FPGA Cortex-M1 SoC 快速使用手册

(2020.10.28)

V1.0

文档版本修订记录

版本号	发布日期	修订记录
V0.1	2019/10/10	初始版本
V0.2	2020/01/14	替换模糊图片以及修改描述细节
V0.3	2020/04/03	硬件平台搭建与RTL仿真介绍、DEMO使用方法介绍
V0.4	2020/07/30	补充Ethernet DMAC相关信息与细节完善
V0.5	2020/08/21	完善新版本CACHE大小等信息
V1.0	2020/10/28	增加UDP加速功能描述和外设扩展描述

名词术语解释

Abbreviations 缩略语	Full Spelling 英文全拼	Chinese Explanation 中文解释
PDS	Pango Design Stutio	RTL开发工具
ITCM	Instruction Tightly-Coupled Memory	指令紧耦合空间
DTCM	Data Tightly-Coupled Memory	数据紧耦合空间
DDR	Double Data Rate	双倍速率
ICACHE	Instruction Cache	指令缓存
DCACHE	Data Cache	数据缓存

目录

图目录

1 概述

本文档为 Cortex M1 SoC 快速使用文档，主要介绍产品概况、软硬件平台以及 DEMO 的快速使用方法。

Cortex M1 SoC 主要功能与性能情况如下所示：

- 支持32-bit Thumb-2 BL, MRS, MSR, ISB, DSB, and DMB指令集
- 三级流水线，哈弗结构，SoC主频100MHz-125MHz
- 支持AHB系统总线（开放），AHB转APB总线桥（不开放）
- 支持1、8、16、32个外部中断与NMI异常处理
- 支持ModelSim的通用加速仿真平台
- 支持16个GPIO（含16个GPIO输入中断）、2个UART（含接收中断）、2个TIMER（含中断）、1个WATCHDOG（含中断）、1个Master SPI(8个片选)、1个Master I2C（含中断）、1个MEM（软核可挂载16个最大16MB的RAM），支持Systick系统定时器，支持FreeROTS操作系统
- 支持可配置的ITCM与DTCM，默认4KB，最大1MB（当不使用CACHE时，请根据实际工程需求配置合适的ITCM与DTCM空间）
- 支持ICACHE，固定值8KB
- 支持DCACHE，固定值8KB
- 支持千兆以太网的RGMII通信，实现裸机Lwip与FreeROTS Lwip协议栈
- 支持千兆以太网MAC层的发送加速功能，发送最高速率为990Mbps，用户数据长度固定为1460字节（帧总长为42+1460字节）
- 支持Bootloader，系统上电后指令从SPI-FLASH加载至DDR颗粒中
- 支持用户挂载AHB外设或用户直接与软核外部寄存器/RAM数据交互

Cortex M1 Soc 系统结构如图 1 所示：

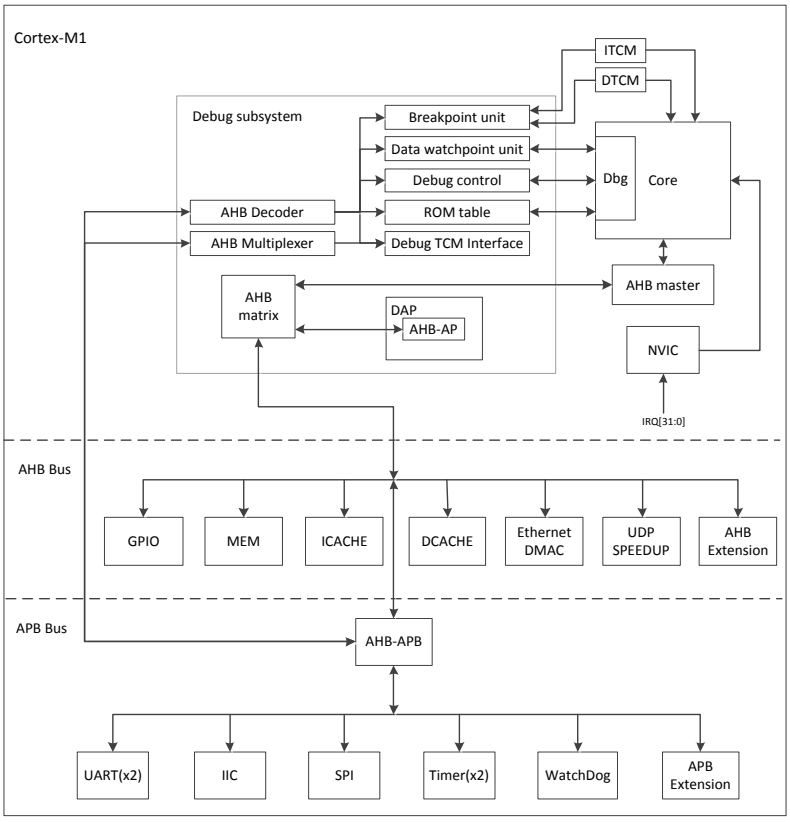

图 1 Cortex M1 SoC 系统结构图

2 KEIL 简介

Keil 是美国 Keil Software 公司出品的单片机 C 语言软件开发系统，通过一个集成开发环境（uVision）将 C 编译器、宏汇编、连接器、库管理和仿真调试器组合在一起。

2.1 软件安装

请参考 ARM Keil MDK 官网提供的《MDK Getting Started》和《uVision User's Guide》。也可百度参考 Keil 的安装教程。Demo 工程使用的 Keil 版本为 Keil V5.15.0。

