Altera FPGA 全速漂移 开发指南

LVDS 数据收发实例

欢迎加入 FPGA/CPLD 助学小组一同学习交流：

EDN:

http://group.ednchina.com/GROUP_GRO_14596_1375.HTM

ChinaAET: http://group.chinaaet.com/273

淘宝店链接：http://myfpga.taobao.com/

技术咨询：orand_support@sina.com

特权 HSC 最新资料例程下载地址：

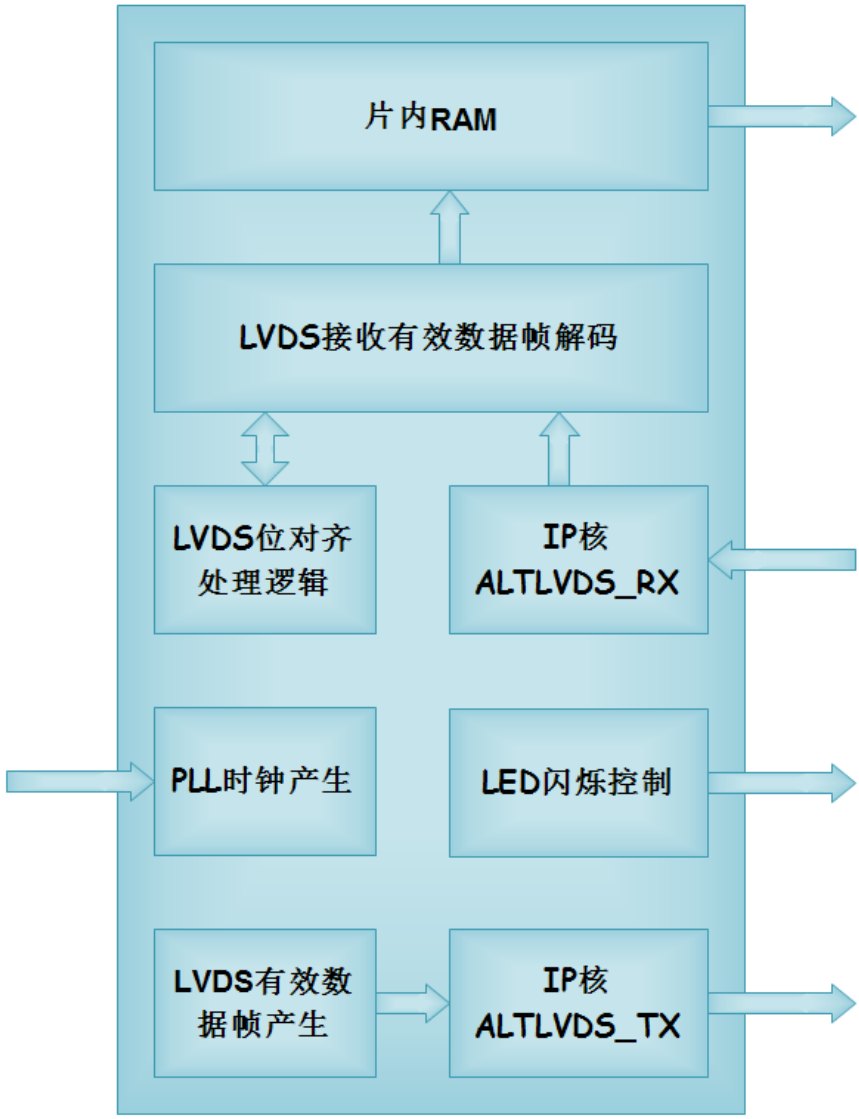
http://pan.baidu.com/s/1pLmZaFx

版本信息		
时间	版本	状态
2016-08-07	V1.00	创建。

目录

1 概述

本实例的 FPGA 功能大体如图所示，由 PLL 产生基准时钟，LED 闪烁用于指示工作运行状态；FPGA 内部产生固定的 1024 字节为单位的有效数据帧，通过 LVDS 发送出去；同时另一则，FPGA 也接收 LVDS 数据，进行位对齐处理，并且解析有效帧数据，同时将最新的有效 LVDS 接收数据帧缓存到片内 RAM 中；PC 上的 Quartus II 可以使用 In-System Memory Content Editor 观察片内 RAM 的数据。在 HSC 板上，将 FPGA 的 LVDS 发送和接收数据、时钟通道互联在一起。这样，我们这个实例就可以通过同一颗 FPGA 芯片完成收发实验。



