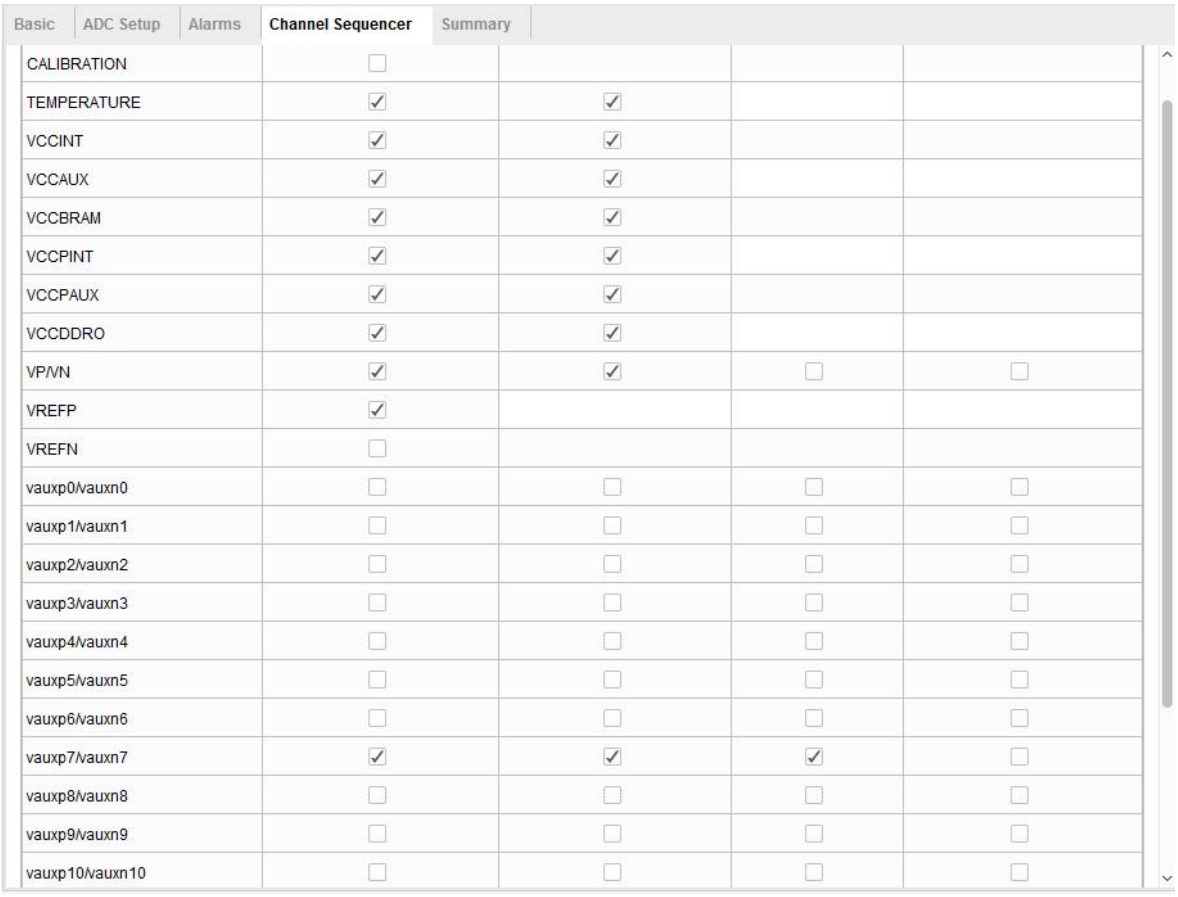


图 6-6 自动定序器的通道选择页面

(2) 单通道选择

如果在 Basic 选项页中选择了 Single Channel，Channel Sequencer 选项页的名称就变成了 Single Channel 选项页。这时页面显示成图 6-6。

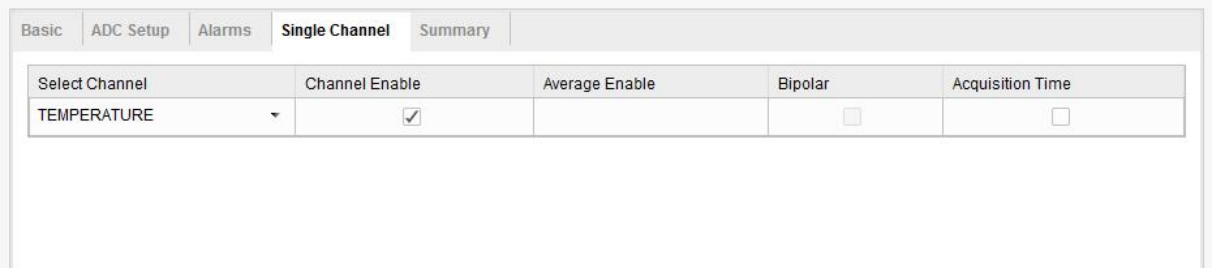


图 6-7 单通道的选择页面

用户可以从 Select Channel 的下拉菜单中选择要监测的通道。其他几个选项如 Channel Enable，Average Enable 和 Acquisition Time 将根据不同的选择被允许或禁止。

