目录

1 基于 GS232 搭建的 SoC_up 说明

1.1 GS232 开源版本简介

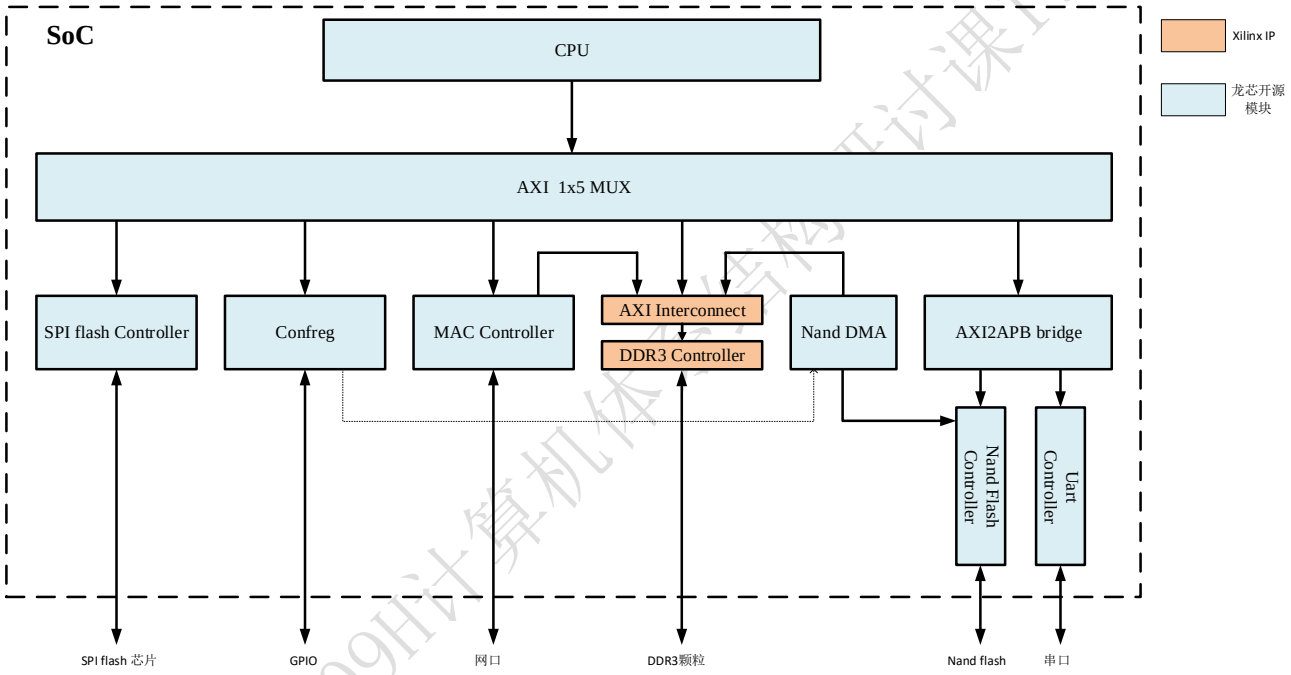
GS232 开源版本不包含 DSP、浮点部件等。

TLB 大小为 32 项。

指令和数据 Cache 为 4 路组相连，每路大小为 4KB，Cache 行大小为 32 bytes。

对外接口为 32 位 AXI 接口。

1.2 SoC_up 结构



SoC_up 如上图所示。开源 GS232 对外有一个 AXI 接口，连接到 AXI 互联网络上与外设相连。

SoC_up 对外连接的设备共有 6 个：SPI flash、GPIO(数码管、LED 灯、开关灯)、网口、DDR3 颗粒、Nand flash 和串口。这些外设在教学实验板上均已集成。

1.3 地址空间分配

各外设地址空间分配如下：

各控制器模块名	分配的虚拟地址段	地址空间大小
SPI flash	0xbfc0_0000_0xbfcf_ffff 和 0xbfe4_0000~0xbfe4_ffff	1MB(flash 存储空间) 64KB(控制器寄存器空间)
GPIO	0xbfd0_0000~0xbfd0_ffff	64KB
MAC	0xbff0_0000~0xbff0_ffff	64KB

DDR3	4GB 空间中的剩余地址[1]	128MB
Nand flash	0xbfe7_8000~0xbfe7_bfff	16KB
Uart	0xbfe4_0000~0xbfe4_3fff	16KB
地址空洞 (软件可以不关注)	0xbfe7_0000~0xbfe7_7fff 0xbfe7_c000~0xbfe7_ffff 0xbfe4_4000~0xbfe4_ffff	分配给 APB 设备的，但由于 APB 设备只有 uart 和 nand，故存在较多的地址空洞，留作后续新增 APB 设备使用。

