

# SoC\_lite 与 SoC\_up 介绍

(v0.01)

## 1 基于 GS132 搭建的 SoC\_lite 说明

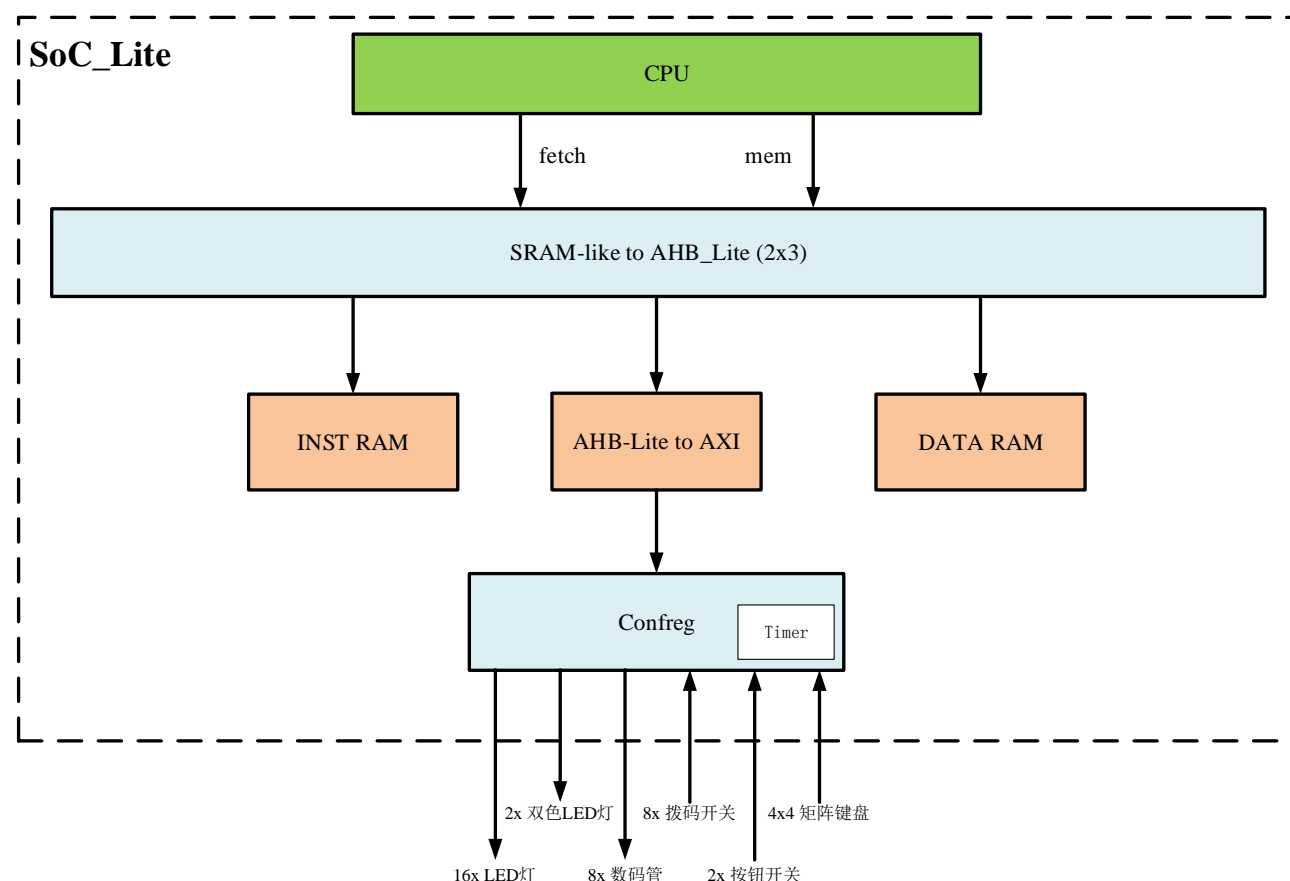
### 1.1 GS132 开源版本简介

龙芯 GS132 是一款实现 MIPS32 基本指令集兼容的 32 位 RISC 处理器核，为单发射三级流水架构，未实验 TLB/Cache 和浮点部件。

对外有三个接口：两个地址空间可配置的 SRAM 接口，分别作为取指和数据访问；一个 32 位 AXI 接口，32 位地址空间除了分配给两个 SRAM 接口外的其余地址都是通过 AXI 接口访问的。

### 1.2 SoC\_lite 结构

在基于 GS132 搭建 SoC\_lite 时，对 GS132 处理器核进行了一定的修改，只保留了类 SRAM 接口的 fetch 和 mem 接口，且取指和数据访问是隔离访问的。SoC\_lite 结构如下图：



Soc\_lite 中 CPU 对外的取指和数据访问接口连接到一个全互联的 2x3 的转化桥上，转换成两个 SRAM 接口和一个 AHB-lite 接口。两个 SRAM 接口分别连到 INST 和 DATA ram 上。AHB-lite 接口则转换后 AXI 接口上，挂上 Confreg 模块，连接 LED 灯、开关等。

