1 Gowin Cortex-M1 软核使用说明

Gowin 软件中提供有 Cortex-M1 软核给用户使用，对单片机熟悉的用户来说，Cortex-M1 并不陌生，本文档主要通过实验的方式带领读者学会使用 Gowin 软件提供的 Cortex-M1 进行软硬件编程，本次实验实现的功能就是通过 Cortex-M1 软核控制 GPIO 输出高低电平，从而控制 LED0 灯的亮灭。

1.1 硬件设计

1.1.1 Gowin 软件中添加 Cortex-M1 软核

我们首先在 Gowin 软件中新建一个工程，建立工程和文件的方法读者可以参考逻辑教程中的“Gowin 软件基本开发流程”一节的内容，这里将不再进行说明，我们这里新建一个名为“M1SC_5A_LED_Twinkle”的工程，然后点击“  ”新建一个 Verilog 文件，也命名为“M1SC_5A_LED_Twinkle”。

然后点击“  ”进入 IP Core Generator 界面搜索“m1”找到 Cortex-M1 软核“Gowin_EMPU_M1”，如下图 1-1 所示。

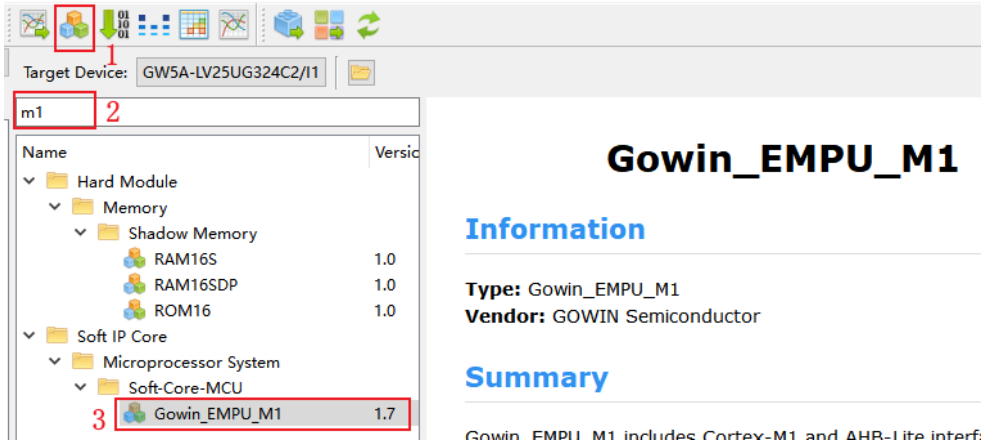

图 1-1 找到 Cortex-M1 软核

双击之后，进入配置界面，如下图 1-2 所示。

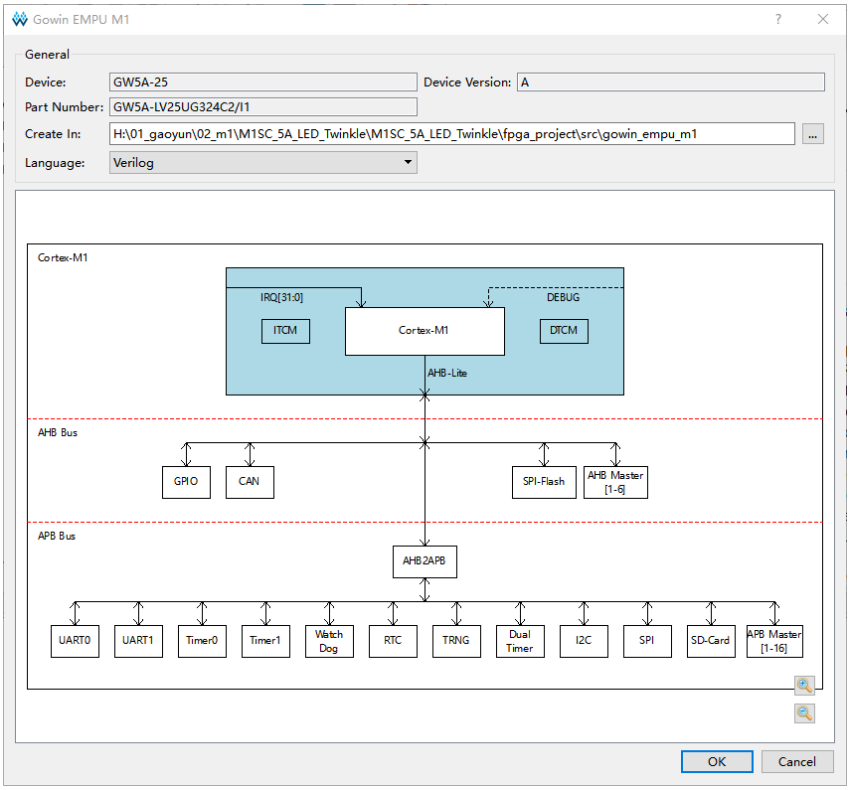

图 1-2 Gowin_EMPU_M1 IP 配置界面

从上图可以看出，Gowin_EMPU_M1 包括三级结构：

第一级，Cortex-M1 内核及 ITCM（指令存储器）、DTCM（数据存储器）；

第二级，AHB 总线及 GPIO、CAN、Ethernet、DDR3 Memory、PSRAM Memory、SPI-Flash Memory、AHB Master [1-6]；

第三级，APB 总线及 UART0、UART1、Timer0、Timer1、Watch Dog、RTC、TRNG、DualTimer、I2C Master、SPI Master、SD-Card、APB Master [1-16]。

