Altera FPGA 全速漂移 开发指南

带 CRC 校验的 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

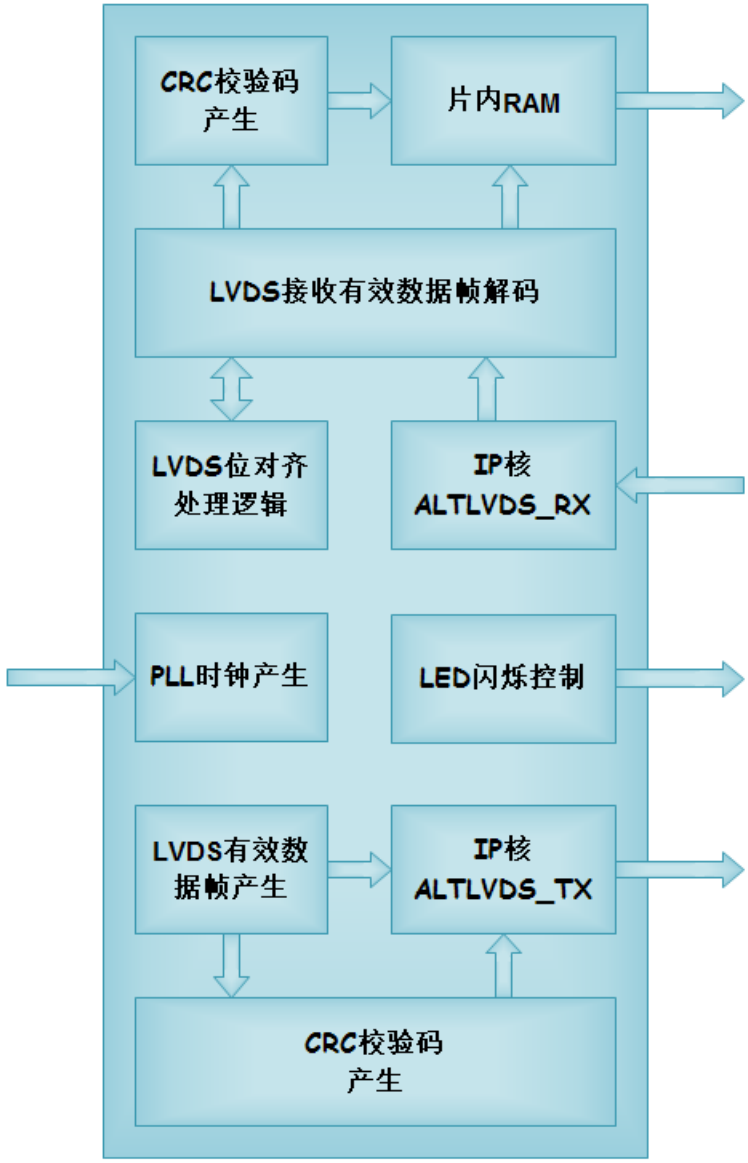
版本信息		
时间	版本	状态
2016-09-04	V1.00	创建。

上海或与电子科技

目录

1 概述

本实例的基本功能大体和 ex14 例程相当，只是在每帧数据的最后增加了一个 CRC 校验码。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，并且对应产生一个 CRC 校验码，并将它们一起通过 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，根据接收到的 1024 字节有效数据也产生一个 CRC 校验码。接收到的 1024 字节有效数据和检验码，以及接收模块自己产生的检验码，都缓存到片内 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：

