华北电力大学
NORTH CHINA ELECTRIC POWER UNIVERSITY

第18章 套上壳子上路

——实现SoC和FPGA原型

计算机科学与技术
计算1702班
国博凯 张博然 史元钧

目录

CONTENTS

华北电力大学

NORTH CHINA ELECTRIC POWER UNIVERSITY

1.1 SoC简介

SoC的作用

01 什么是SoC

单片系统（片上系统）System on a Chip，意思是将一个电脑或者其他电子系统集成到单一芯片的集成电路。单片系统常常应用在嵌入式系统中。

国内外学术界一般将SoC定义为：集成微处理器、模拟IP核、数字IP核和存储器（片外存储控制接口）的单一芯片。

02 SoC的作用

与PC系统不同的是，嵌入式SoC芯片往往将整个系统集成在一个硅片上，包括了存储控制器，图形处理器，通信接口等。

如果说微处理器是大脑，那么SoC就是就是包括大脑、眼睛等的一个完整的微型系统。

我们已经实现了汽车的发动机（处理器核），下面我们来给汽车套个外壳吧~

2.1 Freedom E310 SoC简介

01、Freedom E310 系列

- 由SiFive公司推出
- Freedom Everywhere 是一款可配置的RISC-V SoC 家族系列，主要面向低功耗的嵌入式MCU领域。
- Freedom 310 SoC 基于Rocket Core（支持RISC-V指令集，实现五级流水的处理器核）
- Freedom E310 SoC 是Freedom Everywhere SoC家族系列中的第一款SoC平台

