Gowin FPGA 产品

编程配置手册

UG290-2.8, 2025-05-09

版权所有 © 2025 广东高云半导体科技股份有限公司

GOWIN高云、Gowin、小蜜蜂、LittleBee、晨熙、高云均为广东高云半导体科技股份有限公司注册商标, 本手册中提到的其他任何商标，其所有权利属其拥有者所有。未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部，并不得以任何形式传播。

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止反言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2017/04/17	1.00	初始版本。
2017/05/31	1.01	● 更新不同器件支持的配置模式及 mode 值。 ● 更新编程内置 Flash 时 RECONFIG_N 注意事项。
2017/10/13	1.02	更新编程配置管脚复用相关描述。
2018/03/16	1.03	增加 GW1NS 系列芯片编程配置特性描述。
2018/08/08	1.04	● 更新 Flash 为空时配置流程的描述。 ● 更新多重配置的操作步骤描述。 ● 更新 MODE[0]=1 时 JTAG 管脚复用的描述。 ● 更新 B 版本器件编程配置特性描述。 ● 增加编程配置须知及各配置模式时序图。
2019/01/08	1.05	● 添加 SERIAL 配置模式的时序图及时序参数。 ● 删除上电要求相关内容。
2019/08/16	1.06	● 新增上电及配置流程。 ● 修改配置文件大小章节。
2020/05/15	2.0	● 补充 JTAGSEL_N 用作用户 IO 信息说明。 ● 删除 GW1N(R)-2/GW1N(R)-2B/GW1N(R)-6 器件信息。 ● 优化配置模式描述信息。
2020/08/20	2.1	● 补充 JTAG 配置内容。 ● 补充 SSPI 配置内容。 ● 补充 AES 密钥编程内容。
2020/10/30	2.2	增加配置文件加载时长描述。
2021/02/05	2.3	增加 I^2C 配置信息。
2021/09/24	2.4	● 新增“4 配置流程”描述。 ● 新增 GW1N-2 配置或烧录 SRAM/Flash 流程图。 ● 新增内置 Flash 烧录流程相关描述。
2022/01/20	2.4.1	补充 I^2C 配置模式描述。
2022/05/07	2.5	● 删除 GW2AN-9X/18X 的相关信息。 ● 更新 7.5 MSPI 配置模式一节。
2022/05/10	2.5.1	CPU Mode 配置时序示意图。
2022/07/14	2.5.2	● 更新配置文件大小相关信息。 ● 更新 GW1N-2 器件的 SRAM 地址数量和地址长度。 ● 新增 GW1N-2 器件的加载频率的说明。
2022/08/10	2.5.3	● 更新表 7-6 Gowin FPGA ID CODE。 ● 更新表 7-9 JTAG 的 TCK 频率要求。
2022/09/07	2.5.4	● 更新 I^2C 配置模式的注释。 ● 更新表 4-3 管脚功能中关于 READY 管脚和 DONE 管脚的描述。 ● 更新表 7-6 Gowin FPGA ID CODE。
2022/10/28	2.6	● 删除 GW1NS-2/2C、GW1NSR-2/2C、以及 GW1NSE-2C 相关信息。 ● 完善 GW1N-1P5 器件信息。 ● 更新 CLKHOLD_N 管脚的说明。
2022/11/11	2.6.1	● 更新“7.4.5 SSPI 配置模式连接示意图”一节。 ● 更新“编程外部 Flash 或内嵌 SPI-Flash”一节。

日期	版本	说明
		● 更新“读取 Status Register（0x41”一节。 ● 新增关于 JTAG 管脚和 JTAGSEL_N 管脚复用的说明。 ● 更新图 3-7 Program AES Key Flow。 ● 更新 T 工艺 FPGA 擦除流程。
2022/11/24	2.6.2	● 更新表 7-20 CPU 配置模式管脚。 ● 新增表 7-21 CPU 配置模式时序参数。 ● 更新图 7-60 CPU 模式配置流程示意图。 ● 新增图 7-61 CPU 配置模式时序图。 ● 修改关于 I^2C 模式的注释。
2022/12/02	2.6.3	新增关于背景升级的注释。
2023/01/12	2.6.4	● 新增注释到表 7-24 I2C 配置模式管脚定义。 ● 更新表 3-3 MSPI 模式数据流文件加载时长。 ● 更新表 3-4 Autoboot 模式数据流文件加载时长。 ● 更新表 10-1 SPI Flash 操作指令。
2023/01/20	2.6.5	● 新增注释到表 10-1 SPI Flash 操作指令。 ● 更新图 7-60 CPU 模式配置流程示意图。 ● 更新图 7-61 CPU 配置模式时序图。
2023/02/02	2.6.6	● 更新表 7-21 CPU 配置模式时序参数。 ● 修改表 10-1 SPI Flash 操作指令的注释。
2023/03/13	2.6.7	新增关于用 I^2C 编程内部 Flash 的注释。
2023/06/30	2.7	● 调整文档结构。 ● 更新 RECONFIG_N 管脚的注意事项。 ● 更新表 7-1 小蜜蜂(LittleBee)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数。 ● 更新表 7-2 晨熙(Arora)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数。 ● 更新 IDE 软件的截图（基于 Gowin_V1.9.9Beta）。 ● 更新图 6-1 高云半导体 FPGA 配置流程图。 ● 更新图 7-15 正常烧录流程图。 ● 新增关于 MCLK 线下拉电阻的信息。
2023/07/24	2.7.1	● 修改表 10-1 SPI Flash 操作指令的注释。 ● 修改 JTAGSEL_N 管脚的说明。 ● 新增关于 MSPI 模式时钟频率误差的注释。
2023/11/16	2.7.2	● 修正配置文件加载时长的计算公式。 ● 更新 Status Register 的相关信息。 ● 更新 T 工艺 FPGA 擦除流程。 ● 修改表 3-1 高云半导体 FPGA 产品配置文件大小（最大情况）的注释。 ● 修改表 3-2 配置文件最大加载频率。 ● 完善表 7-18 MSPI 配置模式管脚定义的注释。
2023/12/28	2.7.3	● 更新图 7-13 配置 SRAM 流程、图 7-19 编程内部 Flash 流程图、图 7-20 X-page 编程流程图。 ● 更新 MCLK 频率的相关信息。 ● 更新关于背景升级的描述。
2024/05/22	2.7.4	● 更新“表 7-6 Gowin FPGA ID CODE”。 ● 修正“图 7-35 菊花链连接示意图”。 ● 新增“附录 A 管脚状态信息”。

日期	版本	说明
2024/07/26	2.7.5	● 修正表 7-10 标题。 ● 完善 JTAG 配置模式的说明。 ● 完善 I^2C 配置模式的说明。 ● 更新“表 7-1 小蜜蜂(LittleBee)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数”和“表 7-2 晨熙(Arora)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数”：修正 $T_{portready}$ 的值。
2024/12/12	2.7.6	● 优化 JTAGSEL_N 管脚的描述。 ● 修正“图 7-58 双启动配置模式流程图”。 ● 完善“6.1 上电时序”中的描述，更新“图 6-2 POR 上电时序图”。
2025/03/14	2.7.7	● 更新“T 工艺 FPGA 擦除流程”。 ● 更新“表 5-1 配置模式”的注释 4。 ● 新增“附录 B 比特流文件格式”。
2025/05/09	2.8	新增“3.5 背景升级（Background Programming）”。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

本手册主要介绍高云半导体小蜜蜂(LittleBee)家族及晨熙(Arora)家族 FPGA 产品编程配置方面的通用特性及功能，旨在帮助用户更好地使用 Gowin FPGA 产品。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com.cn 可以下载、查看以下相关文档：

- DS100, GW1N 系列 FPGA 产品数据手册
- DS102, GW2A 系列 FPGA 产品数据手册
- DS117, GW1NR 系列 FPGA 产品数据手册
- DS226, GW2AR 系列 FPGA 产品数据手册
- DS961, GW2ANR 系列 FPGA 产品数据手册
- DS821, GW1NS 系列 FPGA 产品数据手册
- DS841, GW1NZ 系列 FPGA 产品数据手册
- DS861, GW1NSR 系列 FPGA 产品数据手册
- DS871, GW1NSE 系列 FPGA 产品数据手册
- DS881, GW1NSER 系列 FPGA 产品数据手册
- DS891, GW1NRF 系列 FPGA 产品数据手册
- DS961, GW2ANR 系列 FPGA 产品数据手册
- DS976, GW2AN-55 器件数据手册
- TN711, GOWIN FPGA 产品状态寄存器说明

1.3 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	描述
Bitstream	Bitstream Data	配置 FPGA SRAM 的数据
Bscan	Boundary Scan	边界扫描测试技术
Configuration	Configuration	配置 FPGA SRAM 区域的过程
Configuration Data	Configuration Data	配置 FPGA SRAM 的数据
Configuration Mode	Configuration Mode	配置模式，决定 Configuration Data 源
CPU	Central Processing Unit	中央处理器
CRC	Cyclic Redundancy Check	循环冗余校验
Edit Mode	Edit Mode	FPGA 处于 Configuration 或 Programming 所在模式
EFlash/EmbFlash	Embedded Flash	内置 Flash
FPGA	Field Programmable Gate Array	现场可编程门阵列
FS file	Fuses file	包含配置数据的 ASCII 文件
GPIO	General Purpose Input Output	通用输入、输出接口
I2C（I^2C、IIC）	Inter-Integrated Circuits	两线式串行总线
ID	Identification	身份标识号
IEEE	Institute of Electrical and Electronics Engineers	电气和电子工程师协会
Internal Flash	Internal Flash	内置 Flash
JTAG	Joint Test Action Group	联合测试行动组
LSB	Least Significant Bit	最低有效位（优先）
LUT	Look-up Table	查找表
MSB	Most Significant Bit	最高有效位（优先）
MSPI	Master Serial Peripheral Interface	主串行外设接口
Programming	Programming	将 Configuration Data 烧录到 Embedded Flash 或 External Flash 存储器的过程
SCL	Serial Clock	I^2C 上的时钟线
SDA	Serial Data	I^2C 上的数据线
Security Bit	Security Bit	安全位（使 SRAM 回读永为高电平）
SPI	Serial Peripheral Interface	串行外设接口
SRAM	Static Random Access Memory	静态随机存储器
SSPI	Slave Serial Peripheral Interface	从串行外设接口
TAP	Test Access Port	测试访问口

术语、缩略语	全称	描述
User Mode	User Mode	FPGA 在配置完成过后，执行相应逻辑功能的模式。

1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 名词解释

本章主要介绍高云半导体 FPGA 产品编程配置过程中常用的一些名词及含义，帮助用户熟悉高云半导体 FPGA 产品编程配置相关的概念。

表 2-1 名词解释

名词	释义
编程（Program）	将高云半导体云源软件生成的比特流数据写入到 FPGA 内置 Flash 或者与 FPGA 相连的外部 SPI Flash 的过程。
配置（Configure）	将 Gowin 云源软件生成的比特流数据通过外部接口或内置 Flash 加载到 FPGA 的 SRAM 存储区的过程。
GowinCONFIG	除通用的 JTAG 配置模式外，高云半导体 FPGA 产品支持的其他配置方式，包括自启动配置，双启动配置，MSPI 配置，SSPI 配置，SERIAL 配置，CPU 配置。每款器件支持的 GowinCONFIG 配置模式多少取决于器件型号和封装类型。
MODE[2:0]	与 GowinCONFIG 相关的三个 MODE 管脚的值的表示方法。
自启动配置（AUTO BOOT）	FPGA 从内置 Flash 读取比特流数据进行配置的过程。只有非易失器件支持此模式。
双启动配置（DUAL BOOT）	2 个比特流文件分别存放在内置 Flash 和外部 Flash，当外置的 Flash 配置失败时切换到内置 Flash 进行配置。只有非易失器件支持此模式。
MSPI 配置	FPGA 作为主器件（master），通过 SPI 接口主动从外部 Flash 读取比特流数据进行配置的过程。
SSPI 配置	FPGA 作为从器件（slave），外部主机（master）通过 SPI 接口写入比特流数据进行配置的过程。
SERIAL 配置	FPGA 作为从器件（slave），外部主机（master）通过串行接口写入比特流数据进行配置的过程。
CPU 配置	FPGA 作为从器件（slave），外部主机（master）通过并行接口（数据位宽 8-bit）写入比特流数据进行配置的过程。
I2C 配置	FPGA 作为从器件（slave），外部主机（master）通过 I2C

名词	释义
	接口写入比特流数据进行配置的过程。
多重配置（MULTI BOOT）	MSPI 配置模式的衍生概念，是指 FPGA 从外部 Flash 的不同地址读取比特流数据进行配置的过程。用户在前一个比特流数据中写入后一次配置的比特流数据的加载地址，在器件不掉电的情况下通过触发 RECONFIG_N 切换数据流文件完成配置。支持 MSPI 配置模式的 FPGA 产品均支持此模式。
远程升级	用户的一种应用场景，即在 FPGA 启动工作后，若有升级需求，先通过远程操作将比特流数据写入到外部 Flash 中，通过触发 RECONFIG_N 或重新上电使 FPGA 读取外部 Flash 完成配置的过程。
菊花链	FPGA 器件以串行的方式依次连接起来的一种方式，可以从链首按照连接顺序依次对器件进行配置，只有相邻的器件之间才能传输数据。
用户模式（User Mode）	FPGA 完成一次配置操作后，将控制权移交给用户的行为。配置管脚复用为普通 I/O 的设置仅在用户模式下生效。
编辑模式（Edit Mode）	可以对 FPGA 进行编程或配置操作的模式。 编辑模式下所有的配置管脚无法作为普通 I/O 使用，所有的普通管脚输出为高阻态（背景升级除外），详细信息请参考附录 A 管脚状态信息。
ID CODE	高云半导体 FPGA 器件的身份标识，每一个系列的器件具有独立的编号。
USER CODE	用户为自己所使用的 FPGA 器件进行的身份标识，可以通过 Gowin 编程软件写入到器件中，最高可支持 32-bit。
安全位（Security Bit）	高云半导体为保护 FPGA 产品配置数据运行时的安全性所做的特殊设计。用户将设置了安全位的比特流数据写入器件 SRAM 后，任何人都将无法进行数据回读操作。云源软件默认为所有 FPGA 产品的比特流数据设置了安全位。
加密（Encryption）	晨熙(Arora)家族 FPGA 产品支持的特性，加密的比特流数据写入 FPGA 后，器件自行与事先存储的密钥匹配，匹配成功后进行解密并唤醒器件；匹配失败后器件无法工作。

3 比特流文件配置

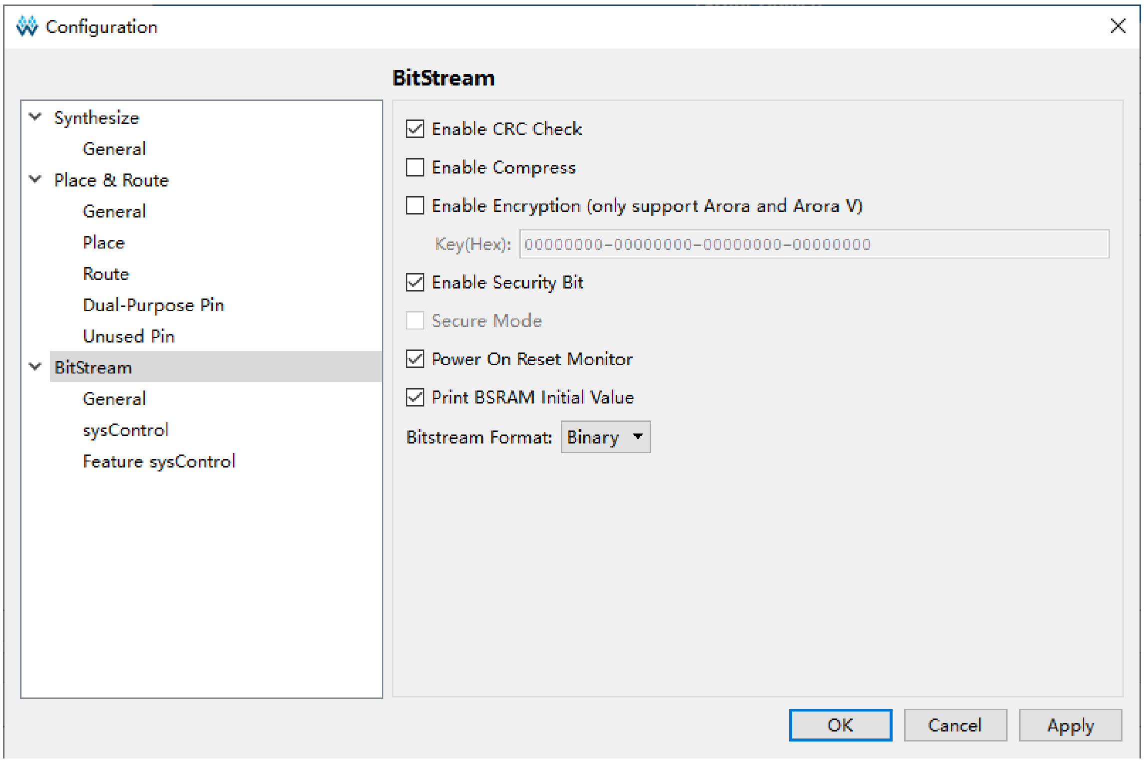
高云半导体 FPGA 产品编程配置相关的特性需要结合云源软件进行选项设置。配置主要包括配置管脚复用相关的选项和比特流文件配置相关选项。本章主要描述比特流文件配置相关信息，配置管脚复用相关信息请参考4.1.2 配置管脚复用。

高云半导体为保障配置数据的安全准确传输，在 FPGA 产品的比特流文件中默认加入了 CRC 校验算法并设置了安全位。数据配置过程中实时校验输入数据是否出错，错误的数据无法唤醒器件，DONE 信号拉低。设置了安全位的比特流数据完成配置后，任何用户无法进行回读操作。

3.1 配置选项设置

比特流数据生成相关的设置界面请参考图 3-1，配置选项包括 CRC 校验使能、比特流数据压缩、加密密钥设置、安全位设置、MSPI 配置频率选择、多重配置模式下 SPI Flash 启动地址设置、USER CODE 设置等。SPI Flash 启动地址低 12 位无效，用户可设置的是 ADDR[23:12]的地址空间。

图 3-1 配置选项



注！

高云半导体云源软件勾选加密密钥设置选项后强制勾选安全位设置选项，用户使用这样的比特流数据进行配置，既可以保证数据传输过程的安全，又能够阻止任何回读操作，最大限度地保障了用户数据的安全性。

3.2 配置数据加密（仅晨熙(Arora)家族支持）

高云半导体晨熙(Arora)家族 FPGA 产品支持比特流数据加密，采用 128 bits 的 AES 加密算法。加密的比特流数据的配置流程如下：

1. 在高云半导体云源软件输入加密密钥生成比特流文件。
2. 在 Gowin 编程软件输入解密密钥存入 FPGA。
3. 将加密的比特流数据加载到器件之后，器件会自行读取解密密钥进行数据解析。

   数据解析成功后，器件完成配置并正常工作；数据解析失败后，器件无法工作，READY 和 DONE 信号拉低。

3.2.1 定义

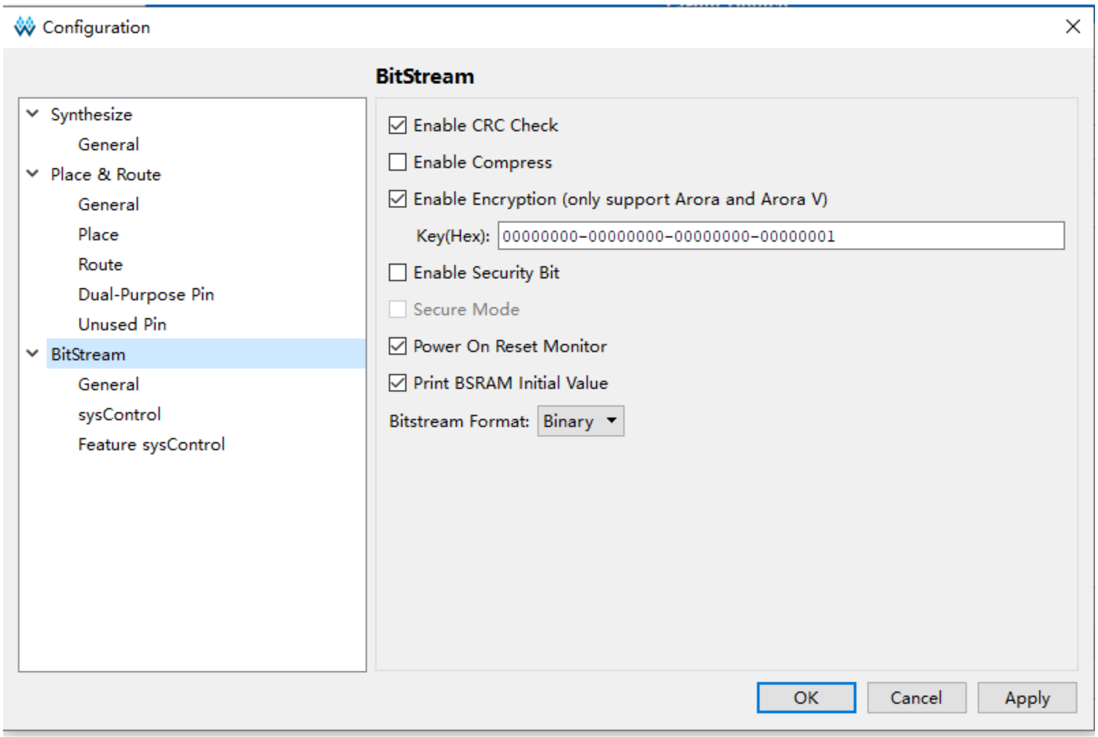
- AES 密钥：也称 AES 私钥，AES 加密算法中用到的私钥部分，由算法外指定，本文简称 key。
- AES 密钥长度：128 位。
- Key：AES 密钥的简称，GW2A(R)系列 FPGA 产品中提供一个 128 位长度的空间用于存储 Key。
- Lock：为保证 AES 密钥的安全，该指令用于限制 key 的读权限，本文将该过程简称 lock，当处于锁定状态后，设置后回读数据所有数据都是 1。

3.2.2 输入加密密钥

在云源软件中输入加密密钥的方法如下：

1. 打开云源软件中相应的工程。
2. 在菜单栏中选择“Project>Configuration”。
3. 单击“BitStream”页签，勾选“Enable Encryption”并输入密钥值，如图 3-2 所示。

图 3-2 加密密钥设置方法



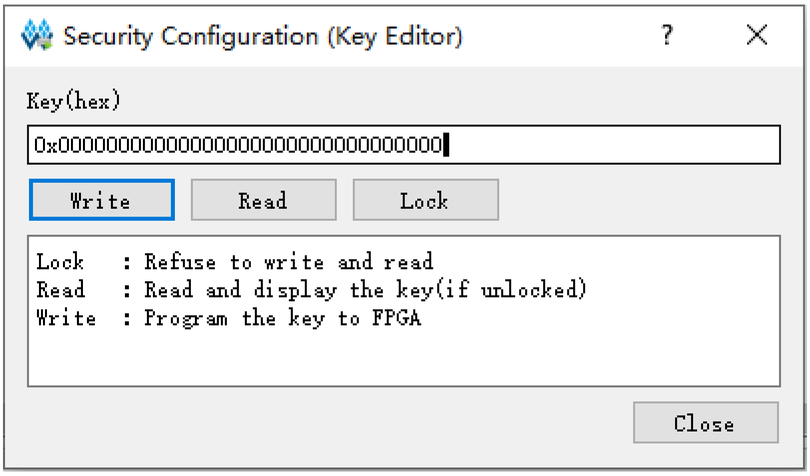
加密密钥设置成功后，还需要将解密的密钥写入到 FPGA 的密钥存储区，器件才能解析加密的比特流数据完成配置。

3.2.3 输入解密密钥

解密密钥的写入方法如下：

1. 打开 Gowin 编程软件。
2. 扫描 FPGA 器件。
3. 右键单击器件选择 Security Key Setting。
4. 在弹出的界面上输入之前加密的密钥值并单击“write”将其写入到 FPGA，如图 3-3 所示。

图 3-3 解密密钥设置方法



解密密钥写入成功后可以选择界面上的读取指令回读写入的密钥进行验证。

密钥写入成功后，用户也可选择 lock 命令将密钥“锁死”在 FPGA 内部，之后任何对密钥的读取和写入操作都将无效：密钥值无法进行修改，读取的密钥所有位全部为“1”。

设置解密密钥后，加密的比特流数据只有与解密密钥匹配成功后才能唤醒。非加密比特流数据的配置不受密钥影响。

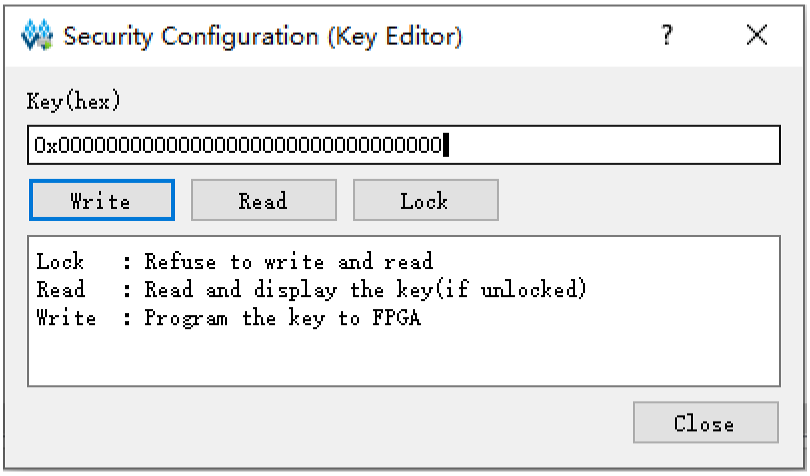
注！

高云半导体 FPGA 的密钥初始值所有位全部为 0，当把密钥值的某一位修改为 1 后便无法改回 0。例如，用户的某次操作中写入的密钥值为 00000000-00000000-00000000-00000001，之后修改此器件的密钥时最低位也必须是 1。

3.2.4 AES 密钥编程操作

Gowin Programmer 工具中提供了 AES 密钥编程工具，在 Gowin Programmer 中单击菜单“Edit”中“Security Key Setting”选项即可开启该工具，如图 3-4 所示。

图 3-4 AES 编程对话框



该程序包含三个功能，分别是：

- Write：编程 Key。
- Read：读取 Key。
- Lock：锁定 Key 的读写权限。

编程 Key（Write）

1. 将自定义的 Key(AES 密钥)填入“Key（hex）”中。
2. 单击“write” 按钮。
3. 工具运行结束，返回验证结果。

读取 Key（Read）

单击“read”按钮可对写入的 AES 密钥进行再次验证，读取出来的 AES 密钥会显示在“单行文本对话框”中。

锁定 Key（Lock）

单击“lock”按钮，锁定 Key 数据的读写，AES 密钥将不能再被读取和写入。

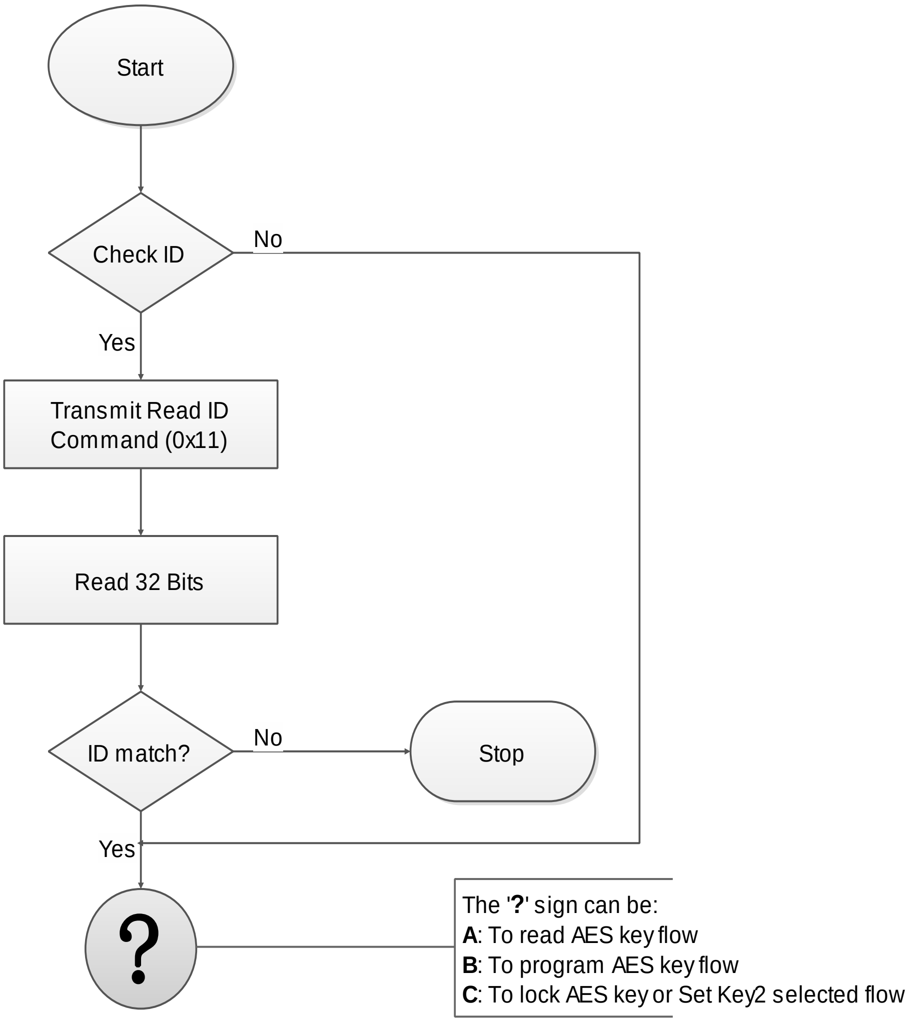
3.2.5 AES 密钥编程流程

图 3-5 ~ 图 3-8 给出了如何编程或锁定 AES 密钥的流程，图示流程均基于 JTAG 协议。

检查 ID CODE

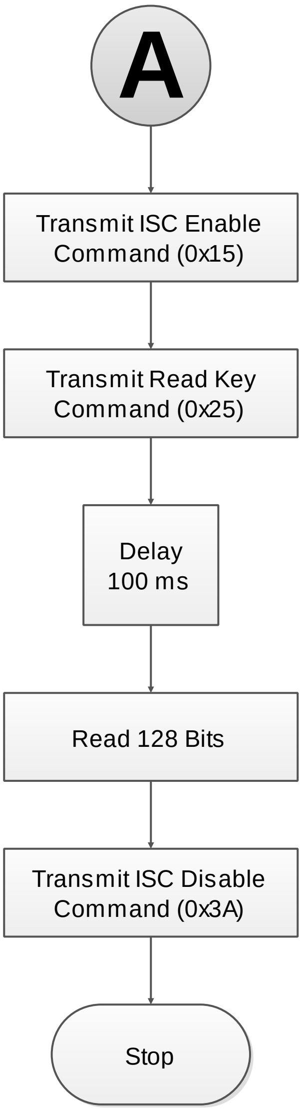
对器件 ID 进行检查，一方面可确定 JTAG 协议是否工作正常，另一方面确定烧录对象是否正确，避免误操作。

图 3-5 Prepare



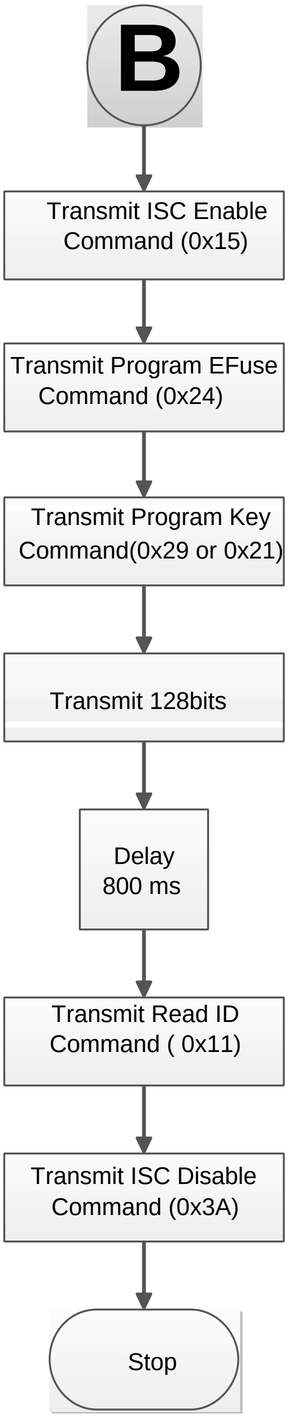
读取 AES Key

图 3-6 Read AES Key Flow



烧录 AES Key

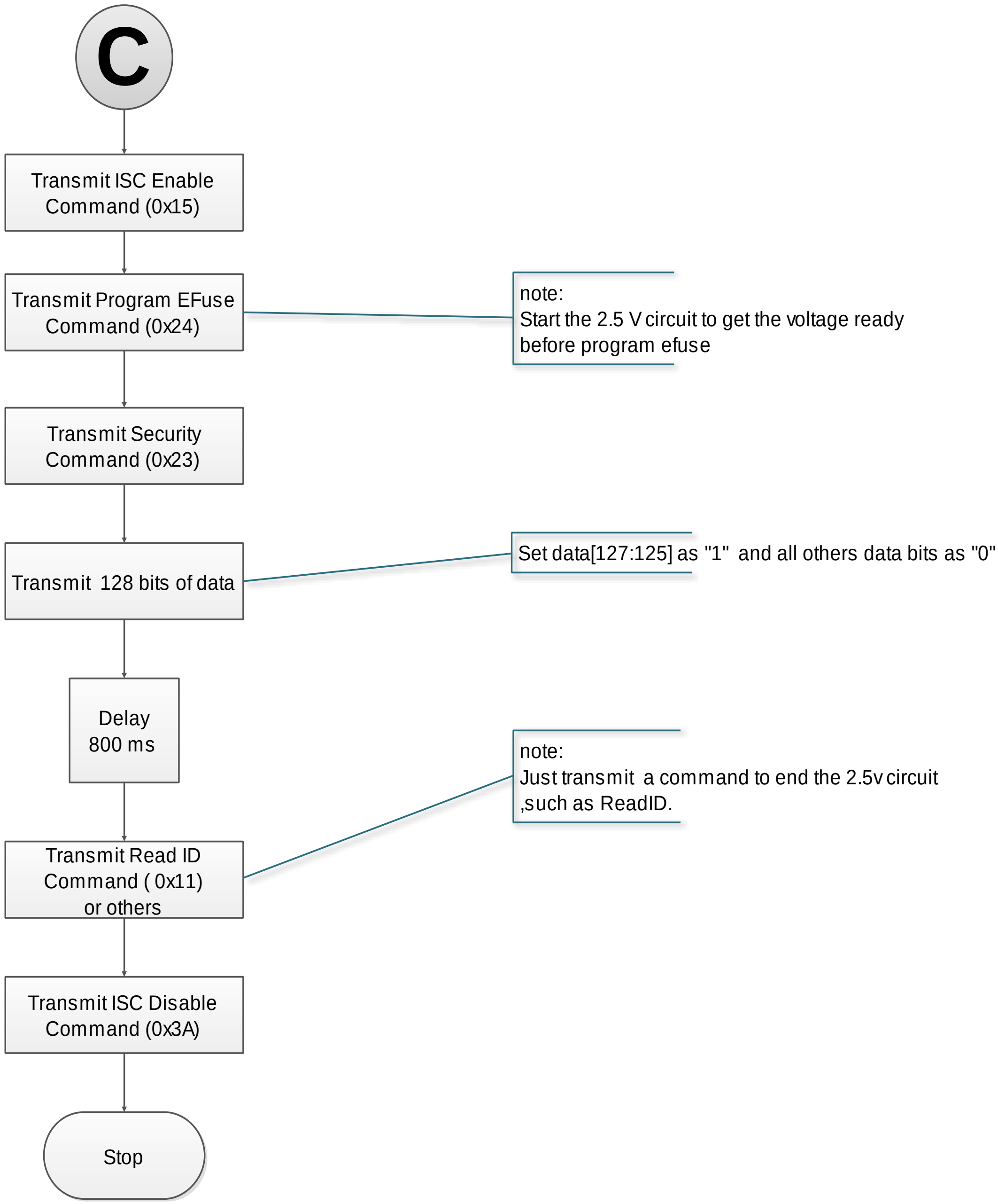
图 3-7 Program AES Key Flow



锁定 AES Key

锁定 AES Key 的作用是防止 Key 泄露。锁定 AES Key 之后将不能读取和配置 AES Key.