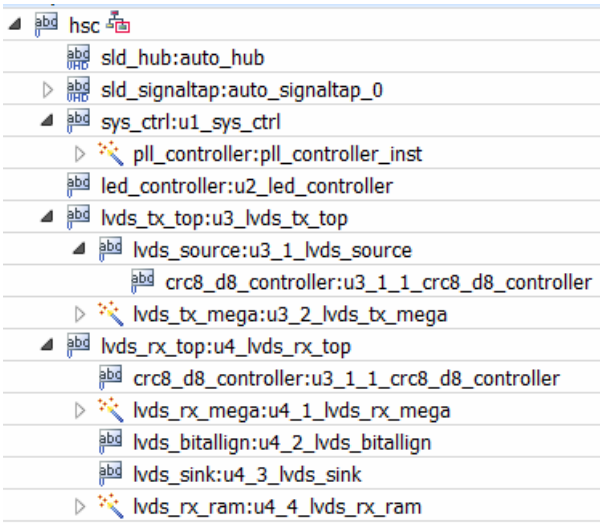
N 个 Pattern 数据+4 个字节帧头+1024 字节有效数据+1 个字节 CRC 检验码+N 个 pattern 数据

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。而 CRC 校验码则为 1024 个字节有效数据所产生的对应 CRC 校验值。

```
////////////////////////////////////////////////
//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，产生 1024 个字节有效数据，加 1 个字节 CRC 校验码，定时发送；lvds_tx_mega.v 模块是 IP 核 ALTLVDS_TX 的例化；crc8_d8_controller.v 模块根据输入的 1024 字节

数据，对应生成 8bit 的 CRC 检验码。

- 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 CRC 校验基本原理

基本原理

CRC 检验原理实际上就是在一个 p 位二进制数据序列之后附加一个 r 位二进制检验码(序列)，从而构成一个总长为 n＝p＋r 位的二进制序列；附加在数据序列之后的这个检验码与数据序列的内容之间存在着某种特定的关系。如果因干扰等原因使数据序列中的某一位或某些位发生错误，这种特定关系就会被破坏。因此，通过检查这一关系，就可以实现对数据正确性的检验。

几个基本概念

① 帧检验序列 FCS（Frame Check Sequence）：为了进行差错检验而添加的冗余码。

② 多项式模 2 运行：实际上是按位异或(Exclusive OR)运算，即相同为 0，相异为 1，也就是不考虑进位、借位的二进制加减运算。如：10011011 + 11001010 = 01010001。

③ 生成多项式（generator polynomial）：当进行 CRC 检验时，发送方与接收方需要事先约定一个除数，即生成多项式，一般记作 G（x）。生成多项式的最高位与最低位必须是 1。常用的 CRC 码的生成多项式有：

CRC8=X8+X5+X4+1

CRC-CCITT=X16+X12+X5+1

CRC16=X16+X15+X5+1

CRC12=X12+X11+X3+X2+1

CRC32=X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X1+1

每一个生成多项式都可以与一个代码相对应，如 CRC8 对应代码：100110001。

CRC 检验码的计算

