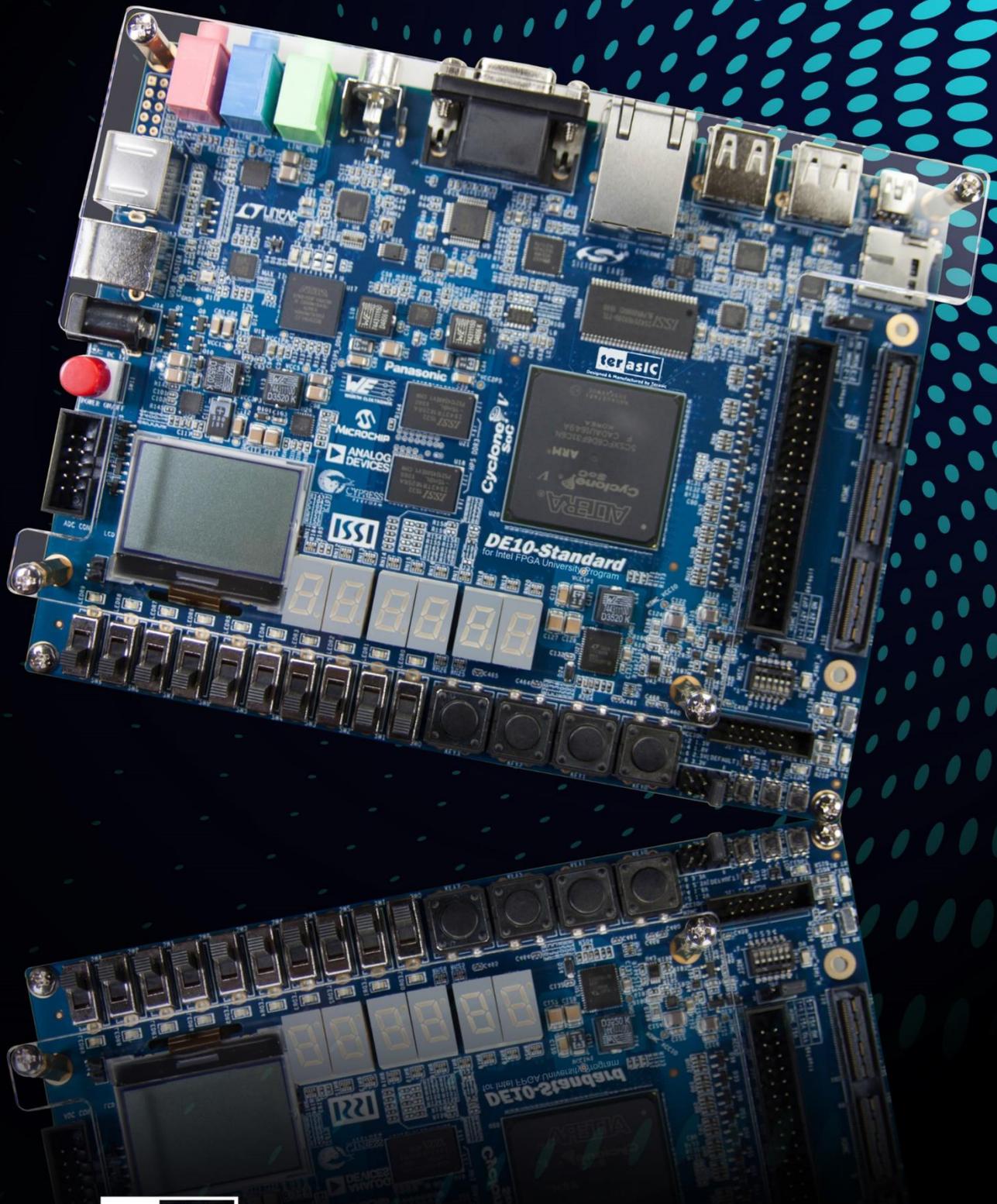


# DE10-Standard

## USER MANUAL



<b>第一章 DE10-Standard 套件</b> .....	<b>4</b>
1.1 包装内容.....	4
1.2 DE10-Standard 系统 CD.....	5
1.3 获得帮助.....	5
<b>第二章 DE10-Standard 开发板简介</b> .....	<b>6</b>
2.1 开发板布局与组件 .....	6
2.2 DE10-Standard 开发板系统框图 .....	8
<b>第三章 使用 DE10-Standard 开发板</b> .....	<b>12</b>
3.1 设置 FPGA 配置模式.....	12
3.2 配置 DE10-Standard 开发板的 Cyclone V SoC FPGA.....	13
3.3 开发板状态指示组件 .....	18
3.4 开发板复位组件 .....	18
3.5 时钟电路.....	20
3.6 FPGA 外围设备.....	21
3.6.1 按钮开关、拨动开关与 LED.....	21
3.6.2 七段数码管.....	24
3.6.3 2x20 GPIO 扩展接口.....	26
3.6.4 HSMC 接口.....	27
3.6.5 24 位音频编解码器.....	30
3.6.6 I2C 多路复用器.....	31
3.6.7 VGA 输出.....	32
3.6.8 TV 解码器.....	35
3.6.9 红外接收器.....	36
3.6.10 红外发射器 LED.....	37
3.6.11 SDRAM 存储器.....	37
3.6.12 PS/2 串行接口.....	39

3.6.13	A/D 转换器与 2x5 引脚接头.....	40
3.7	HPS 外围设备.....	41
3.7.1	按钮开关和 LED.....	41
3.7.2	千兆以太网.....	42
3.7.3	UART 转 USB.....	43
3.7.4	DDR3 存储器.....	44
3.7.5	Micro SD 卡槽.....	46
3.7.6	USB 主设备端口.....	47
3.7.7	加速器 (G-sensor).....	48
3.7.8	LTC 连接头.....	48
3.7.9	128x64 点阵 LCD.....	49
<b>第四章</b>	<b>DE10-Standard System Builder .....</b>	<b>51</b>
4.1	简介.....	51
4.2	设计流程.....	51
4.3	使用 DE10-Standard System Builder.....	52
<b>第五章</b>	<b>FPGA 设计示例.....</b>	<b>59</b>
5.1	DE10-Standard 默认配置 .....	59
5.2	音频录制与回放 .....	60
5.3	卡拉 OK 机.....	62
5.4	Nios II SDRAM 测试.....	64
5.5	Verilog SDRAM 测试 .....	66
5.6	电视盒示例 .....	68
5.7	电视盒示例 (VIP).....	70
5.8	PS/2 鼠标示例 .....	73
5.9	红外发射器 LED 和接收器示例.....	75
5.10	ADC 读取.....	80
<b>第六章</b>	<b>HPS SoC 设计示例.....</b>	<b>85</b>
6.1	Hello World 程序 .....	85
6.2	用户 LED 与 KEY .....	87
6.3	I2C 接口 G-sensor.....	92
6.4	I2C MUX 测试.....	94

6.5 SPI 接口 Graphic LCD .....	96
6.6 设置 USB Wi-Fi Dongle .....	100
6.7 查询网络时间 .....	102
<b>第七章 FPGA 和 HPS 协同工作的示例.....</b>	<b>104</b>
7.1 需求背景.. .....	104
7.2 系统需求.. .....	104
7.3 Intel SoC FPGA 内的 AXI Bridge.....	105
7.4 GHRD 工程.....	106
7.5 编译与编程 .....	107
7.6 开发 C 代码.....	108
<b>第八章 固化 EPCS 器件 .....</b>	<b>114</b>
8.1 固化前的设置 .....	114
8.2 将.sof 文件转换为.jic 文件 .....	114
8.3 将.jic 文件写入 EPCS 器件 .....	117
8.4 擦除 EPCS 器件 .....	119
<b>第九章 Linux BSP. ....</b>	<b>121</b>
9.1 使用 Linux BSP.....	121
9.2 Linux Console BSP .....	122
9.3 Linux LXDE Desktop BSP .....	123
9.4 OpenCL BSP.....	124

# 第一章

## DE10-Standard 套件

DE10-Standard套件提供了以Intel System-on-Chip (SoC) FPGA建立的强大的硬件设计平台，结合了最新的嵌入式双核Cortex-A9和业界领先的可编程逻辑以满足终极设计的灵活性。用户可以使用这个搭配了高性能和低功率处理系统的可重构性力量强大的平台。Intel SoC集成了基于ARM的HPS架构处理器、外设及存储器接口与使用高带宽互联骨干结构的FPGA无缝接合。DE10-Standard开发板包含了诸如高速DDR3内存、视频和音频、以太网等功能硬件，用来实现很多令人兴奋的应用。

DE10-Standard开发套件包含了在Windows XP或者更高阶系统的PC上使用该开发板时所需的所有工具。

### 1.1 包装内容



图 1-1 DE10-Standard 套件包装内容

DE10-Standard 套件包装包含以下内容：

- DE10-Standard开发板
- DE10-Standard快速入门指南
- 用于FPGA编程及控制的A型转B型USB线
- 用于UART控制的A型转Mini-B型USB线
- 12V直流电源适配器
- 4个硅胶脚套

## 1.2 DE10-Standard 系统 CD

DE10-Standard 系统 CD 包含了与 DE10-Standard 开发板相关的所有文档和支持材料，包括用户手册、System Builder、参考设计和器件数据表。用户可以从 <http://de10-standard.terasic.com/cd/>链接下载此系统 CD。

## 1.3 获得帮助

遇到任何问题您可以从以下处获得帮助：

- Terasic Technologies 9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070.

Taiwan

- Email: support@terasic.com.cn
- Tel.: +886-3-575-0880
- Website: www.terasic.com.cn

# 第二章

## DE10-Standard 开发板简介

这一章将介绍开发板的各部分组件及其设计特性。

### 2.1 开发板布局与组件

图 2-1 和图 2-2 所示为开发板的全貌，它描述了开发板的组件配置与布局，并标注出连接器 and 关键组件的位置。

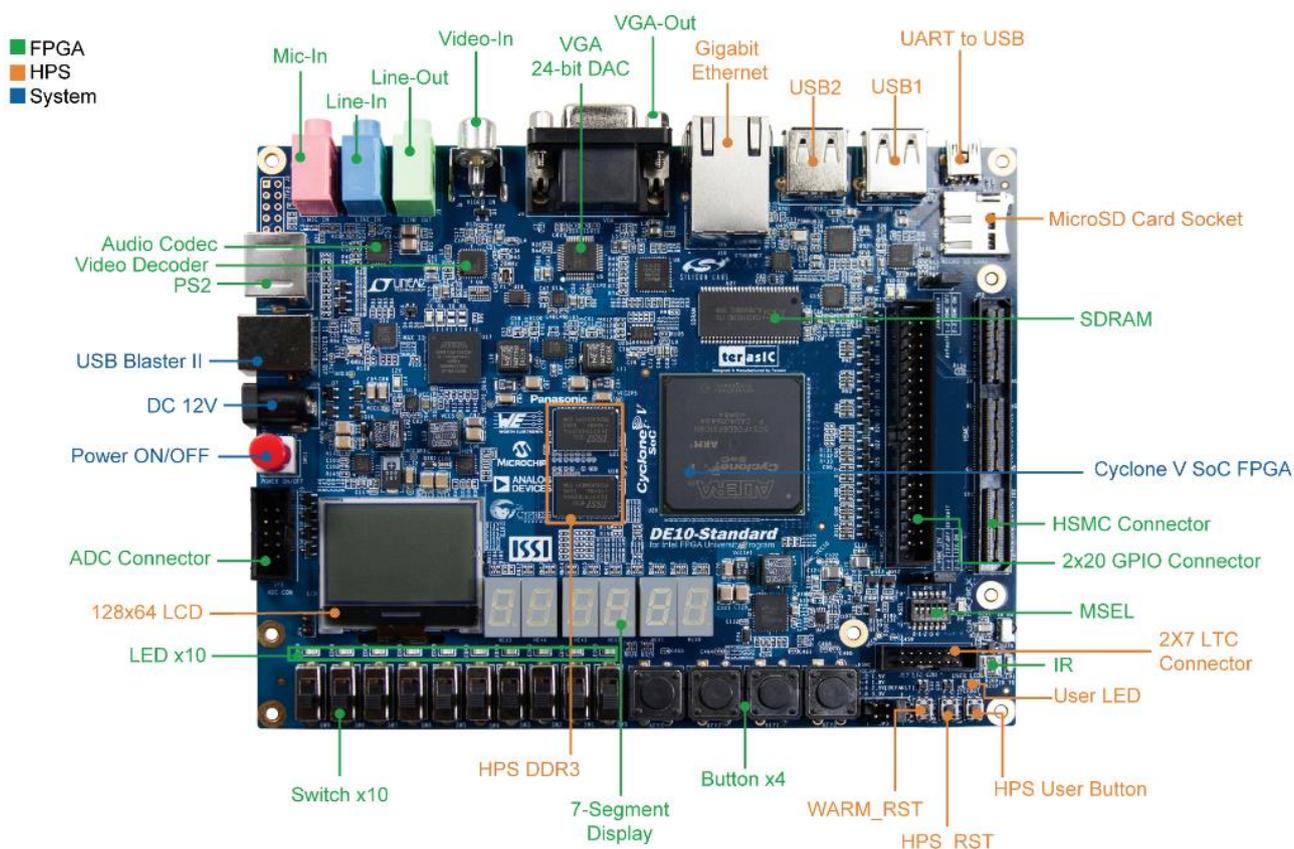


图 2-1 DE10-Standard 开发板 (正面视图)

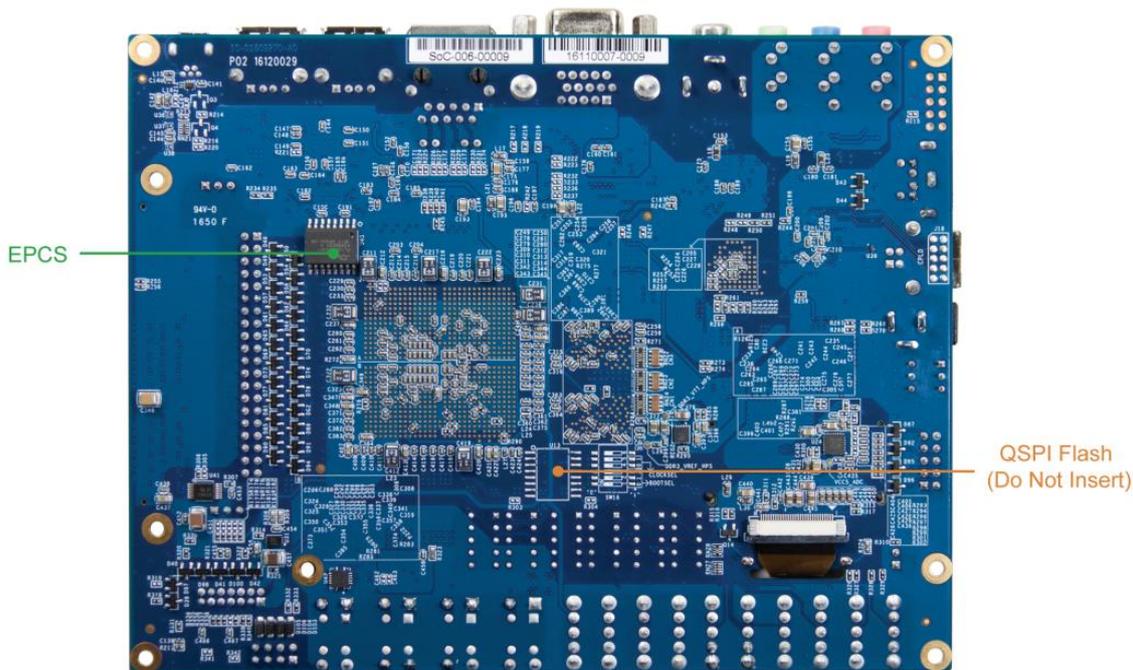


图 2-2 DE10-Standard 开发板(背面视图)

DE10-Standard 开发板拥有许多能够为用户实现广泛设计电路的特征，包括从简单的电路到各种多媒体项目的设计。

它提供了以下硬件资源：

## ■ FPGA

- Intel Cyclone® V SX 5CSXFC6D6F31C6N 芯片
- Intel 串行配置芯片—EPCS128
- 板载 USB-Blaster II 下载电路；支持 JTAG 模式
- 64MB SDRAM (16位数据总线)
- 4个按钮开关
- 10个拨动开关
- 10个红色LED
- 6个七段数码显示管
- 时钟发生器产生的4个50MHz时钟源
- 24位CD音质音频编解码器，具备line-in、line-out和microphone插孔
- VGA DAC (8位、高速三通道)，带VGA输出接口
- TV解码器(NTSC/PAL/SECAM)和TV输入接口

- PS/2鼠标/键盘接口
- 红外发射器和红外接收器
- 1个HSMC接口，可配置的I/O标准为1.5/1.8/2.5/3.3V
- 1个40引脚扩展接口，带二极管保护电路
- A/D转换器，与FPGA相连的4引脚SPI接口

## ■ HPS (硬核处理器系统)

- 925MHz ARM Cortex-A9 MP Core双核处理器
- 1GB DDR3 SDRAM (32位数据总线)
- 千兆以太网PHY，带RJ45接口
- 2个USB主设备端口，普通USB A型接口
- Micro SD卡槽
- 加速器 (I2C接口+中断)
- UART转USB, USB Mini-B连接头
- 热、冷复位按钮
- 1个按钮开关、1个LED
- LTC 2x7扩展接头
- 128x64点阵式背光LCD显示屏

## 2.2 DE10-Standard 开发板系统框图

图 2-3 所示为开发板的系统框图。该开发板基于 Cyclone V SoC FPGA 器件的所有硬件连接都是为了给用户提供最大的灵活性。用户可以配置 FPGA 来实现任何系统设计。

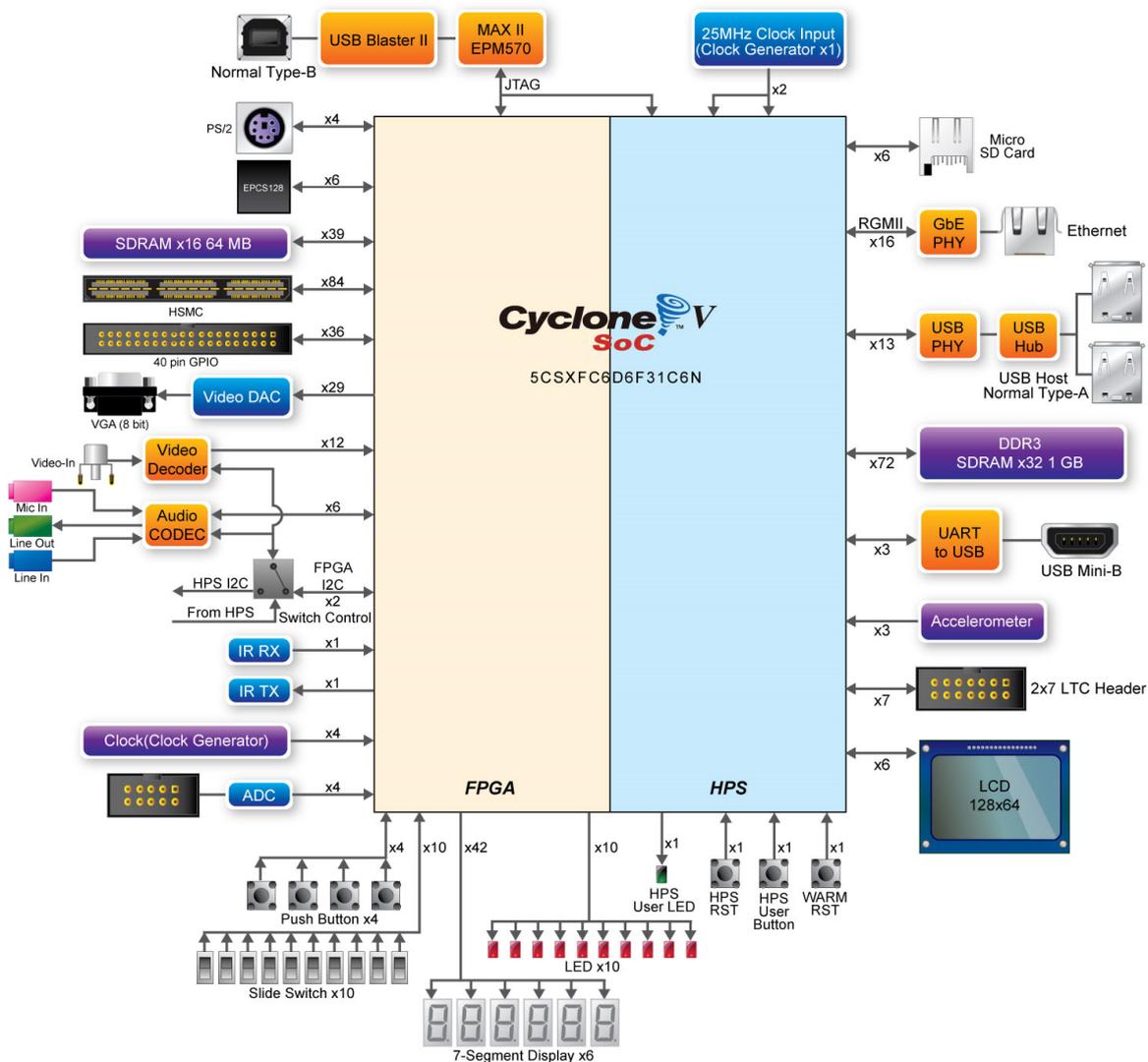


图 2-3 DE10-Standard 开发板系统框图

关于图 2-3 的详细信息如下所示：

## ■ FPGA 器件

- Cyclone V SoC 5CSXFC6D6F31C6N 芯片
- 双核 ARM Cortex-A9 (HPS)
- 110K 可编程逻辑单元
- 5,140 Kbits 嵌入式内存
- 6 个分频锁相环
- 2 个硬核存储控制器

## ■ 配置与调试

- FPGA端四路串行配置芯片– EPCS128
- 板载USB-Blaster II (普通USB B型接口)

## ■ 存储器器件

- FPGA端64MB (32Mx16) SDRAM
- HPS端1GB (2x256Mx16) DDR3 SDRAM
- HPS端Micro SD卡槽

## ■ 通讯

- 2个USB 2.0主设备端口(配备USB A型连接头的ULPI接口)
- HPS端USB转UART接口 (USB Mini-B型连接头)
- HPS端10/100/1000Mbps以太网接口
- PS/2鼠标/键盘接口
- 红外发射、接收器
- I2C多路复用器

## ■ 连接头

- 1个HSMC接口, 可配置的I/O标准为1.5/1.8/2.5/3.3V
- 1个40引脚扩展接口
- 1个10引脚ADC输入接口
- 1个LTC连接头(1个串行外设接口(SPI) 主设备, 1个I2C和1个GPIO接口)

## ■ 显示器

- 24位VGA DAC
- HPS端128x64点阵式背光LCD显示屏

## ■ 音频

- 24位CODEC, line-in、line-out和microphone插孔

## ■ 视频输入

- TV解码器(NTSC/PAL/SECAM)和TV输入接口

## ■ 模数转换器

- SPI接口
- 快速吞吐率：500KSPS
- 8通道
- 12位分辨率
- 模拟输入范围：0 ~ 4.096 V

## ■ 拨动开关、按钮开关、指示器

- 5个用户按钮开关(FPGA端4个, HPS端1个)
- FPGA端10个用户拨动开关
- 11个LED (FPGA端10个, HPS端1个)
- 2 个HPS复位按钮 (HPS\_RESET\_n 和HPS\_WARM\_RST\_n)
- 6个七段数码管

## ■ 传感器

- HPS端重力传感器

## ■ 电源

- 12V直流电源输入

# 第三章 使用 DE10-Standard 开发板

本章将介绍 DE10-Standard 开发板的使用并描述其外围设备。

## 3.1 设置 FPGA 配置模式

当 DE10-Standard 开发板接通电源后，可以从 EPCS 或 HPS 配置 FPGA。MSEL[4:0]引脚用于选择配置模式。如图 3-1 所示，在 DE10-Standard 开发板上可以通过 6 引脚 DIP 开关 SW10 选择配置模式。

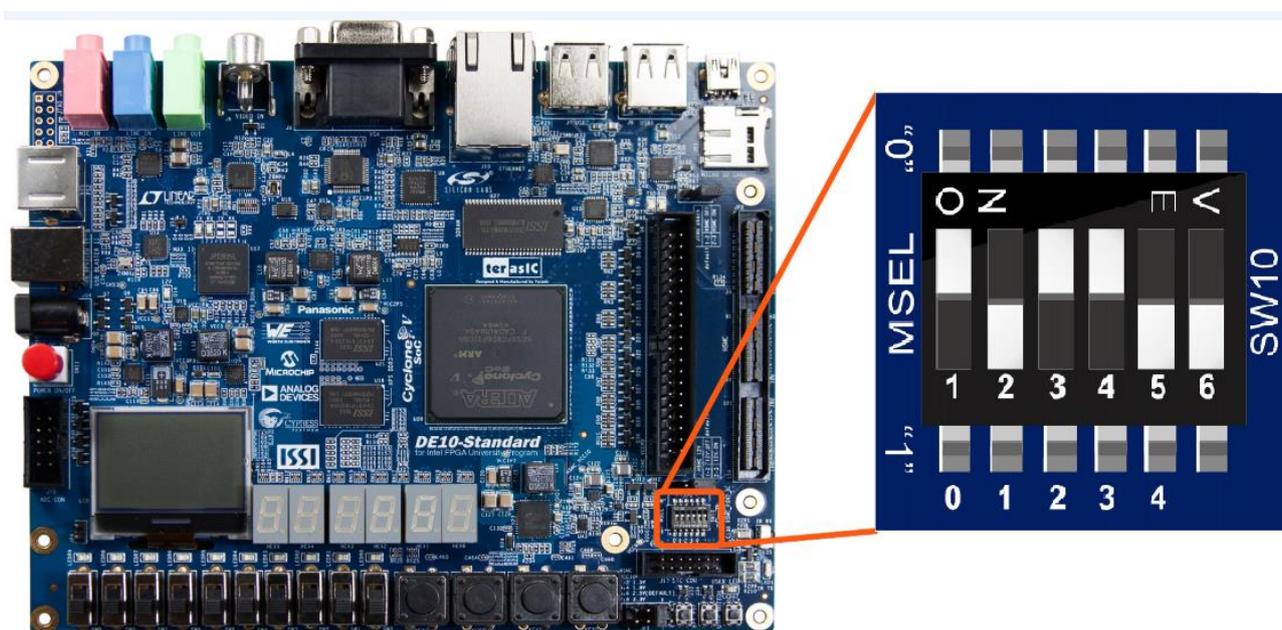


图 3-1 DIP SW10 设置 AS 配置模式

表 3-1 列出了 MSEL[4:0]和 DIP 开关 SW10 之间的对应关系。

表 3-1 FPGA 配置模式开关(SW10)

开发板参照组件	信号名	描述	默认 AS 模式
SW10.1	MSEL0	通过这些引脚选择配置模式	ON (“0”)
SW10.2	MSEL1		OFF (“1”)
SW10.3	MSEL2		ON (“0”)
SW10.4	MSEL3		ON (“0”)
SW10.5	MSEL4		OFF (“1”)
SW10.6	N/A	N/A	N/A

图 3-1 所示的 MSEL[4:0]设置的配置模式是 AS 模式，这也是 DE10-Standard 开发板的默

认配置模式。开发板接通电源后，从预先烧录了默认代码的 EPCS 器件配置 FPGA。如果开发人员想通过运行在 Linux 上的应用软件重新配置 FPGA，就需要在配置开始前将 MSEL[4:0] 设置成 01010。

表 3-2 设置 MSEL 引脚用于配置 DE10-Standard 的 FPGA

MSEL[4:0]	配置模式	描述
10010	AS	从 EPCS 配置 FPGA (默认)
01010	FPPx32	从 HPS 配置 FPGA

### 3.2 配置 DE10-Standard 开发板的 Cyclone V SoC FPGA

DE10-Standard 支持以下两种配置方式：

1. JTAG 配置：它以 IEEE 标准 Joint Test Action Group 来命名。配置比特流直接加载到 Cyclone V SoC FPGA 芯片。只要开发板保持上电状态，FPGA 就会维持当前状态不变；开发板断电后，配置信息就会丢失。

2. AS 配置：这种配置方式称为串行主动编程。配置比特流会加载到四路串行配置器件 EPCS128 中，该器件为非易失性存储器件。即使 DE10-Standard 开发板断电，配置信息也会保存在 EPCS128 中。开发板上电后，EPCS128 芯片中的配置数据会自动加载到 Cyclone V SoC FPGA 芯片。

#### ■ DE10-Standard 开发板的 JTAG 链

通过 DE10-Standard 开发板的 JTAG 接口可以配置 FPGA 芯片，但是 JTAG 链必须形成环路，这样 Quartus Prime 软件才能检测到 FPGA 器件。如图 3-2 所示为 DE10-Standard 开发板的 JTAG 链。

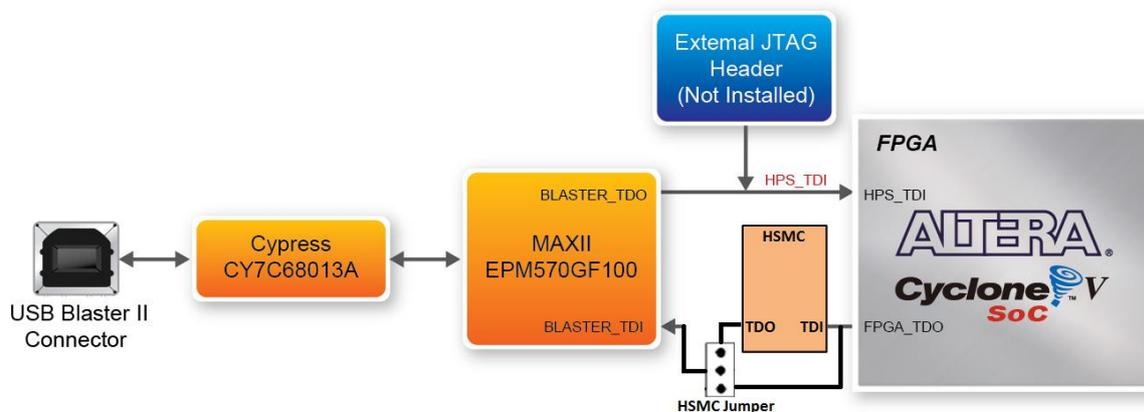


图 3-2 DE10-Standard 开发板的 JTAG 链示意图

## ■ JTAG 模式配置 FPGA

JTAG 链上有 FPGA 和 HPS 两个器件，以下步骤将演示如何在 JTAG 模式下配置 FPGA。

1. 打开 Quartus Prime Programmer 工具，并点击 Auto Detect，如图 3-3 的绿色方框所示。

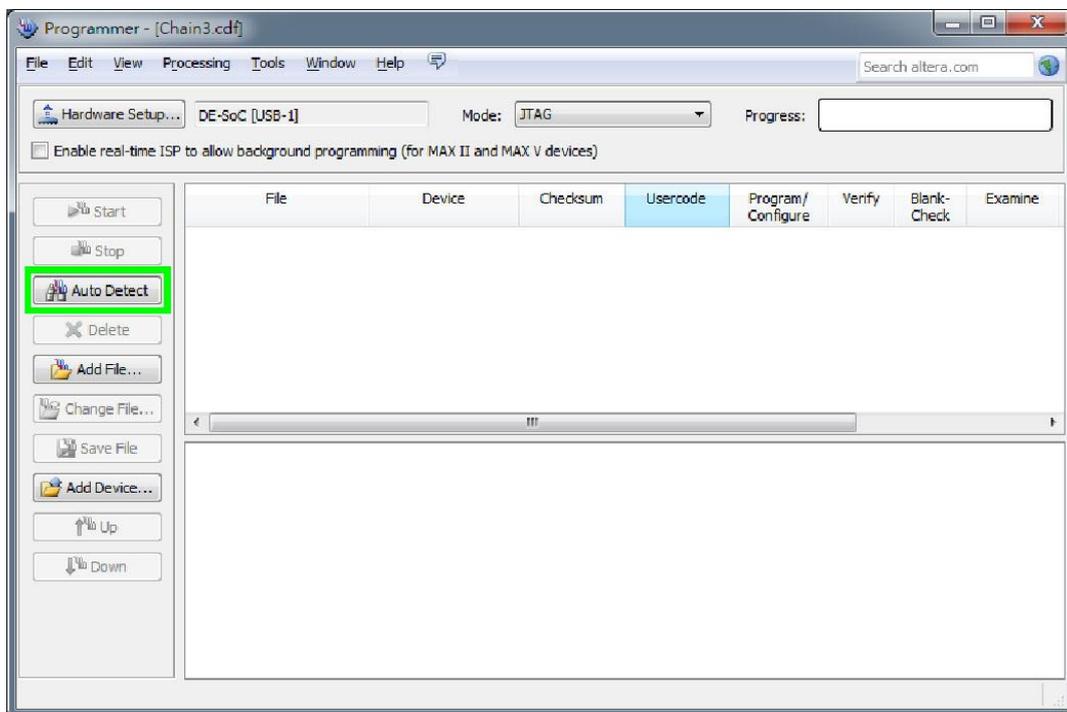


图 3-3 JTAG 模式检测 FPGA 器件

2. 如图 3-4 的绿色方框所示，选择检测到的与开发板相对应的 FPGA 器件。

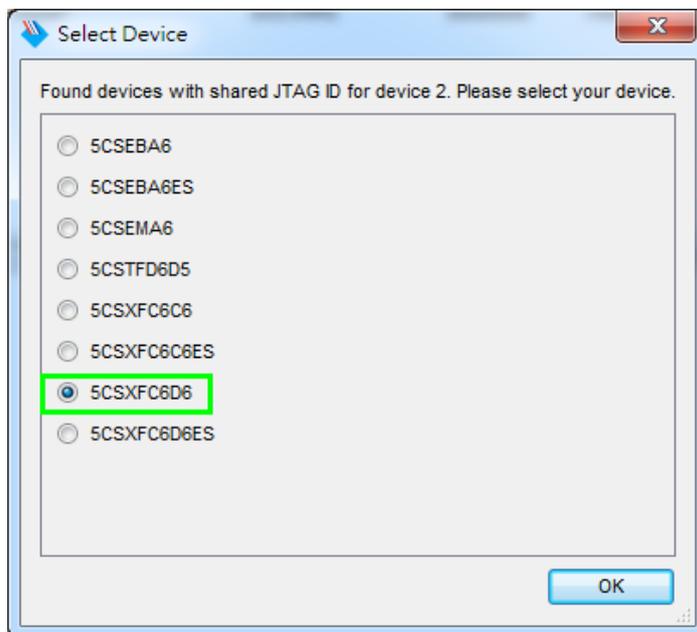


图 3-4 选择 5CSXFC6D6

3. 如图 3-5 所示，FPGA 和 HPS 都被检测到。

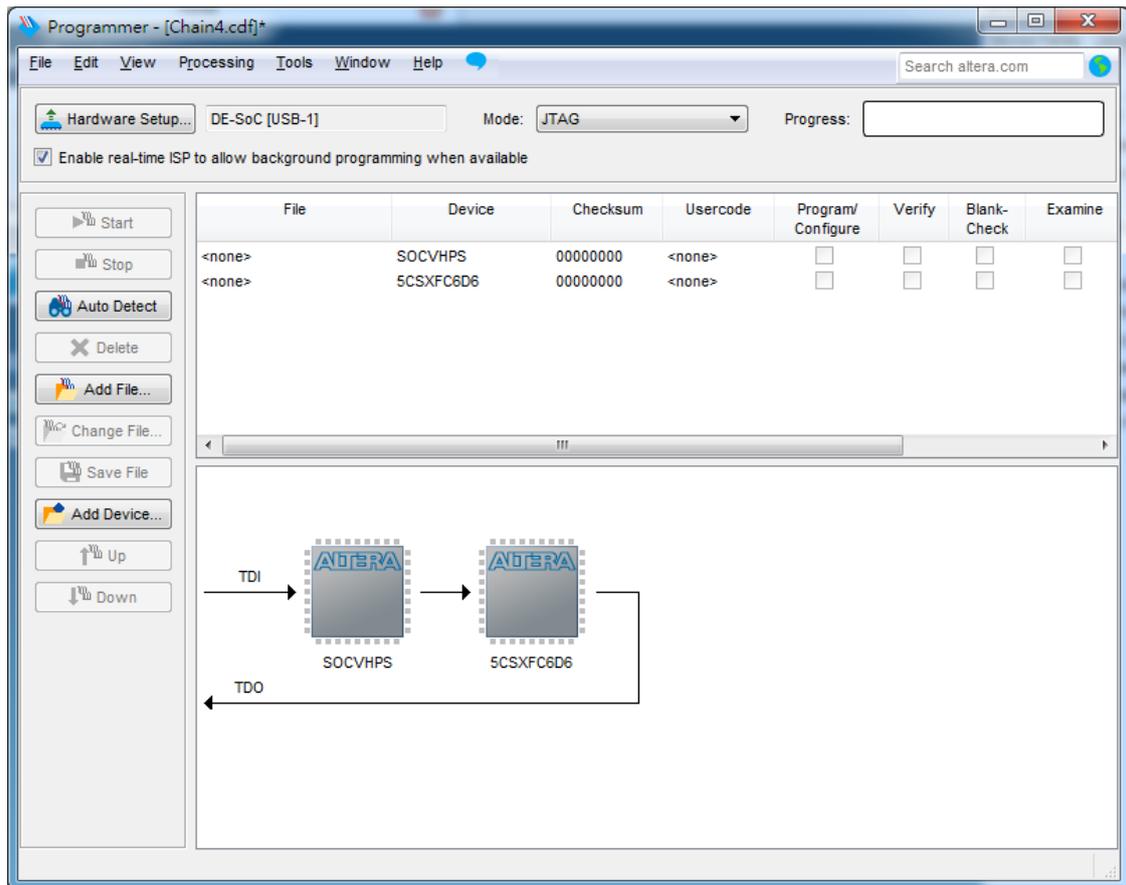


图 3-5 检测到 FPGA 和 HPS

4. 如图 3-6 所示，右键单击 FPGA 器件，选择 Change File 打开 .sof 配置文件所在的文件夹。

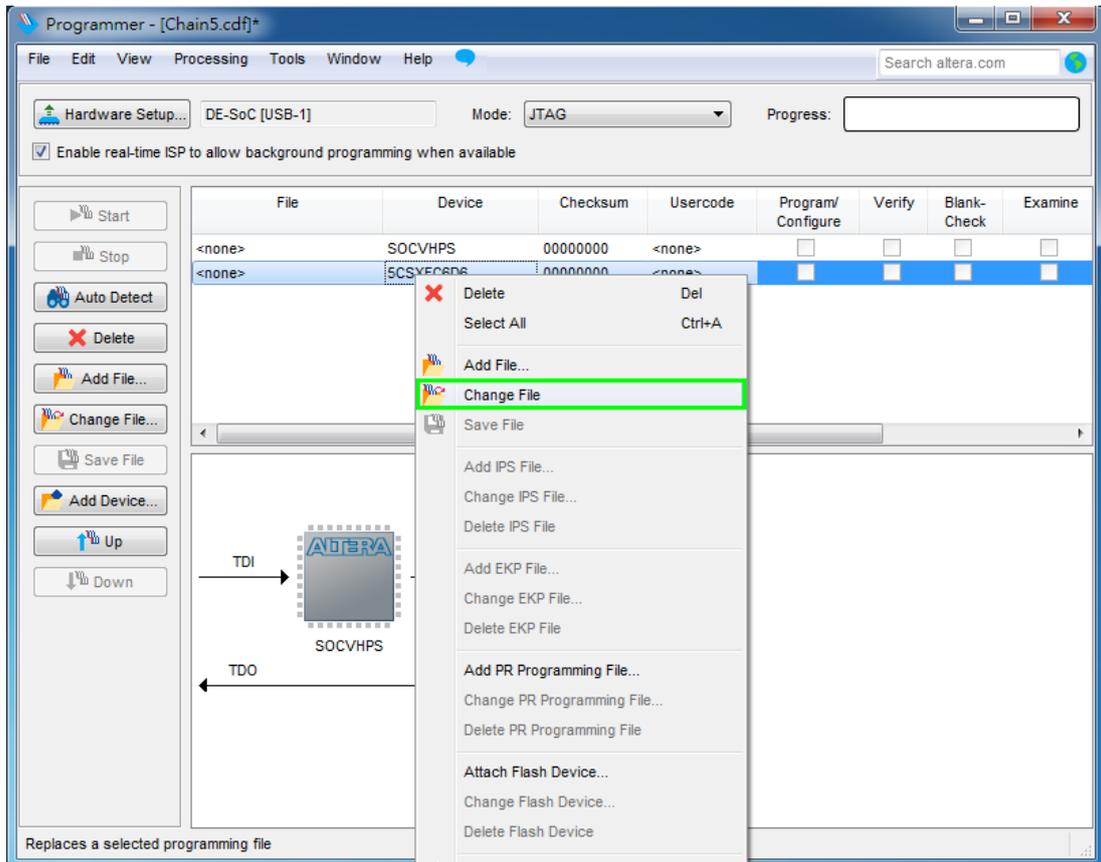


图 3-6 打开.sof 文件所在文件夹

5. 选择.sof 配置文件，如图 3-7 所示。

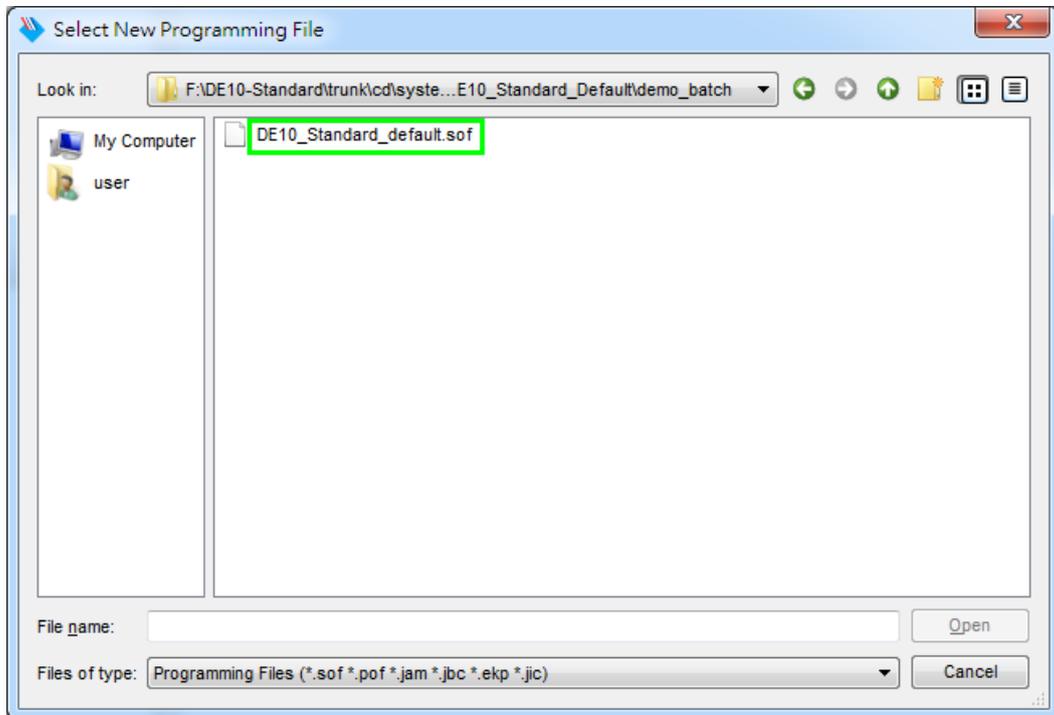


图 3-7 选择.sof 文件

6. 勾选 Program/Configure 复选框，点击 Start 按钮，将.sof 文件下载到 FPGA 芯片，如图 3-8 所示。

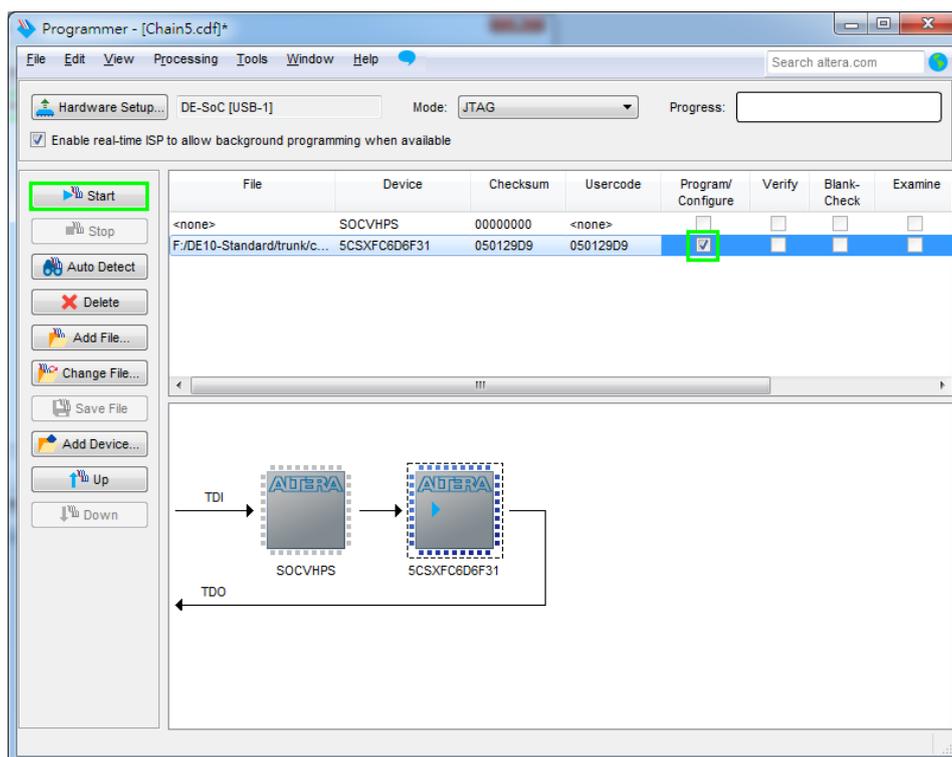


图 3-8 烧录.sof 文件

## ■ 使用 AS 模式配置 FPGA

- DE10-Standard开发板使用EPCS128芯片存储Cyclone V SoC FPGA的配置数据。开发板接通电源后，配置数据会从EPCS128自动加载到FPGA。
- 用户需要使用Serial Flash Loader (SFL)通过JTAG接口配置EPCS128。基于FPGA的SFL是FPGA内的软IP核，用于桥接JTAG和Flash接口。Quartus Prime软件具有SFL宏功能。如图3-9所示为采用了SFL解决方案的配置方法。
- 请参考第八章 固化EPCS器件，了解配置EPCS器件的基本方法。