1.1.2 Cortex-M1 软核配置

进入 Gowin_EMPU_M1 IP 配置界面之后，双击 Cortex-M1，即可对 Cortex-M1 内核系统进行配置，如下图 1-3 所示。

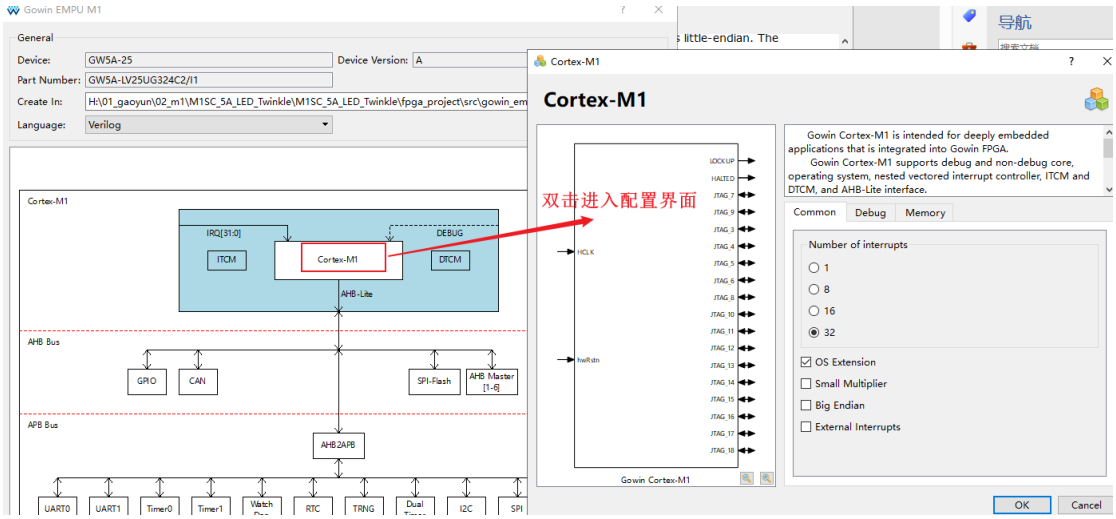

图 1-3 进入 Cortex-M1 内核系统配置

下面对 Cortex-M1 内核系统中各个配置进行说明。

1.1.2.1 通用配置

Cortex-M1 通用配置如下图 1-4 所示。

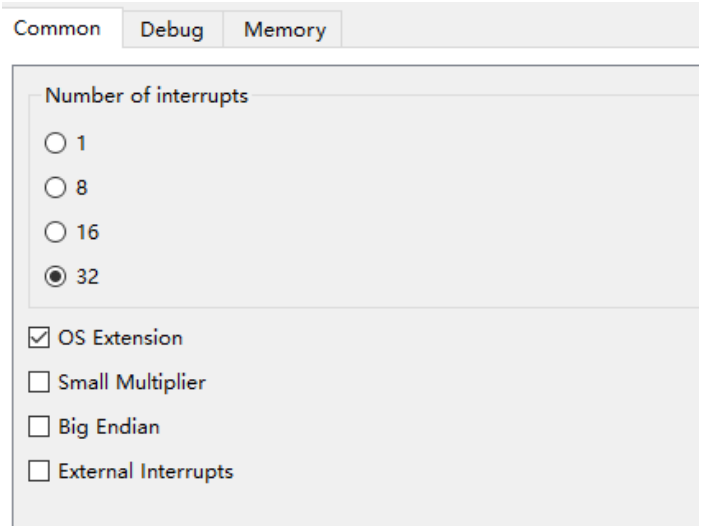

图 1-4 Cortex-M1 通用配置选项

上述各项配置说明：

- Number of interrupts：中断数量配置，可以配置 1 个、8 个、16 个或 32 个外部中断，默认为 32 个。
- OS Extension：操作系统扩展配置，如果选择，则 Cortex-M1 扩展支持操作系统，默认为支持操作系统扩展。
- Small Multiplier：乘法器模式配置，如果选择，则 Cortex-M1 支持 Small

乘法器，否则支持 Normal 乘法器，默认为 Normal 乘法器。

- Big Endian：数据存储格式配置，如果选择，则 Cortex-M1 支持数据大端格式，否则支持数据小端格式。
- External Interrupts：外部中断配置，如果选择，则 Cortex-M1 支持扩展的 4 个外部中断输入，否则不支持扩展的外部中断输入，默认为不支持。

1.1.2.2 调试配置

点击 Debug 进入调试配置界面，如下图 1-5 所示。

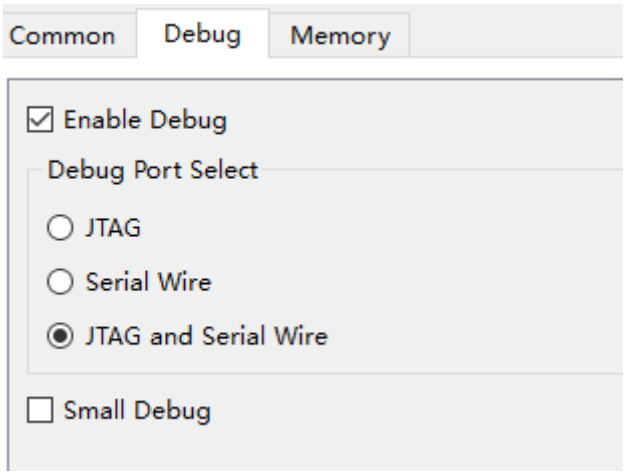

图 1-5 Cortex-M1 调试配置

对上述配置说明如下：

- Enable Debug：是否使能调试端口功能，勾选则使能，不勾选不使能。
- Debug Port Select：调试端口的选择，一共有三个选项可选：JTAG、Serial Wire、JTAG and Serial Wire。Serial Wire 就是“串行线调试”，是由两条线组成 SWDIO 和 SWCLK；JTAG Debug Port，也就是我们常用的 J-link 上面的调试端口（JTAG 模式下），本次实验我们使用 DAP Link 上的调试端口，也就是 Serial Wire 模式。
- Small Debug：调试模式配置，如果选择 Small Debug，则 Cortex-M1 支持 Small 模式调试器，否则支持 Full 模式调试器，默认为 Full 模式。