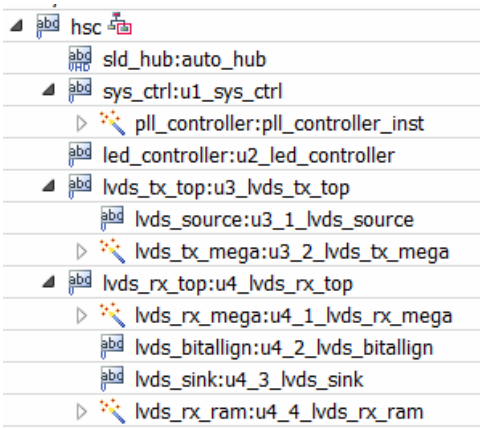
数据帧的结构如下：

N 个 Pattern 数据+4 个字节帧头+1024 字节有效数据+N 个 pattern 数据

如图所示，在 lvds_source.v 模块中定义了 pattern 数据为 0xa6，当 LVDS 数据线上不传输有效帧数据时，总是传输 pattern 数据。4 个字节帧头为固定的 0xcc+0x33+0xc3+0x3c，帧头用于判定接下来要传输有效数据。接下来 1024 个字节是真正有效的 LVDS 数据。

```
////////////////////////////////////////////////////
//pattern data，当不传输有效数据时，不停的发送该数据，用于LVDS RX端bit同步对齐
parameter PATTERN_TRAINING = 8'ha6;
```

本实例的代码层次结构如图所示。



- hsc.v 模式是顶层模块，用于各个子模块间的例化、互联，以及接口引脚的引出。
- sys_ctrl.v 模块实现系统基本的复位与时钟（PLL 例化）生成。
- led_controller.v 模块实现 LED 闪烁计数逻辑，用于指示 FPGA 的运行状态。
- lvds_tx_top.v 模块涉及 LVDS 发送数据相关逻辑，其下例化的 lvds_source.v 模块产生准备发送的 LVDS 数据源，以 1024 个字节为一个帧单位作为有效数据，定时发送；lvds_tx_mega.v 模块是 IP 核 ALTLVDS_TX 的例化。
- lvds_rx_top.v 模块涉及 LVDS 接收数据相关逻辑，其下例化的 lvds_bitalign.v 模块对接收到的 LVDS 数据进行位对齐判断和处理；lvds_rx_mega.v 模块是 IP 核 ALTLVDS_RX 的例化；lvds_sink.v 模块对接收到的 LVDS 有效数据帧进行解析；lvds_rx_ram.v 是 IP 核 RAM 的例化，用于缓存最新收到的一整个帧的 LVDS 数据。

2 IP 核 ALTLVDS_TX 创建与配置

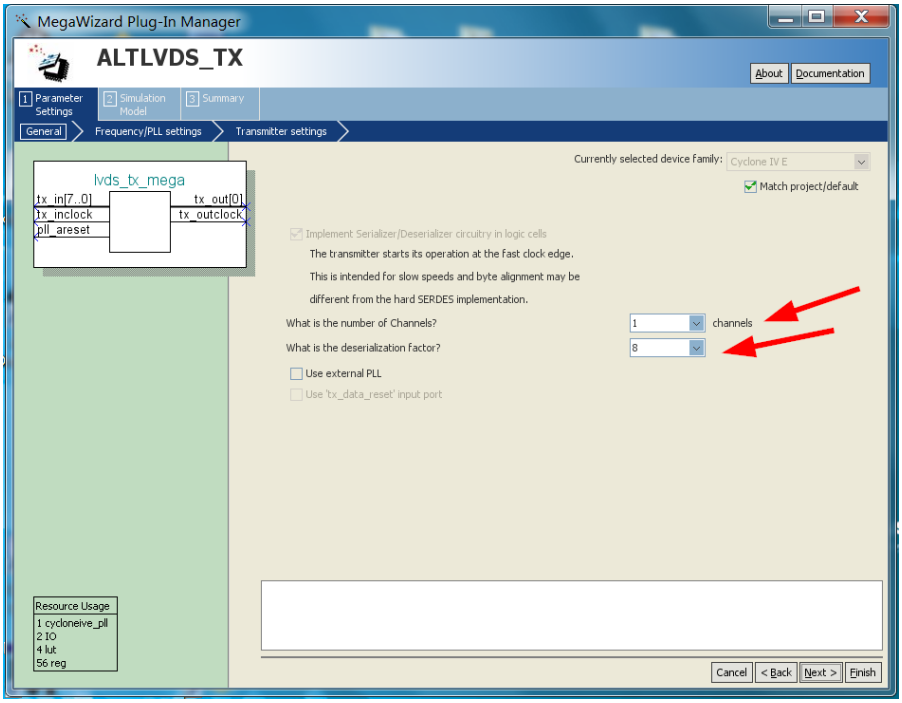
在新建的工程中，点击菜单“Tools→MegaWizard Plug-In Manager”。接着，选择“Creat a new custom megafunction variation”，然后 Next。

接着选择我们所需要的 IP 核，设置如下。

- 在“Select a megafunction from the list below”下面选择 IP 核为“I/O → ALTLVDS_TX”。
- 在“What device family will you be using”后面的下拉栏中选择我们所使用的器件系列为“Cyclone IV E”。
- 在“What type of output file do you want to create?”下面选择语言为“Verilog”。
- 在“What name do you want for the output file?”下面输入工程所在的路径，并且在最后面加上一个名称，这个名称是我们现在正在例化的 ALTLVDS_TX 模块的名称，我们可以给他起名叫 lvds_tx_mega.v，然后点击 Next 进入下一个页面。