图 3-8 Lock AES Key Flow



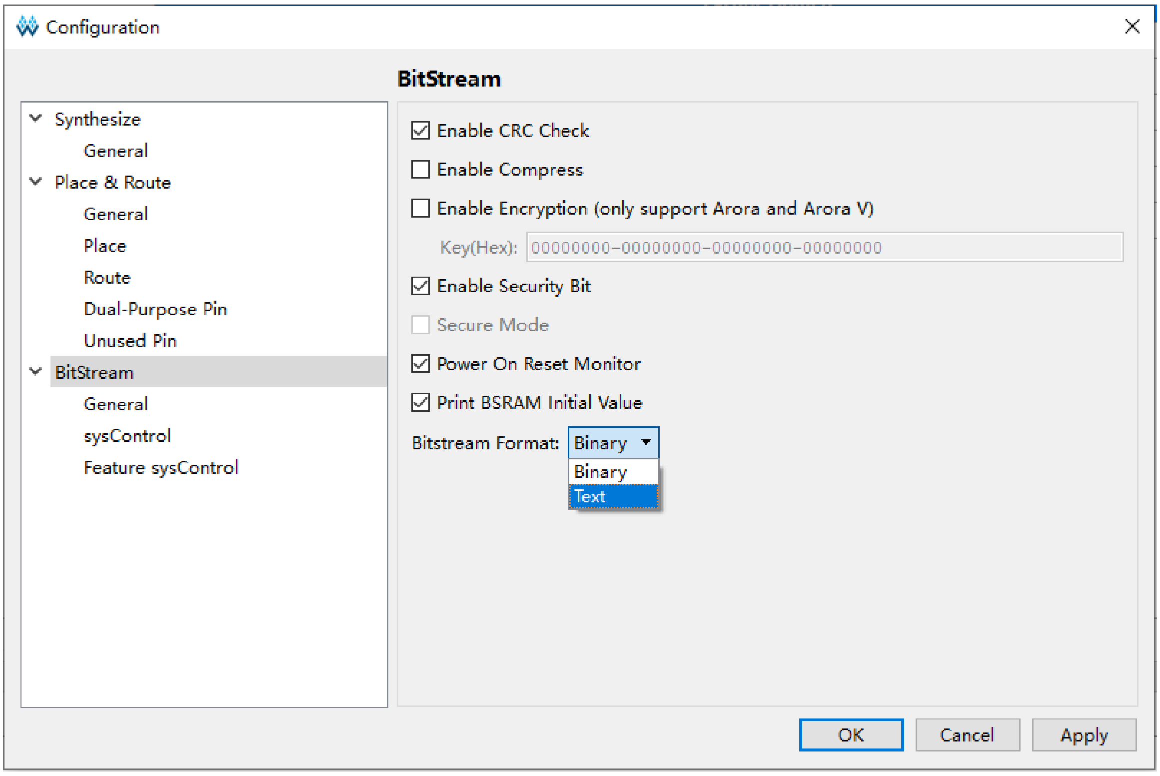
3.3 配置文件大小

高云半导体 FPGA 产品配置数据流文件的存储格式，包括携带注释信息的文本格式(ASCII)的文件和不携带注释信息的二进制格式的文件。文本格式的文件后缀名为.fs，其中以“//”开头的行属于注释信息，其他部分是数据流数据。二进制格式的文件的后缀名为.bin，其不包含注释信息，此文件格式通常用于嵌入式编程。用户可以在高云半导体云源软件中设置存储格式：

1. 打开高云半导体云源软件。
2. 在 Process 选项卡上右键单击 Place&Route 选择 Configuration 中的 bitstream。
3. 在 Bitstream Format 选项中选择 Text 或 Binary 格式即可，如图 3-9 所

示。

图 3-9 比特流格式生成



高云半导体支持比特流数据的压缩，压缩比例与用户的设计相关，本文档只提供未压缩的配置文件大小，如表 3-1 所示。

表 3-1 高云半导体 FPGA 产品配置文件大小（最大情况）

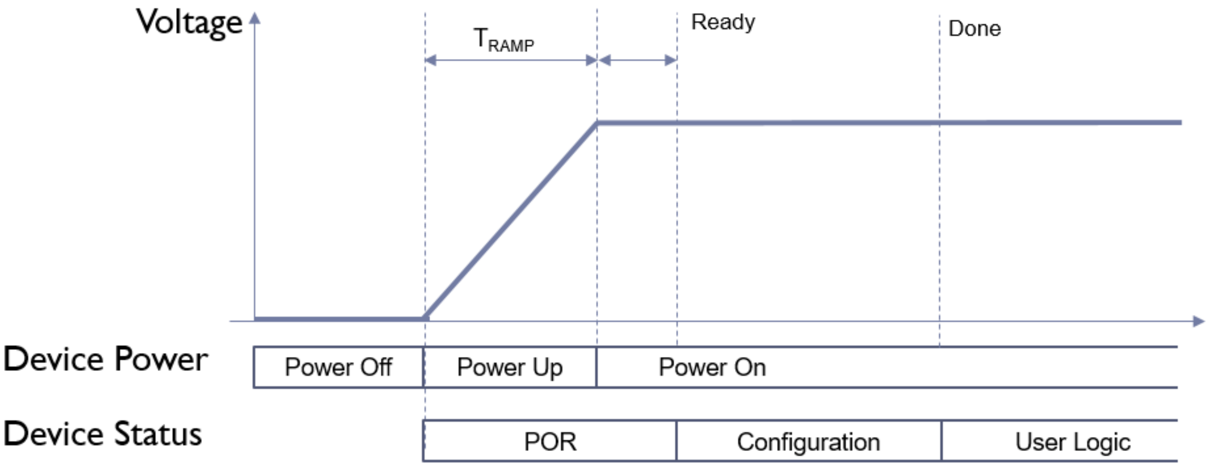
器件名称	逻辑单元数量	最大配置文件
GW1N-1(S)、GW1NR-1、GW1NZ-1	1,152	84 KBytes
GW1N-1P5	1,584	113 KBytes
GW1N-2、GW1NR-2	2,304	113 KBytes
GW1N-4、GW1NR-4、GW1NS-4(C)、GW1NSR-4(C)、GW1NSER-4C、GW1NRF-4B	4,608	217 KBytes
GW1N-9、GW1NR-9	8,640	435 KBytes
GW2A-18、GW2AR-18、GW2ANR-18	20,736	887 KBytes
GW2A-55、GW2AN-55	54,720	2269 KBytes

注！

表中的数据是二进制格式的文件大小，并且配置文件未进行压缩。

3.4 配置文件加载时长

FPGA 可以作为 Master 从 Flash 读取数据流文件并配置 SRAM，包括 Autoboot 和 MSPI 两种情况，分别对应数据源来自内置和外置 Flash。当 FPGA 上电并在 Ready 后开始尝试主动读取数据流文件，当完成加载后，FPGA 进到 User Logic 状态，如下图所示。



高云 FPGA 器件小蜜蜂和晨熙系列均支持 MSPI 模式,即从外部 SPI Flash 自行读取数据流并配置 FPGA，读取配置文件的频率默认是 2.5MHz，每一个 SPI 时钟即读取一个比特，根据文件大小可计算加载所需时长。MSPI 读取 SPI Flash 的时钟频率不大于 66.6MHz。需要注意的是，当使用 Fast Read SPI（0x0B）时，要同时接地 FastRead_N 管脚。

高云小蜜蜂系列不仅支持 MSPI 模式，还支持 Autoboot 模式，加载频率默认是 2.5MHz，Autoboot 每时钟加载一个字节（8 比特）。

注！

对于 GW1N-2 器件而言，若其 MODE[2]的值固定为 1，则其加载频率只能是 2.5MHz。

根据配置文件大小、加载频率和每时钟加载位宽的不同，其加载时间也不一样。

由于内置 Flash 工艺的不同，不同器件的 Autoboot 模式最大加载频率也不同，具体速度请参见表 3-2。

表 3-2 配置文件最大加载频率

器件	Autoboot 最大加载频率	MSPI 最大加载频率
GW2A-55/55C	无 Autoboot 模式	<66.6MHz
GW2A-18/18C		
GW2AR-18/18C		
GW2ANR-18C		
GW1N-1	26MHz	
GW1N-1S		

器件	Autoboot 最大加载频率	MSPI 最大加载频率
GW1NZ-1 GW1N-2/1P5 GW1N-2B/1P5B GW1NSER-4C GW1NS-4 GW1NSR-4 GW1NS-4C GW1NSR-4C GW1N-4B GW1NR-4B GW1NRF-4B GW1N-4 GW1NR-4 GW1N-9 GW1N-9C GW1NR-9 GW1NR-9C	40MHz	

MSPI 模式数据流文件加载时长如表 3-3 所示。

表 3-3 MSPI 模式数据流文件加载时长

逻辑单元数量	最大配置文件	加载频率=2.5MHz 所需时间（ms）	加载频率=25MHz 所需时间（ms）	加载频率=41.6MHz 所需时间（ms）	加载频率=62.5MHz 所需时间（ms）
1,152	84 KBytes	275	28	17	11
1584	116 KBytes	381	40	25	17
2304	116 KBytes	381	40	25	17
4,608	217 KBytes	711	71	42	28
8,640	435 KBytes	1425	142	85	57
20,736	887 KBytes	2906	290	174	116
54,720	2269 KBytes	7435	743	446	297

AUTO BOOT 模式数据流文件加载时长如表 3-4 所示。

表 3-4 Autoboot 模式数据流文件加载时长

逻辑单元数量	最大配置文件	加载频率=2.5MHz 所需时间（ms）（默认频率）	加载频率=25MHz 所需时间（ms）	加载频率=31.25MHz 频率所需时间（ms）
1,152	84 KBytes	34	4	3
1584	116 KBytes	48	7	6
2304	116 KBytes	48	7	6
4,608	217 KBytes	88	9	7
8,640	435 KBytes	178	17	14

以上列出的是加载时间的参考，设备从上电到配置完成，除了配置所占时间，还有设备上电的时间 Tramp，和设备初始化的时间，具体上电时间与电源器件有关，需要自行测量。所以 FPGA 从上电到加载完成大致时间可按如下公式计算：

Autoboot 模式：

$T_{加载时长}$ = POR 时长 + 数据流比特数/8/加载频率

MSPI 模式：

$T_{加载时长}$ = POR 时长 + 数据流比特数/加载频率

3.5 背景升级（Background Programming）

背景升级：即器件支持在不影响现有工作状态的情况下通过 JTAG/I2C/SSPI 接口编程内嵌 Flash 或外部 Flash 的操作，编程过程中器件可以按照原有的配置正常工作，编程完成后，低电平触发 RECONFIG_N 即可完成在线升级。此特性适合应用于在线时间长但又需要不定期升级的场所。

- C 版本的 GW1N-2 和 GW1N-1P5 器件的内部 Flash 不支持通过原生 I2C 进行编程。但是可以使用 goConfig I2C IP 进行编程，从而实现背景升级。背景升级时，需先使用 goConfig I2C IP 擦除内置 Flash，详细信息请参考 IPUG795, Gowin goConfig I2C IP 用户指南，TN715,Gowin goConfigIP(I2C)示例代码使用说明。
- C 版本的 GW1N-2 和 GW1N-1P5 器件的内部 Flash 可以使用 goConfig SPI IP 进行编程，从而实现背景升级。详细信息请参考 IPUG1188, Gowin goConfig SPI IP 用户指南。
- C 版本的 GW1N-2 和 GW1N-1P5 器件的内部 Flash 可以使用 goConfig JTAG IP 进行编程，从而实现背景升级。适用于 MCU/CPU 模拟 JTAG 时序不满足要求的场景，详细信息请参考 IPUG1035, Gowin goConfig JTAG IP 用户指南。
- 小蜜蜂家族 9K 器件及晨曦家族所有器件的外部 Flash 可以使用 goConfig UART IP 进行编程，从而实现背景升级。详细信息请参考 IPUG1197, Gowin goConfig UART IP 用户指南。

3.5.1 背景升级支持情况

小蜜蜂家族和晨熙家族对背景升级的支持情况如表 3-5 所示。

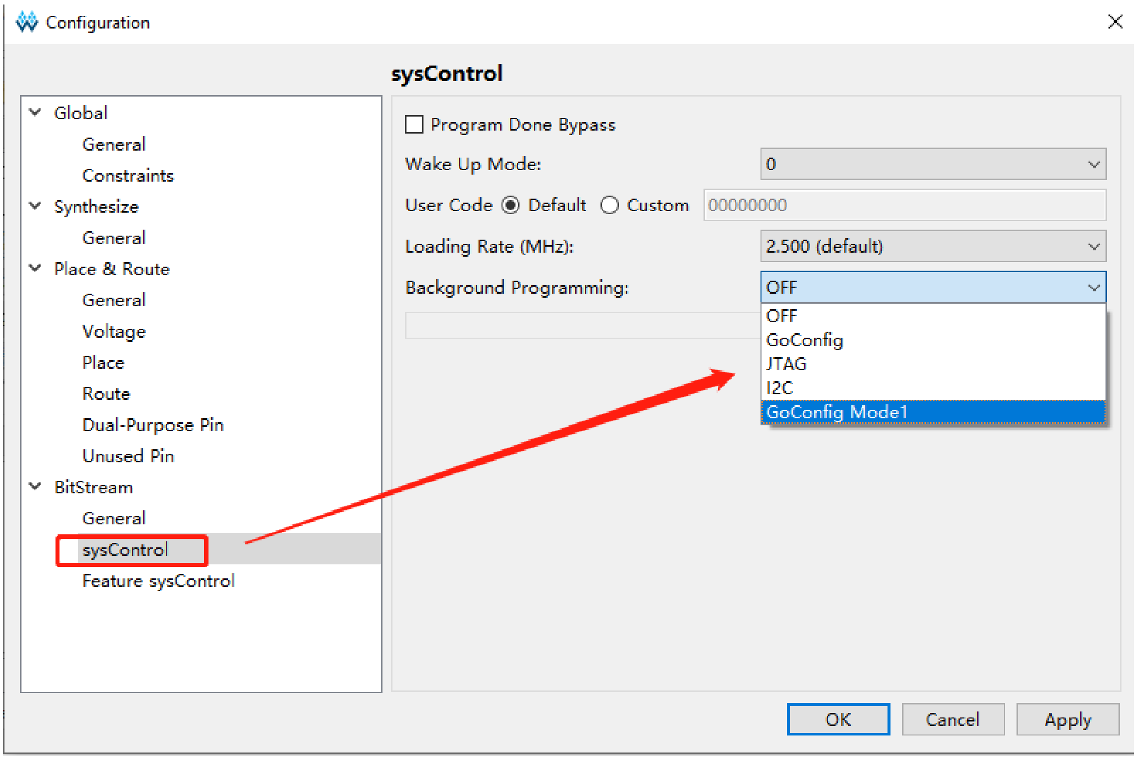
表 3-5 支持背景烧录的器件列表

器件	背景烧录模式	描述
GW1N-1/GW1N-1S	不支持	
GW1NZ-1/GW1NZ-1C	JTAG	需要在云源软件配置界面配置，参考下图
GW1N-1P5/ GW1N-1P5B	JTAG	需要在云源软件配置界面配置，参考下图
GW1N-1P5C	I2C	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig I2C 实现
	GoConfig	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig JTAG/GoConfig SPI 实现
	GoCofig Mode1	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig I2C/GoConfig SPI 实现
	JTAG	需要在云源软件配置界面配置，参考下图
GW1N-2/GW1NR-2/ GW1N-2B/GW1NR-2B//GW1NZ-2B	JTAG	需要在云源软件配置界面配置，

器件	背景烧录模式	描述
		参考下图
GW1N-2C/GW1NR-2C/ GW1NZ-2C	I2C	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig I2C 实现
	GoConfig	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig JTAG/GoConfig SPI 实现
	GoCofig Mode1	需要在云源软件配置界面配置，参考下图，需要配合 IP GoConfig JTAG/GoConfig SPI 实现
	JTAG	需要在云源软件配置界面配置，参考下图
GW1N-4/ GW1NR-4	不支持	
GW1N-4B/GW1N-4D/ GW1NR-4B/ GW1NR-4D/GW1NRF-4B/ GW1NS-4/ GW1NSR-4	JTAG	需要在云源软件配置界面配置，参考下图
GW1NS-4C/GW1NSER-4C/ GW1NSR-4C	不支持	
GW1N-9/GW1N-9C/GW1NR-9/GW1NR-9C	JTAG	通过 JTAG 端口实现背景升级需要在云源软件配置界面配置。也可通过 UART 转 SPI 端口实现背景升级，Gowin 提供 IP GoConfig UART 实现此功能
GW2A-18/GW2AR-18/GW2A-18C/GW2AR-18C/GW2ANR-18C	支持	无需软件配置，可通过 JTAG 或 SSPI 端口实现背景升级。也可通过 UART 转 SPI 端口实现背景升级，Gowin 提供 IP GoConfig UART 实现此功能
GW2A-55/GW2A-55C/ GW2AN-55C	支持	无需软件配置，可通过 JTAG 或 SSPI 端口实现背景升级。也可通过 UART 转 SPI 接口实现背景升级，Gowin 提供 IP

器件	背景烧录模式	描述
		GoConfig UART 实现此功能

图 3-10 背景烧录设置



3.5.2 背景升级调试注意事项

1. 空片模式，即 SRAM 为空的状态下不能进行背景升级。
2. 晨熙家族和小蜜蜂家族器件，对外部 Flash 进行背景升级时不能复用 MSPI。
3. GW1NZ-1、2K、4K、9K 器件背景升级频率要求，见表 7-9。
4. 勾选“Use JTAG as regular IO”选项时，使用 GoConfig Mode1，不勾选“Use JTAG as regular IO”时，使用 GoConfig，GoConfig Mode1 模式要求 JTAG 复用为 GPIO 时，JTAG IO 不能处于激活状态
5. goConfigIP 参考设计请去官网下载：路径为：IP ->gowin goConfig xxxIP。
6. goConfigIP MCU 端代码请联系 FAE 获取。

Programmer 中集成了背景升级模式 debug 选项，方便验证 EDA 生成 goConfigIP 工程；使用说明请参考：SUG502，Gowin_Programmer 用户指南。

4 配置管脚介绍

高云半导体 FPGA 产品配置模式多样，包括通用型的 JTAG 配置、主动配置、被动配置、串行配置和并行配置等，可以满足用户不同外设环境下的各种需求。编程配置相关的管脚既能够完成配置功能，又可以设置为普通的 I/O，用户可根据实际使用情况进行选择。用户也可以根据配置管脚的功能对其灵活控制，满足一些特殊需求。

4.1 配置管脚列表及复用选项

4.1.1 配置管脚列表

高云半导体 FPGA 产品所有与配置相关的管脚如表 4-1 所示，表中也标注了每种配置模式使用到的管脚及芯片封装过程中的管脚共用情况。

表 4-1 配置管脚列表

管脚名称	I/O 类型	JTAG	GowinCONFIG						
			AUTO BOOT	I^2C	SSPI	MSPI	DUAL BOOT	SERIAL	CPU
RECONFIG_N	I	√	√	√	√	√	√	√	√
JTAGSEL_N	I	√							
TDO	O	√							
TMS	I	√							
TCK	I	√							
TDI	I	√							
READY	I/O	√	√	√	√	√	√	√	√
DONE	I/O	√	√	√	√	√	√	√	√
MODE[2:0]	I		√	√	√	√	√	√	√
SCLK	I				√			√	√
CLKHOLD_N/DIN	I				√			√	√

管脚名称	I/O 类型	JTAG	GowinCONFIG						
			AUTO BOOT	I^2C	SSPI	MSPI	DUAL BOOT	SERIAL	CPU
WE_N/DOUT	O							√	√
MI/D7	I/O					√			√
MO/D6	I/O					√			√
MCS_N/D5	I/O					√			√
MCLK/D4	I/O					√			√
FASTRD_N/D3	I/O					√			√
SI/D2	I/O				√				√
SO/D1	I/O				√				√
SSPI_CS_N/D0	I/O				√				√
SCL	I			√					
SDA	I/O			√					

注！

- 不同型号和封装的器件支持的配置模式不同，详细信息请参考 5 配置模式。
- 关于不同配置模式下各管脚的定义请参考 7 配置模式。

4.1.2 配置管脚复用

为最大化地提高 I/O 的利用率，高云半导体的 FPGA 产品支持将配置管脚设置为普通 I/O 使用。所有系列的 FPGA 上电后未进行任何配置操作之前，与配置相关的管脚均默认作为配置管脚使用。配置成功后，器件进入用户模式，按照用户选择的复用选项重新分配管脚的功能。

注！

用户设置管脚复用选项时，需确保管脚的外部初始连接状态不影响器件的配置过程。对于影响配置的连接，需要先进行隔离处理，等待 FPGA 进入用户模式后再进行修改。

配置管脚复用选项如表 4-2 所示。

表 4-2 配置管脚复用选项

设置名称	设置选项	说明
JTAG PORT	默认状态	TMS，TCK，TDI，TDO 作为专用配置管脚，JTAGSEL_N 作为 GPIO。
	设置为普通 I/O	JTAGSEL_N 作为专用配置管脚： ● JTAGSEL_N=0，TMS,TCK,TDI,TDO 作为配置管脚。 ● JTAGSEL_N=1，TMS,TCK,TDI,TDO 在配置结束后作为 GPIO。
I^2C PORT	默认状态	SCL,SDA 作为专用配置管脚。
	设置为普通 I/O	SCL,SDA 在配置结束后作为 GPIO。
SSPI PORT	默认状态	SCLK，CLKHOLD_N，SSPI_CS_N，SI 和 SO 作为专用配置管脚。

设置名称	设置选项	说明
	设置为普通 I/O	SCLK，CLKHOLD_N，SSPI_CS_N，SI 和 SO 在配置结束后作为 GPIO。
MSPI PORT	默认状态	FASTRD_N，MCLK，MCS_N，MO 和 MI 作为专用配置管脚。
	设置为普通 I/O	FASTRD_N，MCLK，MCS_N，MO 和 MI 在配置结束后作为 GPIO。
RECONFIG_N	默认状态	专用配置管脚。
	设置为普通 I/O	配置结束后作为 GPIO。
READY	默认状态	专用配置管脚。
	设置为普通 I/O	配置结束后作为 GPIO。
DONE	默认状态	专用配置管脚。
	设置为普通 I/O	配置结束后作为 GPIO。

注！

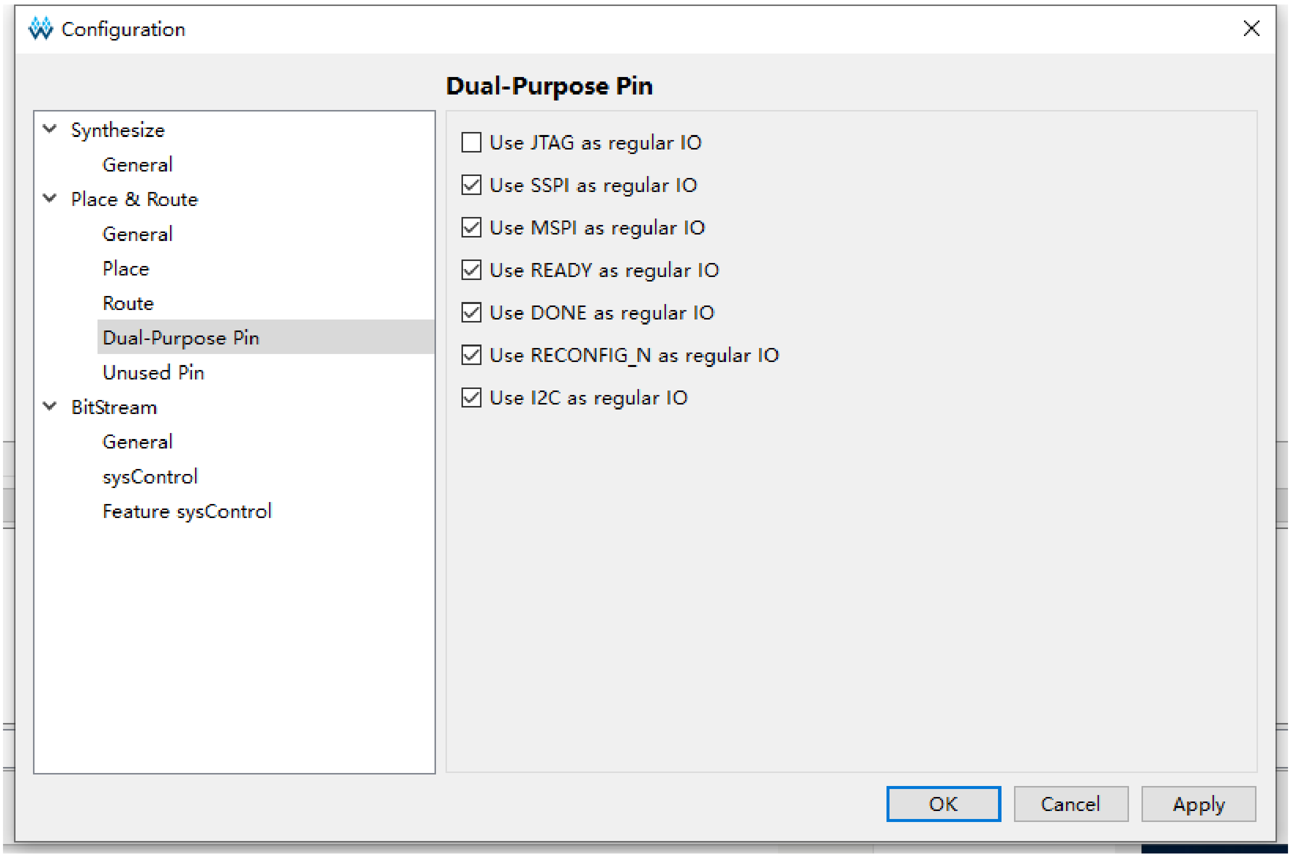
- [1]对于 JTAGSEL_N 未封装出来的器件，在调试 JTAG 管脚复用的案例时，建议用户在 FPGA 上电前确保当前 MODE 值不会使 FPGA 进行配置操作，以避免其他比特流数据影响配置过程。用户上电后手动进行 JTAG 配置后，器件进入用户模式，JTAG 管脚变为 GPIO。对于小蜜蜂(LittleBee)家族 FPGA，当 MODE[2:0]=001 时，JTAGSEL_N 始终为 GPIO，此时可将 JTAGSEL_N 和 JTAG 配置的 4 个管脚（TCK、TMS、TDI、TDO）同时用作 GPIO，但是此时 JTAGSEL_N 无法将 JTAG 管脚恢复为配置 IO，需要让器件重新进入编辑模式以恢复。
- [2]SERIAL 和 CPU 配置模式的管脚由于与其他配置模式共用，无法单独设置为 GPIO，但是当这些管脚工作在非共用配置模式时可以设置为 GPIO。

配置管脚复用

通过 Gowin 云源软件配置管脚复用：

1. 打开 Gowin 云源软件中相应的工程。
2. 在菜单栏中选择“Project>Configuration>Dual-Purpose Pin”，如图 4-1 所示。
3. 勾选对应选项设置配置管脚的复用情况。

图 4-1 配置管脚复用设置



4.2 配置管脚功能及应用

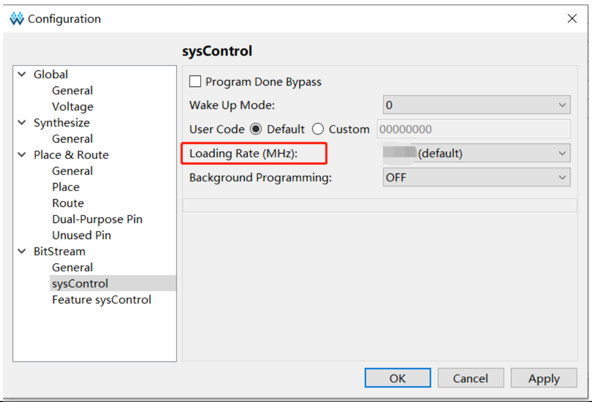
RECONFIG_N、READY 和 DONE 管脚是每种模式都会用到的管脚。其他配置管脚可根据具体应用设置为配置专用管脚或普通管脚。关于各个配置管脚在不同阶段时的状态，请参考附录 A 管脚状态信息。

表 4-3 管脚功能

管脚名称	功能描述
RECONFIG_N	作为配置管脚时，类型为输入，具有内部弱上拉。低电平有效，相当于 FPGA 编程配置的复位功能，**RECONFIG_N 拉低时 FPGA 无法进行任何方式的配置操作。FPGA 上电、初始化、及配置过程中务必保持高电平或悬空，配置完成之后可以释放。** 作为配置管脚时，需要一个脉冲宽度不少于 25ns 的低电平启动 GowinCONFIG 配置模式，使器件按照 MODE 设置值重新加载比特流数据。用户也可以通过编写逻辑控制此管脚，从而按照自身需求触发器件进行重新配置；**作为 GPIO 时，只能用作 output 类型**。为保障配置过程顺利进行，用户复用 RECONFIG_N 管脚时需将其初始值置为高电平。
READY	inout 类型管脚。默认状态为 open-drain 输出，内部弱上拉。高电平有效，只有 READY 拉高时 FPGA 才能进行配置操作，READY 信号拉低后采用器件上电或触发 RECONFIG_N 的方式恢复状态。 作为配置管脚时，若为 output 类型，可以指示 FPGA 当前能否进行配置：当器件具备配置条件时，READY 信号为高电平；若配置失

管脚名称	功能描述
	败，则 READY 信号变为低电平。若为 input 类型，用户可通过自身逻辑或在器件外部人为拉低 READY 信号以延迟配置过程。 作为 GPIO 时，可用作 input 或 output 类型。**READY 用作 input 类型的 GPIO 时需保证配置开始前其初始值为 1，否则 FPGA 无法进行配置。**
DONE	inout 类型管脚。默认状态为 open-drain 输出，内部弱上拉，在配置期间，DONE 输出 0。FPGA 配置成功的标志信号，配置成功后 DONE 信号拉高。 作为配置管脚时，若为 output 类型，可以指示 FPGA 当前配置过程是否成功：当配置成功时，DONE 信号为高电平，器件进入工作状态；配置过程未完成或配置失败时，DONE 信号保持低电平状态。若为 input 类型，用户可通过自身逻辑或在器件外部人为拉低 DONE 信号以延迟其进入用户模式。RECONFIG_N 或 READY 保持低电平状态时，DONE 信号也会保持在低电平状态。使用 JTAG 电路配置 SRAM 的过程中，DONE 信号的值没有参考意义。 作为 GPIO 时，可用作 input 或 output 类型。**DONE 用作 input 类型的 GPIO 时需保证配置开始前其初始值为 1，否则配置结束后 FPGA 无法进入用户模式。**
MODE	GowinCONFIG 配置模式选择信号。作为配置管脚时，类型为输入，内部弱上拉，最多可达 3-bit 位宽。FPGA 上电或低电平脉冲触发 RECONFIG_N 时，器件根据 MODE 值进入相应的 GowinCONFIG 状态，高云半导体每个系列的 FPGA 产品 MODE 值对应的配置模式略有不同。由于每个封装类型的管脚数目不同，有些器件的 MODE 管脚未完全封装出来，未封装出来的 MODE 管脚请参考相应器件的 PINOUT 手册。 MODE 管脚作为 GPIO 时，可用作 input 或 output 类型。 需要注意的是，当 MODE 值改变时，需要重新上电或低电平触发 RECONFIG_N 才能生效。
JTAGSEL_N	JTAG 配置功能恢复信号。 ● JTAG 配置的 4 个管脚（TCK、TMS、TDI、TDO）被复用为 GPIO 时，JTAGSEL_N 作为配置管脚[1]，类型为输入，内部弱上拉。用于恢复被复用为 GPIO 的 JTAG 管脚的配置功能[1]，低电平有效。 **注！** [1] 对于小蜜蜂(LittleBee)家族 FPGA，当 MODE[2:0]=001(包括输入和输出输出)时，JTAGSEL_N 始终为 GPIO，无法用于恢复 JTAG 管脚的配置功能，即此时可将 JTAGSEL_N 和 JTAG 配置的 4 个管脚（TCK、TMS、TDI、TDO）同时用作 GPIO。要恢复 JTAG 管脚的配置功能，可拉低 MODE[0]后再拉低 JTAGSEL_N，或者让器件重新进入编辑模式。 ● JTAG 配置的 4 个管脚（TCK、TMS、TDI、TDO）未被复用为 GPIO 时，JTAGSEL_N 作为 GPIO，可用作 input 或 output 类型。
TCK	作为配置管脚时，类型为输入。 JTAG 配置模式的串行时钟输入管脚。作为 GPIO 时，可用作 input 或 output 类型。

管脚名称	功能描述
TMS	作为配置管脚时，类型为输入，内部弱上拉。 JTAG 配置模式的串行模式输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
TDI	作为配置管脚时，类型为输入，内部弱上拉。 JTAG 配置模式的串行数据输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
TDO	作为配置管脚时，类型为输出。 JTAG 配置模式的串行数据输出管脚。作为 GPIO 时，可用作 input 或 output 类型。
SCLK	作为配置管脚时，类型为输入。 SSPI、SERIAL 和 CPU 配置模式的时钟输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
CLKHOLD_N	作为配置管脚时，类型为输入，内部弱上拉。 SSPI 和 CPU 配置模式的时钟锁定管脚：在 SSPI 模式下，高电平有效；在 CPU 模式下，低电平有效。作为 GPIO 时，可用作 input 或 output 类型。
SSPI_CS_N	作为配置管脚时，类型为输入，内部弱上拉。SSPI 配置模式的片选信号，低电平有效。作为 GPIO 时，可用作 input 或 output 类型。
SI	作为配置管脚时，类型为输入。SSPI 配置模式的串行数据输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
SO	作为配置管脚时，类型为输出。SSPI 配置模式的串行数据输出管脚。作为 GPIO 时，可用作 input 或 output 类型。
MCLK	作为配置管脚时，类型为输出。 MSPI 配置模式的输出时钟管脚，来源于 FPGA 内部晶振。该管脚的输出频率范围和默认输出频率因器件而异。关于片内晶振的详细数据请参考相应器件的数据手册。 用户可通过云源软件界面修改 MCLK 的频率值[1]，如图 4-2 所示：打开云源软件工程，在菜单栏中选择“Project > Configuration > BitStream > sysControl”，在“Loading Rate (MHz)”下拉列表中选择 MCLK 的频率值。作为 GPIO 时，可用作 input 或 output 类型。 **注！** [1] MSPI 配置模式的时钟频率存在±10%(晨熙家族)或±5%(小蜜蜂家族)的误差。

管脚名称	功能描述
	图 4-2 MCLK 频率设置 
MCS_N	作为配置管脚时，类型为输出。 MSPI 配置模式的片选信号，低电平有效。作为 GPIO 时，可用作 input 或 output 类型。
MI	作为配置管脚时，类型为输入。 MSPI 配置模式的串行数据输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
MO	作为配置管脚时，类型为输出。 MSPI 配置模式的串行数据输出管脚。作为 GPIO 时，可用作 input 或 output 类型。
FASTRD_N	作为配置管脚时，类型为输入。 MSPI 配置模式读取 SPI Flash 速度选择信号：当 FASTRD_N 为高电平时为普通读取模式（指令 0x03）；当 FASTRD_N 为低电平时为高速读取模式，各个厂家的 Flash 高速读取操作指令不同，具体请参考相应 Flash 的数据手册。作为 GPIO 时，可用作 input 或 output 类型。
WE_N	作为配置管脚时，类型为输入。 CPU 配置模式的读写使能信号选择管脚：当 WE_N 为高电平时表示读操作；当 WE_N 为低电平时表示写操作。作为 GPIO 时，可用作 input 或 output 类型。
D0~D7	inout 类型管脚。 CPU 配置模式的数据输入输出管脚，8-bit 位宽。根据 WE_N 的值确定 D0~D7 的输入输出方向。作为 GPIO 时，可用作 input 或 output 类型。
DIN	作为配置管脚时，类型为输入，内部弱上拉。 SERIAL 配置模式的串行数据输入管脚。作为 GPIO 时，可用作 input 或 output 类型。
DOUT	作为配置管脚时，类型为输出。 SERIAL 配置模式的串行数据输出管脚，只在 FPGA 级联时用作后一个器件的输入。作为 GPIO 时，可用作 input 或 output 类型。
SCL	作为配置管脚时，类型为 input。作为 GPIO 时，只可作 input 类型。
SDA	作为配置管脚时，类型为 in/out。作为 GPIO 时，可用作 input 或

管脚名称	功能描述
	output 类型。

5 配置模式概述

5.1 小蜜蜂(LittleBee)家族 FPGA 产品

小蜜蜂(LittleBee)家族 FPGA 产品除了支持业界通用的 JTAG 配置模式外，还支持高云半导体特有的 GowinCONFIG 配置模式。每款器件支持的 GowinCONFIG 配置模式多少取决于不同型号和封装形式。所有非易失器件均支持 JTAG 和 AUTO BOOT 模式，器件支持的配置模式最多可达 6 种，如表 5-1 所示。

表 5-1 配置模式

<table>
<tr><th colspan="2">配置模式</th><th>MODE[2:0][1]</th><th>相关说明</th></tr>
<tr><td colspan="2">JTAG</td><td>XXX[2]</td><td>外部 Host 通过 JTAG 接口对小蜜蜂(LittleBee)家族 FPGA 产品进行配置</td></tr>
<tr><td rowspan="7">GowinCONFIG</td><td>AUTO BOOT</td><td>000</td><td>FPGA 从内置 Flash 读取配置数据进行配置</td></tr>
<tr><td>I^2C[6]</td><td>100</td><td>外部 Host 通过 I2C 接口对 FPGA 产品进行配置</td></tr>
<tr><td>SSPI</td><td>001</td><td>外部 Host 通过 SPI 接口对小蜜蜂(LittleBee)家族 FPGA 产品进行配置</td></tr>
<tr><td>MSPI</td><td>010</td><td>FPGA 作为 Master，通过 SPI 接口[3]从外部 Flash（或其他器件）读取配置数据进行配置</td></tr>
<tr><td>DUAL BOOT[4]</td><td>110</td><td>FPGA 优先选择外部 Flash 读取配置数据进行配置，外部 Flash 配置失败时选择从内部 Flash 进行配置</td></tr>
<tr><td>SERIAL[5]</td><td>101</td><td>外部 Host 通过 DIN 接口对小蜜蜂(LittleBee)家族 FPGA 产品进行配置</td></tr>
<tr><td>CPU[5]</td><td>111</td><td>外部 Host 通过 DBUS 接口对小蜜蜂(LittleBee)家族 FPGA 产品进行配置</td></tr>
</table>

注！

- [1]对于一些 MODE 管脚没有全部封装出来的器件，未封装出来的 MODE 默认已接地(GW1N(R)-2 和 GW1N-1P5 器件除外，需参考相应的 pinout 手册)。
- [2]JTAG 配置模式与 MODE 输入值无关。
- [3]SSPI 和 MSPI 模式的 SPI 接口是互相独立的。
- [4]GW1N(R)-4(包括所有版本)目前暂不支持 DUAL BOOT。
- [5]CPU 配置模式的 SCLK、WE_N 和 CLKHOLD_N 管脚与 SERIAL 配置模式共用，CPU 配置模式的数据总线管脚与 MSPI 和 SSPI 配置模式的管脚共用。
- [6]小蜜蜂(LittleBee)家族 FPGA 产品处于 I^2C 配置模式时，同时支持 Autoboot 模式，芯片上电后，FPGA 先自行从内置 Flash 读取比特流数据完成配置。Autoboot 配置期间，I^2C SDA 线必须保持外部上拉状态，否则设备可能无法正确配置；另外，建议同时外部上拉 SCL 线。请注意：此注释亦适用于 SDA 和 SCL 内部弱上拉的 C 版本器件。

注！

关于配置管脚列表、配置管脚复用及配置管脚功能及应用信息请参考 4 配置管脚介绍。

5.2 晨熙(Arora)家族 FPGA 产品

晨熙(Arora)家族 FPGA 产品除了支持业界通用的 JTAG 配置模式外，还支持高云半导体特有的 GowinCONFIG 配置模式。每款器件支持的 GowinCONFIG 配置模式的多少取决于不同型号和封装形式。器件支持比特流数据加密和安全位设置功能，为用户设计提供了安全性保障。晨熙(Arora)家族 FPGA 产品支持比特流数据的解压缩功能，用户可以将比特流数据进行压缩以节约存储空间。

晨熙(Arora)家族 FPGA 产品支持的配置模式如表 5-2 所示。

表 5-2 配置模式

配置模式		MODE[2:0][1]	相关说明
JTAG		XXX[2]	外部 Host 通过 JTAG 接口对晨熙(Arora)家族 FPGA 产品进行配置。
GowinCONFIG	MSPI[3]	000	FPGA 作为 Master，通过 SPI 接口[3]从外部 Flash（或其他器件）读取配置数据进行配置。
	SSPI[3]	001	外部 Host 通过 SPI 接口对晨熙(Arora)家族 FPGA 产品进行配置。
	SERIAL[4]	101	外部 Host 通过 DIN 接口对晨熙(Arora)家族 FPGA 产品进行配置。
	CPU[4]	111	外部 Host 通过 DBUS 接口对晨熙(Arora)家族 FPGA 产品进行配置。

注！

- [1]对于一些 MODE 管脚没有全部封装出来的器件，未封装出来的 MODE 默认已接地。
- [2]JTAG 配置模式与 MODE 输入值无关。
- [3]SSPI 和 MSPI 模式的 SPI 接口是互相独立的。
- [4]CPU 配置模式的 SCLK、WE_N 和 CLKHOLD_N 管脚与 SERIAL 配置模式共用，CPU 配置模式的数据总线管脚与 MSPI 和 SSPI 配置模式的管脚共用。