### 1.3 地址空间分配

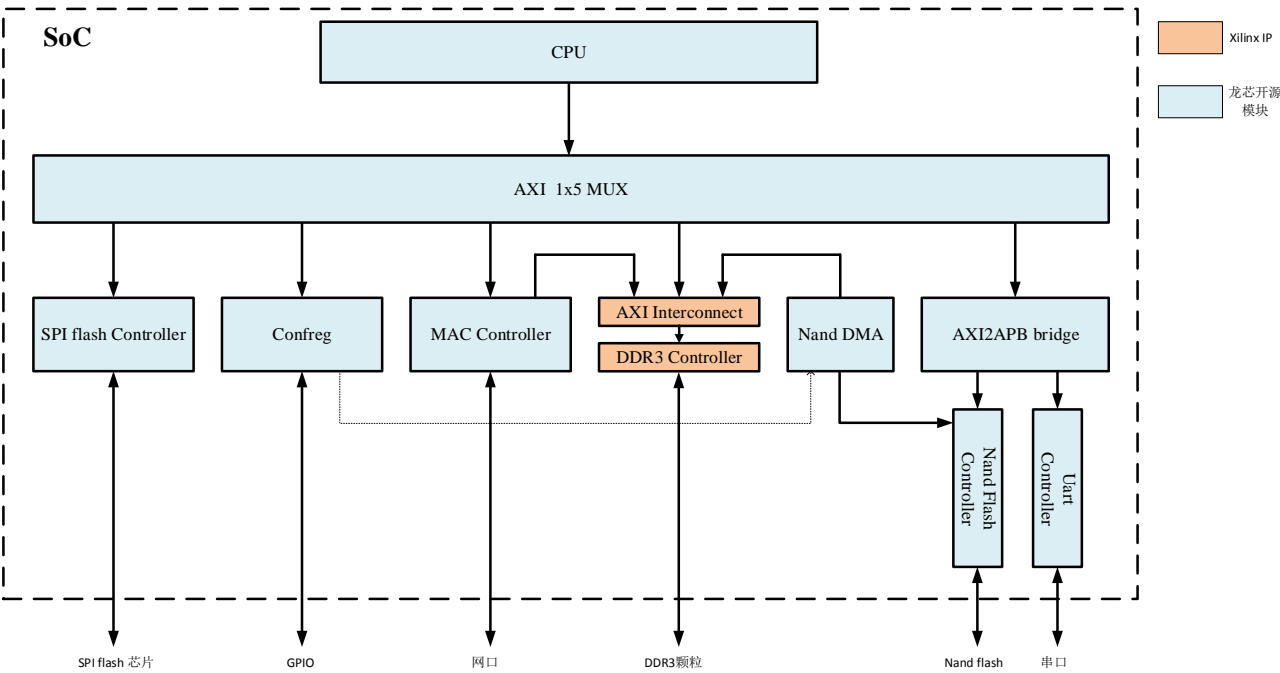
INST 和 DATA ram 的地址空间是可配置的，配置参数参见源码 `sramlike_to_ahb.v` 的宏定义。32 为地址空间中，除了分配给 INST 和 DATA ram 的，其余地址都通过 AHB-lite 访问，也就是去往了 Confreg 模块。

## 2 基于 GS232 搭建的 SoC\_up 说明

### 2.1 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。

### 2.2 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 互连网络上与外设相连。

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。

### 2.3 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	1MB(flash 存储空间)

	0xbfe4_0000~0xbfe4_ffff	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 <sup>1</sup>	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。

## 2.4 各控制器说明

### 2.4.1 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。

### 2.4.2 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。

### 2.4.3 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）。

### 2.4.4 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。

### 2.4.5 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。

<sup>1</sup> 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。

## 2.4.6 SPI FLASH 控制器

通过一个 32 位 AXI 接口接到 AXI 互连网络上。

## 2.4.7 UART 控制器

仅支持一个串口设备。

## 2.5 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<sub>IP7-2</sub>）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 <sub>IP6</sub>	dma_int: NAND DMA 的中断请求
Cause <sub>IP5</sub>	nand_int: NAND FLASH 的中断请求
Cause <sub>IP4</sub>	spi_int: SPI FLASH 的中断请求
Cause <sub>IP3</sub>	uart0_int: 串口的中断请求
Cause <sub>IP2</sub>	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。

## 2.6 时钟方案

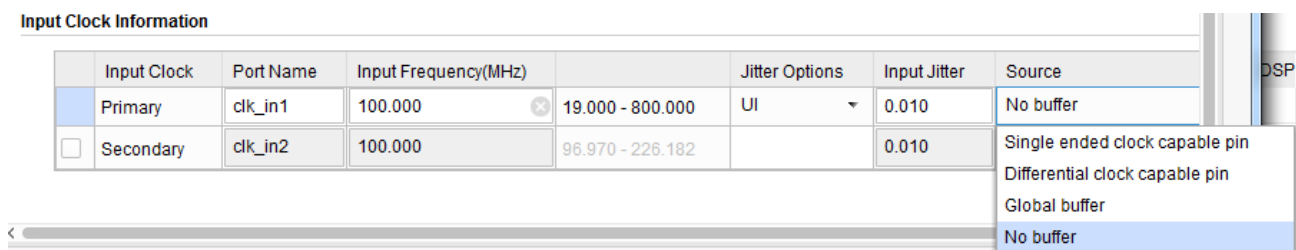
开发板上时钟晶振为 100MHz。

SoC<sub>up</sub> 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<sub>up</sub> 设计比较复杂，运行在 100MHz 的工作频率下，因而 SoC<sub>up</sub> 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<sub>up</sub> 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<sub>up</sub> 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<sub>up</sub> 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。

