Gowin APB I2C IP

用户指南

IPUG1213-1.0, 2025/04/30

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
2025/04/30	1.0	初始版本。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

Gowin APB I2C IP 用户指南主要包含功能特性、结构框图、功能描述、信号描述、参数描述、界面配置、编程模型、编程序列、参考设计等内容，旨在帮助用户快速了解 APB I2C IP 的特性和使用方法。本手册中的软件界面截图参考的是 1.9.11.02 (64-bit) 版本，因软件版本升级，部分信息可能会略有差异，具体以用户软件版本的信息为准。

1.2 相关文档

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

- DS100，GW1N 系列 FPGA 产品数据手册
- DS117，GW1NR 系列 FPGA 产品数据手册
- DS821，GW1NS 系列 FPGA 产品数据手册
- DS861，GW1NSR 系列 FPGA 产品数据手册
- DS841，GW1NZ 系列 FPGA 产品数据手册
- DS961，GW2ANR 系列 FPGA 产品数据手册
- DS102，GW2A 系列 FPGA 产品数据手册
- DS226，GW2AR 系列 FPGA 产品数据手册
- DS971，GW2AN-18X &9X 器件数据手册
- DS976，GW2AN-55 器件数据手册
- DS981，GW5AT 系列 FPGA 产品数据手册
- DS1103，GW5A 系列 FPGA 产品数据手册
- DS1239，GW5AST 系列 FPGA 产品数据手册
- DS1105，GW5AS 系列 FPGA 产品数据手册
- DS1108, GW5AR 系列 FPGA 产品数据手册

- DS1118，GW5ART 系列 FPGA 产品数据手册
- SUG100，Gowin 云源软件用户指南

1.3 术语、缩略语

本手册中出现的相关术语、缩略语及相关释义如表 1-1 所示。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
AMBA	Advanced Microcontroller Bus Architecture	高级微控制器总线架构
APB	Advanced Peripheral Bus	高级外围总线
DMA	Direct Memory Access	直接内存访问
FIFO	First In First Out	先进先出队列
FPGA	Field Programmable Gate Array	现场可编程门阵列
I2C	Inter-Integrated Circuit Bus	内部集成电路总线
IP	Intellectual Property	知识产权
TPM	Timing Parameter Multiplier	时序参数倍增

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，可直接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 概述

2.1 介绍

Gowin APB I2C IP 是一个 I2C 控制器，支持主机模式和从机模式。

表 2-1 Gowin APB I2C IP 概述

Gowin APB I2C IP	
逻辑资源	参见表 2-2
交付文件	
设计文件	Verilog(encrypted)
参考设计	Verilog
TestBench	Verilog
测试设计流程	
综合软件	GowinSynthesis
应用软件	Gowin Software（V1.9.11.02 及以上）

注！

可登录高云半导体网站查看芯片支持信息。

2.2 功能特性

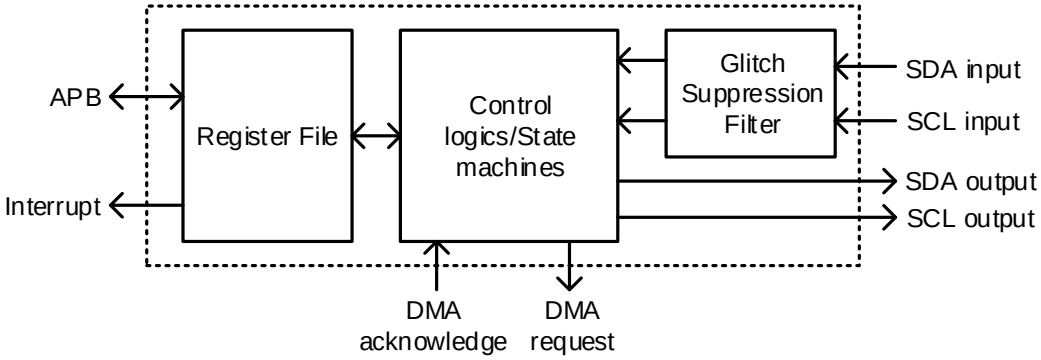
Gowin APB I2C IP 主要包含以下功能特性：

- 支持 AMBA 2.0 APB 总线协议规范
- 支持标准模式（100 Kb/s）、快速模式（400 Kb/s）和快速模式增强版（1 Mb/s）协议
- 可编程的主机/从机模式
- 支持 7 位和 10 位地址模式
- 支持通用呼叫地址
- 自动时钟拉伸
- 可编程的时钟/数据时序
- 支持直接内存访问（DMA）

2.3 结构框图

Gowin APB I2C IP 结构框图如图 2-1 所示。

图 2-1 结构框图



2.4 功能描述

Gowin APB I2C 控制器可以根据控制寄存器的设置，作为 I2C 主设备或 I2C 从设备而运行。

2.4.1 I2C 主设备

作为 I2C 主设备，该控制器提供了一种高效的方式来启动 I2C 事务。每个事务可以分为四个阶段：起始、地址、数据和停止。在起始阶段，生成一个 START 条件。在地址阶段，发送一个地址。在数据阶段，传输一个或多个数据字节。在停止阶段，生成一个 STOP 条件。每个存在的阶段可以被独立控制。

2.4.2 I2C 从设备

作为 I2C 从设备，当 I2C 事务的地址字节与地址寄存器匹配时，控制器被寻址。可以生成一个地址命中中断，以便软件准备进行后续操作。

2.4.3 通用呼叫地址

通用呼叫地址是一个特殊的地址，用于寻址 I2C 总线上的所有从设备。处于从模式的 APB I2C 控制器将使用一个 ACK 信号来响应通用呼叫地址，并设置状态寄存器的 GenCall 字段。