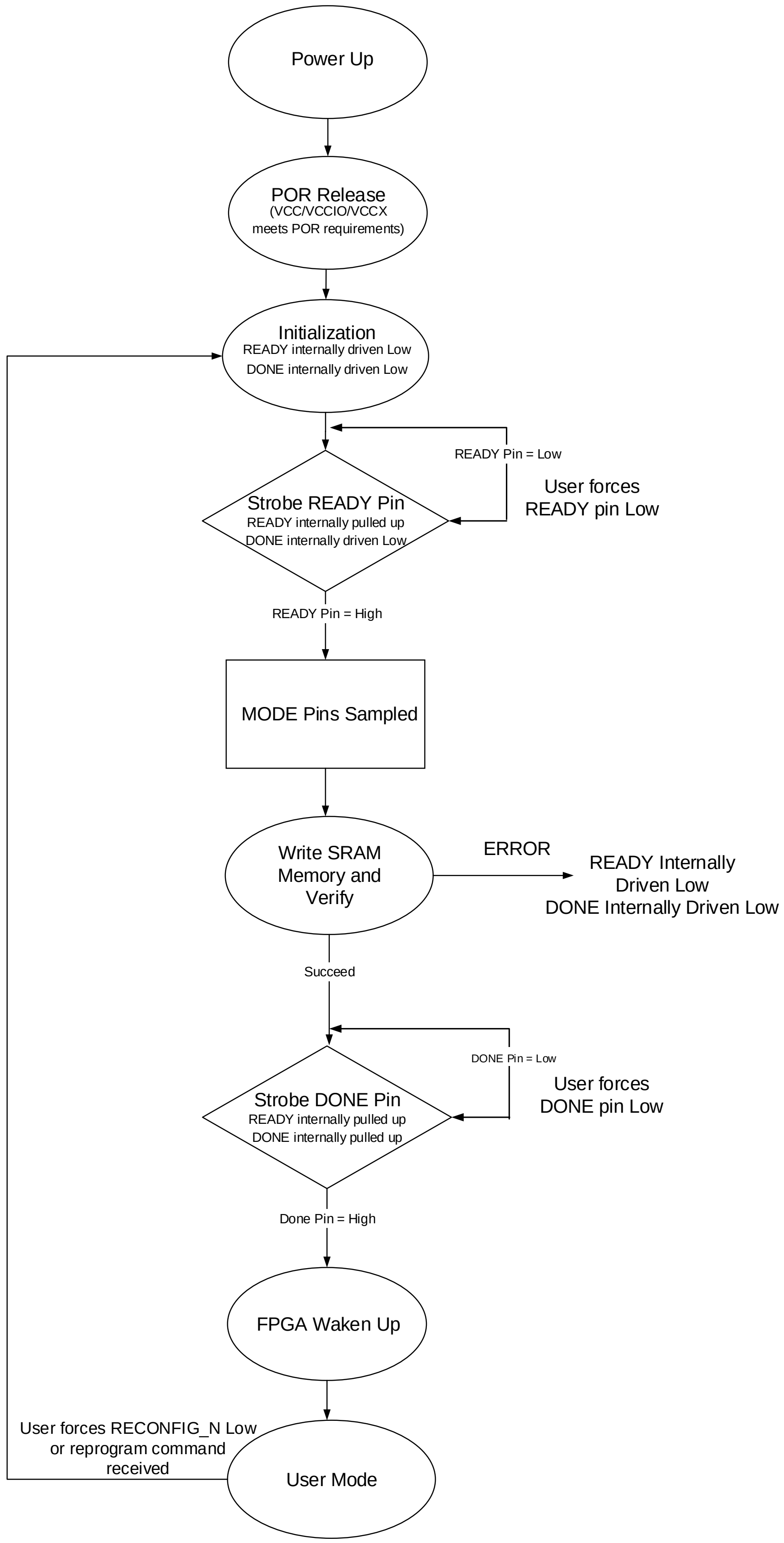
注！

关于配置管脚列表、配置管脚复用及配置管脚功能及应用信息请参考 4 配置管脚介绍。

6 配置流程

高云半导体 FPGA 上电启动后历经初始化、配置 SRAM 和唤醒等几个状态，其配置流程如图 6-1 所示。

图 6-1 高云半导体 FPGA 配置流程图



注！

- READY、DONE、RECONFIG_N 是双向 IO，open drain 输出, 同时内部有弱上拉(上拉电流约为 100uA)。
- 用户可以通过强制拉低 READY(force the READY pin low)来控制器件开始加载的时间点。
- 用户可以通过强制拉低 DONE(force the DONE pin low)来控制器件 wake up 的时间点。
- 从上电到器件加载完整，RECONFIG_N 管脚需要保持为高电平状态。

6.1 上电时序

电源上电的过程中，FPGA 内部的上电复位(POR)电路开始工作。POR 电路确保外部 I/O 管脚处于高阻状态[1]并监控 VCC/VCCX/VCCIOn 电源轨。当 VCC/VCCX/VCCIOn 满足阈值电平时(不同器件的阈值电平不同，不同器件监控的电源轨不同)，POR 电路释放内部复位信号，FPGA 开始初始化流程。当 READY 和 DONE 信号拉低后，器件进入初始化状态，如图 6-2 所示。

注！

[1]详细信息请参考附录 A 管脚状态信息。

图 6-2 POR 上电时序图

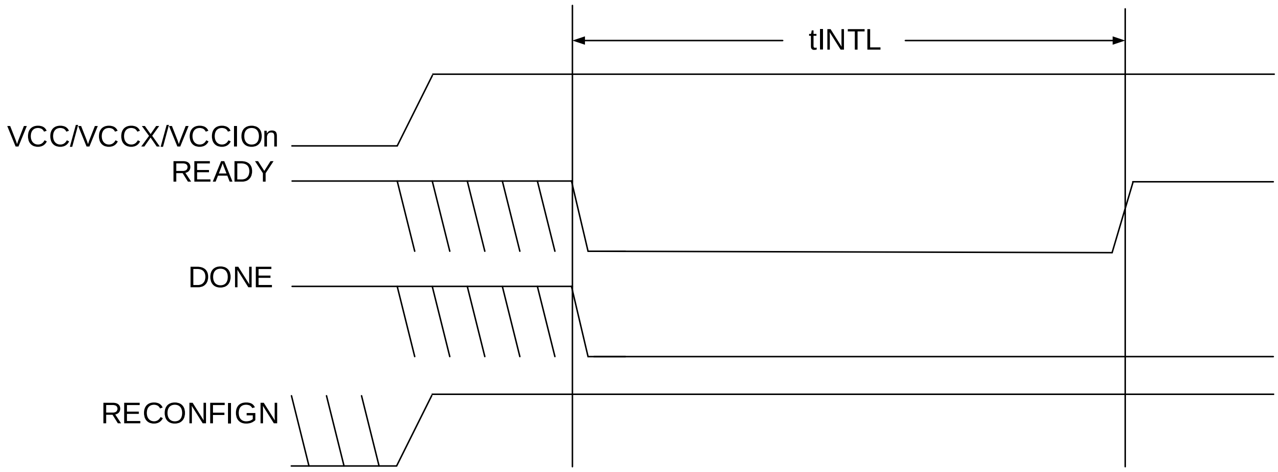


表 6-1 列出了不同器件 POR 模块监控电源轨的详情。

表 6-1 不同器件 POR 模块监控电源轨

系列	器件	POR 模块监控电源轨
GW1N	GW1N-1 GW1N-4 GW1N-9	VCC/VCCX/VCCIO1/VCCIO3
	GW1N-1P5 GW1N-2	VCC/VCCX/VCCIO0
	GW1N-1S	VCC/VCCX/VCCIO0/VCCIO2
GW1NZ	GW1NZ-1	VCC/VCCX/VCCIO1/VCCIO3

系列	器件	POR 模块监控电源轨
GW1NR	GW1NR-1 GW1NR-2 GW1NR-4 GW1NR-9	VCC/VCCX/VCCIO1/VCCIO3
GW1NS	GW1NS-4 GW1NS-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NSR	GW1NSR-4 GW1NSR-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NSE	GW1NSE-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NSER	GW1NSER-4C	VCC/VCCX/VCCIO0/VCCIO1
GW1NRF	GW1NRF-4B	VCC/VCCX/VCCIO1/VCCIO3
GW2A	GW2A-18 GW2A-55	VCC/VCCX/VCCIO3
GW2AR	GW2AR-18	VCC/VCCX/VCCIO3
GW2AN	GW2AN-55	VCC/VCCX/VCCIO3
GW2ANR	GW2ANR-18	VCC/VCCX/VCCIO3

6.2 初始化

在上电复位电路拉低 READY 和 DONE 管脚后，高云半导体 FPGA 立即进入存储器初始化状态。初始化状态的目的是清除 FPGA 内部的配置 SRAM 存储器。

FPGA 满足以下所有条件后跳出初始化状态：

- 初始化状态超过 tINITL
- RECONFIG_N 管脚为高
- READY 管脚不被外部驱动强制拉低

在初始化阶段 READY 管脚提供两种功能：

- 指示 FPGA 正在清除内部的配置 SRAM 区域。
- 作为输入，当被外部强制拉低时可以阻止 FPGA 跳出初始化状态。

6.3 配置

识别到 READY 管脚的上升沿后，FPGA 进入配置状态。根据 MODE 管脚状态，可以通过多种模式配置 FPGA 内部的配置 SRAM。在 FPGA 接收配置数据的期间，可以通过 READY 管脚判别内部状态。READY 管脚高电平指示配置过程正常，READY 管脚低电平指示 FPGA 配置出错，不能正常工作。

6.4 唤醒

当正确接收到所有配置数据后，FPGA 进入唤醒状态并置起内部的 DONE 状态位。在唤醒状态下，FPGA 会依次进行如下操作：

1. 使能全局输出信号(GOE)，FPGA 的 I/O 退出高阻状态[1]，完成 I/O 编程实现预设的功能。通过置位全局置位/复位信号(GSR)，可以防止输入信号影响 FPGA 内部的 Flip-Flop 状态。
2. 释放全局置位/复位信号(GSR)和全局写入禁止信号（GWDISn）。使能全局写入禁止信号可以防止 FPGA 误改写内部 RAM 的初始化数据。

3. 使能外部 DONE 管脚。使能状态下 DONE 管脚是一个双向开漏 I/O。可以通过外部强制拉低 DONE 管脚的方式让 FPGA 保持在唤醒状态。一旦 DONE 管脚拉高，FPGA 将完成唤醒状态，进入用户模式。

注！

[1]详细信息请参考附录 A 管脚状态信息。

6.5 用户模式

进入用户模式后，FPGA 将立刻执行您设计的逻辑运算。FPGA 将保持在用户模式下直到触发以下三种事件：

- 外部拉低 RECONFIG_N 管脚
- 通过配置端口接收到 reprogram 指令
- 电源有下上电动作

一旦出现以上三种操作，FPGA 将重新进入配置流程。

7 配置模式详述

高云半导体 FPGA 产品包含基于 SRAM 工艺的晨熙(Arora)家族的高性能器件和嵌入 Flash 的小蜜蜂(LittleBee)家族的小容量非易失器件。基于 SRAM 工艺的器件掉电后器件内部的配置数据丢失，每次上电后需重新配置；嵌入 Flash 的非易失器件掉电后数据仍然存储在芯片内部，重新上电时可以通过自启动配置或双启动配置的方式由器件自动进行重新配置。

高云半导体 FPGA 产品封装类型丰富，每一种封装的器件支持的配置模式与封装出来的配置管脚数目有关：所有器件均支持通用的 JTAG 配置；只有非易失器件支持自启动或双启动配置；每种配置模式的 MODE 值各不相同。

7.1 配置须知

高云半导体 FPGA 产品目前分为小蜜蜂(LittleBee)家族和晨熙(Arora)家族。两大家族器件名称中是否包含 R 不影响配置特性，主要区别是带 R 的 FPGA 内部集成了 SDRAM/PSRAM。器件名称中包含 S 的 FPGA 除了双启动配置特性与 GW1N 系列略有差异之外，其他特性完全相同。

上电及配置流程

当 FPGA 的 VCC、VCCIO、VCCX 供电电压满足最小供电幅值时，FPGA 进入启动流程：电压稳定且 RECONFIG_N 未被外部电路拉低>FPGA 内部电路拉低 READY 和 DONE 管脚>FPGA 初始化>READY 拉高并采样 MODE 值>根据配置模式读取配置数据并校验>FPGA 唤醒>DONE 拉高>进入用户模式。

FPGA 启动过程中需要保持供电稳定，FPGA 上电、初始化、以及配置过程中 RECONFIG_N 管脚不允许出现低电平，用户可选择将 RECONFIG_N 管脚悬空或外部上拉。从器件上电复位释放到唤醒前，所有 GPIO 为高阻态，内部弱上拉，详细信息请参考附录 A 管脚状态信息。

高云半导体 FPGA 产品按照配置数据的存储和指令的作用位置分为对 SRAM 的操作、对内置 Flash 的操作和对外部 Flash 的操作，其中，对内置 Flash 的操作只有小蜜蜂(LittleBee)家族产品支持，对 SRAM 和外部 Flash

的操作所有产品均可支持。

SRAM 操作

对 SRAM 的操作包括读取器件 ID CODE 和 USER CODE，读取器件状态寄存器信息以及 SRAM 配置。器件 ID 验证是配置操作的前提，只有 ID 验证成功的器件才能进行配置；USER CODE 是为方便用户对 ID CODE 相同的多个器件加以区分进行的编号标识；器件的状态寄存器记录着 FPGA 配置前后的状态信息，用户可据此分析器件状态，状态寄存器的含义请参考表 7-12~表 7-14。SRAM 配置操作时需要注意，只有未设置安全位的比特流数据支持验证功能。设置了安全位的数据任何用户无法进行回读验证。

内置/外部 Flash 操作

对内置 Flash 的操作包括擦除、编程和验证操作。只能通过 JTAG 接口操作内置 Flash，时钟速率不小于 1MHz，时钟速率详见表 7-9 JTAG 的 TCK 频率要求。

注！

使用内置 Flash 配置 SRAM 的操作（自启动配置和双启动配置）和内置 Flash 的编程操作过程中 FPGA 需要保持上电状态并且不能低电平触发 RECONFIG_N，否则可能会对内置 Flash 造成不可修复的破坏。

小蜜蜂(LittleBee)家族的器件(GW1N-4A 除外[1])支持 JTAG[2]背景升级的特性，即器件支持在不影响现有工作状态的情况下通过 JTAG 接口编程内嵌 Flash 或外部 Flash 的操作，编程过程中器件可以按照原有的配置正常工作，编程完成后，低电平触发 RECONFIG_N 即可完成在线升级。此特性适合应用于在线时间长但又需要不定期升级的场所。

注！

- [1]对于 GW1NS-4C、GW1NSR-4C、GW1NSER-4C 器件，若使用了其内嵌的 Cortex-M3，则无法实现 JTAG 背景升级。
- [2] GW1N-1P5C 和 GW1N(R)-2C 可通过使用 goConfig I2C IP 来支持 I^2C 背景升级，推荐使用 JTAG 接口进行背景升级。

配置管脚复用

用户在使用不同配置模式时，需要根据配置管脚的作用，确保 FPGA 工作在已选择的配置模式下。用户端管脚数目不足时，可以通过其他连接方式灵活处理这些管脚，只保留数据传输相关的管脚即可。MODE[2:0]用来选择 GowinCONFIG 的编程配置模式，用户不需要改变模式时可以使用上拉或下拉的方式将其固定在特定的模式，上拉电阻推荐 4.7K，下拉电阻推荐 1K。

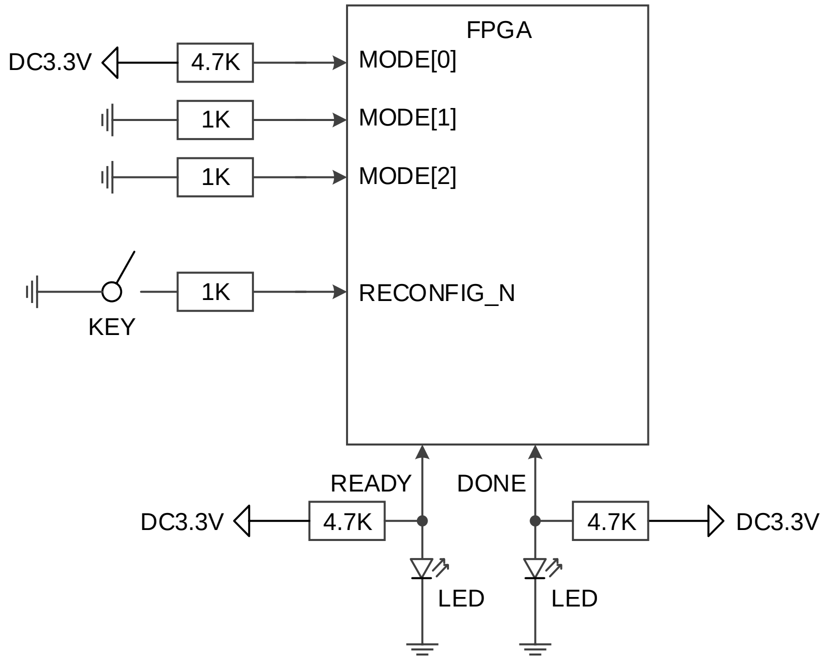
注！

RECONFIG_N、READY 和 DONE 管脚与每种配置模式相关，无论用户是否将它们设置为 GPIO，都需要保证在配置操作完成前它们的初始值或管脚连接状态满足编程配置条件。

固定管脚推荐接法

用户进行电路原理图设计时，推荐接法如图 7-1 所示。

图 7-1 固定管脚推荐接法



注！

- 用户需要改变 MODE 值时可增加拨码开关；部分器件 MODE 管脚未完全封装出来，未封装的 MODE 管脚已内部接地(GW1N(R)-2 和 GW1N-1P5 器件除外，需参考相应的 pinout 手册)。
- JTAG 配置过程中 READY 和 DONE 信号的值没有参考意义。
- RECONFIG_N、READY 和 DONE 未封装出来的管脚已内部处理，不影响配置功能。

重新上电及低电平脉冲触发 RECONFIG_N 时序图

重新上电和低电平脉冲触发 RECONFIG_N 的时序图如图 7-2 和图 7-3 所示。

图 7-2 重新上电时序图

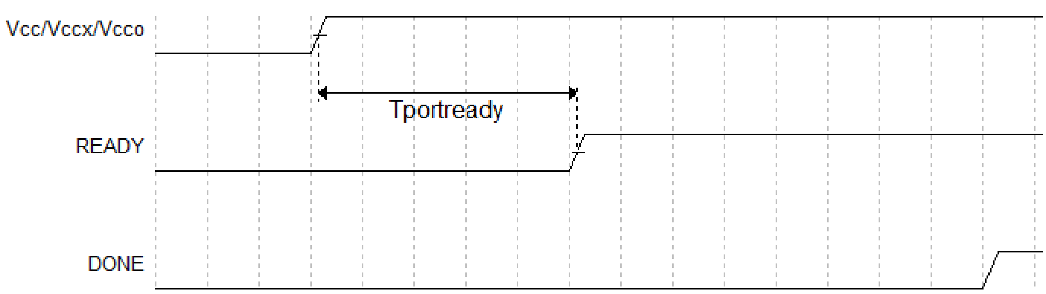
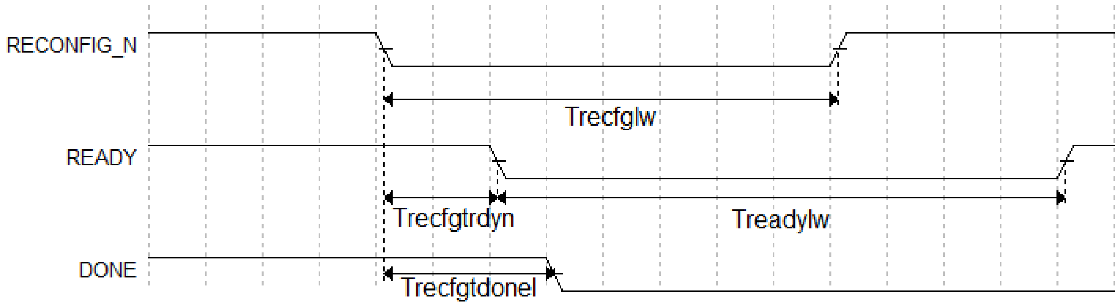


图 7-3 RECONFIG_N 触发时序图



小蜜蜂(LittleBee)家族 FPGA 产品相关的时序参数如表 7-1 所示。

表 7-1 小蜜蜂(LittleBee)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数

参数名称	参数含义	最小值	最大值
$T_{portready}$(MODE0=0)	POR 到 READY 的上升沿的时延（Time from POR to the rising edge of READY）	-	200μs
$T_{portready}$(MODE0=1)	POR 到 READY 的上升沿的时延（Time from POR to the rising edge of READY）	-	50μs
$T_{recfglw}$	RECONFIG_N 低电平脉冲宽度（RECONFIG_N low pulse width）	25ns	-
$T_{recfgtrdyn}$	RECONFIG_N 下降沿到 READY 变低电平的时延（Time from RECONFIG_N falling edge to READY low）	-	70ns
$T_{readylw}$	READY 低电平脉冲宽度（READY low pulse width）	TBD	-
$T_{recfgtdonel}$	RECONFIG_N 下降沿到 DONE 变低电平的时延（Time from RECONFIG_N falling edge to DONE low）	-	80ns

晨熙(Arora)家族 FPGA 产品相关的时序参数如表 7-2 所示。

表 7-2 晨熙(Arora)家族 FPGA 产品重新上电和 RECONFIG_N 触发时序参数

参数名称	参数含义	最小值	最大值
$T_{portready}$(MODE0=0)	POR 到 READY 的上升沿的时延（Time from POR to the rising edge of READY）	-	35ms
$T_{portready}$(MODE0=1)	POR 到 READY 的上升沿的时延（Time from POR to the rising edge of READY）	-	10ms
$T_{recfglw}$	RECONFIG_N 低电平脉冲宽度（RECONFIG_N low pulse width）	25ns	-
$T_{recfgtrdyn}$	RECONFIG_N 下降沿到 READY 变低电平的时延（Time from RECONFIG_N falling edge to READY low）	-	70ns
$T_{readylw}$	READY 低电平脉冲宽度（READY low pulse width）	TBD	-
$T_{recfgtdonel}$	RECONFIG_N 下降沿到 DONE 变低电平的时延（Time from RECONFIG_N falling edge to DONE low）	-	80ns

7.2 JTAG 配置模式

高云半导体 FPGA 产品的 JTAG 配置模式符合 IEEE1532 标准和 IEEE1149.1 边界扫描标准。

JTAG 配置模式是将比特流数据写入到高云半导体 FPGA 产品的 SRAM 中，掉电后配置数据丢失。高云半导体所有封装的 FPGA 产品均支持 JTAG 配置模式。

7.2.1 JTAG 配置模式管脚

JTAG 配置模式的相关管脚如表 7-3 所示。

表 7-3 JTAG 配置模式管脚定义

管脚名称	I/O 类型	说明
JTAGSEL_N[1]	I,内部弱上拉	将 JTAG 管脚从 GPIO 恢复为配置管脚，低电平有效
TCK[2]	I	JTAG 串行时钟输入
TMS	I,内部弱上拉	JTAG 串行模式输入
TDI	I,内部弱上拉	JTAG 串行数据输入
TDO	O	JTAG 串行数据输出

注！

- [1] JTAGSEL_N 信号只有当 JTAG 管脚设置为 GPIO 并且器件启动工作后才起作用；对于小蜜蜂(LittleBee)家族 FPGA，当 MODE[2:0]=001 时，JTAGSEL_N 始终为 GPIO，此时可将 JTAGSEL_N 和 JTAG 配置的 4 个管脚（TCK、TMS、TDI、TDO）同时用作 GPIO，但是此时 JTAGSEL_N 无法将 JTAG 管脚恢复为配置 IO，需要让器件重新进入编辑模式以恢复。
- [2] TCK 需在 PCB 上连接 4.7K 下拉电阻。

在部分器件中，如果 JTAG 的 4 个管脚或 JTAGSEL_N 复用为 GPIO，此时若需重新配置，需要先发送一次 reprogram 指令。具体如表 7-5 所示。

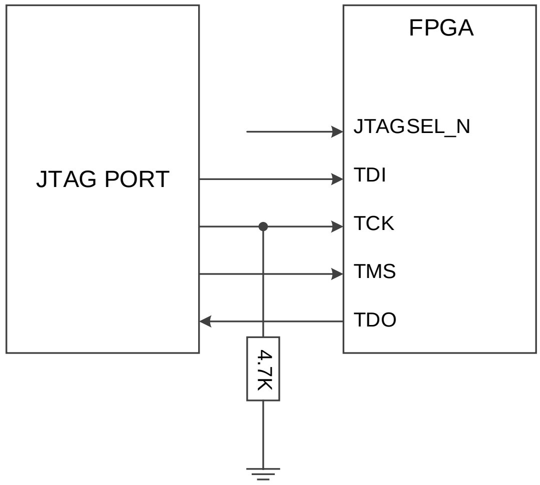
表 7-4 需要/不需要发送 reprogram 指令的器件列表

系列	器件	是否需要发送 reprogram 指令
GW1N	GW1N-1、GW1N-1S、GW1N-4、GW1N-4B、GW1N-4D、GW1N-9、GW1N-9C、	需要发送 reprogram 指令
GW1NZ	GW1NZ-1、GW1NZ-1C、	
GW1NR	GW1NR-1、GW1NR-4、GW1NR-4B、GW1NR-4D、GW1NR-9、GW1NR-9C	
GW1NRF	GW1NRF-4B	
GW2A	GW2A-18、GW2A-18C、GW2A-55C、GW2A-55	
GW2AR	GW2AR-18、GW2AR-18C	
GW2AN	GW2AN-55C	
GW2ANR	GW2ANR-18C	
GW1N	GW1N-1P5、GW1N-1P5B、GW1N-1P5C、GW1N-2、GW1N-2B、GW1N-2C	不需要发送 reprogram 指令
GW1NS	GW1NS-4、GW1NS-4C	
GW1NR	GW1NR-2、GW1NR-2B、GW1NR-2C	
GW1NSR	GW1NSR-4C、GW1NSR-4	
GW1NSER	GW1NSER-4C	

7.2.2 JTAG 配置模式连接示意图

JTAG 配置模式器件连接关系如图 7-4 所示。

图 7-4 JTAG 配置模式连接示意图



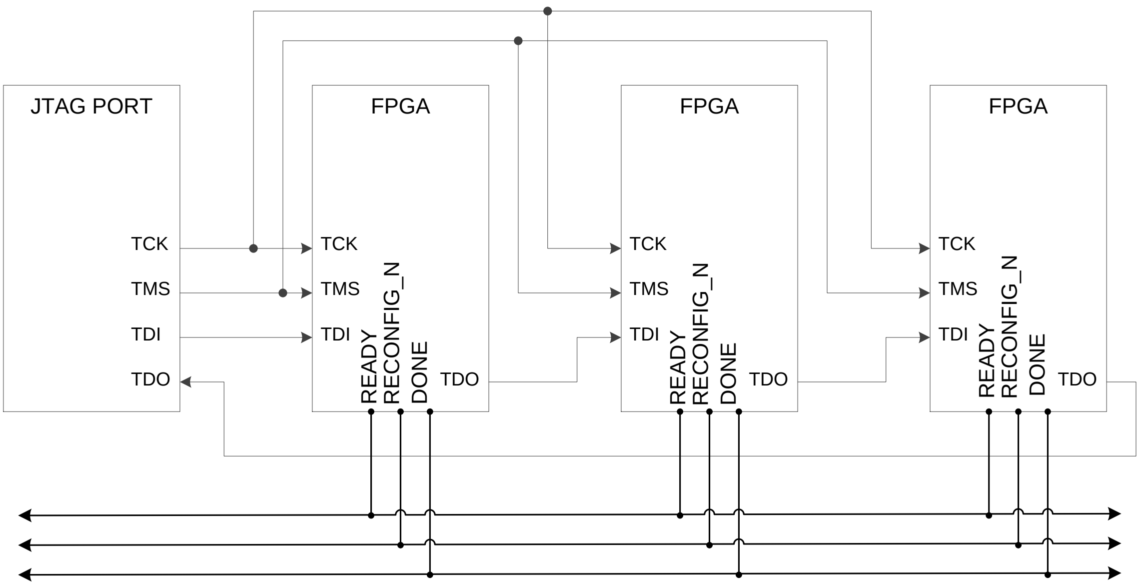
注！

- 对于 JTAGSEL_N 未封装出来的器件，用户在调试 JTAG 管脚复用的案例时，建议上电前将 MODE 值设置为非自动配置的模式（即非自启动、双启动和 MSPI）避免其他比特流数据影响配置过程，用户上电后手动进行 JTAG 配置后，器件进入用户模式，JTAG 管脚变为 GPIO。
- JTAG 配置模式时钟频率不能高于 40MHz。

除了进行常规的 JTAG 配置 SRAM 操作外，高云半导体非易失 FPGA 器件（小蜜蜂(LittleBee)家族）的内置 Flash 和其他所有系列的 FPGA 产品的外部 SPI Flash 的编程操作也可通过 JTAG 管脚进行。非易失器件内置 Flash 的编程操作连线方式与 JTAG 配置模式相同，外部 SPI Flash 的编程操作请参考图 7-51 及 9 边界扫描操作。

此外，高云半导体 FPGA 产品支持 JTAG 菊花链操作，即，把一个 FPGA 的 TDO 管脚连接到下一个 FPGA 的 TDI 管脚，Gowin 编程软件会自动识别连接在一起的 FPGA 器件，依次进行配置。菊花链配置的连接示意图如图 7-5 所示。

图 7-5 JTAG 菊花链配置模式连接示意图



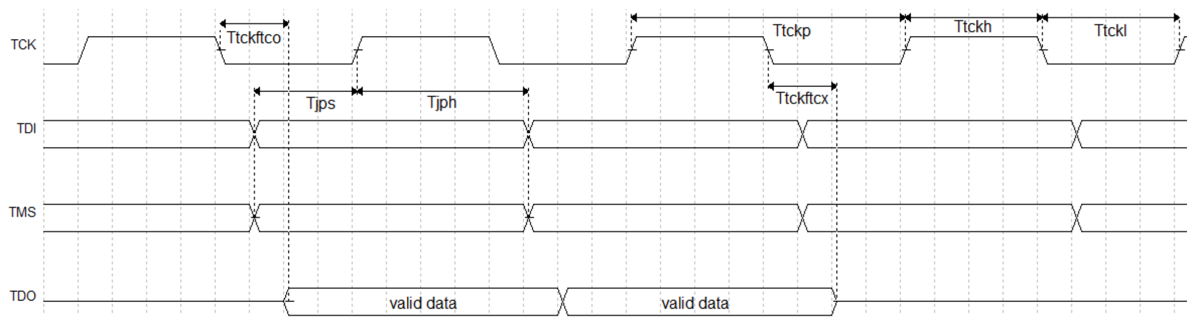
注！

DONE、RECONFIG_N 及 READY 信号视情况决定是否连接。

7.2.3 JTAG 配置模式时序图

JTAG 配置模式的时序图如图 7-6 所示。

图 7-6 JTAG 配置模式时序图



图中各个参数的含义如表 7-5 所示。

表 7-5 JTAG 配置模式时序参数

参数名称	参数含义	最小值	最大值
$T_{tckftco}$	TCK 下降沿到输出数据时延（Time from TCK falling edge to output）	-	10ns
$T_{tckftcx}$	TCK 下降沿到输出高阻时延（Time from TCK falling edge to high impedance）	-	10ns
T_{tckp}	TCK 时钟周期（TCK clock period）	40ns	-
T_{tckh}	TCK 时钟高电平时间（TCK clock high time）	20ns	-
T_{tckl}	TCK 时钟低电平时间（TCK clock low time）	20ns	-
T_{jps}	JTAG PORT 建立时间（JTAG PORT setup time）	10ns	-

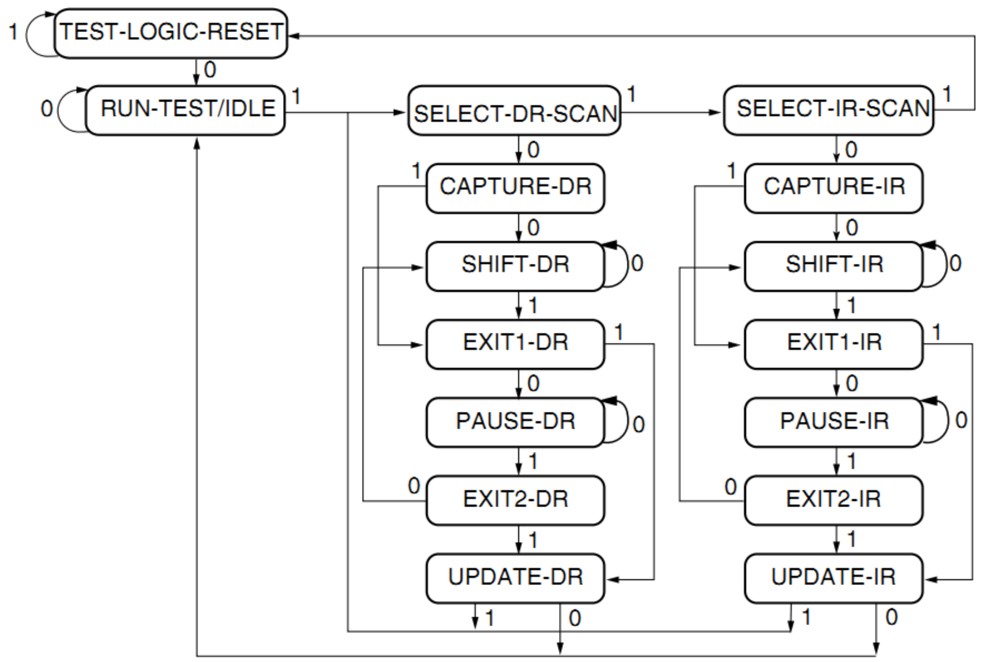
参数名称	参数含义	最小值	最大值
T_{jph}	JTAG PORT 保持时间（JTAG PORT hold time）	8ns	-

7.2.4 JTAG 相关配置流程

TAP 状态机

测试访问口状态机旨在选择指令寄存器或数据寄存器，使其连接至 TDI 和 TDO 之间。一般来说，指令寄存器用于选择需扫描的数据寄存器，在状态机框图中，位于箭头一侧的数字表示 TCK 变高时 TMS 的逻辑状态，如图所示。写一个 JTAG 指令后，要在 Run-Test-Idle 状态保持至少 3 个时钟周期。

图 7-7 TAP 状态机



TAP 复位

通过保持 TMS 为高电平（逻辑“1”）并在 TCK 端输入至少 5 个选通脉冲（变高后再变低）后，复位 TAP 逻辑，从而实现将处于其它状态的 TAP 状态机转换成测试逻辑复位状态，对 JTAG 接口和测试逻辑复位。

注！

该状态不复位 CPU 和外设。

注！

- 在进入 Shift_DR 或 Shift_IR 状态时，TDO 上的数据从 TCK 的下降沿开始有效。
- 在进入 Shift_DR 或 Shift_IR 状态时，数据不移位。
- 在离开 Shift_DR 或 Shift_IR 时，数据被移位。
- 最先移出的是数据的最低位 LSB。
- 一旦复位，所有指令将被重置或失效。

指令寄存器和数据寄存器

除测试逻辑复位外，状态机亦可控制两个基本操作：

- 指令寄存器（IR）扫描。
- 数据寄存器（DR）扫描。

在指令寄存器扫描操作中，在 Shift_IR 状态时，传送数据或指令给指令寄存器，发送时采用 LSB 的方式，低数据位首先被发送，回到 Run-Test-Idle 后指令即被发送完毕，如图 7-8 所示。

在数据寄存器扫描操作中，在 Shift_DR 状态时，传送数据或指令给数据寄存器，如图 7-9 所示。数据发送采用 LSB 还是 MSB 取决于具体操作。

图 7-8 指令寄存器访问时序

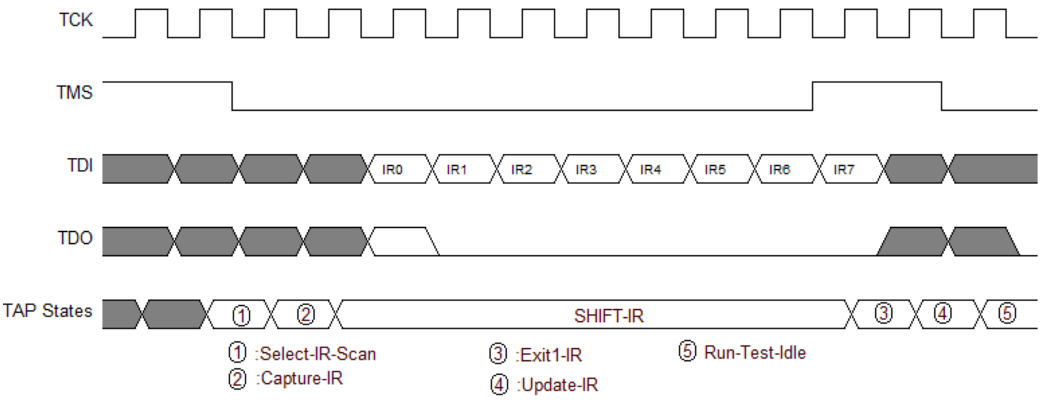
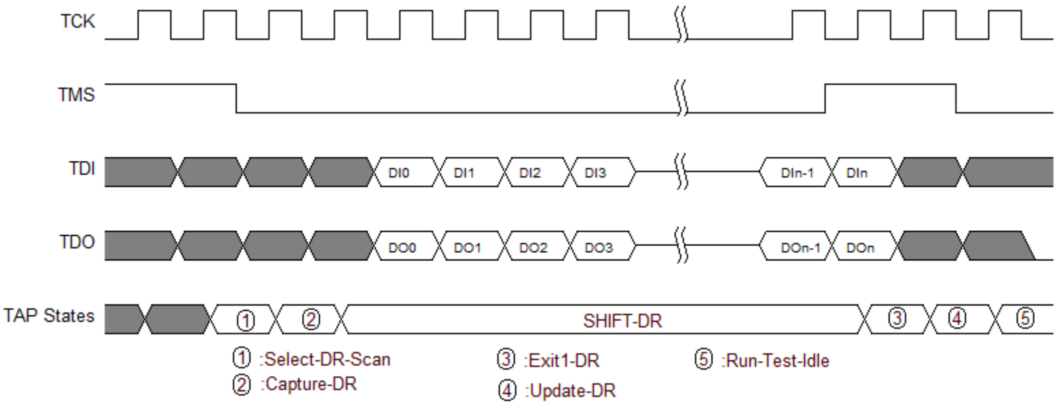