### 3 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。

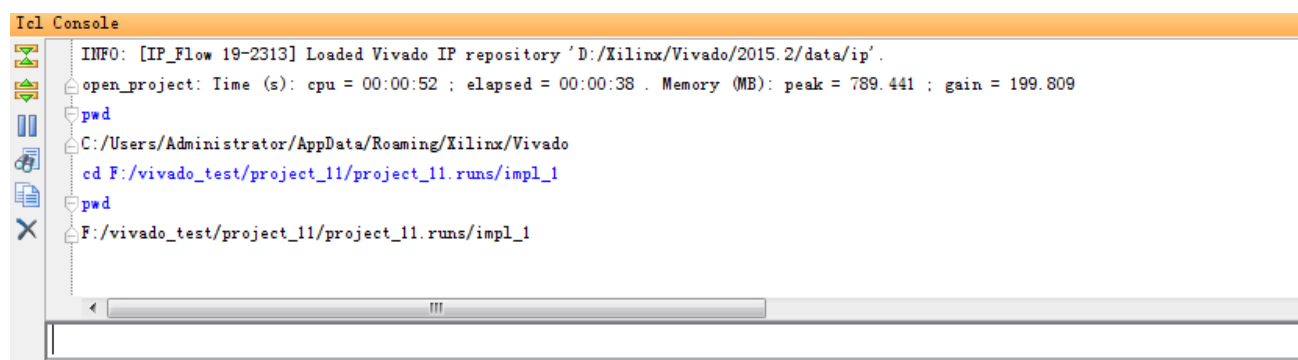
固化后，每次上电时，实验板会自动加载设计到 FPGA 芯片上。因而断电重新上电后不需要重新下载 bit 流文件，极大方便了实验板上软件编程。

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。

#### 3.1 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。

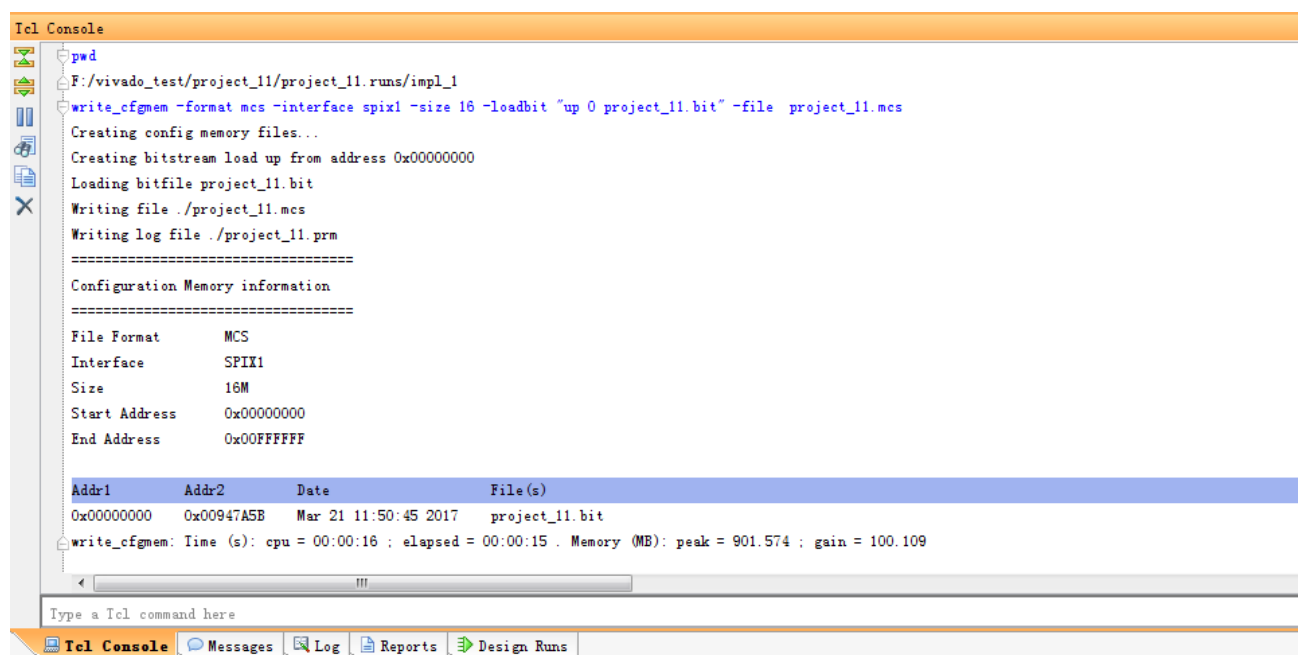
在 ISE 工具里，可以再图形界面下点击选择就可生成 mcs 文件，但在 Vivado 工具里，生成 mcs 文件需要在命令控制器（tcl console）里输入命令。如下图：



```
Tcl Console
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Xilinx/Vivado/2015.2/data/ip'.
open_project: Time (s): cpu = 00:00:52 ; elapsed = 00:00:38 . Memory (MB): peak = 789.441 ; gain = 199.809
pwd
C:/Users/Administrator/AppData/Roaming/Xilinx/Vivado
cd F:/vivado_test/project_11/project_11.runs/impl_1
pwd
F:/vivado_test/project_11/project_11.runs/impl_1
```