1.4 各控制器说明

1.4.1 NAND DMA 控制器

其一端通过 64 位 AXI 接口接到 DDR3 内存上，一端通过 APB 接口接到 APB 设备上（可认为接到 NAND 控制器上）。

该 DMA 只用于 nand flash 与内存交换数据。

该 DMA 的配置寄存器 ORDER_ADDR_IN 位于 CONFREG 模块，地址还是为 0xbfd0_1160。

1.4.2 NAND FLASH 控制器

通过 APB 接口接在 APB 桥上。

该 NAND FLASH 控制器不支持上电从 flash 启动和校验纠错。

FPGA 板上 NAND FLASH 颗粒 K9F1G08U0C-PCB0 的 main 区容量为：1K blocks * 64 pages/block * 2K Bytes/page =128M bytes。也就是共有 64k 页，一页为 2k bytes。每页的 spare 区为 64bytes。

1.4.3 CONFREG 模块

包含 8 个 32 位内存映射读写寄存器和一个 dma 的 order_addr_in 寄存器（0xbfd0_1160）。

1.4.4 MAC 控制器

一个从端 32 位 AXI 接口，接到 AXI 互联网络上供 CPU 访问。一个主端 32 位 AXI 接口，接到 DDR3 内存上，自带 DMA 功能。

1.4.5 DDR3 控制器

为 Xilinx IP。

不需要软件配置。一上电，整个 SoC 会先等 DDR3 完成复位，才会撤掉 CPU 的复位信号。

其实分为，一个 3x1 的 AXI 仲裁器，和一个 DDR3 控制器，均为 XilinxIP。

其中 3x1 的 AXI 仲裁器为 2 个 32 位 AXI 接口：一个接 AXI 互联网络供 CPU 访问，一个接 MAC 控制器供网口访问；还有一个 64 位 AXI 接口供 NAND DMA 访问。

[1] 剩余地址：指 4GB 地址空间中，除了其他外设分配的地址外的地址。由于实验板上 DDR3 颗粒大小为 1Gb，即 128MB，故软件访问时注意不要越界，建议软件使用虚拟地址 0x8000_0000~0x87ff_ffff 对其进行访问。

1.4.6 SPI FLASH 控制器

通过一个 32 位 AXI 接口接到 AXI 互联网络上。

1.4.7 UART 控制器

仅支持一个串口设备。

1.5 中断连接

由于教学 SoC 比较小，故没有集成中断控制器，也就只实现 1 级中断。

其中 6 个硬件中断（$Cause_{IP7-2}$）高位未连接外部中断，其余 5 位硬件中断和外设连接关系如下。

硬件中断名	来源
$Cause_{IP6}$	dma_int：NAND DMA 的中断请求
$Cause_{IP5}$	nand_int：NAND FLASH 的中断请求
$Cause_{IP4}$	spi_int：SPI FLASH 的中断请求
$Cause_{IP3}$	uart0_int：串口的中断请求
$Cause_{IP2}$	mac_int：网口的中断请求

目前 SoC 的各控制器的中断连接为上表的方案。

1.6 时钟方案

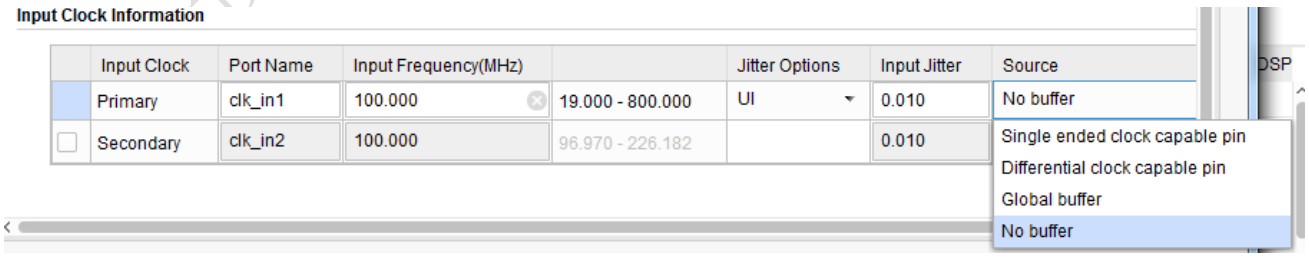
开发板上时钟晶振为 100MHz。

SoC_up 中的 DDR3 内存控制器使用的是 xilinx IP，其需要一个 100MHz 的输入时钟，以及一个 200MHz 的参考时钟。因而需要使用 xilinx IP 中的 PLL 单元，将 100MHz 的源时钟倍频到 200MHz。

另外，由于 SoC_up 设计比较复杂，运行在 100MHz 的工作频率下，因而 SoC_up 的工作时钟，也就是 CPU 时钟，通用需要使用 xilinx IP 中的 PLL 单元进行分频，设置到 33MHz。

因而在 SoC_up 中使用了两个 PLL 单元：一个用于将 100MHz 的源时钟倍频到 200MHz，作为 DDR3 内存控制器的参考输入时钟；另一个用于将 100MHz 的源时钟分频到 33MHz，作为 SoC_up 的输入时钟。