Debug 调试配置界面最终配置如下图 1-6 所示。

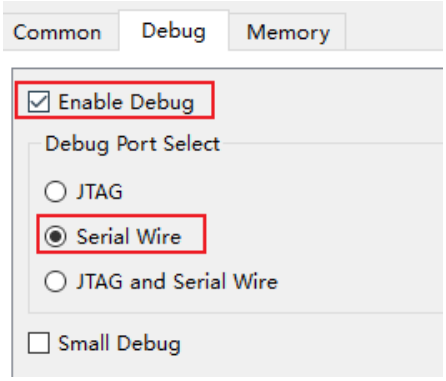

图 1-6 调试配置

1.1.2.3 存储配置

点击“Memory”进入存储配置界面，如下图 1-7 所示。

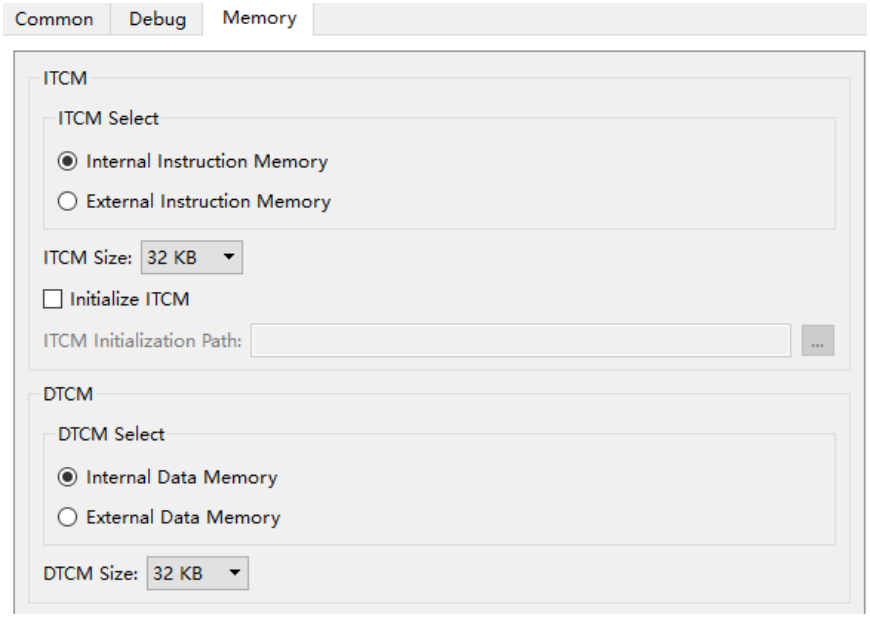

图 1-7 Cortex-M1 存储配置界面

对上述配置说明如下：

- ITCM Select：ITCM 配置选择，ITCM 是 Cortex 内核中指令传输总线，支持使用内部指令存储器“Internal Instruction Memory”或者外部指令存储器“External Instruction Memory”。
- ITCM Size：只有选择了“Internal Instruction Memory”，该配置才能被启用，可以选择为 1KB、2KB、4KB、8KB、16KB、32KB、64KB、128KB、256KB 或 512KB；根据不同的芯片类型支持的最大 Size 都是不一样的，GW5A-25 的 ITCM Size 最大选择为 64KB，默认 32KB。

- ITCM Initialization：ITCM 初始化配置，只有在勾选了“Initialize ITCM”的情况下，该配置才能被启用，如果选择 Initialize ITCM，则支持 ITCM 初始化，可以在 ITCM Initialization Path 导入 ITCM 初始值文件路径。如果选择使用片外 SPI-Flash Memory 下载引导方式，ITCM 初始值根据不同的 ITCM Size 导入不同的 bootload 文件路径。注意 ITCM Initialization Path 导入的文件路径中，不能有以数字命名或“\r”、“\n”等转义字符的文件夹路径，本次实验这里勾选使用“Initialize ITCM”，具体的初始化文件的获取将会在后面的内容进行说明，
- DTCM Select：DTCM 配置，可以选择“Internal Data Memory”或者“External Data Memory”，默认“Internal Data Memory”。Internal Data Memory：内部数据存储器，片内 Block RAM 硬件存储资源，起始地址 0x20000000。External Data Memory：外部数据存储器，如 DDRx Memory 等，起始地址 0x20100000。
- DTCM Size：当选择了“Internal Data Memory”，该选项才可配置，可以选择 1KB、2KB、4KB、8KB、16KB、32KB、64KB、128KB、 256 KB 或 512KB；不同芯片型号，支持的最大选择不同，GW5A-25 最大支持为 64KB，默认 32KB。

ITCM 是 Cortex 内核中指令传输总线，DTCM 是 Cortex 内核同 flash 及 sram 之间传输指令和数据的通道，指令的取指和执行及数据的读写在性能及管理上存在差异性，由于是高速缓存，所以这两块内存区域被当做特殊的用途。比如某些对时间要求非常严格的代码，就可以被放到 ITCM 中执行。这可以有效地提高运行速度。某些需要频繁存取的数据，也可以放到 DTCM 中以节省存取时间。本次实验将代码放到 ITCM 中去执行，本次实验的 Cortex-M1 存储配置如下图 1-8 所示。

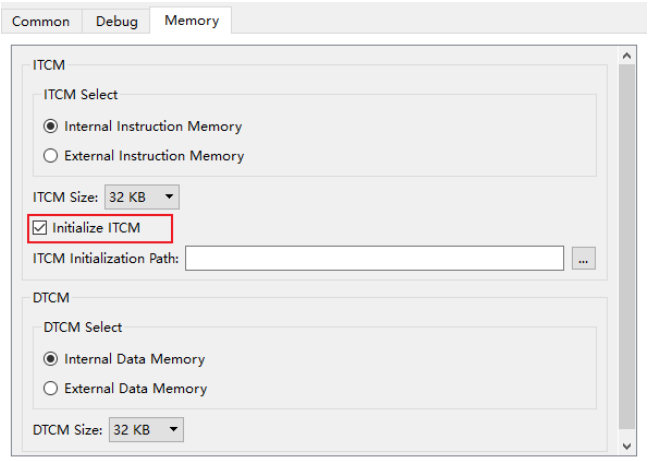

图 1-8 Cortex-M1 存储器配置示意图

1.1.3 添加 GPIO 外设

在 Gowin_EMPU_M1 IP 的配置界面可以看到，该 IP 支持多种外设，AHB 总线上：GPIO、CAN、Ethernet、DDR3 Memory、PSRAM Memory、SPI-Flash Memory、AHB Master [1-6]；APB 总线上：UART0、UART1、Timer0、Timer1、Watch Dog、RTC、DualTimer、TRNG、I2C Master、SPI Master、SD-Card、APB Master [1-16]。双击便可以对外设进行使能配置，本次实验以 GPIO 为例，双击进入 GPIO 配置界面，如下图 1-9 所示。