如图所示，在第一个配置页面“Parameter Settings → General”中，设定如下。

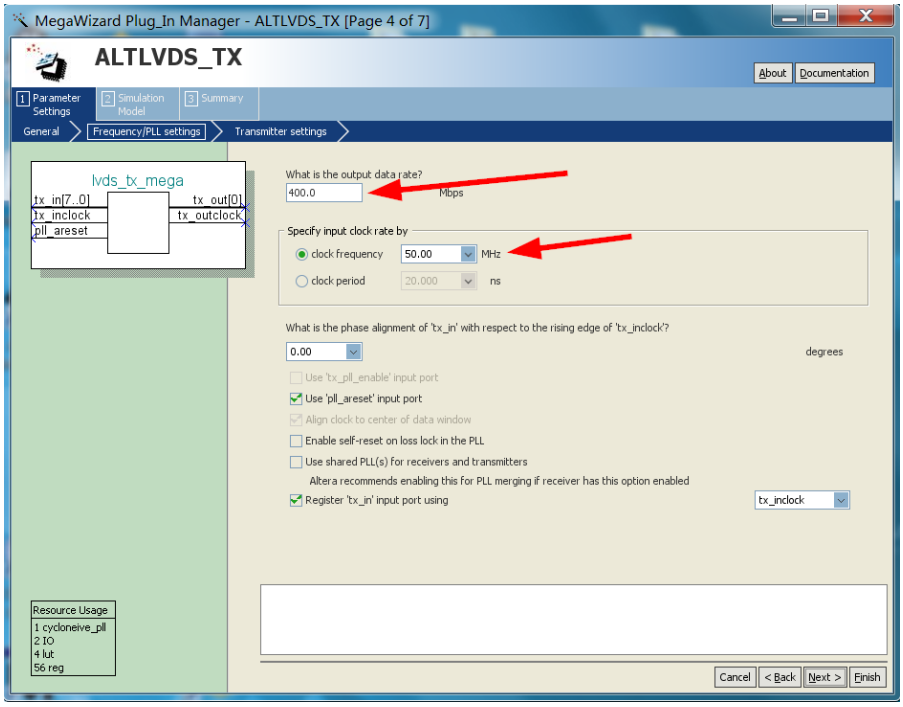
- 在“What is the number of Channels?”后面的下拉栏中选择 LVDS 的通道数为“1”channels。
- 在“What is the deserialization factor?”后面的下拉栏中选择串化因子为“8”。



如图所示，在第二个配置页面“Parameter Settings → Frequency/PLL settings”中，设定如下。

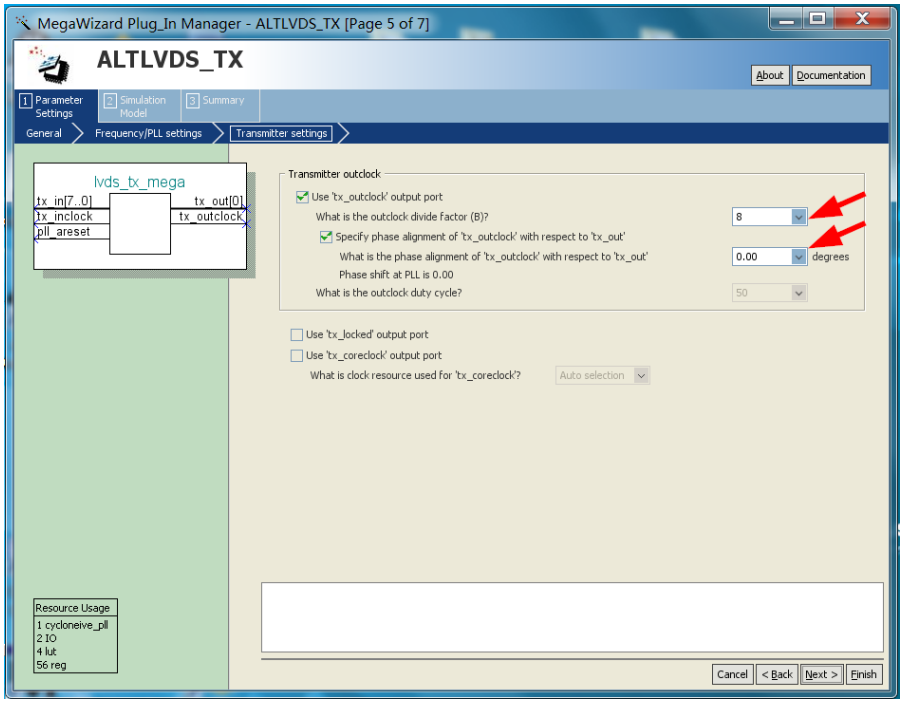
- 在“What is the output data rate?”下面输入单通道的 LVDS 数据速率为“400.0”Mbps。
- 在“Specify input clock rate by 下面的“Clock frequency”后面输入 LVDS 时钟频率为“50.00”MHz。
- 其他选项使用默认设置。