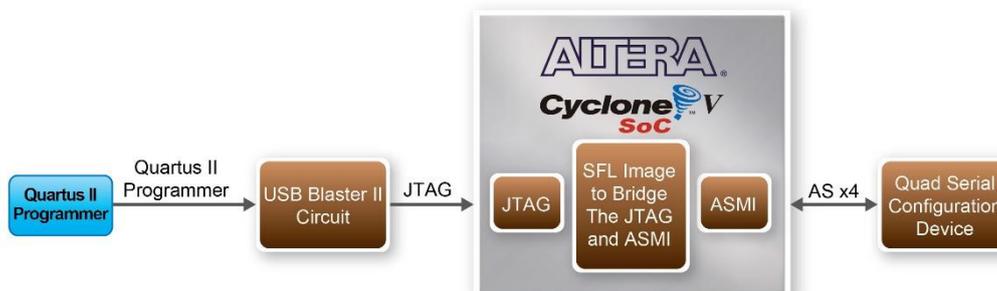


图 3-9 采用 SFL 解决方案配置 EPCS128

### 3.3 开发板状态指示组件

除了 FPGA 器件控制的 10 个 LED 之外，还有 5 个 LED 可以指示开发板状态，如图 3-10 所示，详细信息如表 3-3 所示。

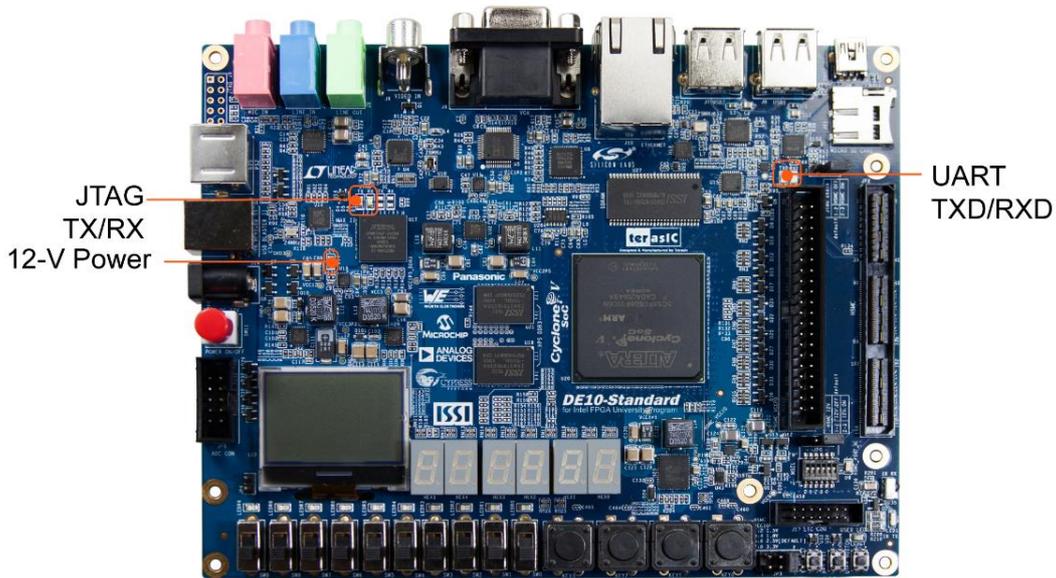


图 3-10 开发板状态 LED 指示灯

表 3-3 开发板状态 LED 指示灯信息

开发板参照组件	LED 名	描述
D14	12V Power	接通 12V 直流电源时点亮
TXD	UART TXD	数据从 FT232R 传到 USB Host 时点亮
RXD	UART RXD	数据从 USB Host 传到 FT232R 时点亮
D5	JTAG_RX	预留
D4	JTAG_TX	

### 3.4 开发板复位组件

DE10-Standard 开发板上有热、冷两个 HPS 复位按钮，如图 3-11 所示。表 3-4 描述了这两个复位按钮的用途。图 3-12 所示为 DE10-Standard 开发板的复位树状图。

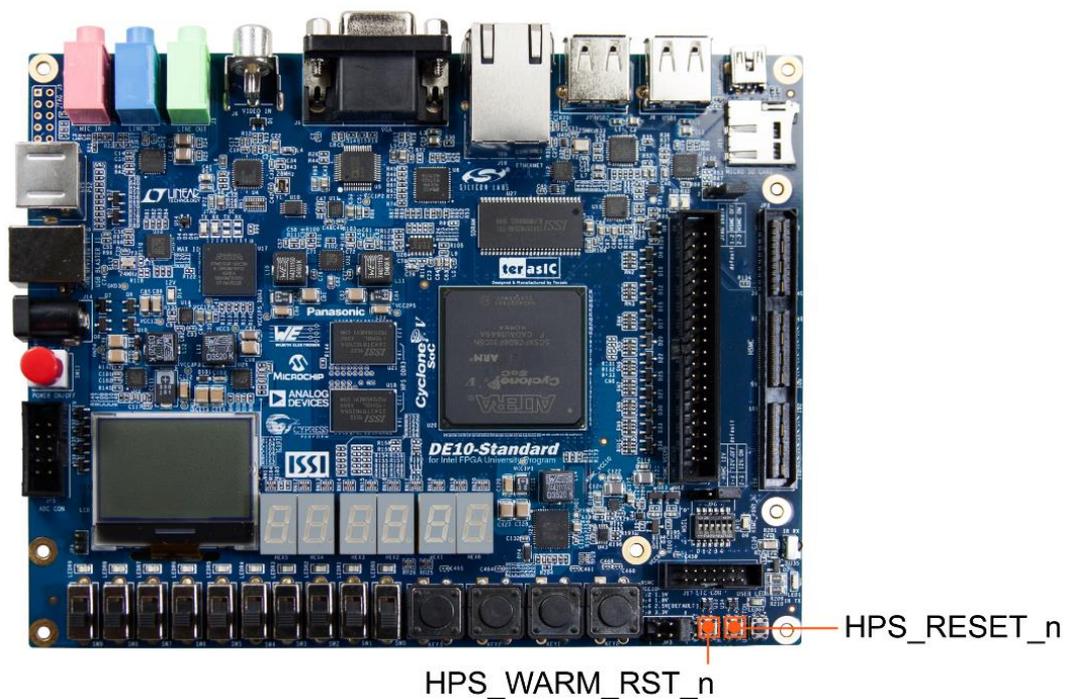


图 3-11 HPS 的热、冷复位按钮

表 3-4 DE10-Standard 的两个 HPS 复位按钮

开发板参照组件	信号名	描述
KEY5	HPS_RESET_N	对 HPS、以太网、USB 主设备进行冷复位。低电平输入有效，复位所有可以被复位的 HPS 逻辑。
KEY7	HPS_WARM_RST_N	对 HPS 模块进行热复位。低电平输入有效，作用于系统复位域来实现调试功能。

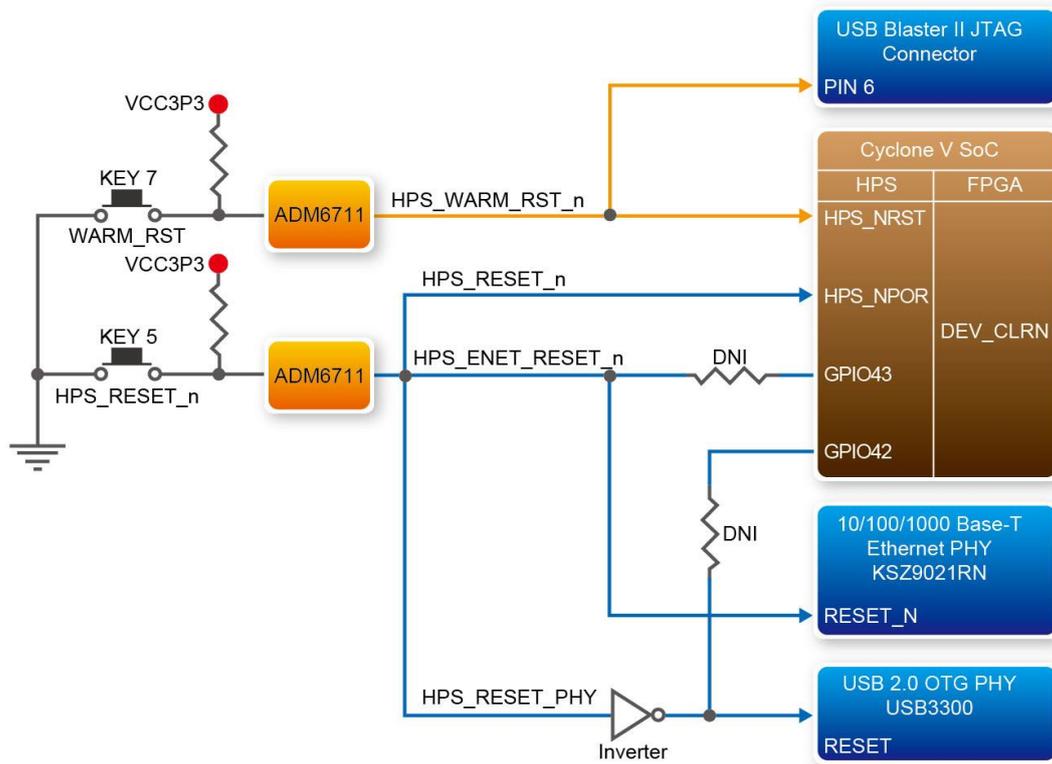


图 3-12 HPS 复位树状图

### 3.5 时钟电路

图 3-13 所示为连接到 Cyclone V SoC FPGA 的所有外部时钟的默认频率。时钟发生器可以产生低抖动时钟信号。连接到 FPGA 的四个 50MHz 时钟信号可以用作用户逻辑电路的时钟源。其中一个 25MHz 时钟信号连接到两个 HPS 时钟输入端，另外一个 25MHz 时钟信号连接到千兆以太网收发器的时钟输入端。两个 24MHz 时钟信号连接到 USB 主设备/OTG PHY 和 USB Hub 控制器的时钟输入端。FPGA I/O 时钟输入的相关引脚分配如表 3-5 所示。

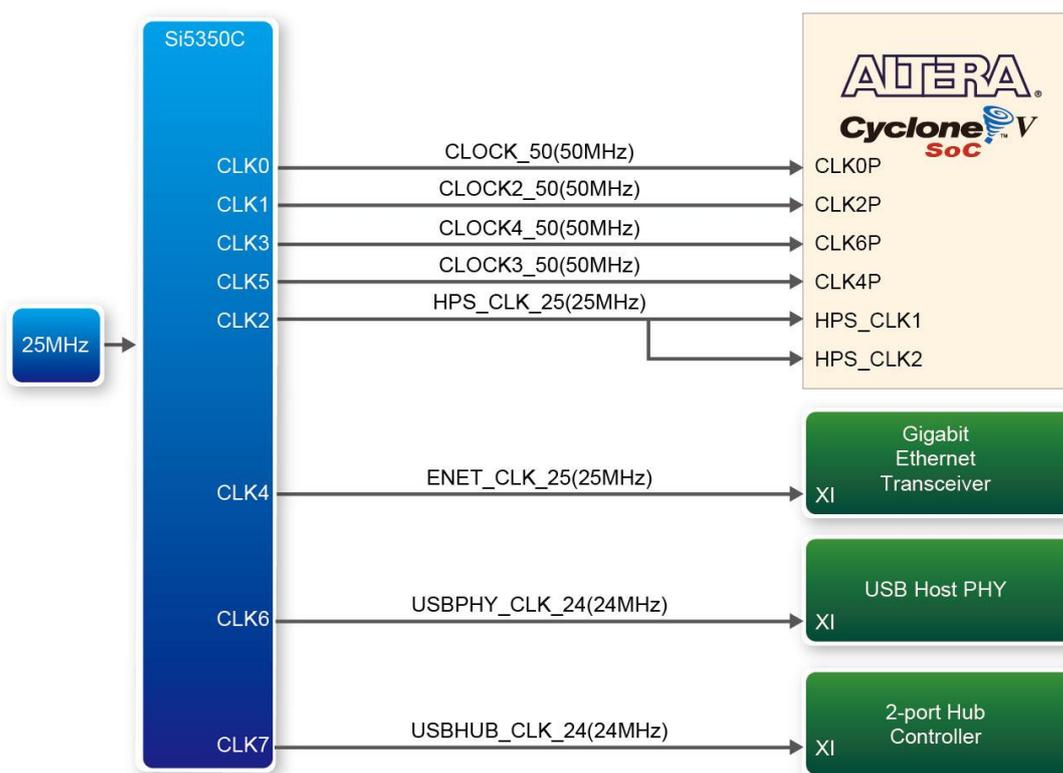


图 3-13 DE10-Standard 时钟分配方框图

表 3-5 时钟输入引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
CLOCK_50	PIN_AF14	50 MHz 时钟输入	3.3V
CLOCK2_50	PIN_AA16	50 MHz 时钟输入	3.3V
CLOCK3_50	PIN_Y26	50 MHz 时钟输入	3.3V
CLOCK4_50	PIN_K14	50 MHz 时钟输入	3.3V
HPS_CLOCK1_25	PIN_D25	25 MHz 时钟输入	3.3V
HPS_CLOCK2_25	PIN_F25	25 MHz 时钟输入	3.3V

## 3.6 FPGA 外围设备

本节将介绍连接 FPGA 的外设接口。使用 FPGA 用户自定义逻辑电路可以控制或监测不同的接口。

### 3.6.1 按钮开关、拨动开关与 LED

DE10-Standard 开发板上有 4 个按钮开关与 FPGA 相连，如图 3-14 所示为按钮开关和 Cyclone V SoC FPGA 连接示意图。如图 3-15 所示，每个按钮开关都实现了施密特触发并可以

用做去抖动开关。由施密特触发器引出的 KEY0、KEY1、KEY2 和 KEY3 四个按钮开关都直接连到了 Cyclone V SoC FPGA。按下按钮开关会输出低电平（低电平有效）。因为这些按钮开关已经去抖动，所以在电路中它们就可以用作复位输入。

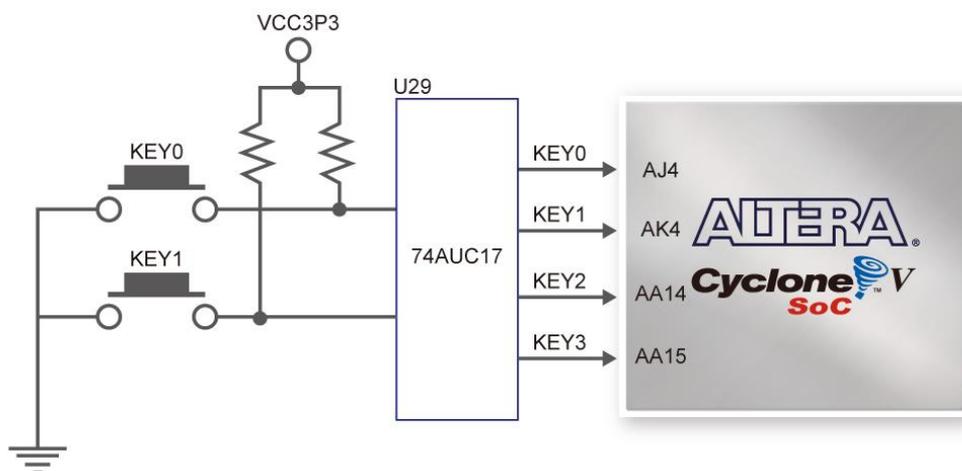


图 3-14 按钮开关连接 Cyclone V SoC FPGA 示意图

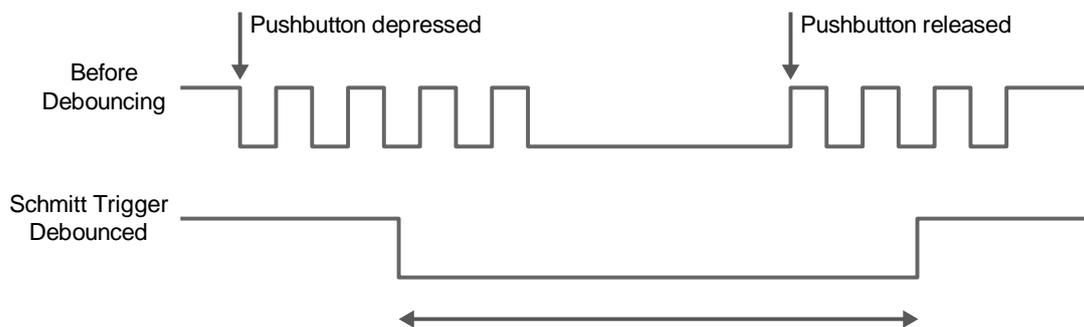


图 3-15 开关去抖动

如图 3-16 所示，还有 10 个拨动开关与 FPGA 相连。这些开关没有去抖动，它们可以用做电路的敏感电平数据输入。每个开关都独立的直接连到 FPGA。当拨动开关在 DOWN 位置（靠近开发板边缘）时产生低电平输入到 FPGA；在 UP 位置时产生高电平输入到 FPGA。

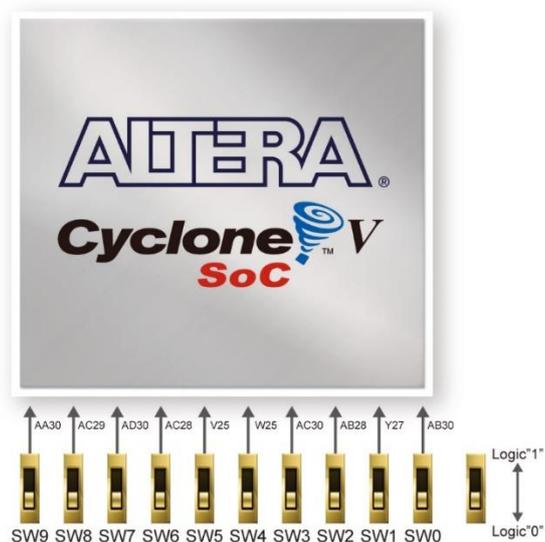


图 3-16 拨动开关连接 Cyclone V SoC FPGA 示意图

开发板上还有 10 个用户可控制的 LED 与 FPGA 相连。每一个 LED 都由 Cyclone V SoC FPGA 直接独立驱动，驱动对应的引脚为高电平或低电平来点亮或熄灭 LED。图 3-17 所示为 LED 和 Cyclone V SoC FPGA 的连接示意图。表 3-6、表 3-7 和表 3-8 列出了拨动开关、按钮开关和 LED 的引脚分配。



图 3-17 LED 和 Cyclone V SoC FPGA 连接示意图

表 3-6 拨动开关引脚分配

组件名	FPGA 引脚号	描述	I/O 标准
SW[0]	PIN_AB30	Slide Switch[0]	取决于 JP3
SW[1]	PIN_Y27	Slide Switch[1]	取决于 JP3
SW[2]	PIN_AB28	Slide Switch[2]	取决于 JP3
SW[3]	PIN_AC30	Slide Switch[3]	取决于 JP3
SW[4]	PIN_W25	Slide Switch[4]	取决于 JP3

SW[5]	PIN_V25	Slide Switch[5]	取决于 JP3
SW[6]	PIN_AC28	Slide Switch[6]	取决于 JP3
SW[7]	PIN_AD30	Slide Switch[7]	取决于 JP3
SW[8]	PIN_AC29	Slide Switch[8]	取决于 JP3
SW[9]	PIN_AA30	Slide Switch[9]	取决于 JP3

表 3-7 按钮开关引脚分配

组件名	FPGA 引脚号	描述	I/O 标准
KEY[0]	PIN_AJ4	Push-button[0]	3.3V
KEY[1]	PIN_AK4	Push-button[1]	3.3V
KEY[2]	PIN_AA14	Push-button[2]	3.3V
KEY[3]	PIN_AA15	Push-button[3]	3.3V

表 3-8 LED 引脚分配

组件名	FPGA 引脚号	描述	I/O 标准
LEDR[0]	PIN_AA24	LED [0]	3.3V
LEDR[1]	PIN_AB23	LED [1]	3.3V
LEDR[2]	PIN_AC23	LED [2]	3.3V
LEDR[3]	PIN_AD24	LED [3]	3.3V
LEDR[4]	PIN_AG25	LED [4]	3.3V
LEDR[5]	PIN_AF25	LED [5]	3.3V
LEDR[6]	PIN_AE24	LED [6]	3.3V
LEDR[7]	PIN_AF24	LED [7]	3.3V
LEDR[8]	PIN_AB22	LED [8]	3.3V
LEDR[9]	PIN_AC22	LED [9]	3.3V

### 3.6.2 七段数码管

DE10-Standard 开发板上有 6 个七段数码管，这些数码管两两一对显示不同大小的数字和字符。图 3-18 显示了七段数码管的每个引脚（共阳模式）均连接到 Cyclone V SoC FPGA。FPGA 输出低电压到数码管时，数码管对应的 LED 会点亮，反之则熄灭。

数码管的 LED 都从 0 到 6 依次编号，图 3-18 所示为编号次序。表 3-9 列出了所有数码管和 FPGA 芯片的引脚连接信息。

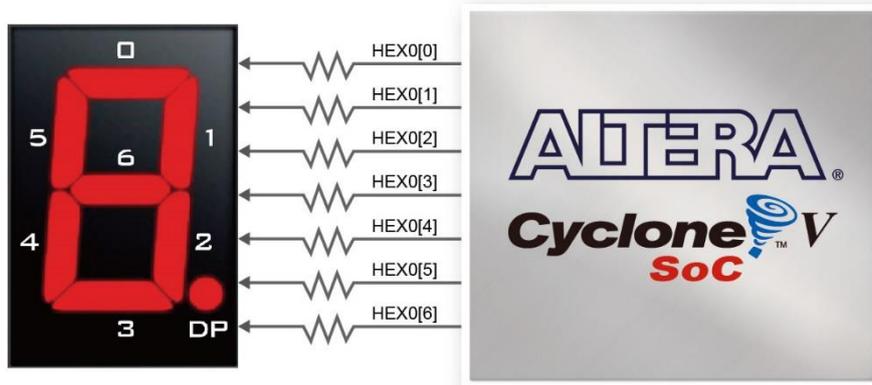


图 3-18 Cyclone V SoC FPGA 和七段数码管连接示意图

表 3-9 七段数码管引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HEX0[0]	PIN_W17	Seven Segment Digit 0[0]	3.3V
HEX0[1]	PIN_V18	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_AG17	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_AG16	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_AH17	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_AG18	Seven Segment Digit 0[5]	3.3V
HEX0[6]	PIN_AH18	Seven Segment Digit 0[6]	3.3V
HEX1[0]	PIN_AF16	Seven Segment Digit 1[0]	3.3V
HEX1[1]	PIN_V16	Seven Segment Digit 1[1]	3.3V
HEX1[2]	PIN_AE16	Seven Segment Digit 1[2]	3.3V
HEX1[3]	PIN_AD17	Seven Segment Digit 1[3]	3.3V
HEX1[4]	PIN_AE18	Seven Segment Digit 1[4]	3.3V
HEX1[5]	PIN_AE17	Seven Segment Digit 1[5]	3.3V
HEX1[6]	PIN_V17	Seven Segment Digit 1[6]	3.3V
HEX2[0]	PIN_AA21	Seven Segment Digit 2[0]	3.3V
HEX2[1]	PIN_AB17	Seven Segment Digit 2[1]	3.3V
HEX2[2]	PIN_AA18	Seven Segment Digit 2[2]	3.3V
HEX2[3]	PIN_Y17	Seven Segment Digit 2[3]	3.3V
HEX2[4]	PIN_Y18	Seven Segment Digit 2[4]	3.3V
HEX2[5]	PIN_AF18	Seven Segment Digit 2[5]	3.3V
HEX2[6]	PIN_W16	Seven Segment Digit 2[6]	3.3V
HEX3[0]	PIN_Y19	Seven Segment Digit 3[0]	3.3V
HEX3[1]	PIN_W19	Seven Segment Digit 3[1]	3.3V
HEX3[2]	PIN_AD19	Seven Segment Digit 3[2]	3.3V
HEX3[3]	PIN_AA20	Seven Segment Digit 3[3]	3.3V
HEX3[4]	PIN_AC20	Seven Segment Digit 3[4]	3.3V
HEX3[5]	PIN_AA19	Seven Segment Digit 3[5]	3.3V
HEX3[6]	PIN_AD20	Seven Segment Digit 3[6]	3.3V
HEX4[0]	PIN_AD21	Seven Segment Digit 4[0]	3.3V
HEX4[1]	PIN_AG22	Seven Segment Digit 4[1]	3.3V

HEX4[2]	PIN_AE22	Seven Segment Digit 4[2]	3.3V
HEX4[3]	PIN_AE23	Seven Segment Digit 4[3]	3.3V
HEX4[4]	PIN_AG23	Seven Segment Digit 4[4]	3.3V
HEX4[5]	PIN_AF23	Seven Segment Digit 4[5]	3.3V
HEX4[6]	PIN_AH22	Seven Segment Digit 4[6]	3.3V
HEX5[0]	PIN_AF21	Seven Segment Digit 5[0]	3.3V
HEX5[1]	PIN_AG21	Seven Segment Digit 5[1]	3.3V
HEX5[2]	PIN_AF20	Seven Segment Digit 5[2]	3.3V
HEX5[3]	PIN_AG20	Seven Segment Digit 5[3]	3.3V
HEX5[4]	PIN_AE19	Seven Segment Digit 5[4]	3.3V
HEX5[5]	PIN_AF19	Seven Segment Digit 5[5]	3.3V
HEX5[6]	PIN_AB21	Seven Segment Digit 5[6]	3.3V

### 3.6.3 2x20 GPIO 扩展接口

DE10-Standard开发板有一个40引脚扩展接口。接口上有36个引脚与Cyclone V SoC FPGA 直接连接，其余4个引脚是DC +5V (VCC5)、DC +3.3V (VCC3P3)和两个GND引脚。表3-10列出了连接GPIO接口的子卡允许的最大功耗。

表 3-10 GPIO 扩展接口的电压和最大电流限制

供电电压	最大电流
5V	1A
3.3V	1.5A

扩展接口的每个引脚都连接两个二极管和一个电阻，用来防范高、低电平。图 3-19 所示为用于 36 个数据引脚的保护电路。表 3-11 列出了 GPIO 扩展接口的引脚分配。

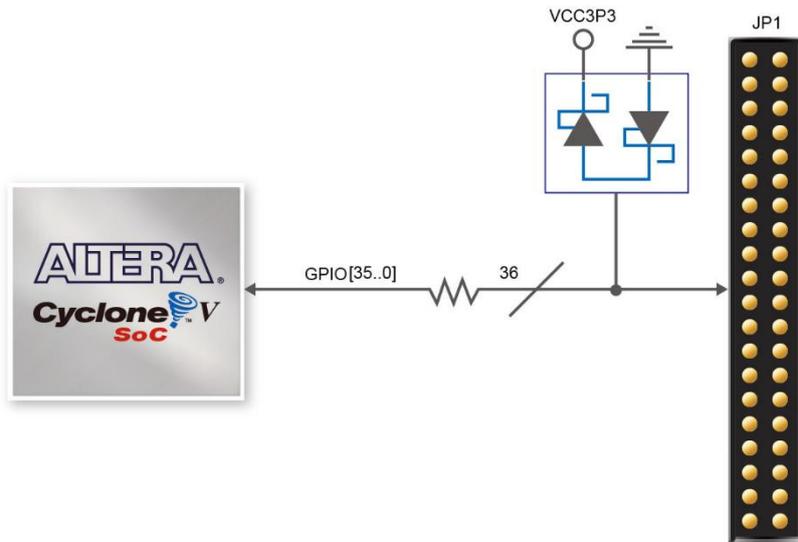


图 3-19 GPIO 扩展接口和 Cyclone V SoC FPGA 连接示意图

表 3-11 扩展接口引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
GPIO[0]	PIN_W15	GPIO Connection 0[0]	3.3V
GPIO[1]	PIN_AK2	GPIO Connection 0[1]	3.3V
GPIO[2]	PIN_Y16	GPIO Connection 0[2]	3.3V
GPIO[3]	PIN_AK3	GPIO Connection 0[3]	3.3V
GPIO[4]	PIN_AJ1	GPIO Connection 0[4]	3.3V
GPIO[5]	PIN_AJ2	GPIO Connection 0[5]	3.3V
GPIO[6]	PIN_AH2	GPIO Connection 0[6]	3.3V
GPIO[7]	PIN_AH3	GPIO Connection 0[7]	3.3V
GPIO[8]	PIN_AH4	GPIO Connection 0[8]	3.3V
GPIO[9]	PIN_AH5	GPIO Connection 0[9]	3.3V
GPIO[10]	PIN_AG1	GPIO Connection 0[10]	3.3V
GPIO[11]	PIN_AG2	GPIO Connection 0[11]	3.3V
GPIO[12]	PIN_AG3	GPIO Connection 0[12]	3.3V
GPIO[13]	PIN_AG5	GPIO Connection 0[13]	3.3V
GPIO[14]	PIN_AG6	GPIO Connection 0[14]	3.3V
GPIO[15]	PIN_AG7	GPIO Connection 0[15]	3.3V
GPIO[16]	PIN_AG8	GPIO Connection 0[16]	3.3V
GPIO[17]	PIN_AF4	GPIO Connection 0[17]	3.3V
GPIO[18]	PIN_AF5	GPIO Connection 0[18]	3.3V
GPIO[19]	PIN_AF6	GPIO Connection 0[19]	3.3V
GPIO[20]	PIN_AF8	GPIO Connection 0[20]	3.3V
GPIO[21]	PIN_AF9	GPIO Connection 0[21]	3.3V
GPIO[22]	PIN_AF10	GPIO Connection 0[22]	3.3V
GPIO[23]	PIN_AE7	GPIO Connection 0[23]	3.3V
GPIO[24]	PIN_AE9	GPIO Connection 0[24]	3.3V
GPIO[25]	PIN_AE11	GPIO Connection 0[25]	3.3V
GPIO[26]	PIN_AE12	GPIO Connection 0[26]	3.3V
GPIO[27]	PIN_AD7	GPIO Connection 0[27]	3.3V
GPIO[28]	PIN_AD9	GPIO Connection 0[28]	3.3V
GPIO[29]	PIN_AD10	GPIO Connection 0[29]	3.3V
GPIO[30]	PIN_AD11	GPIO Connection 0[30]	3.3V
GPIO[31]	PIN_AD12	GPIO Connection 0[31]	3.3V
GPIO[32]	PIN_AC9	GPIO Connection 0[32]	3.3V
GPIO[33]	PIN_AC12	GPIO Connection 0[33]	3.3V
GPIO[34]	PIN_AB12	GPIO Connection 0[34]	3.3V
GPIO[35]	PIN_AA12	GPIO Connection 0[35]	3.3V

### 3.6.4 HSMC 接口

DE10-Standard 开发板上有一个 HSMC 接口, 通过该接口连接 HSMC 子卡可以扩展 FPGA 主板的外围设备, 从而满足当前各种高频信号需求以及支持低速设备接口。HSMC 接口支持 JTAG、时钟输入输出、高速串行 I/O(收发器)以及单端或差分信号。HSMC 接口信号如图 3-20 所示。表 3-12 列出了连接 HSMC 接口的子卡的最大功耗。

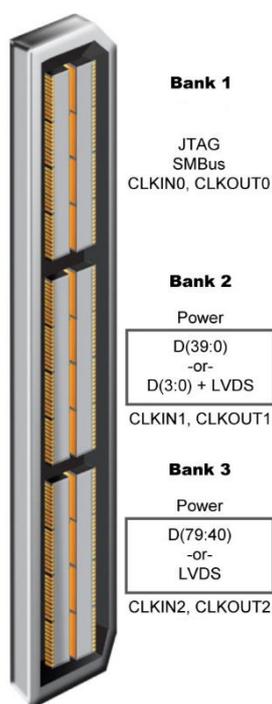


图 3-20 HSMC 信号 Bank 图

表 3-12 HSMC 接口的电压和最大电流限制

供电电压	最大电流
12V	1A
3.3V	1.5A

用户可以通过 JP3 设定 HSMC 接口上的信号 I/O 电压标准为 3.3V、2.5V、1.8V 或 1.5V（默认设置为 2.5V）。因为 HSMC I/O 连接到 FPGA 的 Bank 5B&8A，而这两个 Bank 的 VCCIO 电压由 JP3 控制，所以用户可以使用跳帽短接 JP3 对应的引脚，选择 VCCIO 5B&VCCIO 8A 的输入电压为 3.3V、2.5V、1.8V 和 1.5V 来控制 I/O 引脚的电压标准。表 3-13 列出了 JP3 的设置，表 3-14 列出了 HSMC 接口的所有引脚分配。

表 3-13 不同 I/O 标准的 JP3 设置

JP3 跳接器设置	VCCIO5B 和 VCCIO8A 的供电电压	HSMC 接口(JP2) I/O 电压
短接引脚 1 和 2	1.5V	1.5V
短接引脚 3 和 4	1.8V	1.8V
短接引脚 5 和 6	2.5V	2.5V (默认)
短接引脚 7 和 8	3.3V	3.3V

表 3-14 HSMC 接口引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HSMC_CLKIN0	PIN_J14	专用时钟输入	取决于 JP3
HSMC_CLKIN_N1	PIN_AB27	LVDS RX 或 CMOS I/O 或差分时钟输入	取决于 JP3
HSMC_CLKIN_N2	PIN_G15	LVDS RX 或 CMOS I/O 或差分时钟输入	取决于 JP3
HSMC_CLKIN_P1	PIN_AA26	LVDS RX 或 CMOS I/O 或差分时钟输入	取决于 JP3
HSMC_CLKIN_P2	PIN_H15	LVDS RX 或 CMOS I/O 或差分时钟输入	取决于 JP3
HSMC_CLKOUT0	PIN_AD29	专用时钟输出	取决于 JP3
HSMC_CLKOUT_N1	PIN_E6	LVDS TX 或 CMOS I/O 或差分时钟输入输出	取决于 JP3
HSMC_CLKOUT_N2	PIN_A10	LVDS TX 或 CMOS I/O 或差分时钟输入输出	取决于 JP3
HSMC_CLKOUT_P1	PIN_E7	LVDS TX 或 CMOS I/O 或差分时钟输入输出	取决于 JP3
HSMC_CLKOUT_P2	PIN_A11	LVDS TX 或 CMOS I/O 或差分时钟输入输出	取决于 JP3
HSMC_D[0]	PIN_C10	LVDS TX 或 CMOS I/O	取决于 JP3
HSMC_D[1]	PIN_H13	LVDS RX 或 CMOS I/O	取决于 JP3
HSMC_D[2]	PIN_C9	LVDS TX 或 CMOS I/O	取决于 JP3
HSMC_D[3]	PIN_H12	LVDS RX 或 CMOS I/O	取决于 JP3
HSMC_SCL	PIN_AA28	管理串行数据	取决于 JP3
HSMC_SDA	PIN_AE29	管理串行时钟	取决于 JP3
HSMC_RX_D_N[0]	PIN_G11	LVDS RX bit 0n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[1]	PIN_J12	LVDS RX bit 1n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[2]	PIN_F10	LVDS RX bit 2n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[3]	PIN_J9	LVDS RX bit 3n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[4]	PIN_K8	LVDS RX bit 4n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[5]	PIN_H7	LVDS RX bit 5n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[6]	PIN_G8	LVDS RX bit 6n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[7]	PIN_F8	LVDS RX bit 7n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[8]	PIN_E11	LVDS RX bit 8n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[9]	PIN_B5	LVDS RX bit 9n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[10]	PIN_D9	LVDS RX bit 10n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[11]	PIN_D12	LVDS RX bit 11n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[12]	PIN_D10	LVDS RX bit 12n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[13]	PIN_B12	LVDS RX bit 13n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[14]	PIN_E13	LVDS RX bit 14n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[15]	PIN_G13	LVDS RX bit 15n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_N[16]	PIN_F14	LVDS RX bit 16n 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[0]	PIN_G12	LVDS RX bit 0 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[1]	PIN_K12	LVDS RX bit 1 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[2]	PIN_G10	LVDS RX bit 2 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[3]	PIN_J10	LVDS RX bit 3 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[4]	PIN_K7	LVDS RX bit 4 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[5]	PIN_J7	LVDS RX bit 5 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[6]	PIN_H8	LVDS RX bit 6 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[7]	PIN_F9	LVDS RX bit 7 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[8]	PIN_F11	LVDS RX bit 8 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[9]	PIN_B6	LVDS RX bit 9 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[10]	PIN_E9	LVDS RX bit 10 或 CMOS I/O	取决于 JP3