2.2 工程模板

2.2.1 创建工程

打开 Keil MDK，选择菜单栏 Project 中 New uVision Project…，创建新的工程，如图 2 所示：

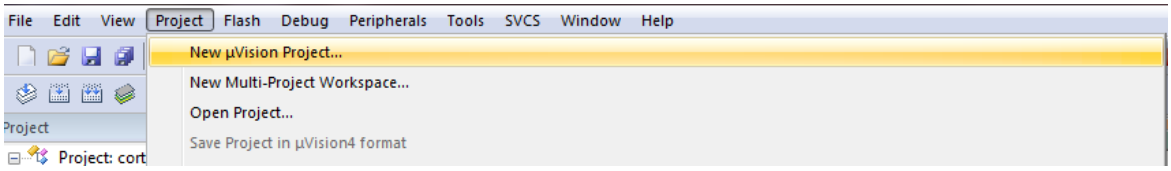

图 2 新建工程

2.2.2 器件选型

Cortex-M1 内置 ARM Cortex-M1 内核，所以器件选择 ARM Cortex-M1 的"ARMCM1"，如图 3 所示。

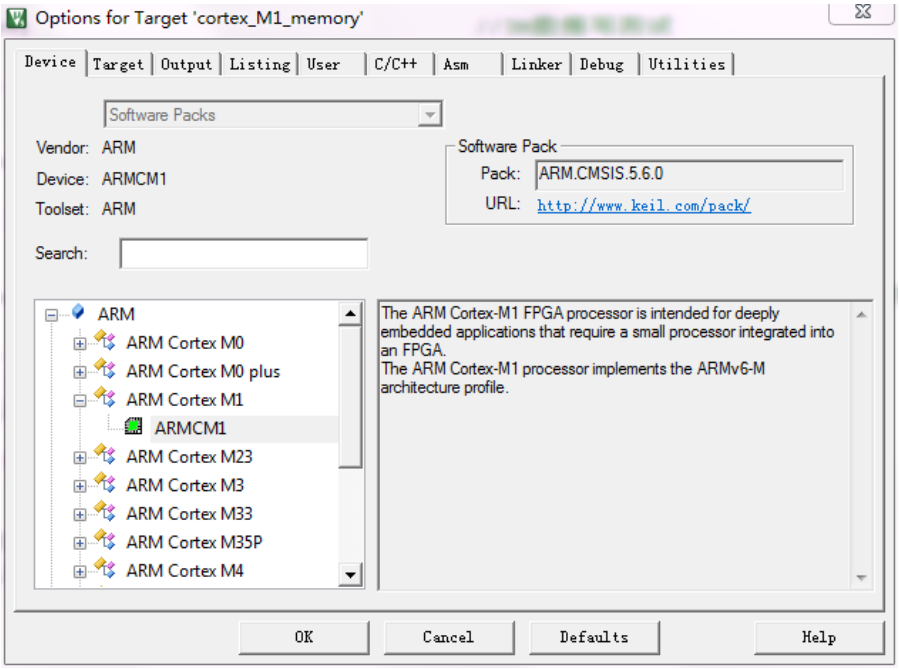

图 3 器件选择

2.2.3 配置 ROM 和 RAM

在不使用 CACHE 的 Cortex M1 中，ITCM 作为 ROM（起始地址为 0x00000000），DTCM 作为 RAM（起始地址为 0x20000000），大小根据用户实际需求进行设置。

在使用 CACHE 的 Cortex M1 中，ICACHE 作为 ROM（起始地址为 0x10000000），DCACHE 作为 RAM（起始地址为 0x30000000），大小根据用户实际需求进行设置。

ROM 和 RAM 的配置如图 4 所示。

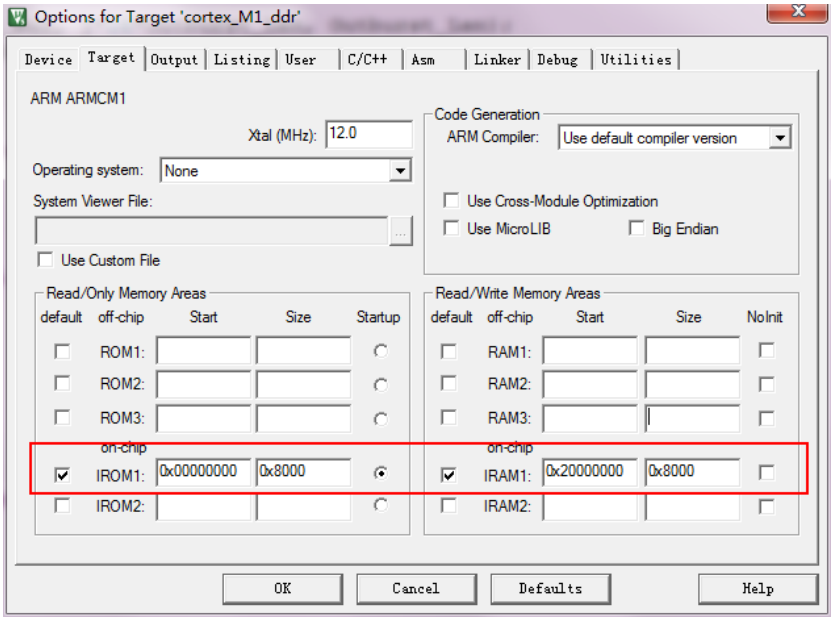

图 4 ROM 和 RAM 配置

2.2.4 配置输出 bin 文件

Cortex-M1 软件编程设计通过 Run #1 操作，将 axf 文件转换为 bin 文件格式并输出；Cortex-M1 软件编程设计通过 Run #2 操作，需要使用 make_hex.exe 工具将 bin 文件转换为 4 个 ITCM 文件格式并输出。具体的转换方法如图 5 所示。