简单提下这里我们所设置的 LVDS 数据速率、LVDS 时钟频率以及上一个配置页面的串化因子之间的关系。LVDS 数据速率 = LVDS 时钟频率 * 串化因子，即 400Mbps = 8 * 50.00MHz。

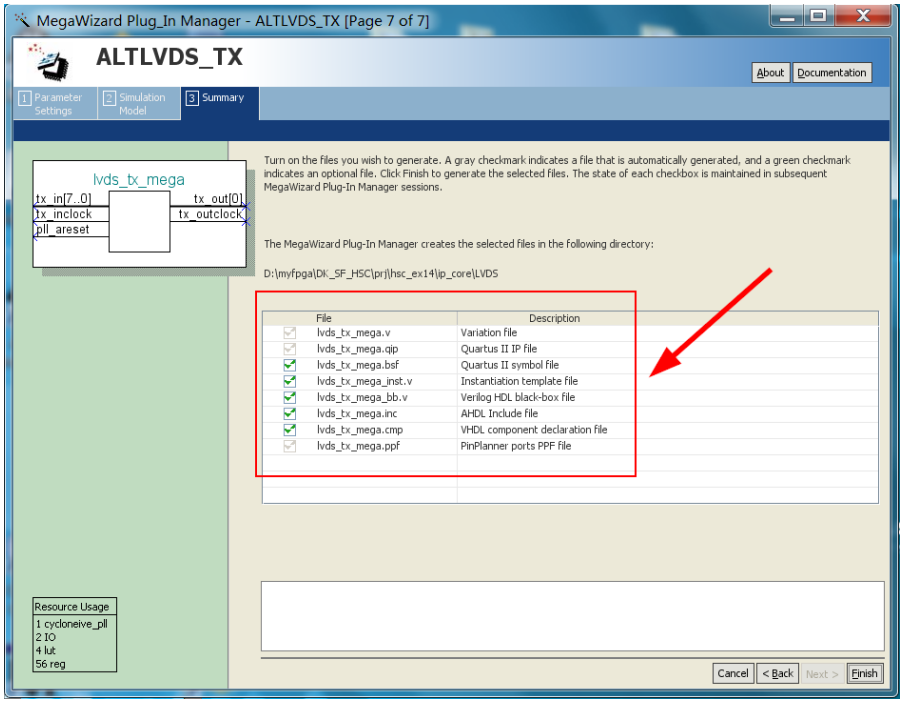


如图所示，在第三个配置页面“Parameter Settings → Transmitter settings”中，设定如下。

- 在“Transmitter outclock”下面勾选 “Use ’tx_outclock’ output port”，设置“What is the outclock divide factor (B)?”为“8”。
- 设置“Specify phase alignment ……”选项的相位偏差为“0.00” degrees。
- 其他选项使用默认设置。



配置完成后，最后在 Summary 页面，如图所示，勾选上*_inst.v 文件，这是一个 ALTLVDS 例化的模板文件，一会我们可以在工程目录下找到这个文件，然后打开它，将它的代码复制到工程中，修改对应接口即可完成这个 IP 核的集成。



ug_altlvds.pdf 文档是 Altera 对 IP 核 ALTLVDS_TX 的配套说明，关于以上配置的详细说明，大家可以参考该文档。

3 IP 核 ALTLVDS_RX 创建与配置

在新建的工程中，点击菜单“Tools→MegaWizard Plug-In Manager”。接着，选择“Creat a new custom megafunction variation”，然后 Next。