HSMC_RX_D_P[11]	PIN_E12	LVDS RX bit 11 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[12]	PIN_D11	LVDS RX bit 12 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[13]	PIN_C13	LVDS RX bit 13 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[14]	PIN_F13	LVDS RX bit 14 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[15]	PIN_H14	LVDS RX bit 15 或 CMOS I/O	取决于 JP3
HSMC_RX_D_P[16]	PIN_F15	LVDS RX bit 16 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[0]	PIN_A8	LVDS TX bit 0n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[1]	PIN_D7	LVDS TX bit 1n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[2]	PIN_F6	LVDS TX bit 2n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[3]	PIN_C5	LVDS TX bit 3n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[4]	PIN_C4	LVDS TX bit 4n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[5]	PIN_E2	LVDS TX bit 5n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[6]	PIN_D4	LVDS TX bit 6n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[7]	PIN_B3	LVDS TX bit 7n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[8]	PIN_D1	LVDS TX bit 8n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[9]	PIN_C2	LVDS TX bit 9n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[10]	PIN_B1	LVDS TX bit 10n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[11]	PIN_A3	LVDS TX bit 11n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[12]	PIN_A5	LVDS TX bit 12n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[13]	PIN_B7	LVDS TX bit 13n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[14]	PIN_B8	LVDS TX bit 14n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[15]	PIN_B11	LVDS TX bit 15n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_N[16]	PIN_A13	LVDS TX bit 16n 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[0]	PIN_A9	LVDS TX bit 0 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[1]	PIN_E8	LVDS TX bit 1 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[2]	PIN_G7	LVDS TX bit 2 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[3]	PIN_D6	LVDS TX bit 3 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[4]	PIN_D5	LVDS TX bit 4 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[5]	PIN_E3	LVDS TX bit 5 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[6]	PIN_E4	LVDS TX bit 6 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[7]	PIN_C3	LVDS TX bit 7 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[8]	PIN_E1	LVDS TX bit 8 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[9]	PIN_D2	LVDS TX bit 9 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[10]	PIN_B2	LVDS TX bit 10 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[11]	PIN_A4	LVDS TX bit 11 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[12]	PIN_A6	LVDS TX bit 12 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[13]	PIN_C7	LVDS TX bit 13 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[14]	PIN_C8	LVDS TX bit 14 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[15]	PIN_C12	LVDS TX bit 15 或 CMOS I/O	取决于 JP3
HSMC_TX_D_P[16]	PIN_B13	LVDS TX bit 16 或 CMOS I/O	取决于 JP3

### 3.6.5 24 位音频编解码器

DE10-Standard 开发板通过 Wolfson WM8731 音频编解码器可以提供 24 位高品质音频。该芯片支持麦克风输入、line-in 及 line-out 端口，采样率在 8KHz 到 96KHz 之间可调。WM8731 芯片通过串行 I2C 总线进行控制，该总线通过 I2C 多路复用器连接到 HPS 或 Cyclone V SoC

FPGA 音频电路与 FPGA 的连接如图 3-21 所示，对应的引脚分配如表 3-15 所列。有关 WM8731 编解码器的更多信息可以参考芯片数据表，在制造商官网或 DE10-Standard 系统 CD 的 \datasheets\Audio CODEC 目录中可以找到该数据表。

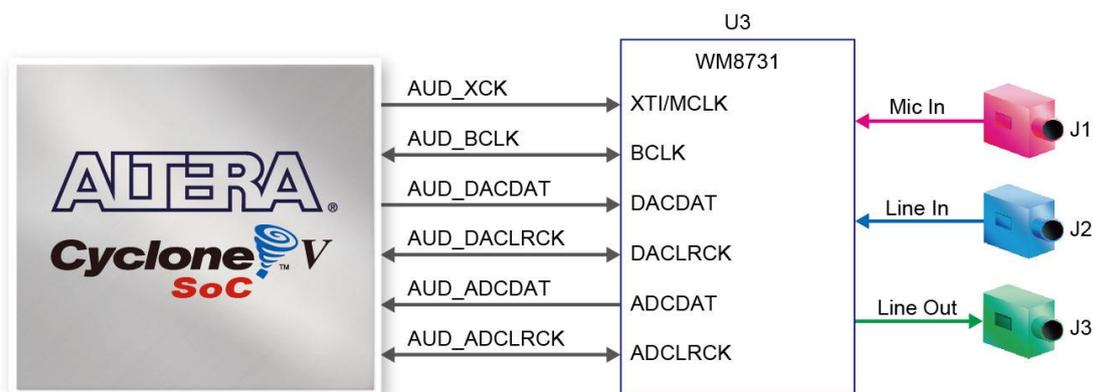


图 3-21 FPGA 和音频编解码芯片连接示意图

表 3-15 音频 CODEC 芯片引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
AUD_ADCLK	PIN_AH29	Audio CODEC ADC LR 时钟	3.3V
AUD_ADCDAT	PIN_AJ29	Audio CODEC ADC 数据	3.3V
AUD_DACLK	PIN_AG30	Audio CODEC DAC LR 时钟	3.3V
AUD_DACDAT	PIN_AF29	Audio CODEC DAC 数据	3.3V
AUD_XCK	PIN_AH30	Audio CODEC 芯片时钟	3.3V
AUD_BCLK	PIN_AF30	Audio CODEC 比特流时钟	3.3V
I2C_SCLK	PIN_Y24 or PIN_E23	I2C 时钟	3.3V
I2C_SDAT	PIN_Y23 or PIN_C24	I2C 数据	3.3V

### 3.6.6 I2C 多路复用器

DE10-Standard 开发板实现了一个 I2C 多路复用器，以便 HPS 能访问由 FPGA 控制的 I2C 总线。图 3-22 所示为 I2C 多路复用器与 FPGA、HPS 的连接示意图。当且仅当 HPS\_I2C\_CONTROL 信号设置为高电平时，HPS 才能访问音频编解码器和 TV 解码器。I2C 总线引脚分配如表 3-16 所列。

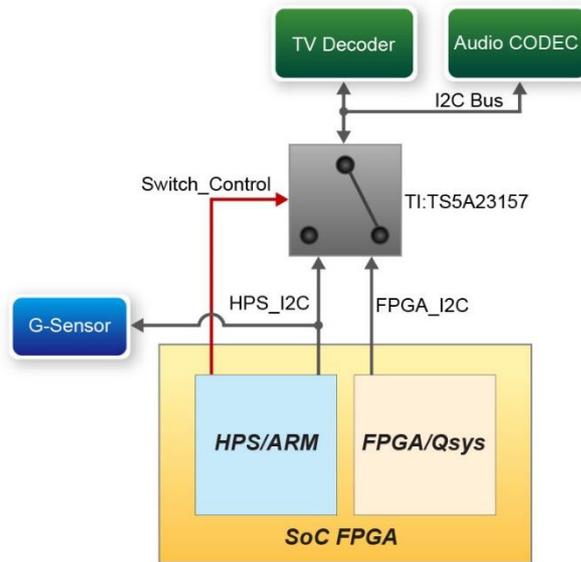


图 3-22 I2C 多路复用器控制机制

表 3-16 I2C 总线引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
FPGA_I2C_SCLK	PIN_Y24	FPGA I2C 时钟	3.3V
FPGA_I2C_SDAT	PIN_Y23	FPGA I2C 数据	3.3V
HPS_I2C1_SCLK	PIN_E23	第一个 HPS I2C 控制器的 I2C 时钟	3.3V
HPS_I2C1_SDAT	PIN_C24	第一个 HPS I2C 控制器的 I2C 数据	3.3V
HPS_I2C2_SCLK	PIN_H23	第二个 HPS I2C 控制器的 I2C 时钟	3.3V
HPS_I2C2_SDAT	PIN_A25	第二个 HPS I2C 控制器的 I2C 数据	3.3V

### 3.6.7 VGA 输出

DE10-Standard 开发板上有一个用于 VGA 输出的 15 引脚 D-USB 接口。VGA 同步信号直接从 Cyclone V SoC FPGA 产生，ADV7123 芯片是三通道、10 位（仅使用高 8 位）高速视频数模转换器(DAC)，它将数字信号转换为三种基色（红、绿、蓝）模拟信号。该芯片最高能支持 1280\*1024 分辨率、信号以 100MHz 传输的 SXGA 标准。图 3-23 所示为 FPGA 和 VGA 的连接示意图。

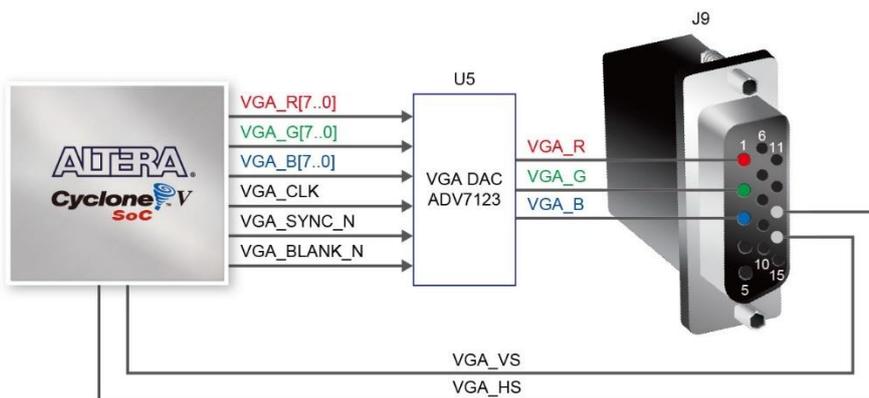


图 3-23 FPGA 与 VGA 连接示意图

用户在网络上可以轻易查找到 VGA 同步和 RGB 数据的时序规范。图 3-24 所示为 VGA 显示器上显示的每一行（水平）的基本时序要求。特定持续时间的低电平有效脉冲信号作为显示器的水平同步信号(hsync)输入，表示上一数据行扫描结束和下一数据行扫描开始。水平同步脉冲信号产生之后，输出到显示器的 RGB 数据在后沿时间(b)内必须是无效的（驱动为 0V），后沿时间之后是显示间隔。在数据显示间隔时间内，RGB 数据驱动每个像素在行上依次显示。最后是前沿时间(d)，这段时间内，必须在下一个水平同步信号产生之前使 RGB 信号再次无效。场同步信号(vsync)时序与图 3-24 所示的行同步信号时序类似，只不过场同步脉冲信号是表示某一帧扫描结束和下一帧扫描开始，RGB 数据是帧中所有行的数据集合。表 3-17 和表 3-18 列出了行和场时序中的不同分辨率和 a、b、c、d 的持续时间。

有关 ADV7123 视频数模转换器的更多信息可以参考芯片数据表，在制造商官网或 DE10-Standard System CD 的\Datasheet\Video DAC 目录中可以找到该数据表。Cyclone V SoC FPGA 和 ADV7123 之间的引脚分配如表 3-19 所列。

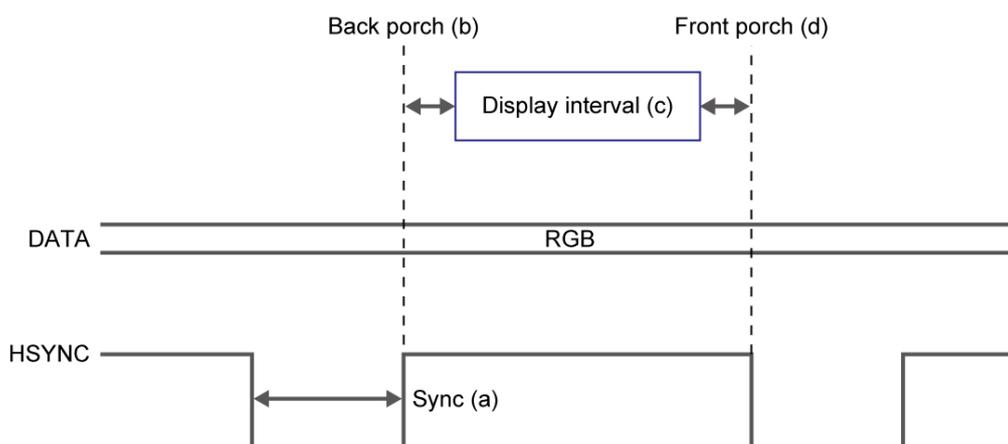


图 3-24 VGA 行扫描时序

表 3-17 VGA 行时序规范

VGA 模式		水平时序规范				
配置	分辨率(HxV)	a(us)	b(us)	c(us)	d(us)	像素时钟 (MHz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36
SVGA(60Hz)	800x600	3.2	2.2	20	1	40
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95
1280x1024(60Hz)	1280x1024	1.0	2.3	11.9	0.4	108

表 3-18 VGA 场时序规范

VGA 模式		垂直时序规范				
配置	分辨率(HxV)	a(lines)	b(lines)	c(lines)	d(lines)	像素时钟 (MHz)
VGA(60Hz)	640x480	2	33	480	10	25
VGA(85Hz)	640x480	3	25	480	1	36
SVGA(60Hz)	800x600	4	23	600	1	40
SVGA(75Hz)	800x600	3	21	600	1	49
SVGA(85Hz)	800x600	3	27	600	1	56
XGA(60Hz)	1024x768	6	29	768	3	65
XGA(70Hz)	1024x768	6	29	768	3	75
XGA(85Hz)	1024x768	3	36	768	1	95
1280x1024(60Hz)	1280x1024	3	38	1024	1	108

表 3-19 VGA 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
VGA_R[0]	PIN_AK29	VGA Red[0]	3.3V
VGA_R[1]	PIN_AK28	VGA Red[1]	3.3V
VGA_R[2]	PIN_AK27	VGA Red[2]	3.3V
VGA_R[3]	PIN_AJ27	VGA Red[3]	3.3V
VGA_R[4]	PIN_AH27	VGA Red[4]	3.3V
VGA_R[5]	PIN_AF26	VGA Red[5]	3.3V
VGA_R[6]	PIN_AG26	VGA Red[6]	3.3V
VGA_R[7]	PIN_AJ26	VGA Red[7]	3.3V
VGA_G[0]	PIN_AK26	VGA Green[0]	3.3V
VGA_G[1]	PIN_AJ25	VGA Green[1]	3.3V
VGA_G[2]	PIN_AH25	VGA Green[2]	3.3V
VGA_G[3]	PIN_AK24	VGA Green[3]	3.3V
VGA_G[4]	PIN_AJ24	VGA Green[4]	3.3V
VGA_G[5]	PIN_AH24	VGA Green[5]	3.3V

VGA_G[6]	PIN_AK23	VGA Green[6]	3.3V
VGA_G[7]	PIN_AH23	VGA Green[7]	3.3V
VGA_B[0]	PIN_AJ21	VGA Blue[0]	3.3V
VGA_B[1]	PIN_AJ20	VGA Blue[1]	3.3V
VGA_B[2]	PIN_AH20	VGA Blue[2]	3.3V
VGA_B[3]	PIN_AJ19	VGA Blue[3]	3.3V
VGA_B[4]	PIN_AH19	VGA Blue[4]	3.3V
VGA_B[5]	PIN_AJ17	VGA Blue[5]	3.3V
VGA_B[6]	PIN_AJ16	VGA Blue[6]	3.3V
VGA_B[7]	PIN_AK16	VGA Blue[7]	3.3V
VGA_CLK	PIN_AK21	VGA Clock	3.3V
VGA_BLANK_N	PIN_AK22	VGA BLANK	3.3V
VGA_HS	PIN_AK19	VGA H_SYNC	3.3V
VGA_VS	PIN_AK18	VGA V_SYNC	3.3V
VGA_SYNC_N	PIN_AJ22	VGA SYNC	3.3V

### 3.6.8 TV 解码器

DE10-Standard 开发板搭载了 ADV7180 TV 解码器芯片。ADV7180 是一款集成视频解码器，可以自动检测并将标准模拟基带信号(NTSC、PAL 和 SECAM)转换为兼容 8 位 ITU-R BT.656 接口标准的 4:2:2 分量视频数据。ADV7180 兼容各种视频设备，包括 DVD 播放机、磁带机、广播级视频源以及安全/监控类摄像机。

TV 解码器中的寄存器可以通过 Cyclone V SoC FPGA 或 HPS 的串行 I2C 总线进行访问和设置。注意读/写 TV 解码器的 I2C 总线地址为 0x40/0x41。TV 解码器的引脚分配如表 3-20 所列。有关 ADV7180 芯片的更多信息可以参考芯片数据表，在制造商官网或 DE10-Standard 系统 CD 的\Datasheet\Video Decoder 目录中可以找到该数据表。

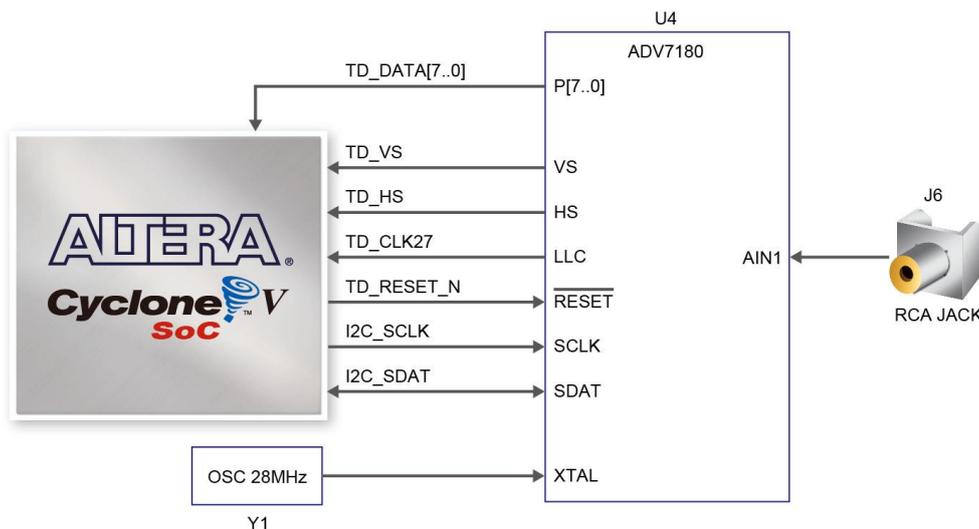


图 3-25 FPGA 和 TV 解码器连接示意图

表 3-20 TV 解码器引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
TD_DATA [0]	PIN_AG27	TV Decoder Data[0]	3.3V
TD_DATA [1]	PIN_AF28	TV Decoder Data[1]	3.3V
TD_DATA [2]	PIN_AE28	TV Decoder Data[2]	3.3V
TD_DATA [3]	PIN_AE27	TV Decoder Data[3]	3.3V
TD_DATA [4]	PIN_AE26	TV Decoder Data[4]	3.3V
TD_DATA [5]	PIN_AD27	TV Decoder Data[5]	3.3V
TD_DATA [6]	PIN_AD26	TV Decoder Data[6]	3.3V
TD_DATA [7]	PIN_AD25	TV Decoder Data[7]	3.3V
TD_HS	PIN_AH28	TV Decoder H_SYNC	3.3V
TD_VS	PIN_AG28	TV Decoder V_SYNC	3.3V
TD_CLK27	PIN_AC18	TV Decoder Clock Input	3.3V
TD_RESET_N	PIN_AC27	TV Decoder Reset	3.3V
I2C_SCLK	PIN_Y24 or PIN_E23	I2C Clock	3.3V
I2C_SDAT	PIN_Y23 or PIN_C24	I2C Data	3.3V

### 3.6.9 红外接收器

DE10-Standard 开发板搭载了红外遥控接收器模块（型号：IRM-V538/TR1），在 DE10-Standard 系统 CD 的 \Datasheet\ IR Receiver and Emitter 目录中可以找到该模块的数据表。遥控器是选购配件，可以在 Terasic 官网上购买，它内置编码芯片（uPD6121G）用来产生红外信号。图 3-26 所示为红外接收器与 FPGA 的连接示意图。表 3-21 列出了红外接收器与 FPGA 连接的引脚分配。

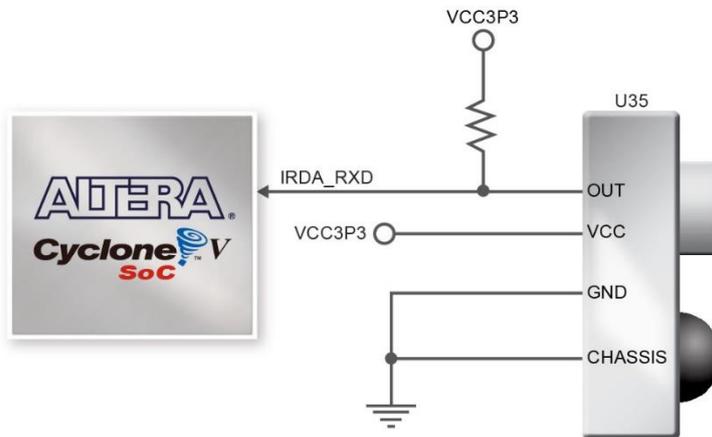


图 3-26 FPGA 和 IR 接收器连接示意图

表 3-21 IR 接收器引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
IRDA_RXD	PIN_W20	IR Receiver	3.3V

### 3.6.10 红外发射器 LED

DE10-Standard 开发板上有一个红外发射器 LED，可用于 IR 通信，它广泛应用于短距离无线控制电视设备。通过匹配另一端的 IR 接收器，它还能和其他系统进行通信。图 3-27 所示为 IR 发射 LED 与 FPGA 的连接示意图。表 3-22 列出了 IR 发射 LED 与 FPGA 连接的引脚分配。

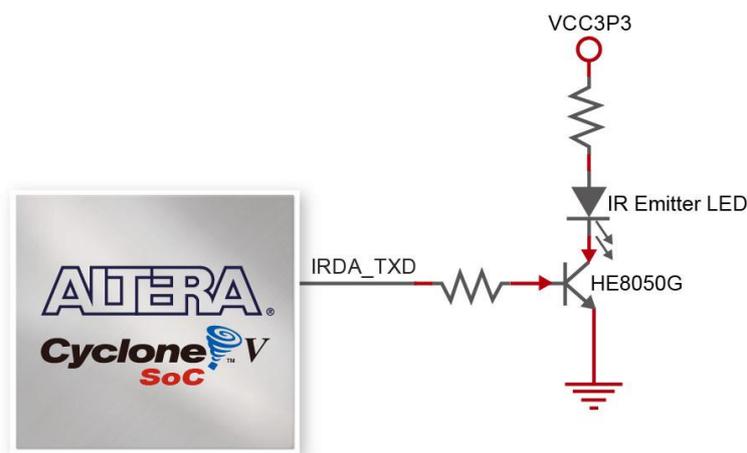


图 3-27 FPGA 与 IR 发射 LED 连接示意图

表 3-22 IR 发射 LED 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
IRDA_TXD	PIN_W21	IR Emitter	3.3V

### 3.6.11 SDRAM 存储器

DE10-Standard 开发板上有一个 64MB(32Mx16) SDRAM 芯片。该芯片具有与 FPGA 相连的 16 位数据线、控制线和地址线。该芯片采用 3.3V LVCMOS 信号标准。FPGA 与 SDRAM 的连接如图 3-28 所示，引脚分配如表 3-23 所示。



图 3-28 FPGA 与 SDRAM 连接示意图

表 3-23 SDRAM 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
DRAM_ADDR[0]	PIN_AK14	SDRAM Address[0]	3.3V
DRAM_ADDR[1]	PIN_AH14	SDRAM Address[1]	3.3V
DRAM_ADDR[2]	PIN_AG15	SDRAM Address[2]	3.3V
DRAM_ADDR[3]	PIN_AE14	SDRAM Address[3]	3.3V
DRAM_ADDR[4]	PIN_AB15	SDRAM Address[4]	3.3V
DRAM_ADDR[5]	PIN_AC14	SDRAM Address[5]	3.3V
DRAM_ADDR[6]	PIN_AD14	SDRAM Address[6]	3.3V
DRAM_ADDR[7]	PIN_AF15	SDRAM Address[7]	3.3V
DRAM_ADDR[8]	PIN_AH15	SDRAM Address[8]	3.3V
DRAM_ADDR[9]	PIN_AG13	SDRAM Address[9]	3.3V
DRAM_ADDR[10]	PIN_AG12	SDRAM Address[10]	3.3V
DRAM_ADDR[11]	PIN_AH13	SDRAM Address[11]	3.3V
DRAM_ADDR[12]	PIN_AJ14	SDRAM Address[12]	3.3V
DRAM_DQ[0]	PIN_AK6	SDRAM Data[0]	3.3V
DRAM_DQ[1]	PIN_AJ7	SDRAM Data[1]	3.3V
DRAM_DQ[2]	PIN_AK7	SDRAM Data[2]	3.3V
DRAM_DQ[3]	PIN_AK8	SDRAM Data[3]	3.3V
DRAM_DQ[4]	PIN_AK9	SDRAM Data[4]	3.3V
DRAM_DQ[5]	PIN_AG10	SDRAM Data[5]	3.3V
DRAM_DQ[6]	PIN_AK11	SDRAM Data[6]	3.3V
DRAM_DQ[7]	PIN_AJ11	SDRAM Data[7]	3.3V
DRAM_DQ[8]	PIN_AH10	SDRAM Data[8]	3.3V
DRAM_DQ[9]	PIN_AJ10	SDRAM Data[9]	3.3V
DRAM_DQ[10]	PIN_AJ9	SDRAM Data[10]	3.3V
DRAM_DQ[11]	PIN_AH9	SDRAM Data[11]	3.3V
DRAM_DQ[12]	PIN_AH8	SDRAM Data[12]	3.3V
DRAM_DQ[13]	PIN_AH7	SDRAM Data[13]	3.3V
DRAM_DQ[14]	PIN_AJ6	SDRAM Data[14]	3.3V

DRAM_DQ[15]	PIN_AJ5	SDRAM Data[15]	3.3V
DRAM_BA[0]	PIN_AF13	SDRAM Bank Address[0]	3.3V
DRAM_BA[1]	PIN_AJ12	SDRAM Bank Address[1]	3.3V
DRAM_LDQM	PIN_AB13	SDRAM byte Data Mask[0]	3.3V
DRAM_UDQM	PIN_AK12	SDRAM byte Data Mask[1]	3.3V
DRAM_RAS_N	PIN_AE13	SDRAM Row Address Strobe	3.3V
DRAM_CAS_N	PIN_AF11	SDRAM Column Address Strobe	3.3V
DRAM_CKE	PIN_AK13	SDRAM Clock Enable	3.3V
DRAM_CLK	PIN_AH12	SDRAM Clock	3.3V
DRAM_WE_N	PIN_AA13	SDRAM Write Enable	3.3V
DRAM_CS_N	PIN_AG11	SDRAM Chip Select	3.3V

### 3.6.12 PS/2 串行接口

DE10-Standard 开发板上有一个标准的 PS/2 接口，可以连接 PS/2 键盘或鼠标。图 3-29 所示为 PS/2 电路与 FPGA 的连接示意图。如图 3-30 所示，用户可以通过 PS/2 Y 型线在 DE10-Standard 开发板上同时使用 PS/2 键盘和鼠标。在各种教育网站上可以找到有关如何使用 PS / 2 鼠标和/或键盘的说明。该接口对应的引脚分配如表 3-24 所示。



注意：如果用户只连接一个 PS/2 设备，那么连接到 FPGA I/O 的 PS/2 接口信号就是 PS2\_CLK 和 PS2\_DAT。

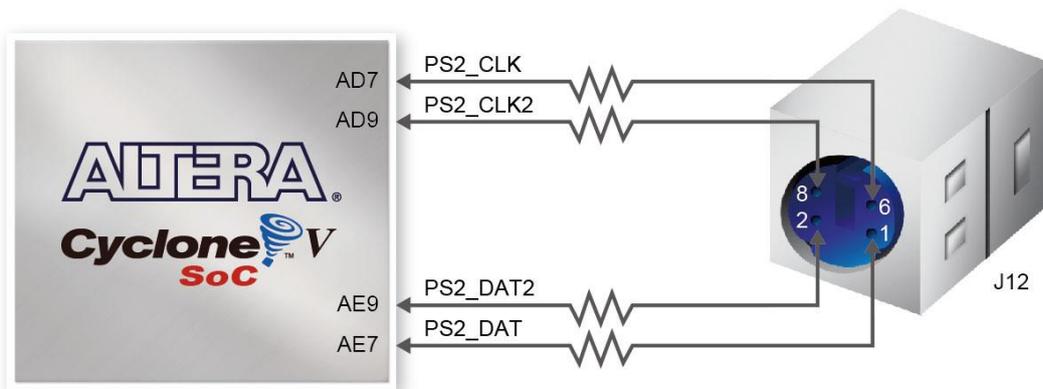


图 3-29 FPGA 与 PS/2 连接示意图



图 3-30 使用 Y-Cable 同时使用 PS/2 鼠标和键盘

表 3-24 PS/2 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
PS2_CLK	PIN_AB25	PS/2 时钟	3.3V
PS2_DAT	PIN_AA25	PS/2 数据	3.3V
PS2_CLK2	PIN_AC25	PS/2 时钟 (预留给第二个 PS/2 设备)	3.3V
PS2_DAT2	PIN_AB26	PS/2 数据 (预留给第二个 PS/2 设备)	3.3V

### 3.6.13 A/D 转换器与 2x5 引脚接头

DE10-Standard开发板搭载一款低噪声、8 通道、12 位CMOS ADC芯片。该ADC提供高达 500KSPS的转换吞吐率。所有输入通道的模拟输入范围为0V到4.096V。内部转换时钟允许外部串行输出时钟SCLK在最高为40MHz的任意频率下工作。它可以配置成在ADC\_IN0至ADC\_IN7输入端接收8个输入信号。如图3-31所示，这8个输入信号连接到一个2x5引脚接头。

有关该 A/D 转换器芯片的更多信息可以参考芯片数据表，在制造商官网或 DE10-Standard 系统 CD 的\Datasheet\ADC 目录中可以找到该数据表。

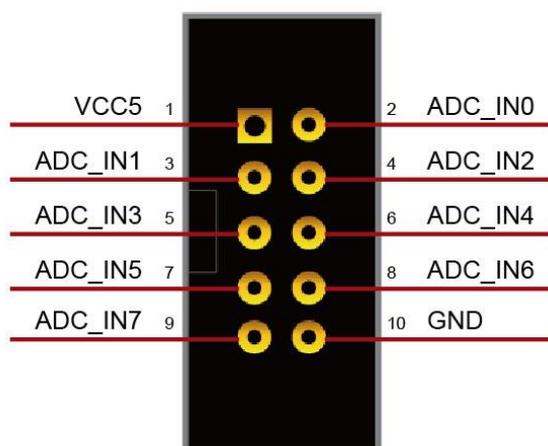


图 3-31 2x5 接头信号示意图

图 3-32 所示为 FPGA、2x5 引脚接头和 A/D 转换器的连接示意图。表 3-25 列出了 A/D 转换器的引脚分配。

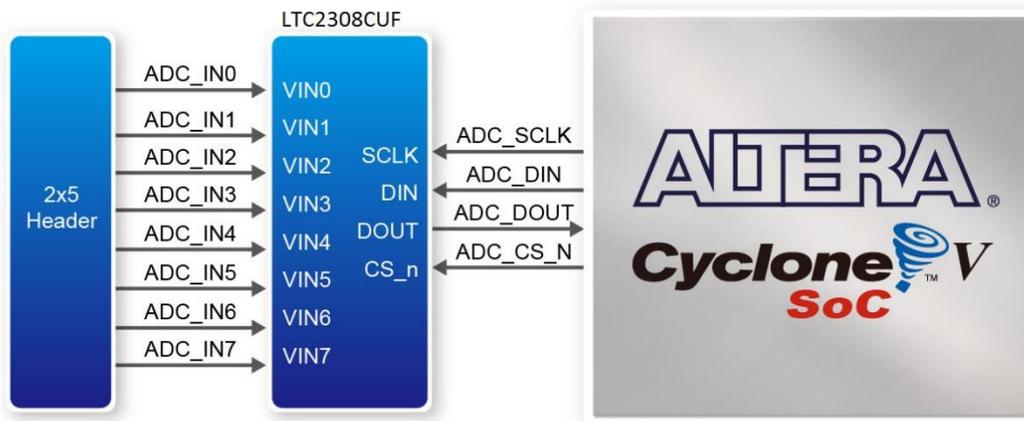


图 3-32 FPGA、2x5 引脚接头、A/D 转换器连接示意图

表 3-25 ADC 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
ADC_CONVST	PIN_Y21	开始转换	3.3V
ADC_DOUT	PIN_V23	数字数据输入	3.3V
ADC_DIN	PIN_W22	数字数据输出	3.3V
ADC_SCLK	PIN_W24	数字时钟输入	3.3V

### 3.7 HPS 外围设备

本节将介绍连接 Cyclone V SoC FPGA 的 HPS 部分的外设接口。通过 HPS 处理器可以访问这些接口。

#### 3.7.1 按钮开关和 LED

与 FPGA 端类似，HPS 端也有和它独立连接的按钮开关、LED 和其他接口。用户可以控制这些接口来监测 HPS 状态。

表 3-26 列出了所有的 LED 和按钮开关的引脚分配。

表 3-26 LED 和按钮开关的引脚分配

信号名	HPS GPIO	寄存器/bit	功能
HPS_KEY	GPIO54	GPIO1[25]	I/O
HPS_LED	GPIO53	GPIO1[24]	I/O

### 3.7.2 千兆以太网

DE10-Standard 开发板支持通过外部 Micrel KSZ9021RN PHY 芯片和 HPS 以太网 MAC 功能进行千兆以太网传输。集成 10/100/1000 Mbps 千兆以太网收发器的 KSZ9021RN 芯片也支持 RGMII MAC 接口。图 3-33 所示为 HPS、千兆以太网 PHY 和 RJ45 连接头的连接示意图。

千兆以太网接口对应的引脚分配如表 3-27 所列。有关该 KSZ9021RN PHY 芯片的更多信息可以参考芯片数据表，在制造商官网或 DE10-Standard 系统 CD 的 \Datasheet\Ethernet 目录中可以找到该数据表。

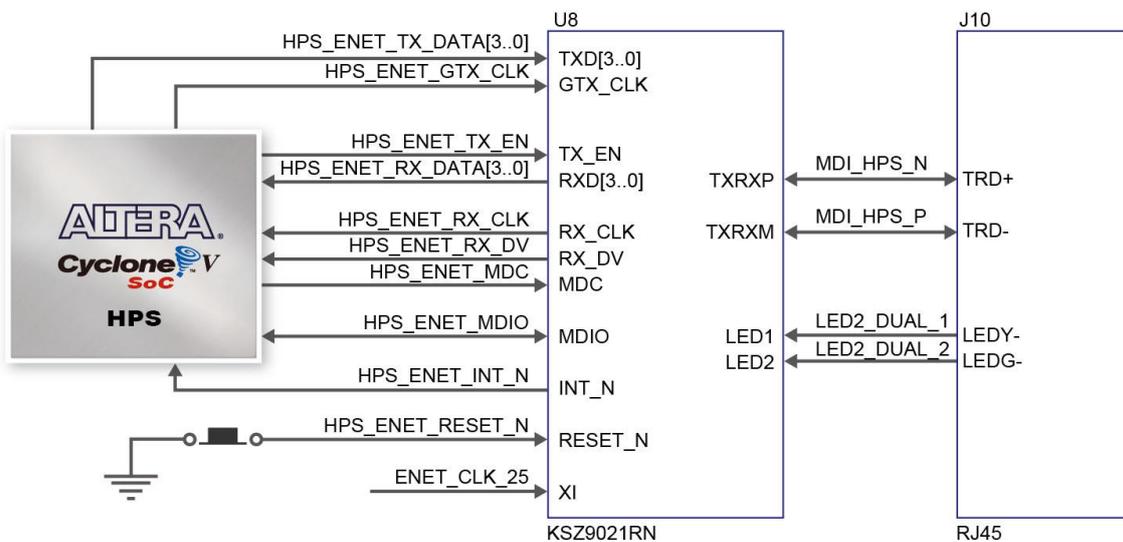


图 3-33 HPS 和千兆以太网连接示意图

表 3-27 千兆以太网 PHY 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_ENET_TX_EN	PIN_A20	GMII and MII transmit enable	3.3V
HPS_ENET_TX_DATA[0]	PIN_F20	MII transmit data[0]	3.3V
HPS_ENET_TX_DATA[1]	PIN_J19	MII transmit data[1]	3.3V
HPS_ENET_TX_DATA[2]	PIN_F21	MII transmit data[2]	3.3V
HPS_ENET_TX_DATA[3]	PIN_F19	MII transmit data[3]	3.3V
HPS_ENET_RX_DV	PIN_K17	GMII and MII receive data valid	3.3V
HPS_ENET_RX_DATA[0]	PIN_A21	GMII and MII receive data[0]	3.3V
HPS_ENET_RX_DATA[1]	PIN_B20	GMII and MII receive data[1]	3.3V
HPS_ENET_RX_DATA[2]	PIN_B18	GMII and MII receive data[2]	3.3V
HPS_ENET_RX_DATA[3]	PIN_D21	GMII and MII receive data[3]	3.3V
HPS_ENET_RX_CLK	PIN_G20	GMII and MII receive clock	3.3V
HPS_ENET_RESET_N	PIN_E18	Hardware Reset Signal	3.3V
HPS_ENET_MDIO	PIN_E21	Management Data	3.3V

HPS_ENET_MDC	PIN_B21	Management Data Clock Reference	3.3V
HPS_ENET_INT_N	PIN_C19	Interrupt Open Drain Output	3.3V
HPS_ENET_GTX_CLK	PIN_H19	GII Transmit Clock	3.3V

RJ45 接口处的绿色 LED (LEDG)和黄色 LED (LEDY)可以指示以太网(KSZ9021RNI)的状态。LED 控制信号连接到 RJ45 接口上的 LED。LEDG、LEDY 的状态和定义如表 3-28 所列。例如，当 LEDG 点亮时，DE10-Standard 开发板就与以太网建立了有效连接。

表 3-28 LED 模式引脚的状态及定义

LED (状态)		LED (定义)		网络连接/数据传输
LEDG	LEDY	LEDG	LEDY	
H	H	熄灭	熄灭	无效网络连接
L	H	点亮	熄灭	千兆以太网连通/无数据传输
Toggle	H	闪烁	熄灭	千兆以太网连通/ 有数据传输 (发送和接收)
H	L	熄灭	点亮	百兆以太网连通/无数据传输
H	Toggle	熄灭	闪烁	百兆以太网连通/有数据传输 (发送和接收)
L	L	点亮	点亮	十兆以太网连通/无数据传输
Toggle	Toggle	闪烁	闪烁	十兆以太网连通/有数据传输 (发送和接收)

### 3.7.3 UART 转 USB

DE10-Standard 开发板上有一个 UART 接口可以和 HPS 通讯。该接口不支持 HW 流量控制信号。物理接口是通过连接 FT232R 芯片和具有 USB Mini-B 接口的主设备的板载桥接器实现。有关该芯片的信息可以在制造商网站或 DE10-Standard 系统 CD 的 \Datasheet\UART TO USB 目录中查看。图 3-34 所示为 HPS、FT232R 芯片和 USB Mini-B 接口之间的连接示意图。表 3-29 列出了与 HPS 连接的 UART 接口的引脚分配。

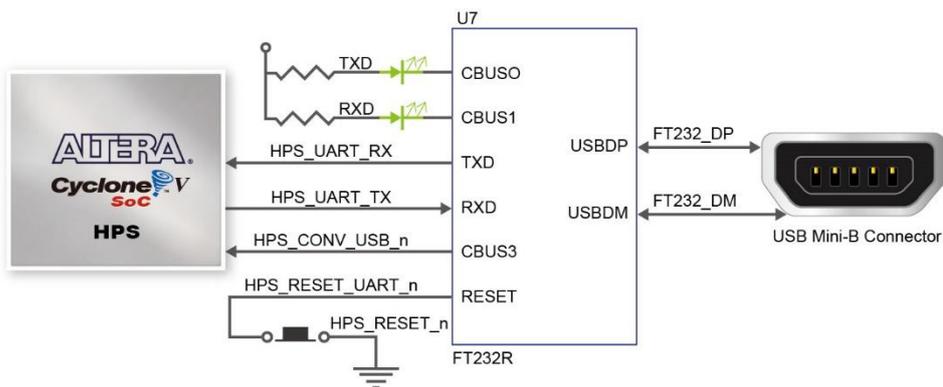


图 3-34 HPS 与 FT232R 芯片连接示意图

表 3-29 UART 接口引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_UART_RX	PIN_B25	HPS UART Receiver	3.3V
HPS_UART_TX	PIN_C25	HPS UART Transmitter	3.3V
HPS_CONV_USB_N	PIN_B15	Reserve	3.3V

### 3.7.4 DDR3 存储器

DE10-Standard 开发板支持 1GB DDR3 SDRAM，它由 HPS 端的两个 16 位 DDR3 器件组成。信号连接到 HPS I/O Banks 的专用硬核存储器控制器（Hard Memory Controller），目标速度是 400MHz。图 3-35 所示为 DDR3 和 Cyclone V SoC FPGA 的连接示意图。表 3-30 列出了 DDR3 的引脚分配、描述以及 I/O 标准。

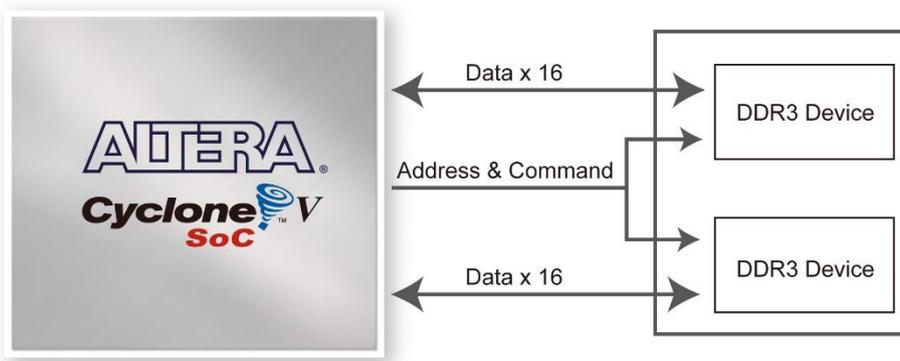


图 3-35 FPGA 与 DDR3 连接示意图

表 3-30 DDR3 内存引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_DDR3_A[0]	PIN_F26	HPS DDR3 Address[0]	SSTL-15 Class I
HPS_DDR3_A[1]	PIN_G30	HPS DDR3 Address[1]	SSTL-15 Class I
HPS_DDR3_A[2]	PIN_F28	HPS DDR3 Address[2]	SSTL-15 Class I
HPS_DDR3_A[3]	PIN_F30	HPS DDR3 Address[3]	SSTL-15 Class I
HPS_DDR3_A[4]	PIN_J25	HPS DDR3 Address[4]	SSTL-15 Class I
HPS_DDR3_A[5]	PIN_J27	HPS DDR3 Address[5]	SSTL-15 Class I
HPS_DDR3_A[6]	PIN_F29	HPS DDR3 Address[6]	SSTL-15 Class I
HPS_DDR3_A[7]	PIN_E28	HPS DDR3 Address[7]	SSTL-15 Class I
HPS_DDR3_A[8]	PIN_H27	HPS DDR3 Address[8]	SSTL-15 Class I
HPS_DDR3_A[9]	PIN_G26	HPS DDR3 Address[9]	SSTL-15 Class I
HPS_DDR3_A[10]	PIN_D29	HPS DDR3 Address[10]	SSTL-15 Class I
HPS_DDR3_A[11]	PIN_C30	HPS DDR3 Address[11]	SSTL-15 Class I
HPS_DDR3_A[12]	PIN_B30	HPS DDR3 Address[12]	SSTL-15 Class I
HPS_DDR3_A[13]	PIN_C29	HPS DDR3 Address[13]	SSTL-15 Class I
HPS_DDR3_A[14]	PIN_H25	HPS DDR3 Address[14]	SSTL-15 Class I
HPS_DDR3_BA[0]	PIN_E29	HPS DDR3 Bank Address[0]	SSTL-15 Class I