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。



```
Tcl Console
pwd
F:/vivado_test/project_11/project_11.runs/impl_1
write_cfgmem -format mcs -interface spix1 -size 16 -loadbit "up 0 project_11.bit" -file project_11.mcs
Creating config memory files...
Creating bitstream load up from address 0x00000000
Loading bitfile project_11.bit
Writing file ./project_11.mcs
Writing log file ./project_11.prm
=====
Configuration Memory information
=====
File Format      MCS
Interface        SPIX1
Size             16M
Start Address    0x00000000
End Address      0x00FFFFFF

Addr1    Addr2    Date           File(s)
0x00000000 0x00947A5B  Mar 21 11:50:45 2017  project_11.bit
write_cfgmem: Time (s): cpu = 00:00:16 ; elapsed = 00:00:15 . Memory (MB): peak = 901.574 ; gain = 100.109
```

其中命令串里的 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，

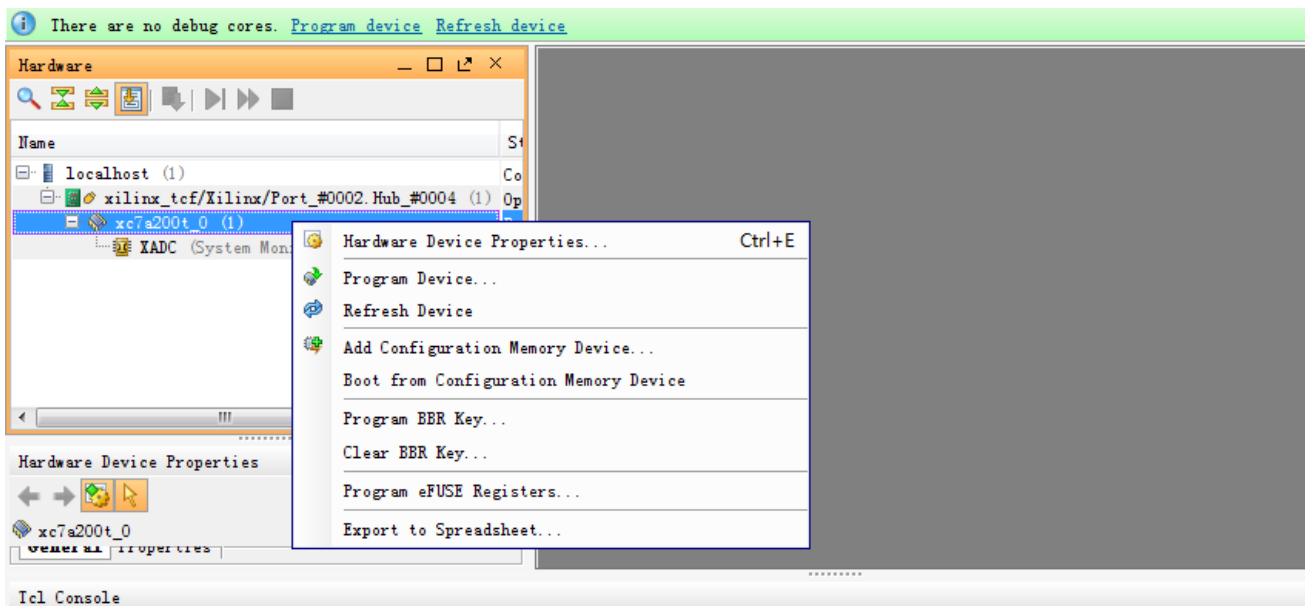
可以自定义。

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。