图 7-9 数据寄存器访问时序



注！

- 在高云半导体 GW1N(R)、GW2A(R)系列 FPGA 中，指令寄存器的总长度为 8 位。
- 根据所选择的寄存器，数据寄存器的长度可变化。

读取 ID CODE 实例

ID Code 即 JEDEC ID Code，是 FPGA 器件的一个基本标识。

高云 FPGA ID Code 长度为 32 位，下表列出了高云部分 FPGA 的 ID Code。

表 7-6 Gowin FPGA ID CODE

Gowin FPGA Device Family ID CODE			
Device Family	Device Part Bits 31-12	Manufacturer ID Bits 11-0 h81B	ID CODE
GW1N-1	h09002	h81B	h0900281B
GW1N-1S	h09003		h0900381B
GW1NZ-1	h01006		h0100681B
GW1N(R/Z)-2/2B/2C	h01206		h0120681B
GW1N-1P5/1P5B/1P5C	h01206		h0120681B
GW1N(R)-4	h01003		h0100381B
GW1N(R)-4B	h11003		h1100381B
GW1N(R)-4D	h11003		h1100381B
GW1NS(ER)-4C	h01009		h0100981B
GW1N(R)-9	h11005		h1100581B
GW1N(R)-9C	h11004		h1100481B
GW2A(R)-18/18C	h00000		h0000081B
GW2A-55/55C	h00002		h0000281B

读取 FPGA 的指令是 0x11，以下步骤以读取 GW1N-4 ID Code 为例说明 JTAG 的工作方式。

1. TAP 复位：TMS 置为高电平，连续发送至少 5 个时钟周期。
2. 移动状态机从 Test-Logic-Reset 到 Run-Test-Idle。
3. 移动状态机到 Shift-IR，从最低位开始发送 Read ID 指令 0x11，最高位（最后一位）发送的同时移动状态机到 Exit1-IR，即最高位发送前 TMS 要置于高电平，表 7-7 给出 8 个时钟周期内发送 0x11 过程中 TDI 和 TMS 的值变化，时序如图 7-11 所示。

表 7-7 发送指令过程中 TDI 和 TMS 的值变化

	TCK 1	TCK 2	TCK 3	TCK 4	TCK 5	TCK 6	TCK 7	TCK 8
TDI value (0x11)	1	0	0	0	1	0	0	0
TMS value	0	0	0	0	0	0	0	1

4. 移动状态机，从 Exit1-IR 经过 Update-IR 后回到 Run-Test-Idle，并在 Run-Test-Idle 运行至少 3 个时钟周期。
5. 移动状态机到 Shift-DR，发送 32 个时钟周期，并在第 32 个时钟发送前，置 TMS 为高电平，完成 32 个时钟周期的同时，跳出 Shift-DR 到 Exit1-DR。这期间，发送 32 个时钟即可读出 32bits 数据，即为 0x0100381B，如图 7-12 所示。
6. 移动状态回到 Run-Test-Idle。

图 7-10 读取 ID Code 状态机流程图

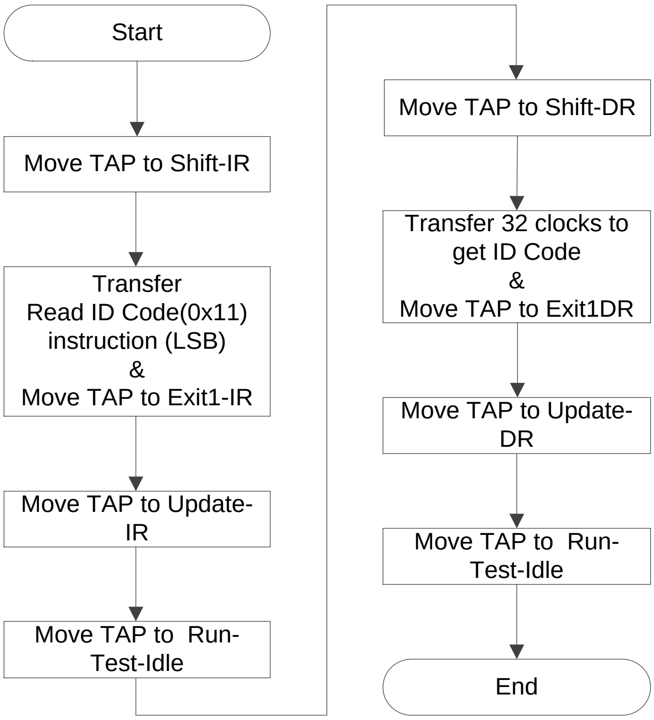


图 7-11 读取 ID Code 指令-0x11 访问时序

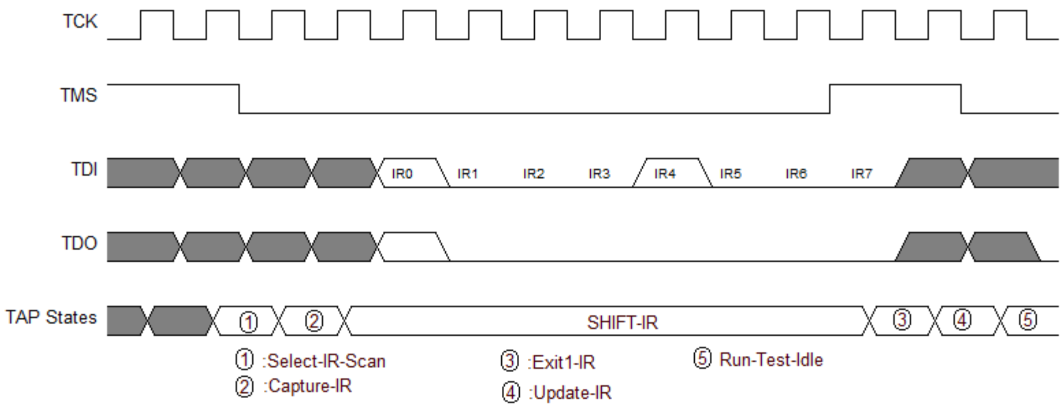
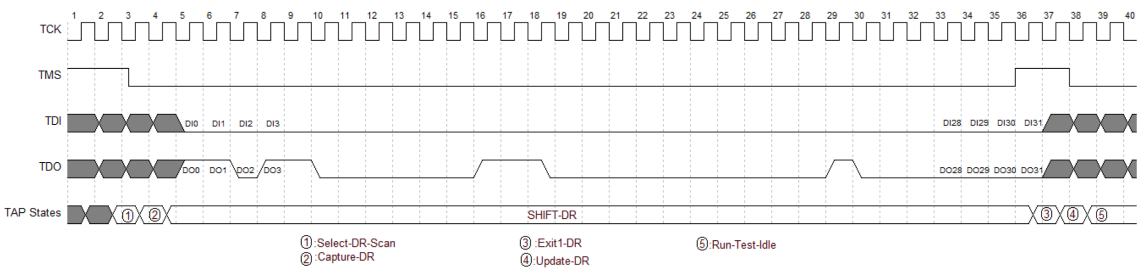


图 7-12 读取 ID Code 数据寄存器访问时序



配置 SRAM 的流程

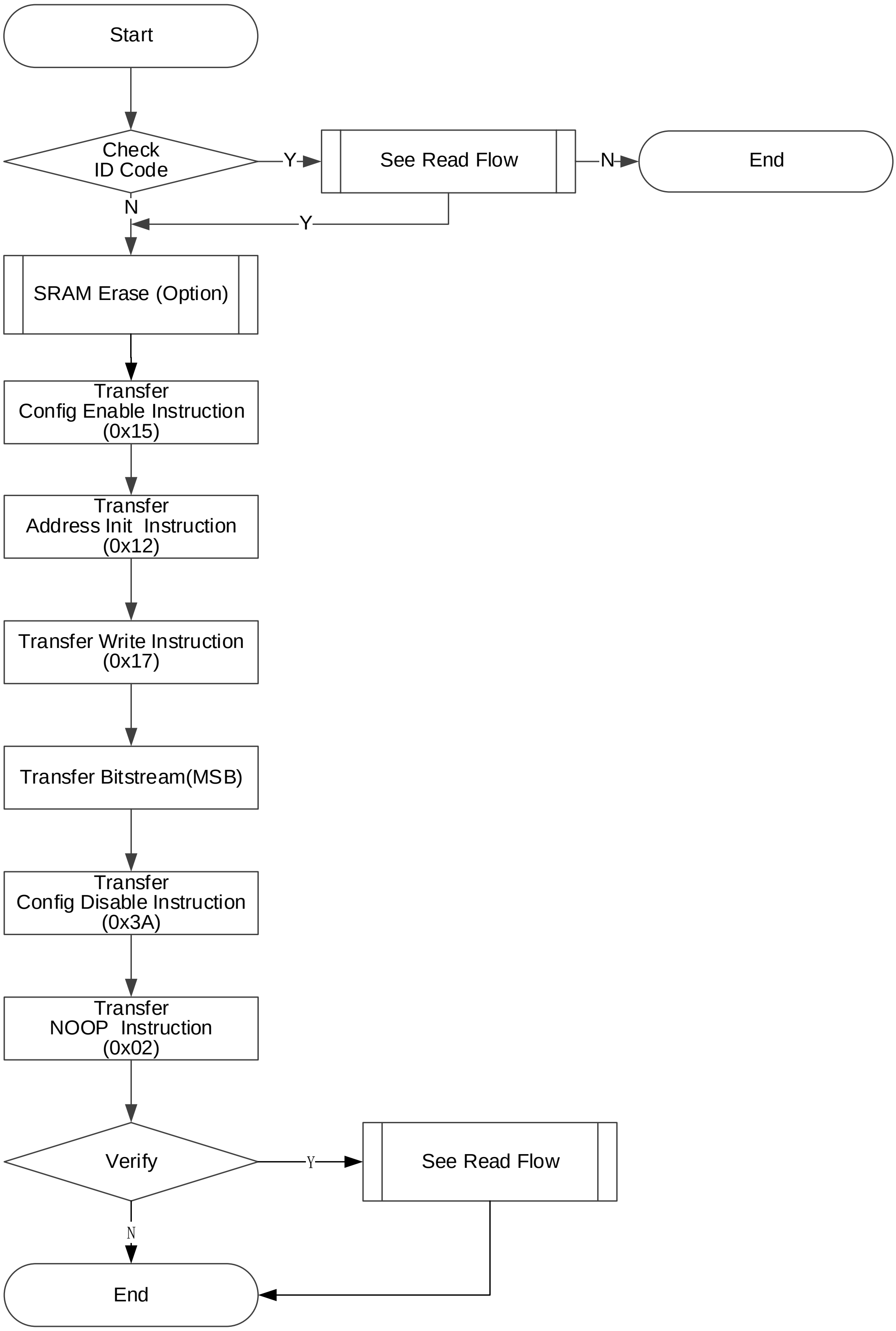
通过外部 Host 配置 FPGA SRAM，使 FPGA 实现功能，通过 JTAG 配

置 SRAM 不受 Configration Mode Pins 的影响。SRAM 写完后等待 60ms，以待 status code 刷新。

通过 Gowin software 设计软件生成数据流文件，利用 JTAG 实现 SRAM 的配置，下面介绍外部 Host 配置 SRAM 的过程，如图 7-13 所示。

1. 建立 JTAG 链路，TAP 复位。
2. 读取设备 ID CODE，检查 ID CODE 是否匹配。
3. 如 SRAM 已被配置，擦除 SRAM，流程参考“擦除 SRAM 的流程”。
4. 发送 ConfigEnable 指令 0x15。
5. 发送 Address Initialize 指令 0x12。
6. 发送 Transfer Configuration Data 指令 0x17。
7. 移动状态到 Shift-DR（数据寄存器），将 Bitstream Data 从最高位开始（MSB），逐位发送，发送全部数据流文件内容，并回到 Run-Test-Idle 状态。
8. 发送 Config Disable 指令 0x3A。
9. 发送 Noop 指令 0x02，结束配置流程。
10. 如需回读 Configuration Data 进行校验，请参考“读取 SRAM 的流程”。

图 7-13 配置 SRAM 流程



读取 SRAM 的流程

警告：出于数据保密设计，SRAM 数据默认不被允许回读。

从 FPGA 的 SRAM 区域读取 SRAM 数据，首先应保证写入 SRAM 时未配置安全位（Security Bit），安全位是用于保护运行时数据，保证数据安全。安全位完成设置后，从 SRAM 取回的数据均为 1（高电平）。

在加载过程中，FPGA 对写入数据进行 CRC 校验，以确保数据写入正确，CRC 是否报错，可以作为配置 SRAM 的校验机制。

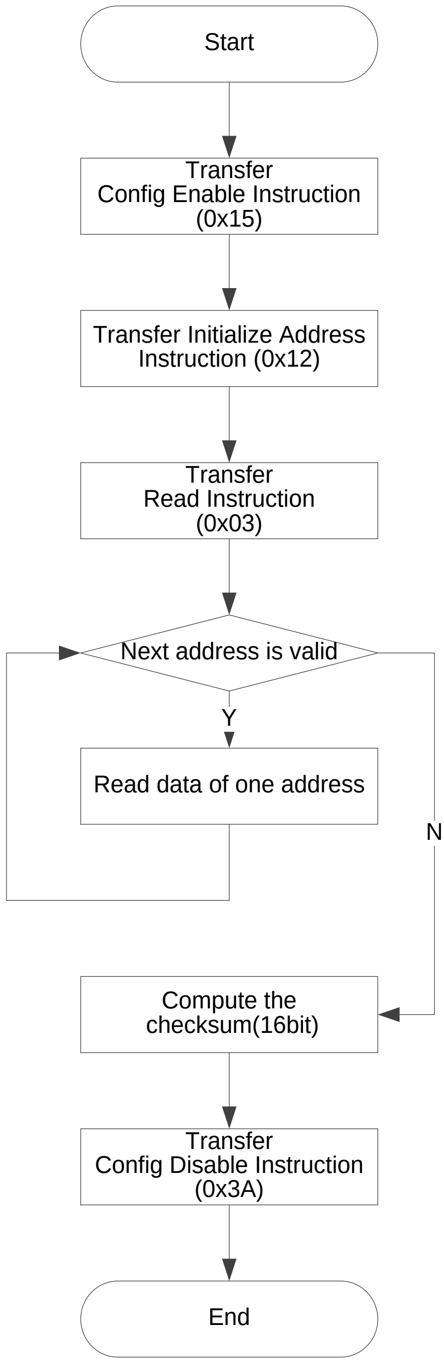
表 7-8 器件 SRAM 地址数量和地址长度

Device	Length of one address (bits/address)	Count of address
GW1N-1/GW1N-1S/ GW1NZ-1/GW1NR-1	1216	274
GW1N-1P5/GW1N-2/GW1NR-2	1216	466
GW1N(R)-4/GW1NS(R)-4/GW1NS(R)-4C/GW1NSE(R)-4C/GW1NRF-4B	2296	494
GW1N(R)-9	2836	712
GW2A(R)-18/GW2ANR-18	3376	1342
GW2A(R)-55(ES)/GW2AN-55	5536	2038

下面详细介绍读取流程，如图 7-14 所示。

1. 发送 ConfigEnable 指令 0x15。
2. 发送 Address Initialize 指令 0x12。
3. 发送 SRAM Read 指令 0x03。
4. 移动状态机到 Shift-DR（数据寄存器），发送地址长度数量的时钟，请参见表 7-8。在发送最后一个时钟同时拉高 TMS，跳到 Exit1-DR，此时 TDO 读取相应长度的数据。最后回到 Run-Test-Idle。
5. 重复步骤 4，每次读取一个地址的数据，其地址会自动累加。
6. 发送 Config Disable 指令 0x3A。
7. 发送 Noop 指令 0x02，结束读取流程。

图 7-14 读取 SRAM 的流程



擦除 SRAM 的流程

当重新配置 SRAM 时，需要擦除已存在的 SRAM。流程如下：

1. 发送 ConfigEnable 指令 0x15。
2. 发送 SRAM Erase 指令 0x05。
3. 发送 Noop 指令 0x02。
4. 延时或 Run Test 2~10ms。
5. 发送 SRAM Erase Done 指令 0x09。
6. 发送 Config Disable 指令 0x3A。

7. 发送 Noop 指令 0x02，结束流程。

注！

在发送 EraseSram（0x05）指令、Noop（0x02）之后，要给足够的时间等待其擦除完毕：

- GW1N(*)-1 参考时间为 1ms。
- GW1N(*)-4 参考时间为 2ms。
- GW1N(*)-9 参考时间为 4ms。
- GW2A(*)-18 参考时间为 6ms。
- GW2A(*)-55 参考时间为 10ms。

内置 Flash 烧录流程

烧录内置 Flash 分为正常烧录和背景烧录。两种配置的流程图如图 7-15 及图 7-16 所示。

图 7-15 正常烧录流程图

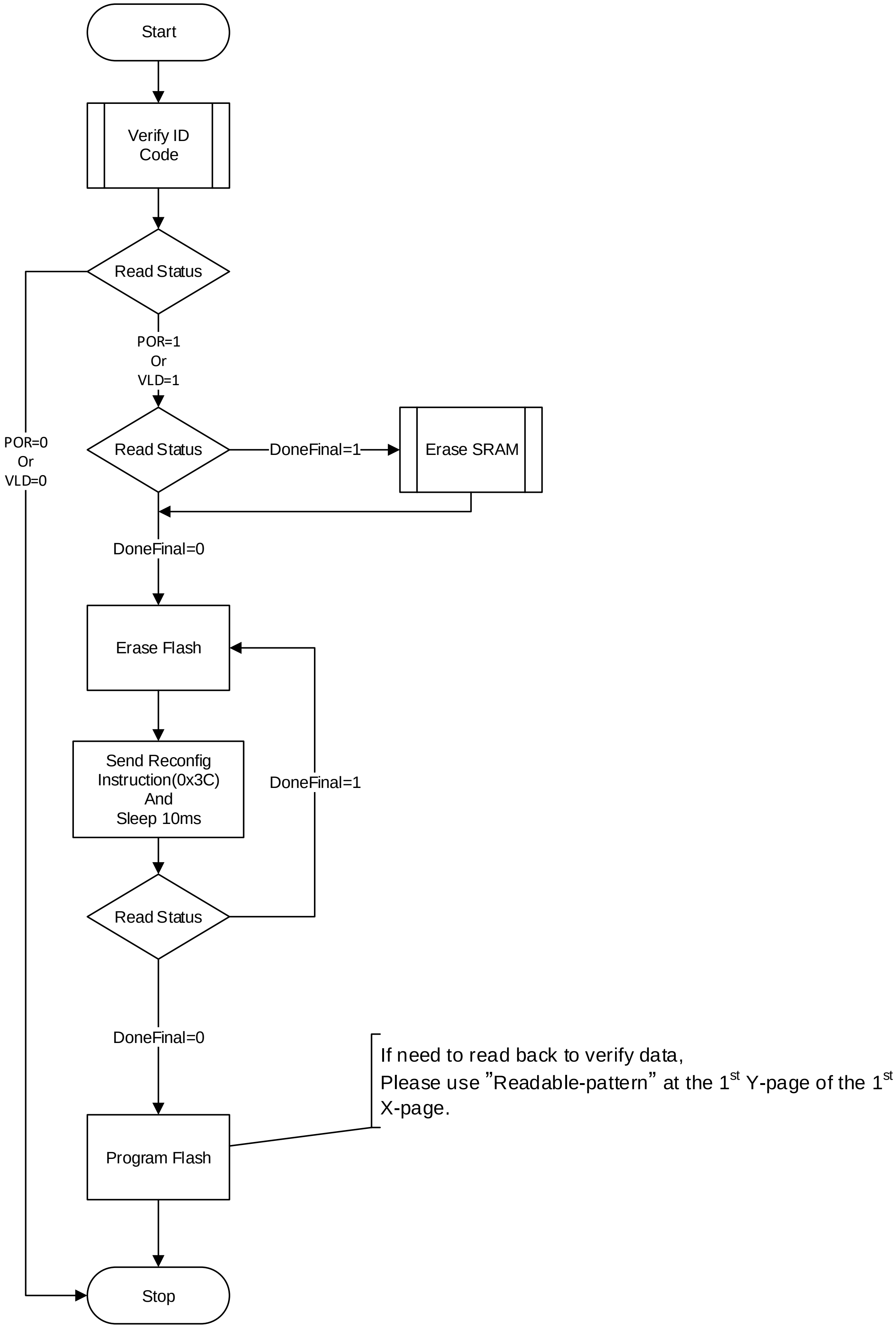
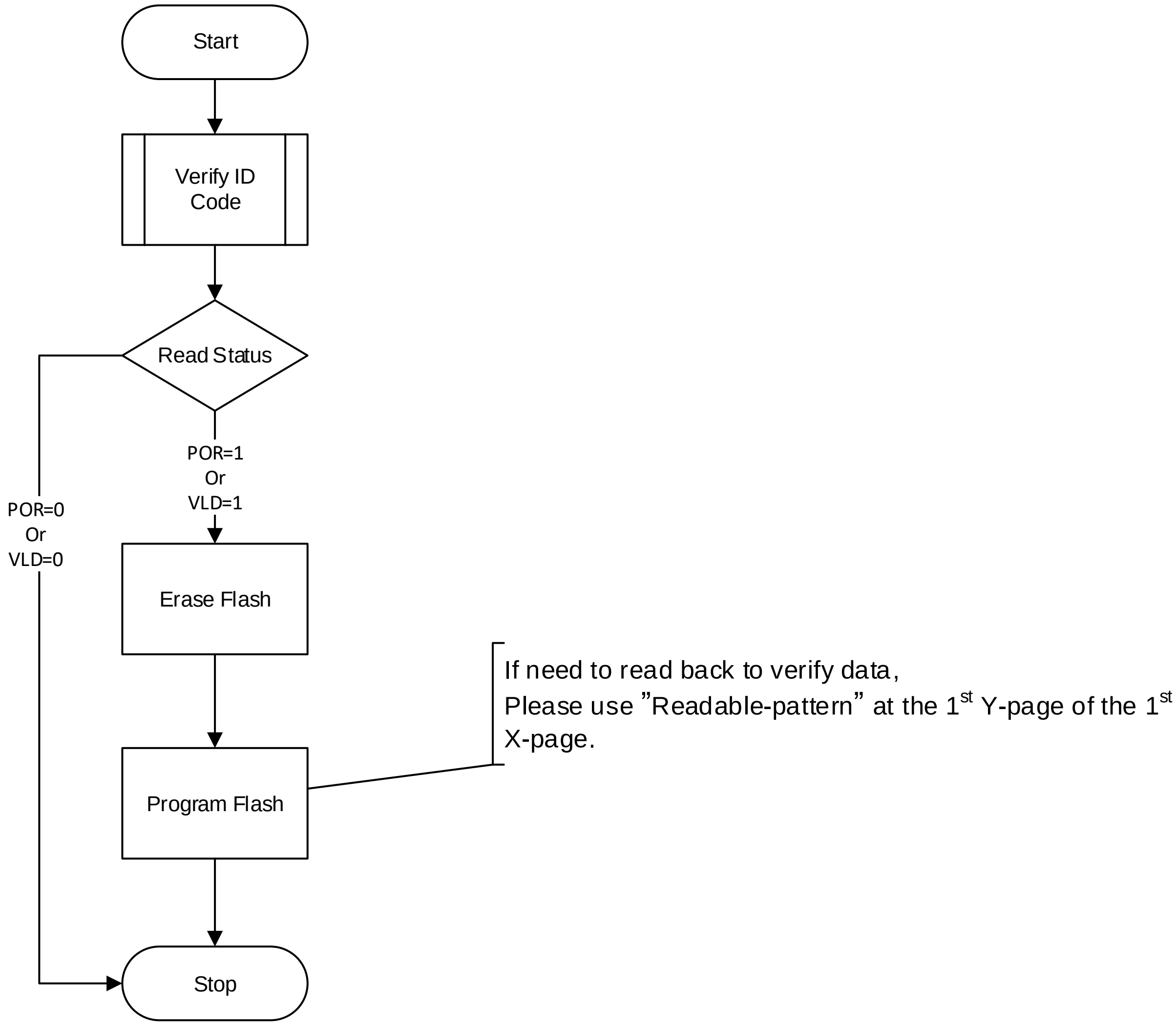


图 7-16 背景烧录流程图



擦除内部 Flash

高云 GW1N 系列内置 Flash 存储器，在每次编程之前需要先擦除内置 Flash，为保证数据安全，内置 Flash 只提供整片擦除的操作。

当前，内置 Flash 因工艺不同，对 JTAG 编程频率有不同要求，请参见表 7-9。

表 7-9 JTAG 的 TCK 频率要求

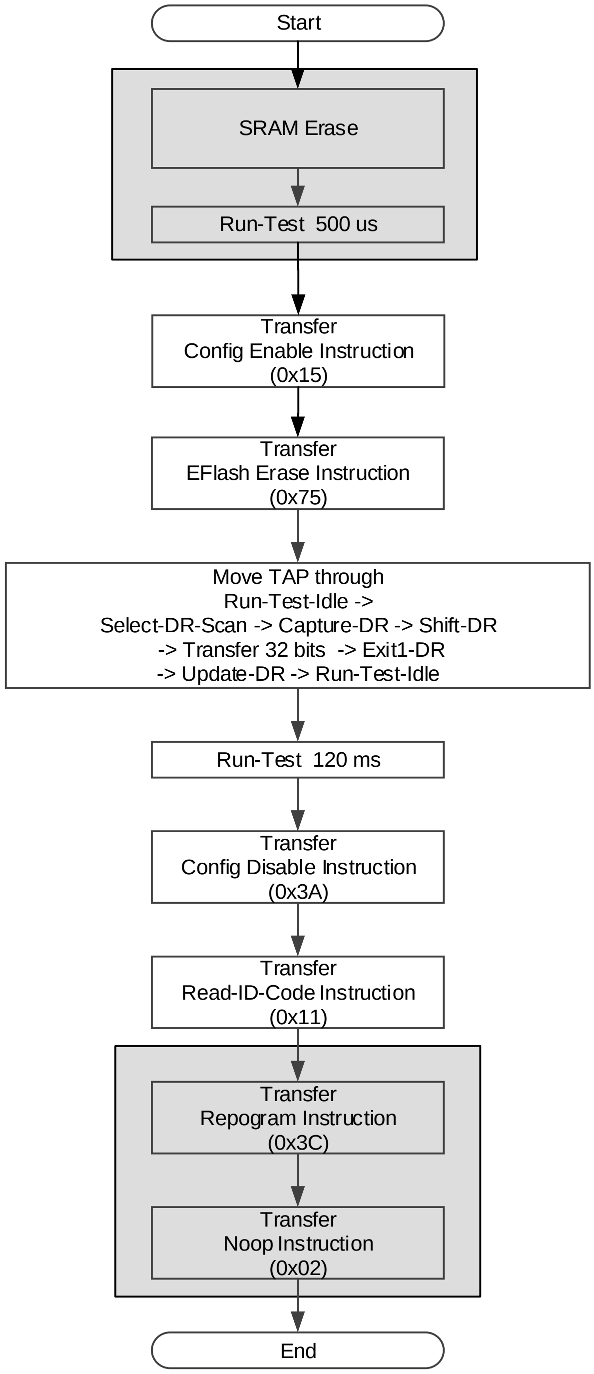
器件	TCK 频率范围	工艺代号
GW1N-1 GW1N-1S	1.4MHz ~ 5MHz	H
GW1N-2, GW1N-1P5	1.3MHz ~ 30MHz	T
GW1N(RF)-4B GW1NSER-4C GW1N(R)-9(C) GW1NZ-1	1.3MHz ~ 30MHz	T
GW2AN-55	0MHz ~ 25MHz	-
GW2ANR-18	0MHz ~ 40MHz	-

T 工艺 FPGA 擦除流程

下面详细介绍 T 工艺，GW1NZ-1 系列芯片的擦除流程（其他型号请忽略），如图 7-17 所示。

1. 建立 JTAG 链路，TAP 复位。
2. 读取设备 ID CODE，检查是否匹配。
3. 如果 SRAM 被配置过，先擦除 SRAM。
4. 在 Run-Test-Idle 持续产生时钟（Run-Test），持续时间为 500μs。
5. 发送 ConfigEnable 指令 0x15。
6. 发送 EFlash Erase 指令 0x75。
7. 依次移动状态机：Run-Test-Idle -> Select-DR-Scan -> Capture-DR -> Shift-DR -> Transfer 32 bits-> Exit1-DR -> Update-DR -> Run-Test-Idle。
8. 在 Run-Test-Idle 持续产生时钟（Run-Test），持续时间为 120ms，此处有频率要求，见表 7-9。
9. 发送 Config Disable 指令 0x3A。
10. 发送 Noop 指令 0x02，擦除流程结束。
11. 发送 Reprogram 指令 0x3C，使器件重配置，检查是否擦除成功。

图 7-17 擦除 T 工艺内部 Flash 擦除流程



注！

背景烧录时，忽略底纹区域操作。

H 工艺 FPGA 擦除流程

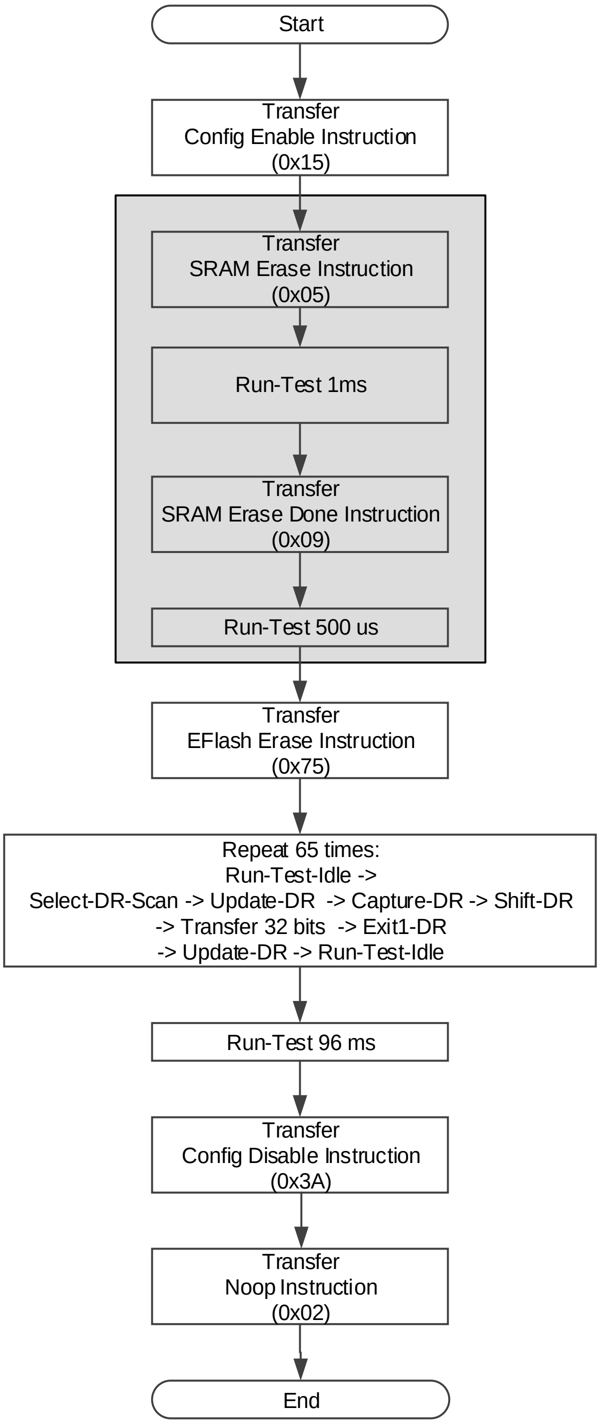
H 工艺 FPGA 芯片的擦除流程：

1. 发送 ConfigEnable 指令 0x15。
2. 发送 EFlash Erase 指令 0x75。
3. 移动状态机，从 Run-Test-Idle 到 Shift-DR，产生 32 个时钟（TDI 信号保持低电平）。在第 32 个时钟同时移动状态机到 Exit1-DR，再经过

Update-DR 回到 Run-Test-Idle。

4. 重复上述步骤，总共 65 次。
5. 在 Run-Test-Idle 持续产生时钟（Run-Test），持续时间为 95ms，此处有频率要求，见表 7-9。
6. 发送 Config Disable 指令 0x3A。
7. 发送 Repogram 指令 0x3C，检验是否擦除成功。
8. 发送 Noop 指令 0x02，擦除结束。

图 7-18 擦除 H 工艺 FPGA 内部 Flash 流程



编程内部 Flash 流程

内置 Flash 以 256Bytes 为一个 X-page，每个 X-page 分成 64 个 Y-page，每 Y-page 包含 4Bytes。

第一个 X-page 的第一个 Y-page，用于标识 Flash 是否可以具备 Autoboot（自动加载）功能或回读功能。如表 7-10 所示。当第一个 Y-page 写入 Readable-pattern 后，可读取 Flash 数据；当第一个 Y-page 写入

Autoboot-pattern 后，器件在 autoboot mode 下会自动把 Flash 数据加载到 SRAM 中；只有写入 Readable-pattern 后才能读取 Flash，其他情况均不能读取。具备 Backgroud programming 功能的器件，仅需使用 Autoboot-pattern。

在不需要回读数据的情况下，必须在数据流文件头部插入 Autoboot-pattern 数据。当一个 X-Page 不足 256bytes 时，可使用 0xFF 或者 0x00 补齐。

当前，GW1N 系列内置 Flash 因工艺不同，对 JTAG 编程频率有不同要求，请参见擦除 SRAM 的流程>表 7-9 JTAG 的 TCK 频率要求。

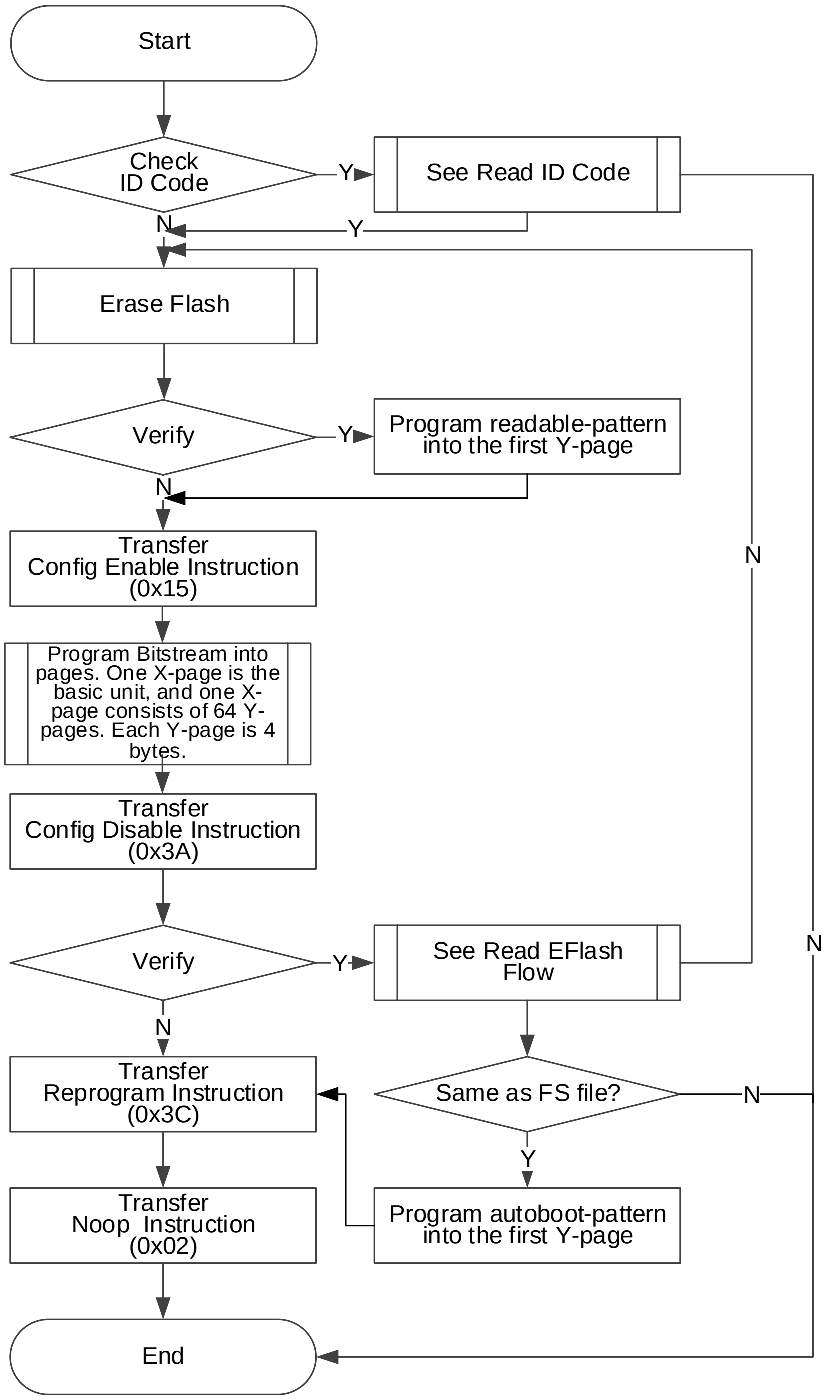
表 7-10 Readable-pattern / Autoboot-pattern

Device	Readable-pattern(4 Bytes)	Autoboot-pattern(4 Bytes)
H 工艺器件	0x07,0x07,0x30,0x40	0x47,0x57,0x31,0x4E
T 工艺器件	0xF7,0xF7,0x3F,0x4F	

编程内部 Flash 流程如图 7-19 所示：

1. 检查 ID Code 是否匹配。
2. 擦除内置 Flash。
3. 验证是否擦除成功，可通过读取 Status 寄存器，看器件是否已还原为裸片的初始状态，对背景烧录和 GW1NS 系列器件不能通过查看 Status 来判断。
4. 发送 ConfigEnable 指令 0x15。
5. 以 X-page 为单位，每次写一个 X-page，直至烧录完成。
6. 发送 Config Disable 指令 0x3A。
7. 发送 Reprogram 指令 0x3C，使器件加载 Flash 的数据到 SRAM。
8. 读取 Status Code/User Code 验证是否加载成功。

图 7-19 编程内部 Flash 流程图



编程一个 X-page 流程

编程一个 X-page 流程如下描述，如图 7-20 所示。

1. 发送 ConfigEnable 指令 0x15。
2. 发送 EF-Program 指令 0x71。
3. 进入 Shift-DR 发送地址数据 [1]。
4. 写入一个 X-page 的数据。
   一个 X-page 共 256 个字节，分 64 次，每次编程 4Bytes（即编程一个

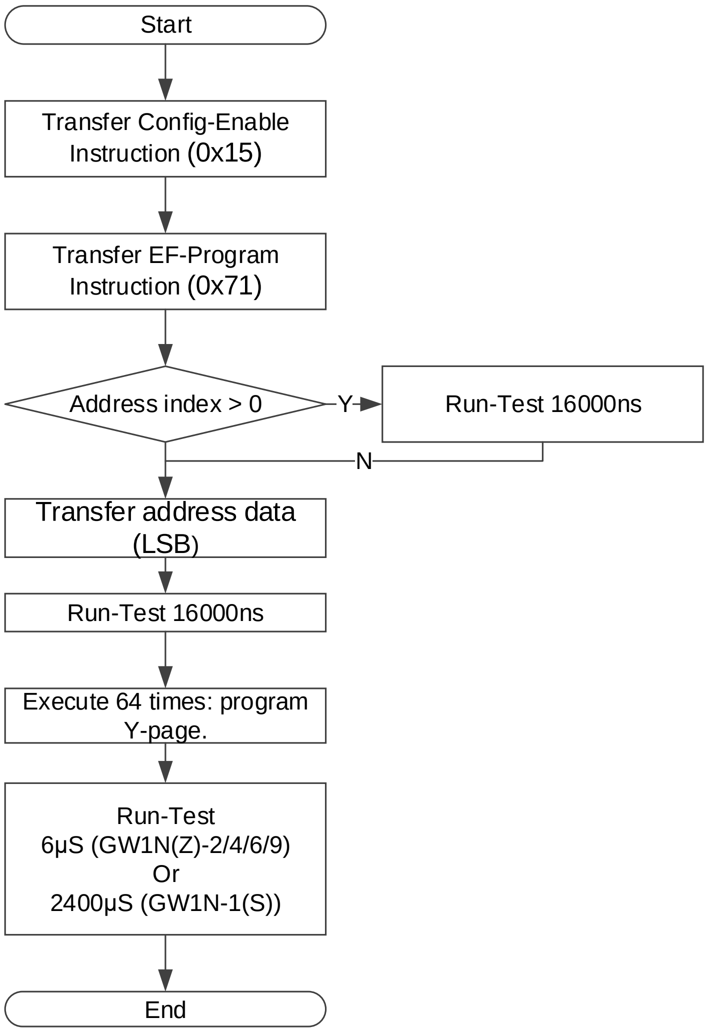
Y-page）。Y-page 数据遵循 LSB 方式写入。此处流程可见图 7-20。

5. 写完一个 X-page 之后，GW1N-1(S)器件需要执行 2400μs 时长的时钟，GW1N(Z)-2/4/6/9 系列器件需要执行 6μs 时长的时钟，其他系列器件不需要额外时钟。
6. 本次 X-page 编程完毕。

注！

[1]地址数据格式共 32bits，其中低 6 位保留，例如地址为 b'**00010011**(0x13)时，写入的地址为 b' **00000000000000000000010011**000000，该地址数据遵循 LSB 方式写入，最后一个 bit 跳出 Shift-DR。

图 7-20 X-page 编程流程图



编程一个 Y-page 流程

Y-page 编程是烧录过程的最小单位，每次写入 4Bytes，数据写入遵循 LSB 方式写入，如图 7-21 所示。

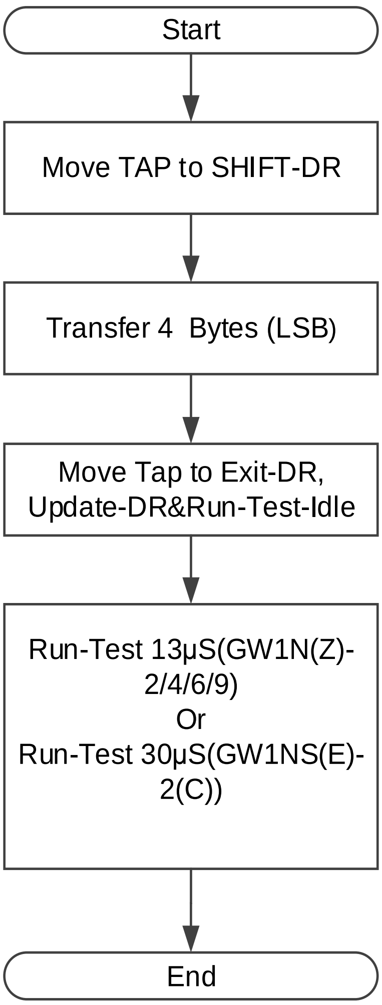
写入完成，不同系列的器件都有要求执行 Run-Test 以等待写入完成，并且要 JTAG 时钟要满足最低频率要求，见表 7-9。

每次写完一个 Y-page, GW1N(Z)-2/4/6/9 系列要求 Run-Test 13-15μs，GW1N-2(C)系列要求 Run-Test 30-35μs，其他系列器件不需要。

注!