2.4.4 自动时钟拉伸

当软件未准备好接收下一个数据字节或 FIFO 满时，APB I2C 可以通过在 I2C 总线上拉伸时钟来自动暂停总线事务。在主模式和从模式下，自动时钟拉伸都被支持。

2.4.5 自动应答

通过自动应答功能，APB I2C 可对接收到的每个字节自动生成相应的应答信号。根据 I2C 总线协议，除最后一个字节需响应 NACK 外，所有接收字节均会响应 ACK。若需由软件逐字节控制应答状态，可通过使能字节接收中断（Byte Receive Interrupt）来关闭自动应答功能。

2.4.6 时序参数倍增

Gowin APB I2C 提供 I2C 时序参数倍增寄存器，使其在较高的 APB 时钟频率下仍能满足 I2C 总线的时序要求。设置寄存器中所有时序参数的实际值均会被乘以（TPM+1）的系数。

2.5 资源使用

通过 Verilog 语言实现 Gowin APB I2C IP。因使用器件的密度、速度和等级不同，其性能和资源使用情况可能不同。以高云 GW5AT 系列 FPGA 为例，APB I2C IP 资源使用情况如表 2-2 所示。

表 2-2 资源使用情况

器件系列	资源	资源使用	配置
GW5AT	Logic	605	Data FIFO Depth: 4Byte Enable DMA
	Register	333	
	BSRAM	0	

3 信号描述

Gowin APB I2C IP 的信号描述如表 3-1 所示。

表 3-1 信号描述

信号	方向	位宽	描述
i2c_int	output	1	Interrupt signal
paddr	input	[5:2]	APB address bus
penable	input	1	APB enable signal
prdata	output	[31:0]	APB read data bus
psel	input	1	APB slave select signal from the APB decoder
pwdata	input	[31:0]	APB write data bus
pwrite	input	1	APB transfer direction signal This signal indicates a write access when driven as HIGH and a read access when driven as LOW
pready	output	1	APB ready signal
pslverr	output	1	APB slave error signal
pclk	input	1	APB clock
presetn	input	1	APB reset signal, active low
scl_o	output	1	I2C serial clock output
sda_o	output	1	I2C serial data output
scl_i	input	1	I2C serial clock input
sda_i	input	1	I2C serial data input
dma_ack	input	1	DMA acknowledge
dma_req	out	1	DMA request

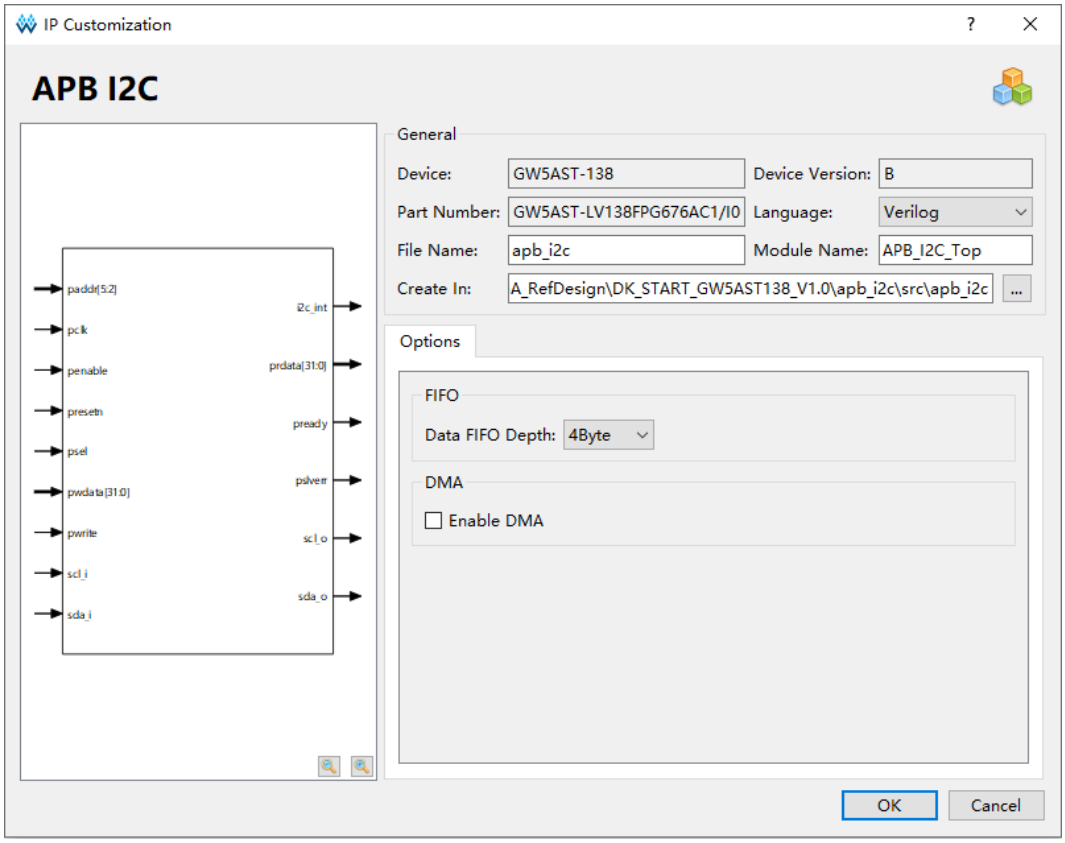
4 界面配置

用户可以在高云半导体云源软件的 IP Core Generator 工具，调用并配置 APB I2C IP。

选择菜单栏“Tool > IP Core Generator”或工具栏“”，打开 IP Core Generator，IP 列表中选择 Soft IP Core > Microprocessor System > Peripheral > APB I2C 1.0。