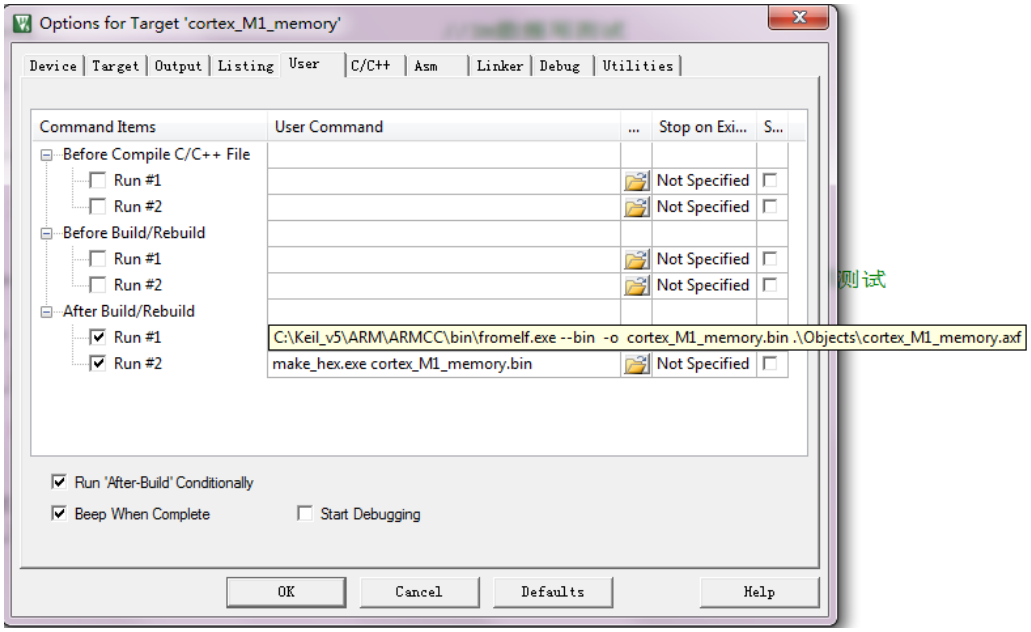

图 5 配置文件输出

2.2.5 配置头文件路径

配置头文件路径是为了编译时能够找到调用使用的头文件，其配制方法如图 6 所示。

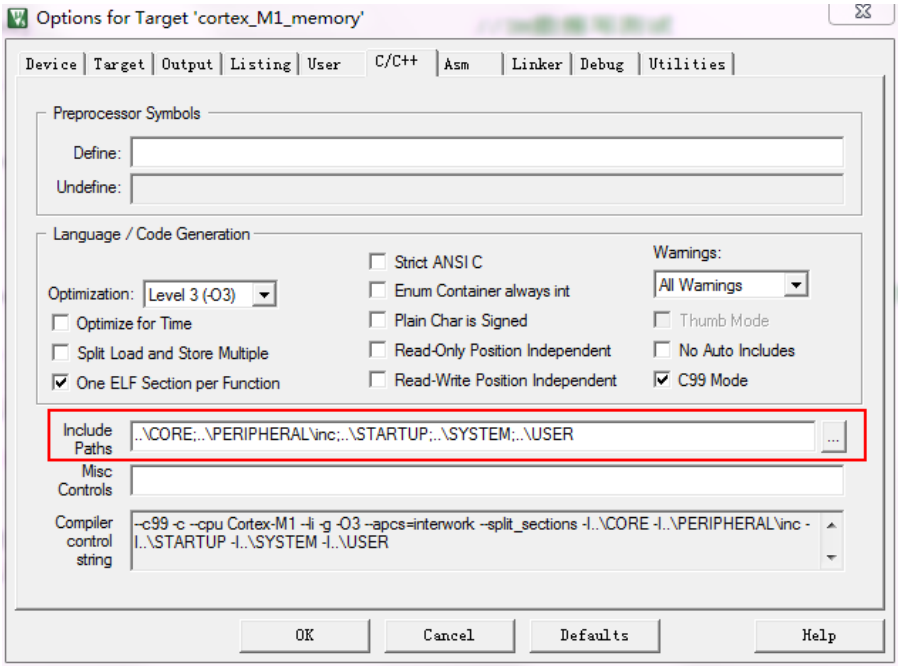

图 6 配置头文件路径

2.2.6 配置调试器选项

在线调试器使用野火(Fire Debugger)的调试器。

- 仿真器类型属于：CMSIS-DAP Debugger
- 调试器接口：JTAG 接口

配置 JTAG 调试器接口如图 7 所示。

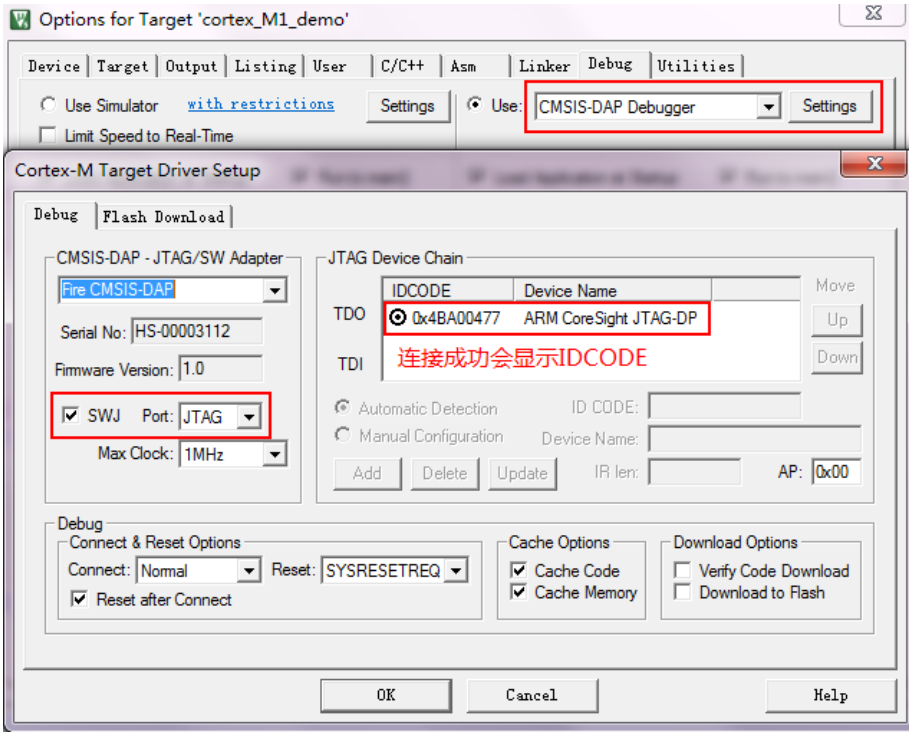

图 7 配置 JTAG 调试接口

2.2.7 配置 Flash 选项

在线调试时，不要选中“Update Target before Debugging”选项，具体操作如图 8 所示。

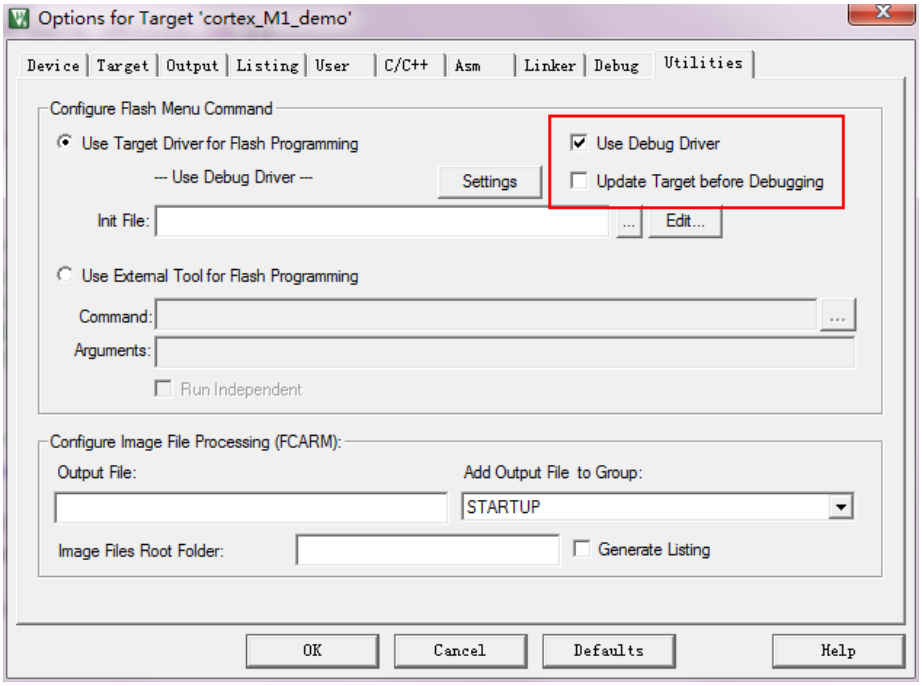

图 8 配置 Flash 选项

2.2.8 添加工程文件列表

在创建工程完成并且相关配置完成后，需要添加工程文件列表（即添加需要编译的*.c 文件），具体操作如图 9 所示。

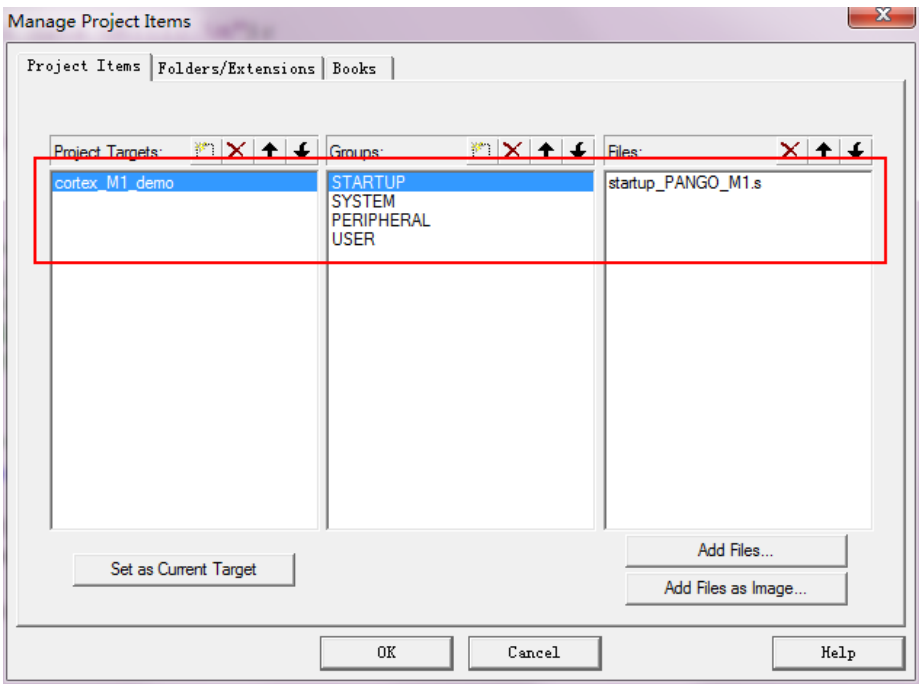

图 9 添加工程文件列表

2.2.9 工程编译

完成工程配置和工程文件导入以后，编译输出 bin 格式文件和 4 个 16 进制格式的文件，itcm0、itcm1、itcm2 和 itcm3，如图 10 所示。