5.Summary 选项页

Summary 选项卡列出 XADC 设计的所有关键参数，如下图所示。

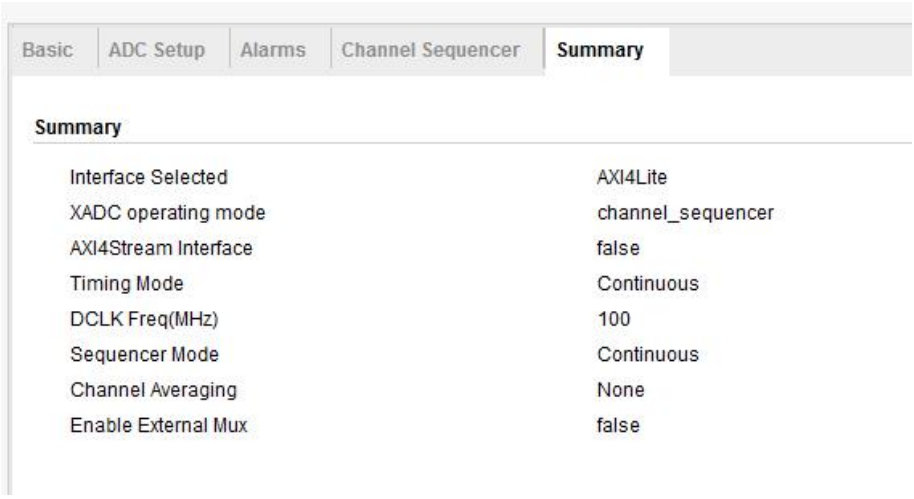


图 6-8 概括页面

6. 生成原理框图

完成配置后生成的框图如下所示

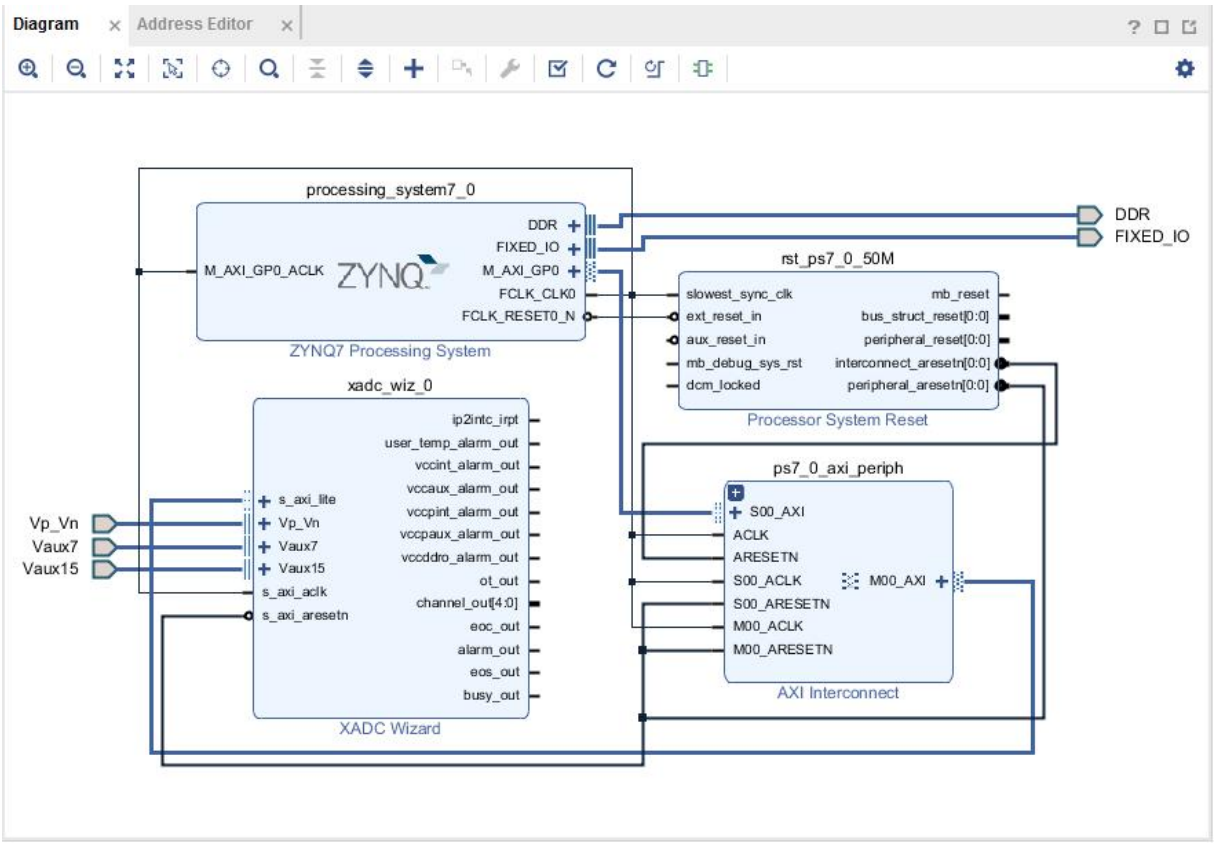


图 6-9 最终的原理框图

7. 生成和编辑 XDC 约束文件，生成输出产品（Create product）和顶层文件（wrapper）

XDC 约束文件如下：

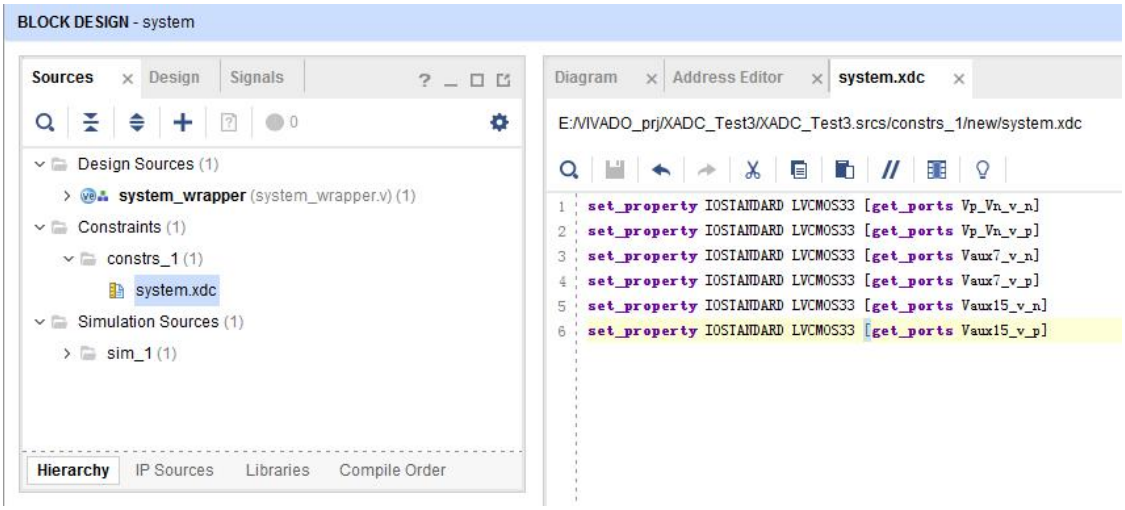


图 6-10 约束文件

8. 执行综合和实现

执行综合和实现，生成 bit 流文件。输出硬件和启动 SDK。

9. 在 SDK 中生成新项目，为 FPGA 下载 bit 流文件

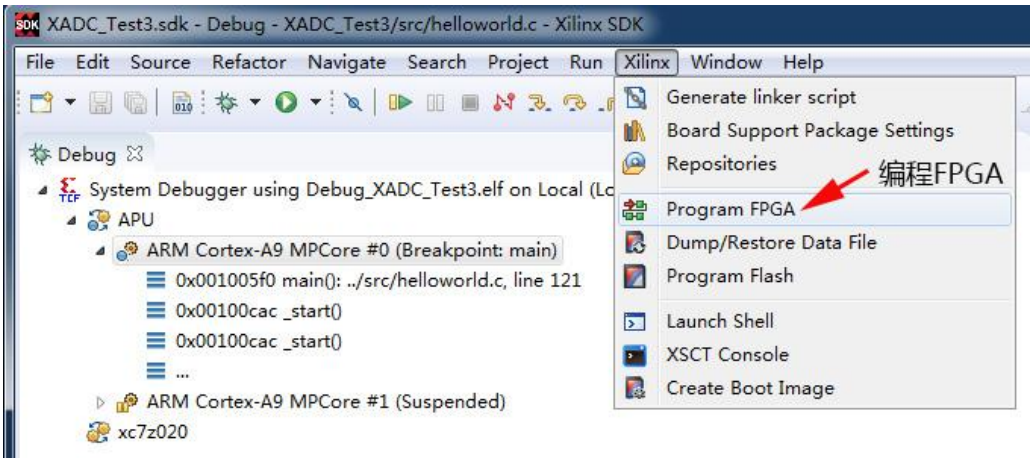


图 6-11 下载 bit 流

10. 编写 C 语言测试程序，并调试

下面的测试程序测试片内温度和电源传感器、专用模拟输入 VpVn、VREFP 和 VREFN、2 个辅助模拟通道 AUX7 和 AUX15。输出结果从串口打印显示（串口为 UART1，波特率 115200）。程序如下：

```
#include <stdio.h>
#include "xparameters.h"
#include "xadcps.h"
#include "xil_types.h"
#include "xil_printf.h"
#include "sleep.h"
#define XADCPS_CH_AUX_7      23        // 辅助模拟通道 7 的地址

int main()
{
    static XAdcPs    XAdcInst;              // 创建一个 XADC 的实例
    XAdcPs_Config    *XAdcInst_Cfg;      // 声明实例的配置表指针
    int Status;

    u32 VPVN_Adc;                       // 测试 Vp/Vn
    float VPVN_True;

    u32 VREFP_Adc;                      // 测试 VREFP 电压参考
    float VREFP_True;

    u32 Temp_Adc;                       // 测试 Temp
    float Temp_True;

    u32 VCCINT_Adc;                     // 测试 FPGA 内核电压 VCCINT
    float VCCINT_True;

    u32 VCCAUX_Adc;                     // 测试 FPGA 辅助电压 VCCAUX
    float VCCAUX_True;

    u32 VCCBRAM_Adc;                    // 测试 FPGA 的 RAM 电压
    float VCCBRAM_True;

    u32 VCCPINT_Adc;                    // 测试 PS 内核电压 VCCPINT
    float VCCPINT_True;

    u32 VCCPAUX_Adc;                    // 测试 PS 辅助 VCCPAUX
    float VCCPAUX_True;