两个 PLL 单元的输入时钟都是开发板上提供的 100MHz 时钟，其余“Source”栏都需要选择为”No buffer”，如下图。



另外，SoC_up 中的内存控制器工作频率设置为 400MHz，而 CPU 时钟为 33MHz，因而生成的内存控制器 xilinx IP 需要勾选跨时钟域选项。

2 Artix-7 教学实验板固化方法

本章给出基于 Artix-7 的教学实验板上固化一个 FPGA 设计的方法。

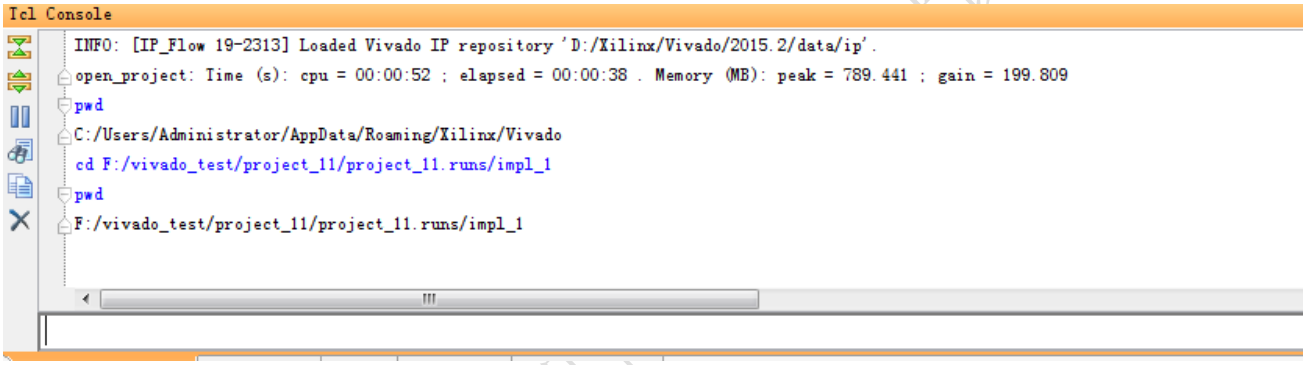
固化后，每次上电时，实验板会自动加载设计到 FPGA 芯片上。因而断电重新上电后不需要重新下载 bit 流文件，极大方便了实验板上软件编程。

固化的流程：先将一个 bit 流文件转换为 mcs 文件，将 mcs 文件下载到板上一个 SPI flash 上。

2.1 生成 mcs 文件

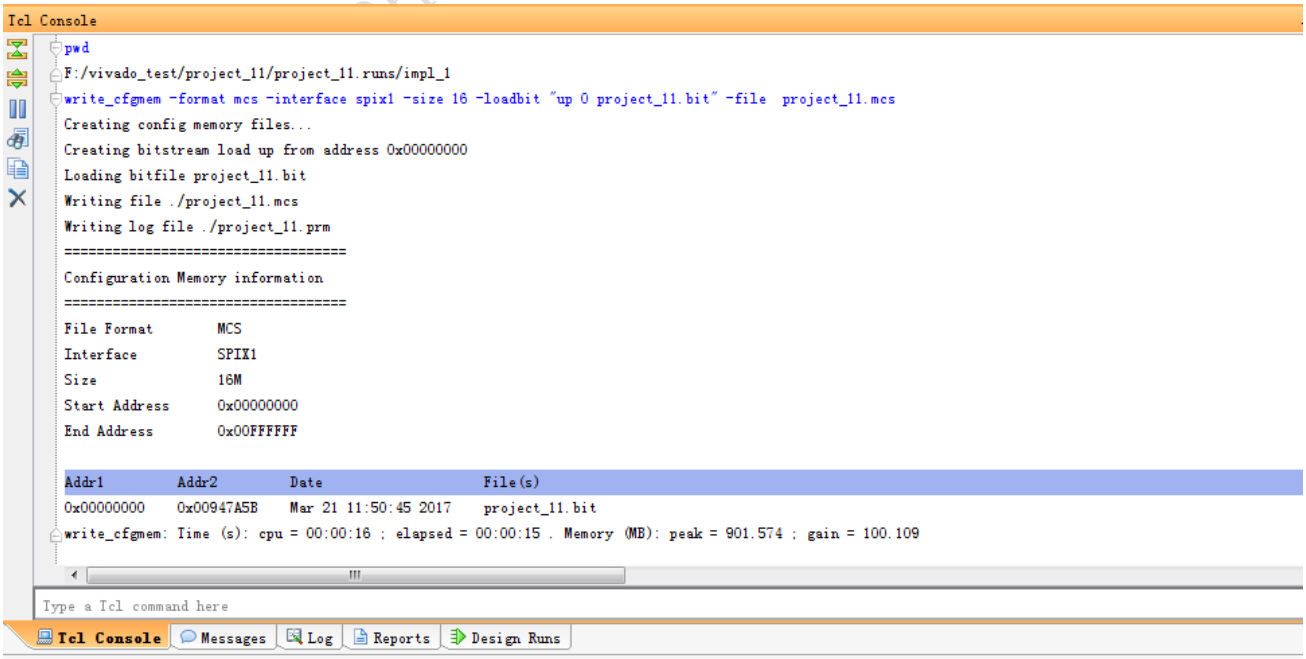
首先，需要确保 FPGA 设计的 bit 流文件已经生成。但 bit 流文件是用于直接下载到 FPGA 芯片里的文件，而不能下载到 flash 芯片里，因而需要转换为 mcs 文件。