02、Freedom 310 SoC 结构

架构配置为RV32IMAC架构

- 16KB的指令Cache
- 16KB的数据SRAM
- 硬件乘除法器
- 调试模块
- 丰富的外设
- etc

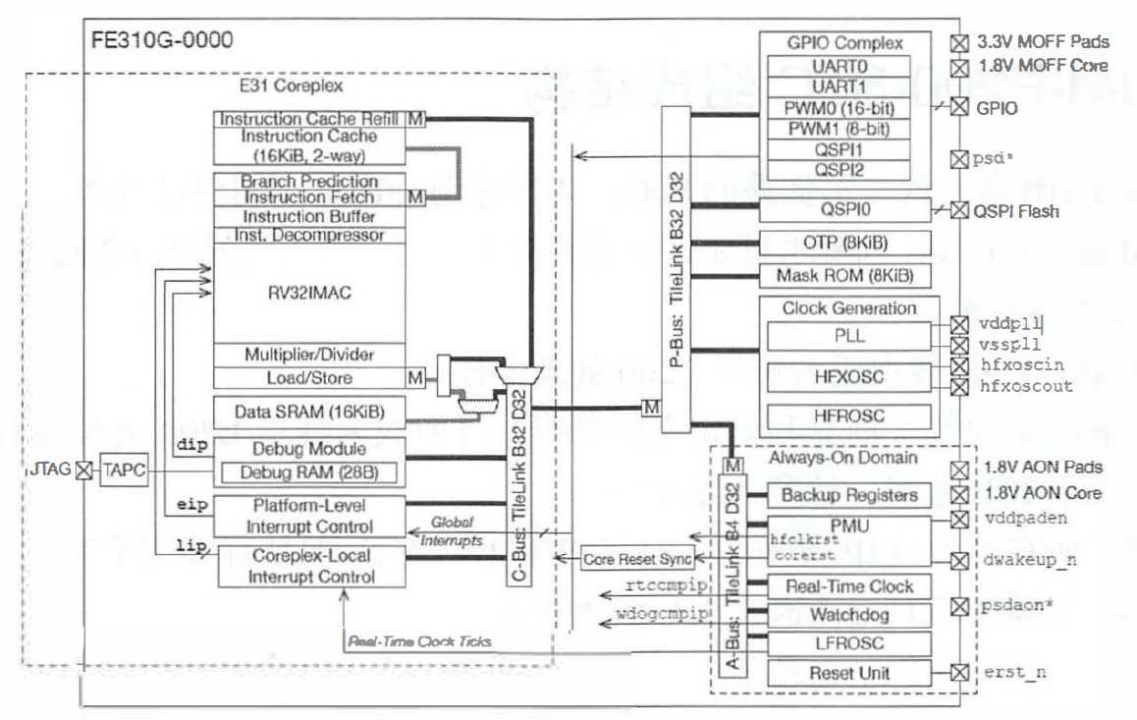


图1. Freedom 310 SoC 结构图

2.2 E310流片应用

- 仅有银行卡大小的硬件开发板HiFivel
- 能运行到320MHz以上的主频
- 目前市场上最早在售的RISC-V硬件开发板
- 集成了128Mb的Flash、RISC-V SoC、LED灯、高频晶振，低频振荡器、USB以及排针序列等等部件。

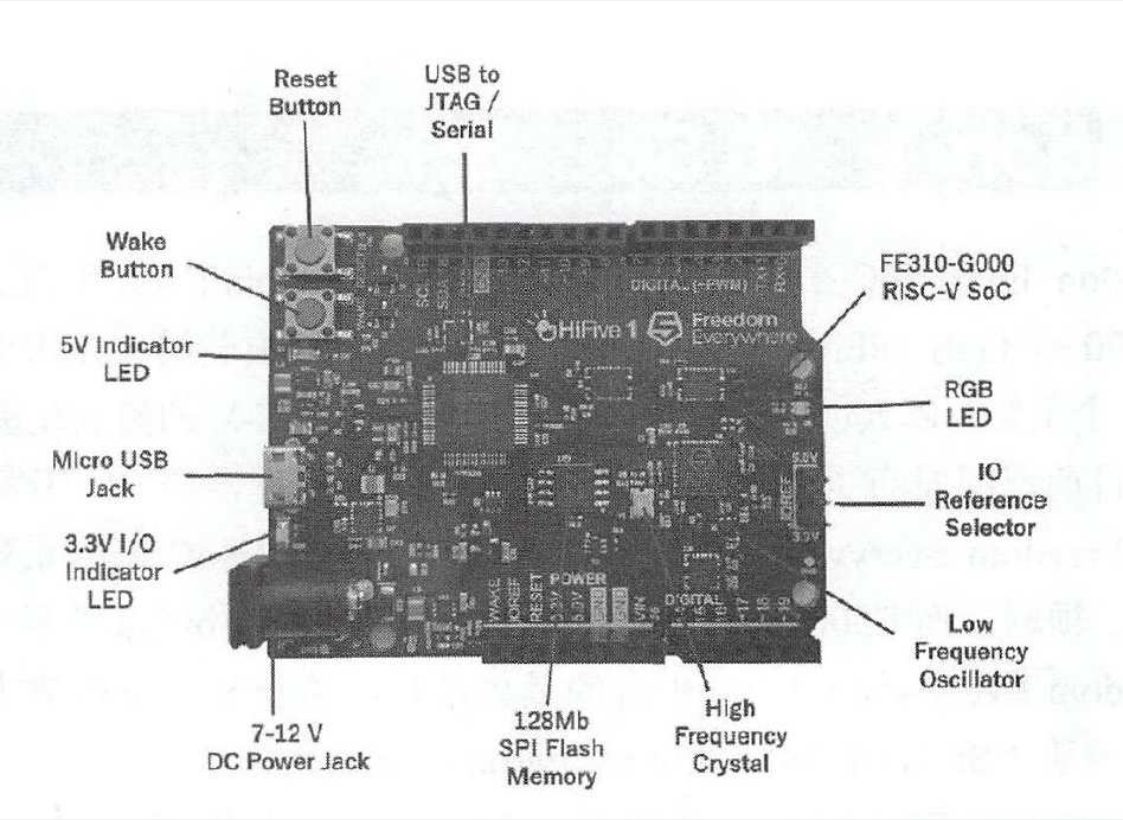


图2. HiFivel 开发板结构图

3.1 Hbird-E200-SoC 简介

由于Hbird-E200-SoC与原先的E310的地址分配表是一致的，因此从软件角度看与原始Freedom E310几乎完全兼容！
以Freedom E300 SoC为蓝本，主要的修改：