HPS_DDR3_BA[1]	PIN_J24	HPS DDR3 Bank Address[1]	SSTL-15 Class I
HPS_DDR3_BA[2]	PIN_J23	HPS DDR3 Bank Address[2]	SSTL-15 Class I
HPS_DDR3_CAS_n	PIN_E27	DDR3 Column Address Strobe	SSTL-15 Class I
HPS_DDR3_CKE	PIN_L29	HPS DDR3 Clock Enable	SSTL-15 Class I
HPS_DDR3_CK_n	PIN_L23	HPS DDR3 Clock	Differential 1.5-V SSTL Class I
HPS_DDR3_CK_p	PIN_M23	HPS DDR3 Clock p	Differential 1.5-V SSTL Class I
HPS_DDR3_CS_n	PIN_H24	HPS DDR3 Chip Select	SSTL-15 Class I
HPS_DDR3_DM[0]	PIN_K28	HPS DDR3 Data Mask[0]	SSTL-15 Class I
HPS_DDR3_DM[1]	PIN_M28	HPS DDR3 Data Mask[1]	SSTL-15 Class I
HPS_DDR3_DM[2]	PIN_R28	HPS DDR3 Data Mask[2]	SSTL-15 Class I
HPS_DDR3_DM[3]	PIN_W30	HPS DDR3 Data Mask[3]	SSTL-15 Class I
HPS_DDR3_DQ[0]	PIN_K23	HPS DDR3 Data[0]	SSTL-15 Class I
HPS_DDR3_DQ[1]	PIN_K22	HPS DDR3 Data[1]	SSTL-15 Class I
HPS_DDR3_DQ[2]	PIN_H30	HPS DDR3 Data[2]	SSTL-15 Class I
HPS_DDR3_DQ[3]	PIN_G28	HPS DDR3 Data[3]	SSTL-15 Class I
HPS_DDR3_DQ[4]	PIN_L25	HPS DDR3 Data[4]	SSTL-15 Class I
HPS_DDR3_DQ[5]	PIN_L24	HPS DDR3 Data[5]	SSTL-15 Class I
HPS_DDR3_DQ[6]	PIN_J30	HPS DDR3 Data[6]	SSTL-15 Class I
HPS_DDR3_DQ[7]	PIN_J29	HPS DDR3 Data[7]	SSTL-15 Class I
HPS_DDR3_DQ[8]	PIN_K26	HPS DDR3 Data[8]	SSTL-15 Class I
HPS_DDR3_DQ[9]	PIN_L26	HPS DDR3 Data[9]	SSTL-15 Class I
HPS_DDR3_DQ[10]	PIN_K29	HPS DDR3 Data[10]	SSTL-15 Class I
HPS_DDR3_DQ[11]	PIN_K27	HPS DDR3 Data[11]	SSTL-15 Class I
HPS_DDR3_DQ[12]	PIN_M26	HPS DDR3 Data[12]	SSTL-15 Class I
HPS_DDR3_DQ[13]	PIN_M27	HPS DDR3 Data[13]	SSTL-15 Class I
HPS_DDR3_DQ[14]	PIN_L28	HPS DDR3 Data[14]	SSTL-15 Class I
HPS_DDR3_DQ[15]	PIN_M30	HPS DDR3 Data[15]	SSTL-15 Class I
HPS_DDR3_DQ[16]	PIN_U26	HPS DDR3 Data[16]	SSTL-15 Class I
HPS_DDR3_DQ[17]	PIN_T26	HPS DDR3 Data[17]	SSTL-15 Class I
HPS_DDR3_DQ[18]	PIN_N29	HPS DDR3 Data[18]	SSTL-15 Class I
HPS_DDR3_DQ[19]	PIN_N28	HPS DDR3 Data[19]	SSTL-15 Class I
HPS_DDR3_DQ[20]	PIN_P26	HPS DDR3 Data[20]	SSTL-15 Class I
HPS_DDR3_DQ[21]	PIN_P27	HPS DDR3 Data[21]	SSTL-15 Class I
HPS_DDR3_DQ[22]	PIN_N27	HPS DDR3 Data[22]	SSTL-15 Class I
HPS_DDR3_DQ[23]	PIN_R29	HPS DDR3 Data[23]	SSTL-15 Class I
HPS_DDR3_DQ[24]	PIN_P24	HPS DDR3 Data[24]	SSTL-15 Class I
HPS_DDR3_DQ[25]	PIN_P25	HPS DDR3 Data[25]	SSTL-15 Class I
HPS_DDR3_DQ[26]	PIN_T29	HPS DDR3 Data[26]	SSTL-15 Class I
HPS_DDR3_DQ[27]	PIN_T28	HPS DDR3 Data[27]	SSTL-15 Class I
HPS_DDR3_DQ[28]	PIN_R27	HPS DDR3 Data[28]	SSTL-15 Class I
HPS_DDR3_DQ[29]	PIN_R26	HPS DDR3 Data[29]	SSTL-15 Class I
HPS_DDR3_DQ[30]	PIN_V30	HPS DDR3 Data[30]	SSTL-15 Class I
HPS_DDR3_DQ[31]	PIN_W29	HPS DDR3 Data[31]	SSTL-15 Class I
HPS_DDR3_DQS_n[0]	PIN_M19	HPS DDR3 Data Strobe n[0]	Differential 1.5-V SSTL Class I

HPS_DDR3_DQS_n[1]	PIN_N24	HPS DDR3 Data Strobe n[1]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_n[2]	PIN_R18	HPS DDR3 Data Strobe n[2]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_n[3]	PIN_R21	HPS DDR3 Data Strobe n[3]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[0]	PIN_N18	HPS DDR3 Data Strobe p[0]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[1]	PIN_N25	HPS DDR3 Data Strobe p[1]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[2]	PIN_R19	HPS DDR3 Data Strobe p[2]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[3]	PIN_R22	HPS DDR3 Data Strobe p[3]	Differential 1.5-V SSTL Class I
HPS_DDR3_ODT	PIN_H28	HPS DDR3 On-die Termination	SSTL-15 Class I
HPS_DDR3_RAS_n	PIN_D30	DDR3 Row Address Strobe	SSTL-15 Class I
HPS_DDR3_RESET_n	PIN_P30	HPS DDR3 Reset	SSTL-15 Class I
HPS_DDR3_WE_n	PIN_C28	HPS DDR3 Write Enable	SSTL-15 Class I
HPS_DDR3_RZQ	PIN_D27	External reference ball for output drive calibration	1.5 V

### 3.7.5 Micro SD 卡槽

DE10-Standard开发板支持x4数据线的Micro SD卡接口。它不仅为HPS提供了外部存储器，同时也是开发板的备用启动选项。图3-36所示为HPS与Micro SD卡槽的连接示意图。表 3-31列出了连接HPS的Micro SD卡槽引脚分配。

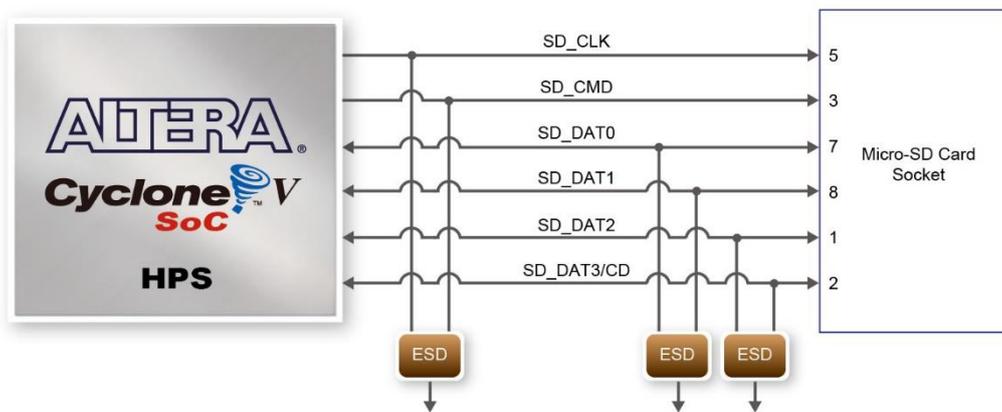


图 3-36 HPS 与 Micro SD 卡槽的连接示意图

表 3-31 Micro SD 卡槽引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_SD_CLK	PIN_A16	HPS SD Clock	3.3V
HPS_SD_CMD	PIN_F18	HPS SD Command Line	3.3V
HPS_SD_DATA[0]	PIN_G18	HPS SD Data[0]	3.3V

HPS_SD_DATA[1]	PIN_C17	HPS SD Data[1]	3.3V
HPS_SD_DATA[2]	PIN_D17	HPS SD Data[2]	3.3V
HPS_SD_DATA[3]	PIN_B16	HPS SD Data[3]	3.3V

### 3.7.6 USB 主设备端口

DE10-Standard 开发板有两个 USB 2.0 A 型主设备端口，每个端口都连接一个 SMSC USB3300 控制器和一个 2 端口集线（HUB）控制器。32 引脚、QFN 封装的 SMSC USB3300 器件与 SMSC USB2512B 集线控制器连接。该器件支持通过 UTM 协议与 HPS 中的 USB 2.0 控制器通讯。将 USB3300 的 ID 引脚接地，PHY 可以在主设备模式下工作。当在主设备模式下工作时，USB3300 通过两个 USB A 型端口供电。图 3-37 所示为 USB OTG PHY 与 HPS 的连接示意图。表 3-32 列出了 USB OTG PHY 与 HPS 连接的引脚分配。

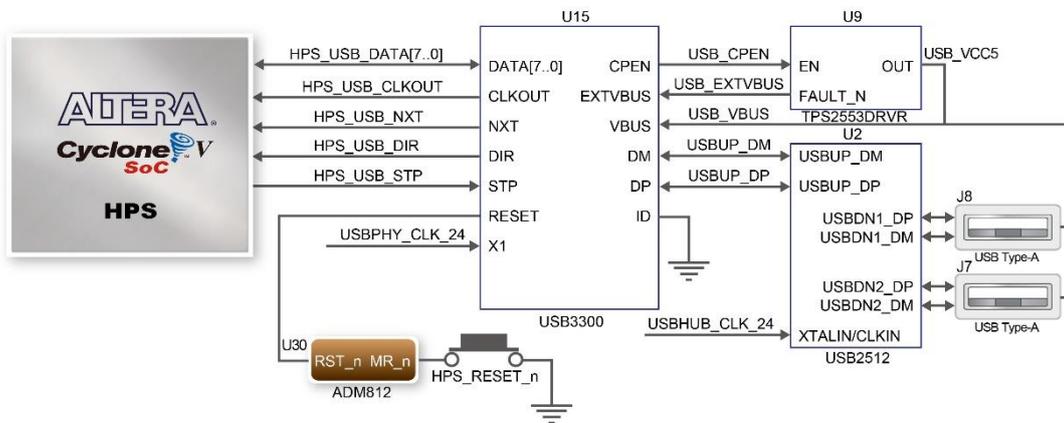


图 3-37 HPS 与 USB OTG PHY 连接示意图

表 3-32 USB OTG PHY 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_USB_CLKOUT	PIN_N16	60MHz 参考时钟输出	3.3V
HPS_USB_DATA[0]	PIN_E16	HPS USB_DATA[0]	3.3V
HPS_USB_DATA[1]	PIN_G16	HPS USB_DATA[1]	3.3V
HPS_USB_DATA[2]	PIN_D16	HPS USB_DATA[2]	3.3V
HPS_USB_DATA[3]	PIN_D14	HPS USB_DATA[3]	3.3V
HPS_USB_DATA[4]	PIN_A15	HPS USB_DATA[4]	3.3V
HPS_USB_DATA[5]	PIN_C14	HPS USB_DATA[5]	3.3V
HPS_USB_DATA[6]	PIN_D15	HPS USB_DATA[6]	3.3V
HPS_USB_DATA[7]	PIN_M17	HPS USB_DATA[7]	3.3V
HPS_USB_DIR	PIN_E14	数据总线方向	3.3V
HPS_USB_NXT	PIN_A14	限制数据传输	3.3V
HPS_USB_RESET	PIN_G17	HPS USB PHY 复位	3.3V
HPS_USB_STP	PIN_C15	停止总线上的数据流	3.3V

### 3.7.7 加速器 (G-sensor)

DE10-Standard开发板搭载一个数字加速传感器模块ADXL345，俗称重力传感器（G-sensor）。该传感器是一款小而薄的超低功耗三轴加速计，分辨率高。数字输出数据为16位二进制补码格式，可通过I2C接口访问。重力传感器的I2C地址为0xA6/0xA7。有关该芯片的更多信息可以参考芯片数据表，在制造商官网或DE10-Standard系统CD的\Datasheet\G-sensor目录中可以找到该数据表。图3-38所示为HPS与重力感应器的连接示意图。表3-33列出了重力感应器与HPS连接的引脚分配。

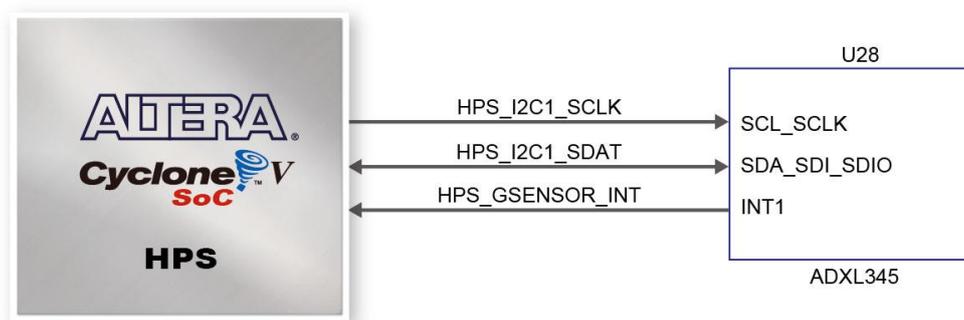


图 3-38 Cyclone V SoC FPGA 与 G-Sensor 连接示意图

表 3-33 G-senor 引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_GSENSOR_INT	PIN_B22	HPS GSENSOR 中断输出	3.3V
HPS_I2C1_SCLK	PIN_E23	HPS I2C 时钟 (与 LTC 共享总线)	3.3V
HPS_I2C1_SDAT	PIN_C24	HPS I2C 数据 (共享总线)	3.3V

### 3.7.8 LTC 连接头

DE10-Standard 开发板上有一个 14 引脚连接头，最初是设计用来连接 Linear Technology 的各种不同子卡进行通信。它连接到 SPI Master 和 HPS 的 I2C 端口，与这两种协议进行双向通信。该 14 引脚连接头可与 HPS 进行基于 GPIO、SPI 或 I2C 协议的通讯。图-39 所示为 HPS 和 LTC 的连接示意图，表 3-34 列出了 LTC 连接头的引脚分配。

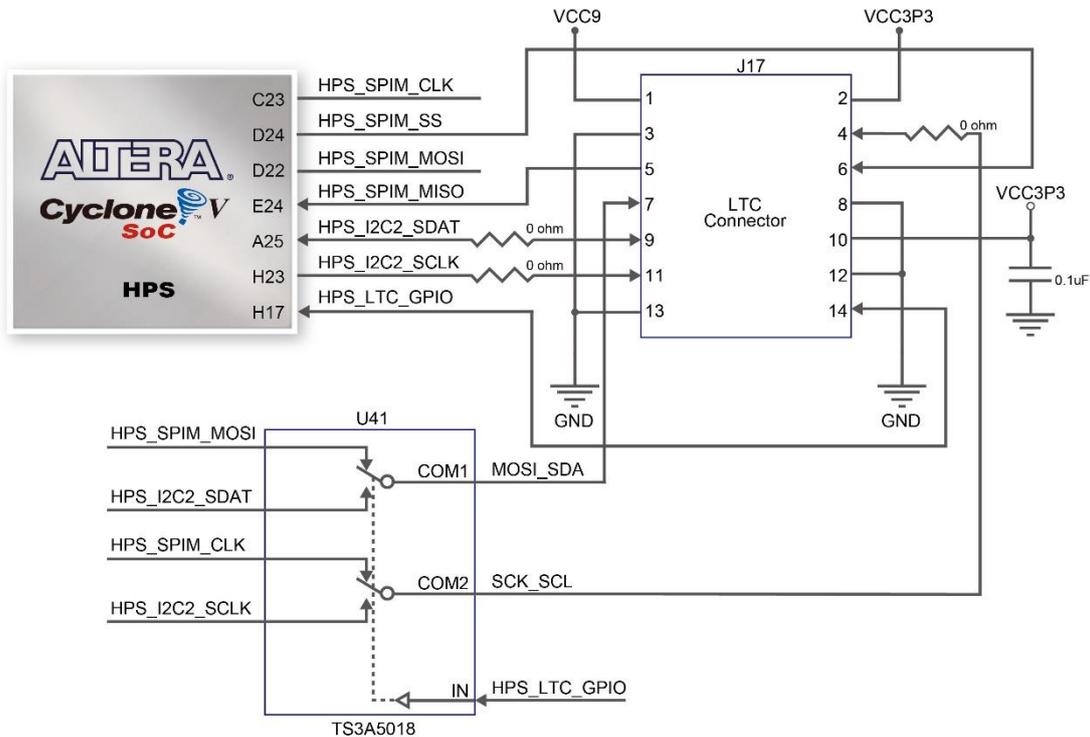


图-39 HPS 与 LTC 连接头连接示意图

表 3-34 LTC 连接头引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_LTC_GPIO	PIN_H17	HPS LTC GPIO	3.3V
HPS_I2C2_SCLK	PIN_H23	HPS I2C2 Clock (share bus with G-Sensor)	3.3V
HPS_I2C2_SDAT	PIN_A25	HPS I2C2 Data (share bus with G-Sensor)	3.3V
HPS_SPIM_CLK	PIN_C23	SPI Clock	3.3V
HPS_SPIM_MISO	PIN_E24	SPI Master Input/Slave Output	3.3V
HPS_SPIM_MOSI	PIN_D22	SPI Master Output /Slave Input	3.3V
HPS_SPIM_SS	PIN_D24	SPI Slave Select	3.3V

### 3.7.9 128x64 点阵 LCD

DE10-Standard 开发板搭载一个 128x64 点阵式 LCD 模块，实现显示器功能。LCD 模块使用串行外设接口与 HPS 连接。用户可以参考 DE10-Standard 系统 CD 里的数据手册来使用 LCD 模块。图 3-40 所示为 HPS 和 LCD 模块的连接示意图。通过短接 JP4 的两个引脚，LCD 的背光电源默认设置为接通状态。表 3-35 列出了 LCD 模块连接 Cyclone V SoC FPGA 的引脚分配。

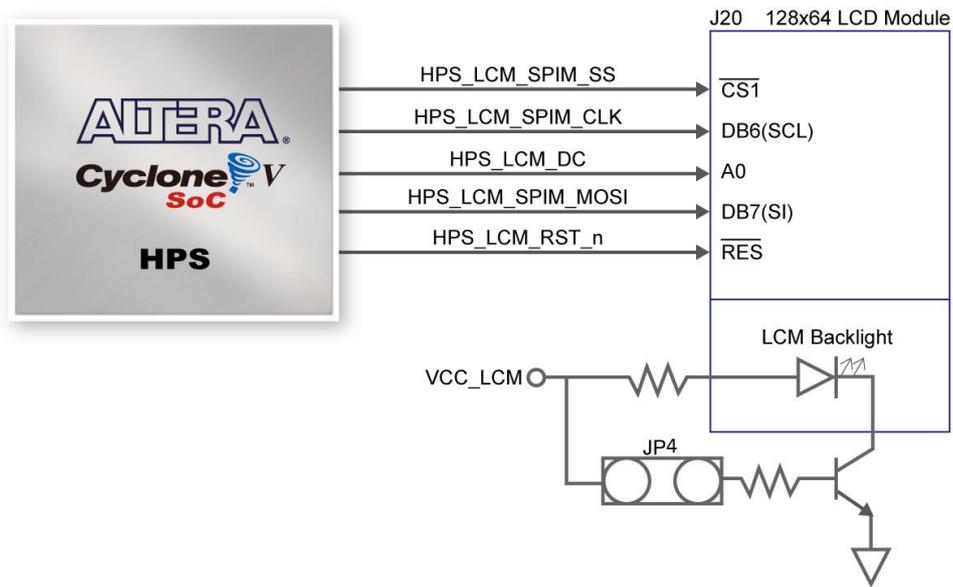


图 3-40 Cyclone V SoC FPGA 与 LCD 模块连接示意图

表 3-35 LCD 模块引脚分配

信号名	FPGA 引脚号	描述	I/O 标准
HPS_LCM_D_C	PIN_C18	HPS LCM Data bit is Data/Command	3.3V
HPS_LCM_RST_N	PIN_E17	HPS LCM Reset	3.3V
HPS_LCM_SPIM_CLK	PIN_A23	SPI Clock	3.3V
HPS_LCM_SPIM_MOSI	PIN_C22	SPI Master Output /Slave Input	3.3V
HPS_LCM_SPIM_SS	PIN_H20	SPI Slave Select	3.3V

# 第四章

## DE10-Standard System Builder

开发人员可以基于DE10-Standard系统CD中的golden\_top Quartus工程开始设计自定义Quartus工程。如果计划搭配使用Terasic的子卡，或者只使用DE10-Standard开发板上的部分接口，System Builder工具可以帮助开发人员在短短几分钟内创建Quartus工程。

本章将描述如何使用DE10-Standard System Builder工具创建用户自定义工程。

### 4.1 简介

DE10-Standard System Builder是一款基于Windows的工具，旨在帮助用户快速为DE10-Standard创建Quartus II工程。产生的Quartus II工程包括以下文件：

- Quartus II project file (.qpf)
- Quartus II setting file (.qsf)
- Top-level design file (.v or .vhd)
- Synopsis design constraints file (.sdc)
- Pin assignment document (.htm)

上述由DE10-Standard System Builder生成的文件可以避免用户手动编辑顶层设计文件或分配引脚时出现编译错误的情况。用户遇到的常见错误有：

- Bank电压或引脚分配错误导致DE10-Standard开发板损坏
- 器件选择错误、引脚位置或方向声明错误或缺失引起DE10-Standard开发板故障
- 不合理的引脚分配引起DE10-Standard开发板性能退化

### 4.2 设计流程

本节将介绍用DE10-Standard System Builder创建Quartus II工程的设计流程。图4-1阐述了该设计流程。

用户启动DE10-Standard System Builder并根据自定义设计需求创建新的工程之后，DE10-Standard System Builder会生成两个主要文件：顶层设计文件（.v或.vhd）和Quartus II工程设置文件(.qsf)。

顶层设计文件包含顶层Verilog或VHDL HDL描述文件，供用户添加自定义设计或逻辑。

Quartus II 工程设置文件包含诸如 FPGA 器件类型、顶层引脚分配及每一个用户自定义 I/O 引脚的 I/O 电平标准等。

最后，Quartus II Programmer 通过 JTAG 接口将 .sof 文件下载至 DE10-Standard 开发板。

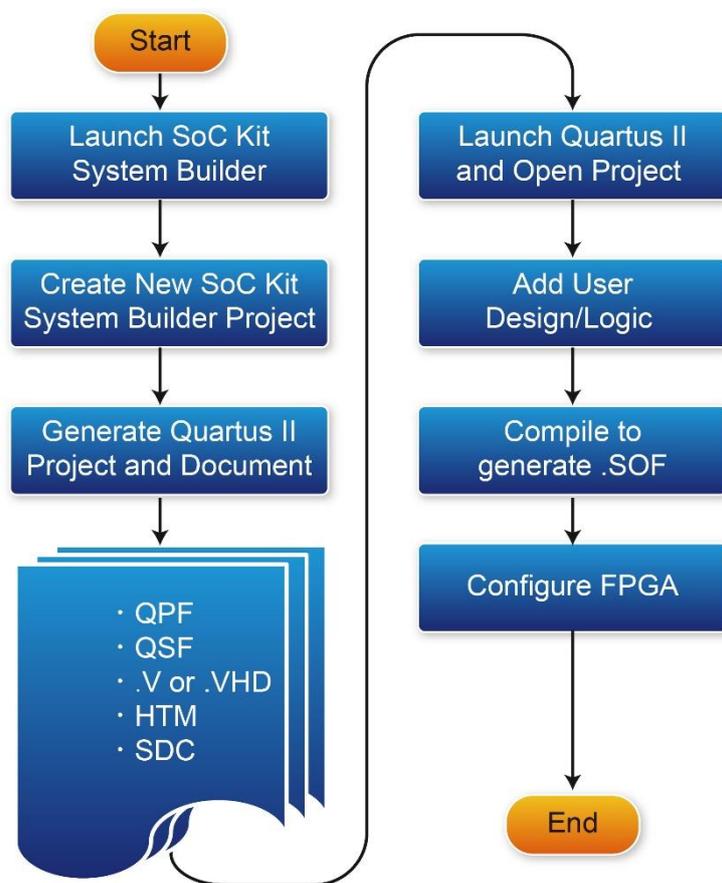


图 4-1 DE10-Standard System Builder 创建工程的设计流程

### 4.3 使用 DE10-Standard System Builder

本节将详细介绍如何使用 DE10-Standard System Builder 工具。

#### ■ 安装和启动 DE10-Standard System Builder

DE10-Standard System Builder 在 DE10-Standard 系统 CD 的 Tools\SystemBuilder 目录中。用户只需要将整个 SystemBuilder 文件夹复制到主机而不用安装该工具。在主机上执行 DE10-Standard SystemBuilder.exe 后会弹出如图 4-2 所示的窗口。

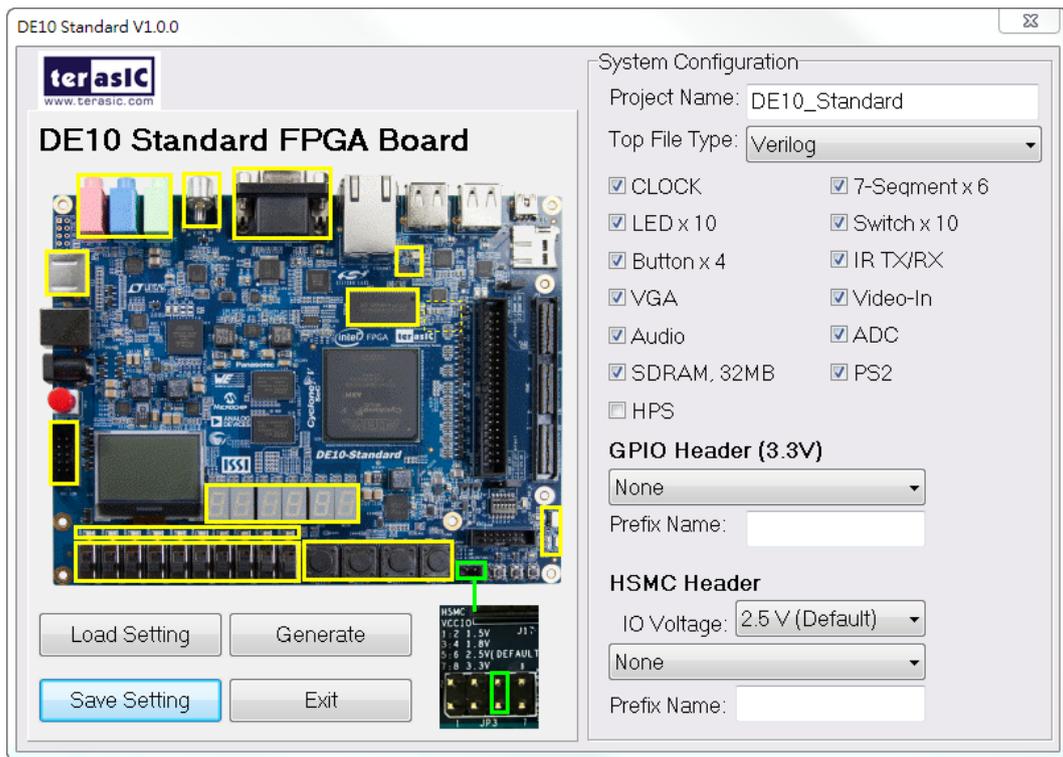


图 4-2 DE10-Standard System Builder 图形用户界面

## ■ 输入工程名称

如图 4-3 所示，在绿色圈所示的 Project Name 空白处输入工程名称。  
输入的工程名称将自动定义为顶层设计实体的名称。

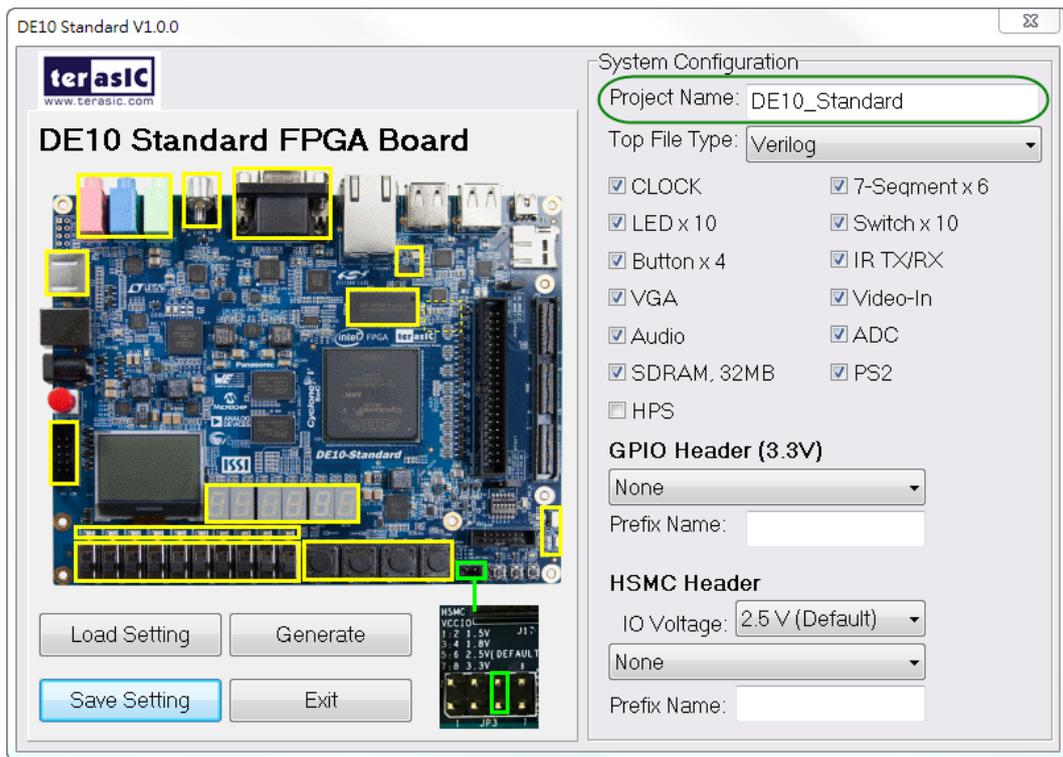


图 4-3 输入工程名称

## ■ 顶层文件类型

如图 4-4 所示，选择顶层文件的 HDL 类型。如果选择 Verilog，将产生文件类型为.v 的 Verilog HDL 顶层文件；如果选择 VHDL，将产生文件类型为.vhd 的 VHDL 顶层文件。

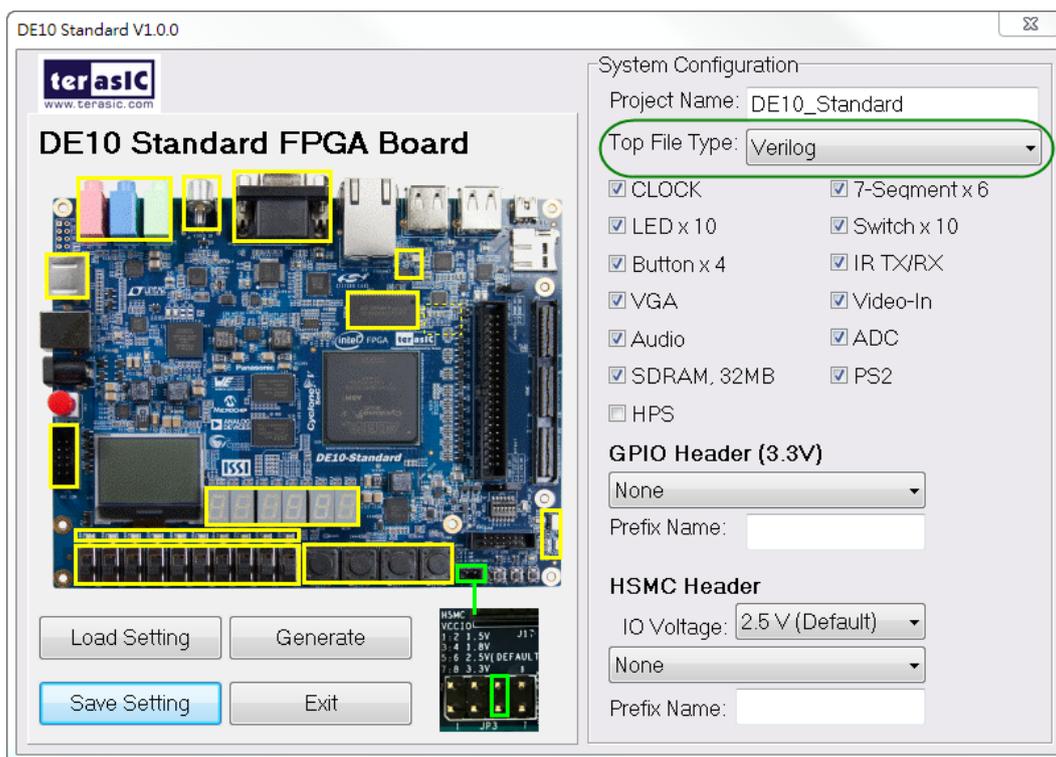


图 4-4 选择顶层文件类型

## ■ 系统配置

如图 4-5 所示，用户可以在工程中灵活选择并添加开发板上的组件。开发板上的每个组件都有列出，用户可以根据自己的想法启用或禁用一个或多个组件。如果启用了某个组件，DE10-Standard System Builder 会自动生成与其相关的引脚分配，包括引脚名、引脚位置、引脚方向和 I/O 标准。

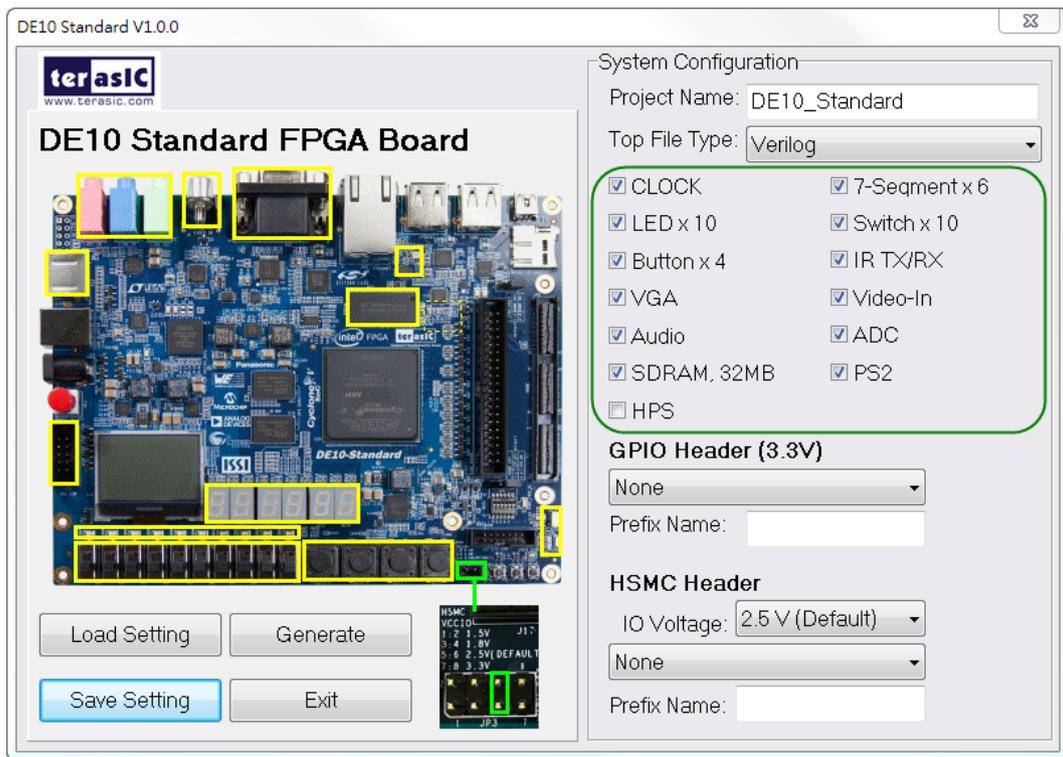


图 4-5 系统配置栏

## ■ GPIO 和 HSMC 扩展

如图4-6所示,如果用户将基于GPIO或HSMC的任意一个Terasic子卡连接到DE10-Standard开发板的GPIO或HSMC接口, DE10-Standard System Builder会生成包含相应模块的工程, 它还将自动生成相关的引脚分配, 包括引脚名、引脚位置、引脚方向和I/O标准。

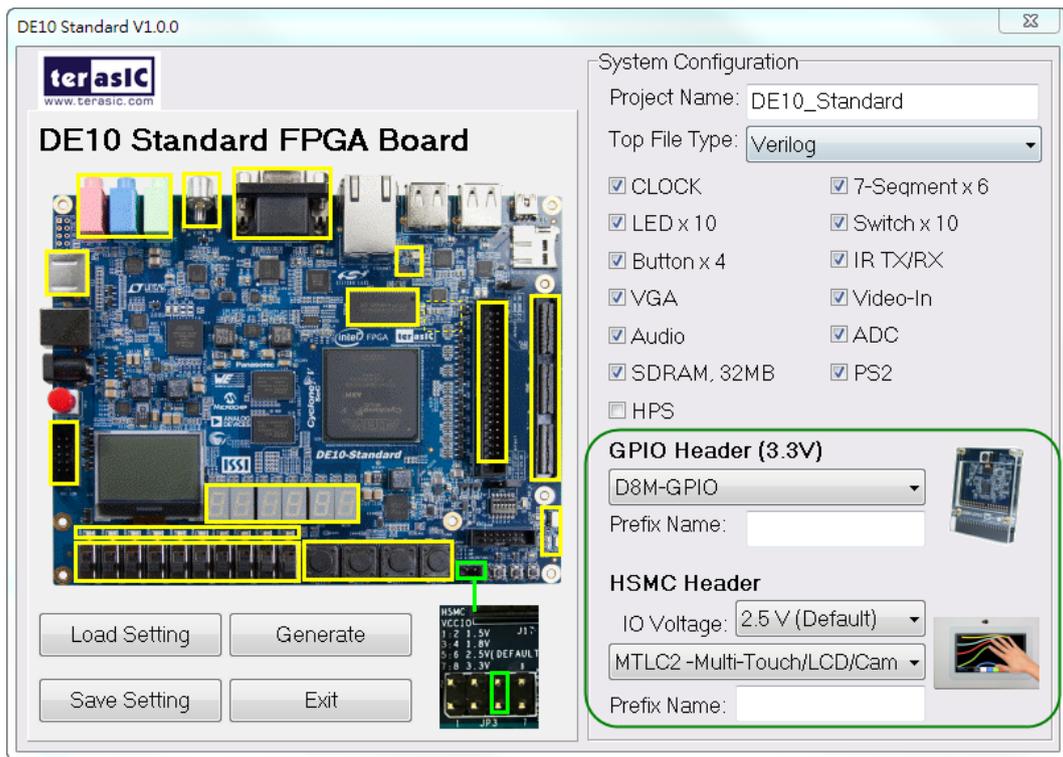


图 4-6 GPIO 与 HSMC 扩展栏

Prefix Name 是选填项，表示在设计中分配的子卡引脚名，这一项可以忽略不填。

## ■ 工程设置管理

如图 4-7 所示，DE10-Standard System Builder 还可以加载某种设置或将开发板当前的配置保存在.cfg 文件中。

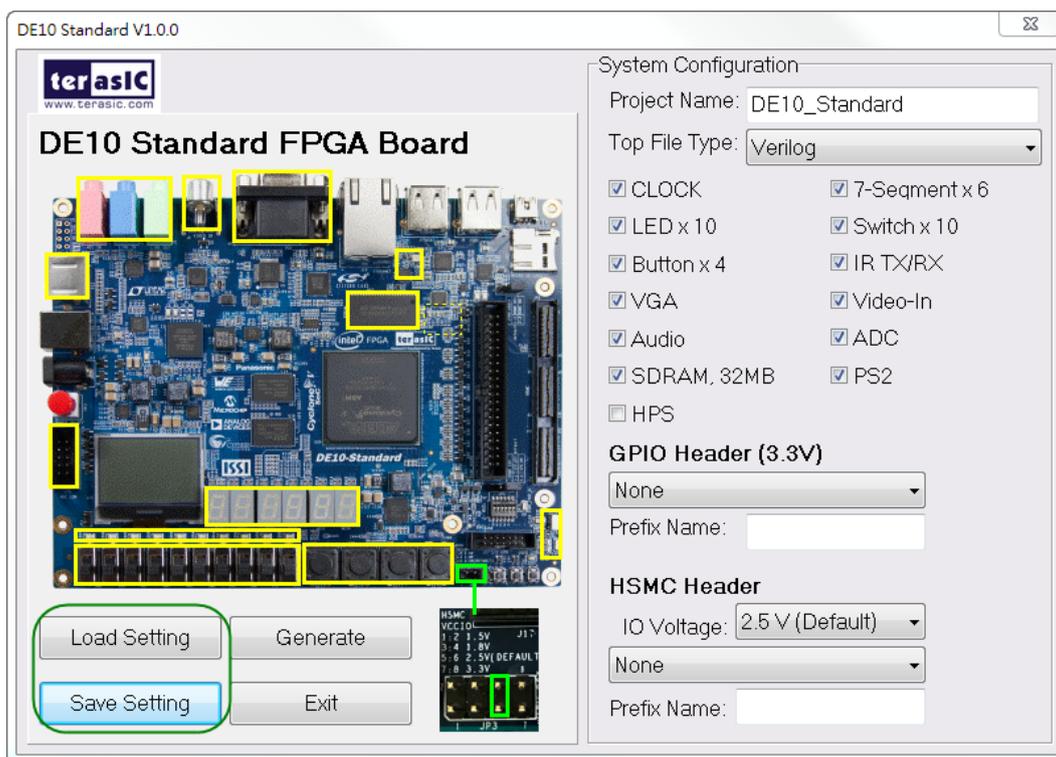


图 4-7 工程设置

## ■ 工程生成

按下 Generate 按钮后，DE10-Standard System Builder 会生成相应的 Quartus II 文件和文档，如表 4-1 所列。

表 4-1 DE10-Standard System Builder 生成的文件

No.	文件名	描述
1	<Project name>.v or .vhd	顶层 Verilog 或 VHDL HDL Quartus II 文件
2	<Project name>.qpf	Quartus II 工程文件
3	<Project name>.qsf	Quartus II 设置文件
4	<Project name>.sdc	Quartus II 设计约束文件
5	<Project name>.htm	引脚分配文档