数据从 Configuration Data 取高位 4Bytes，在 Shift-DR 写数据时要从最低位开始写入（LSB）。

图 7-21 Y-page 编程流程图



读取内部 Flash 流程

读取内部 Flash 流程概览，对 JTAG 的 TCK 没有速率要求。如图 7-22 所示。

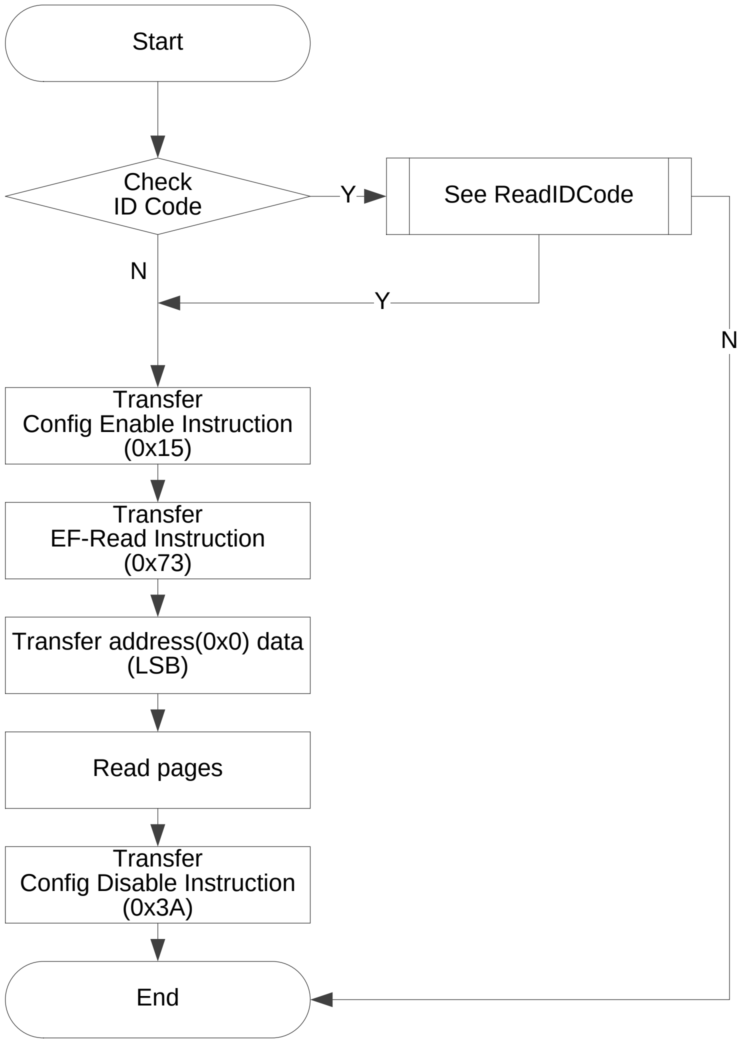
读取内部 Flash 可以理解为是烧录 flash 的逆向过程，但首先要确保写入的 Readable-pattern 已经生效。对于 GW1N 而言，写入 Readable-pattern 后依次发送 Reprogram（0x3C）和 Noop(0x02)可使内部 flash 处于 Readable 状态。

流程简述：

1. 校验 ID Code（可选）。
2. 发送 ConfigEnable 指令 0x15。
3. 发送 EF-Read 指令 0x73。
4. 发送读 Flash 起始地址 0x0。方法同编程内部 Flash 流程中写 X-address 相同。
5. 每读 64 个 Y-page 就是一个 X-page。

6. 每次读完一个 X-page 并不需要重新发送地址，其地址会自动递归。
7. 读取完毕后，发送 ConfigDisable 指令 0x3A 结束流程。

图 7-22 读取内部 Flash 流程图

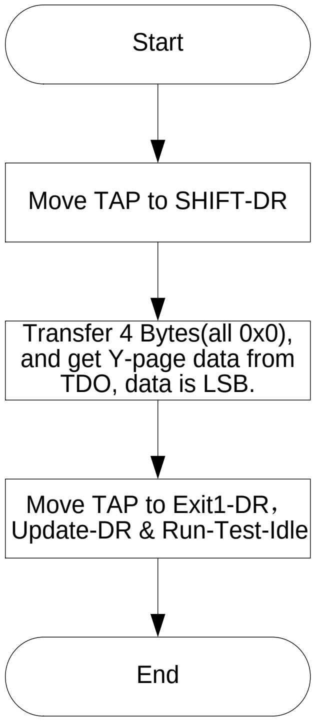


读取一个 Page（Y-page）的过程

与写一个 Y-page 相似，但无写入 Flash 的等待时间，如图 7-23 所示。

数据最先输出的是数据最低位。

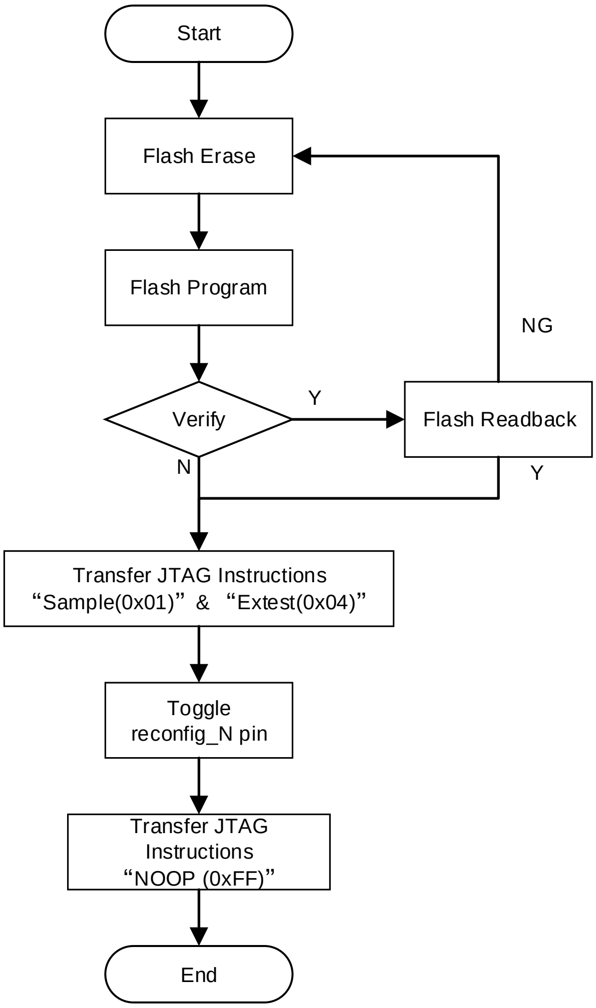
图 7-23 读取一个 Y-page 的过程



背景烧录（Background Programming）

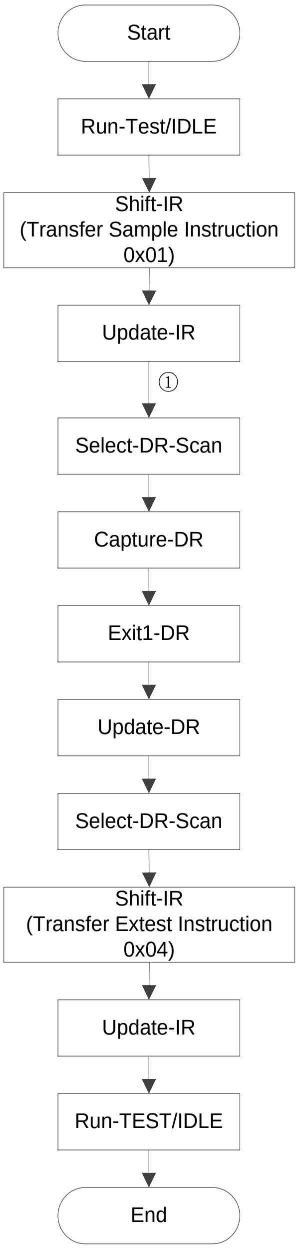
设备有时需要在不影响当前功能的情况下升级数据文件，对 Flash 进行烧录。并且在加载新的数据流文件时，能够保持 IO 状态。下图是 GW1N-4 使用背景烧录技术（Background Programming）升级内置 Flash 数据的流程示意图。

图 7-24 GW1N-4 背景烧录流程图



Transfer JTAG Instructions Sample & Extest 流程图如图 7-25 所示。

图 7-25 Transfer JTAG Instruction Sample & Extest 流程图



注！

①处直接从 Update-IR 跳入 Select-DR-Scan。

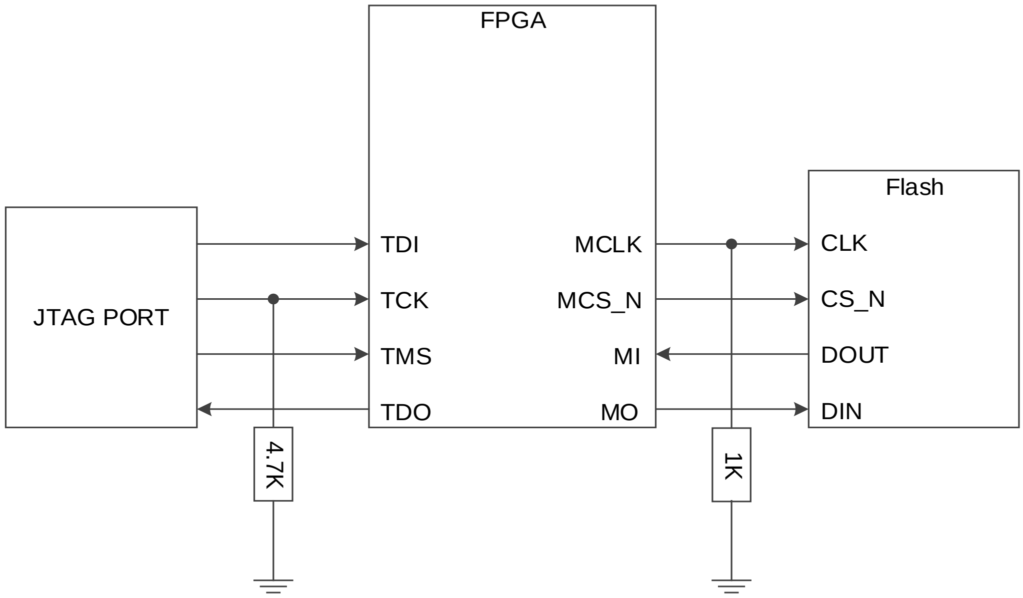
编程外部 Flash 或内嵌 SPI-Flash

高云 FPGA 可从外部 Flash 中加载数据流文件，可以通过 JTAG 直接烧录外部 Flash。

注！

GW2ANR-18/GW2AN-55 内部封了一颗 SPI-Flash，编程方式与 GW2A-18、GW2A-55 相同，GW2ANR-18/GW2AN-55 外部的 MCLK，MCS_N，MI，MO 四个管脚必须悬空。

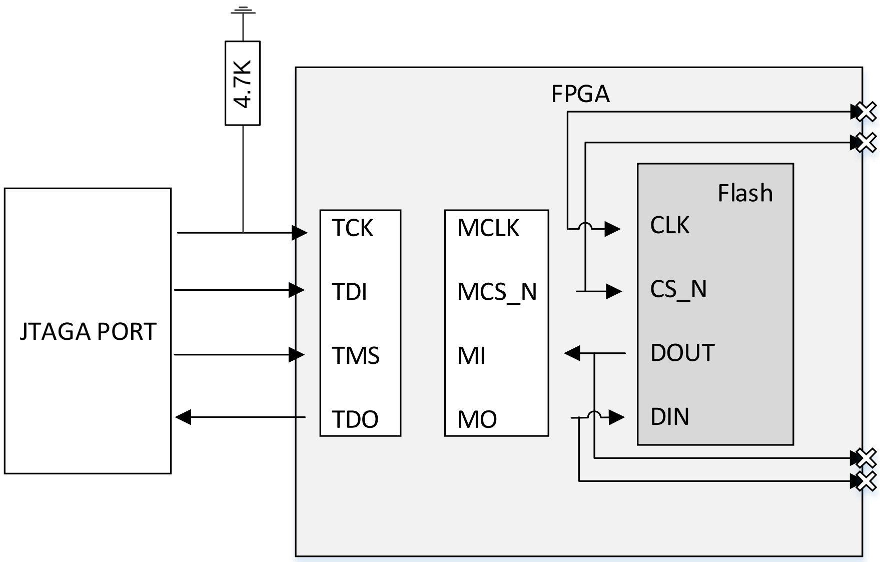
图 7-26 JTAG 接口编程外部 Flash 连接示意图（GW2A(R)-18/GW2A-55/小蜜蜂家族）



注！

- 此图为 JTAG 接口编程外部 Flash 的最小系统图。

图 7-27 JTAG 接口编程内部 SPI-Flash 连接示意图（GW2ANR-18/GW2AN-55）



注！

此图为 JTAG 接口编程内嵌 SPI-Flash 的最小系统图，MSPI 的四个管脚要悬空。

JTAG 转换 SPI 烧录外部 Flash

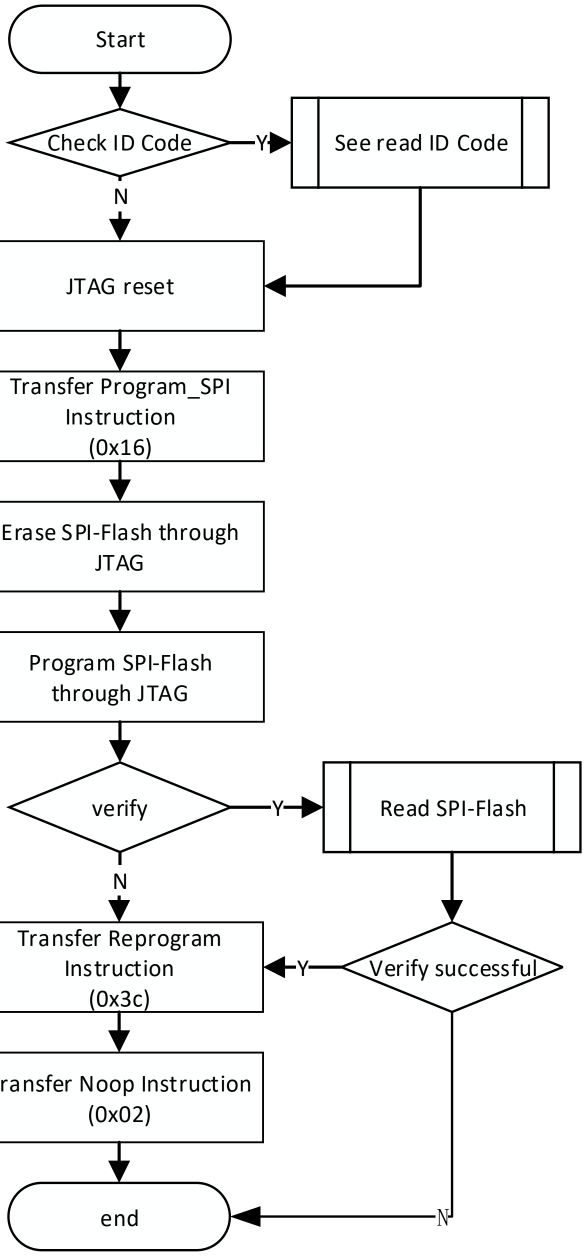
此模式通过 JTAG 接口编程外部 Flash。

此模式的原理是将 JTAG 的接口以转发的形式接入 Flash 的接口，用户通过 JTAG 模拟 Master SPI 时序对 SPI Flash 进行编程。

注！

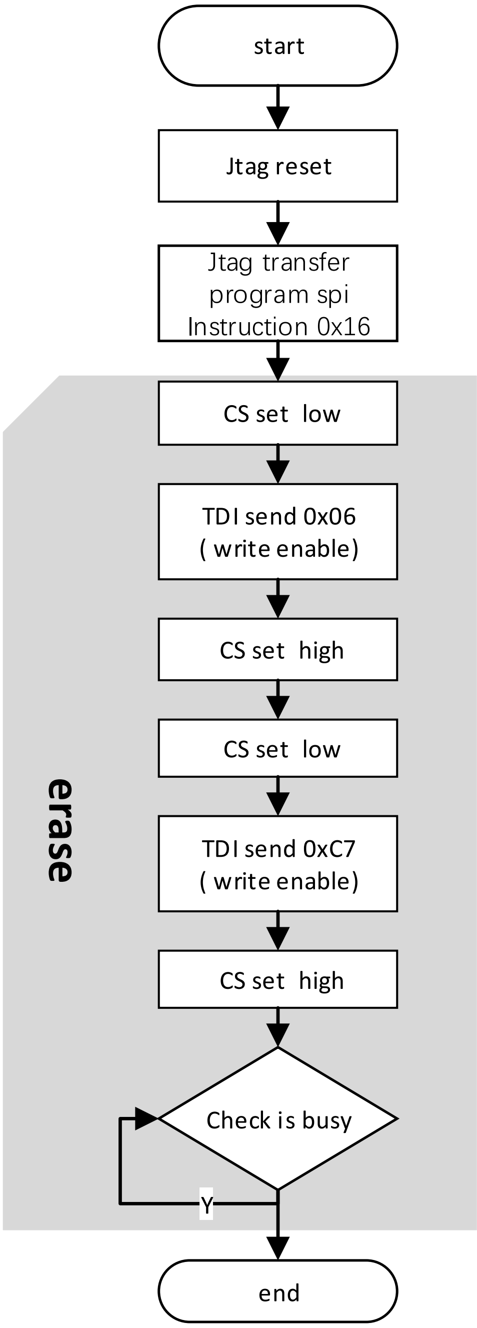
- 发送 0x16 之后，FPGA 将 JTAG 信号转发至 MSPI 管脚，以配置 SPI-Flash； 当 JTAG 复位时该转发功能失效。
- 当从 SPI-Flash 回读数据时，第一个时钟数据为无效数据。如：回读 Flash ID code，发完 0x9F 指令后需要多发一个 clock 再回读 3Byte 数据。
- JTAG 需要在 SHIFT-DR 状态下模拟 SPI 的时序。

图 7-28 编程 SPI Flash 流程示意图



SPI-Flash 擦除流程图如图 7-29 所示：

图 7-29 擦除 SPI Flash 流程示意图



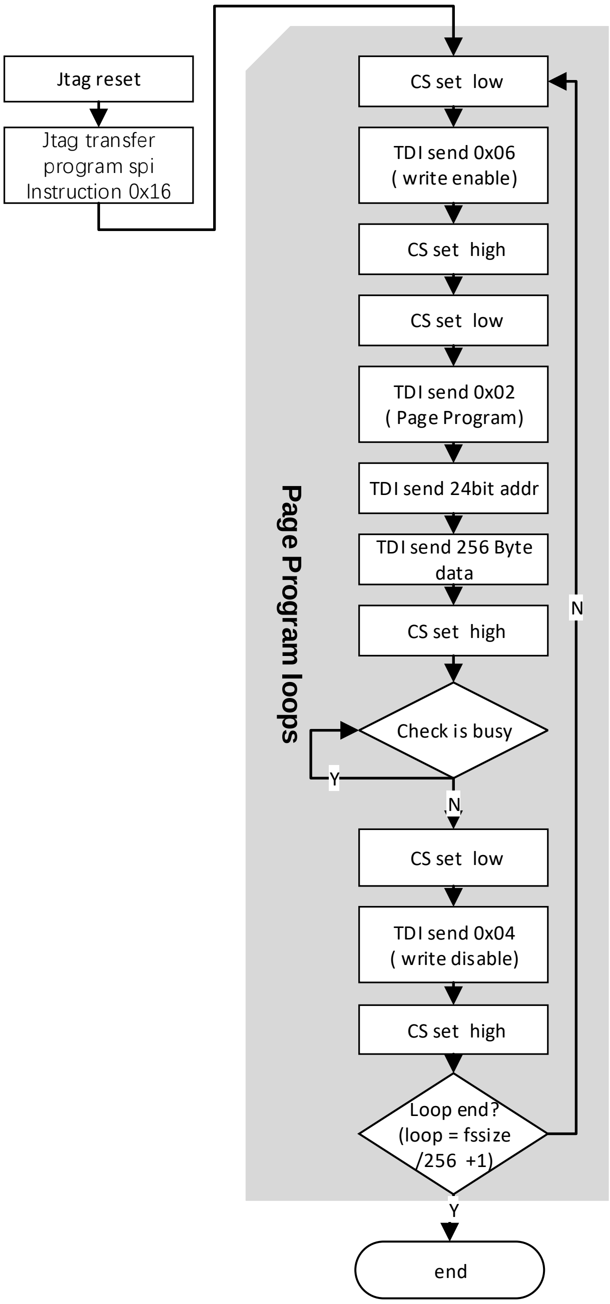
SPI-Flash 擦除流程：

1. JTAG 复位。
2. JTAG 转发 program spi Instruction 0x16 (LSB)。
3. JTAG 的 TCK ,TMS, TDI, TDO 信号分别接到 MCLK ,CS,MOSI,MISO。
4. JTAG 控制 CS 拉低，控制 MOSI 写指令 0x06。
5. JTAG 控制 CS 拉高。
6. JTAG 控制 CS 拉低，控制 MOSI 写指令 0xc7。
7. JTAG 控制 CS 拉高。
8. 检查 SPI 是否 busy。

9. 擦除结束。

SPI-Flash 编程一个 page 流程，编程 SPI-Flash 以 page 为单位，循环编程：

图 7-30 SPI-Flash 编程一个 page 流程



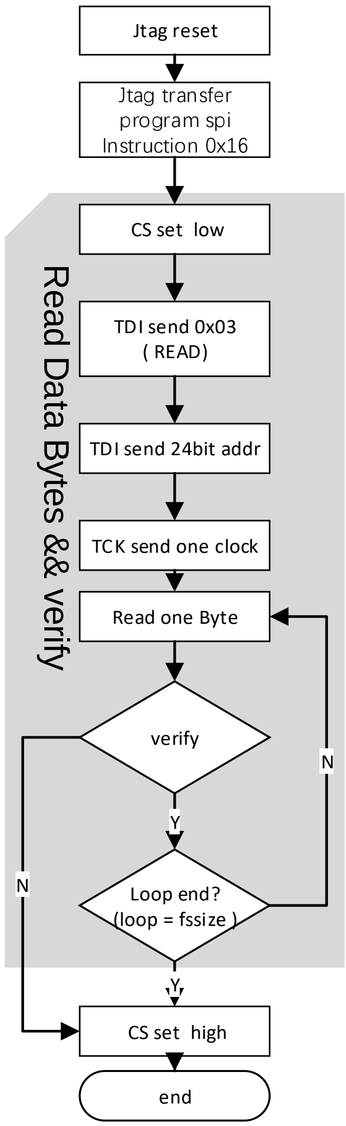
SPI-Flash 编程一个 page 流程：

1. JTAG reset。
2. JTAG 转发 program spi Instruction 0x16 (LSB)。
3. JTAG 的 TCK ,TMS, TDI, TDO 信号分别接到 MCLK ,CS,MOSI,MISO。

4. JTAG 控制 CS 拉低，控制 MOSI 写指令 0x06。
5. JTAG 控制 CS 拉高。
6. JTAG 控制 CS 拉低，控制 MOSI 写指令 0x02 和 3Byte 的地址和 256Byte fs 数据。
7. JTAG 控制 CS 拉高。
8. 检查 SPI 是否 busy。
9. JTAG 控制 CS 拉低，控制 MOSI 写指令 0x04。
10. JTAG 控制 CS 拉高。
11. 写一个 page 结束。

SPI-Flash 回读并校验数据流文件流程图如图 7-31 所示：

图 7-31 SPI-Flash 回读并校验数据流文件流程图



SPI-Flash 回读并校验数据流文件流程：

1. JTAG reset。
2. JTAG 转发 program SPI Instruction 0x16 (LSB)。
3. JTAG 的 TCK, TMS, TDI, TDO 信号分别接到

MCLK ,CS,MOSI,MISO。

4. JTAG 控制 CS 拉低，控制 MOSI 写指令 0x03 和 3Byte 的地址。
5. JTAG 控制 MCLK 发送一个 clock。
6. JTAG 回读数据，一次回读 1Byte。
7. 回读数据与写入的数据流文件进行比对，比对一致则继续比对下一 Byte 直到最后一个 Byte；如果不一致则跳出循环。
8. JTAG 控制 CS 拉高。
9. 回读校验结束。

图 7-32 GW2A 系列 JTAG 模拟 SPI 发送 0x06 指令时序图

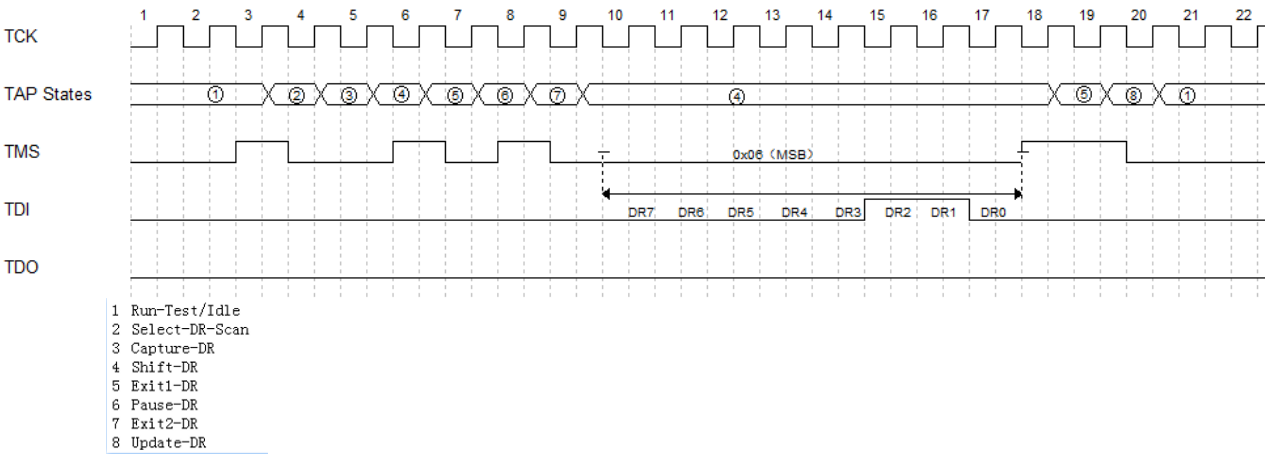
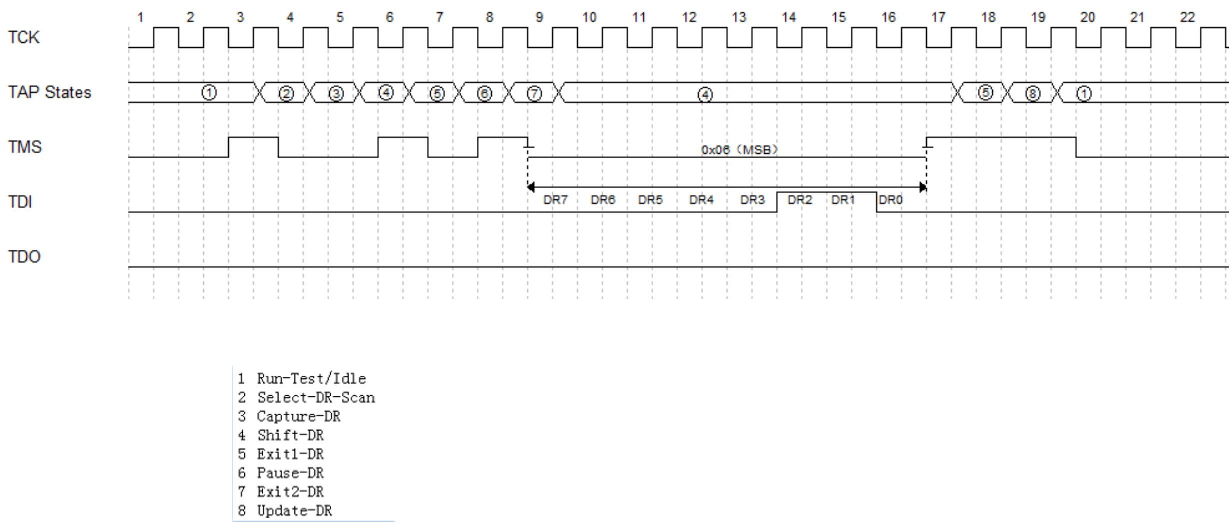


图 7-33 GW1N 系列 JTAG 模拟 SPI 发送 0x06 指令时序图



JTAG Boundary Scan 模式烧录 SPI Flash

该模式的原理，是使用 Boundary Scan 的方式改变与 SPI 相连管脚的状态来实现 SSPI 时序，从而编程内部 Flash。

该模式采用的 Boundary Scan Chain 长度为 8 位，每 2 位组合对应管脚的状态，如表 7-11 所示，每发送两次 Boundary Scan Chain 完成一次 SCLK 驱动。

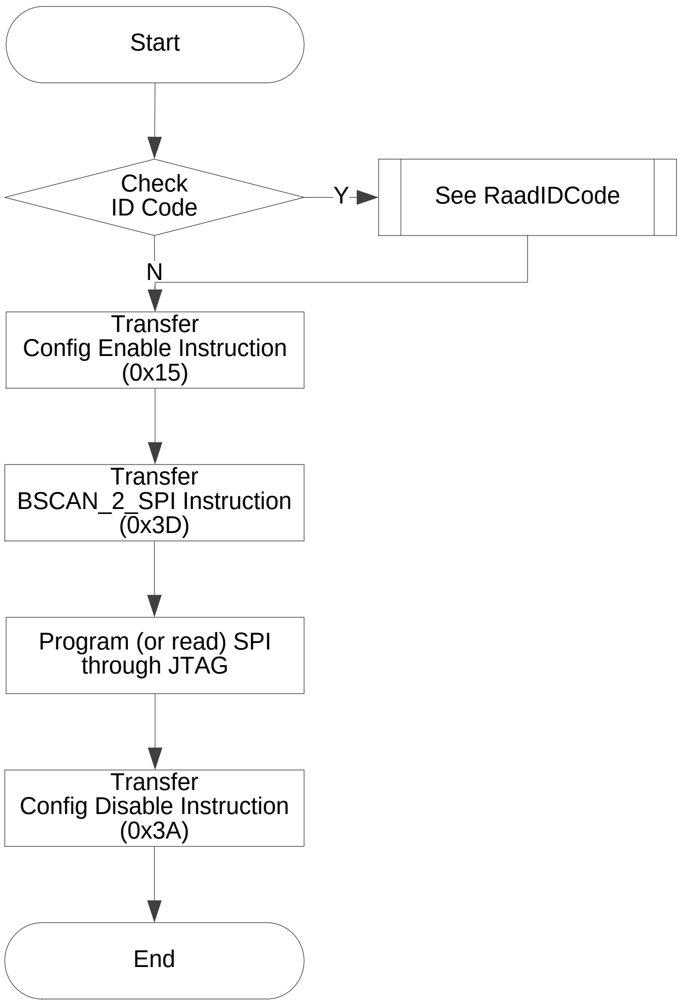
表 7-11 管脚状态

Pins Name of SPI Flash	SCLK		CS		DI		DO	
Bscan Chain[7:0]	7	6	5	4	3	2	1	0
(ctrl & data)	0		0		0		1	

注！

- ctrl:0 表示输出，1 表示输入。
- data:0 表示低电平，1 表示高电平。

图 7-34 采用 Boundary Scan 模式编程 SPI Flash 流程示意图



读取 Status Register（0x41）

Status Register 在器件调试和观察器件状态有很大帮助，通过读取 Status Register，可初步判断器件的状态，如判断是否成功 wakeup、是否存在加载错误等。

Status Register 共有 32 位，读取指令是 0x41，时序与 Read ID Code 一致。

Status Register 的含义如表 7-12~表 7-14 所示。

表 7-12 Status Register 与配置加载相关的条目(一)

Device / Status Register[31:0]	GW1N(R)-(1/4B/4C/4D)/GW1NRF-4B
0	CRC Error Flag(1 表示发生错误，0 表示未发生错误)
1	Bad Command Error Flag (1 表示发生错误，0 表示未发生错误)
2	ID Verify Failed Error Flag (1 表示发生错误，0 表示未发生错误)
3	Timeout Error Flag (1 表示发生错误，0 表示未发生错误)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	0
10	Non-jtag active
11	bypass state
12[1]	Gowin VLD(1 表示正常，0 表示异常)
13	Done Final （通常情况下成功加载后为 1，加载失败为 0）
14	Security Final （设置了安全位为 1，未设置安全位为 0）
15	Ready(1 表示正常，0 表示异常)
16	POR(1 表示正常，0 表示异常)
17-31	0

注！

[1] Gowin VLD 是内置 Flash 相关参数。

表 7-13 Status Register 与配置加载相关的条目(二)

Status Register[31:0] \ Device	GW1N(R)-(1P5/2/6/9/9C)/GW1NS-4(4C)/ GW1NSR-4(4C)/GW1NSE-4C/GW1NSER-4C/GW1NZ-(1/2)
0	CRC Error Flag(1 表示发生错误，0 表示未发生错误)
1	Bad Command Error Flag (1 表示发生错误，0 表示未发生错误)
2	ID Verify Failed Error Flag (1 表示发生错误，0 表示未发生错误)
3	Timeout Error Flag (1 表示发生错误，0 表示未发生错误)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	AutoBoot State
10	Non-jtag active
11	bypass state
12[1]	Gowin VLD(1 表示正常，0 表示异常)
13	Done Final （通常情况下成功加载后为 1，加载失败为 0）
14	Security Final （设置了安全位为 1，未设置安全位为 0）
15	Ready(1 表示正常，0 表示异常)
16	POR(1 表示正常，0 表示异常)
17	Flash Lock
18-31	0

注！

[1] Gowin VLD 是内置 Flash 相关参数。

表 7-14 Status Register 与配置加载相关的条目(三)

Status Register[31:0] \ Device	GW2A(NR)-18/18C/55/55C
0	CRC Error Flag(1 表示发生错误，0 表示未发生错误)

Device / Status Register[31:0]	GW2A(NR)-18/18C/55/55C
1	Bad Command Error Flag (1 表示发生错误，0 表示未发生错误)
2	ID Verify Failed Error Flag (1 表示发生错误，0 表示未发生错误)
3	Timeout Error Flag (1 表示发生错误，0 表示未发生错误)
4	0
5	Memory Erase
6	Preamble
7	Edit Mode
8	program SPI directly
9	0
10	Non-jtag active
11	bypass state
12	0
13	Done Final （通常情况下成功加载后为 1，加载失败为 0）
14	Security Final （设置了安全位为 1，未设置安全位为 0）
15	Encryption Format（1 表示使用了加密的数据流文件）
16	Encryption Key Match（1 表示密钥正确，0 表示密钥错误）
17-31	0

GW1N 家族 FPGA 器件编程

状态寄存器的 Bit-15 READY 仅在编程出错时返回 0x0。编程时出现的错误包括 CRC 错误、错误的命令、ID CODE 不匹配等。

如果状态寄存器 Bit-15 READY 返回 0x0，可通过检查状态寄存器 Bit[3:0]以确定下载错误的原因。

状态寄存器 Bit-13 DONE 必须始终结合 READY（见上文）来确认下载是否成功，无法单独使用。

GW1N 家族 FPGA 器件状态寄存器返回值

0x0001B020(安全位未置 1)表示 FPGA 已配置成功(不建议在生产中采用这种做法，因为这会导致可以从 SRAM 中读取下载数据)。

0x0001F020 (安全位置 1)亦表示 FPGA 已配置成功。

即下载成功情况下的状态寄存器返回值为：

- Bit-16 POR = 0x1
- Bit-15 Ready = 0x1
- Bit-14 Security Final = 0x1 或 0x0 (见上文)
- Bit-13 DONE Final = 0x1
- Bit-12 VLD = 0x1