    u32 VCCODDR_Adc;                    // 测试 DDR 电压 VCCODDR
```

```
float VCCODDR_True;

u32 AuxCH7_Adc;                    // 测试 AuxCH7 电压
float AuxCH7_True;

u32 AuxCH15_Adc;                   // 测试 AuxCH15 电压
float AuxCH15_True;

XAdcInst_Cfg = XAdcPs_LookupConfig(XPAR_PS7_XADC_0_DEVICE_ID);   // 检查实例配置表
if(XAdcInst_Cfg == NULL)
{
        return XST_FAILURE;
}

// 初始化实例
Status = XAdcPs_CfgInitialize(&XAdcInst, XAdcInst_Cfg, XAdcInst_Cfg->BaseAddress);
if(Status != XST_SUCCESS)
{
      return XST_FAILURE;
}

//设置连续转换

XAdcPs_SetSequencerMode(&XAdcInst, XADCPS_SEQ_MODE_CONTINPASS);

//选择采样通道

XAdcPs_SetSeqChEnables(&XAdcInst,
XADCPS_CH_TEMP|XADCPS_CH_VPVN|XADCPS_CH_VREFP|XADCPS_CH_VCCINT
|XADCPS_CH_VCCAUX|XADCPS_CH_VBRAM|XADCPS_CH_VCCPINT|XADCPS_CH_VCCPAUX
|XADCPS_CH_VCCPDRO|XADCPS_CH_AUX_7|XADCPS_CH_AUX_MAX);

while(1)                           //读取并打印测量数据
{
      Temp_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_TEMP);
      Temp_True = XAdcPs_RawToTemperature(Temp_Adc);
      printf("ADC_Temp %lu True_Temp %f \n", Temp_Adc, Temp_True);

      VPVN_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VPVN);
      VPVN_True = XAdcPs_RawToVoltage(VPVN_Adc);
      printf("ADC_VPVN %lu True_VPVN %f \n", VPVN_Adc, VPVN_True);

      VREFP_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VREFP);
      VREFP_True = XAdcPs_RawToVoltage(VREFP_Adc);
```

```
        printf("ADC_VREFP %lu True_VREFP %f \n", VREFP_Adc, VREFP_True);

        VCCINT_Adc= XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VCCINT);
        VCCINT_True = XAdcPs_RawToVoltage(VCCINT_Adc);