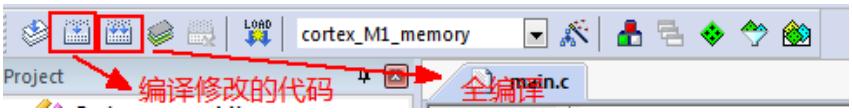

图 10 工程编译

2.2.10 软件调试

在线调试器使用野火(Fire Debugger)的调试器。正确连接调试器连线，点击在线调试按钮，进入调试模式。最多同时支持两个断点调试。调试按钮如图 11 所示。

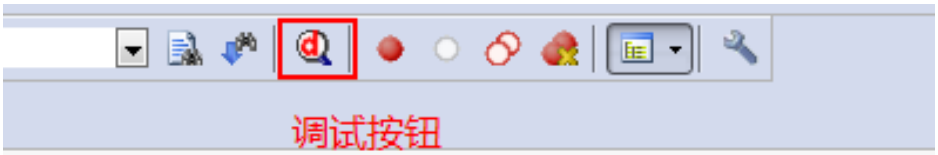

图 11 进入调试界面

3 示例 DEMO

带 ICACHE 和 DCACHE 的 Cortex M1 SoC 在使用时，需要引导应用程序（即通过 Bootloader 工程产生的 itcm0、itcm1、itcm2、itcm0 文件）将应用工程的指令（即由应用工程编译生成的 BIN 文件）装载到 DDR 颗粒中并实现地址跳转。

Bootloader 工程文件位于：../boot 文件夹下面，工程文件名：Cortex-M1_spi_flash_bootloader。

应用工程工程位于：../boot 文件夹下面，工程文件名：Cortex-M1_Icache_Dcache_Demo。

3.1 Bootloader 工程

双击 cortex_M1_bootloader.uvprojx （software_design\boot\Cortex-M1_spi_flash_bootloader\PROJECT）打开 Bootloader 工程，之后点击编译按键（  ），生成可执行文件 itcm0、itcm1、itcm2、itcm3（software_design\boot\Cortex-M1_spi_flash_bootloader\PROJECT）。