在 ISE 工具里，可以再图形界面下点击选择就可生成 mcs 文件，但在 Vivado 工具里，生成 mcs 文件需要在命令控制器（tcl console）里输入命令。如下图：



上图中蓝色的为输入的命令，pwd 用于查看目录。随后使用 cd 命令进入 bit 流文件所在的目录。

假设生成的 bit 流文件为 project_11.bit，则输入命令串 **write_cfgmem -format mcs -interface spix1 -size 16 -loadbit "up 0 project_11.bit" -file project_11.mcs** 即可生成 mcs 文件，如下图。



其中命串里的 project_11.bit 为待转换的 FPGA 设计的 bit 文件，project_11.mcs 为生成的 mcs 文件名，可以自

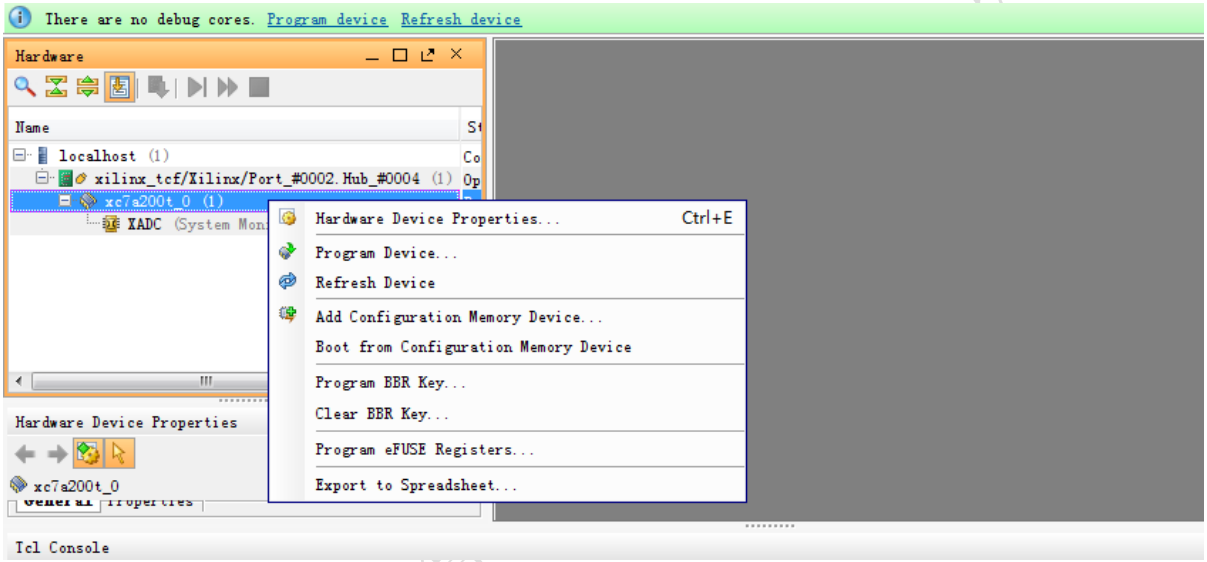
定义。

上述命令，是先输入 cd 命令将目录切换到了 bit 流文件的目录，再使用 write_cfgmem 命令将 bit 流文件转换为 mcs 文件。这两步可以使用命令串 **write_cfgmem -format mcs -interface spix1 -size 16 -loadbit "up 0 F:/vivado_test/project_11/project_11.runs/impl_1/project_11.bit" -file F:/vivado_test/project_11/project_11.runs/impl_1/project_11.mcs** 一步到位。这一命令串中明确指定了 bit 流文件的目录，和生成的 mcs 文件的保存目录，bit 流文件的目录和文件名必须正确，但 mcs 文件的目录和文件名可以自定义。