设信息字段为 K 位，校验字段为 R 位，则码字长度为 N(N=K+R)。设双方事先约定了一个 R 次多项式 g(x)，则 CRC 码：

V(x)=A(x)g(x)=xRm(x)+r(x)

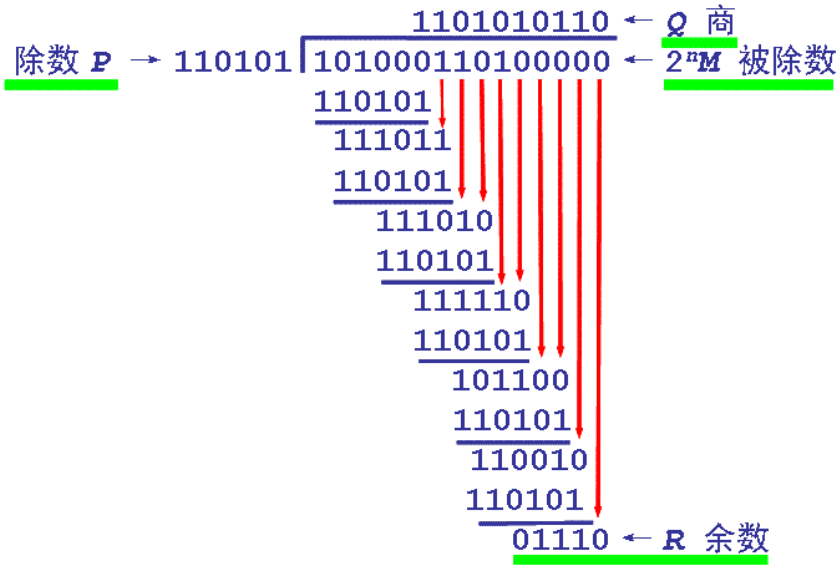
其中: m(x)为 K 次信息多项式， r(x)为 R-1 次校验多项式。

这里 r(x)对应的代码即为冗余码，加在原信息字段后即形成 CRC 码。

r(x)的计算方法为：在 K 位信息字段的后面添加 R 个 0，再除以 g(x)对应的代码序列，得到的余数即为 r(x)对应的代码(应为 R－1 位；若不足，而在高位补 0)。

计算示例

设需要发送的信息为 M = 1010001101，产生多项式对应的代码为 P = 110101，R＝5。在 M 后加 5 个 0，然后对 P 做模 2 除法运算，得余数 r(x)对应的代码：01110。故实际需要发送的数据是 101000110101110。



a. 错误检测

当接收方收到数据后，用收到的数据对 P（事先约定的）进行模 2 除法，若余数为 0，则认为数据传输无差错；若余数不为 0，则认为数据传输出现了错误，由于不知道错误发生在什么地方，因而不能进行自动纠正，一般的做法是丢弃接收的数据。

b. 几点说明：

① CRC 是一种常用的检错码，并不能用于自动纠错。

② 只要经过严格的挑选，并使用位数足够多的除数 P，那么出现检测不到的差错的概率就很小很小。

③ 仅用循环冗余检验 CRC 差错检测技术只能做到无差错接受（只是非常近似的认为是无差错的），并不能保证可靠传输。

3 CRC8 检验代码生成

在每一帧 1024Byte 的 LVDS 传输数据的末尾增加一个 8bit 传输的数据是 CRC 校验码。8bit 模式的 CRC 公式为：

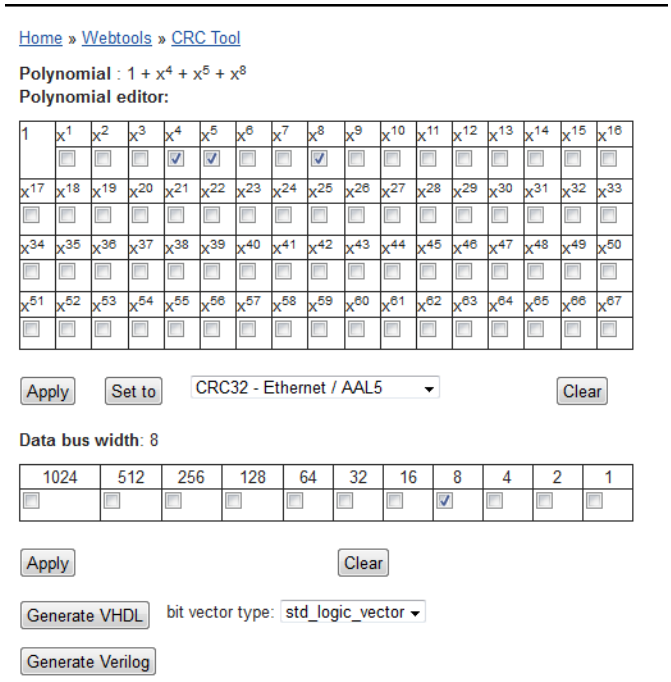
CRC8=X8+X5+X4+1

这个检验码生成逻辑 我们自己手动编写代码，我们可以 easics 公司的网页，它为我们提供了自动生成 Verilog 代码的便捷功能。该网站网址：http://www.easics.com/webtools/crctool（此页面为 easics 公司网页，不能确保长期可供使用，链接可能失效）。

如图所示，输入我们的需求，然后点击“Generate Verilog”生成代码。

Polynomial editor 设置我们需要的 CRC 多项式，点击 Apply 完成设置。

Data bus width 设定产生 CRC 码每次输入的数据位宽。一般而言，若产生 8bit 的 CRC 校验码，输入的 data bus width 都是 8bit。



产生代码如下。这段代码以 8bit 为单位不断的输入数据并计算产生 8bit 的 CRC 校验码。