- 将Rocket Core 替换为蜂鸟E200处理器核
- 将TileLink总线替换为蜂鸟E200处理器使用的ICB总线
- 保留了所有的外设IP（UART、SPI和PWM等）但是直接使用其可综合的Verilog代码，无需使用Chisel语言进行编译转换

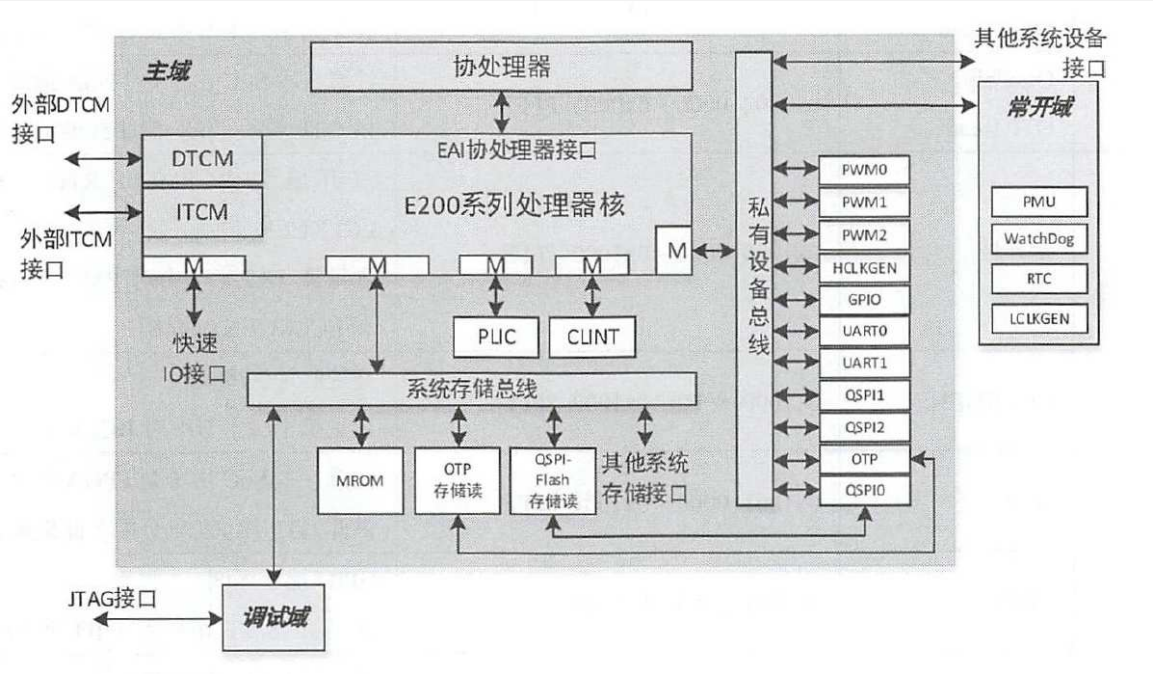


图3. HBird-E200-SoC结构图

3.2 Hbird-E200-SoC 地址分配

表 18-1 HBird-E200-SoC 地址分配表

总线分组	组件	地址区间	描述
处理器核直属	CLINT	0x0200_0000～0x0200_FFFF	CLINT（Core Local Interrupt Controller）模块寄存器地址区间 详见第 18.2.1 节中对 CLINT 的介绍
	PLIC	0x0C00_0000～0x0CFF_FFFF	PLIC（Platform Level Interrupt Controller）模块寄存器地址区间 详见第 18.2.1 节中对 PLIC 的介绍
	ITCM	0x8000_0000～ 取决于 ITCM 配置大小	ITCM 地址区间
	DTCM	0x9000_0000～ 取决于 DTCM 配置大小	DTCM 地址区间