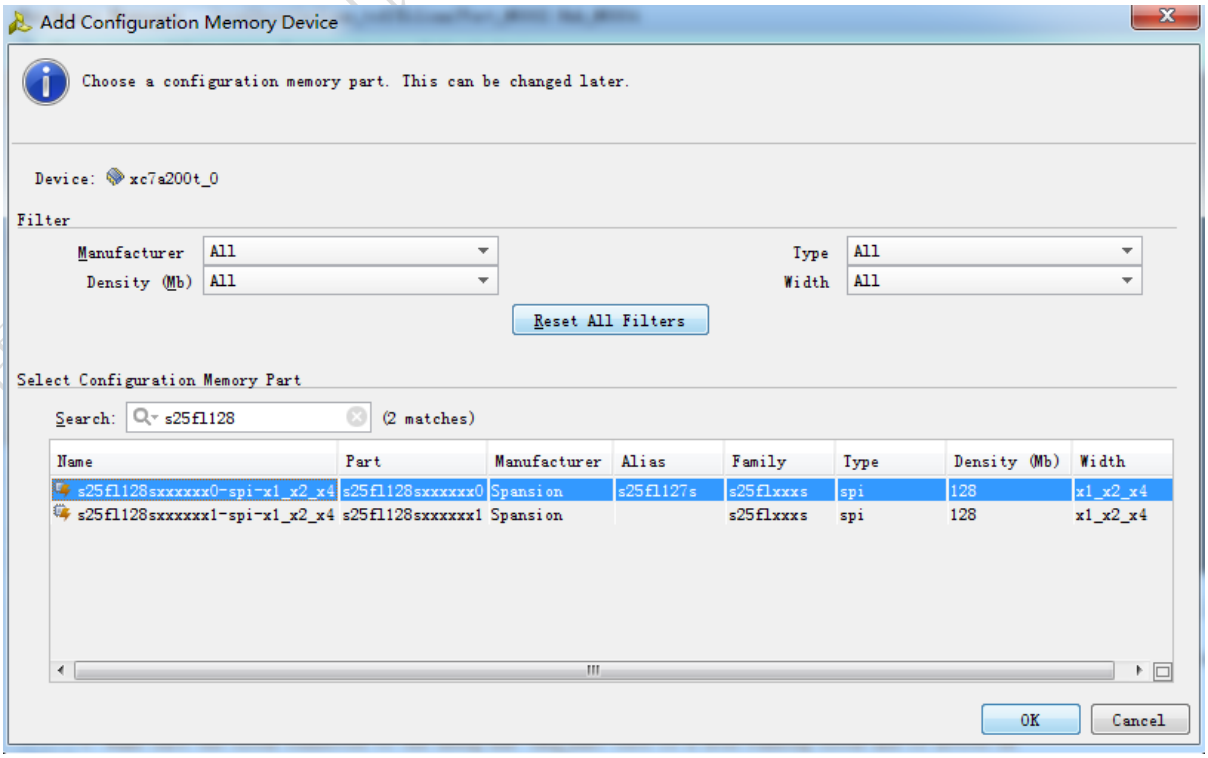
2.2 下载 mcs 文件

生成好 mcs 文件后，就需要将其下载到实验板上 SPI flash 上。

像下载 bit 流文件一样，打开 Vivado 工具里的“Open Hardwar Target”，连接设备。

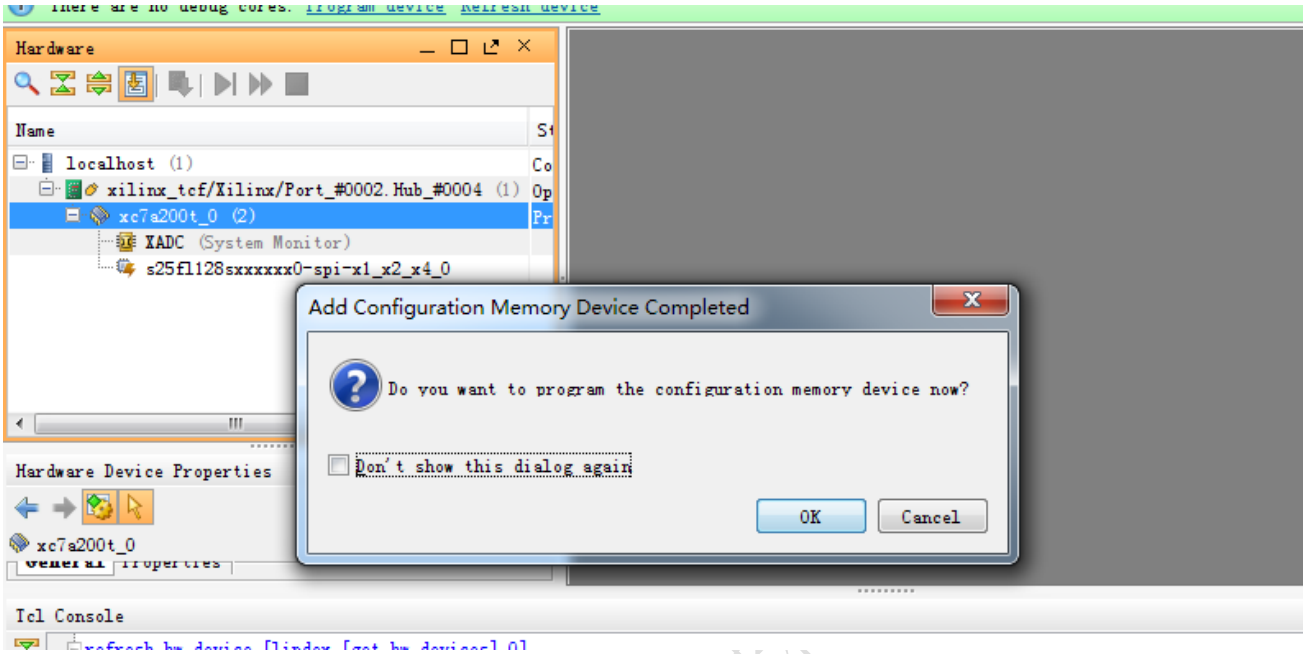


选中 xc7a200t 后，右键选择“Add Configuration Memory Device”，出现如下界面：