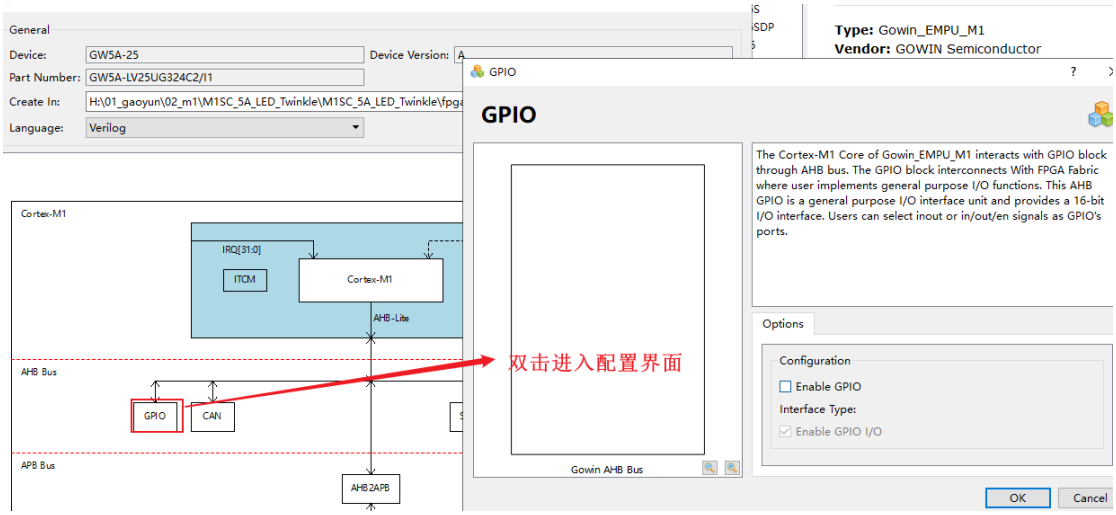

图 1-9 进入 GPIO 外设配置界面

进入配置界面之后，我们只需要使能 GPIO 即可，如下图 1-10 所示。

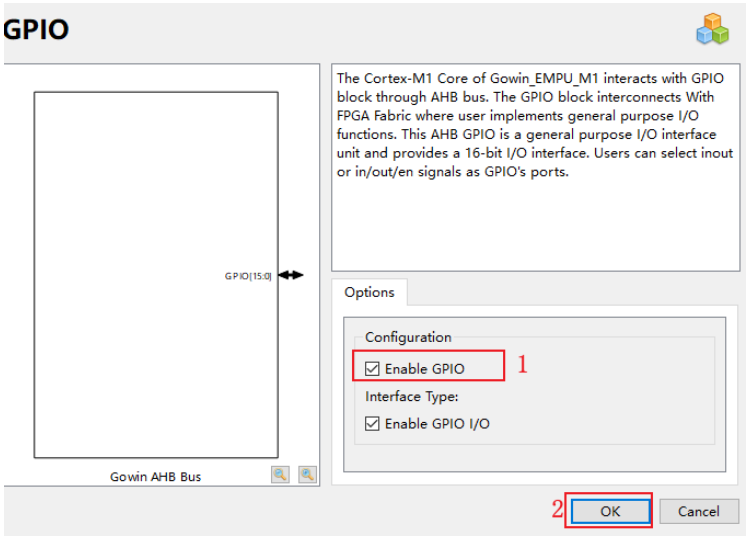

图 1-10 使能 GPIO 外设

点击 OK 之后，我们便可以看到 GPIO 选项变成了绿色，这也就表示外设配

置成功，如下图 1-11 所示，然后点击 OK，Gowin_EMPU_M1 IP 配置就完成了。

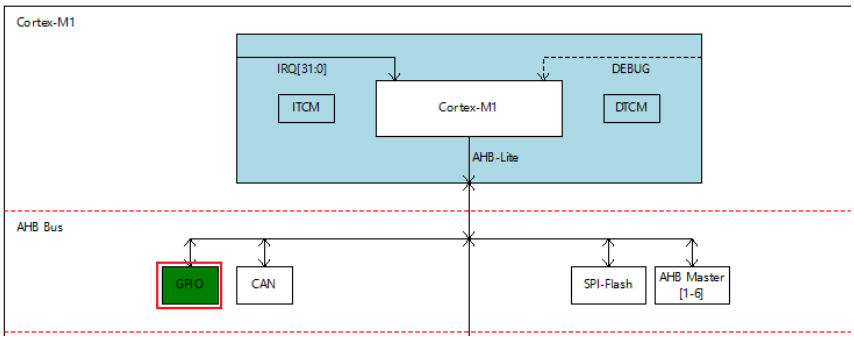

图 1-11 外设配置成功

1.1.4 完成顶层模块代码设计

我们将 Gowin_EMPU_M1 IP 添加至顶层模块“M1SC_5A_LED_Twinkle”中，并将对应端口引出，最终 M1SC_5A_LED_Twinkle 文件中代码如下所示。

```
module M1SC_5A_LED_Twinkle(
    inout [15:0] GPIO,
    inout JTAG_7,
    inout JTAG_9,
    input HCLK,
    input hwRstn
);
Gowin_EMPU_M1_Top Gowin_EMPU_M1_Top_Inst0(
    .GPIO(GPIO),          //inout [15:0] GPIO
    .JTAG_7(JTAG_7),      //inout JTAG_7
    .JTAG_9(JTAG_9),      //inout JTAG_9
    .HCLK(HCLK),          //input HCLK
    .hwRstn(hwRstn)       //input hwRstn
);
endmodule
```

1.1.5 引脚约束

本次实验 Cortex-M1 的工作时钟 HCLK 设置为 50M，由开发板上的晶振提供，复位 hwRstn 由开发板上按键控制，由于我们 ACG525 开发板上没有专门设置关于 Cortex-M1 的调试接口，我们可以使用开发板上 40 pin 扩展 GPIO 的 GPIO[0]和 GPIO[1]来实现调试下载，其中 GPIO[0]连接 DAP-Link 上的 SWDIO，GPIO[1]连接 DAP-Link 上的 SWCLK。本次实验我们需要通过 Cortex-M1 软核控制 GPIO 输出高低电平，从而控制 LED0 灯的亮灭，所以我们只需要分配 Cortex-M1 GPIO 外设的 GPIO[0]分配到 LED0 上即可。综上所述，可以得到本次实验的引脚分配表如下表 1-1 所示。

表 1-1 引脚分配表

引脚名称	引脚编号
HCLK	T9
hwRstn	C15
JTAG_7	C17
JTAG_9	C18
GPIO[0]	D14

引脚分配完成之后，对工程进行全编译，无错误无警告，我们便可以进行软件编程了。

1.2 软件代码设计

高云支持使用 ARM Keil MDK 或者他们自己的 CMD 软件进行软件代码设计，本次实验我们将带领读者使用 ARM Keil MDK 进行软件设计。关于 MDK 软件的安装和工程的建立，学习过单片机都会，如果读者对 MDK 软件使用不熟悉请自行查找资料学习相关内容，这里我们将不做详细介绍，本次实验建立的 MDK 工程名为 led。

Gowin 针对 Gowin_EMPU_M1 提供软件编程库，读者可以在我们提供的例程中获取，也可以在高云官方网站上下载，下载方式如下图 1-12 所示。

图 1-12 下载软件编程库

在 MDK 软件中将工程建立完成之后，将提供的库文件添加至工程中，然后建立 main.c 文件，然后编写本次实验需要的代码，如下所示：

```
/* Includes ------------------------------------------------------------
--------*/
#include "GOWIN_M1.h"

/* Declarations --------------------------------------------------------
--------*/
void GPIOInit(void);
```

```
void delay(__IO uint32_t nCount);

/* Definitions ---------------------------------------------------------
--------*/
int main(void)
{
  SystemInit(); //Initializes system clock
  GPIOInit();       //Initializes GPIO0

  while(1)
  {
    GPIO_WriteBits(GPIO0,0x0);
    delay(433300);

    GPIO_WriteBits(GPIO0,0x1);
    delay(433300);
  }
}

//Initialize GPIO
void GPIOInit(void)
{
  GPIO_InitTypeDef GPIO_InitType;

  GPIO_InitType.GPIO_Pin = GPIO_Pin_0 |
                           GPIO_Pin_1 |
                           GPIO_Pin_2 |
                           GPIO_Pin_3;
  GPIO_InitType.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitType.GPIO_Int = GPIO_Int_Disable;

  GPIO_Init(GPIO0,&GPIO_InitType);

  GPIO_WriteBits(GPIO0,0xF);//light : low level
}

//delay
void delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}
```