图4. Hbird-E200-SoC地址分配表

3.2 Hbird-E200-SoC 地址分配

总线分组	组件	地址区间	描述
系统存储总线接口	调试模块	0x0000_0000 ~ 0x0000_0FFF	注意：调试模块主要用于调试器使用，普通软件程序不应该使用此区间
	Mask-ROM	0x0000_1000 ~ 0x0000_1FFF	注意：开源的 E200 项目由于是 FPGA 原型，因此 Mask-ROM 代码为一行为模型
	Off-Chip QSPI0 Flash Read	0x2000_0000 ~ 0x3FFF_FFFF	外部 SPI Flash 只读地址区间 详见第 18.2.1 节中对 QSPI Flash 的介绍
	On-Chip OTP Read	0x0002_0000 ~ 0x0003_FFFF	注意：本 SoC 由于是 FPGA 原型，并未实际提供 OTP 模块，仅分配了此地址且连接一空模块

图5. Hbird-E200-SoC地址分配表

3.2 Hbird-E200-SoC 地址分配

私有外设总线接口	常开域	0x1000_0000～0x1000_7FFF	常开域包含 PMU、RTC、WatchDog 和 LCLKGEN 详见第 18.2.1 节中对 PMU、RTC、WatchDog 和 LCLKGEN 的介绍
	HCLKGEN	0x1000_8000～0x1000_8FFF	高速时钟生成模块 详见第 18.2.1 节中对 HCLKGEN 的介绍
	OTP	0x1001_0000～0x1001_0FFF	注意：本 SoC 由于是 FPGA 原型，并未实际提供此 OTP 模块，仅分配了此地址连接一空模块
	GPIO	0x1001_2000～0x1001_2FFF	GPIO 地址区间 详见第 18.2.1 节中对 GPIO 的介绍
	UART0	0x1001_3000～0x1001_3FFF	第一个 UART 模块地址区间 详见第 18.2.1 节中对 UART 的介绍
	QSPI0	0x1001_4000～0x1001_4FFF	第一个 QSPI 模块地址区间 详见第 18.2.1 节中对 QSPI 的介绍
	PWM0	0x1001_5000～0x1001_5FFF	第一个 PWM 模块地址区间 详见第 18.2.1 节中对 PWM 的介绍
	UART1	0x1002_3000～0x1002_3FFF	第二个 UART 模块地址区间
	QSPI1	0x1002_4000～0x1002_4FFF	第二个 QSPI 模块地址区间
	PWM1	0x1002_5000～0x1002_5FFF	第二个 PWM 模块地址区间
	QSPI2	0x1003_4000～0x1003_4FFF	第三个 QSPI 模块地址区间
	PWM2	0x1003_5000～0x1003_5FFF	第三个 PWM 模块地址区间
其他地址区间	上表中未使用到的地址区间，则均为写忽略，读返回 0		

图6. Hbird-E200-SoC地址分配表

3.3 Hbird-E200-SoC 模块简述

HCLKGEN（High-Speed Clock Generation）

> 该模块主要为主 体域生成高速时钟。 高速时钟来自于外部 16MHz 晶振产生或者内部的 PLL 产生的高频时钟。

CLINT（Core-Local Interrupt Controller）

> 主要实现 RISC-V 架构手册中规定的标准计时器 (Timer) 和软件中断功能。

JTAG

> 标准（1149.1）JTAG 连接模块用千连接系统外部调试器 (Debugger) 与内部的调试模块 (Debug Module)。

PLIC（Platform-Level Interrupt Controller）

> 主要实现 RISC-V 架构手册中规定的 PLIC 功能。该 PLIC 能够支待多个中断源，并且每个中断可以配置中断优先级，所有中断源经过 PLIC 仲裁后，生成一根终的中断信号通给处理器核作为其外部中断信号。在本 SoC 中，PLIC 的中断来源包括 UART、SPI 和 GPIO 等。