Gowin APB I2C IP 界面配置，如图 4-1 所示。

图 4-1 界面配置



Gowin APB I2C IP 选项配置描述如下：

- Data FIFO Depth：数据 FIFO 的大小，取值范围为 2、4、8、

16Byte，默认值为 4Byte。Gowin APB I2C 控制器采用 FIFO 作为 I2C 总线的缓冲区，待发送或接收到的数据均保存在 FIFO 中。

- Enable DMA：Gowin Enable DMA 是否开启 DMA 功能，可以勾选或不勾选，默认不勾选。

5 编程模型

5.1 寄存器

5.1.1 寄存器概述

Gowin APB I2C 的寄存器定义如表 5-1 所示。Gowin APB I2C 寄存器定义，位于…\lib\driver\apb_i2c.h。

表 5-1 寄存器定义

地址偏移	名称	描述
0x00	IDREV	ID 和修订寄存器
0x04~0x0C	-	保留
0x10	CFG	配置寄存器
0x14	INTEN	中断使能寄存器
0x18	STATUS	状态寄存器
0x1C	ADDR	地址寄存器
0x20	DATA	数据寄存器
0x24	CTRL	控制寄存器
0x28	CMD	命令寄存器
0x2C	SETUP	设置寄存器
0x30	TPM	时序参数倍增寄存器

以下各节详细描述 APB I2C 寄存器定义。当 I2C 控制器作为主机模式或从机模式时，寄存器定义有所不同，将分别描述主机模式或从机模式的寄存器定义。

寄存器类型缩略语概括如下：

- RO：Read-only
- WO：Write-only (read as zero)
- R/W：Readable and writable
- R/W1C：Readable and Write 1 to clear

5.1.2 ID 和修订寄存器（0x00）

ID 和修订寄存器，用于保存 ID 和修订编号，初始值依赖于所用版本。

ID 和修订寄存器定义，如表 5-2 所示。

表 5-2 ID 和修订寄存器

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for I2C	0x000006
RevMajor	7:4	RO	Major revision number	Revision Dependent
RevMinor	3:0	RO	Minor revision number	Revision Dependent

5.1.3 配置寄存器（0x10）

配置寄存器，用于保存数据 FIFO 的大小。

配置寄存器定义如表 5-3 所示。

表 5-3 配置寄存器

Name	Bit	Type	Description	Reset
-	31:2	-	Reserved	-
FIFOSize	1:0	RO	FIFO size: 0: 2 bytes 1: 4 bytes 2: 8 bytes 3: 16 bytes	Configuration Dependent

5.1.4 中断使能寄存器（0x14）

中断使能寄存器，用于开启或关闭中断。

中断使能寄存器定义如表 5-4 所示。

表 5-4 中断使能寄存器

Name	Bit	Type	Description	Reset
-	31:10	-	Reserved	-
Cmpl	9	R/W	Set to enable the Completion Interrupt. Master: interrupts when a transaction is issued from this master and completed without losing the bus arbitration. Slave: interrupts when a transaction addressing the controller is completed.	0x0
ByteRecv	8	R/W	Set to enable the Byte Receive Interrupt. Interrupts when a byte of data is received. Auto-ACK will be disabled if this interrupt is enabled, that is, the software needs to ACK/NACK the received byte manually.	0x0
ByteTrans	7	R/W	Set to enable the Byte Transmit Interrupt. Interrupts when a byte of data is transmitted.	0x0
Start	6	R/W	Set to enable the START Condition Interrupt. Interrupts when a START condition/repeated START condition is detected.	0x0

Name	Bit	Type	Description	Reset
Stop	5	R/W	Set to enable the STOP Condition Interrupt. Interrupts when a STOP condition is detected.	0x0
ArbLose	4	R/W	Set to enable the Arbitration Lose Interrupt. Master: interrupts when the controller loses the bus arbitration. Slave: not available in this mode.	0x0
AddrHit	3	R/W	Set to enable the Address Hit Interrupt. Master: interrupts when the addressed slave returned an ACK. Slave: interrupts when the controller is addressed.	0x0
FIFOHalf	2	R/W	Set to enable the FIFO Half Interrupt. Receiver: Interrupts when the FIFO is half-full, i.e. there is ≥ 1/2 entries in the FIFO. Transmitter: Interrupts when the FIFO is half-empty, i.e. there is ≤ 1/2 entries in the FIFO. This interrupt depends on the transaction direction; don't enable this interrupt unless the transfer direction is determined, otherwise unintended interrupts may be triggered.	0x0
FIFOFull	1	R/W	Set to enable the FIFO Full Interrupt. Interrupts when the FIFO is full.	0x0
FIFOEmpty	0	R/W	Set to enable the FIFO Empty Interrupt. Interrupts when FIFO is empty.	0x0

5.1.5 状态寄存器（0x18）

状态寄存器，用于保存中断状态和 I2C 总线状态。

状态寄存器定义如表 5-5 所示。

表 5-5 状态寄存器

Name	Bit	Type	Description	Reset
-	31:15	-	Reserved	-
LineSDA	14	RO	Indicates the current status of the SDA line on the bus. 1: High 0: Low	SDA line status
LineSCL	13	RO	Indicates the current status of the SCL line on the bus. 1: High 0: Low	SCL line status
GenCall	12	RO	Indicates that the address of the current transaction is a general call address. This status is only valid in slave mode. 1: General call 0: Not general call	0x0