上述代码也非常好理解，首先初始化时钟，然后初始化 GPIO 口，然后在 while 循环中通过 GPIO_WriteBits 函数控制对应 GPIO 口输出高低电平，使用的是 GPIO0，并通过延时实现 LED 灯闪烁的功能。

1.2.1 MDK 软件配置

我们需要在 MDK 中进行相关设置才能启动 Cortex-M1 中程序的运行。我们在前面讲解 Gowin_EMPU_M1 IP 配置的时候，我们提到使用 ITCM 的方式，将代码放到 ITCM 中去执行，我们需要对 MDK 进行设置相关命令参数使其生成需要的 itcm 文件。点击  ，对 MDK 工程进行相关配置。

1. 设置 ROM 的初始地址为 0x0

找到 Target 选项，将片上 IROM1 的起始地址设置为 0x0，如下图 1-13 所示。如果这里不修改为 0x0，Coretx-M1 软核无法启动。

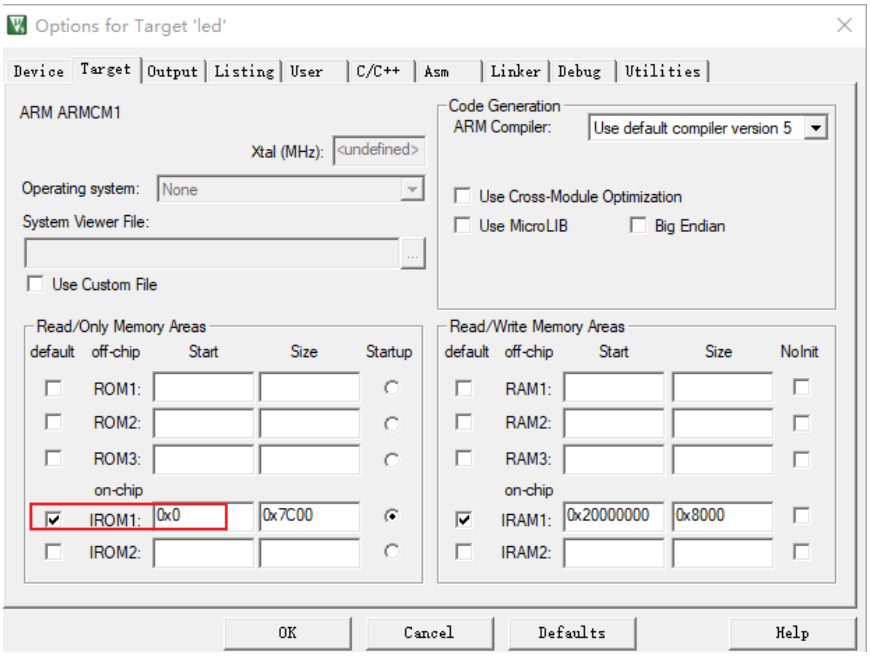

图 1-13 设置 ROM 初始地址

2. 输出文件设置

找到 User 选项，在 After Build/Rebulid 选项下进行设置，添加如下命令：

```
H:\keil\ARM\ARMCC\bin\fromelf.exe --bin  -o  led.bin .\Objects\led.axf
ITCM\make_hex.exe led.bin
```

需要注意的是，上述命令中的“H:\keil\ARM\ARMCC\bin\fromelf.exe”需要根据读者电脑上 Keil 的安装路径进行修改，make_hex.exe 软件是用来生成 ITCM 文件的，这个软件读者可以在我们提供的例程中获取，如下所示。

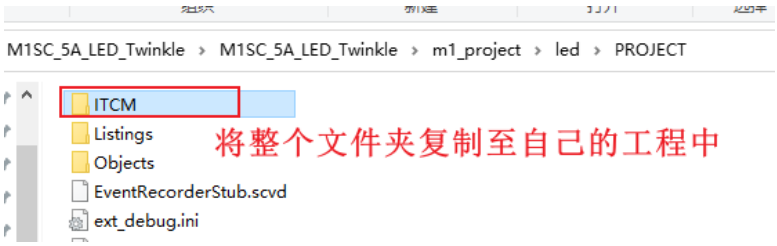

图 1-14 复制 make_hex.exe 软件至自己的工程

1.2.2 编译工程

经过上述操作之后，点击，编译整个工程，直至整个工程无错误，如下图 1-15 所示。

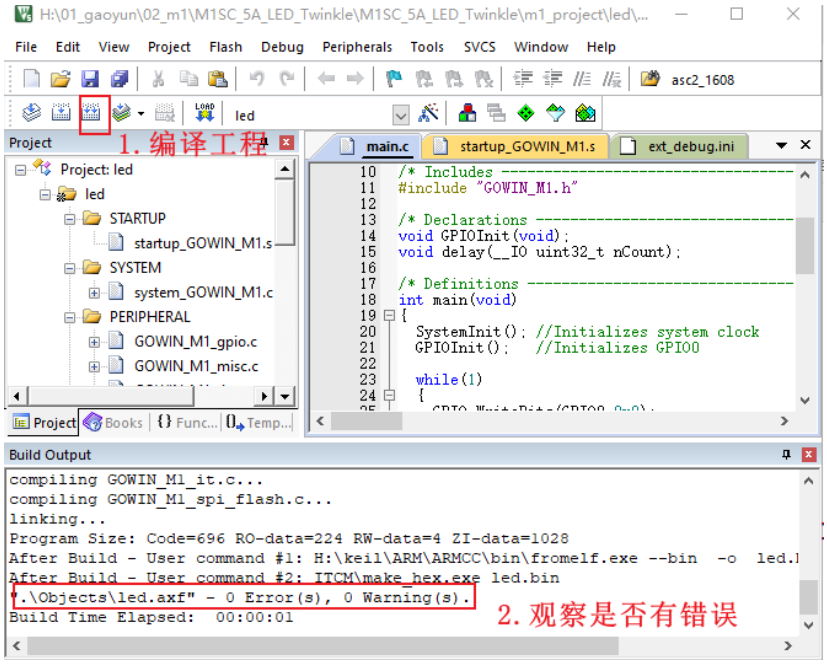

图 1-15 编译工程

通过上述操作，对于 Gowin Cortex-M1 软核的软硬件编程就完成了，接下来就是程序的下载和验证。

1.3 程序下载及功能验证

本章将带领读者使用两种方式下载，第一种就是 Gowin 软件和 MDK 软件分开下载，这种方式适合于程序开发环节，需要调试的时候使用；第二种就是将 MDK 软件中生成的 itcm 文件放至 Cortex-M1 软核中，然后编译工程，将生成的 fs 文件下载至 FPGA 芯片中，这种方式适合程序开发完成，代码已经测试没