3.3 Hbird-E200-SoC 模块简述

调试模块

调试模块，用千支持外部 JTAG 通过该模块调试处理器核，使得处理器核能够通过 GDB对其进行交互式调试， 譬如设置断点、 单步执行等调试功能。

Quad-SPI Flash

专用于连接外部 Flash 的 Quad-SPI (QSPI) 接口。该 QSPI 接口可以被软件配置成为 execute-In-Place 模式，在此模式下，Flash 可以被一当作 段只读区间直接被存储器读取。

GPIO（General Purpose I/O)

用于提供一组32 I/O 的通用输入输出接口，每个I/O可被软件配置为输入或者输出，如果是输出，则可以设置具体的输出值。
每个I/O还可以被配置为IOF (Hardware I/O Functions), 也就是将I/O供SoC内部的其他模块复用，譬如SPI、UART和PWM等。
另外，每个GPIO的I/O均作为一个中断源被连接到PLIC的中断源上。

3.3 Hbird-E200-SoC 模块简述

QSPI

除了上述专用于 Flash的QSPI接口之外，SoC 还有两个独立的QSPI接口控制器。一个QSPI使用4个片选信号(Chip Selects), 1个QSPI使用1个片选信号。 两个QSPI均使用GPIO的IOF功能与外界通信。

UART（ Universal Asynchronous Receiver-Transmitte

SoC有两个独立的UART, 两个UART均使用GPIO的IOF功能与外界通信。

PWM（ Pulse-Width Modulator ）

SoC有3个独立的PWM , 其中 PWM1和PWM2是16比特宽的精度， 另一个PWM0是8比特宽的精度。3个PWM均使用GPIO的IOF功能与外界通信。

3.3 Hbird-E200-SoC 常开域简述

LCLKGEN

全称为低速时钟生成(Low-Speed Clock Generation)。 该模块主要为常开域生成低速时钟，使用32.768KHz的低速实时时钟，该时钟来自于外部晶振产生。

RTC

全称为实时计数器(Real-Time Counter)。该计数器位于常开域中， 因此使用低速时钟进行计数，并且还能产生中断。

Watch Dog

全称为看门狗计数器(Watch Dog Timer)。该计数器位于常开域中， 因此使用低速时钟进行计数，并且可以通过配置其计数的目标值产生中断。

PMU

全称为电源管理单元(Power Management Unit), 用于控制SoC的电源管理。整个SoC除了WatchDog、 RTC和PMU等模块处于常开域之外，其他主域可以在PMU的控制下被置千断电状态以节省功耗，或者重新唤醒等。

3.4 HBird-E200-SoC 代码结构

```
e200_opensource
    |----rtl                    // 存放 RTL 的目录
        |----e203               // E203 核和 SoC 的 RTL 目录

            |----general        // 存放一些公用的通用 RTL 代码
            |----core           // 存放 e203 Core 的 RTL 代码
            |----fab            // 存放总线结构（bus fabric）的 RTL 代码
                |---- sirv_icb1to4_bus.v // 将 1 组 ICB 总线转换成 4 路 ICB 总线
                |---- sirv_icb1to8_bus.v // 将 1 组 ICB 总线转换成 8 路 ICB 总线
                |---- sirv_icb1to16_bus.v // 将 1 组 ICB 总线转换成 16 路 ICB 总线
            |----subsys         // 存放完整子系统顶层的 RTL 代码
                |---- e203_subsys_top.v      // 子系统的顶层
                |---- e203_subsys_main.v     // 子系统的主体部分（可关电）顶层
                |---- e203_subsys_plic.v     // PLIC 顶层
                |---- e203_subsys_clint.v    // CLINT 顶层
                |---- e203_subsys_mems.v     // 子系统的存储部分顶层
                |---- e203_subsys_perips.v   // 子系统的外设部分顶层
            |----mems             // 存放存储器模块的 RTL 代码
            |----perips           // 存放外设（peripherals）模块的 RTL 代码
                |---- sirv_aon_*.v           //常开域部分模块
                |---- sirv_clint*.v          // CLINT 的模块
                |---- sirv_flash_qspi*.v     // Flash 专用的 QSPI 模块
                |---- sirv_gpio*.v           // GPIO 的模块
                |---- sirv_plic*.v           // PLIC 的模块
                |---- sirv_pmu*.v            // PMU 的模块
                |---- sirv_pwm16*.v          // 16bits 精度的 PWM 模块
                |---- sirv_pwm8*.v           // 8bits 精度的 PWM 模块
                |---- sirv_qspi_1cs*.v       // 1 个 CS 选通的 QSPI 模块
                |---- sirv_qspi_4cs*.v       // 4 个 CS 选通的 QSPI 模块
                |---- sirv_qspi*.v           // 其他 QSPI 子模块
                |---- sirv_rtc*.v            // RTC 模块
                |---- sirv_uart*.v           // UART 模块
                |---- sirv_wdog*.v           // WatchDog 模块
            |----debug          // 存放调试相关模块的 RTL 代码
            |----soc            // 存放 soc 顶层的 RTL 代码
                |----e203_soc_top.v          // SoC 顶层
```