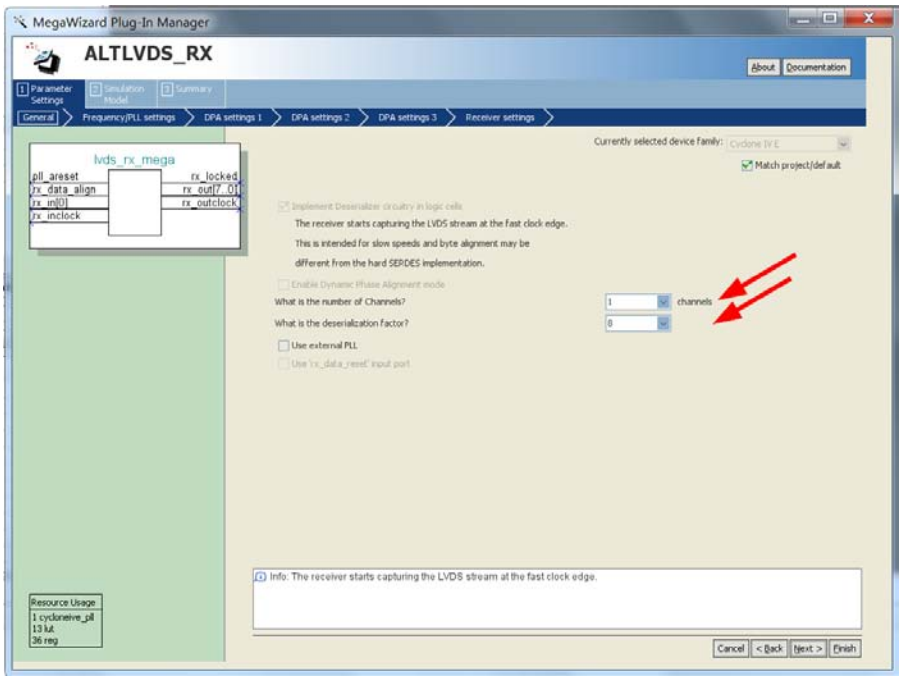
接着选择我们所需要的 IP 核，设置如下。

- 在“Select a megafunction from the list below”下面选择 IP 核为“I/O → ALTLVDS_RX”。
- 在“What device family will you be using”后面的下拉栏中选择我们所使用的器件系列为“Cyclone IV E”。

- 在“What type of output file do you want to create?”下面选择语言为“Verilog”。
- 在“What name do you want for the output file?”下面输入工程所在的路径，并且在最后面加上一个名称，这个名称是我们现在正在例化的 ALTLVDS_RX 模块的名称，我们可以给他起名叫 lvds_rx_mega.v，然后点击 Next 进入下一个页面。

如图所示，在第一个配置页面“Parameter Settings → General”中，设定如下。

- 在“What is the number of Channels?”后面的下拉栏中选择 LVDS 的通道数为“1”channels。
- 在“What is the deserialization factor?”后面的下拉栏中选择串化因子为“8”。