有问题，需要量产的时候使用。

1.3.1 MDK 与 Gowin 软件分开下载

在使用 MDK 软件下载程序的时候，需要使用到 DAP-Link 调试器或者其它调试器，本次实验使用 DAP-Link，将 DAP-Link 的 GND、SCK、SWD 三个引脚连接至开发板上，前面我们提到过，开发板上没有专用的调试接口，使用的是 40-pin 拓展接口的 GPIO[0]、GPIO[1]，所以我们用三根杜邦线将 DAP-Link 和 ACG525 开发板进行连接，然后还需要使用高云下载器下载 FPGA 程序，所以本次分开下载的硬件连接图如下图 1-16 所示。

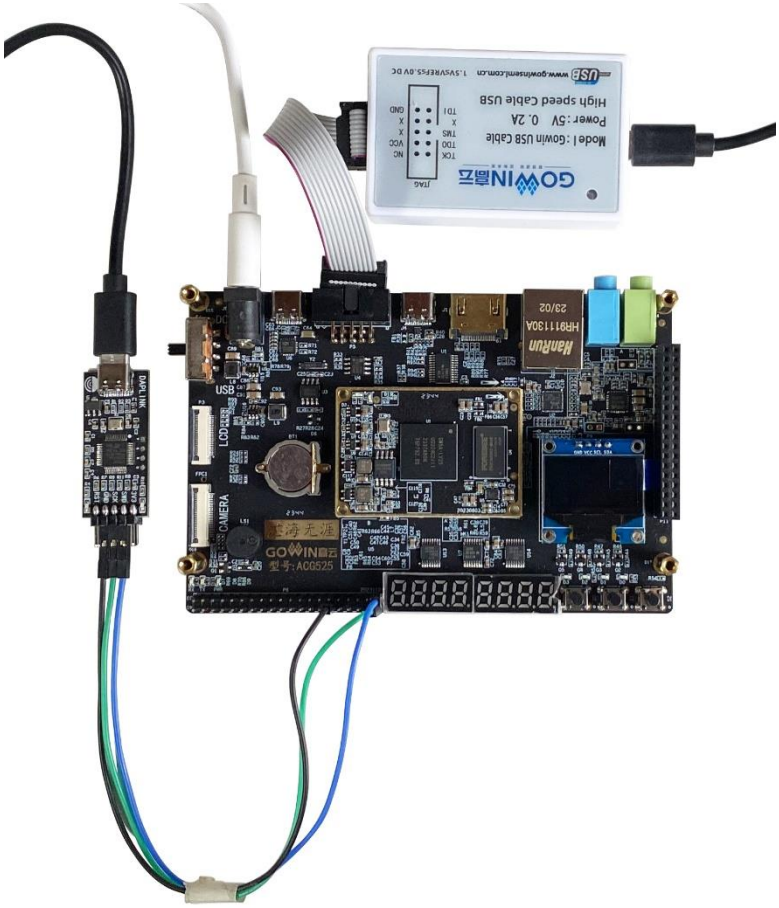

图 1-16 分开下载硬件连接图

1.3.1.1 下载 FPGA 侧代码

需要注意的是，分开下载程序的时候，必须先下载 FPGA 侧的代码，然后下载 ARM 侧的代码，顺序如果反了，在 MDK 软件中将会检测不到调试器。我们这里打开 FPGA 侧的工程点击 Programmer 下载程序，如下图 1-17 所示。

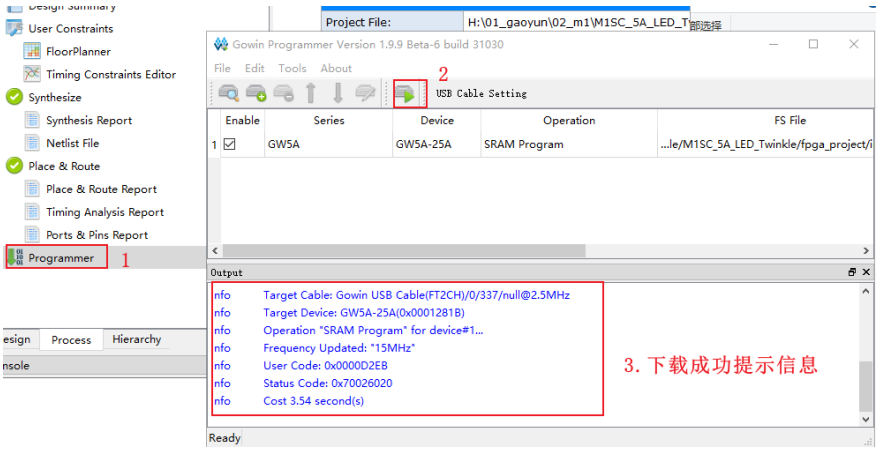

图 1-17 下载 FPGA 侧代码

1.3.1.2 下载 ARM 端代码

在 MDK 软件中，首先点击 进入 Debug 选项，找到 Use 选项，选择自己使用的调试器，我们使用的 DAP-Link，这里就需要选择为“CMSIS-DAP Debugger”，如下图 1-18 所示。