用户可以用 Quartus Prime 软件打开工程再添加自定义逻辑，然后编译工程生成 SRAM Object File (.sof)。

# 第五章

## FPGA 设计示例

本章提供了许多基于RTL或Qsys并且在DE10-Standard开发板上实现的FPGA高阶设计示例。这些参考设计涵盖了连接FPGA的外围设备的许多功能特性，例如音频、SDRAM及IR接收器。所有相关的文件都能在DE10-Standard系统CD的\Demonstration\FPGA目录中找到。

### ■ 安装设计示例

在主机上安装设计示例：

将整个Demonstration文件夹复制到您所选的PC本地目录中，要确保本地目录的路径不包含空格，否则在Nios II EDS中运行时会出现报错。**注意，必须安装v16.1或更高版本的Quartus Prime软件运行DE10-Standard设计示例。**

### 5.1 DE10-Standard 默认配置

DE10-Standard 开发板预烧录了默认配置数据，用来演示开发板的一些基本功能特性。以下介绍了该默认示例文件所在的位置以及运行示例时的设置。

#### ■ 演示设置、文件位置和操作说明

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_Default。
- 配置文件：DE10\_Standard\_Default.sof或DE10\_Standard\_Default.jic。
- 用USB线连接开发板的USB-Blaster II接口和PC的USB接口，将12V直流电源线连接开发板的电源接口(J14)和电源插座，按下电源开关SW11接通DE10-Standard开发板电源。如有必要（比如当前EPCS器件没有烧录出厂默认配置），需要通过JTAG接口下载数据到开发板。
  - 用户现在能观察到七段数码管按顺序显示0~F，并且红色LED在闪烁。
  - 如果VGA D-SUB接口与VGA显示器连接，显示器会显示DE10-Standard开发板的彩色图像。
  - 如果立体声line-out插孔连接到扬声器，按下KEY[1]后line-out端口会有频率为1KHz的嗡嗡声。
- 为了方便运行示例，本工程提供了demo\_batch文件夹。它不仅能将数据以命令行的形

式加载到FPGA，还能通过执行test.bat文件将.jic文件烧写到EPCS或擦除.jic文件，如图5-1所示。

- 如果用户想将新的设计烧写到EPCS器件，最便捷的办法就是将新的.sof文件复制到demo\_batch 文件夹并执行test.bat。运行选项“2”可以将.sof文件转换为.jic文件，运行选项“3”可以将.jic文件烧写到EPCS器件。

```
*****
Please choose your operation
"1" for programming .sof to FPGA.
"2" for converting .sof to .jic
"3" for programming .jic to EPCS.
"4" for erasing .jic from EPCS.
"5" for EXIT batch.
*****
Please enter your choice: [1,2,3,4,5]?_
```

图 5-1 批处理文件的命令行配置 FPGA 和 EPCS 器件

## 5.2 音频录制与回放

本设计示例展示了如何使用 DE10-Standard 开发板搭载的音频编解码芯片来实现音频录制和播放。该示例是基于 Qsys 与 Nios II EDS 开发的。图 5-2 所示为开发板上和这个示例互动的按钮和拨动开关。用户可以通过这两个按钮开关和四个拨动开关配置该音频系统。

- SW0用于指定录制的音源是Line-in或MIC-In。
- SW1、SW2和SW3用于指定录音采样率，如96K、48K、44.1K、32K或8K。
- 表 5-1与表 5-2总结了用于配置音频录制和播放的拨动开关的用法。

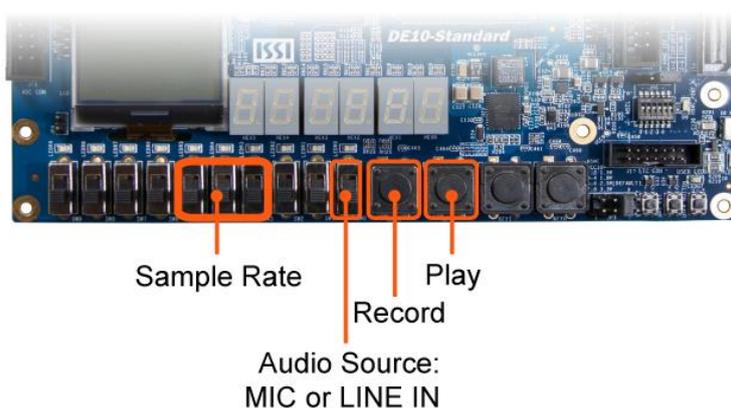


图 5-2 音频录制和播放的按钮及开关

图 5-3 所示为音频录制和播放器设计的系统框图。系统框图中有硬件与软件两部分。软件部分将 Nios II 程序存储在片上内存中，该程序是在 Nios II EDS 软件环境下用 C 语言编写

的。硬件部分是在 Quartus Prime 软件的 Qsys 环境下创建的。硬件部分包括所有其他模块，例如用户自定义的 Qsys 组件 AUDIO Controller，该组件用于发送音频数据到音频芯片或接收来自音频芯片的音频数据。音频芯片是通过以 C 语言实现的 I2C 协议进行配置的，音频芯片的 I2C 引脚通过 PIO 控制器连接到 Qsys 系统互联总线。该示例中音频芯片配置成 Master 模式，音频接口配置成 16 位 I2S 模式。PLL 生成的 18.432MHz 时钟信号通过音频控制器连接到音频芯片的 MCLK/XTI 引脚。

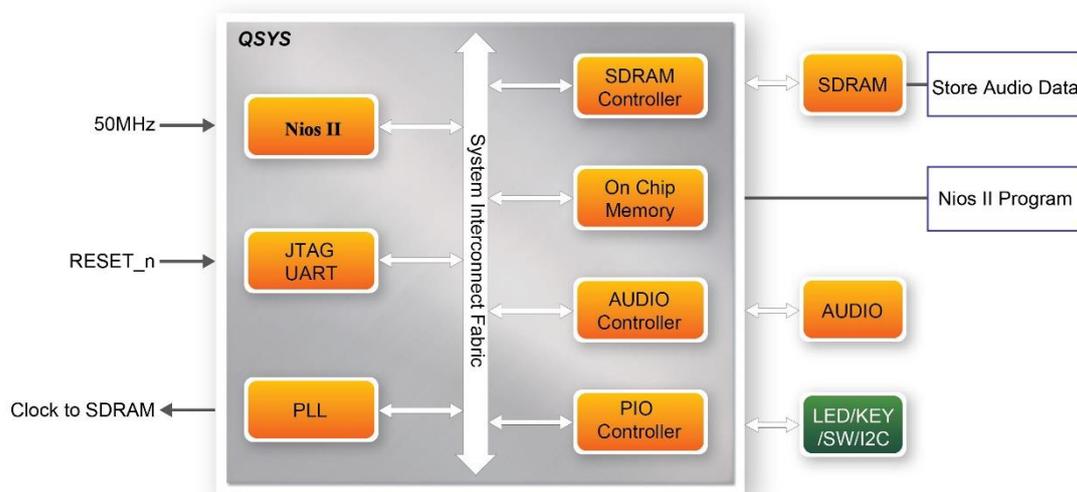


图 5-3 音频录制与播放器系统框图

## ■ 演示设置、文件位置及操作说明

- 硬件工程目录：\Demonstration\FPGA\DE10\_Standard\_Audio。
- 配置文件：DE10\_Standard\_Audio.sof。
- 软件工程目录：DE10\_Standard\_Audio\software。
- 连接音频源到Line-in接口。
- 连接麦克风到MIC-in接口。
- 连接扬声器或耳机到Line-out接口。
- 加载FPGA配置数据 (注意\*1)。
- 加载软件可执行文件到FPGA (注意\*1)。
- 如表 5-1所示，用SW0配置音频。
- 按KEY3，开始/停止音频录制 (注意\*2)。
- 按KEY2，开始/停止音频播放 (注意\*3)。

表 5-1 用于选择音频源的拨动开关

拨动开关	0 – DOWN Position	1 – UP Position
SW0	MIC-in 输入音源	Line-in 输入音源

表 5-2 用于音频录制与播放器采样率的拨动开关设置

SW5 (0 – DOWN; 1- UP)	SW4 (0 – DOWN; 1-UP)	SW3 (0 – DOWN; 1-UP)	Sample Rate
0	0	0	96K
0	0	1	48K
0	1	0	44.1K
0	1	1	32K
1	0	0	8K
Unlisted combination			96K



注意:

1. 执行 DE10\_Standard\_Audio\demo\_batch\test.bat，下载.sof 和.elf 文件。
2. 如果音频缓冲区已满，录制进程将停止。
3. 如果音频数据播放完毕，播放进程将停止。

## 5.3 卡拉 OK 机

本示例使用DE10-Standard开发板上的microphone-in、line-in和line-out接口创建卡拉OK机。WM8731编解码芯片配置成Master模式。音频编解码芯片自动生成AD/DA串行位时钟信号(BCK) 和左/右声道时钟信号(LRCK)。如图5-4所示，I2C接口用于配置音频编解码芯片，芯片的采样率和增益用相似的方式设置，来自line-in接口的输入数据与microphone-in接口的输入数据混合后发送到line-out接口。

该示例中采样率设置为48KHz。按下KEY0后，音频编解码器的增益会通过I2C总线重新配置，增益值(音量级别)在10个预定义值之间循环。

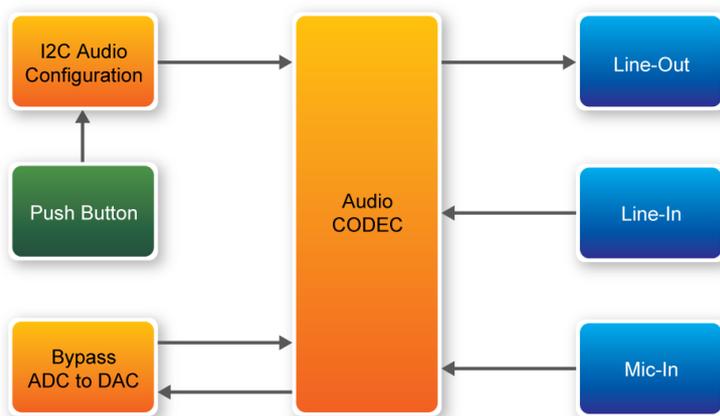


图 5-4 卡拉 OK 机示例系统框图

## ■ 演示设置、文件位置及操作说明

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_i2sound。
- 配置文件：DE10\_Standard\_i2sound.sof。
- 连接麦克风到microphone-in接口（粉红色）。
- 连接音乐播放器的音频输出到line-in接口（蓝色），例如MP3或者电脑。
- 连接耳机/扬声器到line-out 接口（绿色）。
- 执行DE10\_Standard\_i2sound\demo\_batch 目录中的批处理文件test.bat，加载配置数据到FPGA。

据到FPGA。

- 用户可以听到来自音乐播放器和麦克风的混合声音。
- 按KEY0调整音量，音量会在0到9级间循环。

图 5-5 所示为运行该示例的设置。

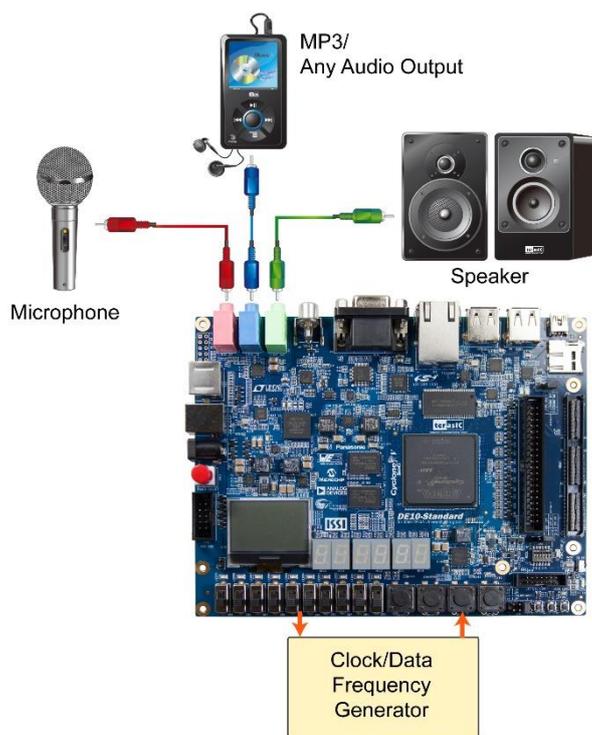


图 5-5 卡拉 OK 示例设置

## 5.4 Nios II SDRAM 测试

很多应用程序会使用SDRAM作为临时存储器。本示例分别从硬件和软件介绍如何在Qsys中进行内存访问，并演示了Intel SDRAM Controller IP如何访问SDRAM，以及Nios II处理器如何读/写SDRAM进行硬件验证。SDRAM控制器处理访问SDRAM时遇到的许多复杂的问题，例如初始化存储设备、管理SDRAM模块以及保证设备在特定时间间隔内刷新。

### ■ 系统方框图

图5-6所示为本示例的系统框图。该系统需要开发板提供50MHz时钟信号。SDRAM配置成64MB控制器。SDRAM控制器的工作频率为100MHz，Nios II程序在片上内存中运行。

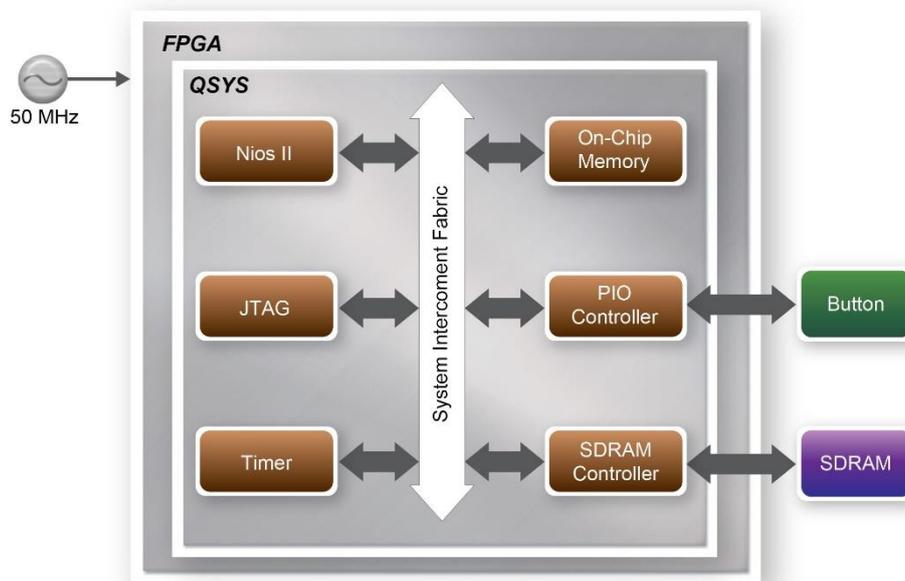


图 5-6 Nios II SDRAM 测试系统方框图

在Nios II EDS软件中运行的某个程序控制这个系统流程。在调用Nios II系统函数alt\_dcache\_flush\_all之前，Nios II程序首先将测试模式写入整个64MB SDRAM，确保所有数据都写入SDRAM。然后从SDRAM读取数据进行数据验证。当写入数据到SDRAM或从SDRAM读取数据时，该程序会在Nios II终端显示程序进程。当验证进度达到100%，Nios终端将显示结果。

## ■ 设计工具

- Quartus Prime v16.1
- Nios II EDS v16.1

## ■ 示例源代码

- Quartus硬件工程目录：\Demonstration\FPGA\SDRAM\_Nios\_Test
- Nios II EDS软件目录：SDRAM\_Nios\_Test \Software

## ■ Nios II 工程编译

- 在Nios II EDS编译参考设计之前，点击Nios II EDS软件Project菜单中的**Clean**。

## ■ 示例批处理文件

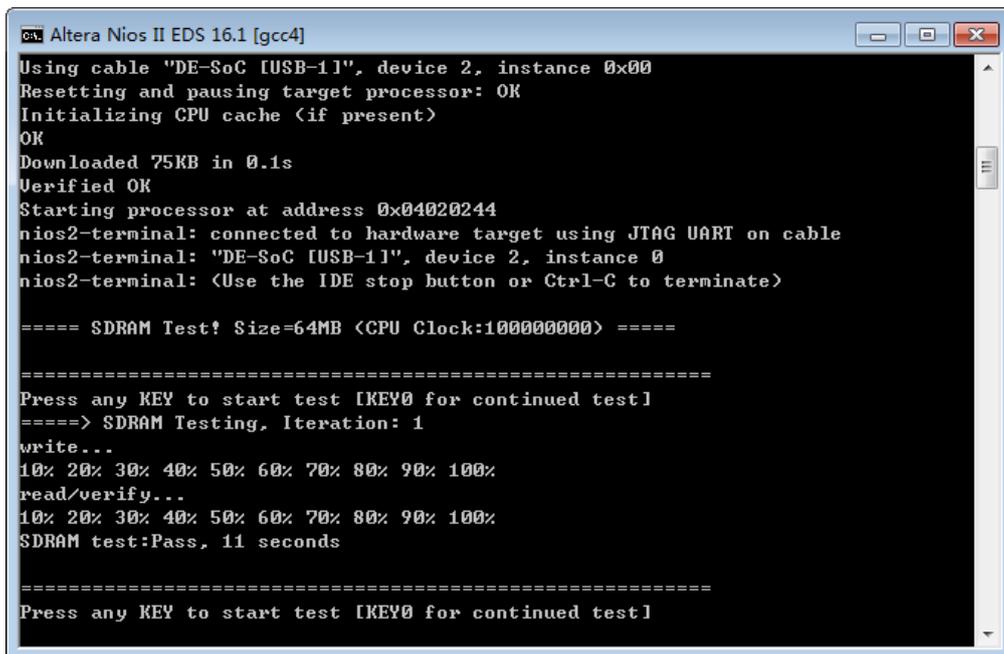
该文件位于\SDRAM\_Nios\_Test\demo\_batch 目录中。

该文件夹包含以下文件：

- USB-Blaster II批处理文件：test.bat
- FPGA配置文件：SDRAM\_Nios\_Test.sof
- Nios II程序：SDRAM\_Nios\_Test.elf

## ■ 演示示例

- 主机上须安装好Quartus Prime v16.1 和Nios II v16.1软件。
- 用USB线连接开发板的USB-Blaster II接口和PC的USB接口。
- 接通DE10-Standard开发板电源。
- 在SDRAM\_Nios\_Test\demo\_batch目录下执行示例批处理文件“test.bat”。
- 程序成功下载并执行后，Nios终端将显示一段提示信息。
- 按任意键(KEY3~KEY0)开始SDRAM验证过程。按 KEY0持续运行该测试。
- 如图 5-7所示，该程序会显示测试进度及结果。



```
Altera Nios II EDS 16.1 [gcc4]
Using cable "DE-SoC [USB-1]", device 2, instance 0x00
Resetting and pausing target processor: OK
Initializing CPU cache <if present>
OK
Downloaded 75KB in 0.1s
Verified OK
Starting processor at address 0x04020244
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "DE-SoC [USB-1]", device 2, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

==== SDRAM Test! Size=64MB <CPU Clock:100000000> ====

=====
Press any KEY to start test [KEY0 for continued test]
====> SDRAM Testing, Iteration: 1
write...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
read/verify...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
SDRAM test: Pass, 11 seconds

=====
Press any KEY to start test [KEY0 for continued test]
```

图 5-7 Nios II SDRAM 测试进度和结果展示

## 5.5 Verilog SDRAM 测试

DE10-Standard 系统 CD 提供了另外一种 SDRAM 测试示例，其测试代码用 Verilog HDL

编写。测试的 SDRAM 模块内存仍为 64MB。

## ■ 功能框图

图 5-8 所示为本示例的功能框图。SDRAM 控制器使用 50MHz 时钟作为参考时钟，并产生 100MHz 时钟作为存储器时钟。

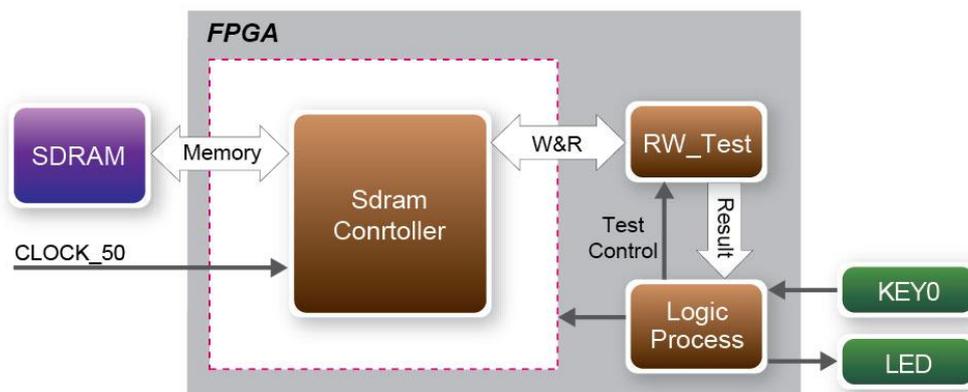


图 5-8 Verilog SDRAM 测试框图

RW\_test 模块首先会将一段测试序列写入整个内存，接着再读取整个内存数据并比较是否与写入的数据一致。根据表 5-3 所示，KEY0 触发 SDRAM 的测试控制信号，LED 会显示测试结果。

## ■ 设计工具

- Quartus Prime v16.1

## ■ 示例源代码

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_DRAM\_RTL\_Test
- 配置文件：DE10\_Standard\_DRAM\_RTL\_Test.sof

## ■ 示例批处理文件

示例批处理文件夹：\DE10\_Standard\_DRAM\_RTL\_Test\demo\_batch

该目录包含以下文件：

- 批处理文件：test.bat
- FPGA配置文件：DE10\_Standard\_DRAM\_RTL\_Test.sof

## ■ 演示示例

- 主机须安装Quartus Prime v16.1软件。
- 用USB线连接开发板的USB-Blaster II接口(J13)和PC的USB接口。
- 接通DE10-Standard开发板电源。
- 在 \DE10\_Standard\_SDRAM\_RTL\_Test\demo\_batch 目录中执行批处理文件 DE10\_Standard\_SDRAM\_RTL\_Test.bat。
- 按DE10-Standard开发板的KEY0开始验证进程。按下KEY0后，LEDR [2:0]常亮。再按一次KEY0，LEDR1和LEDR2开始闪烁。
- 大约8秒后，LEDR1停止闪烁并保持常亮，代表测试PASS，否则测试NG。表 5-3列出了LED指示灯的状态。
- 如果LEDR2没有闪烁，表示50MHz时钟源没有正常工作。
- 再按KEY0重复SDRAM测试。

表 5-3 LED 指示灯状态

名称	描述
LEDR0	复位
LEDR1	KEY0 松开后如果测试通过，会保持常亮
LEDR2	闪烁

## 5.6 电视盒示例

本示例通过使用DE10-Standard 开发板的VGA输出接口、音频编解码器和TV解码器播放来自DVD播放器的音频和视频，实现将DE10-Standard开发板转变成电视盒的效果。图 5-9所示为该设计的框图。该系统有两个主要模块，I2C\_AV\_Config和TV\_to\_VGA。TV\_to\_VGA模块由ITU-R 656 Decoder、SDRAM Frame Buffer、YUV422 转 YUV444、YCbCr 转 RGB和VGA Controller组成。

I2C\_AV\_Config 模块依靠 I2C 协议与 TV 解码器通讯，TV 解码器的寄存器值用于通过该模块来配置 TV 解码器。TV 解码器在上电过程中会持续一段时间的不稳定状态，Lock Detector 模块负责检测该不稳定性。

ITU-R 656 Decoder 模块从 TV 解码器传来的 ITU-R 656 数据流中提取 YcrCb 4:2:2 (YUV 4:2:2) 视频信号，并产生数据有效控制信号表明数据输出的有效期。由于 TV 解码器的视频信号是隔行扫描的，所以该数据源需要做去隔行处理。SDRAM Frame Buffer 和 VGA Controller 控制的 MUX 可以执行该去隔行处理动作。VGA Controller 还会产生数据请求、奇偶选择信号

到 SDRAM Frame Buffer 和 MUX。YUV422 to YUV444 模块将 YcrCb 4:2:2 (YUV 4:2:2) 视频数据转换为 YcrCb 4:4:4 (YUV 4:4:4)格式的视频数据。

最后，YcrCb\_to\_RGB 模块将 YcrCb 数据转换为 RGB 数据输出。VGA Controller 模块产生标准 VGA 同步信号 VGA\_HS 和 VGA\_VS，使能 VGA 显示器的显示信息。

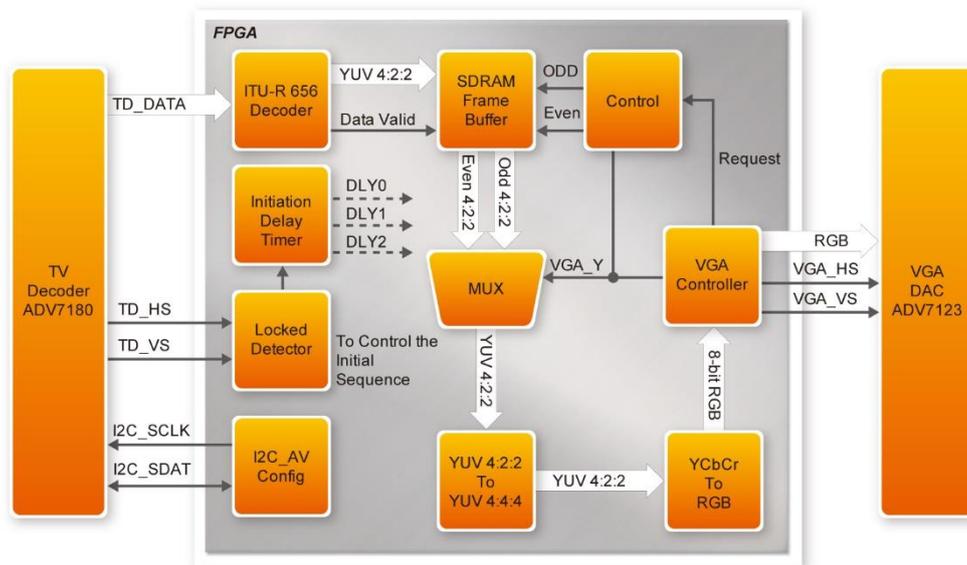


图 5-9 电视盒示例框图

## ■ 示例源代码

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_TV
- 配置文件：DE10\_Standard\_TV.sof

## ■ 示例批处理文件

文件目录：\DE10\_Standard\_TV\demo\_batch

文件夹包含以下文件：

- 批处理文件：DE10\_Standard\_TV.bat
- FPGA配置文件：DE10\_Standard\_TV.sof

## ■ 示例设置、文件位置和操作说明

• 如图5-10所示，将DVD播放器的复合视频输出端（黄色插头）连接到DE10\_Standard开发板的Video-in RCA插孔(J6)。DVD播放器必须配置成NTSC格式输出、60Hz刷新率、4:3显示器宽高比和非逐行扫描格式视频。

- 将VGA显示器连接到DE10\_Standard开发板的VGA输出接口。

- 将DVD播放器的音频输出连接到DE10\_Standard开发板的line-in接口，将扬声器连接到line-out接口。如果DVD播放器的音频输出插孔为RCA型，需要使用适配器将其转换DE10-Standard开发板支持的mini-stereo型插孔。
- 在 \Demonstration\FPGA\DE10\_Standard\_TV \demo\_batch\ 目录中下执行 DE10\_Standard\_TV.bat文件加载数据流到FPGA中。按开发板上的KEY0可以复位运行该示例。

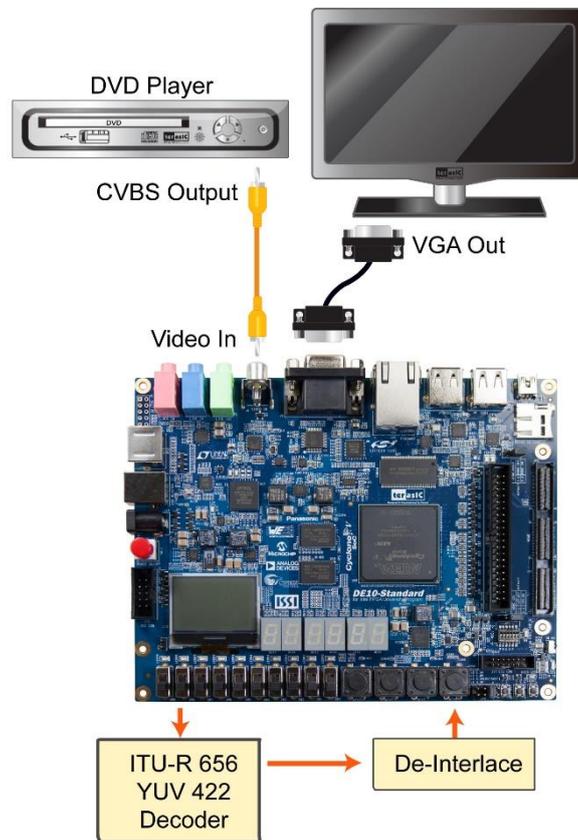


图 5-10 电视盒示例的运行设置

## 5.7 电视盒示例 (VIP)

本节将演示如何使用 Intel FPGA VIP (Video Image Processing) 将 DE10-Standard 开发板转变为电视盒, 用户通过使用 DE10-Standard 开发板的 VGA 输出接口、音频编解码器以及 TV 解码器播放来自 DVD 播放器的视频和音频。

图 5-11 所示为该设计的方框图。该系统有 I2C\_AV\_Config 和 Qsys/Vips 两个主要模块。Qsys/Vips 模块由很多 VIP IP 组成, 例如 Clocked Video Input II、Color Plane Sequencer II、Deinterlacer II、Clipper II、Frame Buffer II、Chroma Resampler II、Color Space Converter II、Scaler II 以及 Clocked Video Output 模块。从图 5-11 还可以看出该设计使用了 TV 解码器 (ADV7180) 和 VGA DAC (ADV7123) 芯片。

I2C\_AV\_Config 模块依靠 I2C 协议与 TV 解码器通讯, TV 解码器的寄存器值用来通过该模块配置 TV 解码器芯片。TV 解码器在上电过程中会持续一段时间的不稳定状态, Lock Detector 模块负责检测该不稳定性。

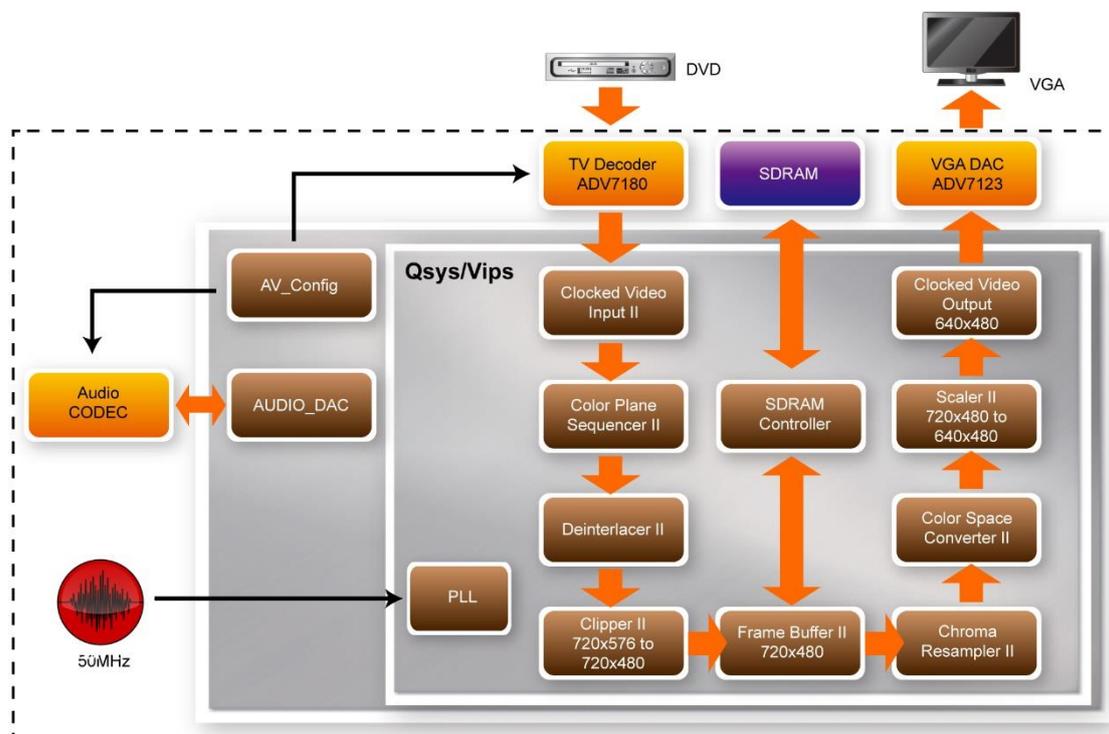


图 5-11 TV 盒(VIP)示例

Qsys/Vips 模块处理从 TV 解码器输入端到 VGA 输出端的全部流媒体视频。

**Clocked Video Input II:** 将 Clocked Video 转换为 Avalon-ST Video(720x576、隔行、Y'CbCr、4:2:2 格式)。

**Color Plane Sequencer II:** 将颜色平面从串行转换为并行传输。

**Deinterlacer II:** 将来自 TV-input (PAL/NTSC) 的隔行视频信号去隔行成逐行视频后用于 VGA 输出。

**Clipper II:** 将 720x576 视频分辨率裁剪为 720 x480。

**Frame Buffer II:** 将视频流缓冲存入 SDRAM。

**Chroma Resampler II:** 将 4:2:2 视频数据格式重新采样为 4:4:4 格式。

**Color Space Converter II:** 将 Y'CbCr 色域转换为 R'G'B'。

**Scaler II:** 缩放视频分辨率, 720x480 缩放为 640 x480。

**Clocked Video Output:** 将 Avalon-ST Video 协议数据转换为时钟视频。

## ■ 示例源代码

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_VIP\_TV
- 配置文件：DE10\_Standard\_VIP\_TV.sof

## ■ 批处理文件

文件目录：\Demonstration\FPGA\DE10\_Standard\_VIP\_TV\demo\_batch

文件夹包含以下文件：

- 批处理文件：DE10\_Standard\_VIP\_TV.bat
- FPGA配置文件：DE10\_Standard\_VIP\_TV.sof

## ■ 示例设置、文件位置及操作说明

• 如图5-12所示，将DVD播放器的复合视频输出端（黄色插头）连接到DE10-Standard开发板的Video-in RCA 插孔(J6)。DVD播放器必须配置成NTSC或PAL 格式输出、60Hz刷新率、4:3显示器宽高比和非逐行扫描格式视频。

- 将VGA显示器连接到DE10-Standard开发板的VGA输出接口。

• （可选项）将DVD播放器的音频输出连接到DE10-Standard开发板的line-in接口，将扬声器连接到line-out接口。如果DVD播放器的音频输出插孔为RCA型，需要用适配器将其转换DE10-Standard开发板所支持的mini-stereo型插孔。

• 在目录\DE10\_Standard\_VIP\_TV \demo\_batch\中执行DE10\_Standard\_VIP\_TV.bat文件，加载数据流到FPGA中。

- 按开发板上的KEY0可以复位运行该示例。
- VGA显示器上会播放视频。

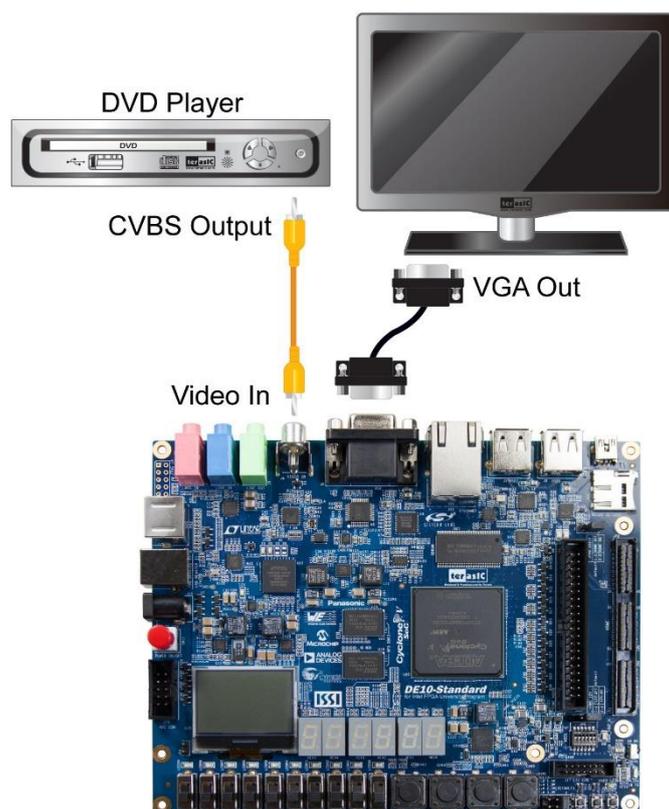


图 5-12 TV 盒 (VIP)示例设置

## 5.8 PS/2 鼠标示例

基于 Verilog HDL 编写的简易 PS/2 控制器用于演示其与 PS/2 鼠标之间的双向通讯。基于此可以开发综合性 PS/2 控制器来实现更多复杂的功能，例如设置采样率或分辨率，该情况下需要一次传送 2 个字节数据。

更多关于 PS/2 协议的相关信息可以在各类网站上查找。

### ■ 简介

PS/2 协议使用时钟线和数据线进行双向通讯。PS/2 控制器完全控制传输线，但是在数据传输过程中由 PS/2 设备产生时钟信号。

### ■ 从设备到控制器的数据传输

PS/2 鼠标在流模式 (stream mode) 下接收到使能信号后开始发送 33 位位移数据。帧数据分为三部分，每部分包含一个起始位 (通常为 0)、8 个数据位 (最低位为 LSB)、一个奇偶检验位 (奇校验) 和一个停止位 (通常为 1)。

PS/2 控制器在 PS/2 时钟信号的下降沿对数据线(data line)进行采样，这可以用一个 33 位

移位寄存器来实现，但是要注意时钟域交叉问题。

## ■ 从控制器到设备的数据传输

当 PS/2 控制器向设备发送数据时，首先将时钟线(clock line)拉低一个以上时钟周期时间，用来禁止当前发送过程或指示一个新发送过程的开始，这种情况称之为禁止状态。随后在释放时钟线之前拉低数据线，这个过程称之为请求状态。由释放动作形成的时钟线上升沿可用来表示采样时间点的“start bit”。设备检测到这一连串动作并在 10ms 内产生一段时钟序列。传输数据共 12 位，包含 1 个起始位，8 个数据位，1 个奇偶校验位（奇校验），1 个停止位（通常为 1），1 个确认位（通常为 0）。发送奇偶校验位后，控制器会释放数据线，设备在下一时钟周期内将检测数据线的状态变化。如果一个时钟周期内数据线状态没有变化，设备会再次拉低数据线确认数据已正确接收。