        printf("ADC_VCCINT %lu True_VCCINT %f \n", VCCINT_Adc,VCCINT_True);

        VCCAUX_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VCCAUX);
        VCCAUX_True = XAdcPs_RawToVoltage(VCCAUX_Adc);
        printf("ADC_VCCAUX %lu True_ACCAUX %f \n", VCCAUX_Adc,VCCAUX_True);

        VCCBRAM_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VBRAM);
        VCCBRAM_True = XAdcPs_RawToVoltage(VCCBRAM_Adc);
        printf("ADC_VCCBRAM %lu True_VCCBRAM %f \n", VCCBRAM_Adc, VCCBRAM_True);

        VCCPINT_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VCCPINT);
        VCCPINT_True = XAdcPs_RawToVoltage(VCCPINT_Adc);
        printf("ADC_VCCPINT %lu VccPInt %f \n", VCCPINT_Adc, VCCPINT_True);

        VCCPAUX_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VCCPAUX);
        VCCPAUX_True = XAdcPs_RawToVoltage(VCCPAUX_Adc);
        printf("ADC_VCCPAUX %lu True_VCCPAUX %f \n", VCCPAUX_Adc, VCCPAUX_True);

        VCCODDR_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_VCCPDRO);
        VCCODDR_True = XAdcPs_RawToVoltage(VCCODDR_Adc);
        printf("ADC_VCCODDR %lu True_VCCODDR %f \n", VCCODDR_Adc, VCCODDR_True);

        AuxCH7_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_AUX_7);
        AuxCH7_True = XAdcPs_RawToVoltage(AuxCH7_Adc);
        printf("ADC_AuxCH7 %lu True_AuxCH7 %f \n", AuxCH7_Adc, AuxCH7_True);

        AuxCH15_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_AUX_MAX);
        AuxCH15_True = XAdcPs_RawToVoltage(AuxCH15_Adc);
        printf("ADC_AuxCH15 %lu True_AuxCH15 %f \n", AuxCH15_Adc, AuxCH15_True);

        usleep(100 * 100 );
    }
    return 0;
}
```

