Uart boot 实现说明

智能驱动组 王彬 2017.1.12

# 1:实现概述

通过uart boot的方式，可以在CM408没有程序的情况下，先通过串口将能够在ram中运行的ldr文件发送给boot rom，boot rom 接收到该ldr文件后，将控制权交给ldr文件中的代码。该代码将在ram中运行起来。因为该代码中包含flash烧写等功能，所以后续可以通过调试口进行烧写固件等操作。

## 1.1：通信方式

PC通过winpcap，使用PCdebug或者是等环网等通信格式和CM408进行通信，不管是pcdebug或者是等环网，物理通道部分都是在FPGA中实现，FPGA与CM408之间通过虚拟的uart和CM408的uart口进行通信。PC通过操作该虚拟的uart口进行ldr数据的发送的接收。PC会设置波特率，通信格式等，然后PC通过CM408具有的自动波特率的功能和CM408建立uart通信。

## 1.2：工程设置

开发使用的是IAR 7.40版本。工程设置有部分地方需要更改：因为该ldr中的代码是运行在ram中，不能访问flash的地址。

(1)：c/c++ complier 的preprocessor中定义下面的宏定义

ADI\_DEBUG

DO\_NOT\_RELOCATE\_IVT

ADI\_NO\_SECURITY\_PROTOCOL

(2)：linker中icf文件选择如下，该文件中会将vectortable放到ram部分，并没有flash部分的定义

$TOOLKIT\_DIR$\config\linker\AnalogDevices\CM40z - NoFlash.icf

(3)：debugger中setup中driver选为Simulator

不勾选Use Macro file

## 1.3:startup文件

修改startup文件如下，因为生成该ldr时定义了ADI\_NO\_SECURITY\_PROTOCOL，所以

根据这个宏定义不同将中断向量表放到不同的位置。

如果需要安全密码，那么放到flash中，如果没有，放到.intvec.这里我们不需要密码，所以放到ram中

// Locate IVT depending on security model

#ifndef ADI\_NO\_SECURITY\_PROTOCOL

/\* For SECURE PARTS: locate IVT at absolute address macro defined in adi\_ecc.h security header \*/

#pragma location = IVT\_ADDRESS

#else

/\* For NON-SECURE PARTS: locate IVT at start of ".intvec" section \*/

#pragma location = ".intvec"

#endif /\* not ADI\_NO\_SECURITY\_PROTOCOL \*/

## 1.4：elf2ldr

IAR生成的.out文件为elf格式，为了生成loader file(ldr),需要进行格式转化，转化后的文件需要四字节对齐，并且需要制定vectorTable的地址。因为boot ram将ldr文件加载后会将控制权交出给ldr中的代码，而ldr的第一个block中存放的就是控制权交出后执行的第一行代码。因此该block存放的应该是中断向量表，通过查询编译完的IAR程序的map文件可知，vectortable的地址是0x10000000.

## 1.5：硬件设置

上电时需要将boot开关拨到uart boot一边。Boot成功后，烧写正常运行的代码后，不能直接使用复位功能，因为这时boot开关还在uart boot一边，复位时不会从flash中将新的代码导入，而只是复位寄存器，将程序指针重新指向中断向量表，因为该中断向量表还是在ram中，并且boot方式还是uartboot，串口在boot成功后已经关闭。所以复位后将没有代码再运行在ram中。

正确的方式是将开关拨到flash启动一边，重启，这时代码已经是从flash加载，可以使用复位功能。因为可以重新从flash进行加载程序。

## 1.6：boot成功

Boot成功后，ram中就会运行该ldr中的代码，因为该代码中包含了接收PC指令的代码，并可以进行flash的操作。因此如果该ldr是完整的工程，不断电的情况下将支持所有的功能。如果该工程只是为了烧写程序。那么只需要支持flash读写功能和接收PC指令的功能即可。

## 1.6：总结

经过以上步骤，可以不经过仿真器，第一次烧写CM408，但是要确定FPGA固件是正确的。因为PC和CM408之间的通信交互是通过FPGA的。

当烧成砖头的时候，也可以恢复。