GW2A 家族 FPGA 器件编程

当对 GW2A 器件进行编程时，以下位只在编程过程中使用，编程完成后，这些位会被自动清零。也就是说，这两个位的最终状态返回值将始终为 0x0。

- Bit-15 Encrypted Format
- Bit-16 Encrypted Key is Right

此外，GOWIN VLD 状态位只适用于有内嵌 Flash 的设备。因此，GW2A 家族器件的 Bit-12 也将返回 0x0。

GW2A 家族 FPGA 器件状态寄存器返回值

0x02020(安全位未置 1)表示 FPGA 已配置成功(不建议在生产中采用这种做法，因为这会导致可以从 SRAM 中读取下载数据)。

0x06020(安全位置 1)亦表示 FPGA 已配置成功。

即下载成功情况下的状态寄存器返回值为：

- Bit-16 = 0x0
- Bit-15 = 0x0
- Bit-14 Security Final = 0x1 或 0x0 (见上文)
- Bit-13 DONE Final = 0x1
- Bit-12 = 0x0

有关状态寄存器的更多信息，请参考 *TN711，GOWIN FPGA 产品状态寄存器说明*。

清除 status code 错误

在配置 SRAM 或烧录 Flash 前，要确保当前 status code 中没有错误，如果有错误，请按以下流程清除错误：

对于 GW1N(X)系列：

1. 发送指令 Reprogram(0x3C)。
2. 进行 JTAG Reset 操作。
3. 依次发送指令 ISC-ENABLE(0x15)，ISC-DISABLE(0x3A)，ISC-NOOP(0x02)。
4. 延时 100ms。
5. 依次发送指令 ISC-NOOP(0x02)，ISC-ENABLE(0x15)，ISC-DISABLE(0x3A)，ISC-NOOP(0x02)。

对于 GW2A(X)系列：

1. 依次发送指令 ISC-NOOP(0x02)，ISC-ENABLE(0x15), reprogram(0x3C)，ISC-NOOP(0x02)。
2. 延时 100ms。
3. 依次发送指令 ISC-DISABLE(0x3A)，ISC-NOOP(0x02)。

4. 延时 100ms。

读取 User Code（0x13）

User Code 共有 32 位，读取指令是 0x13，时序与 Read ID Code 一致。

User Code 默认使用的是 FS 文件的 checksum 值，可在 Gowin Designer 中重新定义。

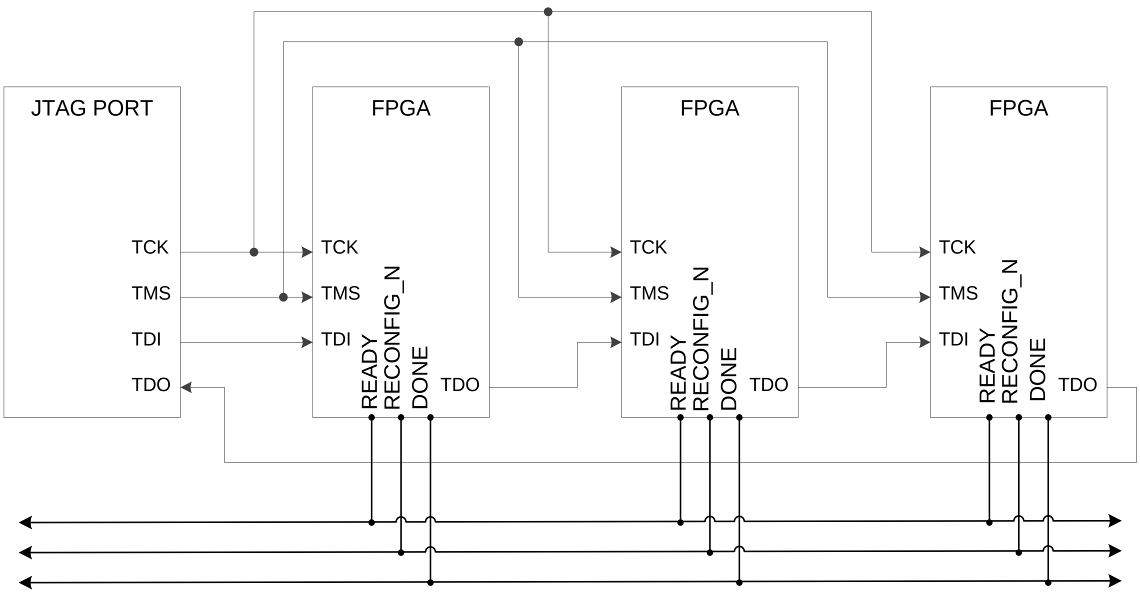
重加载（0x3C）

该指令作用是使 FPGA 从 flash 中读取数据流文件，并配置到 SRAM。

通过 JTAG 依次发送 Reprogram（0x3C）指令、Noop（0x02）指令，可使器件重加载，效果同触发 Reconfig_N 管脚 。

菊花链连接示意图

图 7-35 菊花链连接示意图



例程文件

例程文件，请联系公司技术支持或当地办事处。

7.3 AUTO BOOT 配置模式（仅小蜜蜂(LittleBee)家族支持）

AUTO BOOT 配置模式，是高云半导体针对小蜜蜂(LittleBee)家族非易失 FPGA 产品的瞬时接通特性推出的一种配置模式，晨熙(Arora)家族 FPGA 产品不支持自启动配置模式。自启动模式下，芯片上电后无需连接外部配置接口，FPGA 即可自行从内置 Flash 读取比特流数据完成配置。

用户使用自启动模式时，首先需要通过 JTAG 接口将配置数据编程到 FPGA 的内置 Flash 中（参考图 7-4 JTAG 配置模式连接示意图），然后调

节 MODE 值为“000”，当系统重新上电或低电平脉冲触发 RECONFIG_N 管脚时，芯片会自动读取比特流数据完成配置过程。用户将 MODE 值设置为“000”，使用 Gowin 编程软件编程内置 Flash 结束后 FPGA 会自动配置 SRAM 完成自启动。内置 Flash 的瞬时接通特性为配置过程节约了下载时间，提高了工作效率。

相比于小蜜蜂(LittleBee)家族的其他 FPGA 产品上电后只能支持一次自启动配置操作，GW1N(R)-9 和 GW1NS 系列 FPGA 支持重试两次自启动配置，即上电后自启动配置失败时，器件可以自动进行两次重新配置操作。导致配置失败的因素包括 ID 验证错误，CRC 校验错误和指令错误。

注！

内置 Flash 中只能保存一份比特流数据，配置重试地址无法修改。

7.4 SSPI 配置模式

SSPI（Slave SPI）配置模式，即 FPGA 作为从器件，由外部 Host 通过 SPI 接口对高云半导体 FPGA 产品进行配置的过程。

7.4.1 SSPI 配置模式管脚

SSPI 模式相关的配置管脚如表 7-15 所示。

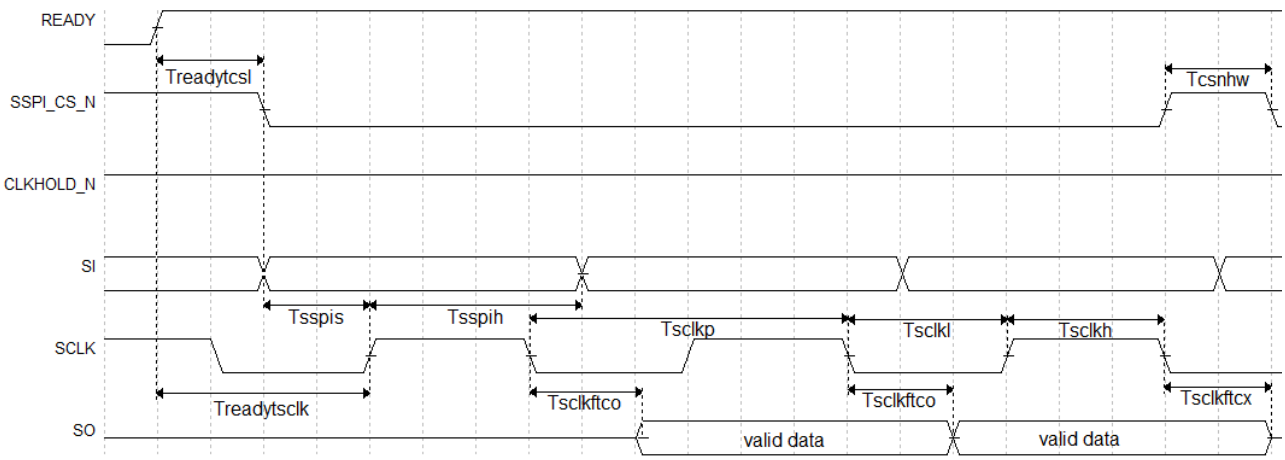
表 7-15 SSPI 配置模式管脚

管脚名称	I/O 类型	说明
RECONFIG_N	I, 内部弱上拉	低电平脉冲：启动 GowinCONFIG 配置
READY	I/O	高电平：当前可以对 FPGA 进行编程配置 低电平：禁止对 FPGA 进行编程配置
DONE	I/O	高电平：成功完成编程配置 低电平：未完成编程配置或编程配置失败
MODE[2:0]	I, 内部弱上拉	配置模式选择，READY 上升沿采样
SCLK	I	输入时钟
CLKHOLD_N	I, 内部弱上拉	高电平有效
SO	O	FPGA 输出数据到 Host
SI	I	Host 向 FPGA 输入数据
SSPI_CS_N	I, 内部弱上拉	SSPI 片选信号，低电平有效

7.4.2 SSPI 配置模式时序图

SSPI 配置模式的时序图如图 7-36 所示。

图 7-36 SSPI 配置模式时序图



时序参数如表 7-16 所示。

表 7-16 SSPI 配置模式时序参数

参数名称	参数含义	最小值	最大值
T_{sclkp}	SCLK 时钟周期(SCLK clock period)	15ns	-
T_{sclkh}	SCLK 时钟高电平时间(SCLK clock high time)	7.5ns	-
T_{sclkl}	SCLK 时钟低电平时间(SCLK clock low time)	7.5ns	-
T_{sspis}	SSPI PORT 建立时间(SSPI PORT setup time)	2ns	-
T_{sspih}	SSPI PORT 保持时间(SSPI PORT hold time)	0ns	-
$T_{sclkftco}$	SCLK 下降沿到数据输出时延(Time from SCLK falling edge to output)	-	10ns
$T_{sclkftcx}$	SCLK 下降沿到输出高阻时延(Time from SCLK falling edge to high impedance)	-	10ns
T_{csnhw}	CSN 高电平脉冲宽度（CSN high time）	25ns	-
$T_{readytcsl}$	READY 上升沿到 CSN 低电平时间(Time from READY rising edge to CSN low)	10μs	
$T_{readytsclk}$	READY 上升沿到第一个 SCLK 沿时间(Time from READY rising edge to first SCLK edge)	10μs	-

除满足上电要求外，SSPI 模式对高云半导体 FPGA 产品进行配置，还需满足以下条件：

- SSPI 接口使能
  上电后初次配置或前一次配置时 RECONFIG_N 未设置为普通 I/O 状态。
- 启动新的配置
  重新上电或低电平脉冲触发 RECONFIG_N 管脚。

7.4.3 SSPI 常用配置指令

当 FPGA 处于 SSPI 模式时，通过 SSPI 可以烧录 FPGA SRAM 或者读取 ID CODE\USER CODE\STATUS CODE 等信息，也可以烧录外部存储设备（例如 SPI Flash）。

FPGA 的 SSPI 指令一般由 1-4 个字节组成，至少包含 1 个指令类字节和多个冗余信息字节，没有指定信息字节的情况下，冗余信息字节可以是任意数（下表用 0x00 表示）。

表 7-17 配置指令

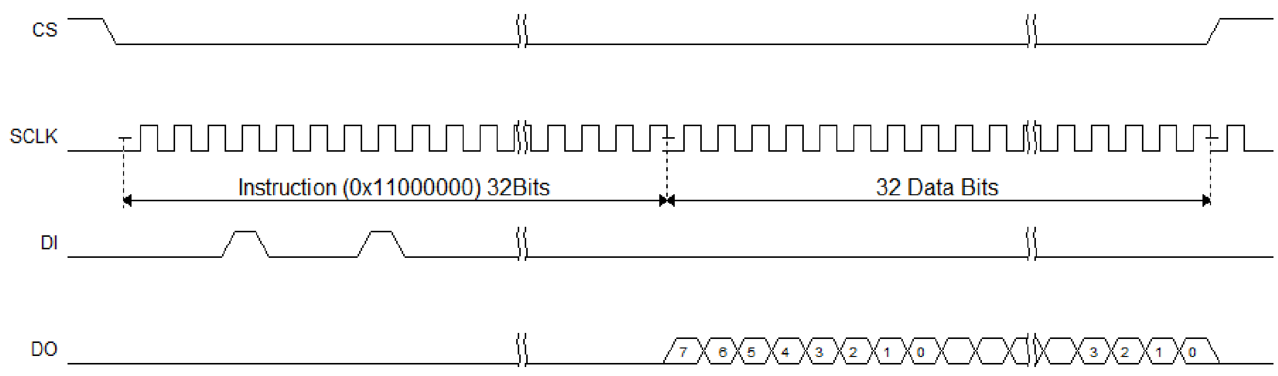
指令名称	完整指令（指令字节+冗余信息字节）
Read ID Code	0x11000000
Read User Code	0x13000000
Read Status Code	0x41000000
Reconfig/Reprogram	0x3C00
Write Enable	0x1500
Write Disable	0x3A00
Write Data	0x3B
Program SPI Flash	0x1600
Init Address	0x1200
Erase SRAM	0x0500

Read ID Code

FPGA 的 ID Code 长度是 32bits，读取 ID 的指令是四个字节，即 0x11000000。在指令发送之前，使 CS 处于高电平状态，并且要在此状态产生若干时钟（两个时钟以上），以驱动 FPGA 获取 CS 状态。

当 CS 拉低之后，数据以 MSB 的方式写入指令 0x11000000，在写完四字节指令后要继续产生 32 个时钟，此时 ID CODE 数据将以 MSB 的形式从 DO 依次位移出来。

图 7-37 读取 ID Code 时序示意图

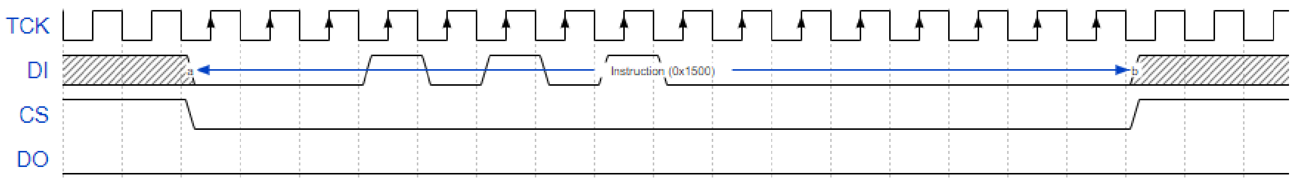


读取 StatusCode/UserCode 与读取 ID Code 的操作类似，替换对应的指令即可。

Write Enable (0x1500)

在配置 SRAM（写 Features）前，使用 Write Enable（0x15）指令进入设备编辑模式，使设备可以接受写数据 Write Data（0x3B）指令。

图 7-38 Write Enable（0x15）时序示意图



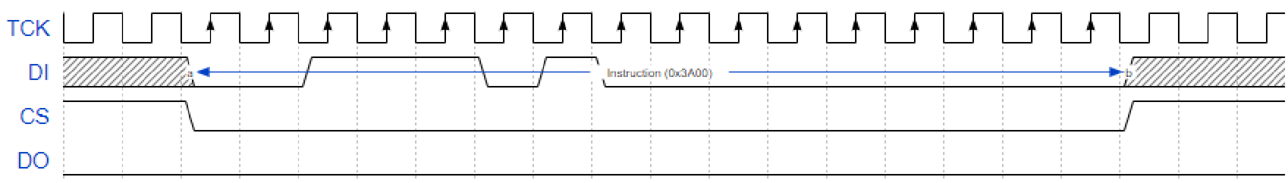
注！

SCLK 驱动规则，在 CS 高电平时，需要给予 SCLK 两个以上时钟，以驱动 FPGA 识别 CS 信号。发送其他指令也遵循这个规则。

Write Disable (0x3A00)

发送数据完成后，需使用 Write Disable 从编辑模式中退出。退出后可唤醒设备，使设备进入工作状态。

图 7-39 Write Disable (0x3A00) 时序示意图



像上述两个指令，0x1500 和 0x3A00 指令的时序基本一致，指令都是在 CS 低电平后开始，并在指令传输完成后拉高 CS，遵循这种时序的指令有 0x3C00（Reconfig/Reprogram）、0x1500（Write Enable）、0x3A000（Write Disable）、0x1600（Program SPI Flash）、0x1200（Init Address）、0x0500（Erase SRAM）。

另外，需要注意的是，因为 SSPI 是由外部时钟驱动，这些指令前后 CS 处于高电平时，需要 2 个以上的时钟以使 FPGA 可以采集到 CS 状态。

Erase SRAM(0x0500)

指令时序与 WriteEnable/WriteDisable 一致，仅替换指令内容为 0x0500。

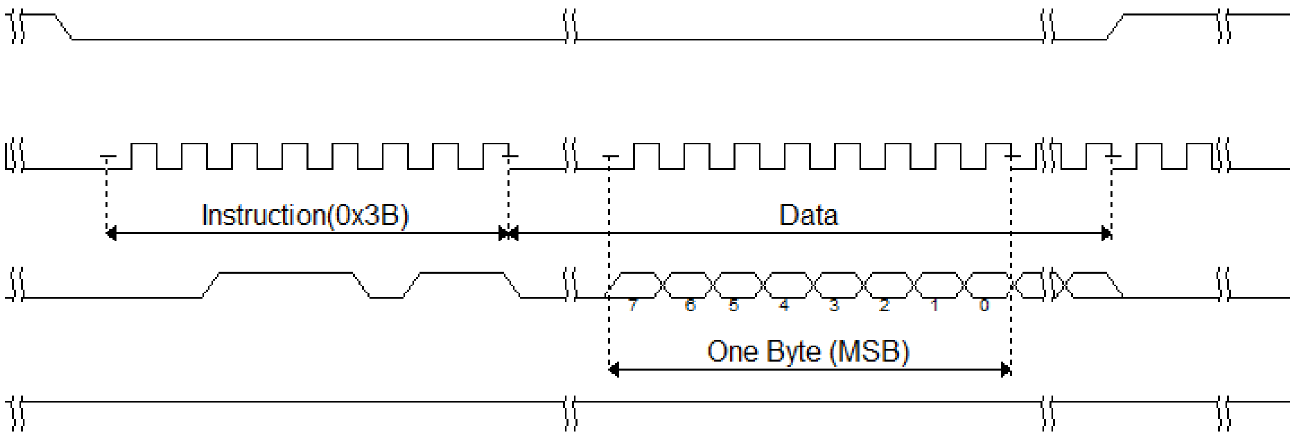
当指令发送后，需延迟至少 10ms 以使指令执行完毕。

Write Data (0x3B)

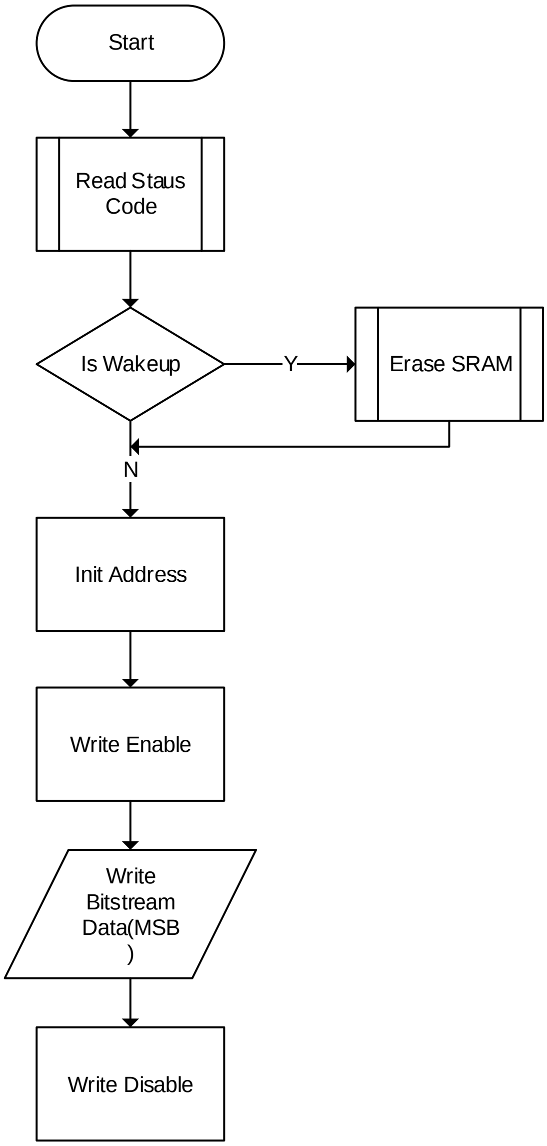
通过使用 WriteData（0x3B）指令直接向 FPGA 设备发送数据流文件。

注意，在数据写入过程中，CS 一直处于低电平。

图 7-40 Write Data（0x3B）时序示意图



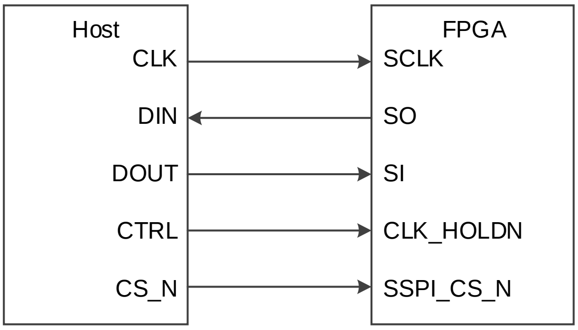
7.4.4 SSPI Configure SRAM 流程图



7.4.5 SSPI 配置模式连接示意图

使用 SSPI 配置模式对高云半导体 FPGA 产品配置的连接示意图如图 7-41 所示。

图 7-41 SSPI 配置模式连接示意图



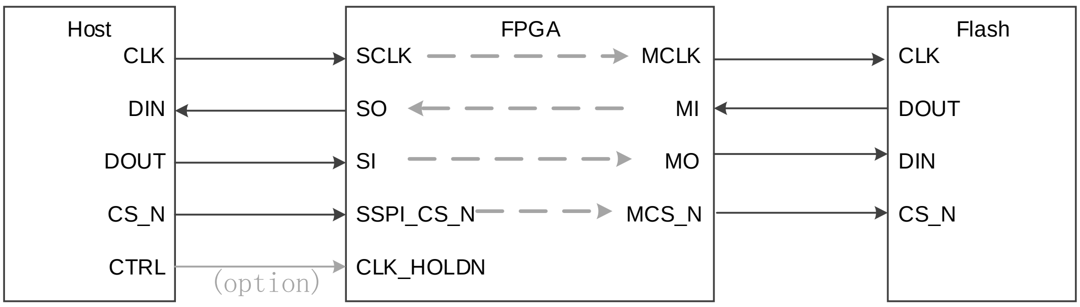
注！

此图为 SSPI 配置模式的最小系统图，SSPI 模式 MODE 值为“001”，其他固定管脚的接法请参考图 7-1。

SSPI 配置管脚除了进行常规的 SRAM 配置操作外，还可编程 FPGA SPI Flash，编程 Flash 操作的 MODE 值与 SSPI 配置模式的 MODE 值相同，用户可以在 Gowin 编程软件中选择将配置数据写入 SRAM 或 Flash。需要从 Flash 加载前，需要将 MODE 值调整为 MSPI MODE，之后通过重新上电或触发 RECONFIG_N 触发 MSPI 加载。

SSPI 接口编程外部 Flash 和内部 SPI-Flash 的连接示意图如图 7-42 和图 7-43 所示。

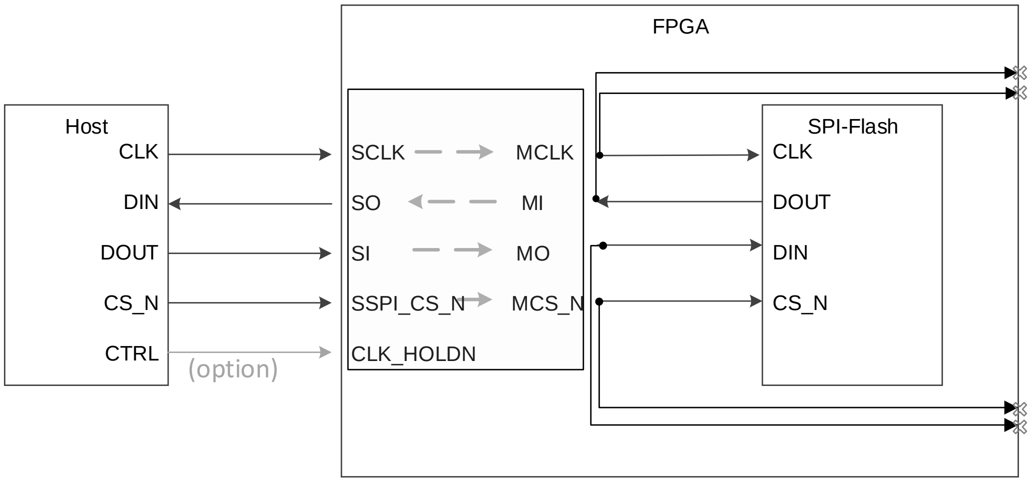
图 7-42 SSPI 编程外部 Flash 连接示意图(GW2A-18/55,GW1N(R)-9)



注！

- 晨熙系列全系支持 SSPI 编程外部 Flash，
- 小蜜蜂系列仅 GW1N(R)-9 器件支持 SSPI 编程外部 Flash。

图 7-43 SSPI 编程内部 Flash 连接示意图(GW2ANR-18/GW2AN-55)



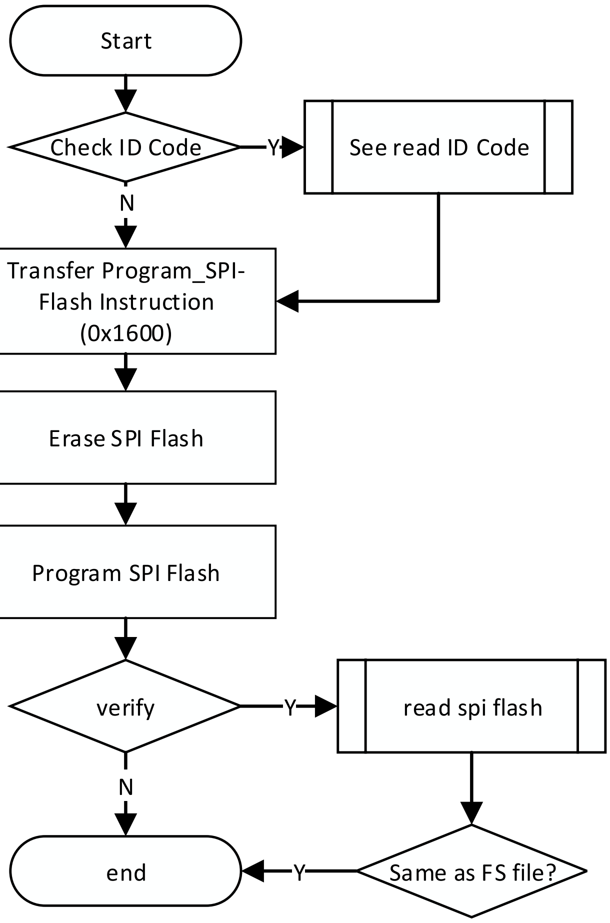
注！

GW2ANR-18/GW2AN-55 内部合封了一颗 SPI-Flash，编程方式与 GW2A-18、GW2A-55 相同，GW2ANR-18/GW2AN-55 外部的 MCLK，MCS_N，MI，MO 四个管脚必须悬空。

烧录流程示意图见图 7-44。首先通过 SSPI 向 FPGA 发送指令“Program SPI Flash”（0x1600）指令，完成后 FPGA 可以转发 SSPI 到 Flash，Host 端的 SSPI 可以直接访问 Flash，接下来就可以按照 Flash 的相关时序对其进行编程。

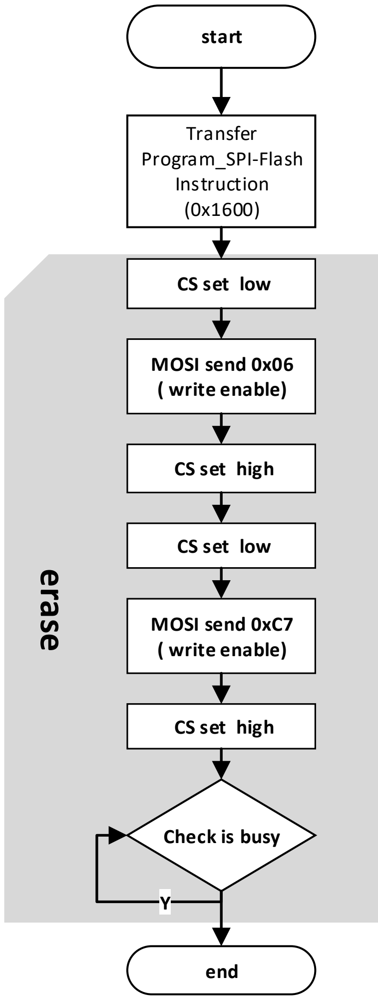
注意，从 Flash 读取数据时，回读的数据会延迟一个 Bit。例如 sspi 读取 Flash 的 IDCode 时，需要额外发送一个时钟从而获取最后一位。

图 7-44 SSPI 编程 Flash 流程图



SPI-Flash 擦除流程图如图 7-45 所示：

图 7-45 擦除 SPI Flash 流程图

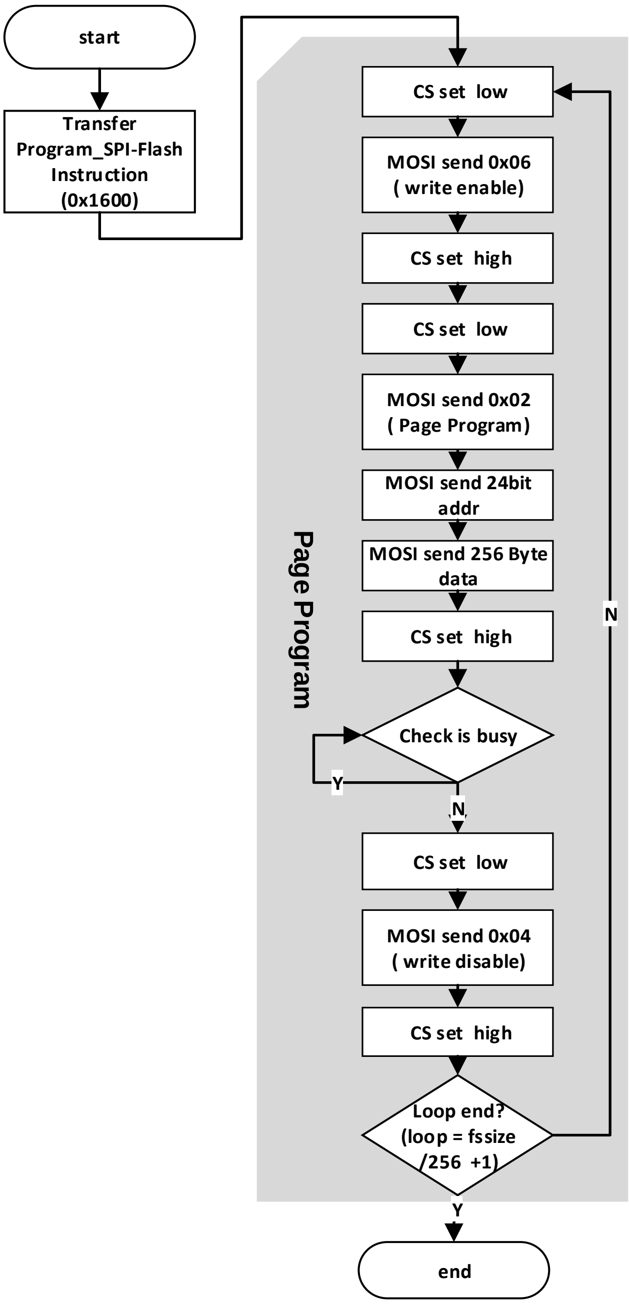


SPI-Flash 擦除流程：

1. start。
2. Host MSPI 转发 program spi Instruction 0x1600 (MSB)。
3. Device-SSPI 的 SCLK ,CS, SI, SO 信号在 fpga 的内部分别转发到 MCLK ,CS,MOSI,MISO。
4. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0x06。
5. Host MSPI 控制 CS 拉高。
6. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0xc7。
7. Host MSPI 控制 CS 拉高。
8. 检查 SPI 是否 busy。
9. 擦除结束。

SPI-Flash 编程一个 page 流程如图 7-46 所示，编程 SPI-Flash 以 page 为单位，循环编程：

图 7-46 SPI-Flash 编程一个 page 流程图



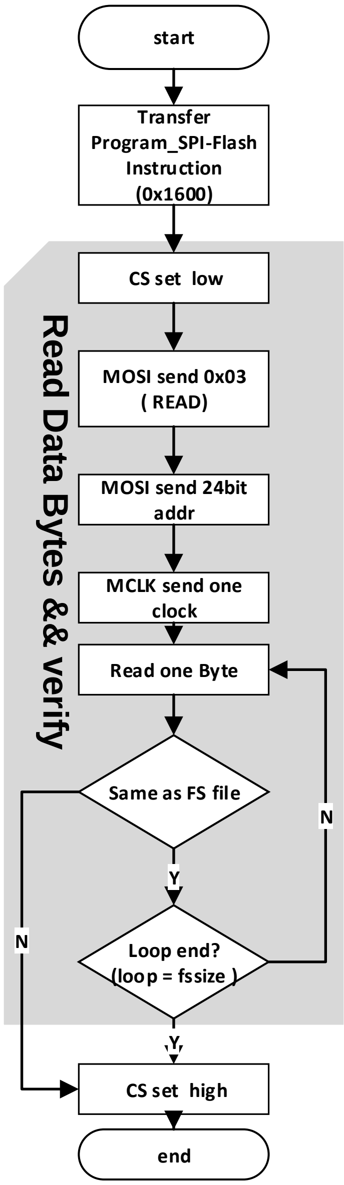
SPI-Flash 编程一个 page 流程：

1. start。
2. Host MSPI 转发 program spi Instruction 0x1600 (MSB)。
3. Device-SSPI 的 SCLK ,CS, SI, SO 信号在 fpga 的内部分别转发到 MCLK ,CS,MOSI,MISO。
4. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0x06。
5. Host MSPI 控制 CS 拉高。
6. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0x02 和 3Byte 的地址和 256Byte fs 数据。
7. Host MSPI 控制 CS 拉高。

8. 检查 SPI 是否 busy。
9. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0x04。
10. Host MSPI 控制 CS 拉高。
11. 写一个 page 结束。

SPI-Flash 回读并校验数据流文件流程图如图 7-47 所示：

图 7-47 SPI-Flash 回读并校验数据流文件流程图



SPI-Flash 回读并校验数据流文件流程：

1. Start。
2. Host MSPI 转发 program spi Instruction 0x1600 (MSB)。
3. Device-SSPI 的 SCLK ,CS, SI, SO 信号在 fpga 的内部分别转发到 MCLK ,CS,MOSI,MISO。
4. Host MSPI 控制 CS 拉低，控制 MOSI 写指令 0x03 和 3Byte 的地址。
5. Host MSPI 控制 MCLK 发送一个 clock。
6. Host MSPI 回读数据，一次回读 1Byte。

7. 回读数据与写入的数据流文件进行比对，比对一致则继续比对下一 Byte 直到最后一个 Byte；如果不一致则跳出循环。
8. Host MSPI 控制 CS 拉高。
9. 回读校验结束。

7.4.6 SSPI 模式下的多 FPGA 连线示意图

图 7-48 多 FPGA 连线示意图 1

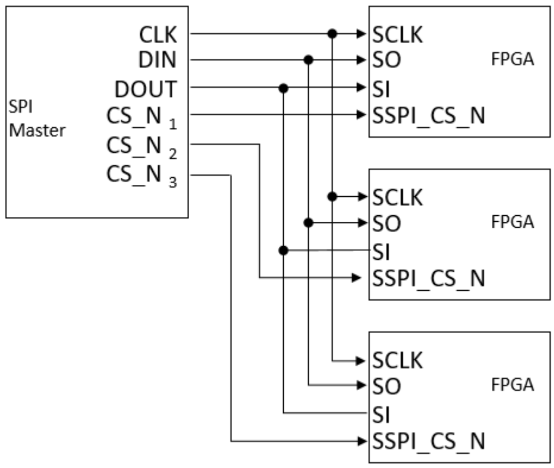
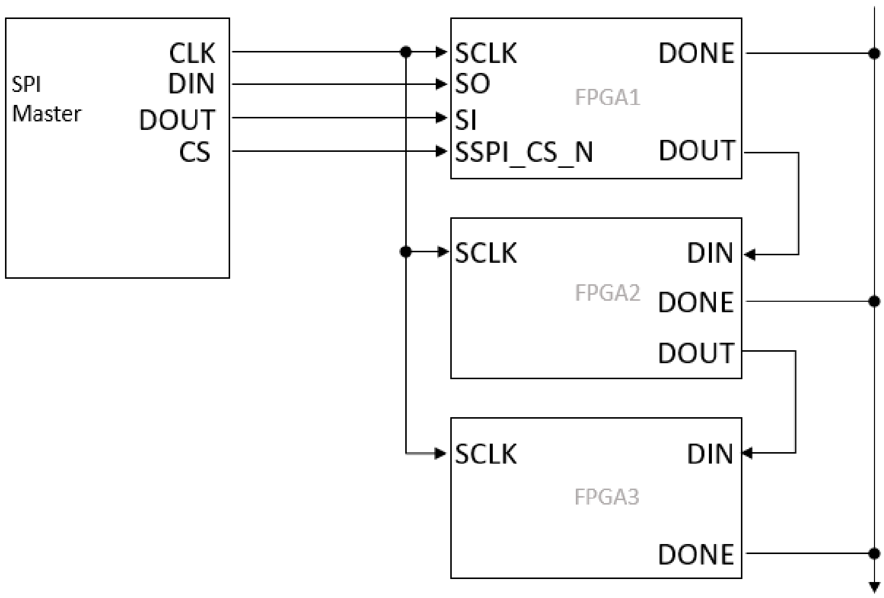


图 7-49 多 FPGA 连线示意图 2



7.5 MSPI 配置模式

在 MSPI（Master SPI）模式下，FPGA 作为主器件（Master），通过其 SPI 接口从外部 Flash 存储器中读取比特流数据来配置 FPGA 的内部 SRAM。

MSPI 模式的配置过程：

1. 将 MODE 管脚的值设置为 MSPI 状态。

2. 通过以下方式使 FPGA 自行从外部 Flash 读取比特流数据，完成配置过程。
   - 将 FPGA 重新上电。
   - 或低电平脉冲触发 RECONFIG_N 管脚。