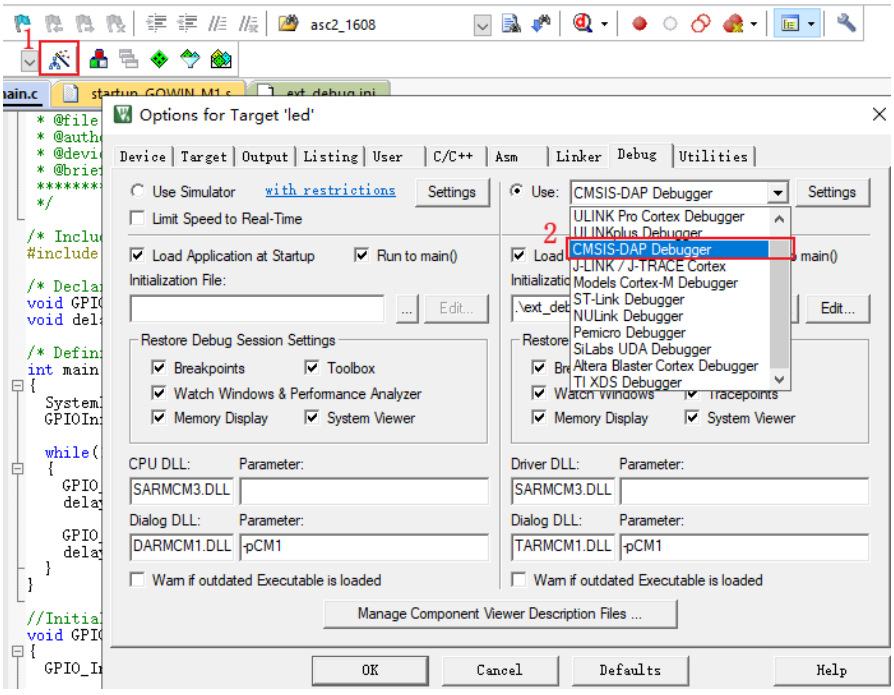

图 1-18 选择自己使用的调试器

然后点击 Settings，进入设置界面，我们就可以看到软件检测到了调试器，如下图 1-19 所示，表示此时我们可以下载程序了。

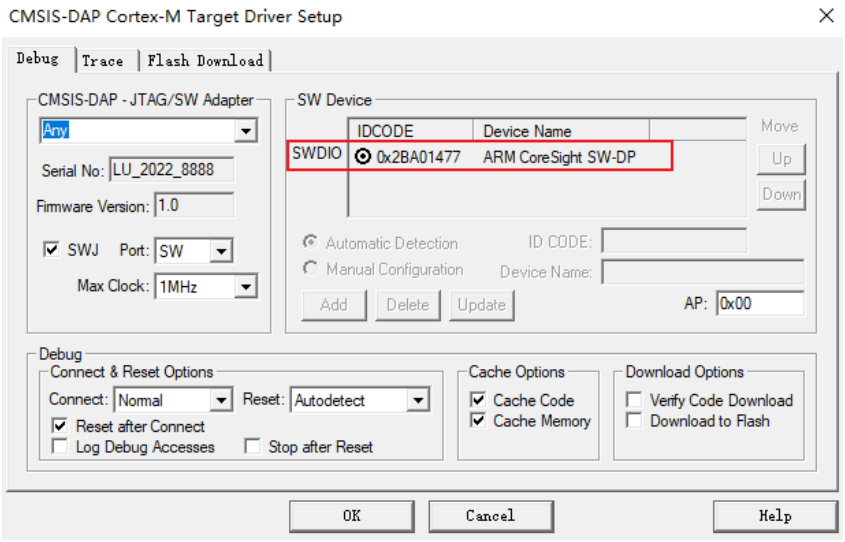

图 1-19 软件检测到了调试器

然后点击 进入调试界面，然后点击 ，让程序开始运行，如下图 1-20 所示。

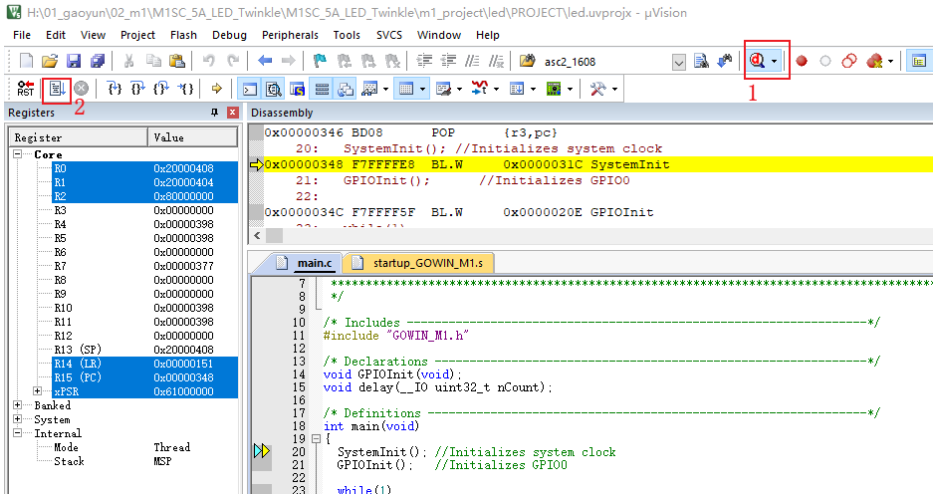

图 1-20 ARM 端程序下载

程序运行起来之后，可以看到开发板上的 LED0 开始闪烁，代表本次实验成功。

1.3.2 Gowin 软件下载完整工程代码

前面我们提到过，我们可以将 ARM 端的代码放到 ITCM 中去执行，也就是将 ARM 的代码作为初始化文件在 Cortex-M1 中进行配置，从而就可以实现脱离调试，直接通过 Gowin 软件下载完整的工程代码，具体操作如下所示。

1. 通过前面我们在 MDK 中的配置，可以看到在 MDK 工程目录下，生成

了 4 个 ITCM 文件，将这 4 个文件进行复制。

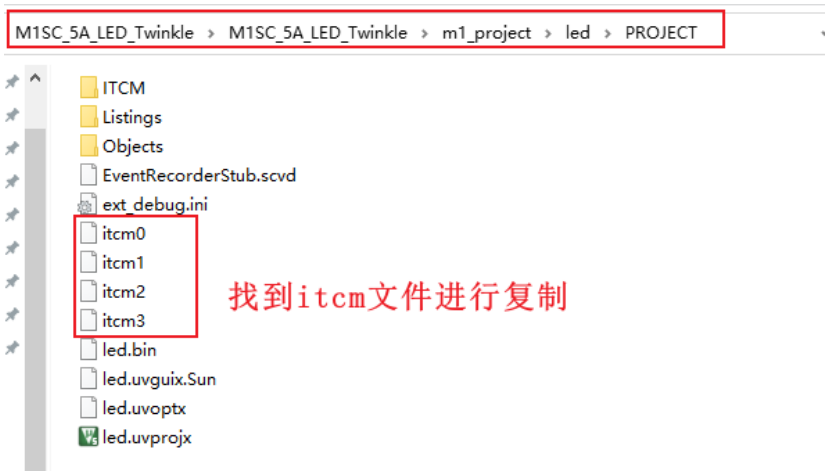

图 1-21 复制 ITCM 文件

2. 在 FPGA 工程的 Cortex-M1 IP 目录下新建一个 itcm 文件夹，将刚刚复制的 4 个 itcm 文件放至该文件夹下，如下图 1-22 所示。