PS/2 鼠标复位后自动进入流模式工作并禁止数据传输，除非接收到使能指令。图 5-13 所示为数据线和时钟线通讯时的波形。

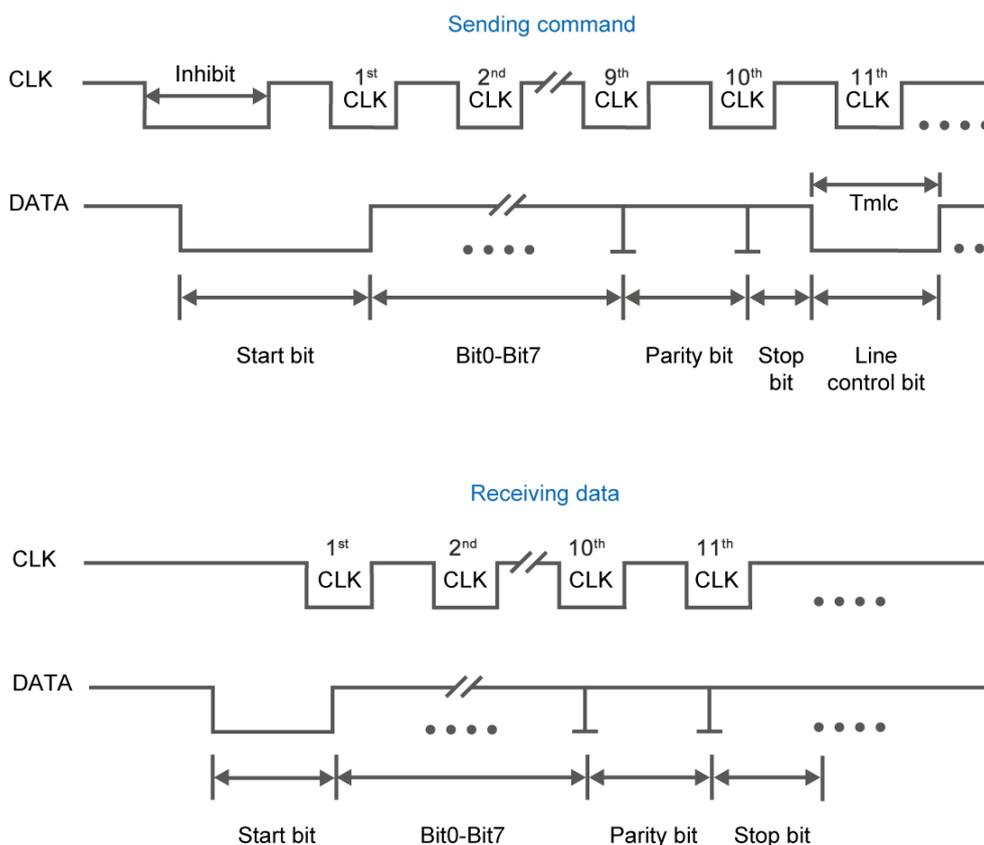


图 5-13 数据传输时时钟信号与数据信号的波形

## ■ 示例源代码

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_PS2
- 配置文件：DE10\_Standard\_PS2.sof

## ■ 批处理文件

文件目录：\DE10\_Standard\_PS2\demo\_batch

文件夹包含以下文件：

- 批处理文件：test.bat
- FPGA配置文件：DE10\_Standard\_PS2.sof

## ■ 示例设置、文件位置及操作说明

- 执行\DE10\_Standard\_PS2\demo\_batch\test.bat，加载数据流到FPGA。
- 将PS/2鼠标插入开发板的PS/2接口。
- 按下KEY[0]，启动数据传输。
- 按下KEY[1]，清除数据缓存。
- 七段数码管显示的数字随PS/2鼠标的移动而变化。根据表5-4，按鼠标的左、中、右按键时LEDR[2:0]会闪烁。

表 5-4 七段数码管描述及 LED 指示灯

指示灯名称	描述
LEDR[0]	指示鼠标左键按下
LEDR[1]	指示鼠标右键按下
LEDR[2]	指示鼠标中间按键按下
HEX0	显示水平位移的低字节
HEX1	显示水平位移的高字节
HEX2	显示垂直位移的低字节
HEX3	显示垂直位移的高字节

## 5.9 红外发射器 LED 和接收器示例

DE10-Standard 系统 CD 中提供了一个使用红外发射器 LED 和接收器的示例,该示例代码是用 Verilog HDL 编写设计的。

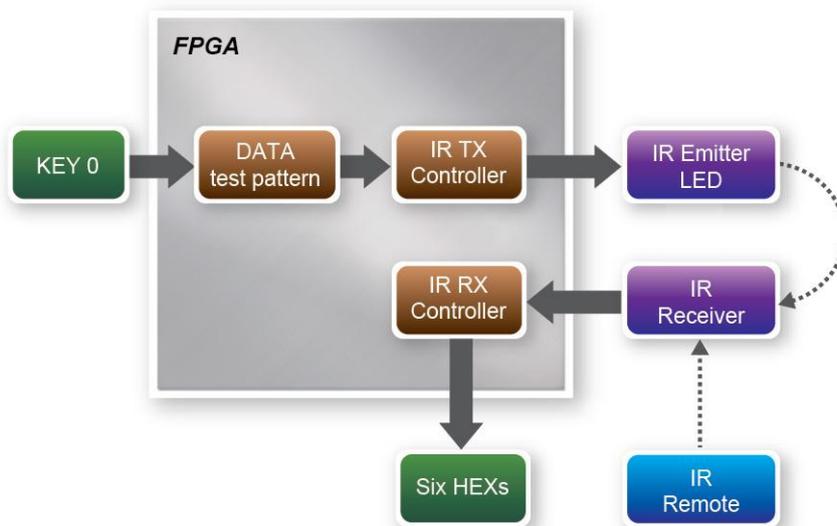


图 5-14 LED 红外发射器和接收器示例框图

图 5-14 所示为本示例的方框图。该示例实现了 IR TX Controller 和 IR RX Controller。按下 KEY0 后，数据测试模式发生器(Data test pattern)将向 IR TX Controller 持续产生数据。IR TX Controller 工作时，它会将数据格式化以便兼容 NEC IR 传输协议，并通过 IR 发射管发送数据。IR 接收器将对接收到的数据进行解码并显示在 6 个七段数码管上。用户也可以用遥控器向 IR 接收器发送数据。以下部分将介绍该示例中的 IR TX /RX controller 和 IR 遥控器的主要功能。

## ■ IR TX Controller

用户可以输入 8 位地址和 8 位指令到 IR TX Controller。IR TX Controller 会先对地址和指令进行编码，然后再根据 NEC IR 传输协议通过 IR 发射管将其发出。IR TX Controller 的输入时钟为 50MHz。

NEC IR 传输协议使用脉冲对信息位进行编码，每个脉冲由 562.5 $\mu$ s 长度的 38KHz 载波构成。

图 5-15 所示为逻辑“1”和“0”的持续时间，逻辑位传输如下：

- 逻辑“0”–562.5 $\mu$ s 脉冲，接着是 562.5 $\mu$ s 空闲，总传输时间 1.125ms。
- 逻辑“1”–562.5 $\mu$ s 脉冲，接着是 1.6875ms 空闲，总传输时间 2.25ms。

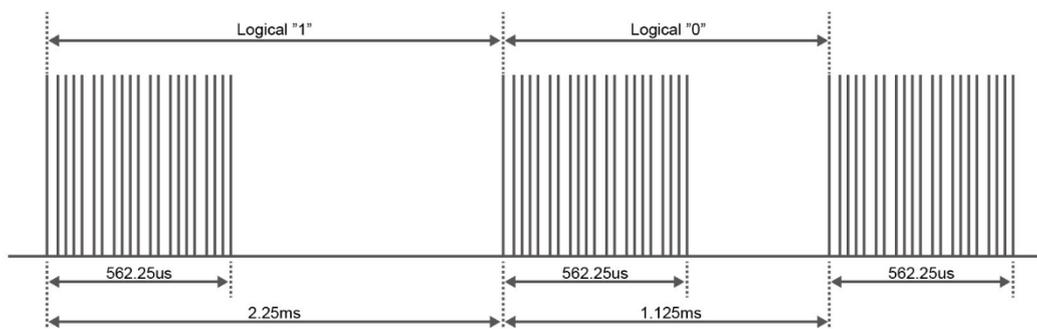


图 5-15 逻辑“1”和“0”的持续时间

图 5-16 所示为 NEC 协议的某一帧的格式。协议先发送一个 9ms 脉冲码和一个 4.5ms 空闲作为引导码。第二次发送的反码用来校验接收到的信息的准确性。最后发送 562.5μs 脉冲表示信息传输结束。根据该协议，数据是源码和反码成对发送，因此总的传输时间是固定的。

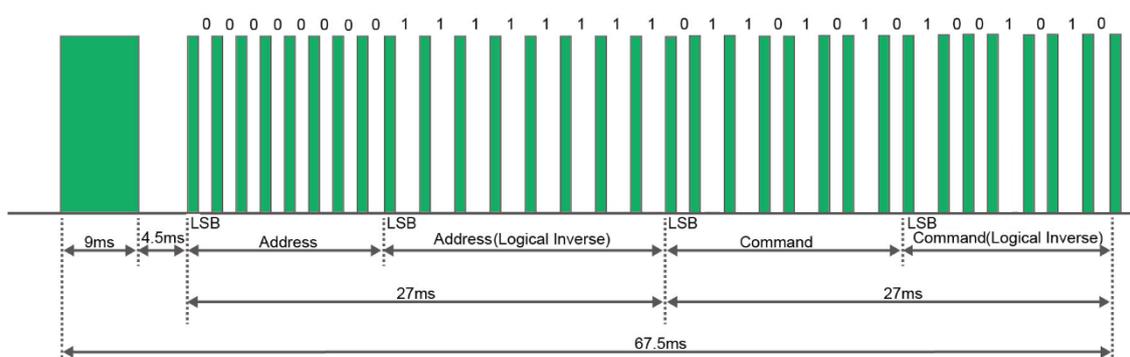


图 5-16 典型的 NEC 协议帧

**注意：** IR 接收器接收到的信号是反码。例如，如果 IR TX Controller 发送 9ms 脉冲，接着是 4.5ms 空闲，那么 IR Receiver 将接收到 9 ms 空闲，接着是 4.5ms 脉冲。

## ■ IR Remote

按如图 5-17 所示遥控器上的按键后，遥控器会发出如表 5-5 所示的标准帧。帧的起始是代表起始位的前置码，接着是相关按键的信息。最后一位结束码表示帧结束。接收端的帧值与原帧完全相反。



图 5-17 本示例所用的遥控器

表 5-5 遥控器按键代码信息

Key	Key Code						
	0x0F		0x13		0x10		0x12
	0x01		0x02		0x03		0x1A
	0x04		0x05		0x06		0x1E
	0x07		0x08		0x09		0x1B
	0x11		0x00		0x17		0x1F
	0x16		0x14		0x18		0x0C

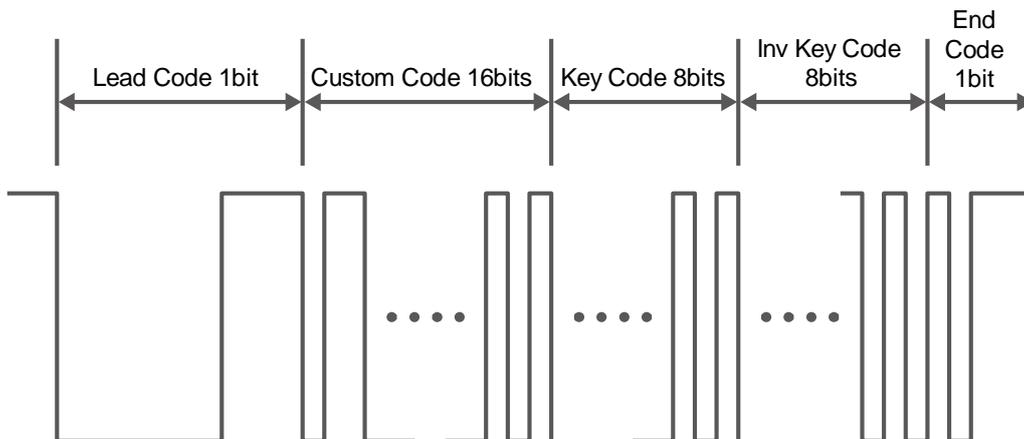


图 5-18 IR 遥控器发射波形

## ■ IR RX Controller

以下部分将展示如何在 FPGA 中实现 IR 接收器的 IP 功能。图 5-19 所示为该示例用到的模块，包括 Code Detector、State Machine 和 Shift Register。IR 接收器首先将信号解调输入到 Code Detector，Code Detector 检测 Lead Code 并反馈结果到 State Machine。

检测到 Lead Code 之后，State Machine 模块的状态从 IDLE 转换成 GUIDANCE。如果 Code Detector 检测到 Custom Code 的状态，当前状态将从 GUIDANCE 转换为 DATAREAD。Code Detector 还会保存接收到的数据并输出到移位寄存器并在七段数码管上显示。图 5-20 所示为 State Machine 模块的状态交替框图。输入时钟为 50MHz。

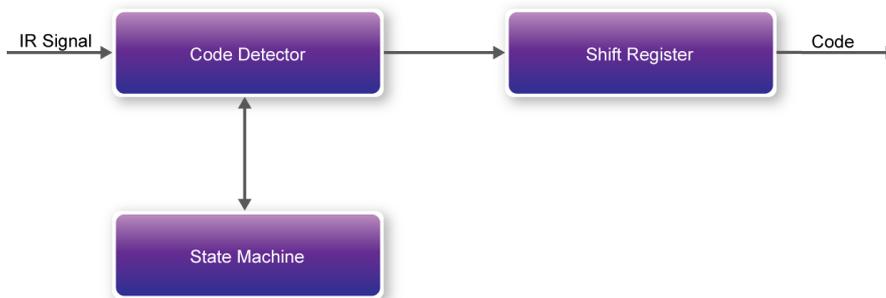


图 5-19 IR Receiver Controller 相关模块

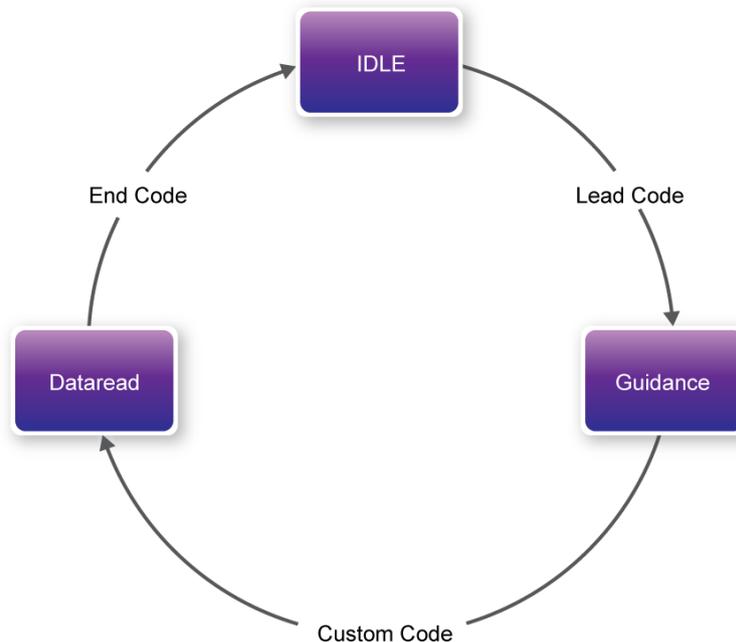


图 5-20 State Machine 模块的状态交替框图

## ■ 示例源代码

- 工程目录：\Demonstration\FPGA\DE10\_Standard\_IR
- 配置文件：DE10\_Standard\_IR.sof

## ■ 批处理文件

文件目录：\Demonstration\FPGA\DE10\_Standard\_IR\demo\_batch

文件夹包含以下文件：

- 批处理文件：test.bat
- FPGA配置文件：DE10\_Standard\_IR.sof

## ■ 示例设置、文档位置及操作说明

- 执行DE10\_Standard\_IR\demo\_batch\ test.bat，将数据流加载到FPGA。
- 持续按KEY[0]，使能IR TX Controller持续发送模式。
- 根据表5-6观察六个HEX的状态。
- 松开KEY[0]，使IR TX Controller停止工作。
- 遥控器指向IR接收器，并按遥控器上的任意按钮。
- 根据表5-6观察六个HEX状态。

表 5-6 指示器信息描述

指示器名称	描述
HEX5	数据（按键码）高字节的反码
HEX4	数据（按键码）低字节的反码
HEX3	地址（用户码）的高字节
HEX2	地址（用户码）的低字节
HEX1	数据（按键码）的高字节
HEX0	数据（按键码）的低字节

## 5.10 ADC 读取

本示例阐述了如何评估8通道、12位LTC2308 A/D转换器的性能。2x5引脚连接头的5.0V直流电源通过微调电位器产生电压在0~4.096V范围内可调的模拟信号。NIOS II 控制台显示经过12位ADC采样后的电压值。图5-21所示为本示例的方框图。

ADC 默认电压区间为 0~4.096V。

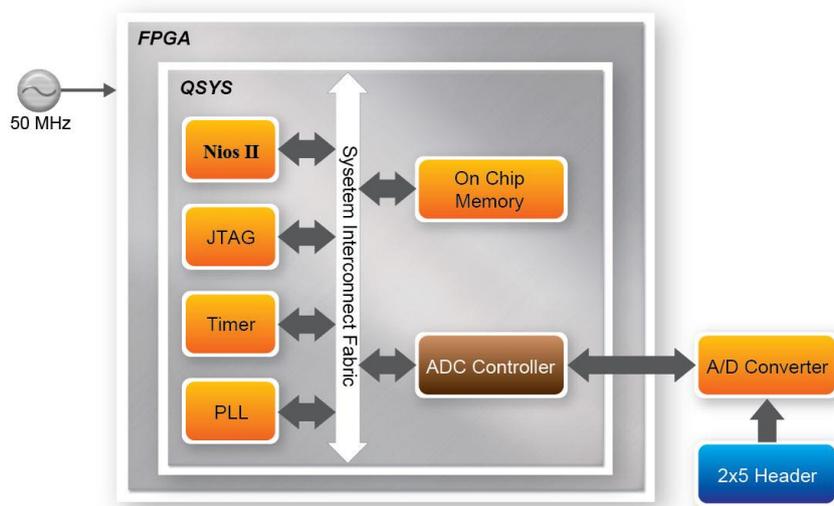


图 5-21 ADC 读取示例系统框图

图5-22所示为2x5引脚连接头的引脚分配。这个连接头是本示例中A/D转换器的输入源。用户可以将微调器连接到指定的ADC通道（ADC\_IN0~ADC\_IN7），为A/D转换器提供电压。FPGA通过串行接口读取A/D转换器中相关的寄存器数据，并将其转化为电压值在Nios II终端上显示。

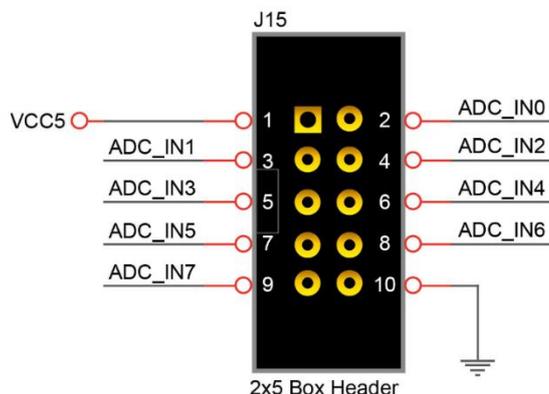


图 5-22 2x5 引脚连接头引脚分配

LTC2308 是一款低噪声、500ksps 采样率、8 通道、带有 SPI/MICROWIRE 兼容性串行接口的 12 位 A/D 转换器。内部转换时钟允许外部串行输出数据时钟（SCK）以最高可达 40MHz 的任意频率工作。本示例中，我们基于 Verilog 实现了 SPI 协议，并将其封装到 Avalon MM slave IP，从而连接到 Qsys。

图 5-23 是 LTC2308 的 SPI 时序规范。

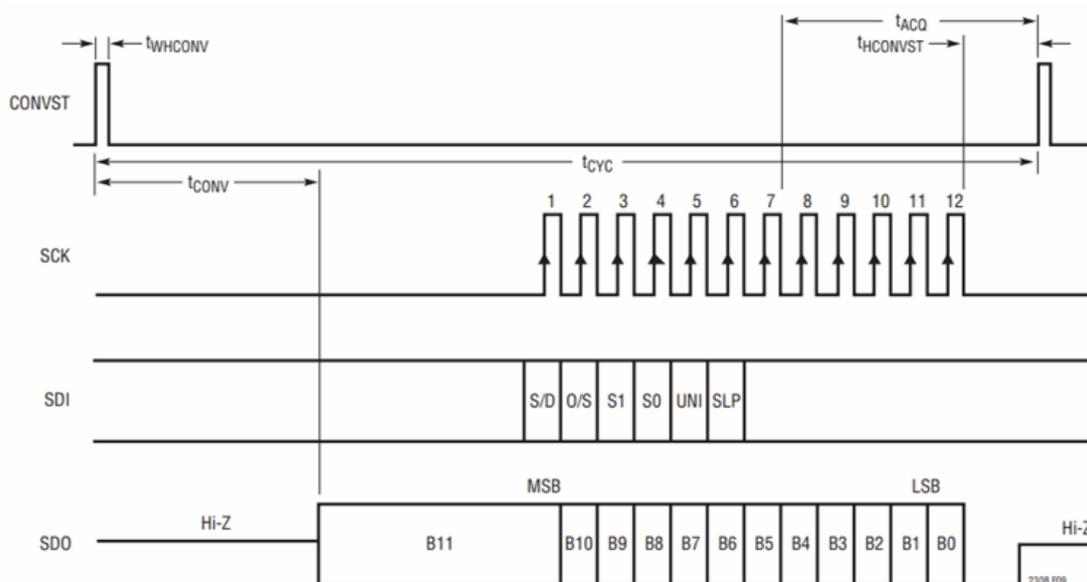


图 5-23 具有短 CONVST 脉冲的 LTC2308 时序规范

**重要：**用户要重点关注输入源与 ADC 电路的阻抗匹配。如果驱动电路电源阻抗低，ADC 输入会被直接驱动。否则，高阻抗电源应允许有更多采集时间。

若要更改  $t_{ACQ}$  采样时间，用户可以在 `adc_ltc2308.v` 中修改 `tHCONVST` 宏定义的值 (macro value)。SCK 设置为 40MHz，即每单元 25ns。tHCONVST 默认设置为 320，达到 100MHz 样本。因此，增加更多 tHCONVST 时间（通过增加 tHCONVST 宏定义的值）将降低 ADC 采样率。

```
`define tHCONVST      320
```

图 5-24 举例显示了 ADC MUX 配置。该示例中用 Verilog 代码将其配置为 8 通道。用户可以更改 `SW[2:0]` 来测量相应的通道。默认参考电压为 4.096V。

计算采样电压公式：采样电压 = ADC 数据 / ADC 满刻度数据 \* 参考电压

本示例中，满刻度数据是  $2^{12} = 4096$ ，参考电压是 4.096V。所以采样电压 = ADC 数据 /  $4096 * 4.096 = \text{ADC 数据} / 1000$

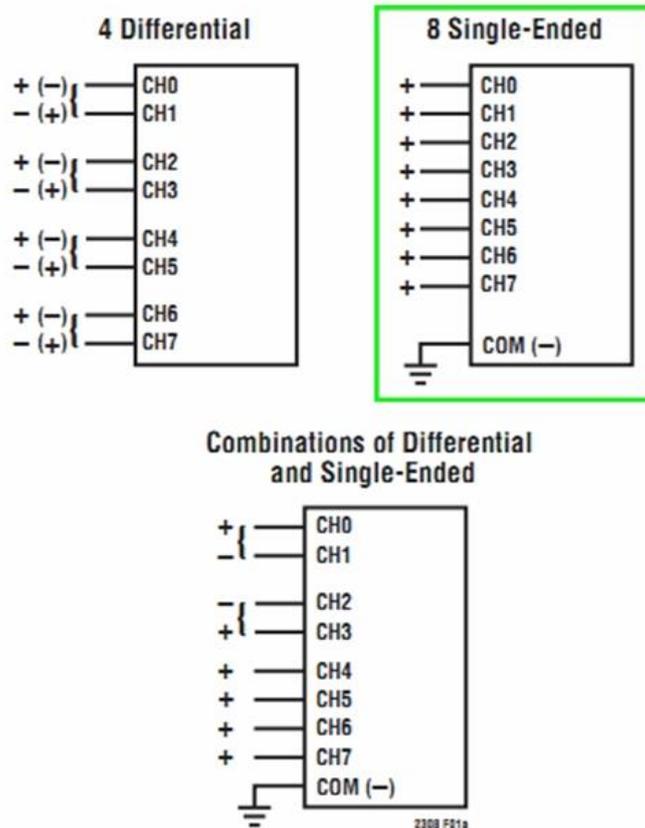


图 5-24 MUX 配置

## ■ 系统要求

运行此示例需要准备以下设备：

- DE10-Standard开发板 x1
- 微调电位器x1
- 导线x3

## ■ 文件位置

- 硬件工程目录：\Demonstration\FPGA\DE10\_Standard\_ADC
- 配置文件：DE10\_Standard\_ADC.sof
- 软件工程目录：DE10\_Standard\_ADC\software
- 批处理文件：DE10\_Standard\_ADC\demo\_batch\DE10\_Standard\_ADC.bat

## ■ 示例设置及操作说明

- 如图5-25所示，将微调器连接到2x5引脚连接头上对应的ADC通道。同时连接好+5V和GND信号。图中所示的微调器连接到ADC的通道0。

- 执行批处理文件DE10\_Standard \_ADC.bat，将比特流和可执行的软件文件加载到FPGA。
- Nios II控制台将显示指定通道电压信息。
- 如果要测量其他通道的电压，可以通过设置SW[2:0]到对应的通道，并提供任意输入电压到该ADC通道。

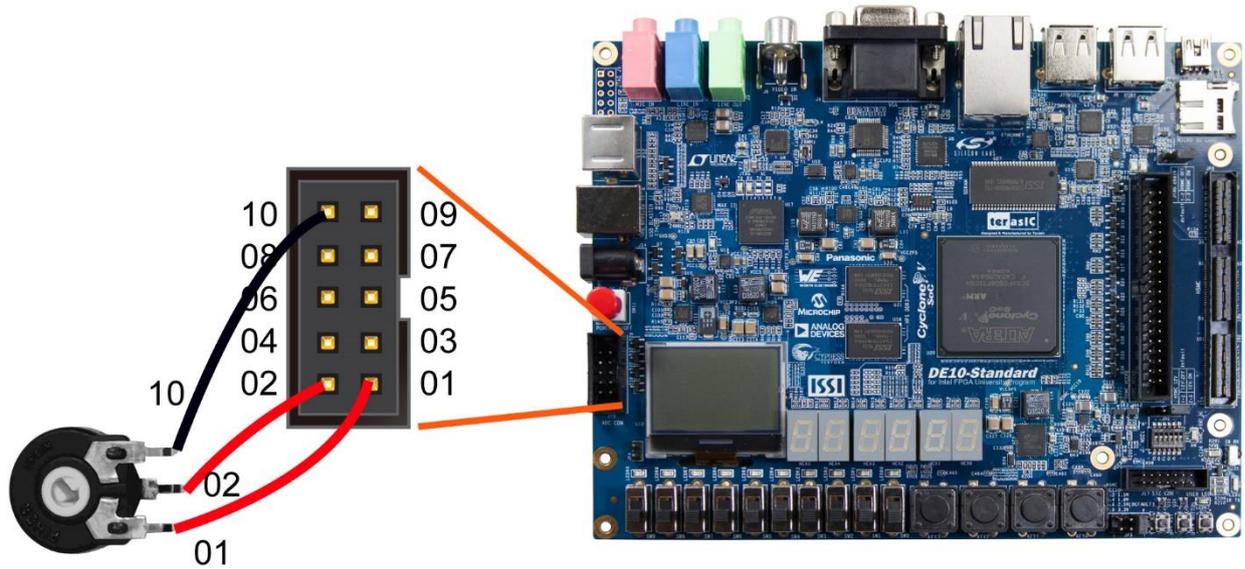


图 5-25 ADC 读取示例的硬件连接设置

# 第六章

## HPS SoC 设计示例

本章提供了一些基于Yocto工程创建的Intel SoC Linux的C代码示例，这些示例演示了DE10-Standard开发板HPS端的主要外设的功能，例如LED/KEY、I2C接口G-sensor和I2C MUX等。示例工程源码可以在DE10-Standard系统CD的Demonstration/SoC目录中找到。请参考 *DE10-Standard\_Getting\_Started\_Guide.pdf* 文档的第五章 **Running Linux on the DE10-Standard board** 在DE10-Standard开发板上运行Linux。

### ■ 示例安装

在主机上安装设计示例：

将DE10-Standard系统CD的整个Demonstration文件夹复制到您所选的PC本地目录中。用户需要用到Intel SoC FPGA EDS v16.1软件编译C代码工程。

### 6.1 Hello World 程序

本示例演示如何用Intel SoC FPGA EDS软件开发第一个HPS程序。详细信息请参考系统CD中的 *My\_First\_HPS.pdf* 文档。

开发并编译HPS工程的主要步骤如下：

- 在主机上安装Intel SoC FPGA EDS软件。
- 使用通用文本编辑器创建.c或.h文件。
- 使用通用文本编辑器创建Makefile。
- 使用Intel SoC FPGA EDS软件编译工程。

### ■ 程序文件

Hello World 示例的主程序如下所示：

```
#include <stdio.h>

int main(int argc, char **argv) {

    printf("Hello World!\r\n");

    return( 0 );

}
```

## ■ Makefile

Makefile 用于编译工程。本示例用到的 Makefile 代码如下所示：

```
#
TARGET = my_first_hps

ALT_DEVICE_FAMILY ?= soc_cv_av
SOCEDS_ROOT ?= $(SOCEDS_DEST_ROOT)
HWLIBS_ROOT = $(SOCEDS_ROOT)/ip/altera/hps/altera_hps/hwlib
CROSS_COMPILE = arm-linux-gnueabihf-
CFLAGS = -g -Wall -D$(ALT_DEVICE_FAMILY) -I$(HWLIBS_ROOT)/include/$(ALT_DEVICE_FAMILY) -I$(HWLIBS_ROOT)/include/
LDFLAGS = -g -Wall
CC = $(CROSS_COMPILE)gcc
ARCH= arm

build: $(TARGET)

$(TARGET): main.o
    $(CC) $(LDFLAGS)  $^ -o $@

%.o : %.c
    $(CC) $(CFLAGS) -c $< -o $@

.PHONY: clean
clean:
    rm -f $(TARGET) *.a *.o *~
```

## ■ 编译

运行Quartus软件安装目录如C:\intelFPGA\16.1\embedded中的Embedded\_Command\_Shell.bat, 启动SoC EDS Command Shell工具编译工程。

用 cd 命令将当前目录切换到 Hello World 工程所在的目录。用 make 命令编译工程, 编译成功后会生成可执行文件 my\_first\_hps, 用 clean all 命令清除所有临时文件。

## ■ 源代码

- 开发工具：SoC FPGA EDS v16.1软件
- 工程目录：\Demonstration\SoC\my\_first\_hps
- 二进制文件：my\_first\_hps
- 编译命令：make ("make clean"用于清除所有临时文件)
- 执行命令：./my\_first\_hps

## ■ 运行示例

- 用USB线连接开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 在Linux系统PC主机中, 将my\_first\_hps文件复制到microSD卡（烧录了DE10-Standard Linux镜像文件）的/home/root文件夹中。

- 将microSD卡插入DE10-Standard开发板的SD卡槽。
- 将SW10 MSEL[4:0]设置为01010。
- 接通DE10-Standard开发板电源。
- 启动PuTTY终端，与UART串口建立连接。输入用户名root登陆Yocto Linux系统。
- 在PuTTY终端输入“./my\_first\_hps”命令运行示例程序，终端将会打印输出运行结果“Hello World!”。

```
root@socfpga:~# ./my_first_hps
Hello World!
root@socfpga:~#
```

## 6.2 用户 LED 与 KEY

本示例演示如何通过内存映射(memory-mapped)设备驱动访问GPIO控制器的寄存器来控制HPS端的用户LED和KEY。内存映射设备驱动允许应用程序访问系统物理内存(system physical memory)。

### ■ 功能框图

图 6-1所示为该示例的功能框图。用户LED和KEY都连接到HPS的GPIO1控制器。GPIO控制器的行为通过其寄存器来控制。应用程序通过Intel SoC Linux内置的内存映射设备驱动访问寄存器。

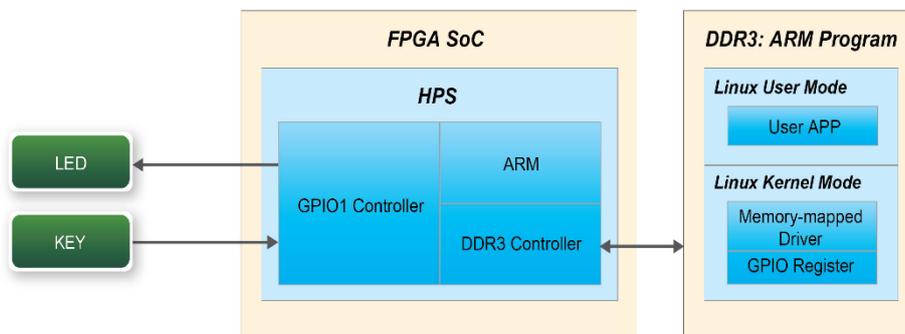


图 6-1 hps\_gpio 示例方框图

### ■ GPIO 接口功能框图

HPS提供了三个通用I/O（GPIO）接口模块。图 6-2所示为GPIO接口的功能框图。GPIO [28..0]由GPIO0控制器控制，GPIO [57..29]由GPIO1控制器控制。GPIO [70..58]和只输入型GPIO[13..0]由GPIO2控制器控制。

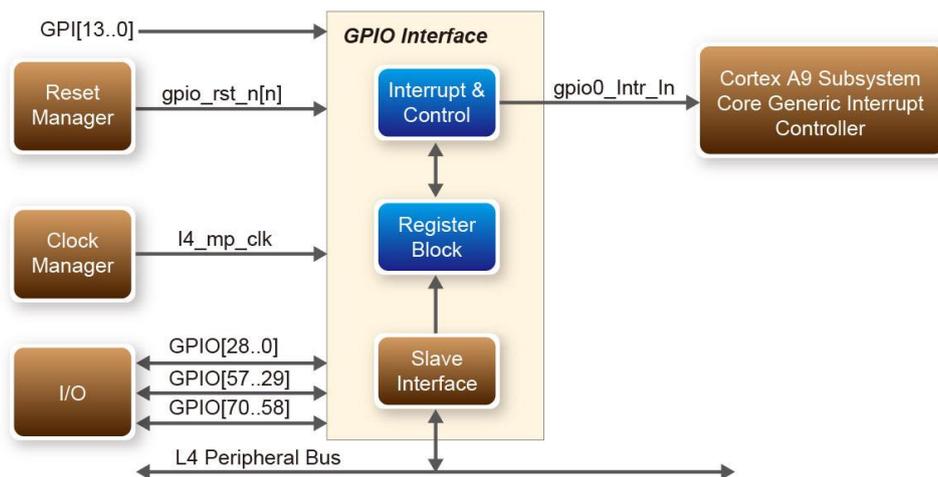


图 6-2 GPIO 接口方框图

## ■ GPIO 寄存器模块

I/O引脚行为由寄存器块中的寄存器控制。本示例用到了GPIO控制器的三个32位寄存器，如下所示：

- `gpio_swporta_dr`: 将输出数据写入I/O输出引脚
- `gpio_swporta_dds`: 配置I/O引脚方向
- `gpio_ext_porta`: 从输入引脚读取数据

`gpio_swporta_dds`寄存器将LED引脚配置为输出引脚，并将数据写入`gpio_swporta_dr`寄存器来驱动引脚为高电平或低电平。`gpio_swporta_dr`寄存器的第一位（最低有效位）控制相应GPIO控制器第一个I/O引脚的方向，第二位控制第二个I/O引脚的方向，并以此类推。寄存器位的值如果是1，I/O引脚方向为输出；如果是0，则I/O引脚方向为输入。

`gpio_swporta_dr`寄存器的第一位控制相关GPIO控制器的第一个I/O引脚的输出值，第二位控制第二个I/O引脚的输出值，并以此类推。寄存器位的值如果是1，则输出值为高电平；如果是0，则为低电平。

读取`gpio_ext_porta`寄存器的值可以查询KEY的状态。第一位代表相关GPIO控制器第一个I/O引脚的输入状态，第二位代表第二个I/O引脚的输入状态，并以此类推。寄存器位的值如果是1，则输入为高电平；如果是0，则为低电平。

## ■ GPIO 寄存器地址映射

HPS外设的寄存器映射到HPS基地址0xFC000000，共64MB寻址空间。如图 6-3所示，



```
#include <sys/mman.h>
#include "hwlib.h"
#include "socal/socal.h"
#include "socal/hps.h"
#include "socal/alt_gpio.h"
```

## ■ 控制 LED 与 KEY

图 6-4所示为DE10-Standard开发板HPS端的用户LED和KEY的引脚分配。LED与HPS\_GPIO53相连，KEY与HPS\_GPIO54相连。LED和KEY都由GPIO1控制，GPIO1也控制HPS\_GPIO29~HPS\_GPIO57。

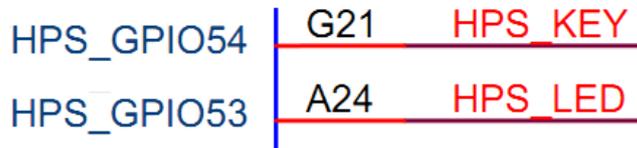


图 6-4 HPS 端 LED 与 KEY 的引脚分配

图 6-5所示为GPIO1控制器的gpio\_swporta\_ddr寄存器。Bit-0控制HPS\_GPIO29引脚的方向，Bit-24控制与HPS\_LED相连的HPS\_GPIO53引脚的方向，Bit-25控制与HPS\_KEY相连的HPS\_GPIO54引脚的方向，并以此类推。HPS\_LED和HPS\_KEY的引脚方向分别由GPIO1控制器的gpio\_swporta\_ddr寄存器中的 Bit-24与 Bit-25控制。与之类似，HPS\_LED的输出状态由GPIO1控制器的gpio\_swporta\_dr寄存器中的Bit-24控制。KEY的状态可以通过读取gpio\_ext\_porta寄存器中的Bit-24值来查询，该寄存器位于GPIO1控制器中。

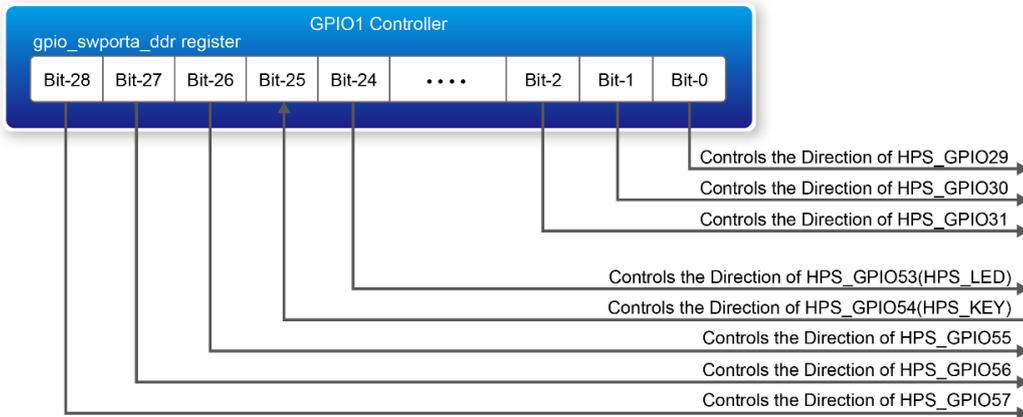


图 6-5 GPIO1 控制器的 gpio\_swporta\_ddr 寄存器

本示例代码定义了以下掩码(Mask)，用于控制LED与KEY的方向及LED输出值：

```
#define USER_IO_DIR      (0x01000000)

#define BIT_LED          (0x01000000)

#define BUTTON_MASK     (0x02000000)
```

以下代码语句将LED引脚配置为输出引脚：

```
alt_setbits_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_SWPORTA_DDR_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ), USER_IO_DIR );
```

以下代码语句可以点亮LED：

```
alt_setbits_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_SWPORTA_DR_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ), BIT_LED );
```

以下代码语句可以读取gpio\_ext\_porta寄存器的内容。位掩码用于检查KEY的状态：

```
alt_read_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_EXT_PORTA_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ) );
```

## ■ 源代码

- 开发工具：SoC FPGA EDS V16.1软件
- 工程目录：\Demonstration\SoC\hps\_gpio
- 二进制文件：hps\_gpio
- 编译命令：make ("make clean"用于清除所有临时文件)
- 执行命令：./hps\_gpio

## ■ 运行示例

- 用USB线连接开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 在Linux系统PC主机中，将hps\_gpio文件复制到microSD卡（烧录了DE10-Standard Linux镜像文件）的/home/root文件夹中。
  - 将micro SD卡插入DE10-Standard开发板的SD卡槽。
  - 将开发板的SW10 MSEL[4:0]设置为01010。
  - 接通DE10-Standard开发板电源。
  - 启动PuTTY终端，与UART串口建立连接。输入root登陆Intel Yocto Linux系统。

- 在PuTTY终端输入“./hps\_gpio”命令运行程序。

```

root@socfpga:~# ./hps_gpio
led test
the led flash 2 times
user key test
press key to control led

```

- HPS端的用户LEDG7会闪烁两次，用按钮开关控制用户LEDG7。
- 按开发板上HPS端的USER BUTTON按钮开关可以点亮LEDG7。
- 按主机键盘上的CTRL + C终止程序运行。

### 6.3 I2C 接口 G-sensor

本示例演示如何通过Intel SoC Yocto Powered Embedded Linux内置的I2C内核驱动访问寄存器的方式来控制G-sensor。

#### ■ 功能框图

图 6-6所示为本示例的功能框图。DE10-Standard开发板的G-sensor与HPS的I2C0控制器相连。G-Sensor I2C 7位设备地址为0x53。系统I2C总线驱动用于访问G-sensor中的寄存器文件。G-sensor中断信号与PIO控制器连接。本示例用轮询(polling)方法读取寄存器数据。

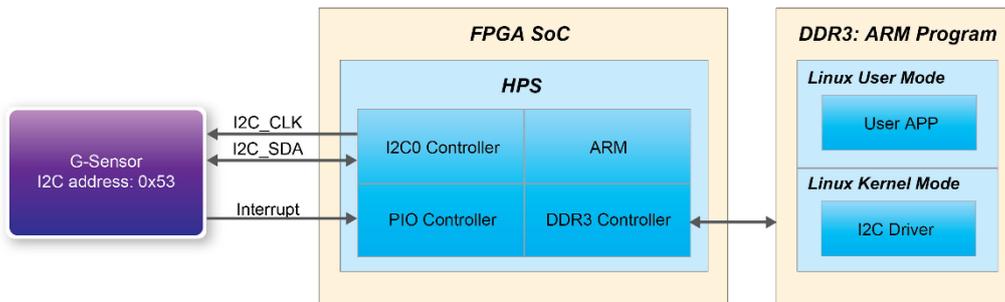


图 6-6 G-sensor 示例框图

#### ■ I2C 驱动

通过系统现有的I2C总线驱动从G-sensor寄存器文件读取寄存器值的过程如下：

1. 打开 I2C 总线驱动“/dev/i2c-0”：file = open(“/dev/i2c-0”, O\_RDWR)。
2. 指定 G-sensor I2C 地址 0x53：ioctl(file, I2C\_SLAVE, 0x53)。
3. 指定 G-sensor 所需的寄存器索引：write(file, &Addr8, sizeof(unsigned char))。
4. 读取 1 字节寄存器值：read(file, &Data8, sizeof(unsigned char))。