```
////////////////////////////////////////////////////////////////////////////////
// Copyright (C) 1999-2008 Easics NV.
// This source file may be used and distributed without restriction
// provided that this copyright statement is not removed from the file
// and that any derivative work contains the original copyright notice
// and the associated disclaimer.
//
// THIS SOURCE FILE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS
// OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
// WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.
//
// Purpose : synthesizable CRC function
//     * polynomial: (0 4 5 8)
//     * data width: 8
//
// Info : tools@easics.be
//              http://www.easics.com
////////////////////////////////////////////////////////////////////////////////
module CRC8_D8;

  // polynomial: (0 4 5 8)
  // data width: 8
  // convention: the first serial bit is D[7]
  function [7:0] nextCRC8_D8;

    input [7:0] Data;
    input [7:0] crc;
    reg [7:0] d;
    reg [7:0] c;
    reg [7:0] newcrc;
  begin
    d = Data;
    c = crc;

    newcrc[0] = d[6] ^ d[4] ^ d[3] ^ d[0] ^ c[0] ^ c[3] ^ c[4] ^ c[6];
    newcrc[1] = d[7] ^ d[5] ^ d[4] ^ d[1] ^ c[1] ^ c[4] ^ c[5] ^ c[7];
    newcrc[2] = d[6] ^ d[5] ^ d[2] ^ c[2] ^ c[5] ^ c[6];
    newcrc[3] = d[7] ^ d[6] ^ d[3] ^ c[3] ^ c[6] ^ c[7];
    newcrc[4] = d[7] ^ d[6] ^ d[3] ^ d[0] ^ c[0] ^ c[3] ^ c[6] ^ c[7];
    newcrc[5] = d[7] ^ d[6] ^ d[3] ^ d[1] ^ d[0] ^ c[0] ^ c[1] ^ c[3] ^ c[6] ^ c[7];
```

```
    newcrc[6] = d[7] ^ d[4] ^ d[2] ^ d[1] ^ c[1] ^ c[2] ^ c[4] ^ c[7];
    newcrc[7] = d[5] ^ d[3] ^ d[2] ^ c[2] ^ c[3] ^ c[5];
    nextCRC8_D8 = newcrc;
  end
  endfunction
endmodule
```

将前面生成的 CRC 校验码逻辑封装成我们设计所需的模块。该模块一个时钟周期输入 1 个 8bit 的待校验数据，同时产生当前的 8bit 的 CRC8 校验值。

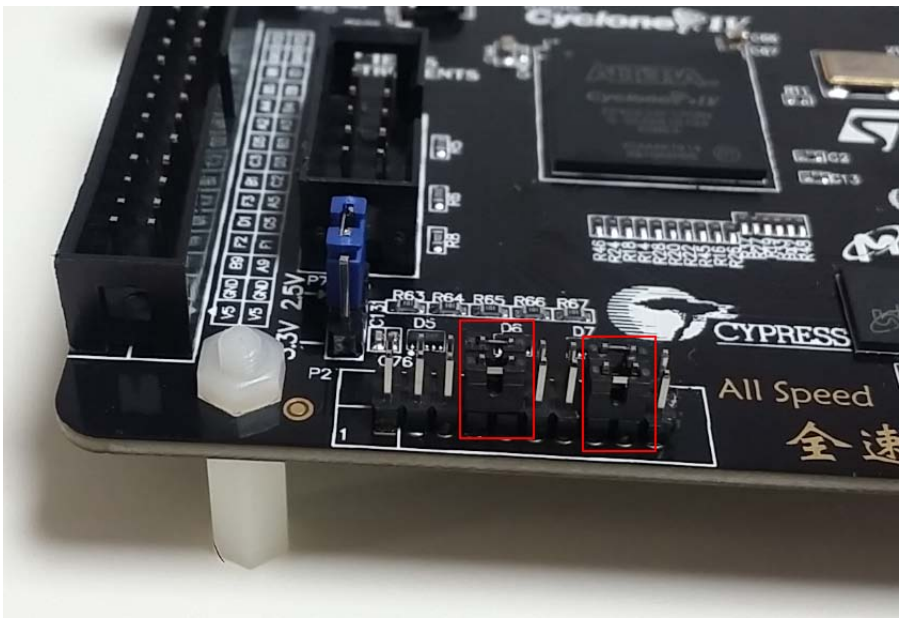
代码如下。