11. 程序详解

Wivado 的 IP 中为每个 IP 都编写了完备的应用函数 API。用户只需为已在 Vivado 硬件平台中设计好的硬件 IP 创建实例，完成初始化就可以轻松地调用 Vivado 提供的 API 函数完成各种操作。

当用户在 Vivado 开发环境中已经加进了 *Xilinx* 提供的的 IP 后，进入 SDK 在“xparameters.h”头文件中就已经定义了与该 IP 相关的硬件设备的 ID、基地址等参数。以本例 XADC 为例，在 xparameters.h 头文件中就能找到以下定义：

```
/* Definitions for driver XADCPS */
#define XPAR_XADCPS_NUM_INSTANCES 1

/* Definitions for peripheral PS7_XADC_0 */
#define XPAR_PS7_XADC_0_DEVICE_ID 0
#define XPAR_PS7_XADC_0_BASEADDR 0xF8007100
#define XPAR_PS7_XADC_0_HIGHADDR 0xF8007120
```

上述定义的含义是：硬件平台已经创建了一个 XADC 实例，该实例的设备 ID 号是 0。同时列出了该实例的基地址和地址范围。在 SDK 中用户想调用这个实例，可按以下流程操作。

⑴ 创建一个 XADC 结构体的实例

```
static XAdcPs    XAdcInst;              // 创建一个 XADC 的实例
```

XADC 的结构体如下：

```
typedef struct {
        XAdcPs_Config    Config;        /**< XAdcPs_Config of current device */
        u32    IsReady;                 /**< Device is initialized and ready    */
} XAdcPs;
```

⑵ 为实例中的 XAdcPs_Config 声明一个指针，指向设备的 ID。

```
XAdcPs_Config    *XAdcInst_Cfg;      // 声明实例的配置表指针
```

XAdcPs_Config 也是一个结构体中，是一张存放与 XADC/ADC 有关的初始化信息的配置表格。

```
typedef struct {
      u16    DeviceId;          /**< Unique ID of device */
      u32    BaseAddress;      /**< Device base address */
} XAdcPs_Config;
```

⑶ 查找实例的配置表，为初始化实例做准备工作

头文件 "xadcps.h" 中包含了所有 XADC 的 API 函数（xadcps.c），下面是执行查找配置表的函数。

```
XAdcInst_Cfg = XAdcPs_LookupConfig(XPAR_PS7_XADC_0_DEVICE_ID);
```

用户程序执行上面这条语句查找实例的配置信息表时，在 xparameters.h 头文件中就找到了 XPAR_PS7_XADC_0_DEVICE_ID=0 并指向了实例的基地址 XPAR_PS7_XADC_0_BASEADDR 0xF8007100。

⑷ 执行实例初始化

```
Status = XAdcPs_CfgInitialize(&XAdcInst, XAdcInst_Cfg, XAdcInst_Cfg->BaseAddress);
```

用**&XAdcInst** 指明实例的地址，并指向实例结构体和配置表的基地址就顺利完成初始化了。

⑸ 调用 API 函数

"xadcps.h"头文件中包含了所有 XADC 的 API 函数，调用 XADC IP 的其他 API 函数的方法很简单，直接用**&XAdcInst** 指明实例的地址就可以了。

① 如下面调用的函数指定 XADC 定序器的操作模式：

```
XAdcPs_SetSequencerMode( &XAdcInst, XADCPS_SEQ_MODE_CONTINPASS);
```

采样模式有以下这些，前面章节已经详细介绍过：

```
Default safe mode                (XADCPS_SEQ_MODE_SAFE)
One pass through sequence        (XADCPS_SEQ_MODE_ONEPASS)
Continuous channel sequencing    (XADCPS_SEQ_MODE_CONTINPASS)
Single Channel/Sequencer off     (XADCPS_SEQ_MODE_SINGCHAN)
Simulataneous sampling mode      (XADCPS_SEQ_MODE_SIMUL_SAMPLING)
Independent mode                 (XADCPS_SEQ_MODE_INDEPENDENT)
```

② 开通选择的模拟通道

下面的函数指定需要开通的模拟通道

```
XAdcPs_SetSeqChEnables(&XAdcInst, XADCPS_CH_TEMP||XADCPS_CH_VPVN|XADCPS_CH_VREFP|.... );
```