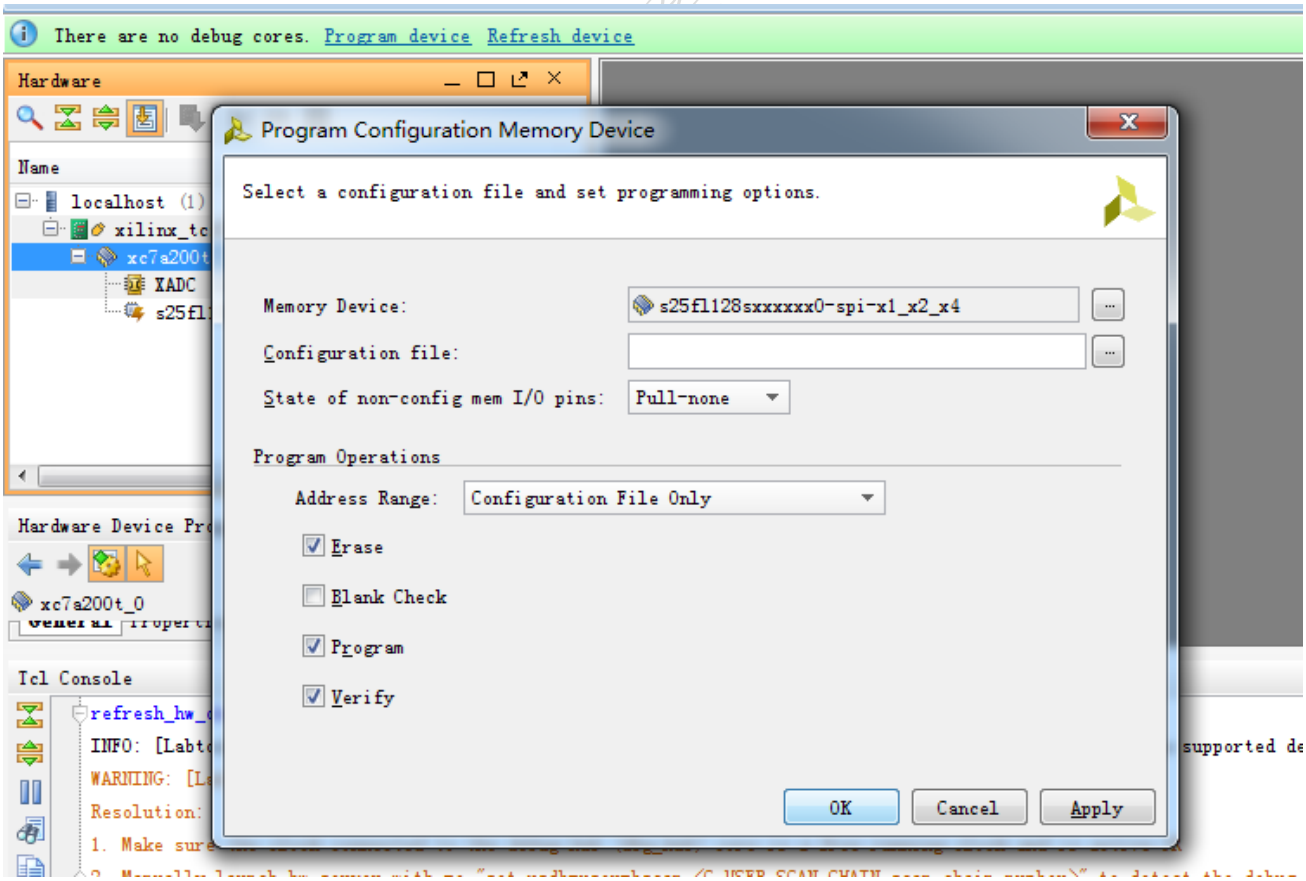


在 search 栏输入 s25fl128，出现两个可选的芯片型号：s25fl128sxxxxxx0-spi-x1_x2_x4 和 s25fl128sxxxxxx1-spi-x1_x2_x4。具体选择哪个型号的，需要与板上固定的 flash 芯片型号相同（看板上 flash 型号标识的末尾是 0 还是 1）。也可以两者先任选一个，如果后续编程 flash 失败，再回来选另一个型号的。