在此过程中需要注意两个地方：

其一，软核应用数据位流的 SPI Flash 起始地址与 PDS 合成软核应用数据位流的起始地址相同，默认 oxc0000，对应 Bootloader 软件代码的宏定义为：SPI_FLASH_START_ADD。

其二，需要配置应用数据位流的大小，与实际的应用位流大小相等，以使 Bootloader 的启动时间达到最优，默认 4096*64（即：支持软核可执行*.bin 文件大小 64KB），对应 Bootloader 软件代码的宏定义为：APP_BIN_SIZE。

3.2 应用工程

双击 cortex_M1_demo.uvprojx（software_design\boot\Cortex-M1_Icache_Dcache_Demo\PROJECT）打开应用工程，之后点击编译按键（  ），生成可执行文件 cortex_M1_demo.bin（software_design\boot\Cortex-M1_Icache_Dcache_Demo\PROJECT）。

在此过程中需要将指令和数据的寻址空间修改为 ICACHE 和 DCACHE 的寻址空间，如图 12 所示。

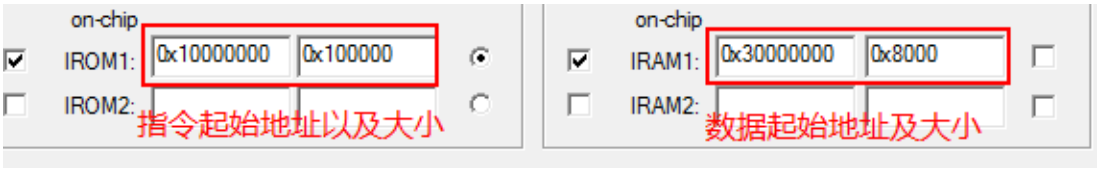

图 12 ICACHE 和 DCACHE 的寻址空间配置

ICACHE 指令起始地址必须配置为：0x10000000，DDR 颗粒映射范围 0x10000000-0x10ffffff（16MB）；DCACHE 数据起始地址必须配置为: 0x30000000，DDR 颗粒映射范围 0x30000000-0x3fffffff（256MB）。不能超过这个映射范围，否者会出错。

3.3 可执行文件使用

上述两个工程编译完成后，将 Bootloader 编译生成的 itcm0、itcm1、itcm2 和 itcm3 四个文件与硬件 RTL 工程一起编译，然后将硬件 RTL 工程生成的*.sbit 文件与应用工程比哪一生成的*.bin 文件拼接成*.sfc 文件，下载到外置的 SPI Flash 里面，其详细操作请参考 4.2 节。

3.4 仿真文件

为了节约硬件仿真时间，就需要对 DDR 颗粒模型进行初始化赋值操作，即跳过引导过程，因此需要利用 make_hex128.exe 可执行文件，生成 DDR 颗粒模型初始化所需的文件，该文件使用方法如图 13 所示。

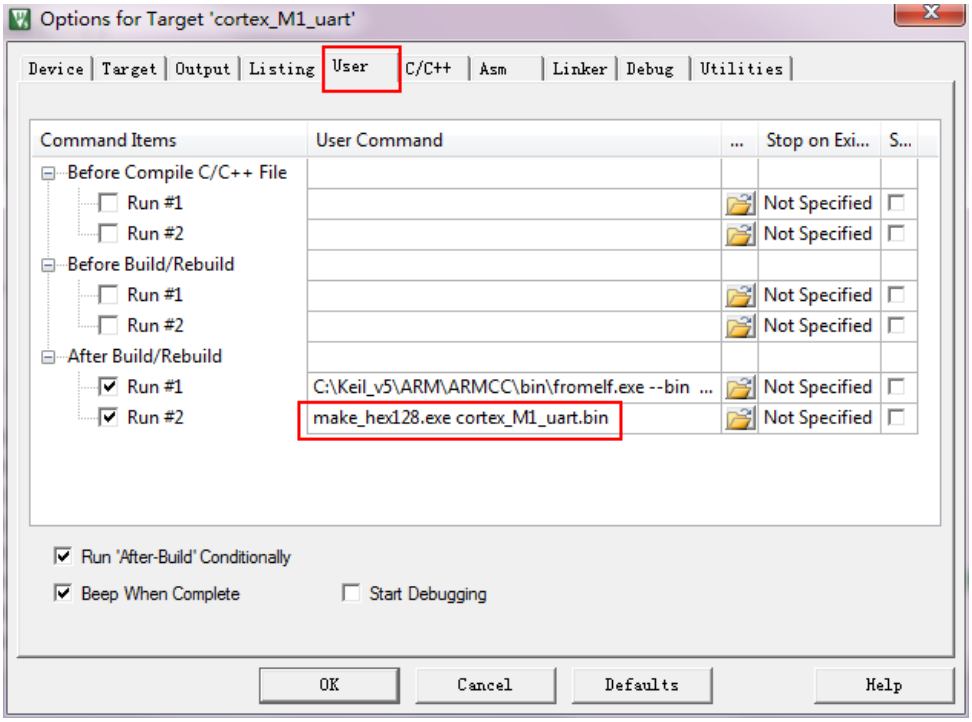

图 13 脚本替换操作

进入 Operstions->User，将原本的 make_hex.exe 脚本可执行文件改为 make_hex128.exe，点击 OK，然后进行编译，生成如图 14 所示的 mem_addr.dat、mem_data.dat 和 mem_used.dat 文件，这三文件的具体使用方法请参考 4.3 节所述。