模拟通道地址如下：

```
#define XADCPS_CH_TEMP          0x0         /**< On Chip Temperature */
#define XADCPS_CH_VCCINT        0x1         /**< VCCINT */
#define XADCPS_CH_VCCAUX        0x2         /**< VCCAUX */
#define XADCPS_CH_VPVN          0x3         /**< VP/VN Dedicated analog inputs */
#define XADCPS_CH_VREFP         0x4         /**< VREFP */
#define XADCPS_CH_VREFN         0x5         /**< VREFN */
#define XADCPS_CH_VBRAM         0x6         /**< On-chip VBRAM Data Reg, 7 series */
```

```
#define XADCPS_CH_SUPPLY_CALIB      0x07        /**< Supply Calib Data Reg */
#define XADCPS_CH_ADC_CALIB         0x08        /**< ADC Offset Channel Reg */
#define XADCPS_CH_GAINERR_CALIB     0x09        /**< Gain Error Channel Reg   */
#define XADCPS_CH_VCCPINT           0x0D        /**< On-chip PS VCCPINT Channel , Zynq */
#define XADCPS_CH_VCCPAUX           0x0E        /**< On-chip PS VCCPAUX Channel , Zynq */
#define XADCPS_CH_VCCPDRO           0x0F        /**< On-chip PS VCCPDRO Channel , Zynq */
#define XADCPS_CH_AUX_MIN           16          /**< Channel number for 1st Aux Channel */
#define XADCPS_CH_AUX_MAX           31          /**< Channel number for Last Aux channel */
```

打开其他 AUX 通道，只需将通道号加 16 就可以，例如打开通道 7 可如下定义：

```
#define XADCPS_CH_AUX_7      23              // 7+16=23
```

③ 读取 AD 转换数据，下面是读取指定通道的 AD 转换结果。

```
Temp_Adc = XAdcPs_GetAdcData(&XAdcInst, XADCPS_CH_TEMP);
```

12. 观察测试结果

打开计算机上的 SecureCRT 终端仿真程序，波特率设置成 115200。

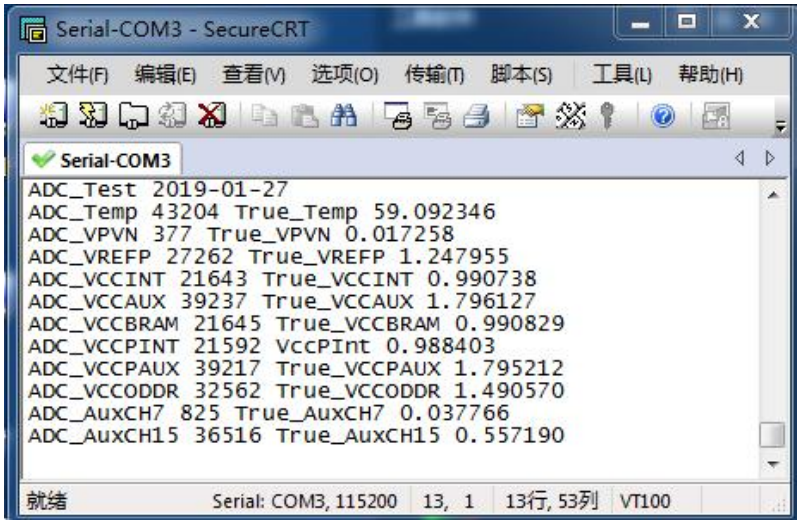


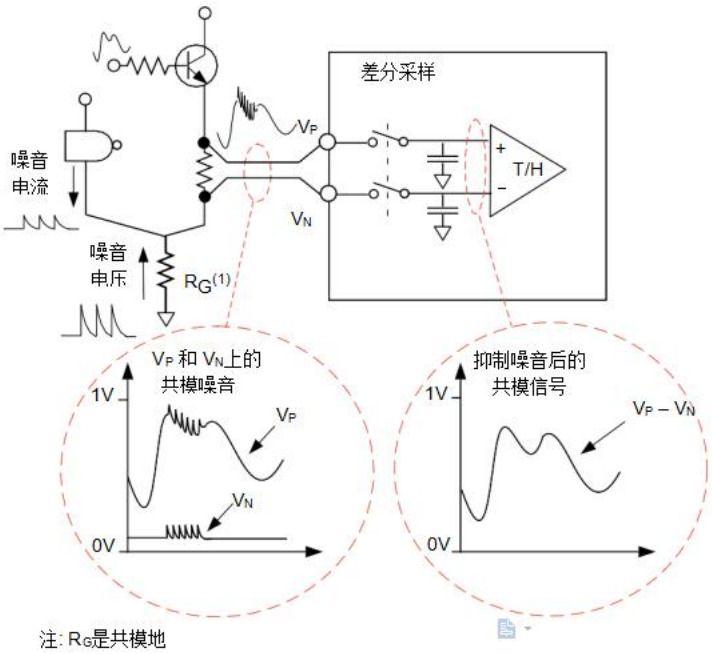
图 6-12 打印测试结果

说明，测试 PCB 板上的辅助模拟通道 AuxCH15 的输入端加了 1 个 0.555V 的电压（万用表测量），XADC 测量结果是 0.557V。PCB 的 Vp/Vn2 个输入端和 GNDADC 相接，AuxCH7 通道开路，所以这 2 个通道的测量结果都接近 0V。