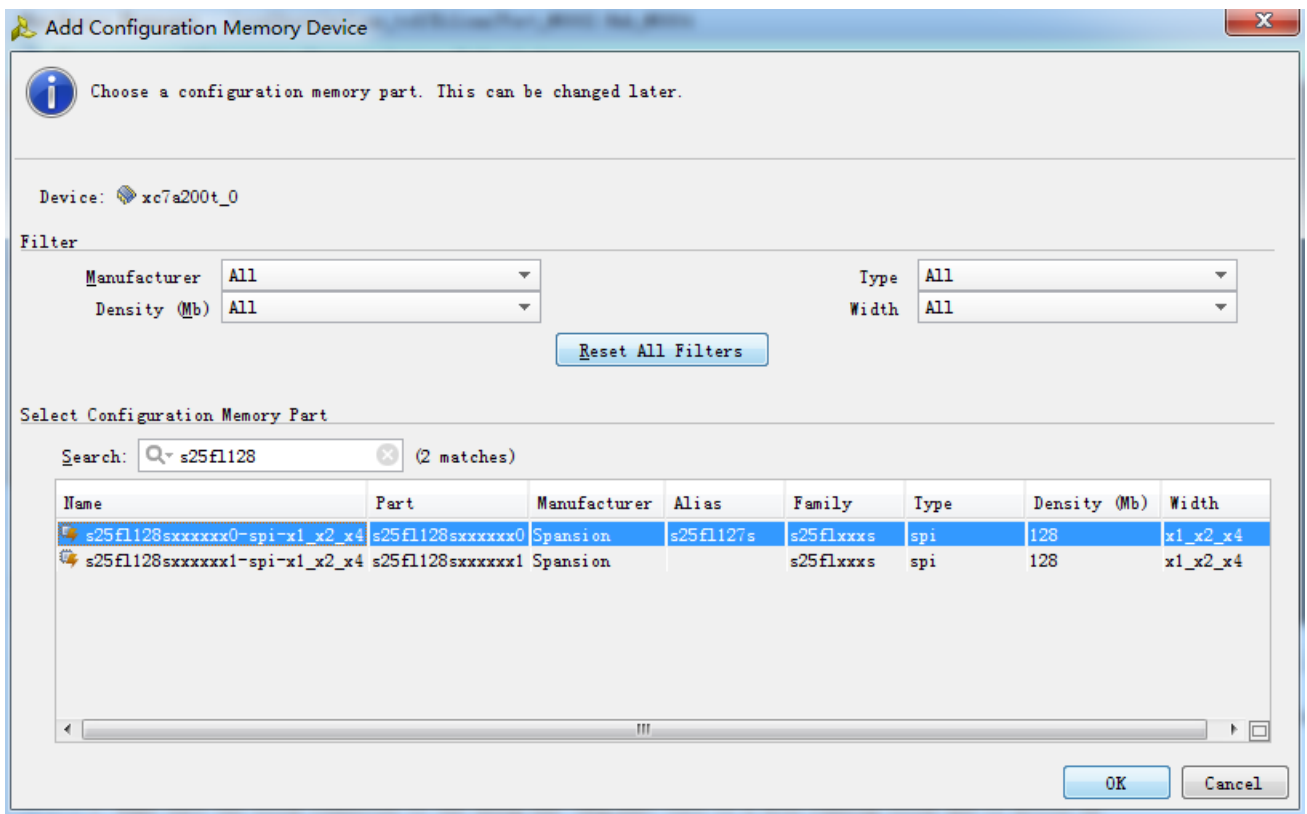
## 3.2 下载 mcs 文件

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。

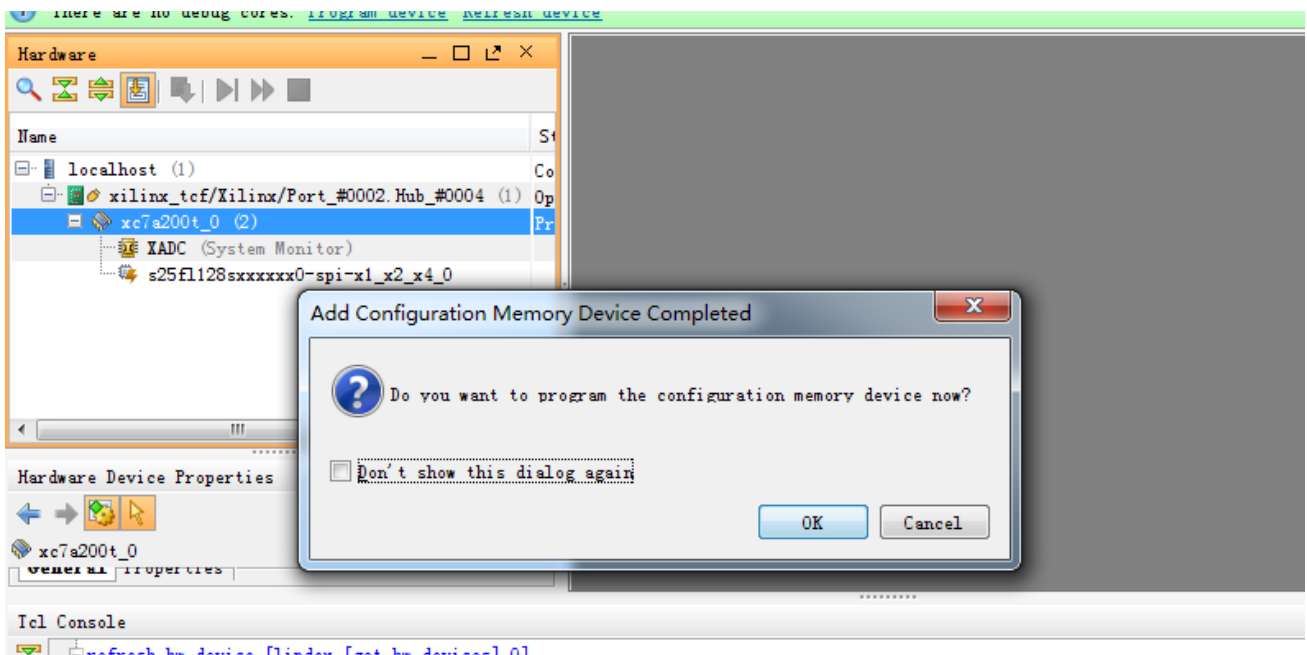


选中 xc7a200t 后，右键选择“Add Configuration Memory Device”，出现如下界面：

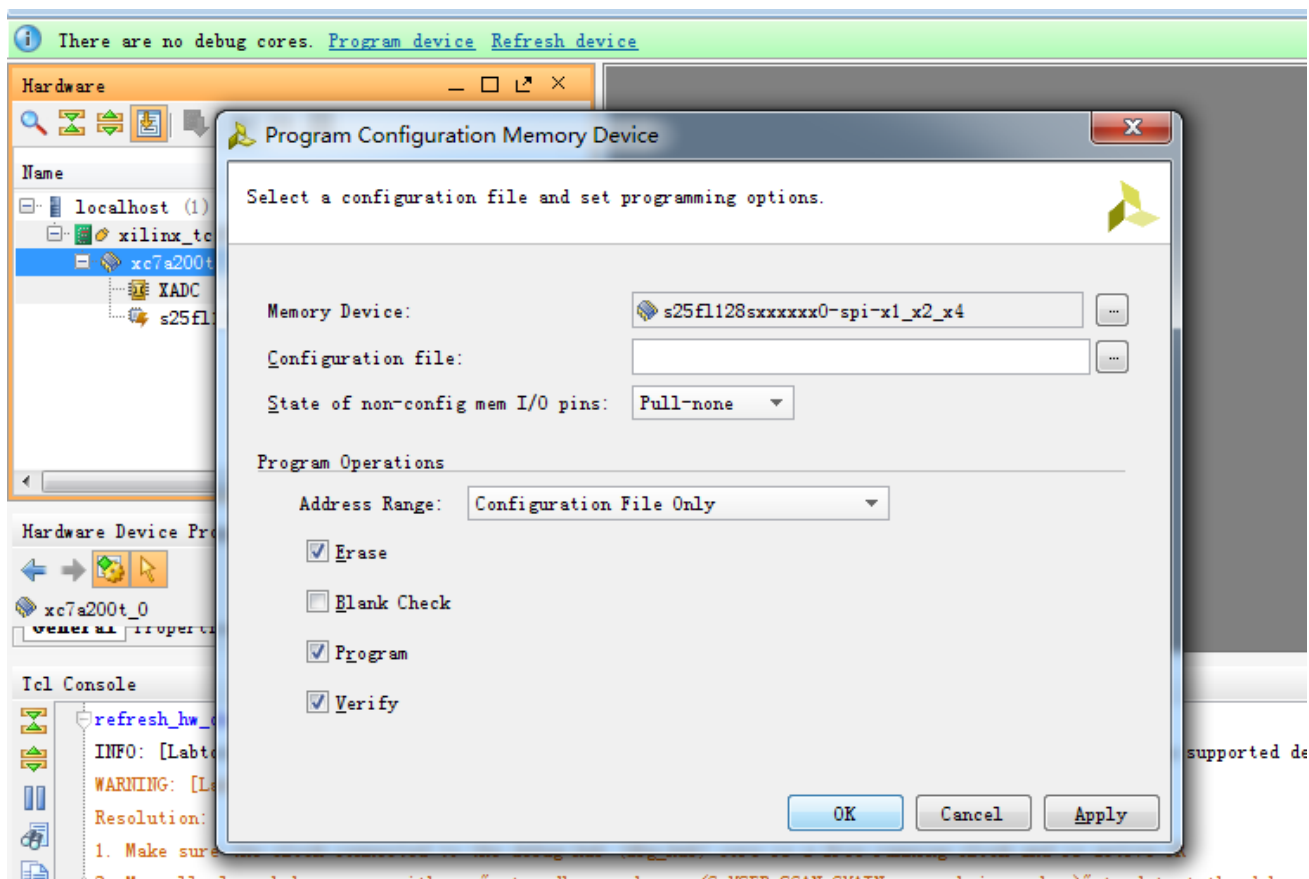


在 search 栏输入 s25fl128，出现两个可选的芯片型号：s25fl128xxxxxx0-spi-x1\_x2\_x4 和 s25fl128xxxxxx1-spi-x1\_x2\_x4。具体选择哪个型号的，需要与板上固定的 flash 芯片型号相同（看板上 flash 型号标识的末尾是 0 还是 1）。也可以两者先任选一个，如果后续编程 flash 失败，再回来选另一个型号的。