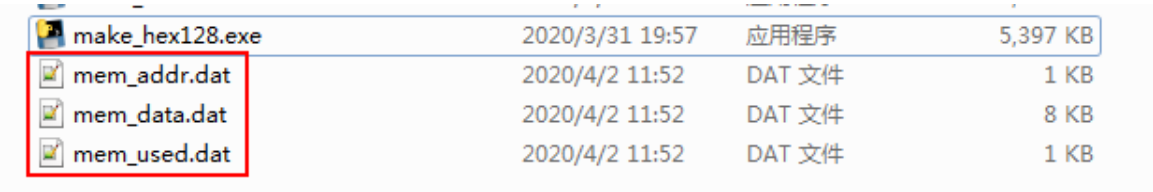

图 14 dat 文件生成

4 PDS 使用

本设计硬件平台选用的是含 PGL22G-FBG256 芯片的 PGL2KF01-A0 DEMO 板卡，以及与该 FPGA 芯片配套的 Pango Design Stutio 2020.2-SP2 开发工具（PDS）。

此外，硬件仿真平台采用的是无 Boot 过程的仿真方法，可实现指令与数据快速仿真，利于用户进行逻辑功能的快速验证与开发。

4.1 工程导入

将工程导入前需要先安装 Design Stutio 2020.2-SP2 开发工具，具体操作方法请参考 PANGO 官网提供的安装包中的安装文档说明与相关设置，在此不赘述。

安装成功之后，打开 PDS 软件，选中 File 下的 Open Project 选项，之后将 pgr_ARM_Cortex_M1_eval 文件夹中的 ARM_M1_SoC_TOP.pds 选中，最后点击 Open 即可完成工程导入操作，其操作如图 15 所示。

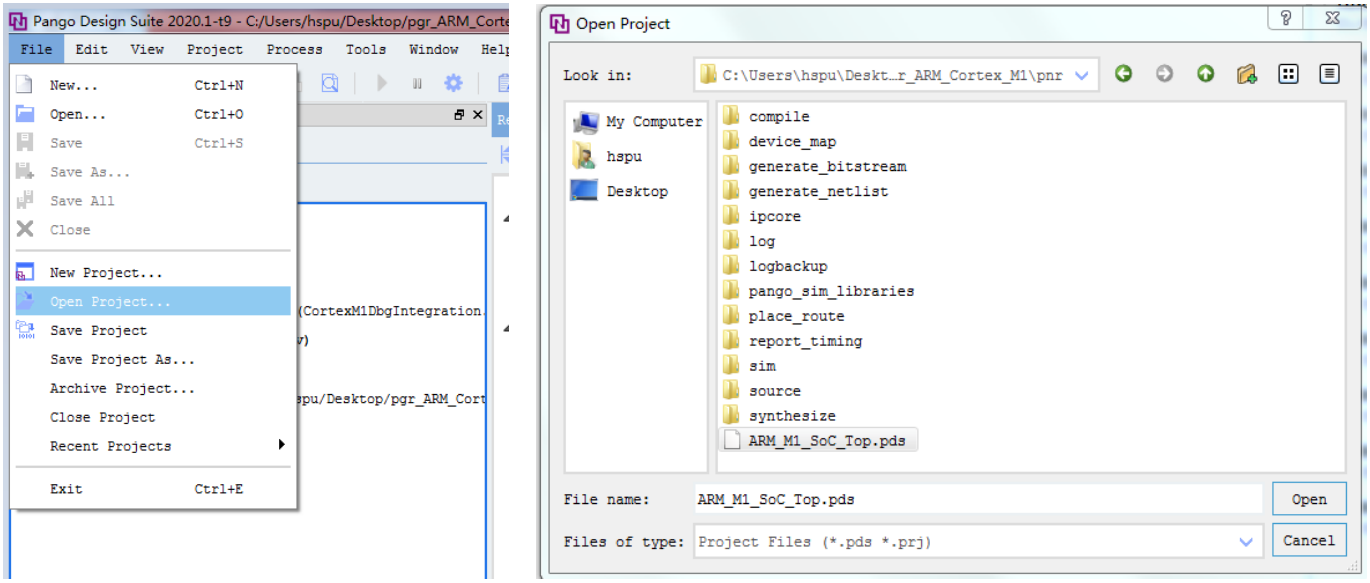

图 15 PDS 工程导入

4.2 Flash 固化

使用带 CACHE 的 Cortex M1 SoC 工程时，一定要将 Bootloader 生成的 itcm0、itcm1、itcm2、itcm3 文件放入该 PDS 工程的 rtl_design\pgr_ARM_Cortex_M1_eval\pnrr 目录下，与 RTL 工程一起编译，否则将不能实现应用程序的引导与地址跳转。

在 DEMO 板卡上运行用户的 C 设计代码时，需要将 PDS 工程所生成的*.sbit 文件与 KEIL 工程生成的*.bin 文件合成一个可固化进板卡 SPI-Flash 的*.sfc 文件，之后即可上电观察板级现象与打印信息。其合成步骤如下所示：

首先，点击 Flow 下的 Generate Bitstream，生成与导入工程匹配的数据流文件（即 m1_soc_top.sbit），操作方法如图 13 所示，同时进行 KEIL 工程编译，生成 bin 格式文件（即 cortex_M1_demo.bin），操作方法如图 16 所示；

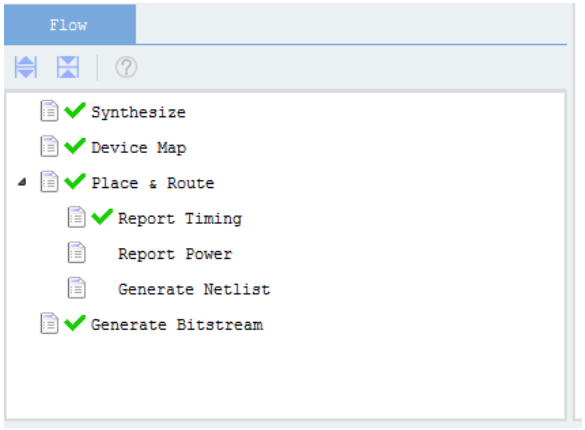

图 16 数据流生成图

其次，通过 PDS Configuration 插件工具，将 KEIL 所编译的 BIN 文件（默认起始地址为 0x000C0000）与 *.sbit 数据流文件拼接，操作方法如图 17 所示；