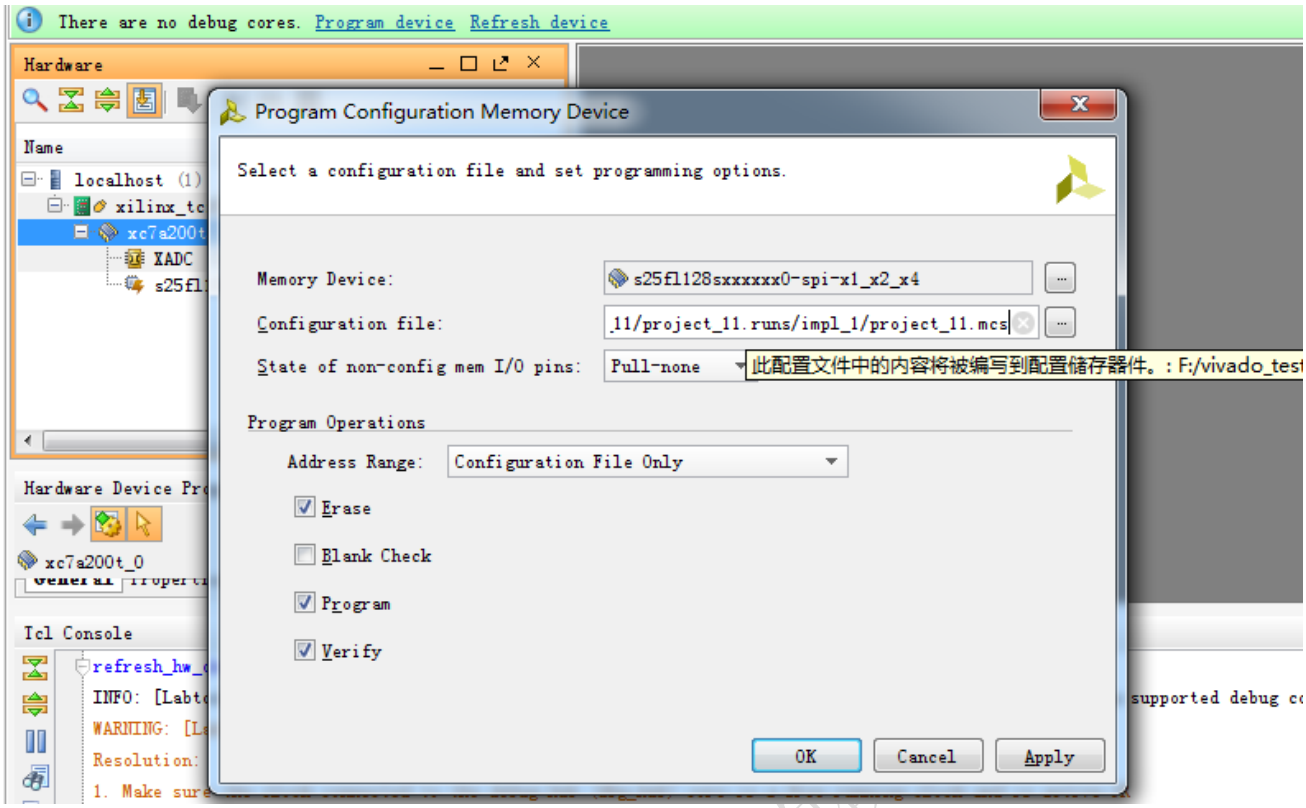
选好 flash 型号后，点击 OK，弹出如下窗口，询问是否现在编程 flash：