五、XADC 的模拟输入说明

1. 专用模拟输入 VP/VN

专用模拟输入 VP/VN 采用差分采样方案，以减少共模噪声信号的影响。PCB 的公共地阻抗将数字电路的开关电流噪音耦合到模拟系统，其幅度可达 100 毫伏或更高。这对于 ADC 来说相当于数百个 LSB，因此会导致很大的测量误差。差分采样方案在两个输入端（VP/VN）对含共模噪声的信号进行采样。ADC 的前置放大器的高共模抑制比有效地抑制了共模噪声信号，送到 ADC 输入口的只剩下 VP 和 VN 之间的差模信号。所以用户如果要获得高共模抑制比，应采用专用模拟输入通道 VP/VN。



UG480 图 2-4 共模噪音抑制

2.辅助模拟输入通道

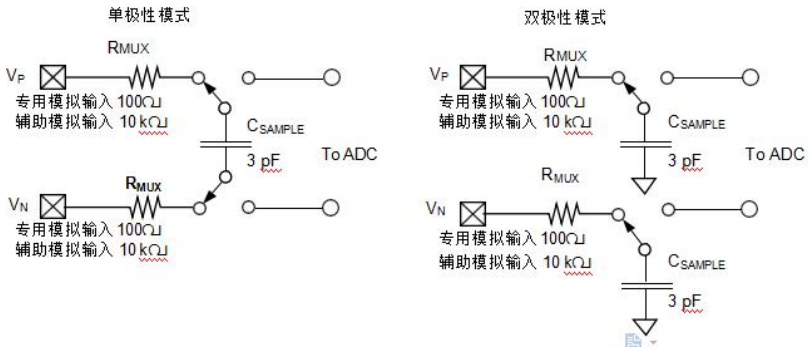
辅助模拟输入（VAUXP[15:0]和 VAUXN[15:0]）是与常规数字 I/O 引脚共享的模拟输入引脚。

辅助模拟输入通道引脚的 IOSTANDARD 必须与所在 BANK 的数字 I/O 标准所要求的电压兼容（在 XDC 约束文件中声明），其输入的模拟信号幅度不应超过数字 I/O BANK 的电源电压。只有被启用为辅助模拟输入的引脚才具有模拟输入属性,其他引脚仍然保留数字 I/O 属性。

UG480 图 2-5 是辅助模拟输入通道单极性和双极性输入模式的等效电路。

单极性输入时只有一个采样电容，双极性输入时有 2 个采样电容。在 ADC 采集阶段，采样开关闭合，采样电容充电至模拟输入信号源的电压，然后采样开关切换连接 ADC。

采样电容器充电至最终值（达到输入信号值±0.5 LSBs）所需的时间取决于采样电容器的大小、模拟多路开关的电阻和外部信号源阻抗。



UG480 图 2-5 模拟输入的等效输入电路

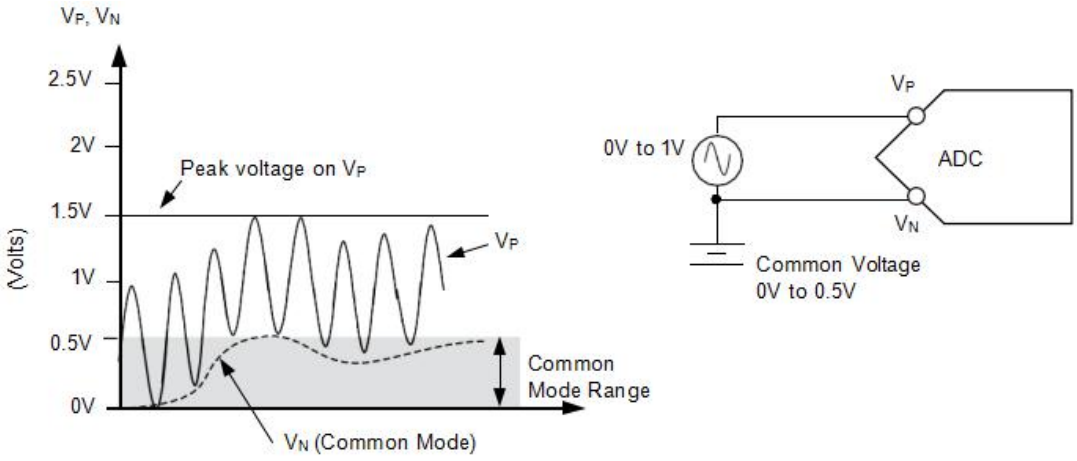
根据测算，专用模拟通道（VP/VN)所需最小采样时间为 3ns，而辅助模拟通道(VAUXP[15:0]和 VAUXN[15:0])因为有一个大约 10KΩ的的开关电阻，最小采样时间是 300ns。任何额外的外部电阻（如去假频滤波器或电阻分压器）都会增加采集时间要求。