图7. HBird-E200-SoC 的代码结构如图所示。

4.1.1 FPGA 开发板

FPGA 开发板使用 Xilinx Artix-7 35T Arty FPGA Evaluation Kit 开发板。该开发板是款低成本的入门级 Xilinx FPGA 开发板，特别适用于电路设计规模不大的 FPGA 爱好者。其主要特性如下：

- 使用Xilinx Artix-7 35T FPGA
- FPGA包含33280个逻辑单元，5200个Slices，每个Slice包含4个6输入的LUT和8个寄存器。
- FPGA包含1800Kbits的快速Block RAM。

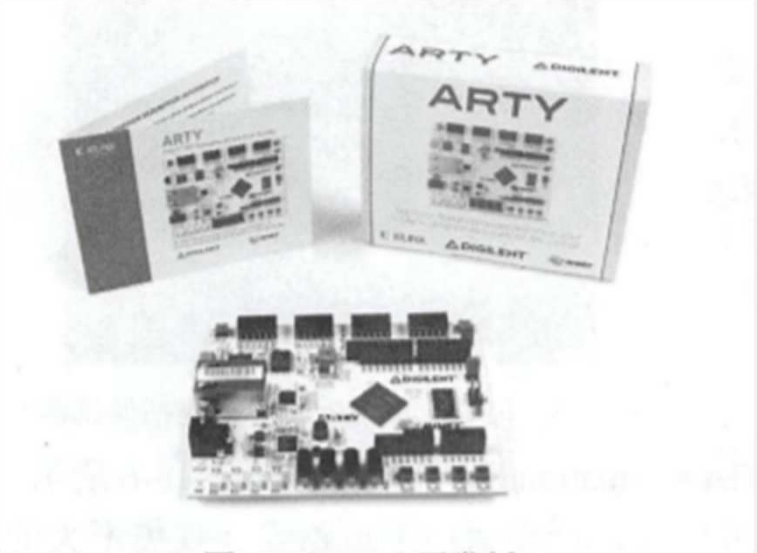


图8. FPGA开发板原型

4.1.2 FPGA 开发板

E200 开源项目 FPGA 项目相关的代码结构如下所示。

```
e200_opensource
        |----fpga                          // 存放 RTL 的目录
              |----artydevkit              // Arty 开发板的项目文件夹
                    |----constrs           // 存放约束文件的文件夹
                    |----arty-master.xdc   // 主约束文件
              |---- Makefile               // 存放Makefile 脚本
              |---- script                 // 存放运行脚本的文件夹
              |---- src                    // 存放 Verilog源代码的文件夹
                    |---- system.org       // FPGA 系统的顶层模块
        |----Makefile                      //Makefile脚本
```