MSPI 模式下的外部 Flash 升级:

可以使用 JTAG 接口对外部 Flash 重新编程。此特性使 FPGA 能够支持比特流的后台更新，通常被称为内场或远程升级。FPGA 配置完成后，用户可以通过 FPGA 将新的配置数据远程写入外部 Flash。

7.5.1 MSPI 配置模式管脚

MSPI 模式相关的配置管脚如表 7-18 所示。

表 7-18 MSPI 配置模式管脚定义

管脚名称	I/O 类型	说明
RECONFIG_N	I，内部弱上拉	低电平脉冲：启动 GowinCONFIG 配置
READY	I/O	1'b1: 当前可以对器件进行编程配置 1'b0: 禁止对器件进行编程配置
DONE	I/O	在非 JTAG 配置模式下， 1'b1: 成功完成编程配置 1'b0: 未完成编程配置
MODE[2:0]	I，内部弱上拉	MODE 选择信号(在 READY 上升沿采样)
MCLK	O	FPGA 输出时钟
MCS_N	O	SPI 片选信号，低有效
MO	O	经 SPI 输出数据到 Slave
MI	I	从 Slave 经 SPI 输入数据
FASTRD_N	I	在 READY 信号的上升沿采样 1'b1: Read SPI 模式（SPI 指令：0x03） 1'b0: Fast Read SPI 模式（SPI 指令：0x0B）

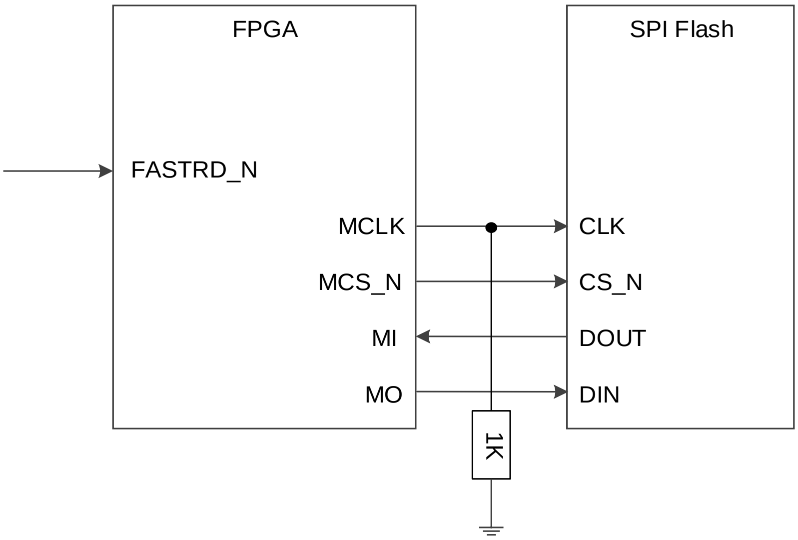
注！

- MSPI 配置模式的时钟频率存在±10%(晨熙家族)或±5%(小蜜蜂家族)的误差。
- MSPI 配置模式的时钟频率上限不应大于 66.6MHz。
- 当时钟频率大于 30MHz 小于 66.6MHz 时需要使用 Flash 的高速访问模式并外部拉低 FASTRD_N 管脚。拉低 FASTRD_N 后，时钟频率需大于 5MHz。
- 时钟频率不高于 30MHz 时，FASTRD_N 管脚悬空即可。

7.5.2 MSPI 配置模式连接示意图

MSPI配置模式下的外部Flash连接如图7-50所示。

图 7-50 MSPI 配置模式连接示意图

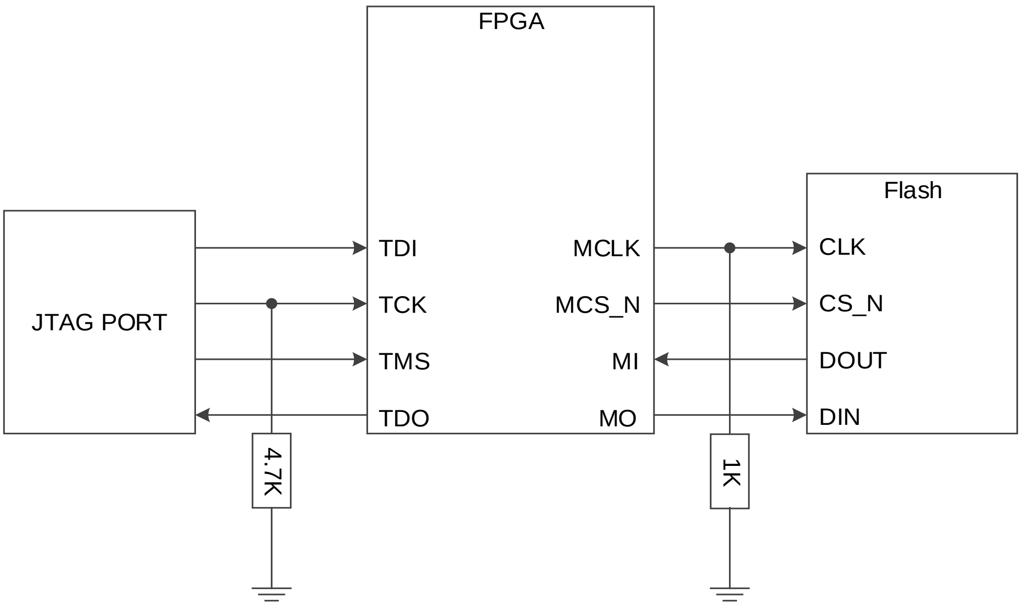


注！

上图为 MSPI 模式的最小系统图。MSPI 模式 MODE 值 GW1N(R)为“010”，GW2A(R)为“000”。其他固定管脚的接法请参考图 7-1。MSPI 配置模式时钟频率不高于 30MHz 时 FASTRD_N 管脚可悬空。

通过 JTAG 接口把配置数据编程到外部 Flash 器件中的连接示意图如图 7-51 所示。通过 SSPI 接口编程外部 Flash 的连接示意图请参考图 7-42。

图 7-51 JTAG 接口编程外部 Flash 的连接示意图



注！

- 上图为 JTAG 接口编程外部 Flash 的最小系统图，其他固定管脚的接法请参考图 7-1。

7.5.3 MSPI 模式配置尝试

高云半导体 FPGA 产品通常上电之后只支持一次自动的 MSPI 配置尝试。

GW1N(R)-9、GW2A(R)-18 和 GW1NS 系列产品在此方面进行了增强处理：上电后 MSPI 配置失败时，上述器件可以按照支持的重试次数自动进行重新配置尝试。

- GW2A(R)-18 系列 FPGA 共支持两次配置尝试。
- GW1N(R)-9 或 GW1NS 系列 FPGA 共支持三次配置尝试。
- 导致配置失败的因素包括 ID 验证错误，CRC 校验错误和指令错误。

当比特流配置失败时，用户可以为下一次重试指定一个其他的 SPI Flash 启动地址。此特性可降低配置失败的风险，并且还可用于在配置失败时加载 Golden (fallback) Image（黄金（备用）映像）。

注！

如果 ID Code 错误，或者比特流的头指令发生错误，则不会去指定的 SPI Flash 地址启动。

在运行 Place & Route 时，可使用 GOWIN EDA 工具的 Bitstream 选项来指定其他 SPI Flash 启动地址（有关更多详细信息，请参阅 7.5.4 多重配置）。

7.5.4 多重配置

多重配置（MULTI BOOT）是指 FPGA 从同一片外部 Flash 的不同地址读取比特流数据进行配置的过程。支持 MSPI 配置模式的 FPGA 产品均支持此模式。

FPGA 上电后的默认 Flash 启动地址为 0x0000，该地址始终用于加载初始比特流。

Gowin Programmer 软件支持在不擦除的情况下将多个比特流数据编程到外部 Flash 中。

使用 GOWIN EDA 工具生成比特流时，用户可以指定下一个要加载的比特流的 SPI Flash 启动地址。即当前的比特流头部包含指向 Flash 中下一个比特流位置的跳转地址。

上电后，FPGA 将自动尝试从 Flash 地址 0x0000 启动。

如果第一次启动尝试失败并且 FPGA 器件支持多次配置尝试，那么下一次启动尝试将使用由当前比特流头部中的 SPI Flash 跳转地址所指定的比特流映像。如果下次启动尝试也失败，则重复此过程，直到达到 FPGA 支持的配置尝试次数上限。

FPGA 上电后，也可以通过拉低 RECONFIG_N 输入信号来跳转到下一个比特流，其中 SPI Flash 跳转地址同样保存在当前比特流头部。请注意，RECONFIG_N 的触发没有次数限制。

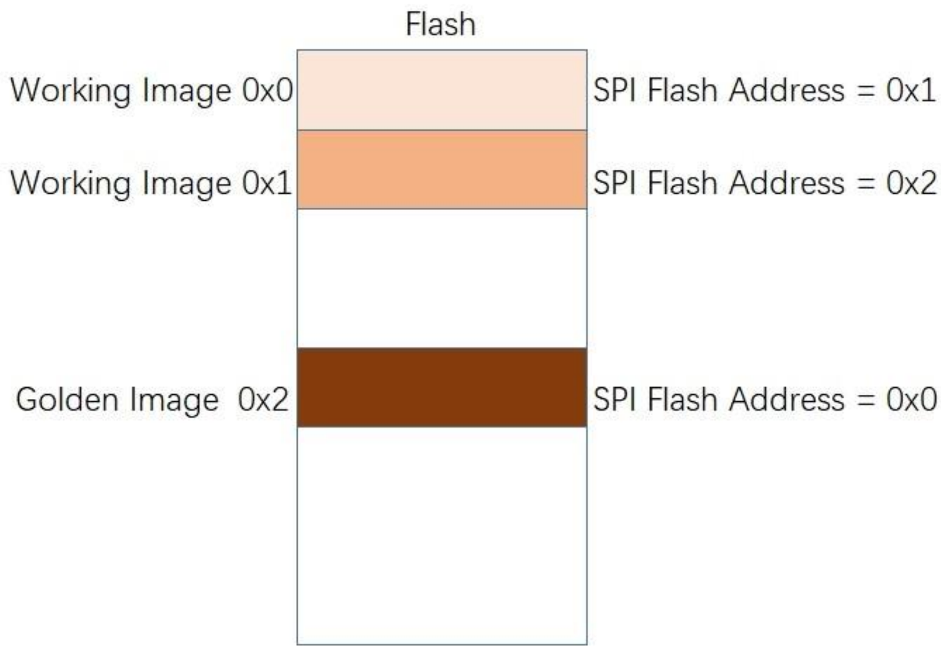
使用可靠 Golden Image 进行多重配置

为了支持远程“内场”比特流更新，可在多重配置时使用可靠的 Golden Image。我们建议将此 Golden (fallback) Image 始终作为最后一个比特流存储在外部 Flash 中。在下面的示例中，如果工作映像 0x0 或 0x1 损坏，我们可以将 RECONFIG_N 拉低以加载下一个比特流，并使用当前映像头部中的 SPI Flash 地址作为跳转地址。这个跳转地址可以是 Flash 中下一个工作映像的启动地址，也可以是 Flash 中 Golden（fallback）Image 的启动地址。

此外，如果所有工作映像都被擦除，那么 FPGA 将继续读取 Flash 地址，直到获取 Golden Image。

如果所有 Flash 映像都损坏，则需要通过 JTAG/SSPI 接口重新编程 SPI Flash。

图 7-52 Flash 存储器内部比特流映像分布示例



如上图所示，工作映像 0x0 位于默认的 0x0000 上电地址。

工作映像 0x0 包含一个指向工作映像 0x1 的 SPI Flash 跳转地址。

工作映像 0x1 包含一个指向 Golden Image 0x2 的 SPI Flash 跳转地址。

上电后，将自动从 0x0000 加载工作映像 0x0。

如果第一个工作映像 0x0 加载失败并且 FPGA 支持多次配置加载尝试，则 FPGA 将尝试加载下一个工作映像 0x1。

如果第二个工作映像 0x1 加载失败并且 FPGA 支持多于两次的配置加载尝试，则 FPGA 将尝试加载 Golden Image 0x2。

如果第一个工作映像 0x0 加载失败并且 FPGA 不支持多次配置加载尝试，则可以拉低 RECONFIG_N 以加载下一个工作映像 0x1。

如果第二个工作映像 0x1 加载失败并且 FPGA 不支持多于两次的配置加载尝试，则可以拉低 RECONFIG_N 以加载 Golden Image 0x2。

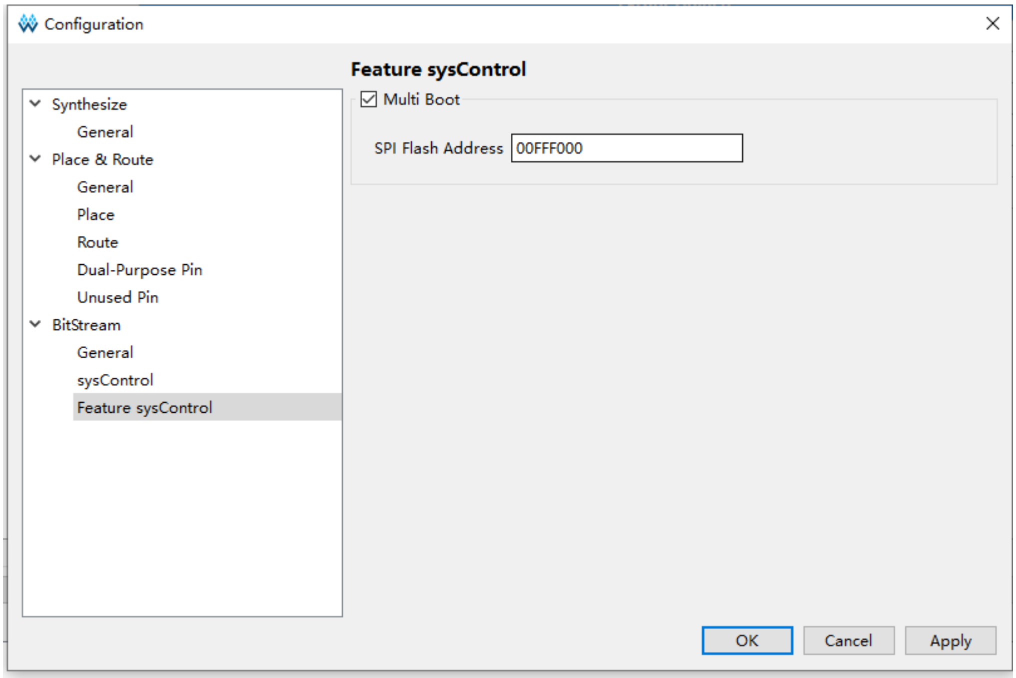
SPI Flash 启动地址

使用 GOWIN EDA 工具生成比特流时，用户可以指定下一个要加载的比特流的 SPI Flash 启动地址。

在高云半导体云源软件中打开“BitStream”配置框。

在“SPI Flash Address”输入框内设置下一个比特流的启动地址，如图 7-53 所示。

图 7-53 设置下一个 BitStream 的启动地址

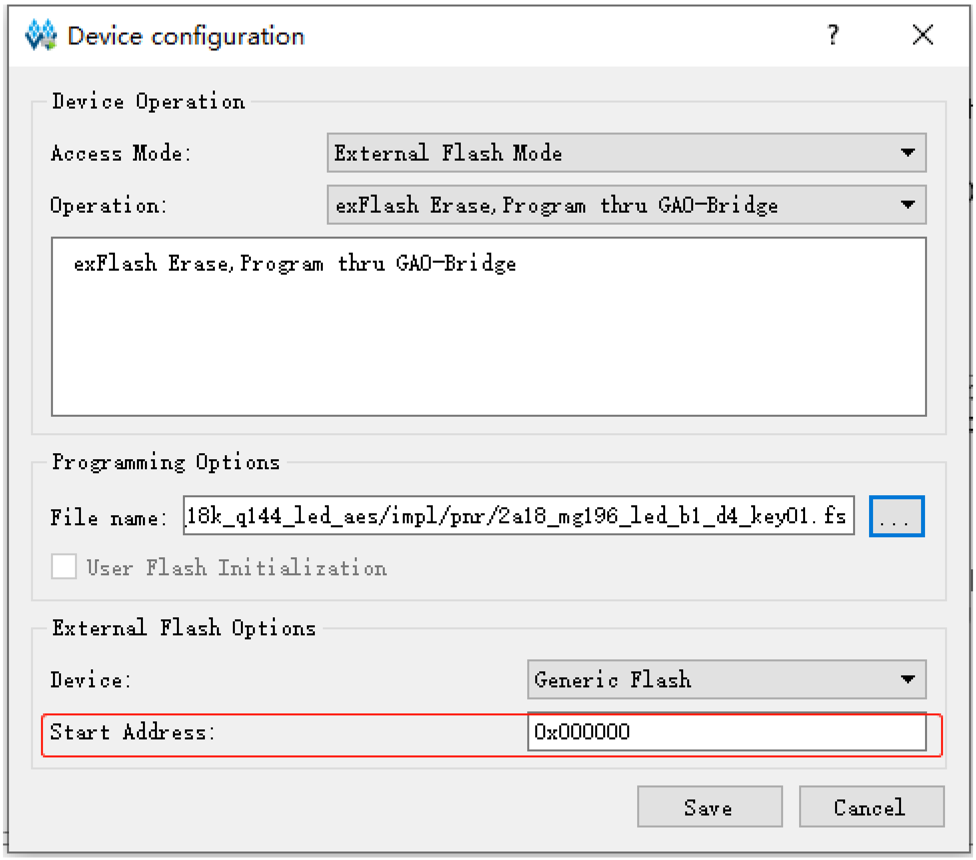


SPI Flash 的编程

Gowin Programmer 软件支持在不擦除的情况下将多个比特流数据编程到外部 Flash 中。

1. 在Programmer软件中选择“External Flash Mode”，并设置BitStream的起始编程地址，如图7-54所示。

图 7-54 设置外部 Flash 的编程地址



2. 单击“Save”，完成所有BitStream的启动地址和编程地址的设置。

注！

- Flash 启动地址会在上电时重置。
- 用户使用多重配置前需计算好比特流数据的大小，确保启动地址不会被前一个比特流数据所覆盖。
- SPI Flash 启动地址低 12 位为保留位，用户可设置的是 ADDR[23:12]的地址位。

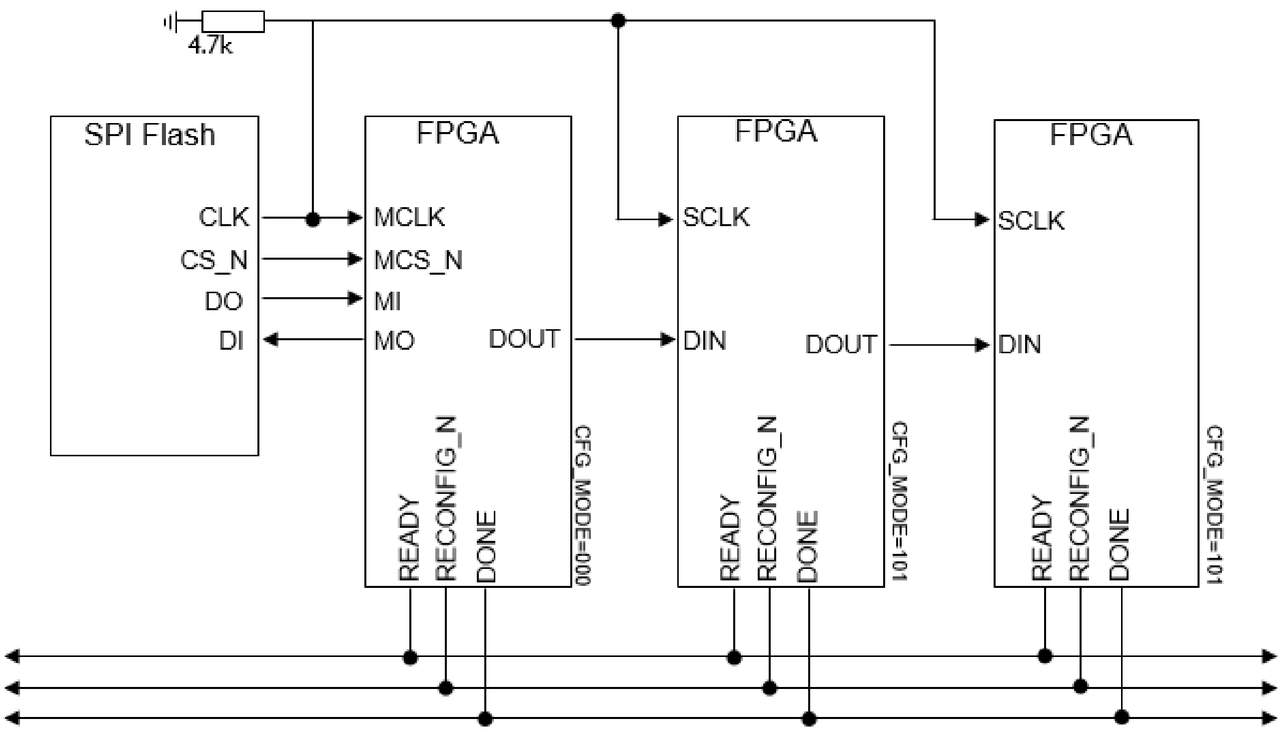
一片 Flash 配置多片 FPGA

高云半导体的 FPGA 产品支持一片 Flash 配置多片 FPGA 的情况。第一片 FPGA 器件采用 MSPI 配置模式与 SPI Flash 直接相连，下游 FPGA 采用 SERIAL 模式进行配置。一片 Flash 配置多片 FPGA 的连接示意图如图 7-55 所示。

注！

- 需要转发数据的器件，需要设置 Wake Up Mode 为 1。Wake Up Mode 一般用于菊花链环境，如一片外部 Flash 配置多个 FPGA。有关 Wake Up Mode 的更多信息，请参考 *SUG100，Gowin 云源软件用户指南*。
- 配置操作前需要将第一片 FPGA 的 MODE 值设置为 MSPI 模式，将下游 FPGA 的 MODE 值设置为 SERIAL 模式。
- 高云半导体的 FPGA 产品尚不支持多片 Flash 配置同一片 FPGA 的情况。

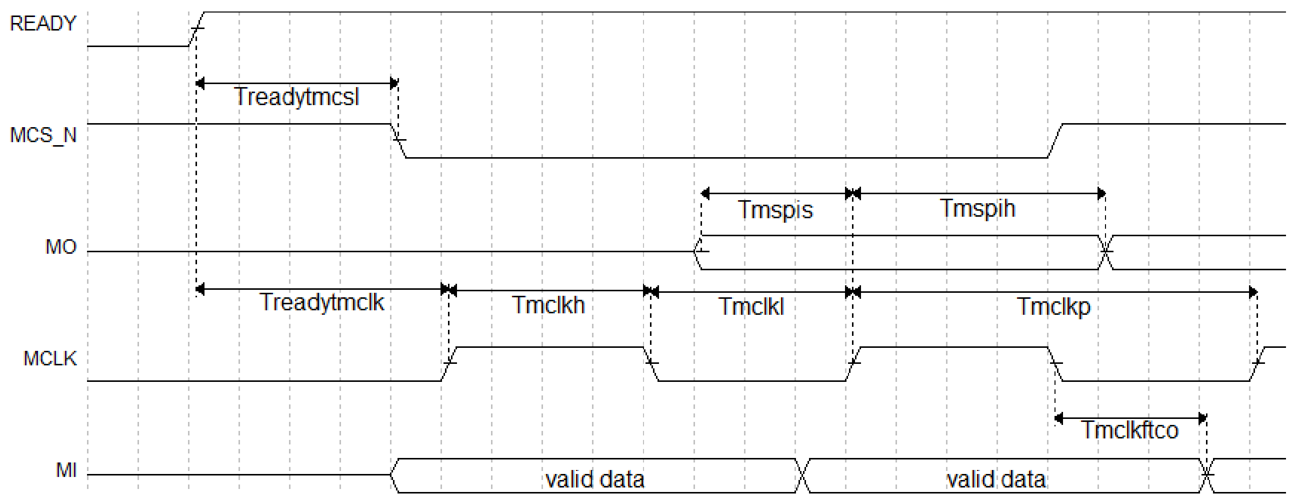
图 7-55 一片 Flash 配置多片 FPGA 连接示意图



7.5.5 MSPI 配置模式时序图

MSPI 下载模式时序图如图 7-56 所示。

图 7-56 MSPI 下载模式时序图



相关的时序参数如表 7-19 所示。

表 7-19 MSPI 配置模式时序参数

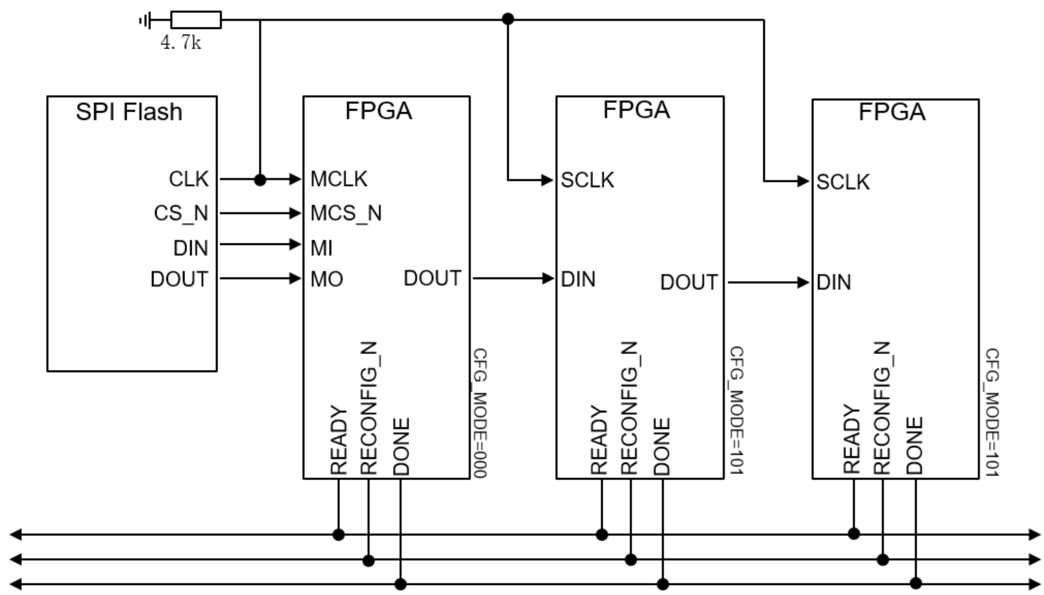
参数名称	参数含义	最小值	最大值
T_{mclkp}	MCLK 时钟周期(MCLK clock period)	15ns	-
T_{mclkh}	MCLK 时钟高电平时间(MCLK clock high time)	7.5ns	-

参数名称	参数含义	最小值	最大值
T_{mclkl}	MCLK 时钟低电平时间(MCLK clock low time)	7.5ns	-
T_{mspis}	MSPI PORT 建立时间(MSPI PORT setup time)	5ns	-
T_{mspih}	MSPI PORT 保持时间(MSPI PORT hold time)	1ns	-
$T_{mclkftco}$	MCLK 下降沿到数据输出时延(Time from MCLK falling edge to output)	-	10ns
$T_{readytmcsl}$	READY 上升沿到 MCS_N 低电平时间(Time from READY rising edge to MCS_N low)	100ns	200ns
$T_{readytmclk}$	READY 上升沿到第一个 MCLK 沿时间(Time from READY rising edge to first MCLK edge)	2.8μs	4.4μs

除满足上电要求外，使用 MSPI 模式对高云半导体产品进行配置，还需满足以下条件：

- MSPI 接口使能
  上电后初次配置或前一次配置时 RECONFIG_N 未设置为普通 I/O 状态。
- 启动新的配置
  重新上电或者低电平脉冲触发 RECONFIG_N 管脚。

图 7-57 MSPI 模式下的多 FPGA 模式连接示意图



7.6 双启动配置模式（仅小蜜蜂(LittleBee)家族支持）

双启动配置模式，即 DUAL BOOT 配置模式，是高云半导体小蜜蜂(LittleBee)家族非易失 FPGA 产品支持的一种配置模式。双启动配置模式下，FPGA 优先从外部 Flash 读取比特流数据完成配置。

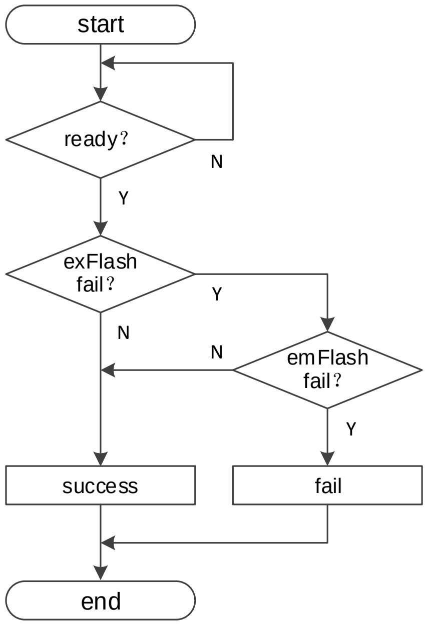
注！

在 DUAL BOOT 模式下，当外部 Flash 为空或不存在时，FPGA 会尝试从内部 flash 加载数据。

双启动配置模式需要选择特定的 MODE 值，内置 Flash 无须外部连线，从外部 Flash 启动的连线方案与 MSPI 配置模式相同，请参考图 7-50。双启动配置模式提供了更多的配置路径选择，用户可以按照自身需求选择配置数据的存储位置。

双启动配置模式的流程如图 7-58 所示。

图 7-58 双启动配置模式流程图



注！

MODE 值设置为“110”时 FPGA 优先选择从外部 Flash 启动。

对于 GW1N(R)-9 和 GW1NS 系列产品，无论是哪种模式的双启动配置，FPGA 均支持 4 次配置尝试：

- 优先选择的存储路径启动 3 次，3 次均失败后选择另一个路径进行配置。内置 Flash 的启动只能开始于 0 地址。
- MODE 值为“110”时，从外部 Flash 启动的 3 次尝试可以选择不同的启动地址，启动地址需要预先通过云源软件写入比特流数据，当外部的 3 次配置尝试都失败时器件选择从内置 Flash 启动。
- 对于 GW1NS 系列产品，可以支持启动失败后的多次重启尝试操作，但是无法修改启动地址。

注！

SPI Flash 启动地址低 12 位无效，用户可设置的是 ADDR[23:12]的地址空间。

GW1N(R)-4 器件目前尚不支持器件自动的双启动配置，高云半导体为用户提供了这两款器件的双启动配置方案，详细信息请参考手册 *TN101,基于高云半导体 GW1N-4 芯片的 DUAL BOOT 下载方案*。

7.7 CPU 配置模式

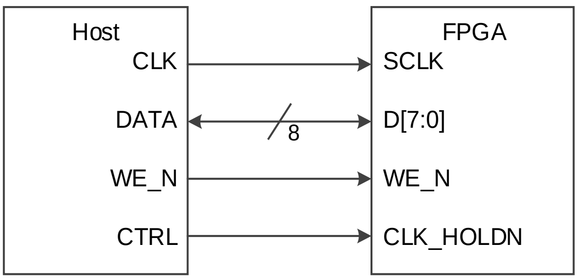
CPU 配置模式下，Host 通过 8-bit 位宽的数据总线接口对高云半导体 FPGA 产品进行配置，CPU 配置模式的管脚如表 7-20 所示。

表 7-20 CPU 配置模式管脚

管脚名称	I/O 类型	说明
RECONFIG_N	I，内部弱上拉	低电平脉冲：启动 GowinCONFIG 配置
READY	I/O	高电平：当前可以对器件进行编程配置 低电平：禁止对器件进行编程配置
DONE	I/O	高电平：成功完成编程配置 低电平：未完成编程配置或编程配置失败
MODE[2:0]	I，内部弱上拉	配置模式选择信号，READY 上升沿采样
SCLK	I	输入时钟
CLKHOLD_N	I，内部弱上拉	CPU 模式下的片选信号，低电平有效。即，若要在 CPU 模式下配置 FPGA，此信号需为低电平。
WE_N	I	读写使能 0：写 1：读
D[7:0]	I/O	数据输入输出端口：CPU 配置模式时作为输入管脚，配置结束后可以转换为输出管脚进行验证

CPU 配置模式的连接示意图如图 7-59 所示。

图 7-59 CPU 配置模式连接示意图



注！

此图为 CPU 配置模式的最小系统图，MODE 值设置为“111”，其他固定管脚的接法请参考图 7-1。

除满足上电要求外，使用 CPU 模式对高云半导体 FPGA 产品进行配置，还需满足以下条件：

- CPU 接口使能
  上电后初次配置或前一次配置时 RECONFIG_N 未设置为普通 I/O 状态。
- 启动新的配置

重新上电或者低电平脉冲触发 RECONFIG_N 管脚。

7.7.1 配置时序

在配置之前确保 MODE[2：0]=111，在配置完成后 DONE 会拉高。如果 DONE 或者 READY 被拉低，说明配置未成功。

配置过程中，数据总线 D[7:0]按大端模式（MSB）表示，FPGA 在 SCLK 下降沿读取数据。

图 7-60 CPU 模式配置流程示意图

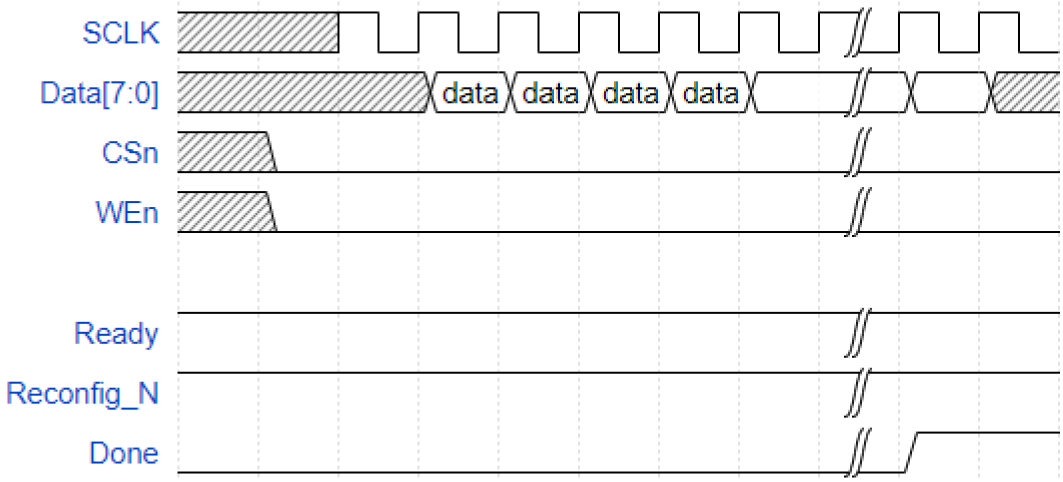


图 7-61 CPU 配置模式时序图

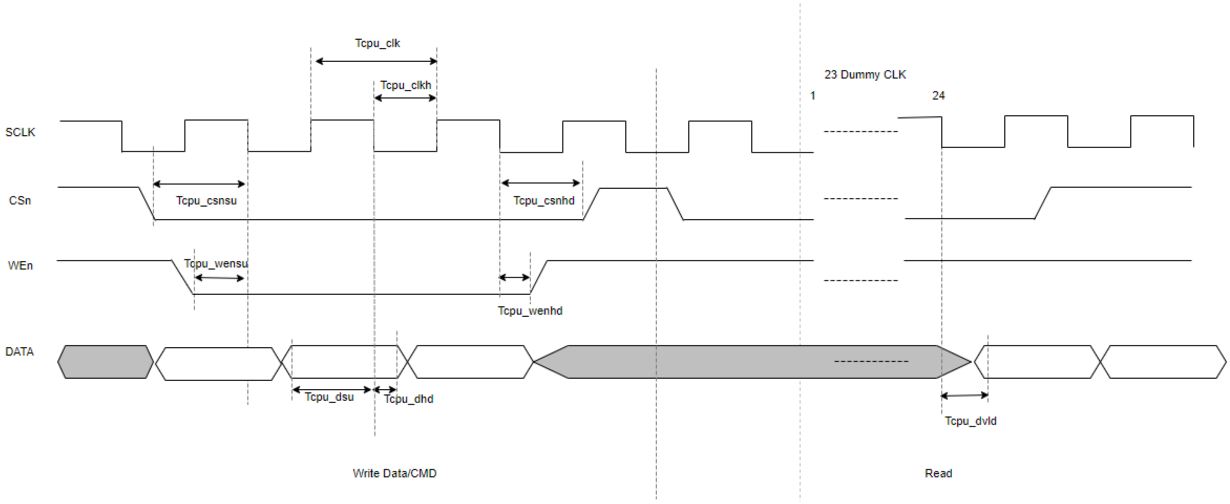


表 7-21 CPU 配置模式时序参数

名称	描述	最小值	最大值	单位
T_{cpu_clk}	CPU input clock period	40	–	ns
T_{cpu_csnsu}	CLKHOLD_N(CSn) setup time to SCLK falling	8	–	ns
T_{cpu_csnhd}	CLKHOLD_N(CSn) hold time from SCLK falling	0	–	ns
T_{cpu_wensu}	WE_N setup time to SCLK falling	8	–	ns
T_{cpu_wenhd}	WE_N hold time from SCLK falling	0	–	ns
T_{cpu_dsu}	Write data input setup time to SCLK falling	10	–	ns
T_{cpu_dhd}	Write data input hold time from SCLK falling	0	–	ns
T_{cpu_dvld}	SCLK falling to read data output valid	–	10	ns
T_{cpu_clkh}	CPU input clock high duration	(clock cycle) *45%	(clock cycle) *55%	–

7.8 SERIAL 配置模式

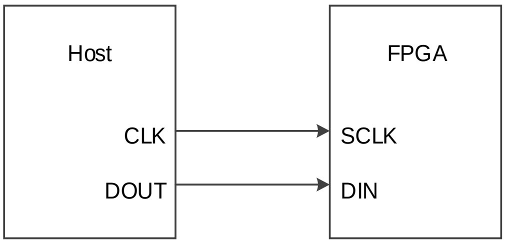
SERIAL 配置模式，Host 通过串行接口对高云半导体 FPGA 产品进行配置。SERIAL 配置模式是使用管脚数量最少的配置模式之一。SERIAL 配置模式只能将比特流数据写入 FPGA，无法从 FPGA 器件回读数据，因此，SERIAL 配置模式无法读取 ID CODE 和 USER CODE 以及状态寄存器信息。SERIAL 配置模式的管脚定义如表 7-22 所示。

表 7-22 SERIAL 配置模式管脚定义

管脚名称	I/O 类型	说明
RECONFIG_N	I,内部弱上拉	低电平脉冲：启动 GowinCONFIG 配置
READY	I/O	高电平：当前可以对器件进行编程配置 低电平：禁止对器件进行编程配置
DONE	I/O	高电平：成功完成编程配置 低电平：未完成编程配置或编程配置失败
MODE[2:0]	I,内部弱上拉	配置模式选择信号，READY 上升沿采样
SCLK	I	输入时钟
DIN	I,内部弱上拉	输入数据
DOUT	O	输出数据，只用于 FPGA 级联时 SERIAL 配置模式