```
////////////////////////////////////////////////////////////////////////////////
// Copyright (C) 1999-2008 Easics NV.
// This source file may be used and distributed without restriction
// provided that this copyright statement is not removed from the file
// and that any derivative work contains the original copyright notice
// and the associated disclaimer.
//
// THIS SOURCE FILE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS
// OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
// WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR
PURPOSE.
//
// Purpose : synthesizable CRC function
//     * polynomial: (0 4 5 8)
//     * data width: 8
//
// Info : tools@easics.be
//              http://www.easics.com
////////////////////////////////////////////////////////////////////////////////
module crc8_d8_controller(
```

```
                input clk,
                input rst_n,
                input crc_in_en,     //CRC 计算使能信号
                input[7:0] crc_in,    //CRC 带校验数据
                output reg[7:0] crc_out, //CRC 校验值输出
                output crc_out_en  //CRC 最终输出值有效
            );

//////////////////////////////////////////////////////////////
//CRC 校验运算

always @(posedge clk or negedge rst_n)
      if(!rst_n) crc_out <= 8'd0;
      else if(crc_in_en) begin
            crc_out[0] <= crc_in[6] ^ crc_in[4] ^ crc_in[3] ^ crc_in[0] ^ crc_out[0]
^ crc_out[3] ^ crc_out[4] ^ crc_out[6];
            crc_out[1] <= crc_in[7] ^ crc_in[5] ^ crc_in[4] ^ crc_in[1] ^ crc_out[1]
^ crc_out[4] ^ crc_out[5] ^ crc_out[7];
            crc_out[2] <= crc_in[6] ^ crc_in[5] ^ crc_in[2] ^ crc_out[2] ^
crc_out[5] ^ crc_out[6];
            crc_out[3] <= crc_in[7] ^ crc_in[6] ^ crc_in[3] ^ crc_out[3] ^
crc_out[6] ^ crc_out[7];
            crc_out[4] <= crc_in[7] ^ crc_in[6] ^ crc_in[3] ^ crc_in[0] ^ crc_out[0]
^ crc_out[3] ^ crc_out[6] ^ crc_out[7];
            crc_out[5] <= crc_in[7] ^ crc_in[6] ^ crc_in[3] ^ crc_in[1] ^ crc_in[0] ^
crc_out[0] ^ crc_out[1] ^ crc_out[3] ^ crc_out[6] ^ crc_out[7];
            crc_out[6] <= crc_in[7] ^ crc_in[4] ^ crc_in[2] ^ crc_in[1] ^ crc_out[1]
^ crc_out[2] ^ crc_out[4] ^ crc_out[7];
            crc_out[7] <= crc_in[5] ^ crc_in[3] ^ crc_in[2] ^ crc_out[2] ^
crc_out[3] ^ crc_out[5];
      end
      else crc_out <= 8'd0;

//////////////////////////////////////////////////////////////
//CRC 最终输出值有效
reg crc_in_enr;

always @(posedge clk or negedge rst_n)
      if(!rst_n) crc_in_enr <= 1'b0;
      else crc_in_enr <= crc_in_en;

assign      crc_out_en = crc_in_enr & ~crc_in_en;