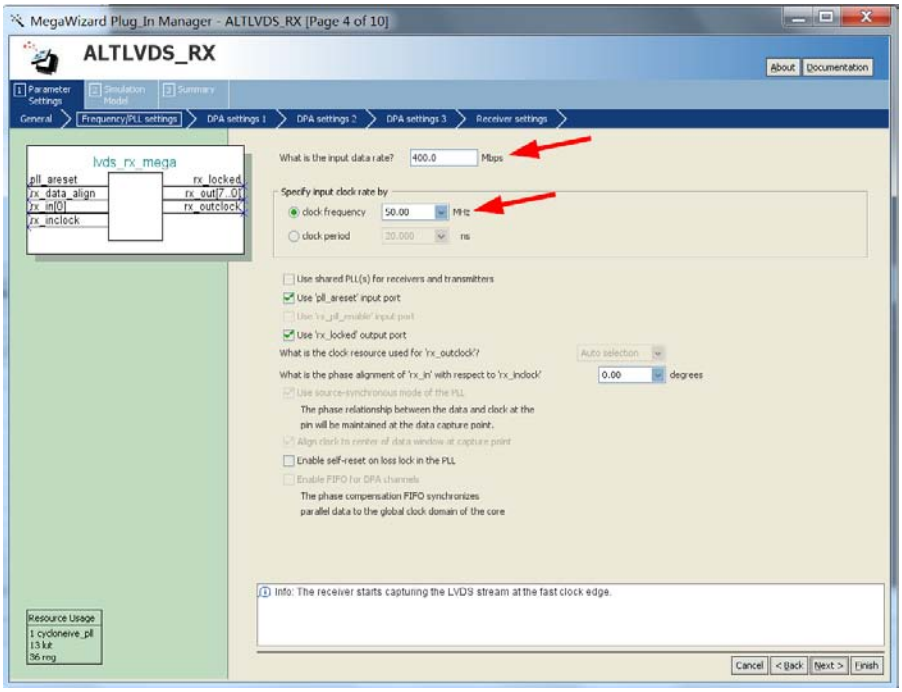


如图所示，在第二个配置页面“Parameter Settings → Frequency/PLL

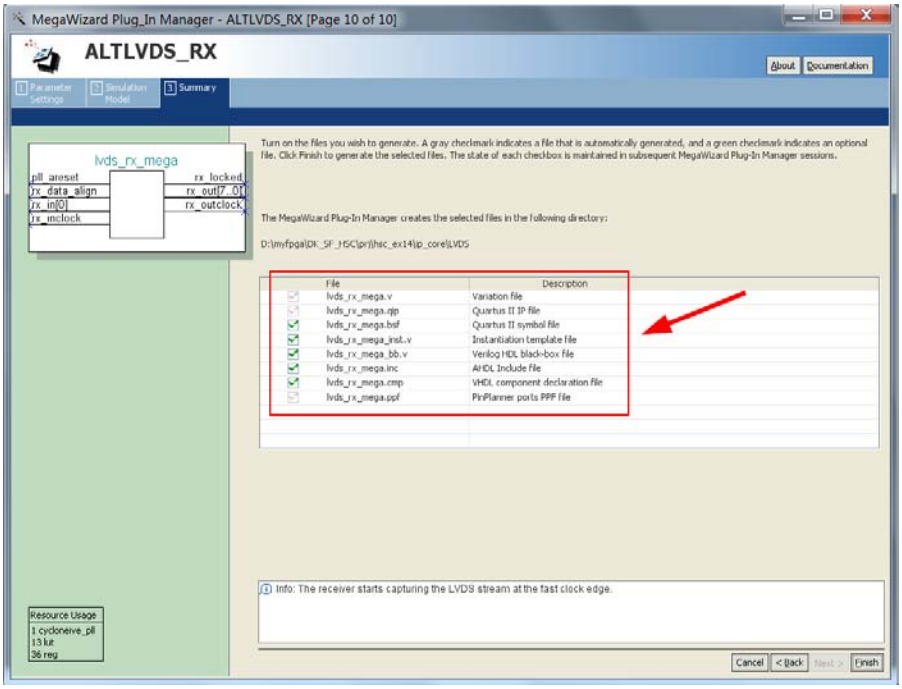
settings”中，设定如下。

- 在“What is the output data rate?”下面输入单通道的 LVDS 数据速率为“400.0”Mbps。
- 在“Specify input clock rate by 下面的“Clock frequency”后面输入 LVDS 时钟频率为“50.00”MHz。
- 其他选项使用默认设置。

这里我们所设置的 LVDS 数据速率、LVDS 时钟频率以及上一个配置页面的串化因子之间的关系。LVDS 数据速率 = LVDS 时钟频率 * 串化因子，即 400Mbps = 8 * 50.00MHz。这和例化 ALTLVDS_TX 的 IP 核是对应的。



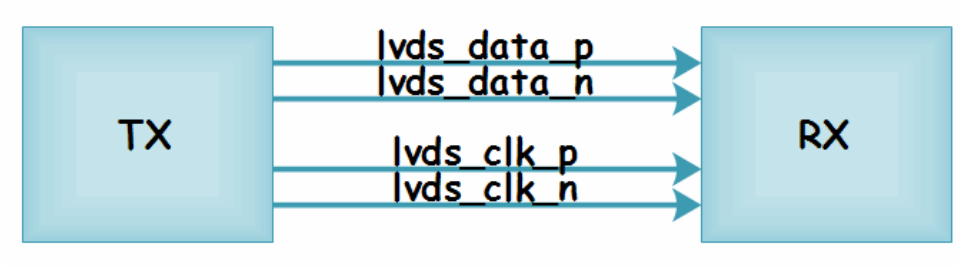
其他页面基本不需要配置，使用默认即可。最后在 Summary 页面，如图所示，勾选上*_inst.v 文件，这是一个 ALTLVDS 例化的模板文件，一会我们可以在工程目录下找到这个文件，然后打开它，将它的代码复制到工程中，修改对应接口即可完成这个 IP 核的集成。



ug_altlvds.pdf 文档是 Altera 对 IP 核 ALTLVDS_RX 的配套说明，关于以上配置的详细说明，大家可以参考该文档。

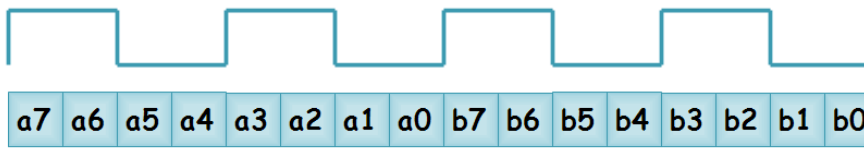
4 bit align 处理

一般情况下，LVDS 传输只有一个固定的时钟差分对和多个数据差分对。每个时钟对应的采集多个数据位的数据。例如，如图所示，只有 1 个时钟和 1 个数据的 LVDS 传输，1 个时钟周期可以传输 1bit、2bit、3bit……多个数据位，我们通常称这个时钟和数据的关系为串化因子或解串因子。



我们所使用这个 LVDS 收发实例的串化因子为 8，即一个时钟信号对应采集的数据是 8bit。

接着，那为什么需要做 bit align 处理呢。如图所示，这里假设有一个 LVDS 时钟，对应每个时钟周期传输 4bit 数据，那么实际在接收端，我们需要将这单通道的数据转换为 8bit 的数据字节。这就存在一个问题，当 LVDS 的 TX 端和 RX 端工作时，他们可以不是同时处于工作状态，可能 TX 端先于 RX 端工作，那么 RX 端开始接收数据时，所抓到的数据就不一定是图示最开始一个时钟周期对应的 a7（bit7）了，那么如果它采集到的数据时 a3 或者其他任何一个数据位，直接就把这个数据位作为一个字节的 bit7，可想而知，并不是我们所期望的，组合出来的所有数据都存在“位移”，这些数据也无法正确的解析为 TX 端的有效数据信息。