Name	Bit	Type	Description	Reset
BusBusy	11	RO	Indicates that the bus is busy. The bus is busy when a START condition is on bus and it ends when a STOP condition is seen on bus. 1: Busy 0: Not busy	0x0
ACK	10	RO	Indicates the type of the last received/transmitted acknowledgement bit. 1: ACK 0: NACK	0x0
Cmpl	9	R/W1C	Transaction Completion Master: Indicates that a transaction has been issued from this master and completed without losing the bus arbitration. Slave: Indicates that a transaction addressing the controller has been completed. This status bit must be cleared to receive the next transaction; otherwise, the next incoming transaction will be locked.	0x0
ByteRecv	8	R/W1C	Indicates that a byte of data has been received.	0x0
ByteTrans	7	R/W1C	Indicates that a byte of data has been transmitted.	0x0
Start	6	R/W1C	Indicates that a START Condition or a repeated START condition has been transmitted/received.	0x0
Stop	5	R/W1C	Indicates that a STOP Condition has been transmitted/received.	0x0
ArbLose	4	R/W1C	Indicates that the controller has lost the bus arbitration (master mode only).	0x0
AddrHit	3	R/W1C	Master: indicates that a slave has responded to the transaction. Slave: indicates that a transaction is targeting the controller (including the General Call).	0x0
FIFOHalf	2	RO	Transmitter: Indicates that the FIFO is half-empty. Receiver: Indicates that the FIFO is half-full.	0x0
FIFOFull	1	RO	Indicates that the FIFO is full.	0x0
FIFOEmpty	0	RO	Indicates that the FIFO is empty.	0x1

5.1.6 地址寄存器（0x1C）

地址寄存器，用于保存从机地址。当 I2C 配置作为主机模式时，这是下一个事务的目标从机地址；当 I2C 配置作为从机模式时，这是总线上控制器的地址。

地址寄存器定义如表 5-6 所示。

表 5-6 地址寄存器

Name	Bit	Type	Description	Reset
-	31:10	-	Reserved	-
Addr	9:0	R/W	The slave address For 7-bit addressing mode, the most significant 3 bits are ignored and only the least-significant 7 bits of Addr are valid.	0x0

5.1.7 数据寄存器（0x20）

数据寄存器，用于 FIFO 的数据访问端口。

数据寄存器定义如表 5-7 所示。

表 5-7 数据寄存器

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	-
Data	7:0	R/W	Write this register to put one byte of data to the FIFO. Read this register to get one byte of data from the FIFO.	0x0

5.1.8 控制寄存器（0x24）

控制寄存器，用于控制事务的阶段选择，以及记录数据阶段的进程。

两种寄存器定义如表 5-8 所示。

表 5-8 控制寄存器

Name	Bit	Type	Description	Reset
-	31:13	-	Reserved	-
Phase_start	12	R/W	Enable this bit to send a START condition at the beginning of transaction. Master mode only.	0x1
Phase_addr	11	R/W	Enable this bit to send the address after START condition. Master mode only.	0x1
Phase_data	10	R/W	Enable this bit to send the data after Address phase. Master mode only.	0x1
Phase_stop	9	R/W	Enable this bit to send a STOP condition at the end of a transaction. Master mode only.	0x1
Dir	8	R/W	Transaction direction	0x0

Name	Bit	Type	Description	Reset
			Master: Set this bit to determine the direction for the next transaction. 0: Transmitter 1: Receiver Slave: The direction of the last received transaction. 0: Receiver 1: Transmitter	
DataCnt	7:0	R/W	Data counts in bytes. Master: The number of bytes to transmit/receive. 0 means 256 bytes. DataCnt will be decreased by one for each byte transmitted/received. Slave: the meaning of DataCnt depends on the DMA mode: If DMA is not enabled, DataCnt is the number of bytes transmitted/received from the bus master. It is reset to 0 when the controller is addressed and then increased by one for each byte of data transmitted/received. If DMA is enabled, DataCnt is the number of bytes to transmit/receive. It will not be reset to 0 when the slave is addressed and it will be decreased by one for each byte of data transmitted/received.	0x0

5.1.9 命令寄存器（0x28）

命令寄存器定义如表 5-9 所示。

表 5-9 命令寄存器

Name	Bit	Type	Description	Reset
-	31:3	-	Reserved	-
CMD	2:0	R/W	Write this register with the following values to perform the corresponding actions: 0: no action 1: issue a data transaction (Master only) 2: respond with an ACK to the received byte 3: respond with a NACK to the received byte 4: clear the FIFO 5: reset the I2C controller (abort current transaction, set the SDA and SCL line to the open-drain mode, reset the Status Register and the Interrupt Enable Register, and empty the FIFO) When issuing a data transaction by writing 0x1 to this register, the CMD field stays at 0x1 for the duration of the entire transaction, and it is only cleared to 0x0 after when the transaction has completed or when the controller loses the arbitration.	0x0

Name	Bit	Type	Description	Reset
			Note! No transaction will be issued by the controller when all phases (Start, Address, Data and Stop) are disabled.	