如图6-7所示，G-sensor I2C总线与I2C0控制器相连。驱动命名为dev/i2c-0。

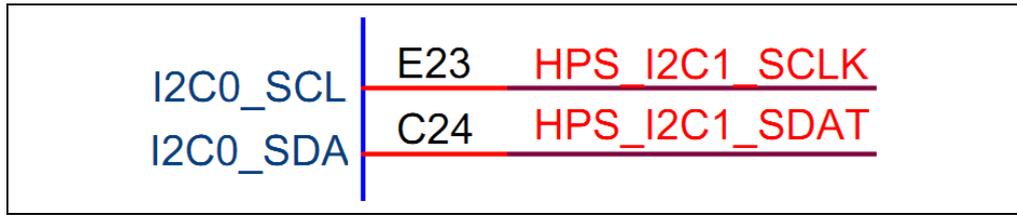


图 6-7 HPS I2C 信号连接示意图

以上的步骤4可以改为如下所示的命令，将某一个值写入寄存器：

```
write(file, &Data8, sizeof(unsigned char));
```

也可以改为以下命令，读取多个字节值：

```
read(file, &szData8, sizeof(szData8)); // where szData is an array of bytes
```

还可以改为以下命令，写入多个字节值：

```
write(file, &szData8, sizeof(szData8)); // where szData is an array of bytes
```

## ■ 控制 G-sensor

ADI ADXL345芯片提供了I2C和SPI接口。将DE10-Standard开发板上的ADXL345芯片的CS引脚设置为高电平来选择I2C接口。

ADI ADXL345 G-Sensor提供了用户可选的分辨率，最高分辨率可达13位，测量范围达 $\pm 16g$ 。通过DATA\_FORMAT(0x31)寄存器可以配置分辨率。本示例中的数据配置如下：

- 全分辨率模式
- $\pm 16g$ 可测量模式
- 左对齐模式

X/Y/Z的值可以从DATA\_X0(0x32)、DATA\_X1(0x33)、DATA\_Y0(0x34)、DATA\_Y1(0x35)、DATA\_Z0(0x36)和DATA\_Z1(0x37)寄存器获取。DATA\_X0表示最低有效字节，DATA\_X1表示最高有效字节。**建议对所有寄存器执行多字节读取，以防止按顺序读取时寄存器的数据发生变化。**

以下语句可以读取X、Y或Z值的6个字节。

```
read(file, szData8, sizeof(szData8)); // where szData is an array of six-bytes
```

## ■ 源代码

- 开发工具：SoC FPGA EDS v16.1软件
- 工程目录：\Demonstration\SoC\hps\_gsenser
- 二进制文件：gsensor

- 编译命令：make ("make clean"用于清除所有临时文件)
- 执行命令：./gsensor [loop count]

## ■ 演示示例

- 用USB线连接开发板的USB-to-UART 接口(J4)和主机的USB接口。
- 在Linux系统PC主机中，将gsensor文件复制到microSD卡(烧录了DE10-Standard Linux 镜像文件)的/home/root文件夹中。
- 将microSD卡插入DE10-Standard开发板的SD卡槽。
- 将SW10 MSEL[4:0]设置为01010。
- 接通DE10-Standard开发板电源。
- 启动PuTTY终端，与UART串口建立连接。输入root登陆Yocto Linux。
- 在PuTTY终端执行“./gsensor”命令运行程序。
- 如图6-8所示，终端将会打印输出X、Y和Z的值。

```

root@socfpga:~# ./gsensor
==== gsensor test ====
id=E5h
[1]X=80 mg, Y=-40 mg, Z=924 mg
[2]X=76 mg, Y=-32 mg, Z=972 mg
[3]X=76 mg, Y=-36 mg, Z=964 mg
[4]X=84 mg, Y=-36 mg, Z=976 mg
[5]X=76 mg, Y=-40 mg, Z=964 mg
[6]X=76 mg, Y=-40 mg, Z=972 mg

```

图 6-8 G-sensor 示例的终端输出结果

- 按主机键盘上的CTRL + C终止程序运行。

## 6.4 I2C MUX 测试

DE10-Standard开发板的I2C总线本来只能通过FPGA访问，本示例演示如何切换I2C多路复用器，实现HPS访问I2C总线。

### ■ 功能框图

图 6-9所示为本示例的功能框图。FPGA和HPS的I2C总线都连接到I2C多路复用器。与HPS中的GPIO1控制器相连的HPS\_I2C\_CONTROL控制I2C多路复用器。与G-sensor一样，HPS I2C也和HPS中的I2C0控制器连接。

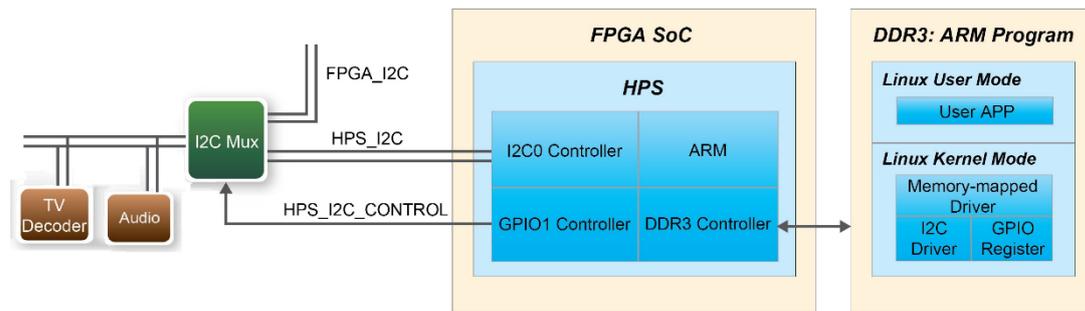


图 6-9 I2C MUX 测试示例功能框图

## ■ 控制 HPS\_I2C\_CONTROL

HPS\_I2C\_CONTROL与GPIO1控制器第19位的HPS\_GPIO48相连。HPS访问I2C总线后，当HPS\_I2C\_CONTROL信号设置为高电平时，HPS就能访问音频编解码器和TV解码器。

示例代码中定义的以下掩码用于控制HPS\_I2C\_CONTROL的方向及输出值：

```
#define HPS_I2C_CONTROL (0x00080000)
```

以下语句将HPS\_I2C\_CONTROL的相关引脚配置为输出引脚：

```
alt_setbits_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_SWPORTA_DDR_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ), HPS_I2C_CONTROL );
```

以下语句将HPS\_I2C\_CONTROL设置为高电平：

```
alt_setbits_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_SWPORTA_DR_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ), HPS_I2C_CONTROL );
```

以下语句将HPS\_I2C\_CONTROL设置为低电平：

```
alt_clrbits_word( ( virtual_base +
  ( ( uint32_t)( ALT_GPIO1_SWPORTA_DR_ADDR ) &
  ( uint32_t)( HW_REGS_MASK ) ), HPS_I2C_CONTROL );
```

## ■ I2C 驱动

通过系统I2C总线驱动从TV解码器读取寄存器值的步骤如下：

- 将HPS\_I2C\_CONTROL设置为高电平，以便HPS访问I2C总线。
- 打开I2C总线驱动“/dev/i2c-0”文件：open(“/dev/i2c-0”，O\_RDWR)。
- 指定ADV7180的I2C地址为0x20：ioctl(file, I2C\_SLAVE, 0x20)。
- 读或写寄存器。
- 将HPS\_I2C\_CONTROL设置为低电平，释放I2C总线。

## ■ 示例源代码

- 开发工具：Intel SoC FPGA EDS v16.1软件
- 工程目录：\Demonstration\SoC\ hps\_i2c\_switch
- 二进制文件：i2c\_switch
- 编译命令：make ("make clean"用于清除所有临时文件)
- 执行命令：./i2c\_switch

## ■ 运行示例

- 用USB线连接DE10-Standard开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 在Linux 系统PC 主机中，将i2c\_switch 文件复制到microSD 卡（烧录了DE10-Standard Linux镜像文件）的/home/root文件夹中。
  - 将microSD卡插入DE10-Standard开发板。
  - 将SW10 MSEL[4:0]设置为01010。
  - 接通DE10-Standard开发板电源。
  - 启动PuTTY，与UART串口建立连接。输入root登陆Yocto Linux系统。
  - 在PuTTY终端输入“./ i2c\_switch”命令，开始测试I2C MUX。
  - 如图 6-10所示，PuTTY终端显示示例运行结果。

```
root@socfpga:~# ./i2c_switch
I2C BUS Switch Test
HPS owns the I2C bus!!!
Open '/dev/i2c-0' successfully.
REG[11h]=1Ch (i2c addr:20h)
HPS release the I2C bus!!!
I2C Switch Test:Success
root@socfpga:~#
```

图 6-10 I2C MUX Test 示例的终端输出

- 按主机键盘上的CTRL + C终止程序运行。

## 6.5 SPI 接口 Graphic LCD

本示例演示如何使用HPS SPIM (SPI Master)控制器和HPS GPIO控制器控制Graphic LCD。

### ■ 功能框图

图 6-11所示为本示例的功能框图。LCD与DE10-Standard 开发板HPS端的SPIM0、GPIO1 控制器相连。通过系统内置的虚拟内存映射器件驱动可以访问HPS SPIM和GPIO控制器的寄存器。SPI 接口用于从HPS传输数据或指令到LCD。因为只能对LCD进行写操作，所以只需要

LCM\_SPIM\_CLK、LCM\_SPIM\_SS和LCM\_SPIM\_MOSI三个SPI信号。LCM\_D\_C信号表示在SPI总线上传输的信号是数据还是指令。当LCM\_D\_C信号上拉为高电平时，SPI总线上的信号即为数据；LCM\_D\_C信号下拉为低电平时，SPI总线上的信号即为指令。LCD\_RST\_n是LCD的复位控制信号，该信号为低电平有效。LCM\_BK信号用于开/关LCD的背光，该信号上拉为高电平时，LCD背光打开。

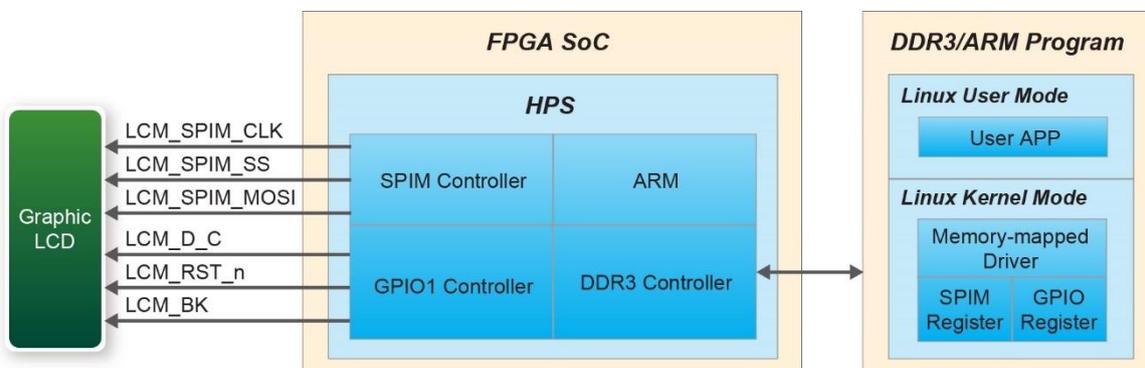


图 6-11 Graphic LCD 示例功能框图

## ■ LCD 控制

在发送任何显示数据之前，开发人员需要初始化LCD。初始化内容包括：

- 常用输出模式选择(Code: 0xC0~0xCF)
- 电源控制集(Code: 0x28~0x2F)
- 显示起始行集(Code: 0x40~0x7F)
- 页面地址集(Code: 0xB0~0xB8)
- 列地址集(Code: 0x00 to 0x18)
- 显示开/关(Code: 0xAE~0xAF)

有关命令集的详细信息，可以参考DE10-Standard系统CD中的NT7534数据表。LCD初始化后，开发人员可以开始传输显示数据。由于显示区域分为8页，在开始传输显示数据之前，开发人员首先需要指定目标页及列地址。图 6-12所示为当page = 0、column = 0和start line = 0时图像数据位与LCD显示像素之间的关系。

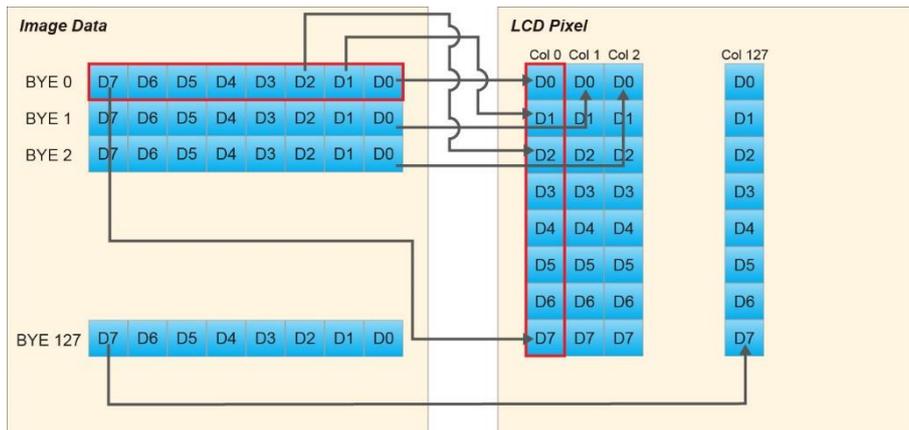


图 6-12 图像数据位与 LCD 显示像素的关系

## ■ SPIM 控制器

本示例中，HPS SPIM0 控制器被配置成只发送(TX-Only)模式的SPI，时钟频率为3.125MHz。详细信息可以参考LCD\_Hw.c文件中的LCDHW\_Init函数。嵌套在SPI控制器程序中的头文件socal/alt\_spim.h定义了SPIM控制器必需的所有常数。

## ■ C 代码解释

本示例包含以下主要文件：

- LCD\_HW.c: 底层SPI 和GPIO API 访问LCD硬件
- LCD\_Driver.c: LCD配置API
- LCD\_Lib.c: 高层LCD 控制 API
- lcd\_graphic.c: LCD的图形和字体API
- font.c: lcd\_graphic.c所用的字体位图源文件
- main.c: 本示例的主程序

主程序调用LCDHW\_Init函数初始化控制LCD的SPIM0和GPIO控制器，然后调用LCDHW\_BackLight函数打开LCD背光，调用LCD\_Init函数初始化LCD配置，最后调用lcd\_graphic.c中的API在LCD上绘制图像。

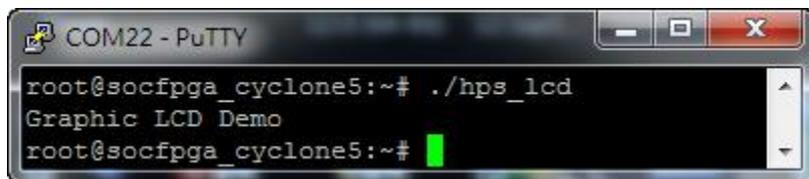
lcd\_graphic.c中的API不会直接驱动LCD绘制图像像素。所有图像像素都存储在临时图像缓冲区Canvas中。当调用DRAW\_Refresh API时，Canvas中的所有图像数据都会传输到LCD。本示例中，主程序首先调用DRAW\_Clear函数清除LCD Canvas，然后调用DRAW\_Rect函数和DRAW\_Circle函数在Canvas中绘制字体，最后调用DRAW\_Refresh函数将Canvas数据传输到LCD上。

## ■ 源代码

- 开发工具：SoC FPGA EDS v16.1软件
- 工程目录：\Demonstration\SoC\hps\_lcd
- 二进制文件：hps\_lcd
- 编译命令：make (“make clean” 用于清除所有临时文件)
- 执行命令：./hps\_lcd

## ■ 运行示例

- 用USB线连接DE10-Standard开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 在Linux 系统 PC 主机中，将 hps\_lcd 文件复制到 microSD 卡（烧录了 DE10-Standard Linux 镜像文件）的/home/root文件夹中。
- 将microSD卡插入DE10-Standard开发板的SD卡槽。
- 将SW10 MSEL[4:0]设置为01010。
- 接通DE10-Standard开发板电源。
- 启动PuTTY，与UART串口建立连接。输入root登陆Yocto Linux系统。
- 如图 6-13所示，在PuTTY终端输入“./hps\_lcd”运行LCD示例程序。



```
COM22 - PuTTY
root@socfpga_cyclone5:~# ./hps_lcd
Graphic LCD Demo
root@socfpga_cyclone5:~#
```

图 6-13 运行 LCD 示例

- LCD上显示会显示程序运行结果，如图 6-14所示。



图 6-14 LCD 示例运行结果

## 6.6 设置 USB Wi-Fi Dongle

本节介绍如何在Linux环境下设置Wi-Fi USB Dongle，这样Linux用户便可以通过Wi-Fi USB Dongle无线连接Wi-Fi AP (Access Point)，并最终连接到互联网。这里假设Wi-Fi AP有DHCP服务器功能并已经连接互联网。用户还需确保知道Wi-Fi AP的SSID和密码。

### ■ 系统框图

图 6-15所示为本示例的功能框图。Wi-Fi AP认为您已具备DHCP服务器功能并连接到LAN（局域网）或互联网。USB Wi-Fi Dongle连接Wi-Fi AP并从Wi-Fi AP获取一个IP地址。通过Wi-Fi AP，USB-Dongle能够与连接到LAN或互联网的设备进行通信。

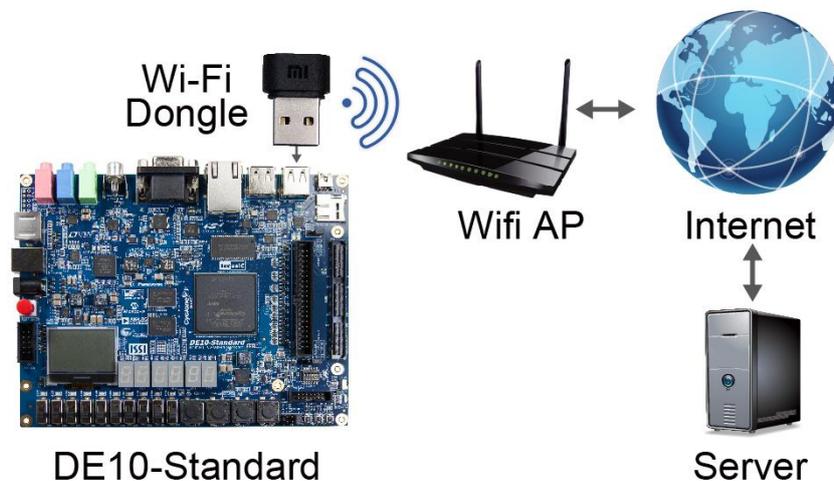


图 6-15 USB Wi-Fi Dongle 功能框图

### ■ Wi-Fi 设置步骤

- 用USB线连接DE10-Standard开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 用USB OTG线连接开发板的micro USB端口(J7或J8)与USB Wi-Fi Dongle。
- 将SW10 MSEL[4:0]设置为01010。
- 接通DE10-Standard开发板电源。
- 启动PuTTY，与UART串口建立连接。输入root进入Linux系统(无需输入密码)。
- 在PuTTY终端输入“ifconfig wlan0 up”命令，配置wlan0 网口。
- 在PuTTY终端输入“iwlist wlan0 scan | grep ESSID”命令，搜索附近的Wi-Fi AP。确保能找到可用的Wi-Fi AP。

```
# iwlist wlan0 scan | grep ESSID
ESSID:"
ESSID:"Terasic"
ESSID:"
```

- 在PuTTY终端输入“vim /etc/wpa\_supplicant/wpa\_supplicant.conf”命令，编辑Wi-Fi配置文件。

```
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid="Your_SSID"
    psk="Your_WPA-Key_ASCII"
}
```

- 将配置文件中的“Your\_SSID”和“Your\_WPA-Key\_ASCII”分别替换为Wi-Fi AP的SSID和密码。

```
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid="Terasic"
    psk="1234567890"
}
```

- 在PuTTY终端输入“ifup wlan0”命令，连接Wi-Fi AP。

```
root@DE10-Standard:~# ifup wlan0
Internet Systems Consortium DHCP Client 4.3.3
Copyright 2004-2015 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/f0:b4:29:3c:eb:7a
Sending on LPF/wlan0/f0:b4:29:3c:eb:7a
Sending on Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 3 (xid=0x34840b19)
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 8 (xid=0x34840b19)
DHCPCREQUEST of 192.168.2.2 on wlan0 to 255.255.255.255 port 67 (xid=0x190b8434)
DHCPOFFER of 192.168.2.2 from 192.168.2.1
DHCPACK of 192.168.2.2 from 192.168.2.1
bound to 192.168.2.2 -- renewal in 38446 seconds.
```

- 在PuTTY终端输入“ifconfig wlan0”命令，确认IP地址已分配给wlan0接口。

```

root@DE10-Standard:~# ifconfig wlan0
wlan0    Link encap:Ethernet  HWaddr f0:b4:29:3c:eb:7a
         inet addr:192.168.2.2    Bcast:192.168.2.255  Mask:255.255.255.0
         inet6 addr: fe80::f2b4:29ff:fe3c:eb7a/64  Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:9 errors:0 dropped:0 overruns:0 frame:0
         TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:2086 (2.0 KB)  TX bytes:2188 (2.1 KB)

```

- 确保Wi-Fi AP 已连接互联网。在PuTTY终端输入“ping -c 4 www.terasic.com”命令，检查网络连接状态。如果丢包率为0，表示网络连接正常。

```

root@DE10-Standard:~# ping -c 4 www.terasic.com
PING www.terasic.com (192.254.233.22) 56(84) bytes of data.
64 bytes from www.terasic.com (192.254.233.22): icmp_seq=1 ttl=50 time=180 ms
64 bytes from www.terasic.com (192.254.233.22): icmp_seq=2 ttl=50 time=203 ms
64 bytes from www.terasic.com (192.254.233.22): icmp_seq=3 ttl=50 time=160 ms
64 bytes from www.terasic.com (192.254.233.22): icmp_seq=4 ttl=50 time=205 ms

--- www.terasic.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 160.602/187.540/205.380/18.345 ms

```

## 6.7 查询网络时间

本示例演示如何使用定时web服务器并通过互联网查询网络时间。UART终端将以HH:MM:SS格式显示时间信息。DE10-Standard开发板通过RJ45端口或Wi-Fi USB Dongle 连接以太网。关于如何设置Wi-Fi USB Dongle的详细信息，请参考6.6 设置USB Wi-Fi Dongle。

### ■ 功能框图

图 6-16所示为本示例的功能框图。第三方免费库文件libcurl用于处理URL传输任务。主程序使用“http get”请求查询网页内容，并在Nios II终端直接显示响应内容。

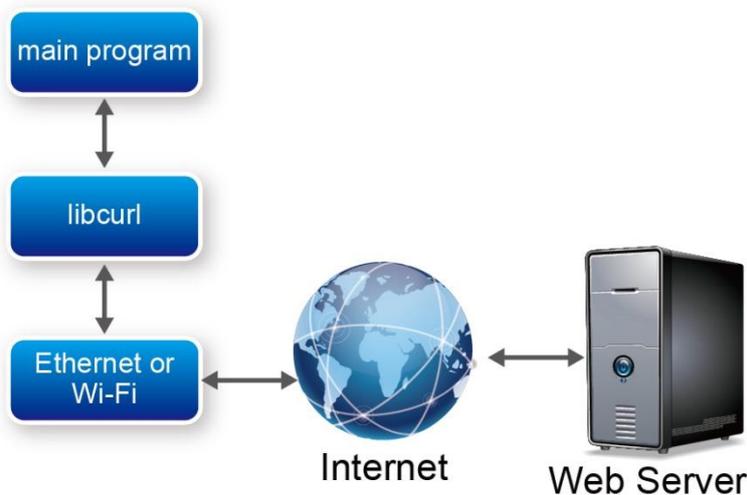


图 6-16 网络时间示例功能框图

## ■ URL 传输库: libcurl

Libcurl库文件是为网络客户端而设计，这个库文件实现了复杂的网络协议，为开发人员提供了简易的C代码API。客户端程序开发人员通过调用库中导出的API能轻松的与服务器进行通信。

更多详细信息请参考<https://curl.haxx.se/libcurl/>

## ■ 如何查询网络时间

网络时间信息在<http://demo.terasic.com>链接中，发送<http://demo.terasic.com/time/>链接到web服务器，它将以HH:MM:SS格式反馈当前时间。

## ■ 示例源代码

- 开发工具: SoC FPGA EDS V16.1软件
- 工程目录: \Demonstration\SoC\_Advanced\NET\_Time
- 二进制文件: NET\_Time
- 编译命令: make (“make clean” 用于清除所有临时文件)
- 执行命令: ./NET\_Time

## ■ 运行示例

- 用USB线连接DE10-Standard开发板的USB-to-UART 接口(J4)与主机的USB接口。
- 在Linux 系统PC 主机中，将NET\_Time 文件复制到microSD 卡（烧录了DE10-Standard Linux镜像文件）的/home/root文件夹中。
  - 将micro SD 卡插入DE10-Standard开发板的SD卡槽。
  - 将SW10 MSEL[4:0]设置为01010。
  - 接通DE10-Standard开发板电源。
  - 启动PuTTY，与UART串口建立连接。输入root进入Linux系统(无需输入密码)。
  - 在PuTTY终端输入“./NET\_Time”命令运行程序。

```
root@DE10-Standard:~# ./NET_Time  
e3:25:26
```

- PuTTY终端将显示当前UTC(Universal Time Coordinated)时间。
- 按主机键盘上的CTRL + C终止程序运行。

# 第七章

## FPGA 和 HPS 协同工作的示例

本章将介绍如何使用HPS/ARM与FPGA进行通信。我们将介绍DE10-Standard开发板的GHRD工程，我们还开发了基于ARM的C代码工程来演示HPS/ARM程序如何控制FPGA端的10个LED。我们还将演示HPS如何通过轻型（Lightweight）HPS-to-FPGA桥接控制FPGA端的LED。本示例中的FPGA是通过HPS中的FPGA管理器进行配置。

### 7.1 需求背景

本节假设开发者已经掌握以下知识：

#### ■ FPGA RTL 设计

- Quartus Prime软件使用基本技能
- RTL基本编程技能
- Qsys工具使用基本技能
- 内存映射接口（Memory-Mapped Interface）知识

#### ■ C 语言程序设计

- SoC FPGA EDS(Embedded Design Suite)软件使用基本技能
- C语言基本编程与编译技能
- 熟练使用给定的镜像文件创建用于DE10-Standard开发板的Linux系统启动SD卡
- 熟练掌握从SD卡启动Linux
- 熟练复制文件到DE10-Standard开发板的Linux文件系统；Linux基本命令与使用技能

### 7.2 系统需求

请注意运行本章的示例工程之前需要准备好以下设备：

#### ■ DE10-Standard 开发板，包括：

- Mini USB线，用于连接UART终端
- 容量至少为4GB的Micro SD卡
- Micro SD卡读卡器

## ■ 一台 x86 主机

- 64位Windows 7操作系统
- USB接口
- V16.1或更新版本的Quartus Prime软件
- V16.1或更新版本的SoC FPGA EDS软件
- Win32 Disk Imager软件

## 7.3 Intel SoC FPGA 内的 AXI Bridge

在 Intel SoC FPGA 中，HPS 逻辑和 FPGA 架构通过 AXI (Advanced eXtensible Interface, 高级可扩展接口) 桥接。为了使 HPS 逻辑与 FPGA 架构相互通讯,在系统设计时需要使用 Intel 系统集成工具 Qsys 添加 HPS 组件, Qsys 组件的内存映射从设备端口与主设备端口相连, HPS 可以访问这些 Qsys 组件。

HPS 包含以下 HPS-FPGA AXI 桥接:

- FPGA-to-HPS桥接
- HPS-to-FPGA桥接
- 轻型HPS-to-FPGA桥接

图 7-1 所示为 FPGA 架构中的 AXI 桥接的功能框图,可以看到 L3 在 HPS 端,每个主设备 (M) 与从设备 (S) 接口都有对应的数据位宽,括号中注明了每个互连的时钟域。

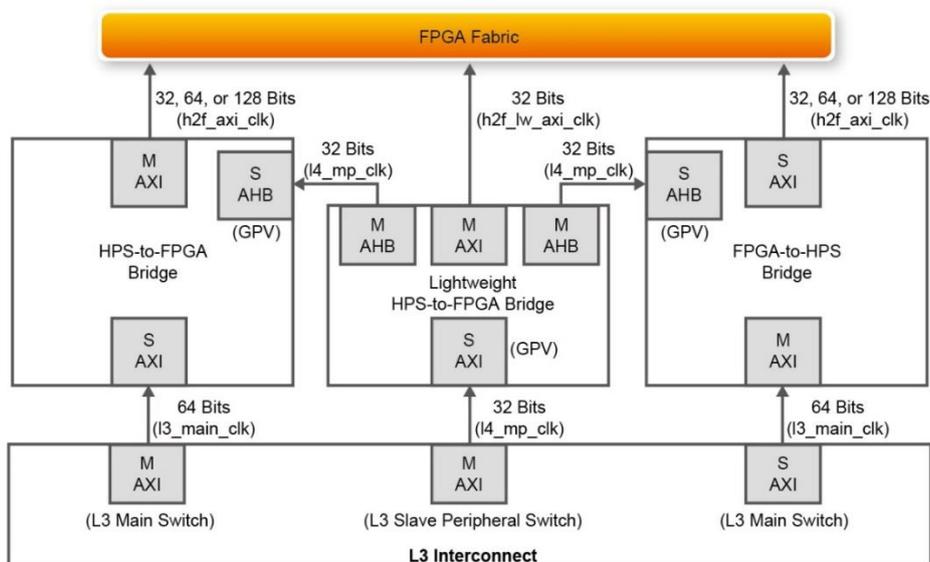


图 7-1 AXI 桥接的功能框图

HPS-to-FPGA 桥接由 L3 Main Switch 控制，轻型 HPS-to-FPGA 桥接由 L3 Slave Peripheral Switch 控制。

FPGA-to-HPS 桥接控制 L3 Main Switch，允许 FPGA 架构中的任何主设备访问 HPS 端的大部分从设备，例如 FPGA-to-HPS 桥接可以访问加速度传感器。

三个桥接都包含了 GPV（Global Programmers View）寄存器。GPV 寄存器控制桥接的行为，能通过轻型 HPS-to-FPGA 桥接访问三个桥接的 GPV 寄存器。

本示例介绍如何用 HPS/ARM 与 FPGA 进行通讯。示例工程包含了 DE10-Standard 开发板的 GHRD 工程和一个演示 HPS/ARM 程序是如何控制 FPGA 端的红色 LED 的 ARM C 工程。

## 7.4 GHRD 工程

GHRD是Golden Hardware Reference Design的缩写。GHRD工程是Terasic为DE10-Standard开发板开发的，它在DE10-Standard系统CD的\Demonstration\SoC\_FPGA\DE10\_Standard\_GHRD目录中。

该工程包括以下组件：

- ARM Cortex™-A9 MPCore HPS
- 4个用户按钮开关输入
- 10个用户拨动开关输入
- 10个用户I/O LED输出
- 64KB片上内存
- JTAG to Avalon主设备桥接
- 与System Console一起使用的中断捕获器（Interrupt capturer）
- System ID

SoC FPGA的FPGA部分的系统外设（相当于MPU的从外设）的内存映射地址始于轻型HPS-to-FPGA的基地址0xFF20\_0000。通过Qsys中的地址偏移设置，MPU可以访问这些外设。用Quartus Prime软件打开GHRD工程，然后用Platform Designer工具打开soc\_system.qsys文件。图7-2所示为连接轻型HPS-to-FPGA桥接的外设的地址映射。

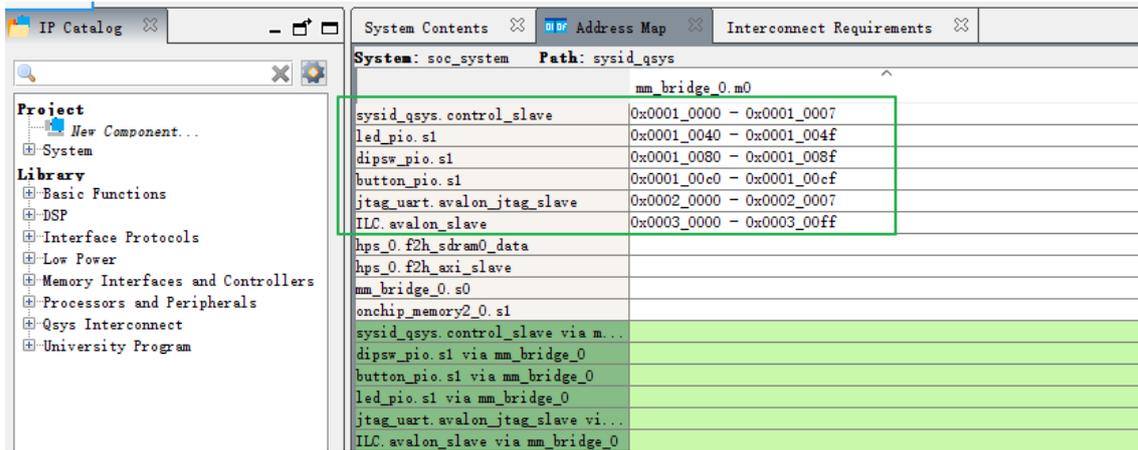


图 7-2 FPGA 外设的地址映射

如图 7-3 所示，这些外设的 Avalon Conduit 信号都连接到 DE10-Standard 开发板的 SoC FPGA 器件的 I/O 引脚。

```

//HPS SPI
.hps_0_hps_io_hps_io_spim1_inst_CLK ( HPS_SPIM_CLK ), //
.hps_0_hps_io_hps_io_spim1_inst_MOSI ( HPS_SPIM_MOSI ), //
.hps_0_hps_io_hps_io_spim1_inst_MISO ( HPS_SPIM_MISO ), //
.hps_0_hps_io_hps_io_spim1_inst_SS0 ( HPS_SPIM_SS ), //
//HPS UART
.hps_0_hps_io_hps_io_uart0_inst_RX ( HPS_UART_RX ), //
.hps_0_hps_io_hps_io_uart0_inst_TX ( HPS_UART_TX ), //
//HPS I2C1
.hps_0_hps_io_hps_io_i2c0_inst_SDA ( HPS_I2C0_SDAT ), //
.hps_0_hps_io_hps_io_i2c0_inst_SCL ( HPS_I2C0_SCLK ), //
//HPS I2C2
.hps_0_hps_io_hps_io_i2c1_inst_SDA ( HPS_I2C1_SDAT ), //
.hps_0_hps_io_hps_io_i2c1_inst_SCL ( HPS_I2C1_SCLK ), //
//GPIO
.hps_0_hps_io_hps_io_gpio_inst_GPI009 ( HPS_CONV_USB_N ), //
.hps_0_hps_io_hps_io_gpio_inst_GPI035 ( HPS_ENET_INT_N ), //
.hps_0_hps_io_hps_io_gpio_inst_GPI040 ( HPS_LTC_GPIO ), //
.hps_0_hps_io_hps_io_gpio_inst_GPI053 ( HPS_LED ), //
.hps_0_hps_io_hps_io_gpio_inst_GPI054 ( HPS_KEY ), //
.hps_0_hps_io_hps_io_gpio_inst_GPI061 ( HPS_GSENSOR_INT ), //
//FPGA Partition
.led_pio_external_connection_export ( fpga_led_internal ), // led_pio_external_connection.export
.dipsw_pio_external_connection_export ( sw ), // dipsw_pio_external_connection.export
.button_pio_external_connection_export ( fpga_debounced_buttons ), // button_pio_external_connection.export
.hps_0_h2f_reset_reset_n ( hps_fpga_reset_n ), // hps_0_h2f_reset.reset_n
.hps_0_f2h_cold_reset_req_reset_n ( ~hps_cold_reset ), // hps_0_f2h_cold_reset_req.reset_n
.hps_0_f2h_debug_reset_req_reset_n ( ~hps_debug_reset ), // hps_0_f2h_debug_reset_req.reset_n
.hps_0_f2h_stm_hw_events_stm_hwevents ( stm_hw_events ), // hps_0_f2h_stm_hw_events.stm_hwevents
.hps_0_f2h_warm_reset_req_reset_n ( ~hps_warm_reset ), // hps_0_f2h_warm_reset_req.reset_n

```

图 7-3 顶层设计中的 Avalon Conduit 信号连接

## 7.5 编译与编程

点击 Qsys 工具菜单栏的 **Generate-->Generate...**，生成 Qsys 系统的源代码，然后关闭 Qsys 工具。依次点击 Quartus 软件菜单栏的 **Processing-->Start Compilation**，开始编译工程。

因为 HPS SDRAM DDR3 控制器的 .tcl 文件在 GHRD 工程中已经执行，开发人员可以跳过此过程。如果 Quartus 工程不是基于 GHRD 工程开发的，那么在编译工程之前请记得运行 SDRAM DDR3 控制器的 .tcl 文件，如图 7-4 所示。

依次点击 Quartus 软件菜单栏的 **Tools-->TCL Scripts...**，启动 TCL Scripts 对话框。需要执行 <qsys\_system\_name>\_parameters.tcl 和 <qsys\_system\_name>\_pin\_assignments.tcl 脚本文件，其中 <qsys\_system\_name> 是 Qsys 系统的名称。运行此脚本为 SDRAM DDR3 组件指定约束。

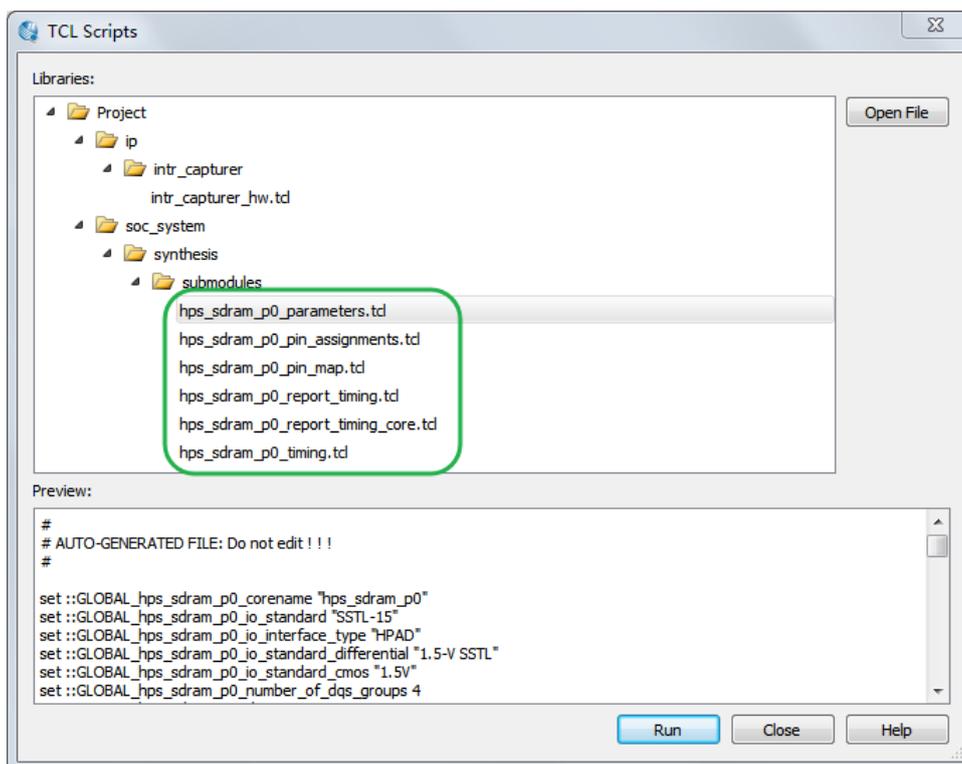


图 7-4 运行 SDRAM 控制器的.tcl 文件

点击 **Processing-->Start Compilation** 编译 Quartus 工程。编译成功后在 DE10\_Standard\_GHRD\output\_files 目录中会生成 DE10\_Standard\_GHRD.sof 文件,通过 DE10-Standard 开发板的 USB-Blaster II 接口可以将该.sof 文件配置到 FPGA。

## 7.6 开发 C 代码

本节介绍如何设计基于 ARM 的 C 程序来控制 led\_pio PIO 控制器。SoC FPGA EDS 软件用于编译 C 代码。ARM 程序需要 led\_pio 地址才能控制 led\_pio PIO 组件。Linux 内置驱动/ dev/mem 和 mmap 函数将 led\_pio 组件的物理基址映射到 Linux 应用程序能直接访问的虚拟地址。

### ■ HPS 头文件

基于 ARM 的 C 程序需要用到 pio\_led 组件信息才能控制 pio\_led 组件。本节介绍如何使用给定的 Linux shell 批处理文件获取 Qsys HPS 信息到头文件,然后嵌套在 C 程序中。

上面提到的批处理文件是 generate\_hps\_qsys\_header.sh,它与 DE10\_Standard\_GHRD Quartus 工程在同一个文件夹中。启动 SoC FPGA EDS 软件的命令行窗口,使用 cd 命令切换到 Quartus 工程所在的文件夹路径,输入 "./generate\_hps\_qys\_header.sh" 命令,按 Enter 键执行后即可生成头文件 hps\_0.h。如图 7-5 所示,在头文件中,LED\_PIO\_BASE 常量表示 led\_pio 的基地址。LED\_PIO\_DATA\_WIDTH 常量表示 led\_pio 的宽度,这两个常量在示例的 C 程序中会用到。

```

/*
 * Macros for device 'led_pio', class 'altera_avalon_pio'
 * The macros are prefixed with 'LED_PIO_'.
 * The prefix is the slave descriptor.
 */
#define LED_PIO_COMPONENT_TYPE altera_avalon_pio
#define LED_PIO_COMPONENT_NAME led_pio
#define LED_PIO_BASE 0x10040
#define LED_PIO_SPAN 16
#define LED_PIO_END 0x1004f
#define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
#define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
#define LED_PIO_CAPTURE 0
#define LED_PIO_DATA_WIDTH 10
#define LED_PIO_DO_TEST_BENCH_WIRING 0
#define LED_PIO_DRIVEN_SIM_VALUE 0
#define LED_PIO_EDGE_TYPE NONE
#define LED_PIO_FREQ 50000000

```

图 7-5 pio\_led 信息

## ■ 映射 LED\_PIO 地址

本节介绍如何将pio\_led 物理地址映射到应用软件可以访问的虚拟地址。图 7-6所示的C程序可以获取led\_pio基地址的虚拟地址。首先，open函数打开内存器件驱动/dev/mem，然后mmap函数将HPS物理地址映射到void virtual\_base表示的虚拟地址。示例代码将外设区域的物理基地址 (HW\_REGS\_BASE = 0xfc000000)映射到虚拟基地址virtual\_base。用户可以将外设区域中的任何控制器相对于外设区域的偏移量添加到虚拟基地址virtual\_base，从而计算控制器的虚拟地址。根据此规则，将以下两个偏移地址添加到virtual\_base就能计算出 led\_pio的虚拟地址。

- 相对于 HPS 基地址的轻型HPS-to-FPGA AXI 总线的偏移地址
- 相对于轻型HPS-to-FPGA AXI 总线的 Pio\_led 偏移地址

第一个偏移地址是 0xff200000，在头文件 hps.h 文件中定义为常量 ALT\_LWFPGASLVS\_OFST。hps.h是SoC EDS的一个头文件，位于Quartus软件的安装文件夹中，比如C:\intel\FPGA\16.1\embedded\ip\altera\hps\altera\_hps\hwlib\include\soc\_cv\_av\socal。第二个偏移地址是0x3000，在头文件hps\_0.h中定义为常量LED\_PIO\_BASE，头文件hps\_0.h文件在7.5 编译与编程这一节中生成。

Void h2p\_lw\_led\_addr表示led\_pio的虚拟地址。应用程序可以直接使用这个指针变量来访问

问LED\_PIO控制器中的寄存器。

```
#define HW_REGS_BASE ( ALT_STM_OFST )
#define HW_REGS_SPAN ( 0x04000000 )
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )

int main() {

    void *virtual_base;
    int fd;
    int loop_count;
    int led_direction;
    int led_mask;
    void *h2p_lw_led_addr;

    // map the address space for the LED registers into user space so we can interact with them.
    // we'll actually map in the entire CSR span of the HPS since we want to access various
    // registers within that span

    if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
        printf( "ERROR: could not open \"/dev/mem\"...\n" );
        return( 1 );
    }

    virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE ), MAP_SHARED, fd, HW_REGS_BASE );

    if( virtual_base == MAP_FAILED ) {
        printf( "ERROR: mmap() failed...\n" );
        close( fd );
        return( 1 );
    }

    h2p_lw_led_addr = virtual_base
        ( (unsigned long) (ALT_LWFGASLVS_OFST + LED_PIO_BASE) & (unsigned long) (HW_REGS_MASK) );
}
```

图 7-6 LED\_PIO 内存映射代码

## ■ 控制 LED

开发人员在控制LED之前需要先了解LED\_PIO PIO核的寄存器映射。图 7-7所示为PIO核的寄存器映射关系图，每个寄存器都是32位宽，可以参考PIO核的数据表获取更详细信息。对于LED控制，只需要将输出值写到相对于基地址0x10040的偏移地址为0的寄存器。由于DE10-Standard开发板的LED为高电平点亮，所以将0x00000000值写入偏移地址为0的寄存器会使9个红色LED熄灭。DE10-Standard开发板上有10个红色LED，其中9个连接到该控制器，最后一个LED (LED0) 用于实现FPGA心跳功能。将0x000001ff值写入偏移地址为0的寄存器将点亮9个红色LED。如下所示是将0x000001ff值写入led\_pio的偏移地址为0的C程序代码：

```
*(uint32_t *) h2p_lw_led_addr= 0x000001ff;
```

该语句将uint32\_t 指定为void型指针，C编译器将32位值0x000001ff 写入虚拟地址h2p\_lw\_led\_addr。

Offset	Register Name		R/W	Fields				
				(n-1)	...	2	1	0
0	data	read access	R	Data value currently on PIO inputs.				
		write access	W	New value to drive on PIO outputs.				
1	direction (1)		R/W	Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.				
2	interruptmask (1)		R/W	IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.				
3	edgecapture (1), (2)		R/W	Edge detection for each input port.				
4	outset		W	Specifies which bit of the output port to set.				
5	outclear		W	Specifies which output bit to clear.				