2. 单极性输入要求

测量单极性模拟输入信号时，ADC 必须设置成单极性输入模式。通过写入配置寄存器 0 来选择此模式。单极性输入时，以专用模拟输入（VP/VN）为例，信号必须满足以下要求：

- VP 和 VN 必须始终高于 GNDADC
- VP 必须始终高于 VN
- VP 和 VN 之间最大幅度差是 1.0V（ADC 的满标电压）
- VN 上的共模信号可以在 0 伏到+0.5 伏之间变化（相对于 GNDADC 测量），所以 VP 上的最大信号不得高于+1.5 伏。
- VN 通常连接到本地接地或共模信号上

UG480 图 2-6 说明了这种关系。



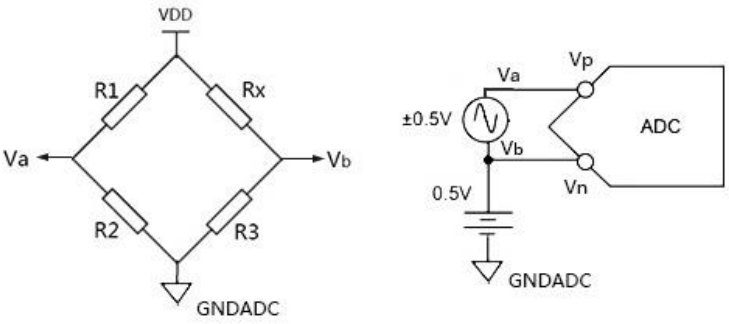
UG480 图 2-6 单极性输入信号要求

6.双极性输入要求

双极性输入信号具有以下属性：

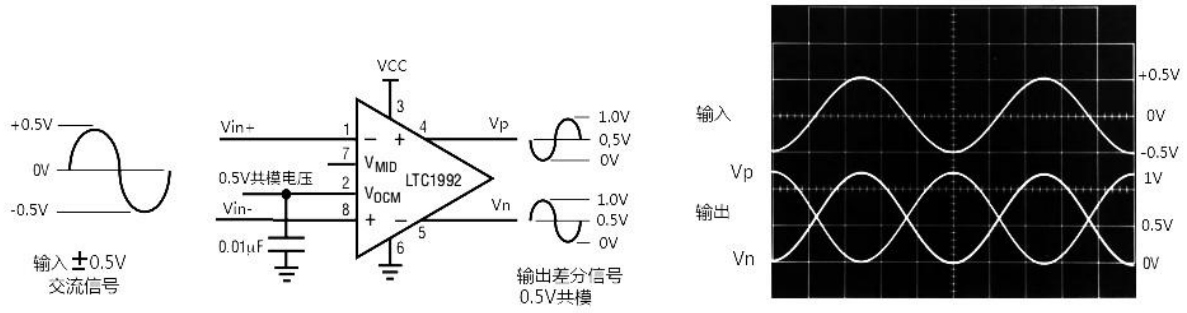
- VP 和 VN 相对关系不总是 VP>VN,两者的关系是交替变化的。VP 可能高于 VN，但 VP 也可能低于 VN，但 VP 和 VN 都必须高于地电平。例如平衡电桥的输出信号就具有这种属性，但 2 个输出端上有一个高于公共地的共模电压。
- VP 和 VN 是一个相对于 GNDADC 的交流电压。这种情况就必须设法引入一个共模电压，将 VP 和 VN 都拉到 GNDADC 电平以上。

在上述第一种情况，因为 XADC 的满标电压是 1.0V，共模电压不应超过 0.5 V。以平衡电桥为例，电桥的 2 个输出上的共模电压应该是 0.5V，如下图所示。将平衡电桥的一个输出连接到 VN，VP 在 VN 的上下摆动，其幅值<±0.5V。



平衡电桥的双极性输入

VP 和 VN 如果是相对于 GND 的交流信号，则可以用 LT1991-1（增益=1）将其转换成双极性差分输出信号，并在 LT1991-1 的 VCOM 引脚上加 1 个 0,5V 的共模电压。结果输出就是在 0.5V 共模电压基线上变化的差分信号。见下图。



交流输入的双极性转换

六、参考资料

1. *Vivado Design Suite User Guide: Designing with IP (**UG896**)*
2. *AM BA AXI4-Stream Protocol Specification*
3. *7 Series FPGAs XADC User Guide (**UG480**)*
4. *XADC Wizard v3.3 LogicCORE IP Product Guide（**PG091**）*
5. *Vivado Design Suite User Guide: Getting Started (**UG910**)*
6. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator (**UG994**)*
7. *Vivado Design Suite User Guide: Logic Simulation (**UG900**)*
8. *7 Series FPGAs Overview (**DS180**)*
9. *ISE to Vivado Design Suite Migration Guide (**UG911**)*
10. *Vivado Design Suite User Guide: Programming and Debugging (**UG908**)*
11. *XADC Wizard Release Notes*
12. *LogiCORE IP XADC Wizard User Guide (**UG772**)*
13. *LogiCORE IP AXI4-Lite IPIF Product Guide (**PG155**)*