SERIAL 配置模式的连接示意图如图 7-62 所示。

图 7-62 SERIAL 配置模式连接示意图



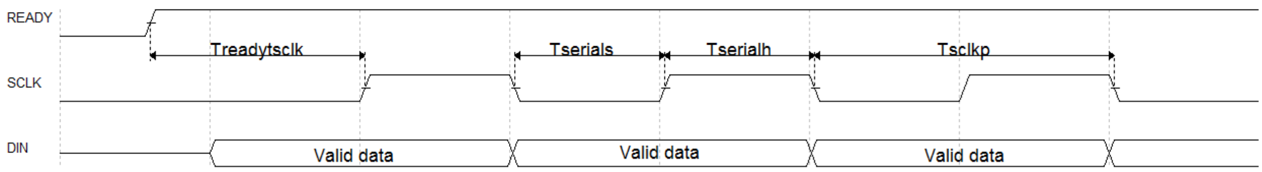
注！

此图为 SERIAL 配置模式的最小系统图，MODE 值为“101”，其他固定管脚的接法请参考图 7-1。

SERIAL 配置模式时序图

SERIAL 配置模式时序如图 7-63 所示。

图 7-63 SERIAL 配置模式时序图



相关的时序参数如表 7-23 所示。

表 7-23 SERIAL 配置模式时序参数

参数名称	参数含义	最小值	最大值
T_{sclkp}	SCLK 时钟周期(SCLK clock period)	15ns	-
$T_{serials}$	SERIAL PORT 建立时间(SERIAL PORT setup time)	2ns	-
$T_{serialh}$	SERIAL PORT 保持时间(SERIAL PORT hold time)	0ns	-
$T_{readytsclk}$	READY 上升沿到第一个 SCLK 沿时间(Time from READY rising edge to first SCLK edge)	TBD	-

除满足上电要求外，使用 SERIAL 模式对高云半导体 FPGA 产品进行配置，还需满足以下条件：

- SERIAL 接口使能
  上电后初次配置或前一次配置时 RECONFIG_N 未设置为普通 I/O 状态。
- 启动新的配置
  重新上电或者低电平脉冲触发 RECONFIG_N 管脚。

7.9 I²C 配置模式

注！

- 小蜜蜂(LittleBee)家族 FPGA 产品处于 I²C 配置模式时，同时支持 Autoboot 模式，芯片上电后，FPGA 先自行从内置 Flash 读取比特流数据完成配置。Autoboot 配置期间，I²C SDA 线必须保持外部上拉状态，否则设备可能无法正确配置；另外，建议同时外部上拉 SCL 线。请注意：此注释亦适用于 SDA 和 SCL 内部弱上拉的 C 版本器件。
- 重新配置 SRAM 时，需要先擦除已存在的 SRAM，并确保内部 Flash 为空。可通过重新上电或者低电平脉冲触发 RECONFIG_N 管脚来擦除 SRAM。
- C 版本的 GW1N-2 和 GW1N-1P5 器件的内部 Flash 不支持通过原生 I2C 进行编程。但是可以使用 goConfig I2C IP 进行编程，从而实现背景升级。背景升级时，需先使用 goConfig I2C IP 擦除内置 Flash，详细信息请参考 IPUG795, Gowin goConfig I2C IP 用户指南，TN715,Gowin goConfigIP(I2C)示例代码使用说明。

I²C 配置模式，Host 通过 I²C 接口对高云半导体 FPGA 产品进行配置。I²C 配置模式是使用管脚数量最少的配置模式之一。I²C 配置模式只能将比特流数据写入 FPGA，无法从 FPGA 器件回读数据，因此，I²C 配置模式无法读取 ID CODE、USER CODE、状态寄存器和回读校验。I²C 配置模式的管脚定义如表 7-24 所示。

表 7-24 I²C 配置模式管脚定义

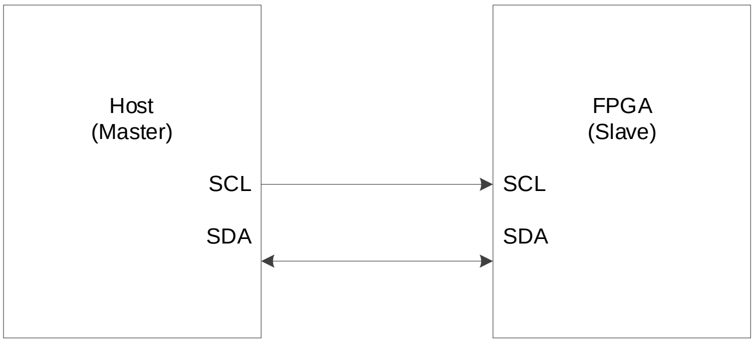
管脚名称	I/O 类型	说明
RECONFIG_N	I，内部弱上拉	低电平脉冲：启动 GowinCONFIG 配置
READY	I/O	高电平：当前可以对器件进行编程配置 低电平：禁止对器件进行编程配置
DONE	I/O	高电平：成功完成编程配置 低电平：未完成编程配置或编程配置失败
MODE[2:0]	I，内部弱上拉	配置模式选择信号，READY 上升沿采样
SCL	I[1]	输入时钟
SDA	I/O[1]	输入数据，或输出 ACK

注！

[1] C 版本器件的 SCL 管脚和 SDA 管脚具有内部弱上拉，但强烈推荐添加外部上拉电阻。

I²C 配置模式的连接示意图如图 7-64 所示。

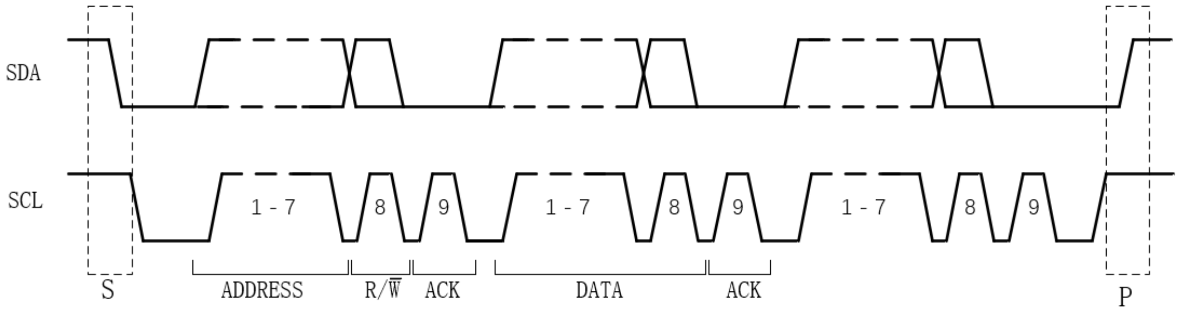
图 7-64 I²C 配置模式连接示意图



注！

此图为 I²C 配置模式的最小系统图，MODE 值为“100”，其他固定管脚的接法请参考图 7-1。

图 7-65 I²C 配置模式时序图



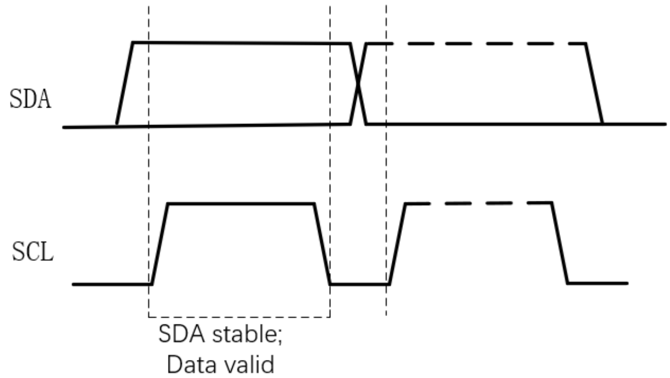
I²C 是串行传输总线，按照上图所示协议进行数据传输，正常状态下，SDA 和 SCL 都处于高电平。

表 7-25 I²C 配置模式时序参数

参数	参数含义	
S	（I2C Start）启动条件	SCL 是高电平时，SDA 从高电平向低电平切换。
P	（I2C Stop）停止条件	SCL 是高电平时，SDA 由低电平向高电平跳变。
ADDRESS	地址帧	每个从属设备唯一的 7 位或 10 位序列，用于在主设备要与之通信时标识该从属设备。
R/W	读/写位	指定主设备是向从设备发送数据（0）还是向从机读取数据（1）。
ACK	ACK/NACK 位	消息中的每个帧后均带有一个 ACK/NACK 位，高云 FPGA 正确时返回 0。
DATA	数据	一个数据有 8bits，并以最高有效位优先发送。

I²C 总线上的所有 DATA 都是以 8 位字节传送的，发送器每发送一个字节，就在时钟脉冲 9 期间释放数据线，由接收器反馈一个应答信号。应答信号为低电平时，规定为有效应答位（ACK 位），表示接收器已经成功地接收了该字节；应答信号为高电平时，规定为非应答位（NACK），一般表示

接收器接收该字节没有成功，对于反馈有效应答位 ACK 的要求是，接收器在第 9 个时钟脉冲之前的低电平期间将 SDA 线拉低，并且确保在该时钟的高电平期间为稳定的低电平。如果接收器是主控器，则在它收到最后一个字节后，发送一个 NACK 信号，以通知被控发送器结束数据发送，并释放 SDA 线，以便主控接收器发送一个停止信号。在 I2C 总线上传送的每一位数据都有一个时钟脉冲相对应（或同步控制），即在 SCL 串行时钟的配合下，在 SDA 上逐位地串行传送每一位数据。进行数据传送时，在 SCL 呈现高电平期间，SDA 上的电平必须保持稳定，低电平为数据 0，高电平为数据 1。只有在 SCL 为低电平期间，才允许 SDA 上的电平改变状态。逻辑 0 的电平为低电压，而逻辑 1 则为高电平，如下图所示。



高云器件支持的 I²C 配置模式信息如表 7-26 所示。

表 7-26 I²C 配置模式频率及地址

模式	器件	频率	地址
配置 SRAM	GW1N-2 （IDCode:0x0120681B）	100Khz~1.33Mhz	7'b1010_000

注!

I2C 操作 Flash，需要使用 goConfig I2C IP 实现。

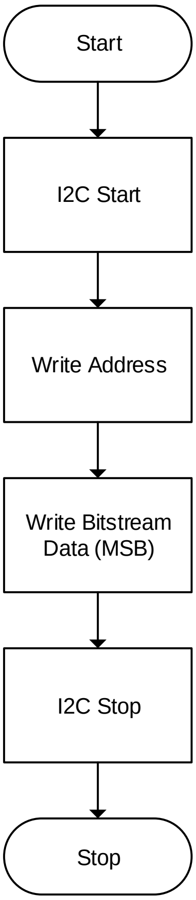
除满足上电要求外，使用 I2C 模式对高云半导体 FPGA 产品进行配置，还需满足以下条件：

- I2C 接口使能
  上电后初次配置或前一次配置时 RECONFIG_N 未设置为普通 I/O 状态。
- 启动新的配置
  重新上电或者低电平脉冲触发 RECONFIG_N 管脚。

7.9.1 配置 GW1N-2 SRAM 流程图

配置 SRAM 使用的数据流文件格式是 FS（.fs 后缀名）或 Binary（.bin 后缀名）文件。无论是哪种文件格式，发送数据时都是逐字节按照 MSB 的方式发送。

图 7-66 GW1N-2 配置或烧录 SRAM 流程图



8 安全性考虑

用户使用 FPGA 进行设计，安全性问题是一个关键的考虑因素，高云半导体的编程软件结合器件特性开发了一系列安全保护措施，为用户的比特流数据提供了完善的安全性保障机制。

安全措施大致分为三个阶段：

- 配置开始前，编程软件自动检查比特流数据的合法性。
- 配置过程中，器件实时校验传输数据的正确性。
- 配置完成后，器件进入工作状态，屏蔽一切形式的回读请求。

三个阶段的详细描述信息如下：

配置开始前

使用高云半导体的编程软件进行配置操作，请参见以下步骤：

1. 进行配置电路的硬件连接。
2. 启动编程软件进行器件扫描，软件自动识别已连接的 FPGA 产品。
3. 选择比特流数据和编程配置模式进行器件的编程配置。

上述过程中，编程软件首先读取已连接器件的 ID，然后将其与用户选择的比特流数据中的 ID 进行比较，只有二者一致才能进行操作，否则，用户选择的比特流数据被判定为非法数据，无法进行编程配置。

注！

高云半导体 FPGA 产品具有特定的 ID，以便与其他系列产品进行区分。使用高云半导体云源软件生成的比特流数据中自动添加了器件的 ID 验证指令，用户只需在建立工程时选择具体器件即可。

配置过程中

配置过程开始后，器件首先读取比特流数据的 ID 信息进行校验，校验通过后开始编程配置过程。为防止比特流数据被篡改的情况和传输过程中可能发生的错误，高云半导体器件采用 CRC 校验的方式确保比特流文件中的所有数据位正确写入 FPGA，具体过程如下：

高云半导体云源软件生成的比特流数据中每段地址后都加入了该段地址对应数据的 CRC 校验码，高云半导体器件在接收数据的过程中也会不断地生成校验码，与器件接收的校验码进行比较，一旦发现校验错误，之后的数据将被忽略，配置完成后 DONE 指示灯不会被点亮，编程软件界面上弹出 CRC 校验出错的提示。

配置完成后

配置完成后，根据用户选择的编程配置模式，器件的比特流数据加载到 SRAM 中完成启动或者存储在内置 Flash 中（仅小蜜蜂(LittleBee)家族 FPGA 产品支持存储在内置 Flash 中。）。

- 对于加载到 SRAM 中的数据，高云半导体云源软件在生成比特流数据的过程中自动设置了安全位，任何用户都无法读取 SRAM 中的数据。
- 对于存储在内置 Flash 中的数据，软件完成 Flash 的编程操作后，Flash 配置为自启动模式，禁止所有的读取请求。

此外，小蜜蜂(LittleBee)家族 FPGA 产品的自启动配置模式，由于不需要与外部下载接口进行连接，从而大大降低了配置过程中数据被截获的风险，为用户提供了更高的安全保障。双启动模式为用户提供了一种备份选择，用户可以根据自身需要将配置数据备份在外部 Flash 中。

注！

高云半导体不对外置 Flash 的存储安全性负责。

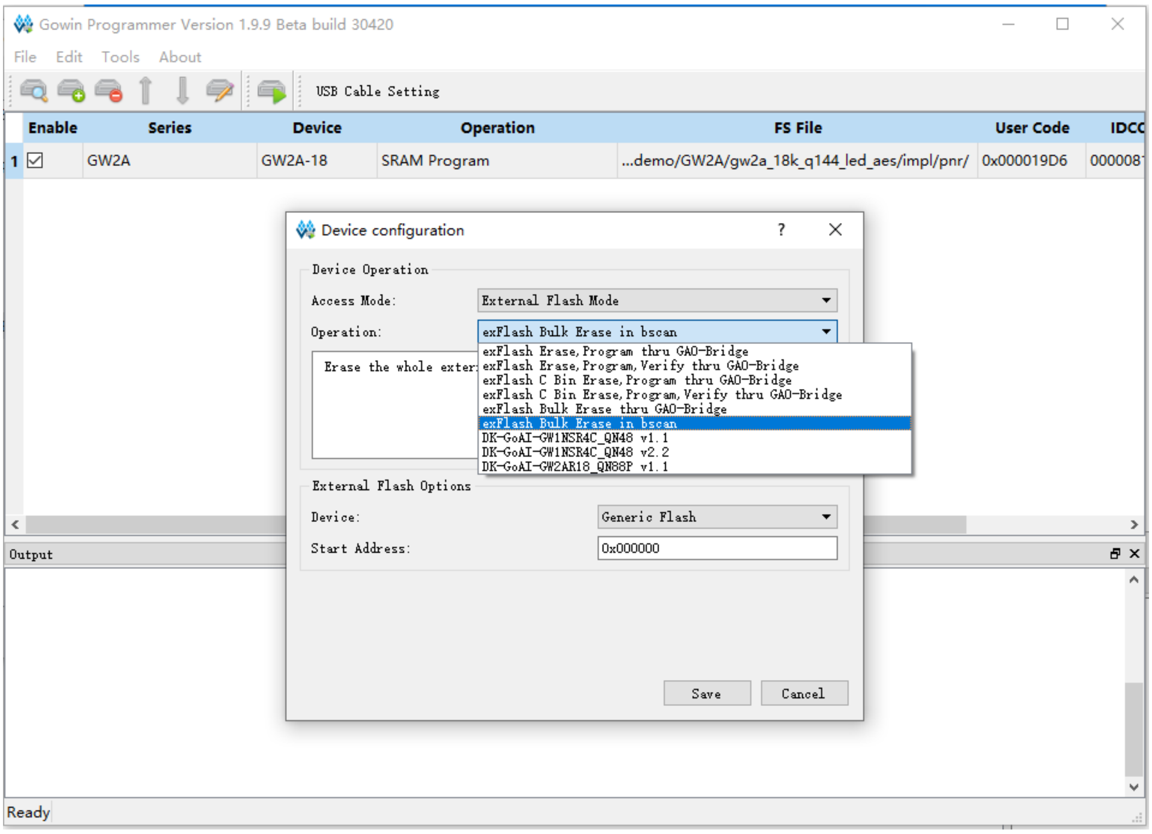
9 边界扫描操作

边界扫描操作是 JTAG 配置模式的扩展，扫描链分为长链和短链：长链主要结合 BSDL 文件进行器件的测试工作；短链主要针对 FPGA 链上的外部 Flash 进行擦除和读写操作。

边界扫描操作步骤如下：

1. 将 FPGA 开发板连接到 PC 并上电。
2. 打开 Gowin 编程软件扫描已连接的器件。
3. 在 Operation 下方双击选择外部 Flash 操作，并选择相关的 bscan 操作，如图 9-1 所示。

图 9-1 边界扫描操作示意图



边界扫描操作只能对 FPGA 外部 Flash 进行操作，无法对内置 Flash 或 SRAM 进行编程配置。边界扫描操作编程外部 Flash 时不必关心 FPGA MODE 值的设置，但是相比于常规 JTAG 编程外部 Flash 的方法，边界扫描操作的编程速度较慢。

10 SPI Flash 选择

高云半导体 FPGA 产品支持的外部 SPI Flash 器件操作指令如表 10-1 所示。

表 10-1 SPI Flash 操作指令

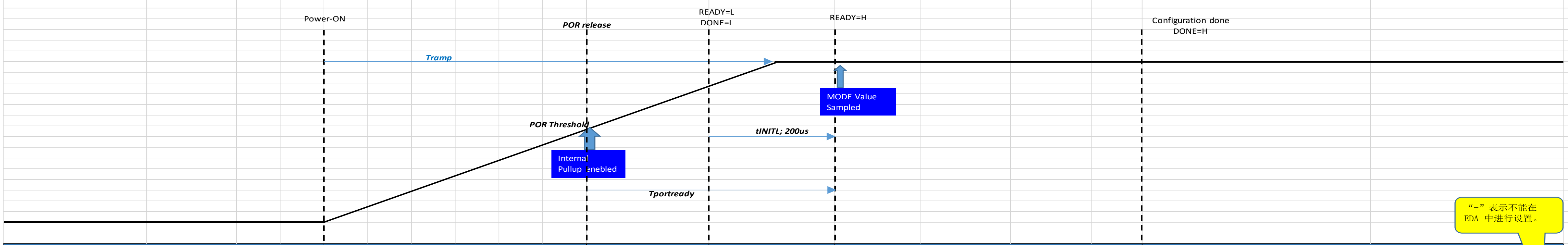
操作	指令
Read	0x03
Fast_Read	0x0B
Page Program	0x02
Sector Erase	0x20
Chip Erase	0xC7
Read Status Code	0x05
Read JEDEC ID	0x9F
Write Enable	0x06
Write Disable	0x04

注！

- 高云半导体 FPGA 支持的 Flash 读指令必须至少有一种是 0x03 或 0x0B，当时钟频率不高于 30MHz 时使用普通读指令；当时钟频率高于 30MHz 时，使用快速读指令，快速读取需要将 FASTRD_N 管脚拉低，时钟频率不能高于 66.6MHz。
- Read(0x03)和 Fast_Read(0x0B)是器件在 MSPI 模式时仅支持的指令；其他指令是用 Programmer 烧录 Flash 时所用到的指令，客户可以自行烧录。
- 默认情况下 SPI Flash 需工作在 Standard SPI 协议。
- 测试中多以 winbond、兆易创新、以及 ISSI(如 ISSI 的 IS25LP064A-JBLE)3 个品牌为主，使用其他品牌的话，即使满足上面的指令要求，可能由于时序的差异而有例外。

附录 A 管脚状态信息

表 A-1 小蜜蜂家族 1K, 4K, 9K 器件的管脚在各个阶段时的状态

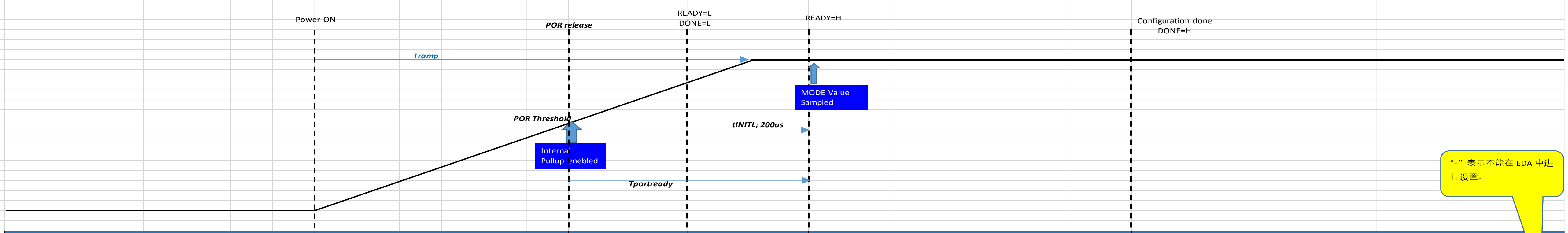


Stage			Power-ON to POR Voltage	POR to Start initialization	During initialization	During Configuration		After Configuration	
						Internal State	External Pull	Default	EDA settings for GPIO
State of Used GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	Depends on FloorPlanner by user constraints	-
State of Unused GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	Hi-Z with Weak pullup	Depends on EDA settings
State of Dual-popose Pins	MODE[2:0]		Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup or Pulldown	GPIO Input or Output	-
	READY		Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 1 with Weak Pullup	Pullup(Recommended)	READY (Open drain output 1 with Weak Pullup)	Input or Output
	RECONFIG_N		Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup(Recommended)	RECONFIG_N (Input with Weak Pullup)	Output
	DONE		Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 0 with Weak Pullup	Pullup(Recommended)	DONE (Open drain output 1 with Weak Pullup)	Input or Output
	JTAG	JTAGSEL_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	JTAGSEL_N Input with Weak pullup
		TCK	Hi-Z without pull	Hi-Z without pull		Input	Pulldown	TCK	Input or Output
		TMS	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TMS (Input with Weak Pullup)	Input or Output
		TDI	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TDI (Input with Weak Pullup)	Input or Output
		TDO	Hi-Z without pull	Hi-Z without pull		Output	-	TDO	Input or Output
	MSPI (非MSPI模式)	**FASTRD_N**/D3	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	Input or Output
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK (Hi-Z with Weak PullDown)	Input or Output
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MCS_N	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	Input or Output
	MSPI	**FASTRD_N**/D3	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	FASTRD_N	-
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	MCLK (Output)	Input or Output
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MCS_N	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MO	Input or Output
	SSPI (非SSPI模式)	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SSPI_CS_N (Hi-Z with Weak Pullup)	Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SI	Input or Output
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SO	Input or Output
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	Input or Output
	SSPI	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SSPI_CS_N (Input with Weak Pullup)	Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SI	Input or Output
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	SO	Input or Output
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	Input or Output
	SERIAL (非SERIAL模式)	CLKHOLD_N/**DIN**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	depend ont SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	GPIO Input or Output
	SERIAL	CLKHOLD_N/**DIN**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	GPIO Input or Output by SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	-
	CPU (非CPU模式)	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/**WE_N**	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	-
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	GPIO Input or Output by SSPI settings
		MI/**D7**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	GPIO Input or Output by MSPI settings
		MO/**D6**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	GPIO Input or Output by MSPI settings
		MCS_N/**D5**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MCS_N	GPIO Input or Output by MSPI settings
		MCLK/**D4**	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK (Hi-Z with Weak PullDown)	GPIO Input or Output by MSPI settings
		FASTRD_N/**D3**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	Input or Output
		SI/**D2**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SI	Input or Output
		SO/**D1**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SO	Input or Output
		SSPI_CS_N/**D0**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SSPI_CS_N (Hi-Z with Weak Pullup)	Input or Output
	CPU	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/**WE_N**	Hi-Z without pull	Hi-Z with Weak PullDown		Input with Weak Pullup	-	GPIO Input or Output	-
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	GPIO Input or Output by SSPI settings
		MI/**D7**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MI	GPIO Input or Output by MSPI settings
		MO/**D6**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MO	GPIO Input or Output by MSPI settings
		MCS_N/**D5**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MCS_N	GPIO Input or Output by MSPI settings
		MCLK/**D4**	Hi-Z without pull	Hi-Z with Weak PullDown		Inout with Weak Pullup	-	MCLK (Hi-Z with Weak PullDown)	GPIO Input or Output by MSPI settings
		FASTRD_N/**D3**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	GPIO Input or Output	-
		SI/**D2**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SI	Input or Output
		SO/**D1**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SO	Input or Output
		SSPI_CS_N/**D0**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SSPI_CS_N (Input with Weak Pullup)	Input or Output

蓝色表示该管脚可在EDA中配置为GPIO。对于多功能管脚，粗体部分有效。

“-”表示不能在EDA 中进行设置。

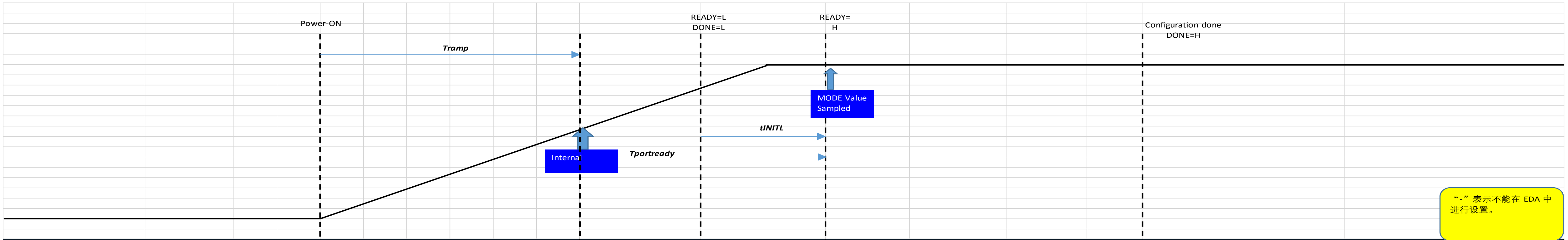
表 A-2 小蜜蜂家族 1P5K, 2K 器件的管脚在各个阶段时的状态



蓝色表示该管脚可在EDA中配置为GPIO。
对于多功能管脚，粗体部分有效。

Stage			Power-ON to POR Voltage	POR to Start initialization	During initialization	During Configuration		After Configuration	
						Internal State	External Pull	Default	EDA settings for GPIO
State of Used GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	Depends on FloorPlanner by user constraints	-
State of Unused GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	Hi-Z with Weak pullup	Depends on EDA settings
State of Dual-popose Pins		MODE[2:0]	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup or Pulldown	GPIO Input or Output	-
		READY	Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 1 with Weak Pullup	Pullup(Recommended)	READY (Open drain output 1 with Weak Pullup)	Input or Output
		RECONFIG_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup(Recommended)	RECONFIG_N (Input with Weak Pullup)	Output
		DONE	Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 0 with Weak Pullup	Pullup(Recommended)	DONE (Open drain output 1 with Weak Pullup)	Input or Output
	JTAG	JTAGSEL_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	JTAGSEL_N Input with Weak pullup
		TCK	Hi-Z without pull	Hi-Z without pull		Input	Pulldown	TCK	Input or Output
		TMS	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TMS (Input with Weak Pullup)	Input or Output
		TDI	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TDI (Input with Weak Pullup)	Input or Output
		TDO	Hi-Z without pull	Hi-Z without pull		Output	-	TDO	Input or Output
	MSPI (非MSPI模式)	FASTRD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	Input or Output
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK (Hi-Z with Weak PullDown)	Input or Output
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MCS_N	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	Input or Output
	MSPI	FASTRD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	FASTRD_N	-
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	MCLK (Output)	Input or Output
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MCS_N	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MO	Input or Output
	SSPI (非SSPI模式)	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SSPI_CS_N (Hi-Z with Weak Pullup)	Input or Output
		SI	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SI	Input or Output
		SO	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SO	Input or Output
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	Input or Output
	SSPI	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SSPI_CS_N (Input with Weak Pullup)	Input or Output
		SI	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SI	Input or Output
		SO	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	SO	Input or Output
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	Input or Output
	SERIAL (非SERIAL模式)	CLKHOLD_N/**DIN**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	depend ont SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	GPIO Input or Output
	SERIAL	CLKHOLD_N/**DIN**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	GPIO Input or Output by SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	-
	CPU (非CPU模式)	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/**WE_N**	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	-
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N (Hi-Z with Weak Pullup)	GPIO Input or Output by SSPI settings
		MI/**D7**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	GPIO Input or Output by MSPI settings
		MO/**D6**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	GPIO Input or Output by MSPI settings
		MCS_N/**D5**	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MCS_N	GPIO Input or Output by MSPI settings
		MCLK/**D4**	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK (Hi-Z with Weak PullDown)	GPIO Input or Output by MSPI settings
		D0-D3	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	-
	CPU	**SCLK**	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/**WE_N**	Hi-Z without pull	Hi-Z with Weak PullDown		Input with Weak Pullup	-	GPIO Input or Output	-
		CLKHOLD_N/DIN	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N (Input with Weak Pullup)	GPIO Input or Output by SSPI settings
		MI/**D7**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MI	GPIO Input or Output by MSPI settings
		MO/**D6**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MO	GPIO Input or Output by MSPI settings
		MCS_N/**D5**	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	MCS_N	GPIO Input or Output by MSPI settings
		MCLK/**D4**	Hi-Z without pull	Hi-Z with Weak PullDown		Inout with Weak Pullup	-	MCLK (Hi-Z with Weak PullDown)	GPIO Input or Output by MSPI settings
		D0-D3	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	-	GPIO Input or Output	-
	I2C (非I2C模式)	SCL	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	External Pullup recommended for packages with fixed MODE=100	SCL (Hi-Z with Weak Pullup)	Input
		SDA	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	External Pullup required for packages with fixed MODE=100	SDA (Hi-Z with Weak Pullup)	Input or Output
	I2C	SCL	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup	SCL (Input with Weak Pullup)	Input
		SDA	Hi-Z without pull	Hi-Z with Weak Pullup		Inout with Weak Pullup	Pullup	SDA (Inout with Weak Pullup)	Input or Output

表 A-3 晨熙家族 18K, 55K 器件的管脚在各个阶段时的状态



“-” 表示不能在 EDA 中进行设置。

蓝色表示该管脚可在EDA中配置为GPIO。

MSPI related pins are MCLK, MCS_N, MI, MO.

SSPI related pins are SCLK, CLKHOLD_N, SI, SO, SSPI_CS_N.

Stage			Power-ON to POR Voltage	POR to Start initialization	During initialization	During Configuration: Internal	During Configuration: External	After Configuration: Default	After Configuration: EDA settings for GPIO
State of Used GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup		Depends on FloorPlanner user settings	-
State of Unused GPIOs			Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup		Hi-Z with Weak pullup	Depends on EDA settings
State of Dual-popose Pins	MODE[2:0]		Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup or Pulldown	GPIO Input or Output	-
	READY		Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 1 with Weak Pullup	Pullup(Recommended)	READY	Input or Output
	RECONFIG_N		Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	Pullup(Recommended)	RECONFIG_N	Output
	DONE		Hi-Z without pull	Open drain output 0 with Weak Pullup		Open drain output 0 with Weak Pullup	Pullup(Recommended)	DONE	Input or Output
	JTAG	JTAGSEL_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	JTAGSEL_N Input with Weak pullup
		TCK	Hi-Z without pull	Hi-Z without pull		Input	Pulldown	TCK	Input or Output
		TMS	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TMS	Input or Output
		TDI	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	TDI	Input or Output
		TDO	Hi-Z without pull	Hi-Z without pull		Output	-	TDO	Input or Output
	MSPI(非MSPI模式)	FASTRD_N/D3	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	Input or Output
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	Input or Output
	MSPI	FASTRD_N/D3	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	FASTRD_N	-
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	MCLK	Input or Output
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MCS_N	Input or Output
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	MI	Input or Output
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	MO	Input or Output
	SSPI（非SSPI模式）	SCLK	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SSPI_CS_N	Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SI	Input or Output
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SO	Input or Output
		DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N	Input or Output
	SSPI	SCLK	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	Input or Output
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SSPI_CS_N	Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SI	Input or Output
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		Output	-	SO	Input or Output
		DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N	Input or Output
	SERIAL（非SERIAL模式）	DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N	depend ont SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	GPIO Input or Output	GPIO Input or Output
	SERIAL	DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N	GPIO Input or Output by SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Output	-	GPIO Input or Output	-
	CPU（非CPU模式）	SCLK	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	GPIO Input or Output	-
		DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	CLKHOLD_N	GPIO Input or Output by SSPI settings
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MI	GPIO Input or Output by MSPI settings
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	MO	GPIO Input or Output by MSPI settings
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		Hi-Z with Weak PullDown	-	MCLK	GPIO Input or Output by MSPI settings
		FASTRD_N/D3	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	GPIO Input or Output	GPIO Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SI	GPIO Input or Output by SSPI settings
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SO	GPIO Input or Output by SSPI settings
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		Hi-Z with Weak Pullup	-	SSPI_CS_N	GPIO Input or Output by SSPI settings
	CPU	SCLK	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	SCLK	GPIO Input or Output by SSPI settings
		DOUT/WE_N	Hi-Z without pull	Hi-Z with Weak PullDown		Input with Weak PullDown	-	GPIO Input or Output	-
		DIN/CLKHOLD_N	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	CLKHOLD_N	GPIO Input or Output by SSPI settings
		MI/D7	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	MI	GPIO Input or Output by MSPI settings
		MO/D6	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	MO	GPIO Input or Output by MSPI settings
		MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	MCS_N	GPIO Input or Output by MSPI settings
		MCLK/D4	Hi-Z without pull	Hi-Z with Weak PullDown		InOut	-	MCLK	GPIO Input or Output by MSPI settings
		FASTRD_N/D3	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	GPIO Input or Output	GPIO Input or Output
		SI/D2	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	SI	GPIO Input or Output by SSPI settings
		SO/D1	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	SO	GPIO Input or Output by SSPI settings
		SSPI_CS_N/D0	Hi-Z without pull	Hi-Z with Weak Pullup		InOut	-	SSPI_CS_N	GPIO Input or Output by SSPI settings
	SSPI	MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		output	-	MCS_N	Input or Output
	SERIAL	MCS_N/D5	Hi-Z without pull	Hi-Z with Weak Pullup		Input with Weak Pullup	-	MCS_N	Input or Output

附录 B 比特流文件格式

表 B-1 比特流文件格式

<table>
<tr><th>行号</th><th>名称</th><th>内容</th><th>说明</th><th>备注</th></tr>
<tr><td>1~N</td><td>Comments</td><td>包含 EDA 版本、器件信息等</td><td>头部注释信息</td><td></td></tr>
<tr><td>N+1</td><td>1111......1111</td><td>160 bits of ones</td><td>用于兼容 MSPI 时序</td><td></td></tr>
<tr><td>N+2</td><td>Header</td><td>FFFF</td><td>2 bytes of dummy</td><td>-</td></tr>
<tr><td rowspan="2">N+3</td><td rowspan="2">Alignment</td><td>A5C3(Encryption=OFF)</td><td rowspan="2">2 bytes of alignment。文件开始标志</td><td rowspan="2">-</td></tr>
<tr><td>A5CB(Encryption=ON)</td></tr>
<tr><td>N+4</td><td>Verify Device ID</td><td>86000000 + IDCODE</td><td>4 bytes of command + 4 bytes of Device information</td><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td rowspan="2">N+5</td><td rowspan="2">Download Speed</td><td rowspan="2">8 bytes</td><td rowspan="2">-</td><td>默认低位全 0；压缩时，从低位数第 14 位变为 1</td></tr>
<tr><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td rowspan="2">N+6</td><td rowspan="2">write compress symbol</td><td>D100FFFF FFFFFFFF(Compress=OFF)</td><td rowspan="2">4 bytes of command + 4 bytes of compressed symbol。
加下划线数字表示压缩 key，根据 bit 文件内容确定</td><td rowspan="2">有 CRC 校验时，首位变为 0</td></tr>
<tr><td>D100FFFF FF<u>07090A</u>(Compress=ON)</td></tr>
<tr><td rowspan="2">N+7</td><td rowspan="2">Secutiry bit</td><td>8B 或 0B(SecurityBit=ON)</td><td>4 bytes of secutity</td><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td>不存在(SecurityBit=OFF)</td><td>-</td><td>-</td></tr>
<tr><td>N+8</td><td>Write SPI address</td><td>D200FFFF ********</td><td>4 bytes of command + 4 bytes of new SPI address</td><td>用于 multiboot 功能，表示下一次的加载地址</td></tr>
<tr><td>N+9</td><td>Initialize Address</td><td>9200 0000</td><td>4 bytes of reset address</td><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td rowspan="2">N+10</td><td rowspan="2">Write Device</td><td rowspan="2">BB ******</td><td rowspan="2">1 byte of command and 3 bytes of total address</td><td>不要在此行写入 CRC</td></tr>
<tr><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td></td><td>fusemap：begin~end</td><td>fuse 内容</td><td>-</td><td>-</td></tr>
<tr><td rowspan="2">end+1</td><td rowspan="2">CRC</td><td>FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 1(CRC Check=OFF)</td><td rowspan="2">-</td><td rowspan="2">-</td></tr>
<tr><td>FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFF9A39 1(CRC Check=ON)</td></tr>
<tr><td>end+2</td><td>Write User Code</td><td>8A000000 ********</td><td>4 bytes of command + 4 bytes of user code</td><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td>end+3</td><td>SED CRC</td><td>FFFFFFFF FFFFFFFF</td><td>8 bytes reserved for SED CRC</td><td>4 bytes of command + 4 bytes of CRC data, soft error detection</td></tr>
<tr><td>end+4</td><td>Write Done</td><td>8800 0000</td><td>4 bytes of command and Done information</td><td>有 CRC 校验时，首位变为 0</td></tr>
<tr><td>end+5</td><td>NOOP</td><td>FFFF FFFF</td><td>8 bytes of NOOP</td><td>-</td></tr>
<tr><td>end+6</td><td>Device End</td><td>FFFF</td><td>2 bytes of Device ending</td><td>-</td></tr>
</table>

GOWIN高云

智慧逻辑 定制未来