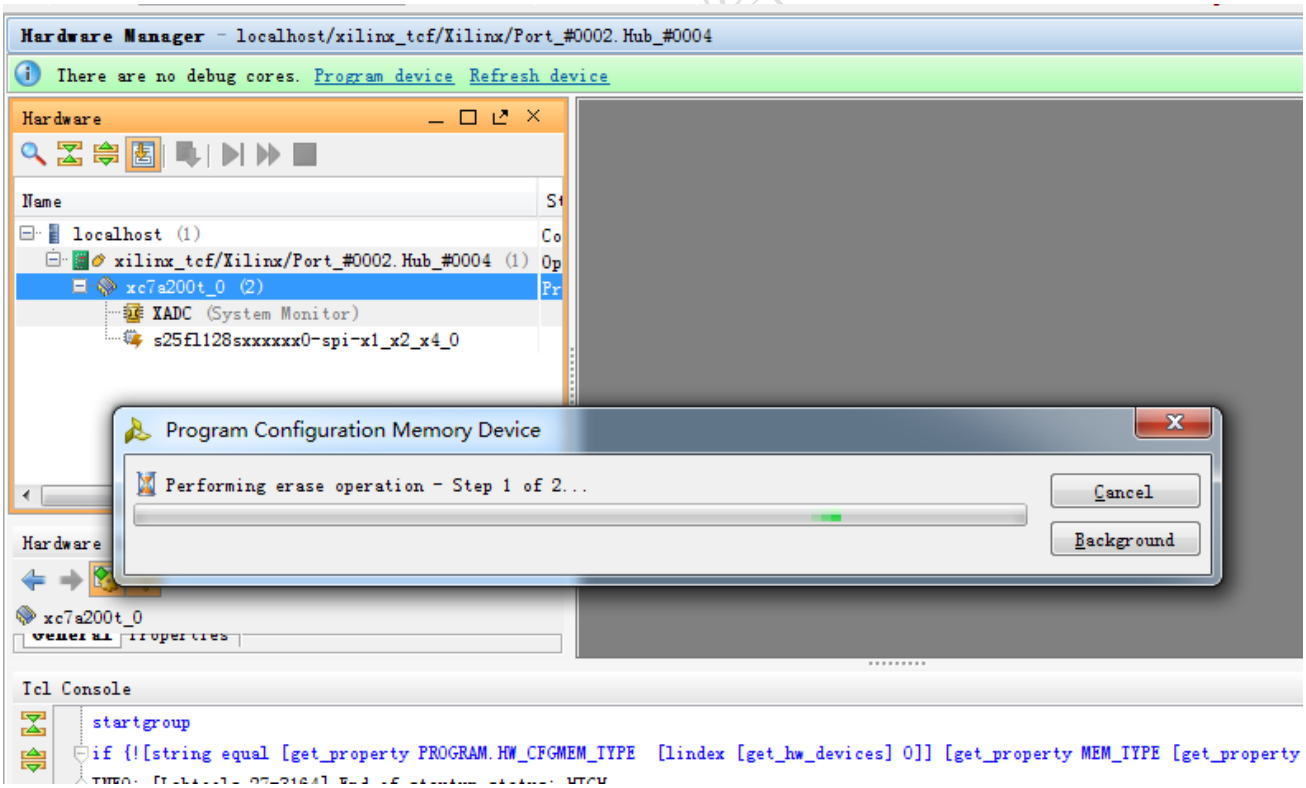
点击 OK，出现编程 flash 的界面：



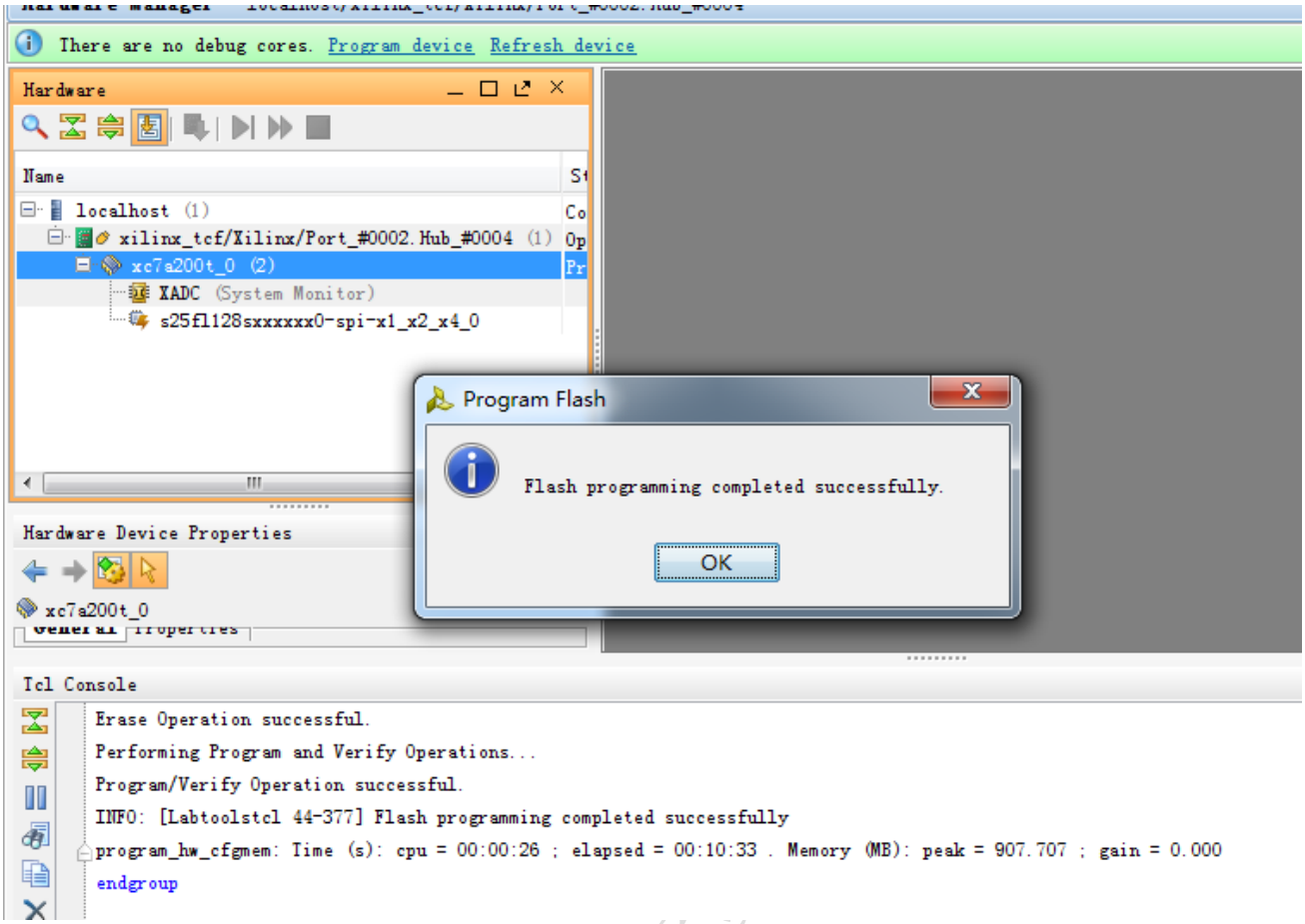
在 Configuration file 那栏选到 2.1 小节生成的 mcs 文件，如下图：



点击 OK 即可。后续等待下载 mcs 到 flash 芯片完成即可，如下图：



Flash 芯片会先进行擦除，在进行编写，完成后会提示 completed Successfully，如下图：



此时烧写就完成了，需要将实验板断电重新上电，等待一段时间，就可发现固化进实验板的 FPGA 设备自动加载完成了。