4.1.2 FPGA 开发板

在FPGA的顶层模块（system.org）中除了例外的Soc的顶层（e203_soc_top）之外，主要是使用Xilinx的I/O Pad单元例外顶层的Pad。譬如JTAG接口TDO连接到名为jd0的Pad上，如图中Xilinx的MMCM单元生成时钟。

```
IOBUF_jtag_TDI
(
   O(iobuf_jtag_TDI_o),
   IO(jd_4),
   I(dut_io_pads_jtag_TDI_o_oval),
   T(~dut_io_pads_jtag_TDI_o_oe)
);
assign dut_io_pads_jtag_TDI_i_ival = iobuf_jtag_TDI_o & dut_io_pads_jtag_TDI_o_ie;
PULLUP pullup_TDI (.O(jd_4));

wire iobuf_jtag_TDO_o;
IOBUF
#(
  .DRIVE(12),
  .IBUF_LOW_PWR("TRUE"),
  .IOSTANDARD("DEFAULT"),
  .SLEW("SLOW")
)
IOBUF_jtag_TDO
(
   O(iobuf_jtag_TDO_o),
   IO(jd_0),
   I(dut_io_pads_jtag_TDO_o_oval),
   T(~dut_io_pads_jtag_TDO_o_oe)
);
assign dut_io_pads_jtag_TDO_i_ival = iobuf_jtag_TDO_o & dut_io_pads_jtag_TDO_o_ie;
```

4.1.2 FPGA 开发板

HBird E200-SoC的顶层I/O Pad经过FPGA的约束文件（arty-master.xdc）进行约束，使之连接到 FPGA 芯片外部真实的pin脚上面，譬如JTAG的Pad (jd0-jd7)被连接到了FPGA芯片的D4/D3等pin脚上，如图所示。

```
#Pmod Header JD
set_property -dict { PACKAGE_PIN D4    IOSTANDARD LVCMOS33 } [get_ports { jd_0 }];
#IO_L11N_T1_SRCC_35 Sch=jd[1]
set_property -dict { PACKAGE_PIN D3    IOSTANDARD LVCMOS33 } [get_ports { jd_1 }];
#IO_L12N_T1_MRCC_35 Sch=jd[2]
set_property -dict { PACKAGE_PIN F4    IOSTANDARD LVCMOS33 } [get_ports { jd_2 }];
#IO_L13P_T2_MRCC_35 Sch=jd[3]
#set_property -dict { PACKAGE_PIN F3    IOSTANDARD LVCMOS33 } [get_ports { jd_3 }];
#IO_L13N_T2_MRCC_35 Sch=jd[4]
set_property -dict { PACKAGE_PIN E2    IOSTANDARD LVCMOS33 } [get_ports { jd_4 }];
#IO_L14P_T2_SRCC_35 Sch=jd[7]
set_property -dict { PACKAGE_PIN D2    IOSTANDARD LVCMOS33 } [get_ports { jd_5 }];
#IO_L14N_T2_SRCC_35 Sch=jd[8]
set_property -dict { PACKAGE_PIN H2    IOSTANDARD LVCMOS33 } [get_ports { jd_6 }];
#IO_L15P_T2_DQS_35 Sch=jd[9]
#set_property -dict { PACKAGE_PIN G2    IOSTANDARD LVCMOS33 } [get_ports { jd_7 }];
#IO_L15N_T2_DQS_35 Sch=jd[10]

##USB-UART Interface (FTDI FT2232H)

set_property -dict { PACKAGE_PIN D10   IOSTANDARD LVCMOS33  } [get_ports { uart_rxd_out }];
#IO_L19N_T3_VREF_16 Sch=uart_rxd_out
set_property -dict { PACKAGE_PIN A9    IOSTANDARD LVCMOS33  } [get_ports { uart_txd_in }];
#IO_L14N_T2_SRCC_16 Sch=uart_txd_in
```