5.1.10 设置寄存器（0x2C）

设置寄存器，用于保存可编程配置和 I2C 总线的时序参数。

设置寄存器定义如表 5-10 所示。

表 5-10 设置寄存器

Name	Bit	Type	Description	Reset
-	31:29	-	Reserved	-
T_SUDAT	28:24	R/W	T_SUDAT defines the data setup time before releasing the SCL. Setup time = (2 * t_{pclk}) + (2 + T_SP + T_SUDAT) * t_{pclk} * (TPM + 1) t_{pclk} = PCLK period TPM = The multiplier value in Timing Parameter Multiplier Register	0x5
T_SP	23:21	R/W	T_SP defines the pulse width of spikes that must be suppressed by the input filter. Pulse width = T_SP * t_{pclk} * (TPM + 1)	0x1
T_HDDAT	20:16	R/W	T_HDDAT defines the data hold time after SCL goes LOW Hold time = (2 * t_{pclk}) + (2 + T_SP + T_HDDAT) * t_{pclk} * (TPM + 1)	0x5
-	15:14	-	Reserved	-
T_SCLRatio	13	R/W	The LOW period of the generated SCL clock is defined by the combination of T_SCLRatio and T_SCLHi values. When T_SCLRatio = 0, the LOW period is equal to HIGH period. When T_SCLRatio = 1, the LOW period is roughly two times of HIGH period. SCL LOW period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) 1: ratio = 2 0: ratio = 1 This field is only valid when the controller is in the master mode.	0x1
T_SCLHi	12:4	R/W	The HIGH period of generated SCL clock is defined by T_SCLHi. SCL HIGH period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) The T_SCLHi value must be greater than T_SP and T_HDDAT values. This field is only valid when the controller is in the master mode.	0x10

Name	Bit	Type	Description	Reset
DMAEn	3	R/W	Enable the direct memory access mode data transfer. 1: Enable 0: Disable	0x0
Master	2	R/W	Configure this device as a master or a slave 1: Master mode 0: Slave mode	0x0
Addressing	1	R/W	I2C addressing mode: 1: 10-bit addressing mode 0: 7-bit addressing mode	0x0
IICEn	0	R/W	Enable the I2C controller 1: Enable 0: Disable	0x0

5.1.11 时序参数倍增寄存器（0x30）

时序参数倍增寄存器，用于保存一个倍增数值，放大定义于设置寄存器中的 I2C 总线时序参数。当控制器运行于较高的 APB 时钟频率时，该倍增数值有助于控制器满足 I2C 总线接口的时序要求。

时序参数倍增寄存器定义如表 5-11 所示。

表 5-11 时序参数倍增寄存器

Name	Bit	Type	Description	Reset
-	31:5	-	Reserved	-
TPM	4:0	R/W	A multiplication value for I2C timing parameters. All the timing parameters in the Setup Register are multiplied by (TPM + 1).	0x0

5.2 驱动函数

5.2.1 驱动函数概述

Gowin APB I2C 驱动函数定义如表 5-12 所示。Gowin APB I2C 驱动函数定义，位于…\lib\driver\apb_i2c_driver.h 和 apb_i2c_driver.c。

表 5-12 驱动函数定义

驱动函数	描述
apb_i2cx_get_capabilities	获取 APB I2C 驱动的功能信息
apb_i2cx_initialize	初始化 APB I2C 接口
apb_i2cx_uninitialize	卸载 APB I2C 接口
apb_i2cx_power_control	指定 APB I2C 接口的功耗模式
apb_i2cx_master_transmit	APB I2C 作为主机模式发送数据
apb_i2cx_master_receive	APB I2C 作为主机模式接收数据
apb_i2cx_slave_transmit	APB I2C 作为从机模式发送数据
apb_i2cx_slave_receive	APB I2C 作为从机模式接收数据

驱动函数	描述
apb_i2cx_get_datacount	获取 APB I2C 传输数据的数量
apb_i2cx_control	配置 APB I2C 接口的设置，执行指定的操作
apb_i2cx_getstatus	获取 APB I2C 接口的状态
apb_i2cx_irq_handler	APB I2C 中断处理程序
apb_i2cx_dma_tx_event	APB I2C DMA 发送事件处理
apb_i2cx_dma_rx_event	APB I2C DMA 接收事件处理

以下各节详细描述 APB I2C 的驱动函数定义。

5.2.2 apb_i2cx_get_capabilities

apb_i2cx_get_capabilities 函数定义如表 5-13 所示。

表 5-13 apb_i2cx_get_capabilities 函数定义

原型	APB_I2C_CAPABILITIES apb_i2cx_get_capabilities(APB_I2C_RESOURCES* i2c)
描述	获取 APB I2C 驱动的功能信息
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	APB I2C 驱动的功能信息

5.2.3 apb_i2cx_initialize

apb_i2cx_initialize 函数定义如表 5-14 所示。

表 5-14 apb_i2cx_initialize 函数定义

原型	int apb_i2cx_initialize(APB_I2C_SignalEvent_t cb_event, APB_I2C_RESOURCES* i2c)
描述	初始化 APB I2C 接口
参数	cb_event：指向 APB_I2C_SignalEvent 回调函数的指针 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.4 apb_i2cx_uninitialize

apb_i2cx_uninitialize 函数定义如表 5-15 所示。

表 5-15 apb_i2cx_uninitialize 函数定义

原型	int apb_i2cx_uninitialize(APB_I2C_RESOURCES* i2c)
描述	卸载 APB I2C 接口
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.5 apb_i2cx_power_control

apb_i2cx_power_control 函数定义如表 5-16 所示。

表 5-16 apb_i2cx_power_control 函数定义

原型	int apb_i2cx_power_control(APB_I2C_POWER_STATE state, APB_I2C_RESOURCES* i2c)
描述	指定 APB I2C 接口的功耗模式
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针 state：APB I2C 接口功耗模式，包括： APB_I2C_POWER_FULL：to set up peripherals for data transfers, enable interrupts and DMA APB_I2C_POWER_LOW：to enable power-saving APB_I2C_POWER_OFF：to terminate pending data transfers and disable peripherals, related interrupts and DMA
返回值	如果发生执行错误，返回一个负值