图 7-7 PIO 内核寄存器映射

## ■ 主程序

主程序控制LED呈流水灯式点亮，如图 7-8所示。完成60个流水灯周期后，程序停止运行。

```

loop_count = 0;
led_mask = 0x01;
led_direction = 0; // 0: left to right direction
while( loop_count < 60 ) {

    // control led, add ~ because the led is low-active
    *(uint32_t *)h2p_lw_led_addr = ~led_mask;

    // wait 100ms
    usleep( 100*1000 );

    // update led mask
    if (led_direction == 0){
        led_mask <<= 1;
        if (led_mask == (0x01 << (PIO_LED_DATA_WIDTH-1)))
            led_direction = 1;
    }else{
        led_mask >>= 1;
        if (led_mask == 0x01){
            led_direction = 0;
            loop_count++;
        }
    }
}
} // while

```

图 7-8 流水灯的 C 程序

## ■ Makefile 与编译

图7-9所示为C程序的Makefile文件内容。该程序嵌套了SoC EDS 提供的头文件。在Makefile中也指定了ARM-linux交叉编译。

```
#
TARGET = HPS_FPGA_LED

#
ALT_DEVICE_FAMILY ?= soc_cv_av
SOCEDS_ROOT ?= $(SOCEDS_DEST_ROOT)
HWLIBS_ROOT = $(SOCEDS_ROOT)/ip/altera/hps/altera_hps/hwlib
CROSS_COMPILE = arm-linux-gnueabihf-
CFLAGS = -g -Wall -D$(ALT_DEVICE_FAMILY) -I$(HWLIBS_ROOT)/include/$(ALT_DEVICE_FAMILY) -I$(HWLIBS_ROOT)/include/
LDFLAGS = -g -Wall
CC = $(CROSS_COMPILE)gcc
ARCH= arm

build: $(TARGET)
$(TARGET): main.o
    $(CC) $(LDFLAGS) $^ -o $@
%.o : %.c
    $(CC) $(CFLAGS) -c $< -o $@

.PHONY: clean
clean:
    rm -f $(TARGET) *.a *.o *~
```

图 7-9 Makefile 内容

在图 7-10所示的窗口中输入“make”命令并执行编译工程，然后输入“ls”命令检查生成的ARM可执行文件“HPS\_FPGA\_LED”。

```
matthew@matthew-PC /cygdrive/e/SVN/DE10_Standard/demonstration/SoC_FPGA/HPS_FPGA_LED
$ ls
hps_0.h main.c Makefile

matthew@matthew-PC /cygdrive/e/SVN/DE10_Standard/demonstration/SoC_FPGA/HPS_FPGA_LED
$ make
arm-linux-gnueabihf-gcc -g -Wall -Dsoc_cv_av -ID:/altera/16.0/embedded/ip/altera/hps/altera_hps/hwlib/include/soc_cv_av -ID:/altera/16.0/embedded/ip/altera/hps/altera_hps/hwlib/include/ -c main.c -o main.o
arm-linux-gnueabihf-gcc -g -Wall main.o -o HPS_FPGA_LED

matthew@matthew-PC /cygdrive/e/SVN/DE10_Standard/demonstration/SoC_FPGA/HPS_FPGA_LED
$ ls
hps_0.h HPS_FPGA_LED main.c main.o Makefile

matthew@matthew-PC /cygdrive/e/SVN/DE10_Standard/demonstration/SoC_FPGA/HPS_FPGA_LED
$
```

图 7-10 编译 ARM C 程序

## ■ 运行示例

运行示例前，首先要从 DE10-Standard 开发板上的 SD 卡启动 Linux。将可执行文件 HPS\_FPGA\_LED 复制到 Linux 目录，输入“`chmod +x HPS_FPGA_LED`”修改可执行文件的执行权限。使用 Quartus Programmer 工具将 7.5 编译与编程这一节中生成的 DE10\_Standard\_GHRD.sof 文件烧录到 FPGA。LED0 会闪烁，犹如 FPGA 在心跳。然后输入“`./HPS_FPGA_LED`”命令运行 ARM 程序。DE10-Standard 开发板上的 LED[9..1]将执行 60 次流水灯式点亮，然后程序停止运行。

请参考 *Getting\_Started\_Guide.pdf* 文档了解如何从 SD 卡启动 Linux 的详细信息。

请参考 *My\_First\_HPS.pdf* 文档了解如何将文件复制到 Linux 目录的详细信息。

# 第八章

## 固化 EPCS 器件

本章将介绍如何通过JTAG接口并利用SFL功能固化四串行配置EPCS器件。用户可以使用JTAG间接配置(JTAG indirect configuration)文件 (.jic) 配置EPCS器件, 该文件由Quartus工程编译成功后生成的.sof文件转换而来。以下步骤介绍如何将.sof 文件转换为.jic文件。

### 8.1 固化前的设置

将 DE10-Standard 开发板设置为 AS x1 FPGA 配置模式。当 MSEL[4:0]设置为 10010 时, 就可以用四串行存储器件 EPCS 作为 FPGA 的配置器件。

### 8.2 将.sof 文件转换为.jic 文件

- 1、如图 8-1 所示, 从 Quartus 软件的 File 菜单中选择 **Convert Programming Files**。

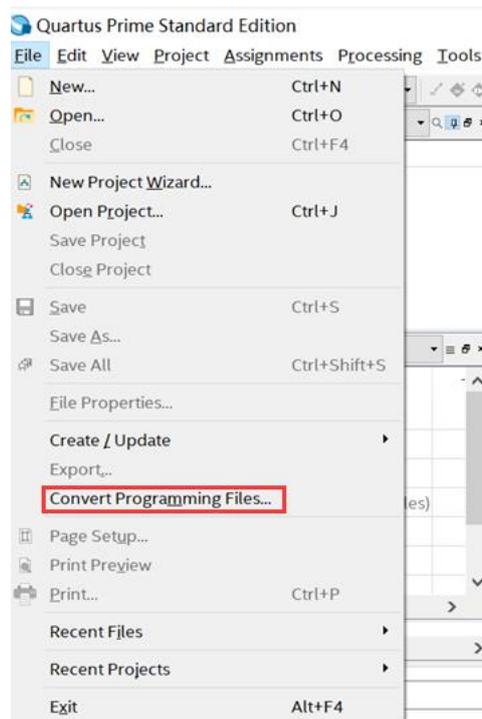


图 8-1 Quartus 软件的 File 菜单

- 2、在 Convert Programming Files 对话框的 **Programming file type** 下拉框中选择 JTAG Indirect Configuration File (.jic)。
- 3、在 Configuration device 下拉框中选择 EPCS128。
- 4、在 **Mode** 下拉框中选择 **Active Serial**。

- 5、点击**File name**后的“...”按钮，选择保存输出文件(.jic)的文件夹并命名输出文件。
- 6、如图 8-2所示，选中**Input files to convert**中的SOF data。

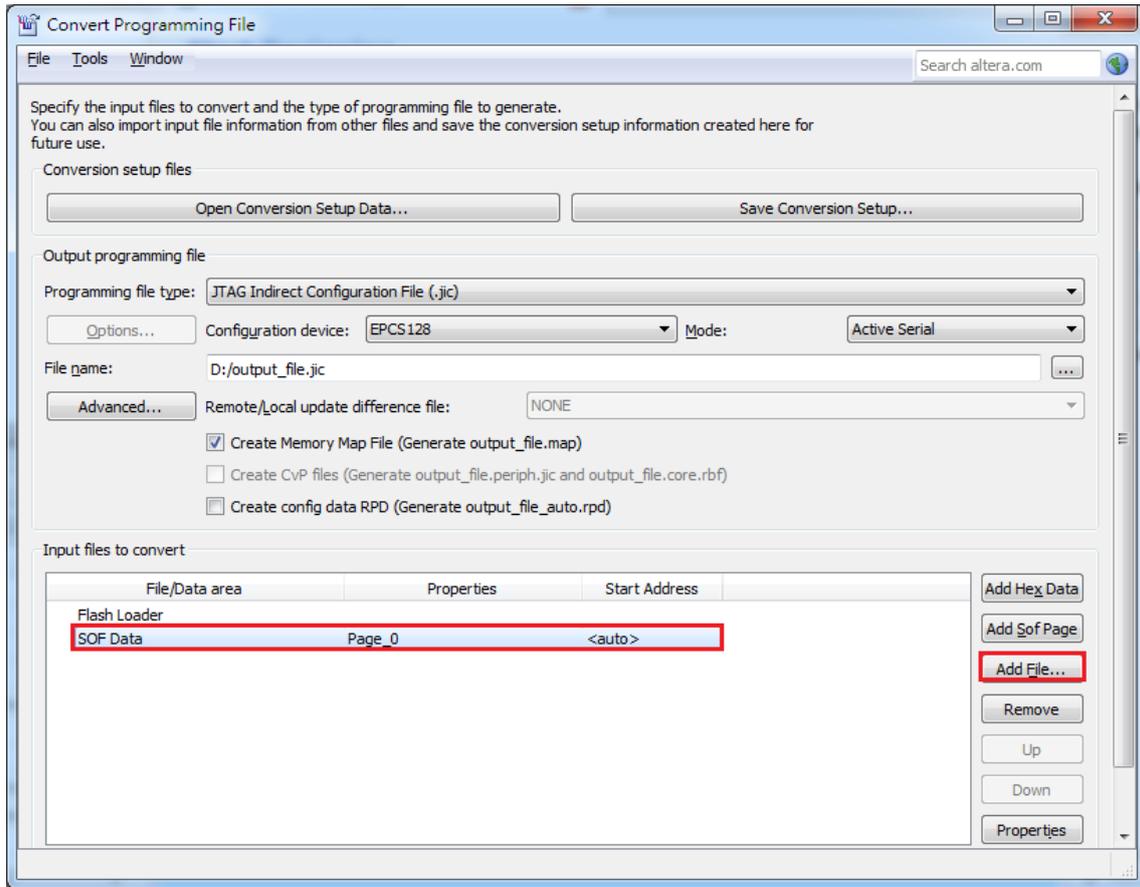


图 8-2 Convert Programming Files 对话框

- 7、点击**Add File**按钮。
- 8、在弹出的**Select Input File**对话框中选择要转换为.jic文件的.sof文件。
- 9、点击**Open**按钮。
- 10、如图 8-3所示，选中**Flash Loader**后再点击**Add Device**按钮，弹出Select Devices窗口。

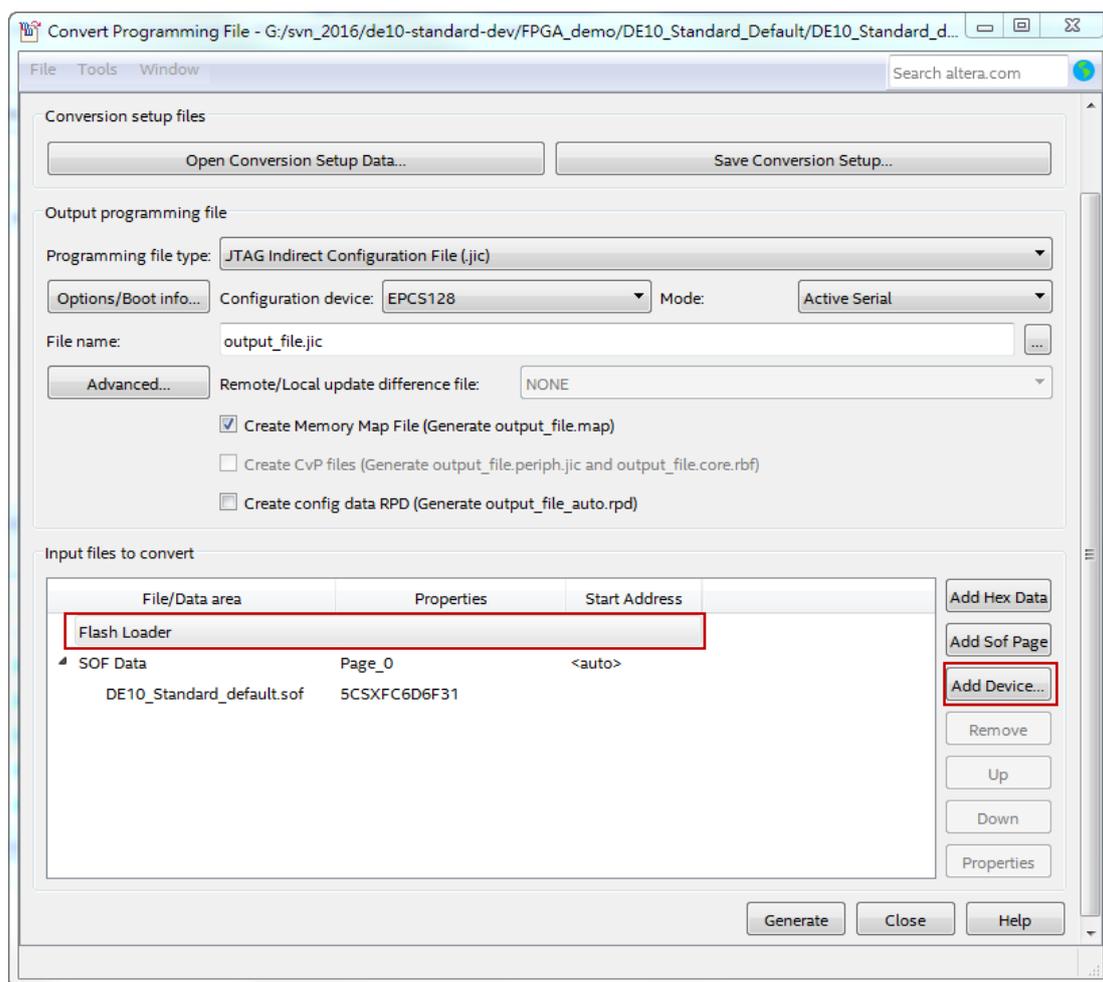


图 8-3 点击 Flash Loader

11、如图 8-4所示，选择对应的FPGA器件。

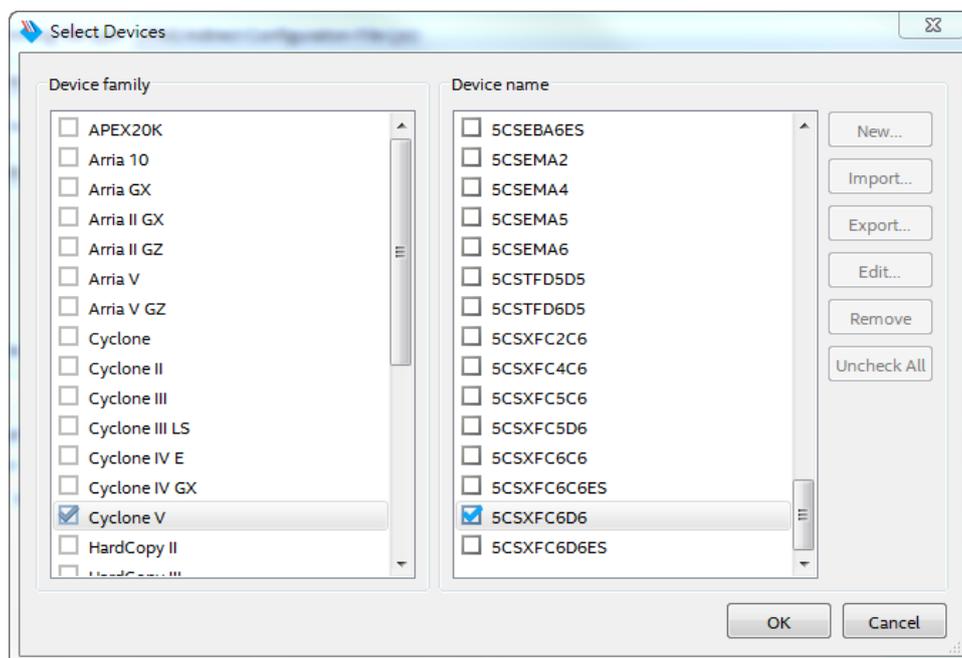


图 8-4 Select Devices 页面

12、点击OK按钮，Convert Programming Files窗口如图 8-5所示。

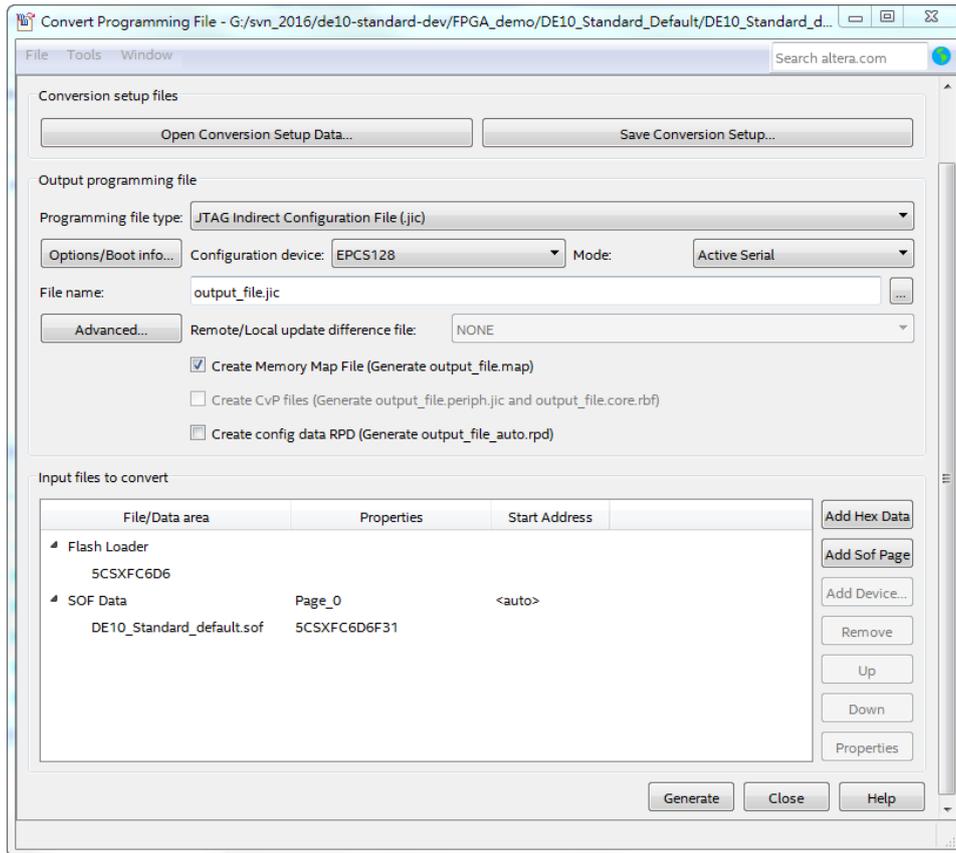


图 8-5 Convert Programming Files 页面

13、点击Generate按钮开始转换。

### 8.3 将.jic 文件写入 EPCS 器件

.sof 文件转换为.jic 文件完成后，请按照以下步骤在 Quartus Prime Programmer 工具中将.jic 文件写入 EPCS 器件。

- 1、将开发板的SW10 MSEL[4..0]设置为10010。
- 2、从Quartus软件的Tools菜单中选择**Programmer**，弹出Programmer窗口。
- 3、点击**Auto Detect**按钮，选择对应的FPGA器件，如图 8-6所示，检测到FPGA器件与HPS。

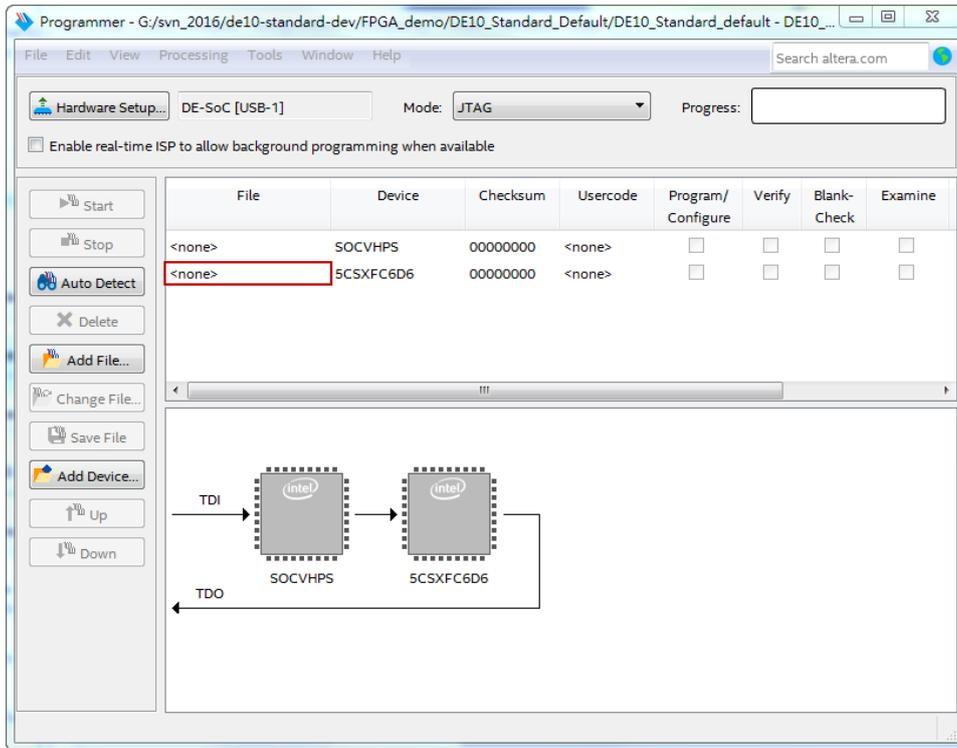


图 8-6 Programmer 工具检测到两个 FPGA 和 HPS

4、双击图 8-6所示的红色矩形区域，在出现的Select New Programming File窗口中选择.jic文件。

5、如图 8-7所示，勾选对应的**Program/Configure**复选框，FPGA Device的所在的File列会自动加载Factory default enhanced SFLImage。

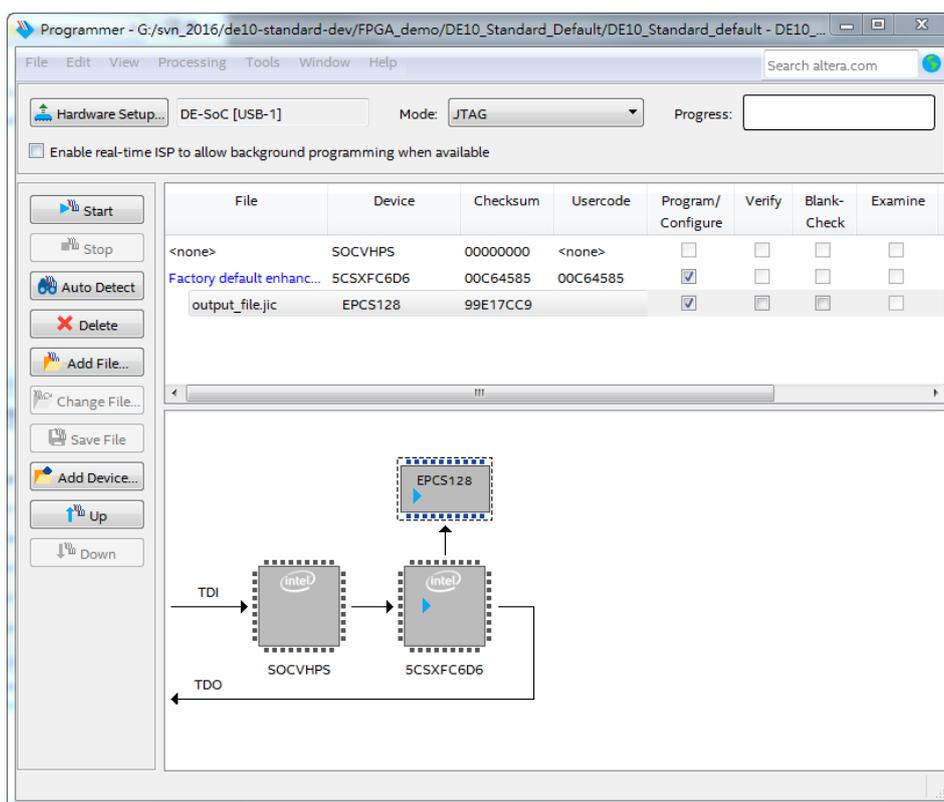


图 8-7 烧录.jic 文件的 Programmer 窗口

6、点击**Start**按钮，将.jic文件写入EPCS器件。

## 8.4 擦除 EPCS 器件

按照以下步骤可以擦除 EPCS 器件中的.jic 文件：

- 1、将开发板的SW10 MSEL[4..0]设置为10010。
- 2、从Quartus软件的Tools菜单中选择**Programmer**，弹出Programmer窗口。
- 3、点击**Auto Detect**，选择对应的FPGA器件，如图 8-6所示，检测到FPGA器件与HPS。
- 4、双击图 8-6所示的红色矩形区域，在出现的Select New Programming File 页面选择.jic 文件。

5、如图 8-8所示，勾选相应的**Erase**复选框，FPGA Device所在的File列会自动加载Factory default enhanced SFL Image。

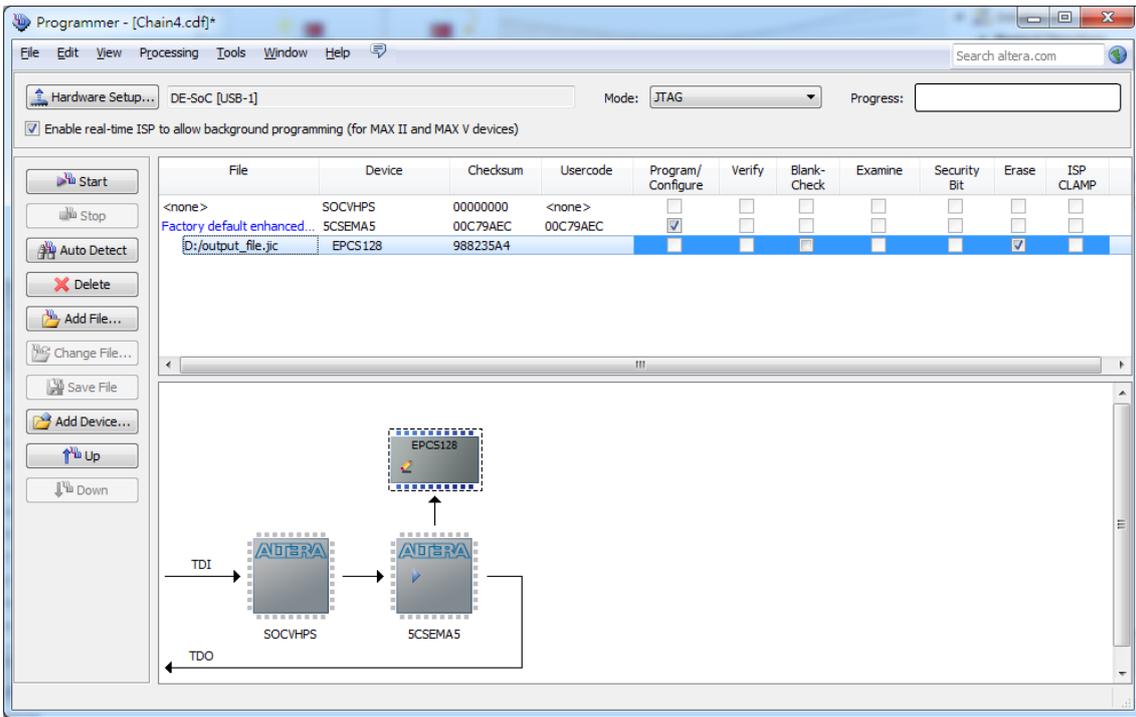


图 8-8 擦除 EPCS 器件

6、点击**Start**按钮，擦除EPCS器件中的.jic文件。

# 第九章

## Linux BSP

DE10-Standard开发板包含以下三个Linux BSP(Board Support Package), 用户可以从中选择合适的BSP来开发基于Linux的应用程序:

- Linux Console BSP
- Linux LXDE Desktop BSP
- OpenCL BSP

以上三个 Linux BSP 可以从 <http://de10-standard.terasic.com/cd> 网页上下载。

请注意, 不是所有的蓝牙/WiFi/摄像头 USB dongle 都能兼容这些 BSP。以下是一些经过我们测试验证可以兼容 BSP 并正常使用的 USB dongle, 建议使用这些 USB dongle (用户可以从友晶官网购买)。

- 蓝牙USB Dongle

Esens D704 (Terasic PN: FXX-3041-ESS)

- WiFi USB Dongle

Mi WiFi (Terasic PN: FXX-3061-MIX)

- 摄像头USB Dongle

Logitech C310

ET USB 2760 Camera

Genius WideCam F100

### 9.1 使用 Linux BSP

本节将描述在 DE10-Standard 开发板上启动 Linux 的步骤。更多详细信息请参考 DE10-Standard 系统 CD 中的 *DE10-Standard\_Getting\_Started\_Guide.pdf* 文档的第五章。

- 1、从 <http://de10-standard.terasic.com/cd> 网页上下载 BSP 镜像文件。
- 2、制作 Linux 启动卡: 用 Win32 Disk Imager 工具将镜像文件写入 micro SD 卡。
- 3、将 micro SD 卡插入 DE10-Standard 开发板的 micro SD 卡槽。

#### ■ Console (控制台) 模式

- 1、用USB线连接PC的USB接口和DE10-Standard的UART接口(J4)。

- 2、运行PC上的PuTTY软件。
- 3、PuTTY终端显示Linux启动信息。

## ■ LXDE Desktop 模式

- 1、将VGA显示器和USB鼠标/键盘连接到DE10-Standard开发板对应的接口。
- 2、接通开发板电源。
- 3、VGA显示器显示LXDE桌面系统。

## 9.2 Linux Console BSP

这是一种控制台模式的 BSP，控制台信息会显示在 PC 上运行的 UART 终端软件上。表 9-1 所示为 Linux Console BSP 的相关信息。

表 9-1 Linux Console BSP 信息

项目	描述
BSP 位置	DE10-Standard_Linux_Console.zip 下载链接: <a href="http://de10-standard.terasic.com/cd">http://de10-standard.terasic.com/cd</a>
micro SD 卡	至少 4G
MSEL[4:0]	01010
账号	用户名: root, 无需密码
UART 终端	Baud rate(波特率): 115200, Data bits(数据位宽): 8 Parity(奇偶校验): None Stop Bits(Stop 位宽): 1 Flow Control(流程控制): no
Quartus 工程	无
BSP 特点	USB 蓝牙 dongle 驱动 USB WiFi dongle 驱动 示例代码
Linux 内核源码	源码链接: <a href="https://github.com/terasic/linux-socfpga">https://github.com/terasic/linux-socfpga</a> 版本分支: socfpga-4.5 配置文件: de10_standard_console.config DTS 文件: arch/arm/boot/dts/socfpga_cyclone5_de10_standard.dts

系统启动时，Linux BSP 不会配置 FPGA。但是当系统完全启动后，用户仍然可以配置 FPGA。这时配置 FPGA 需要两个文件：FPGA 设备树 overlay(fpga.dtbo)和 FPGA 配置比特流(soc\_system.rbf)，这两个文件要存放在 Linux 文件系统的/lib/firmware 目录中。以下将描述如何生成并使用这两个文件。

## ■ 生成 FPGA 配置文件：fpga.dtbo 和 soc\_system.rbf

fpga.dtbo 是通过 Demonstration/SoC\_FPGA /DE10\_Standard\_GHRD 目录中的 fpga.dts 文件生成,文件中的主要内容是从 soc\_system.dts 文件复制而来。将文件内容中的 interrupt-parent 名称修改为 intc, firmware-name = "soc\_system.rbf" 语句指定了 FPGA 配置比特流名称。在 SoC EDS command shell 中使用以下命令创建 fpga.dtbo。

```
dtc -O dtb -o fpga.dtbo -b 0 -@fpga.dts
```

soc\_system.rbf 是通过 Demonstration/SoC\_FPGA/DE10\_Standard\_GHRD/output\_files 目录中的 sof\_to\_rbf.bat 文件生成。双击运行该 .bat 文件, DE10\_Standard\_GHRD.sof 会转换成 soc\_system.rbf 文件。

## ■ 应用 FPGA 配置文件: fpga.dtbo and soc\_system.rbf

成功生成 FPGA 的两个配置文件后,可以按照以下步骤来配置 FPGA。

1、从 <http://de10-standard.terasic.com/cd> 下载 Linux Console 镜像文件并解压缩得到 de10\_standard\_linux\_console.img

2、参考 *DE10-Standard\_Getting\_Started\_Guide.pdf* 文档,将镜像文件写到 micro SD 卡并设置 PuTTY 终端。

3、因为是配置 FPGA,所以在接通 DE10-Standard 开发板电源之前,需要将 MSEL[4:0] 设置为 01010。

4、用 root 账号在 PuTTY 终端登陆进入 Linux Console 系统。

5、将这两个文件复制到 Linux 文件系统的 /lib/firmware 目录(实际上,镜像文件已经预装了这两个文件,所以用户无需复制这两个文件)。

6、输入以下命令配置 FPGA:

```
mount -t configfs configfs /config  
mkdir /config/device-tree/overlays/test  
echo fpga.dtbo > /config/device-tree/overlays/test/path
```

7、成功执行完以上命令后,开发板上的 LED0 会闪烁。现在,用户可以在 PuTTY 终端执行相应命令控制开发板 FPGA 端的 LED,例如:

```
echo 1 > /sys/class/leds/fpga_led2/brightness
```

## 9.3 Linux LXDE Desktop BSP

这是一种带有 LXDE 桌面的 BSP。LXDE 桌面会显示在连接到 DE10-Standard 开发板的 VGA 显示器上。表 9-2 列出了 LXDE 桌面 BSP 的相关信息。该 BSP 提供了帧缓冲功能,用于 VGA 显示,帧缓冲功能在 FPGA 端实现。帧缓冲功能中,HPS 端的 DDR3 用于视频缓冲。

该 BSP 所用的 Quartus 工程在 DE10-Standard 系统 CD 的 Demonstration/SoC\_FPGA/ControlPanel/Quartus 文件夹中。该 Quartus 工程是基于 Demonstration/SoC\_FPGA/DE10\_Standard\_FB 文件夹中的 Quartus 工程创建而成。

表 9-2 Linux LXDE Desktop BSP 信息

项目	描述
BSP 位置	DE10_Standard_LXDE.zip 下载链接: <a href="http://de10-standard.terasic.com/cd">http://de10-standard.terasic.com/cd</a>
micro SD 卡	至少 4G
MSEL[4:0]	01010
账号	用户名: root, 无需密码
UART 终端	Baud rate(波特率): 115200, Data bits(数据位宽): 8 Parity(奇偶校验): None Stop Bits(Stop 位宽): 1 Flow Control(流程控制): no
Quartus 工程	Demonstration/SoC_FPGA/ControlPanel/Quartus (基于 Demonstration/SoC_FPGA/DE10_Standard_FB) Linux DTS 文件: soc_system.dts
BSP 特点	LXDE 桌面 Frame Buffer(帧缓冲) ALSA (Advanced Linux Sound Architecture, 高级 Linux 声音体系结构) Qt 库 OpenCV 库 GNU 工具链 USB 蓝牙 Dongle 驱动以及应用示例代码 USB WiFi Dongle 驱动以及应用示例代码 USB 摄像头 Dongle 驱动以及 OpenCV 示例代码 Control Panel 示例代码 (基于 Qt) 访问、控制 FPGA 和 HPS 端外设的示例代码
Linux 内核源码	源码链接: <a href="https://github.com/terasic/linux-socfpga">https://github.com/terasic/linux-socfpga</a> 内核版本分支: socfpga-4.5 配置文件: 源码链接中的 de10_standard_lxde.config DTS 文件: arch/arm/boot/dts/socfpga_cyclone5_de10_standard_lxde.dts socfpga_cyclone5_de10_standard_lxde.dts 文件内容是从 soc_system.dts 文件中复制而来

## 9.4 OpenCL BSP

这是一种支持 Intel SDK OpenCL 的控制台模式 Linux BSP, 面向 OpenCL(Open Computing Language)的 Intel SDK 允许用户将传统的 FPGA 硬件开发流程抽样化而实现更快、更高级别的软件开发流程。更多详细信息请参考 DE10-Standard 系统 CD 中的

*DE10\_Standard\_OpenCL.pdf* 文档。

表 9-3 列出了 OpenCL BSP 的相关信息。

表 9-3 OpenCL BSP 信息

项目	描述
BSP 位置	DE10-Standard_openCL_BSP.zip 下载链接: <a href="http://de10-standard.terasic.com/cd">http://de10-standard.terasic.com/cd</a>
microSD 卡	至少 4G
MSEL[4:0]	01010
账号	用户名: root, 无需密码
UART 终端	Baud rate(波特率): 115200, Data bits(数据位宽): 8 Parity(奇偶校验): None Stop Bits(Stop 位宽): 1 Flow Control(流程控制): no
Quartus 工程	OpenCL BSP 中的 de10_standard\de10_standard_sharedonly
BSP 特点	帧缓冲(Frame Buffer) OpenCL 示例代码
Linux 内核源码	源码链接: <a href="https://github.com/terasic/linux-socfpga/tree/socfpga-3.10">https://github.com/terasic/linux-socfpga/tree/socfpga-3.10</a> 内核版本分支: socfpga-3.10 配置文件: de10_standard_opencl.config

## 修订历史

版本	修改记录
V1.0	最初版本

## 版权声明

版权© 2017 友晶科技，保留所有权利。