4.2.1 烧写FPGA

01 准备环境

准备Linux运行环境，安装Xilinx Vivado软件（官网下载Vivado hlx版本，利用tar命令解压。启动过程中，可以用source命令启动sh文件，也可以自己编写sh脚本启动。）

02 下载项目

将e200_opensource项目git到本地，设置需要编译的E200core具体型号，切换到/fpga目录执行：

make install CORE=e203

通过 **common.mk**(makefile) 添加特殊宏FPGA_SOURCE至e203_defines.v

03 烧写文件

安装Arty开发板的broad part到Vivado，生成bitstream文件，执行指令：

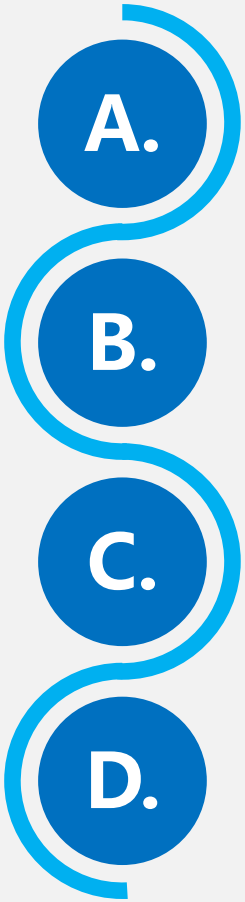
make bit

生成 **system.bit** 文件。通过Vivado的Hardware Manager烧录至FPGA。

make mcs

生成 **system.mcs** 文件，写入Flash，这样在FPGA每次上电后可以直接读用。

4.2.2 烧写步骤



打开Vivado的Hardware Manager，自动连接Arty开发板。

右键FPGA Device，选择”Add Configuration Memory Device”，选择参数。
Part n25q128-3.3v **Manufacturer** Micron **Family** n25q **Type** spi
Density 128 **Width** x1 x2x x4

一路继续后在Configuration file对话框内添加 **system.mcs** 文件选择OK开始烧写。

烧写成功后按动PROG按键触发硬件电路使用Flash中的内容对FPGA进行重新烧录，完成后DONE信号灯亮起。

4.2.3 JTAG调试

01、调试器组装

1、使用公口转母口杜邦线把Olimex ARM-USB-TINY-H域FPGA相连，注意管脚颜色编号的对应。

2、使用USB转接线（USB A to B Cable）将其与上游PC的USB端口相连。

02、实验环境配置

1、准备好Linux环境，组装调试器，将FPGA通电，确认接口被Linux识别。

2、利用 lsusb 命令查看USB设备状态。

3、设置udev rules使得USB设备能够被plugdev group访问。并且确认自己的用户是否属于group。

什么是JTAG?

JTAG是（Joint Test Action Group）联合测试行动组，也是一种边界扫描测试规范。目的是为了解决PCB电路板在完成焊接后如何测试所有被焊接的芯片管脚连通性问题，后来也用于集成电路芯片在完成封装后的Final Testing的测试矢量和测试结果输出。还作为访问片上在线仿真（ICE）逻辑的通信接口。

5.1 总结

FPGA原型平台DIY总结

1. SoC的作用是什么？Hbird-E200和Freedom E310的继承关系？地址的分配和接口分配是怎样的？Hbird-E200-SoC的代码结构?
2. 购买Xilinx Artix-7 35T Arty FPGA Evaluation Kit开发板、USB A to B Cable、 USB A to Micro-B Cable 、Olimex ARM-USB-TINY-H Debugger、杜邦线。
3. 安装Linux操作系统，Xilinx的Vivado软件和Arty开发板的board part。如何烧录？生成bitstream或者mcs文件（如果需要拷贝到FPGA外部的Flash的话）
4. 如何组装JTAG调试器？上游PC机，下游FPGA，中间是个Olimex ARM-USB-TINY-H。

华北电力大学
NORTH CHINA ELECTRIC POWER UNIVERSITY

谢谢聆听

控制与计算机工程学院
计算机科学与技术