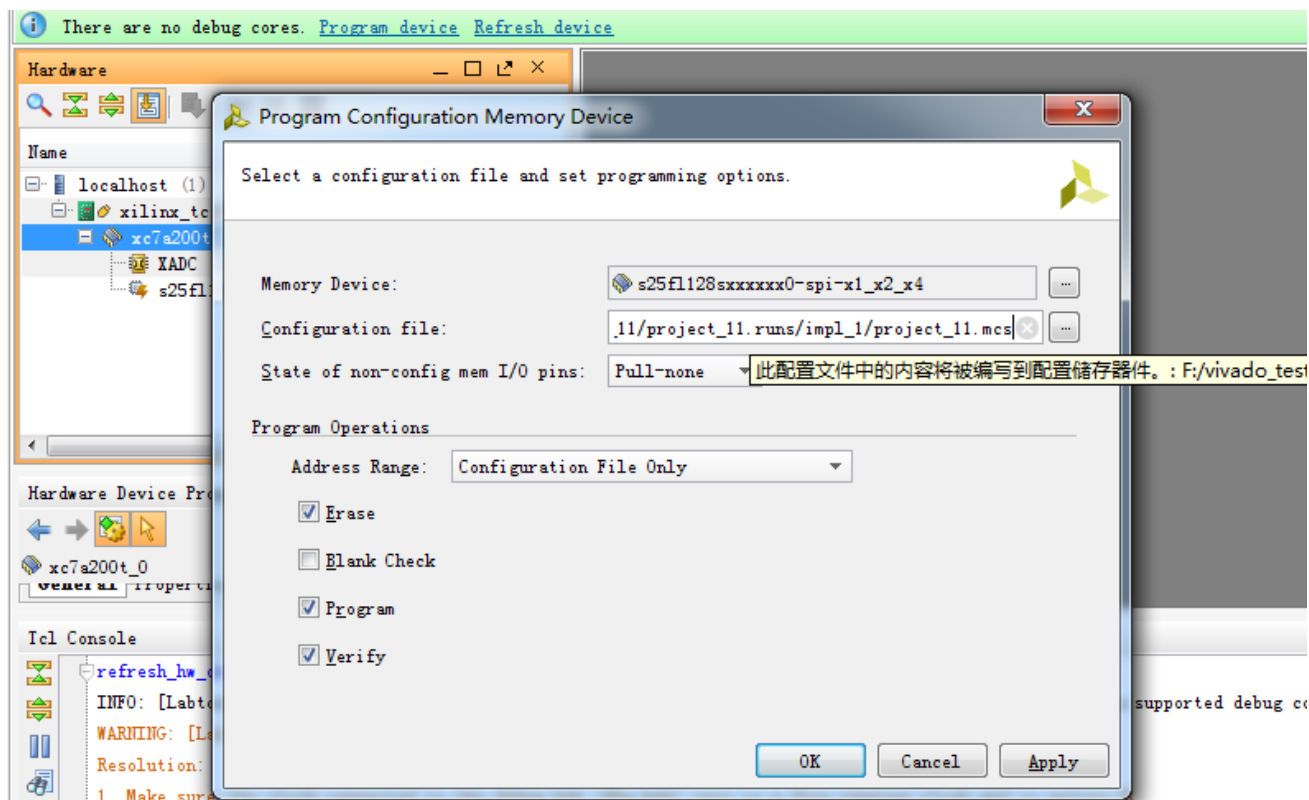
选好 flash 型号后，点击 OK，弹出如下窗口，询问是否现在编程 flash：