5.2.6 apb_i2cx_master_transmit

apb_i2cx_master_transmit 函数定义如表 5-17 所示。

表 5-17 apb_i2cx_master_transmit 函数定义

原型	int apb_i2cx_master_transmit(unsigned int addr, const unsigned char* data, unsigned int num, bool xfer_pending, APB_I2C_RESOURCES* i2c)
描述	APB I2C 作为主机模式发送数据
参数	addr：从机地址 data：指向发送数据缓存区的指针 num：发送数据的长度 xfer_pending：传输最后阶段设置是否产生停止状态 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.7 apb_i2cx_master_receive

apb_i2cx_master_receive 函数定义如表 5-18 所示。

表 5-18 apb_i2cx_master_receive 函数定义

原型	int apb_i2cx_master_receive(unsigned int addr, unsigned char* data, unsigned int num, bool xfer_pending, APB_I2C_RESOURCES* i2c)
描述	APB I2C 作为主机模式接收数据
参数	addr：从机地址 data：指向接收数据缓存区的指针 num：接收数据的长度 xfer_pending：传输最后阶段设置是否产生停止状态 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.8 apb_i2cx_slave_transmit

apb_i2cx_slave_transmit 函数定义如表 5-19 所示。

表 5-19 apb_i2cx_slave_transmit 函数定义

原型	int apb_i2cx_slave_transmit(const unsigned char* data, unsigned int num, APB_I2C_RESOURCES* i2c)
描述	APB I2C 作为从机模式发送数据
参数	data：指向发送到主机的数据缓存区的指针 num：发送数据的长度 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.9 apb_i2cx_slave_receive

apb_i2cx_slave_receive 函数定义如表 5-20 所示。

表 5-20 apb_i2cx_slave_receive 函数定

原型	int apb_i2cx_slave_receive(unsigned char* data, unsigned int num, APB_I2C_RESOURCES* i2c)
描述	APB I2C 作为从机模式接收数据
参数	data：指向接收数据缓存区的指针 num：接收数据的长度 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.10 apb_i2cx_get_datacount

apb_i2cx_get_datacount 函数定义如表 5-21 所示。

表 5-21 apb_i2cx_get_datacount 函数定义

原型	unsigned int apb_i2cx_get_datacount(APB_I2C_RESOURCES* i2c)
描述	获取 APB I2C 传输数据的数量
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	APB I2C 接口传输的数据长度

5.2.11 apb_i2cx_control

apb_i2cx_control 函数定义如表 5-22 所示。

表 5-22 apb_i2cx_control 函数定义

原型	int apb_i2cx_control(unsigned int control, unsigned int arg0, unsigned int arg1, APB_I2C_RESOURCES* i2c)
描述	配置 APB I2C 接口的设置，执行指定的操作
参数	control：APB I2C 驱动接口的一种设置或执行的一种操作 arg0：指定设置或操作的附加信息 arg1：指向设置或操作的附加信息 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

“control”和“arg”设置与操作，如表 5-23 所示。

表 5-23 Control Settings or Operations

Options for control	arg sprcifies	Settings or operations
APB_I2C_OWN_ADDRESS	slave address	Sets the slave address
APB_I2C_BUS_SPEED	bus speed	Sets the bus speed
APB_I2C_BUS_CLEAR		Clears the bus by sending nine clock pulses
APB_I2C_ABORT_TRANSFER		Aborts the data transfer between the master and slave

5.2.12 apb_i2cx_getstatus

apb_i2cx_getstatus 函数定义如表 5-24 所示。

表 5-24 apb_i2cx_getstatus 函数定义

原型	APB_I2C_STATUS apb_i2cx_getstatus(APB_I2C_RESOURCES* i2c)
描述	获取 APB I2C 接口的状态
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	APB I2C 接口的当前状态

5.2.13 apb_i2cx_irq_handler

apb_i2cx_irq_handler 函数定义如表 5-25 所示。

表 5-25 apb_i2cx_irq_handler 函数定义

原型	void apb_i2cx_irq_handler(APB_I2C_RESOURCES* i2c)
描述	APB I2C 中断处理程序
参数	i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	无

5.2.14 apb_i2cx_dma_tx_event

apb_i2cx_dma_tx_event 函数定义如表 5-26 所示。

表 5-26 apb_i2cx_dma_tx_event 函数定义

原型	void apb_i2cx_dma_tx_event (unsigned int event, APB_I2C_RESOURCES* i2c)
描述	APB I2C DMA 发送事件处理
参数	event：DMA 发送事件 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	无

5.2.15 apb_i2cx_dma_rx_event

apb_i2cx_dma_rx_event 函数定义如表 5-27 所示。

表 5-27 apb_i2cx_dma_rx_event 函数定义

原型	void apb_i2cx_dma_rx_event (unsigned int event, APB_I2C_RESOURCES* i2c)
描述	APB I2C DMA 接收事件处理
参数	event：DMA 接收事件 i2c：指向 APB_I2C_RESOURCES 结构体的指针
返回值	无

6 编程序列

6.1 时序设置指南

在启用 APB I2C 控制器之前，必须完成以下操作：

- 通过编程时序参数倍增寄存器，设置时序参数倍增系数
- 通过编程设置寄存器，设置 I2C 总线的时序参数

作为 I2C 从设备，必须根据 APB 时钟频率和 I2C 总线速度，正确设置尖峰抑制宽度、数据建立时间和数据保持时间。而作为 I2C 主设备，则还需要编程 I2C 总线时钟频率。

默认情况下，I2C 总线接口的时序是由设置寄存器中的时序参数（以 APB 时钟周期为单位）来规定的。由于时序参数的位宽受限，如果 APB 时钟频率过高，所设定的时序参数可能无法满足 I2C 总线接口所需的时序要求。在这种情况下，可以通过设置时序参数倍增寄存器来扩大时序参数的实际数值，从而扩展设置寄存器中时序参数的有效范围，使其落入有效的参数值范围内。时序参数倍增寄存器应在 I2C 总线空闲且 APB I2C 处于禁用状态（即 Setup.IICEn=0）时进行编程。