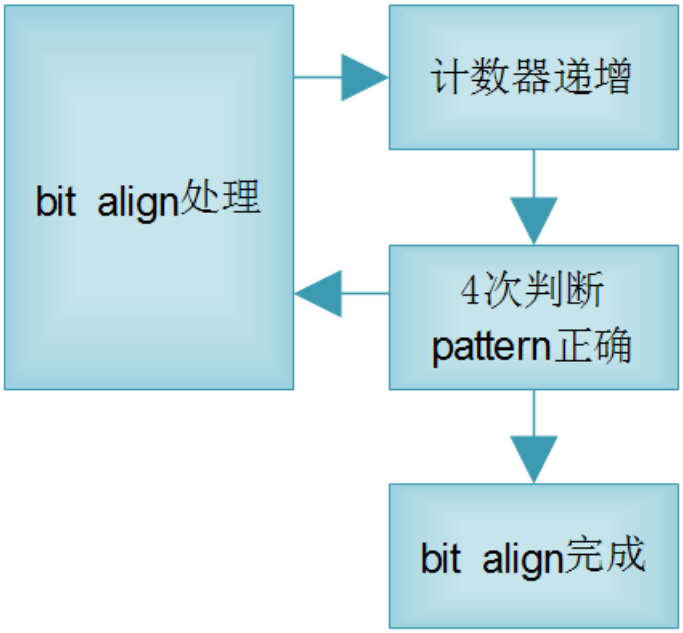
为了解决这个问题，LVDS 的 RX 解串器中提供了一个功能接口，即 bit align（或者称之为 bit slip）。每个时钟周期，LVDS 解串器都会判断这个 bit align 接口信号的状态，若为高电平，则解析出来的数据会相应的移动一位，那么在我们自己的逻辑中，必须设计一个状态机，判断当前数据的各个位是否就绪，即判断所接收到的数据是否为 RX 端和 TX 端事先预定好的数据，

若不是，则拉高一个时钟周期的 bit align 信号，再判断，直到数据接收正确。通常情况下，为了完成这个所谓的 bit align 功能，LVDS 的 TX 端和 RX 端都会约定好，在不传输有效数据时，数据通道上所传输的数据都是固定的数据，并且 TX 端和 RX 端都清楚这个固定数据值。

再回到我们的这里实例上来，每个 LVDS 时钟采样 8bit 数据，正好是 1 个字节。LVDS 的 TX 端和 RX 端约定的数据传输协议是：每秒产生一个 1028*8bit 的一帧数据，其中头 4*8bit 为固定的帧头 0xcc+0x33+0xc3+0x3c，随后 1024*8bit 为实际有效数据，在其他不传输有效数据时，固定传输 Pattern 数据，pattern 数据为固定的数据 8'ha6，在 lvds_source.v 代码中，有如下定义。

```
//////////////////////////////////////////////////
//pattern data，当不传输有效数据时，不停的发送该数据，用于LVDS RX端bit同步对齐
parameter PATTERN_TRAINING = 8'ha6;
```

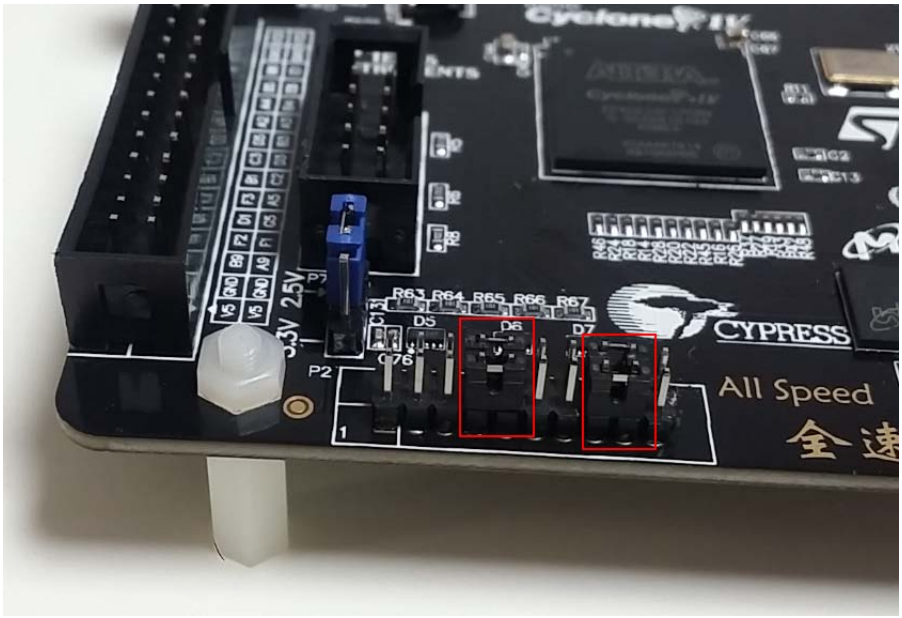
如图所示，实例工程中，我们设定一个计数器，在没有完成 bit align 时会循环递增。当计数器值为 16,17,18 和 19 时，分别判断当前字节数据是否为 pattern 值，若任何一次不是 pattern 值，则 bit align 处理一次；若判定 4 次数据都为 pttern 值，则 bit align 完成，输出标志信号，同时计数器清零，不再计数，即不再重新判定 pattern 是否正确。