点击 OK，出现编程 flash 的界面：

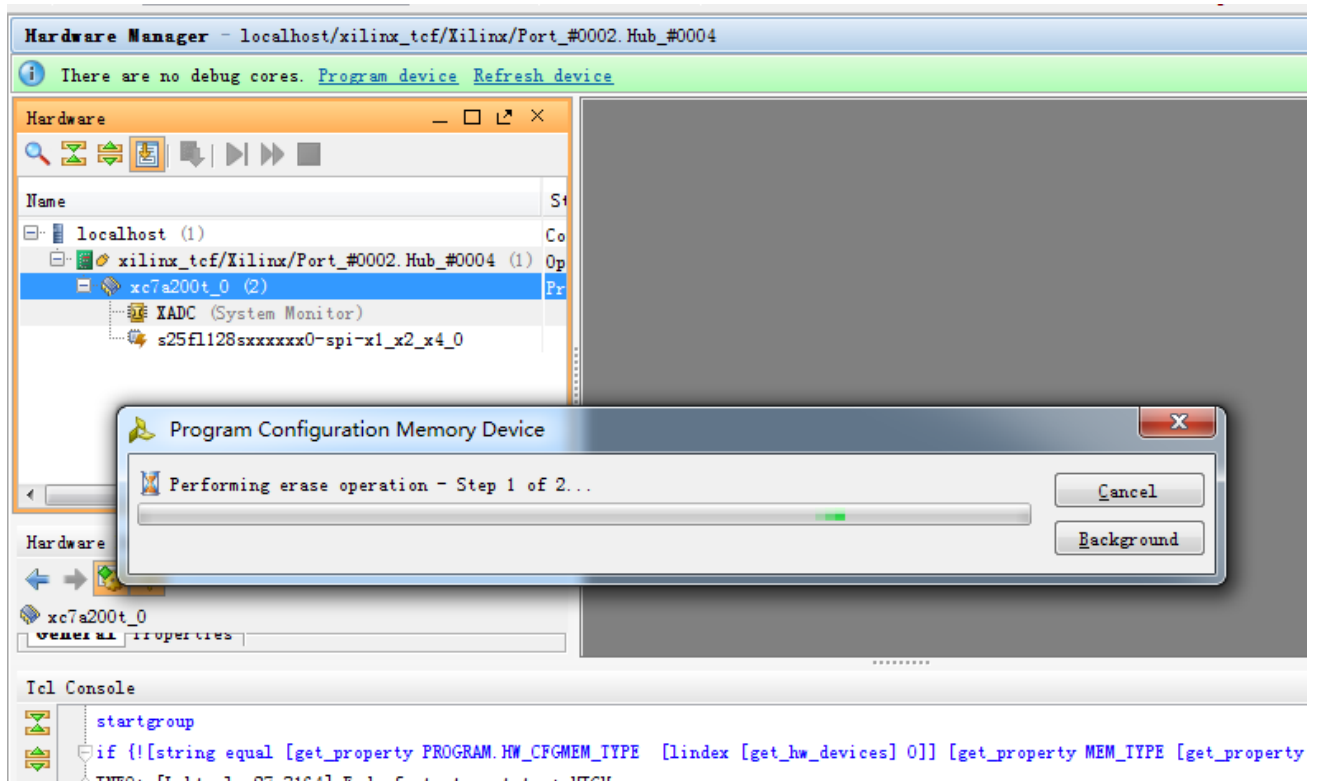


在 Configuration file 那栏选到 3.1 小节生成的 mcs 文件，如下图：

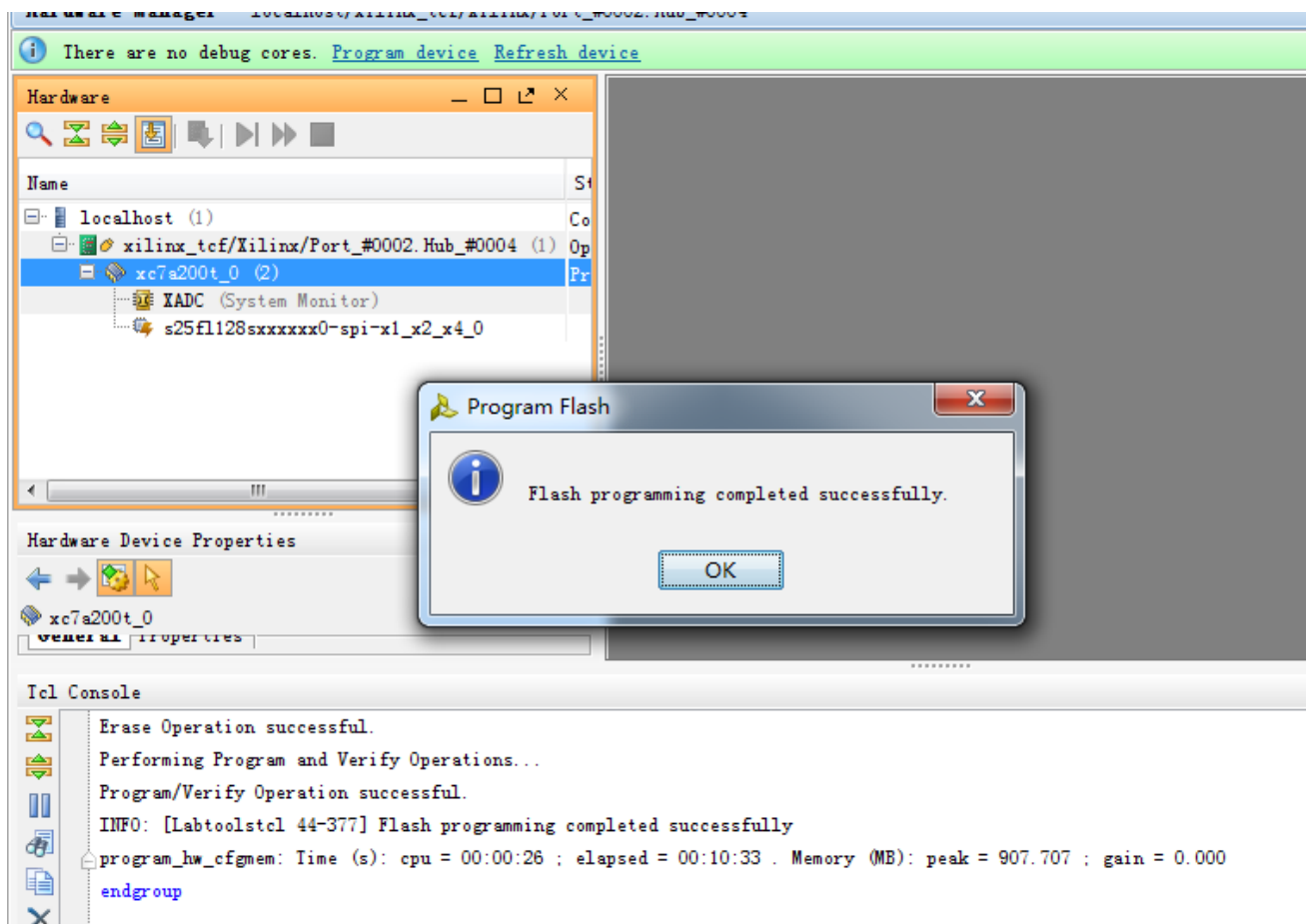


点击 OK 即可。后续等待下载 mcs 到 flash 芯片完成即可，如下图：





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。