endmodule
```

4 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。



5 板级调试

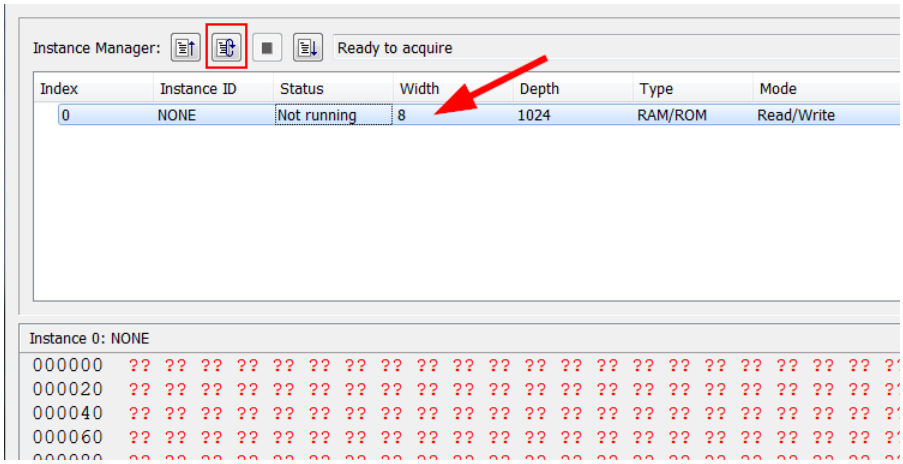
给 HSC 开发板上电。

将工程“…\prj\hsc_ex15\output_files”下的 hsc.sof 下载到 HSC 开发板中，点击 Quartus II 的菜单“Tools → In-System Memory Content Editor”。

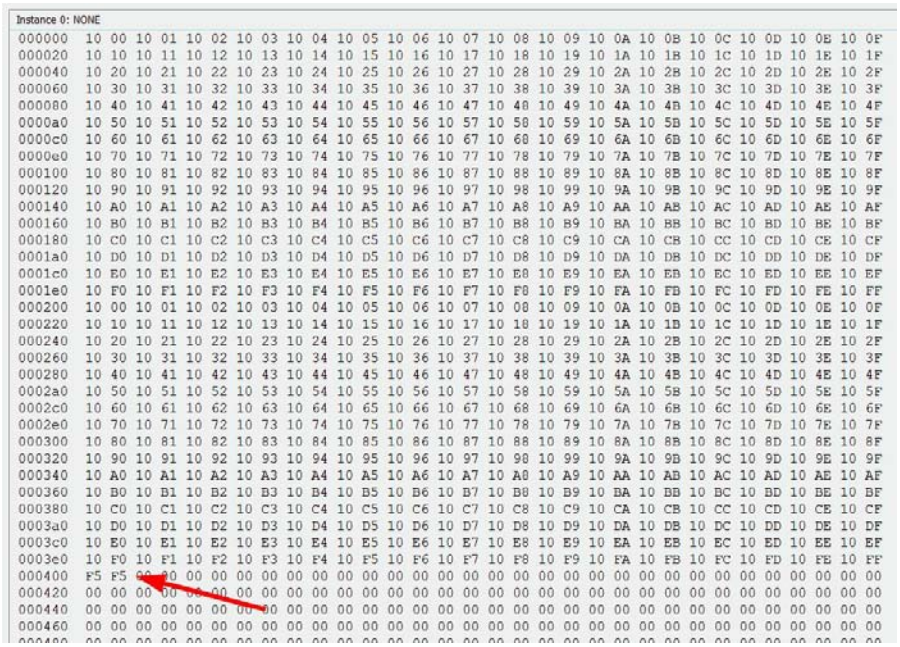
在 In-System Memory Content Editor 的界面右侧，选择 Hardware

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。同时，我们注意到，在地址 0x000400 和 0x000401，总是产生一样的数据，图示为 0xF5。这里地址 0x000400 的数据是 LVDS 发送端产生的 CRC 校验值，而地址 0x000401 的数据是 LVDS 接收端产生的 CRC 检验值，它们的结果若是一致，则表示当前接收到的 1024 字节正确无误。



此外，我们也可以查看 SignalTap II 中抓到的数据收发实时波形。注意图示箭头分别指出了LVDS发送端和接收端根据有效数据所产生的CRC检验值数据，完全一致。