在 LVDS 的接收端，我们在上电初始需要做一次上述的 bit align 处理，完成后则可以正常接收数据。

5 装配说明

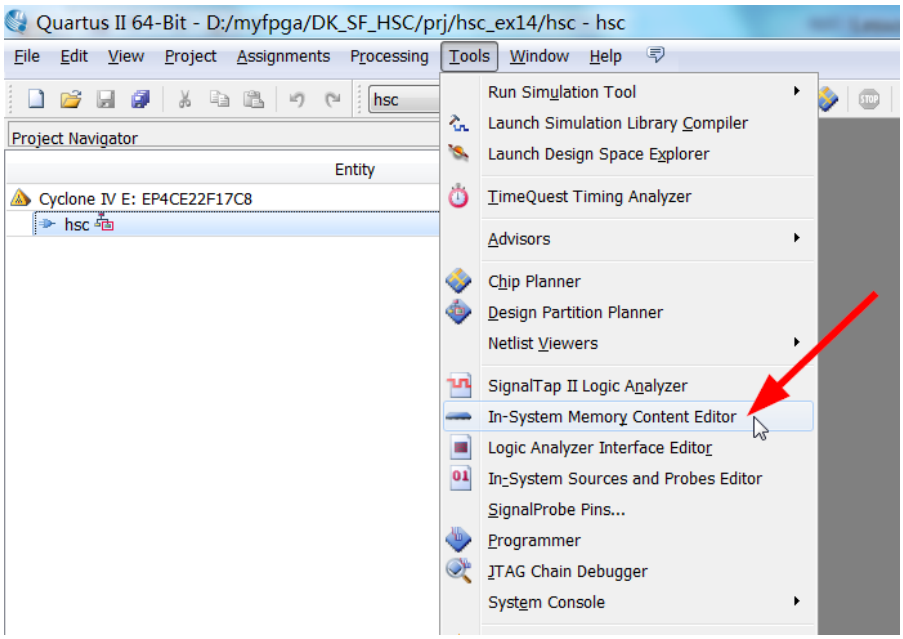
如图所示，首先需要将 P2 连接器用跳线帽短路 P2-7 和 P2-9、P2-8 和 P2-10、P2-15 和 P2-17、P2-16 和 P2-18。



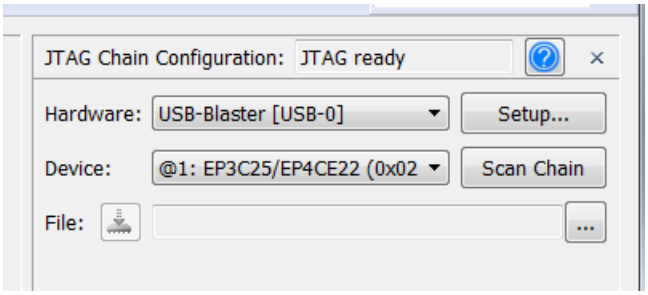
6 板级调试

给 HSC 开发板上电。

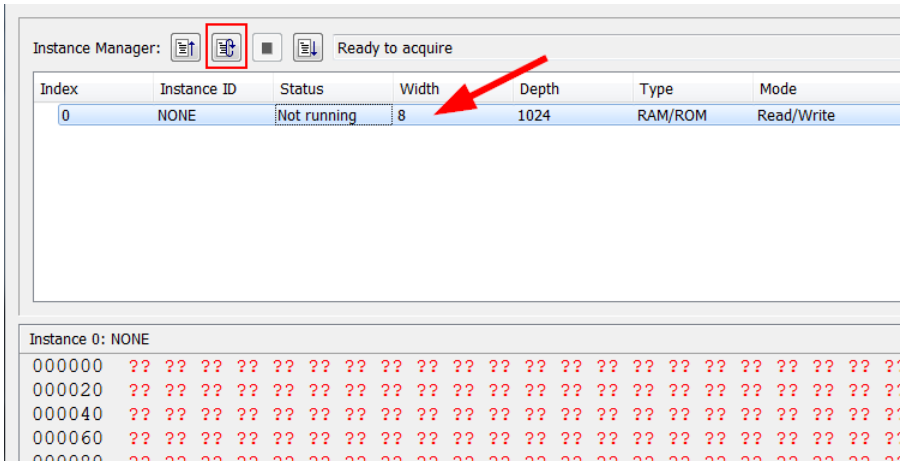
将工程“…\prj\hsc_ex14\output_files”下的 hsc.sof 下载到 HSC 开发板中，接着如图所示，点击 Quartus II 的菜单“Tools → In-System Memory Content Editor”。



在 In-System Memory Content Editor 的界面右侧，如图所示，选择 Hardware 为 USB-Blaster（点击 Setup…），接着点击 Scan Chain 确认器件连接好。



如图所示，选中当前的 RAM/ROM，然后点击 Instance Manager 后面第二个 run 按钮实时读取 FPGA 中的 RAM 数据。



随机读取到的一组数据如图所示。图示中奇数列的数据始终保持不变，而偶数列的数据将会持续递增。