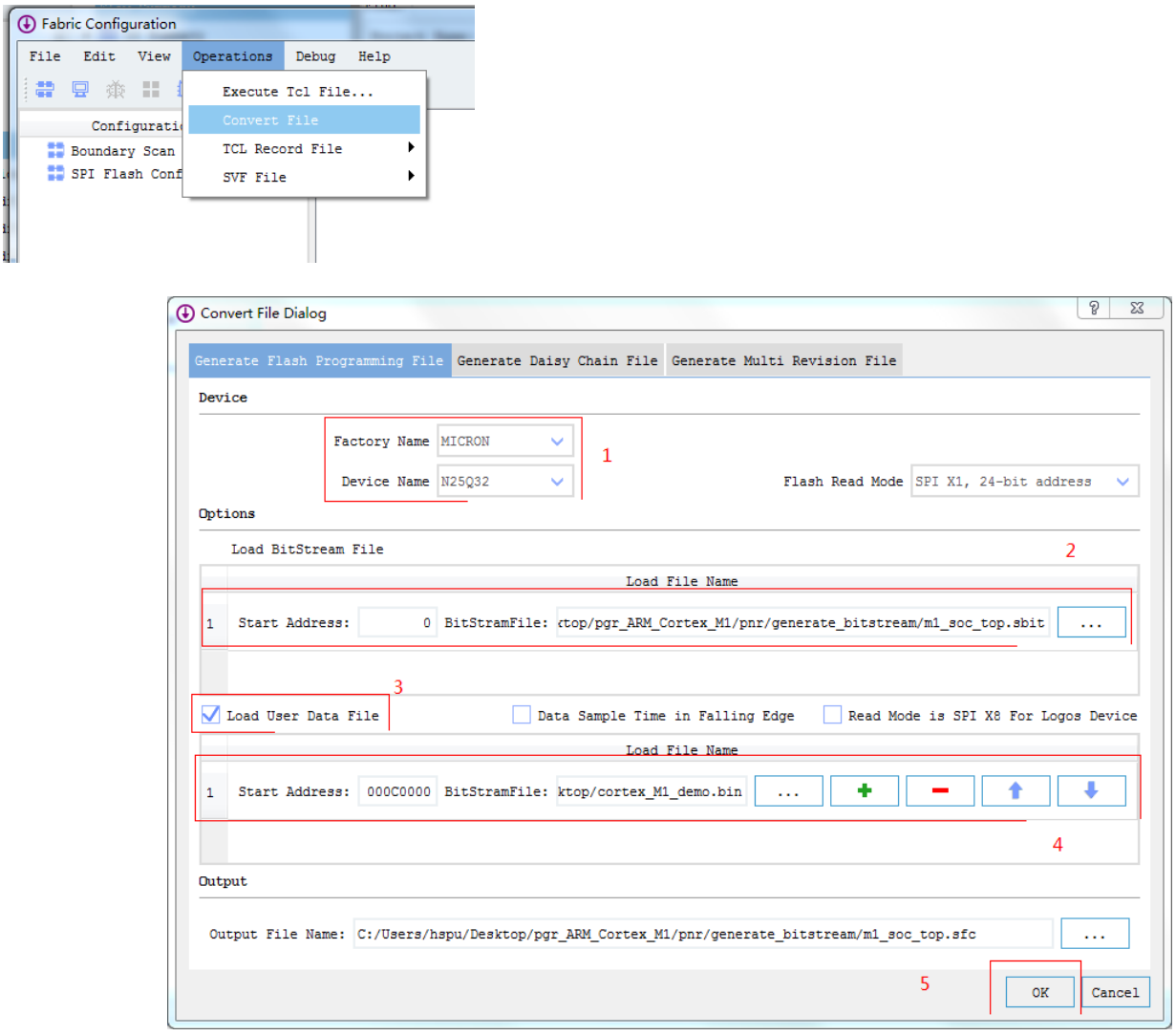

图 17 sfc 文件生成图

最后，利用 PANGO CABLE 与 Configuration 插件直接下载到 Flash 芯片中，即可观察板级现象与打印信息，操作方法如图 18 所示。

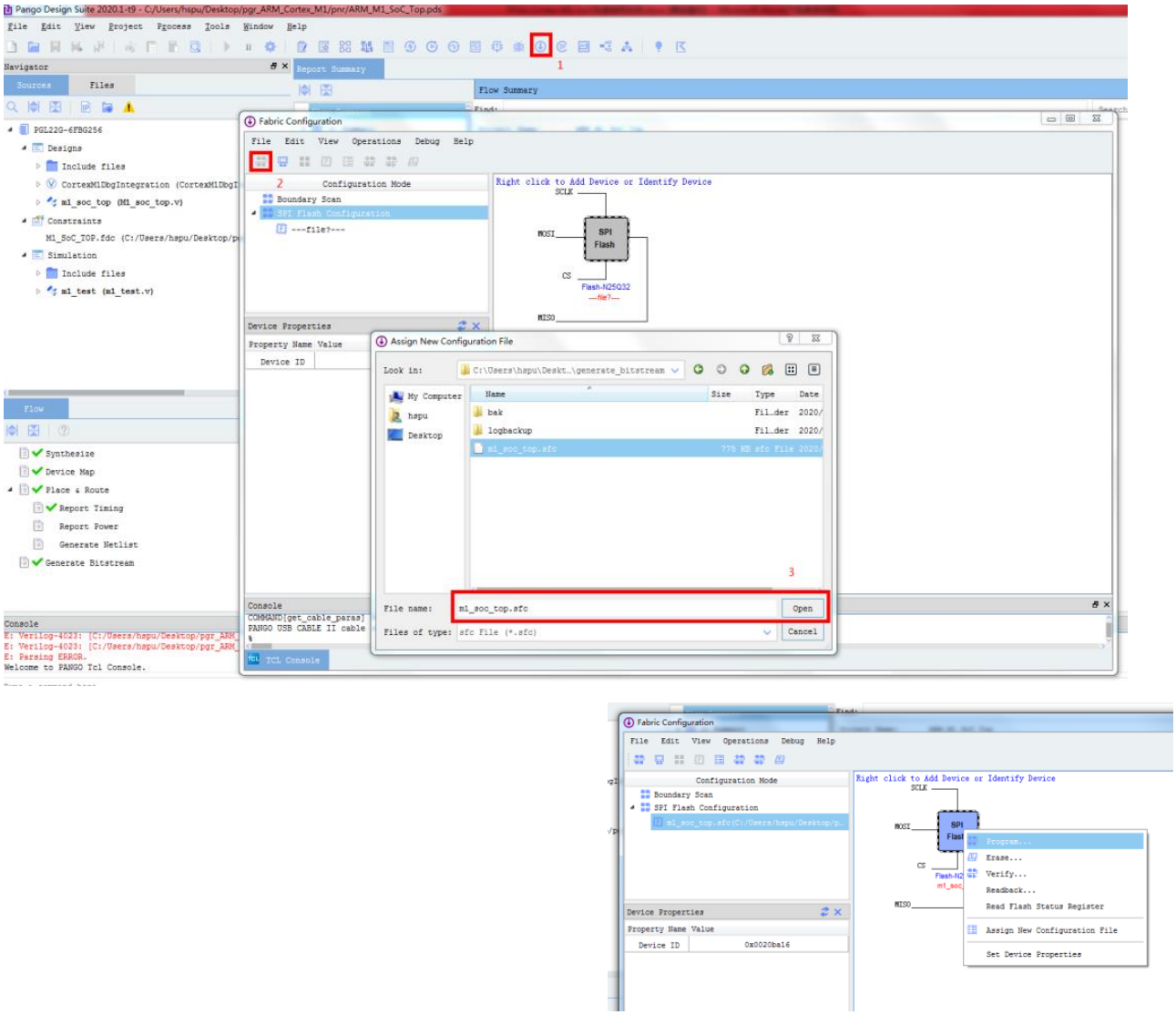

图 18 sfc 下载操作图

4.3 硬件仿真

Cortex M1 SoC 的硬件仿真平台采用的是无 Boot 的仿真方法，即跳过指令从 SPI-FLASH 中加载到 DDR 颗粒的过程，直接将所需指令初始化就进 DDR 颗粒的预定空间中，当系统初始化完成后就可进行指令与数据的加载。

整个行为级仿真过程是利用 ModelSim10.2C 实现的，具体操作方法如下：

首先，利用 make_hex128.exe，将 KEIL 所生成的 BIN 文件，转化为 memory.dat,memory_used.dat,address.dat 文件，具体生成操作请见 3.4 所述；

然后，将 memory.dat,memory_used.dat,address.dat 三个文件放入..\pgr_ARM_Cortex_M1_eval \simulation 中；

最后，双击 sim.bat，即可运行仿真程序。

注意，PDS 版本与手册中不一致，请更新仿真库并添加到..\pgr_ARM_Cortex_M1_eval \simulation 中，操作方法如下：

首先，点击 tool 下的 Compile Simulation Libraries，选择第三方仿真路径，最后点击 Compile，等待运行成功，其操作方法如图 19 所示。

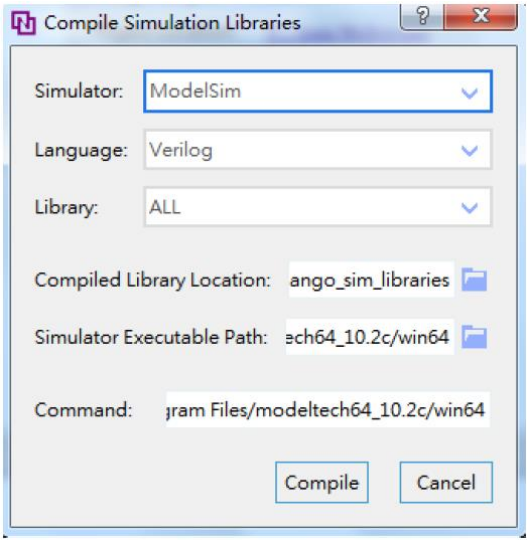

图 19 编译方法图

其次，将..\pgr_ARM_Cortex_M1_eval \pnr\pango_sim_libraries 中的 usim 文件夹拷贝至..\pgr_ARM_Cortex_M1_eval\simulation 中。

最后，更新仿真库成功，即可点击 sim.bat 正常使用仿真功能。

4.4 上板操作

第一步，按照第 3 章操作，准备 itcm0、itcm1、itcm2、itcm0 与 cortex_M1_demo.bin 文件；

第二步，将 DEMO 板卡连接 12V 电源与 PANGO CABLE；