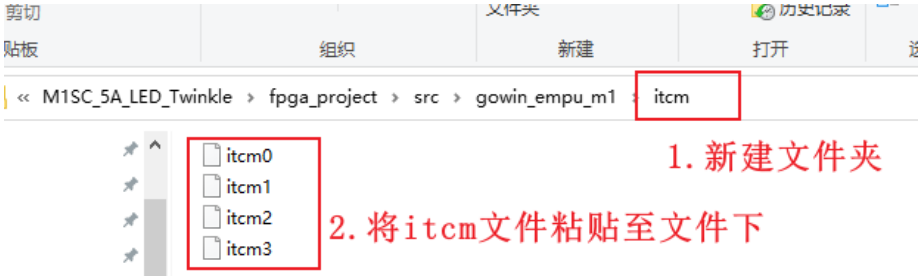

图 1-22 将 itcm 文件放置 FPGA 工程中

3. 在 Gowin 软件的 IP Core Generator 中点击  ，找到之前建立的 Cortex-M1 软核，如下图 1-23 所示。

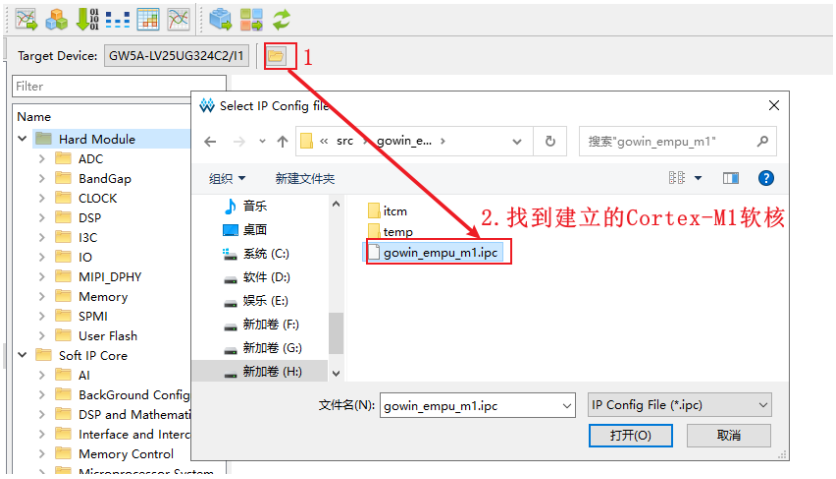

图 1-23 找到建立的 Cortex-M1 软核 IP

4. 点击 gowin_empu_m1.ipc 文件，进入 Cortex-M1 配置界面，在 Memory 配置界面下，找到 ITCM 初始化文件配置，点击 ... 找到我们刚刚建立的存放 itcm 文件的文件夹路径，如下图 1-24 所示。

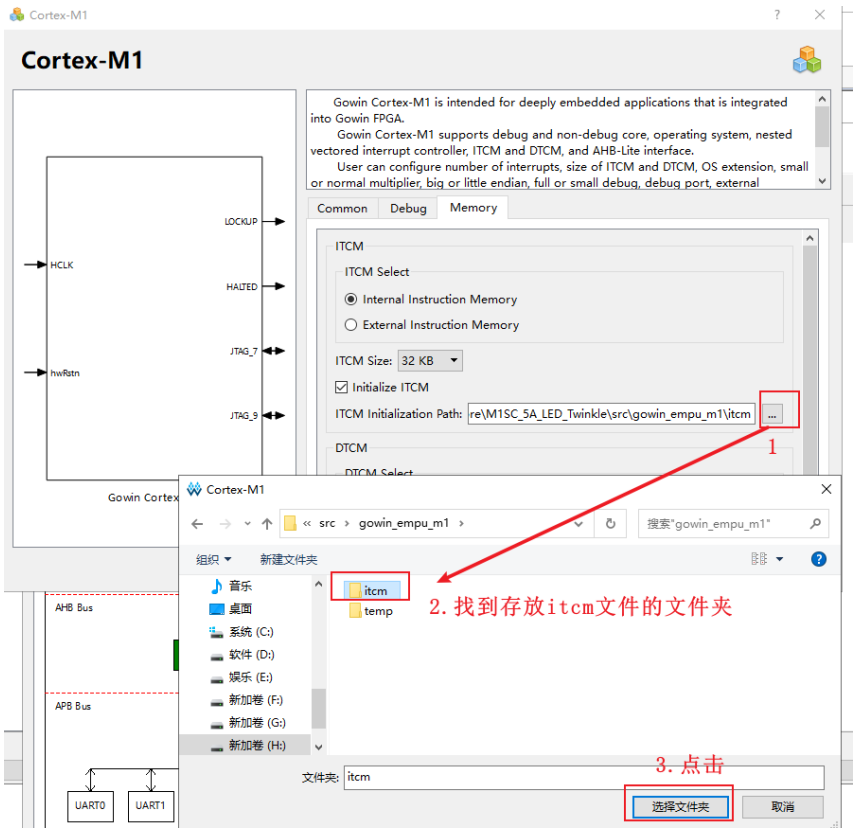

图 1-24 初始化 ITCM 文件

通过上述操作之后，点击 OK 重新生成 IP，然后重新进行分析和综合，生成新的 fs 文件。

然后将板子硬件连接好，硬件连接如下图 1-25 所示，按照 1.3.1.1 节的内容重新下载新生成的 fs 文件。

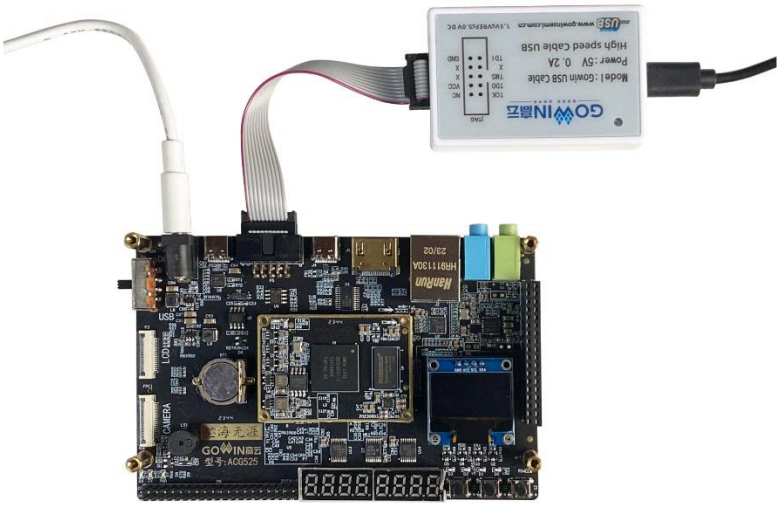

图 1-25 硬件连接图

下载完成之后，可以看到开发板上的 LED0 开始闪烁，这也就说明我们本次实验通过配置 Cortex-M1 的 ITCM 文件下载程序成功。

1.4 思考与总结

我们通过一个简单的 LED 灯闪烁的实验，带领读者学会了使用 Gowin 提供的 Cortex-M1 软核进行软硬件代码设计，并在最后下载程序的时候，教给读者两种程序下载的方法，两种方法使用情况不一样，读者根据实际情况，选择适合的方式进行下载。读者学习完本文档的基本流程之后，自己可以通过 Cortex-M1 软核去设计更多丰富且有意义的工程。