以下各节描述了如何配置设置寄存器，以满足 I2C 总线的时序参数要求。所有示例均假定 APB 时钟频率为 40MHz，即 APB 时钟周期为 25ns，如果用户设计中的 APB 时钟频率不是 40MHz，请按照如下方法设置寄存器。

6.1.1 尖峰抑制宽度

表 6-1 描述了通过输入滤波器必须抑制的尖峰脉冲宽度。

对于快速模式和超快速模式增强版，必须抑制小于 50ns 的尖峰（假设 TPM == 0）。即：

T_SP = 50ns / (25ns * (TPM + 1)) = 2

表 6-1 Timing Parameters for Spike Suppression

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{SP}	Pulse width of spikes that must be suppressed by the input filter.	-	-	0	50	0	50	ns

6.1.2 数据建立时间

数据建立时间，定义了在 SCL 上升沿之前，SDA 应该保持稳定的时间。表 6-2 描述了数据建立时间的时序参数。表 5-10 中描述的数据建立时间等式，如下所示：

Setup time = (2 * t_{pclk}) + (2 + T_SP + T_SUDAT) * t_{pclk} * (TPM + 1)

例如标准模式（假设 TPM == 0），

250ns = 50ns + (2 + 2 + T_SUDAT) * 25ns

计算得出，

T_SUDAT = 4

对于其他模式，T_SUDAT 的计算方法类似。

表 6-2 Timing Parameters for the Data Setup Time

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{SUDAT}	Data setup time	250	-	100	-	50	-	ns

6.1.3 数据保持时间

数据保持时间，定义了在 SCL 下降沿之后，SDA 应该保持稳定的时间。表 6-3 描述了数据保持时间的时序参数。表 5-10 中描述的数据保持时间等式，如下所示：

Hold time = (2 * t_{pclk}) + (2 + T_SP + T_HDDAT) * t_{pclk} * (TPM + 1)

例如标准模式（假设 TPM == 0），

300ns = 50ns + (2 + 2 + T_HDDAT) * 25ns

计算得出，

T_HDDAT = 6

对于其他模式，T_HDDAT 的计算方法类似。

表 6-3 Timing Parameters for the Data Hold Time

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{HDDAT}	Data hold time	300	-	300	-	0	-	ns

6.1.4 I2C 总线时钟频率

I2C 总线时钟频率，通过参数 t_{HIGH} 和 t_{LOW} 定义，表 6-4 描述了设置寄存器中计算 I2C 总线时钟频率的 T_SCLHi 和 T_SCLRatio。

表 6-4 Timing Parameters for the SCL Clock

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		Unit
		Min	Max	Min	Max	Min	Max	
t_{HIGH}	HIGH period of the SCL clock	4.0	-	0.6	-	0.26	-	us
t_{LOW}	LOW period of the SCL clock	4.7	-	1.3	-	0.5		us

对于标准模式，t_{HIGH} 和 t_{LOW} 的最低需求接近，所以 T_SCLRatio 可以设置为 0 来简化计算。中描述的 SCL 周期等式，如下所示：

SCL HIGH period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 4000ns

SCL LOW period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 4700ns

替换等式 2 的 T_SP，ratio = 1，t_{pclk} = 25ns，则等式简化为（假设 TPM == 0）：

50ns + (2 + 2 + T_SCLHi) * 25ns * (0 + 1) >= 4000ns

50ns + (2 + 2 + T_SCLHi * 1) * 25ns * (0 + 1) >= 4700ns

T_SCLHi >= 182

对于快速模式，t_{LOW} 的最低需求为 t_{HIGH} 的 2 倍，所以 T_SCLRatio 可以设置为 1。SCL 周期等式，如下所示：

SCL HIGH period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 600ns

SCL LOW period = (2 * t_{pclk}) + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 1300ns

替换等式 2 的 T_SP，ratio = 2，t_{pclk} = 25ns，则等式简化为（假设 TPM == 0）：

50ns + (4 + T_SCLHi) * 25ns * (0 + 1) >= 600ns

50ns + (4 + T_SCLHi * 2) * 25ns * (0 + 1) >= 1300ns

T_SCLHi >= 23

对于快速模式增强版，T_SCLHi 可以使用快速模式的计算方法。

6.1.5 时序参数倍增

可以调整时序参数倍增寄存器，以增大设置寄存器中时序参数的有效值。仅当 APB 时钟频率过高，以至于仅靠设置寄存器中时序参数的最大值（即 TPM==0）无法满足所需的 I2C 总线接口时序规范时，才需要进行此调整。

确定满足 I2C 总线时序要求的（非零）TPM 值的步骤如下：

1. 通过将 T_SCLHi 赋予最大值（511）和 T_SP 赋予最小有效值（1）来计算 TPM。为避免与最终的 T_SCLHi 和 T_SP 值混淆，下面的讨论中，这些值分别记作 T_SCLHi'和 T_SP'。
2. 根据上述获得的 TPM 值计算 T_SCLHi 和 T_SP。注意，最终的 T_SCLHi 很可能不会是 511，而是某个小于 511 的值。

例如，考虑在 APB 总线时钟周期为 2ns（t_{pclk} = 2ns）的条件下运行 APB I2C，并满足表 6-4 中描述的标准模式 I2C 总线时序要求，同时附加 50ns 尖峰抑制期的要求。也就是说，t_{SP} = 50ns，t_{HIGH} = 4.0μs，t_{LOW} = 4.7μs。