第三步，串口与 PC 机连接，并打开串口助手，选择匹配的 COM 口；

第四步，按照 4.3 节操作，将 itcm0、itcm1、itcm2、itcm0 添加进 rtl_design\pgr_ARM_Cortex_M1_eval\pnr 目录下，之后通过 PDS 工具生成 m1_soc_top.sbit 文件；

第五步，按照 4.3 节操作，利用 PDS 的 Configuratin 插件，将 m1_soc_top.sbit 与 cortex_M1_demo.bin 合成的 m1_soc_top.sfc 文件固化至 SPI-Flash 中；

第六步，观察串口打印信息与板卡指示灯（Init、Done、User LED）现象，以及按键/串口中断，判断工程时候运行成功（Init、Done 全部点亮，User LED 闪烁，触发串口输入中断（打印串口助手输入的信息）和 GPIO 按键中断有对应的打印信息，即 key Interrupt ok 与 Uart interrupt ok）。

具体打印信息如下所示：

PANGO Cortex-M1 Start Run.....

JEDEC id = 0x20ba16

SPI Flash Start.....

SPI Read data OK!!!!

Timer interrupt OK 1.....

Key interrupt OK.....

Smemory Start.....

Get BANK_RAM_BASE0 Data = 10 11 12 13 14 15 16 17 18 19

Get BANK_RAM_BASE1 Data = 10 11 12 13 14 15 16 17 18 19

Get BANK_RAM_BASE2 Data = 10 11 12 13 14 15 16 17 18 19

Get BANK_RAM_BASE3 Data = 10 11 12 13 14 15 16 17 18 19

Timer interrupt OK 2.....

Uart interrupt ok

IIC Start.....

EEPROM irq cnt = 32!!!!

EEPROM irq cnt = 64!!!!

EEPROM irq cnt = 96!!!!

EEPROM irq cnt = 128!!!!

EEPROM Read data OK,EEPROM_SPEED = 400KHz!!!!

Timer interrupt OK 3.....