首先，根据表 5-10 中时序参数的公式：

SCL HIGH period = 2 * t_{pclk} + (2 + T_SP' + T_SCLHi') * t_{pclk} * (TPM + 1) >= 4.0us

SCL LOW period = 2 * t_{pclk} + (2 + T_SP' + T_SCLHi' * ratio) * t_{pclk} * (TPM + 1) >= 4.7us

当 T_SP' = 1，T_SCLHi' = 511，ratio = 1 以及 t_{pclk} = 2ns：

SCL HIGH period = 2 * 2ns + (2 + 1 + 511) * 2ns * (TPM + 1) >= 4.0us

SCL LOW period = 2 * 2ns + (2 + 1 + 511 * 1) * 2ns * (TPM + 1) >= 4.7us

因此，

2 * 2ns + (2 + 1 + 511) * 2ns * (TPM + 1) >= 4000ns

2 * 2ns + (2 + 1 + 511 * 1) * 2ns * (TPM + 1) >= 4700ns

所以：

(TPM + 1) >= 4.57

TPM >= 3.57

计算得出 TPM = 4，以此来计算表 5-10 所示的所有时序参数。

即，计算 T_SP：

T_SP >= 50ns / ((TPM+1) * tpclk) = 50ns / ((4 + 1) * 2ns) = 5

计算 T_SCLHi：

SCL HIGH period = 2 * t_{pclk} + (2 + T_SP + T_SCLHi) * t_{pclk} * (TPM + 1) >= 4000ns

SCL LOW period = 2 * t_{pclk} + (2 + T_SP + T_SCLHi * ratio) * t_{pclk} * (TPM + 1) >= 4700ns

因此，

SCL HIGH period = 2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4000ns

SCL LOW period = 2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4700ns

合并两等式得出：

2 * 2ns + (2 + 5 + T_SCLHi) * 2ns * (4 + 1) >= 4700ns

7 + T_SCLHi >= (4700 – 4) / (2 * 5)

T_SCLHi >= 462.6

因此，对于 APB 总线时钟周期为 2ns，且具有 50ns 尖峰抑制要求的标准模式 I2C 总线，其时序参数为：T_SCLHi = 463，T_SP = 5，TPM = 4。

6.2 主模式

以下示例演示了如何在主模式下发起 I2C 传输。

6.2.1 数据传输（不使用 DMA）

1. 如有必要，通过编程时序参数倍增寄存器来设置时序参数倍增。
2. 通过编程设置寄存器来设置控制器：
   - Master = 1
   - IICEn = 1
   - 以及其他时序参数
3. 在控制寄存器中设置数据计数、传输方向和各阶段的选择：
   - Phase_start = 1
   - Phase_addr = 1
   - Phase_data = 1
   - Phase_stop = 1
   - Dir = 0
   - DataCnt = 数据字节数
4. 将目标从设备的地址写入地址寄存器。
5. 在中断使能寄存器中启用完成中断和 FIFO 空中断：
   - Cmpl = 1
   - FIFOEmpty = 1

6. 向命令寄存器写入 0x1，发起传输。
7. 等待中断：
   - FIFO 空中断：当触发 FIFO 空中断时，通过写数据到数据寄存器将数据推入 FIFO，直到 FIFO 满。如果所有数据都已推入 FIFO，则禁用 FIFO 空中断；否则，重复步骤 6。
   - 完成中断：检查状态寄存器中的 AddrHit 字段，确保目标从设备正确接收了传输。写入 1 到状态寄存器中的 Cmpl 字段，以清除完成状态，然后返回步骤 7。
8. 禁用所有中断，并检查控制寄存器中的 DataCnt 字段，以确认所有数据是否已成功传输。

6.2.2 数据接收（不使用 DMA）

1. 如有必要，通过编程时序参数倍增寄存器来设置时序参数倍增。
2. 通过编程设置寄存器来设置控制器：
   - Master = 1
   - IICEn = 1
   - 以及其他时序参数
3. 在控制寄存器中设置数据计数、传输方向及各阶段选择：
   - Phase_start = 1
   - Phase_addr = 1
   - Phase_data = 1
   - Phase_stop = 1
   - Dir = 1
   - DataCnt = 数据字节数
4. 将目标从设备的地址写入地址寄存器。
5. 在中断使能寄存器中启用完成中断和 FIFO 满中断：
   - Cmpl = 1
   - FIFOFull = 1
6. 向命令寄存器写入 0x1 以发起传输。
7. 等待中断：
   - FIFO 满中断：通过读取数据寄存器从 FIFO 中获取数据，直至 FIFO 清空；然后重复步骤 6。
   - 完成中断：检查状态寄存器中的 AddrHit 字段，以确保目标从设备已正确接收传输。取出 FIFO 中剩余的所有数据，向状态寄存器的 Cmpl 字段写入 1 以清除完成状态，然后返回步骤 7。
8. 禁用所有中断，并检查控制寄存器中的 DataCnt 字段，以确认所有数据

是否已成功接收。

6.2.3 数据传输（使用 DMA）

1. 如有必要，通过设置时序参数倍增寄存器来设置时序参数倍增。
2. 通过设置设置寄存器来设置控制器：
   - Master = 1
   - IICEn = 1
   - DMAEn = 1
   - 以及其他时序参数
3. 设置 DMA 控制器。
4. 在控制寄存器中设置数据计数、传输方向和阶段选择：
   - Phase_start = 1
   - Phase_addr = 1
   - Phase_data = 1
   - Phase_stop = 1
   - Dir = 发送/接收
   - DataCnt = 数据字节数
5. 将目标从设备的地址写入地址寄存器。
6. 在中断使能寄存器中启用完成中断：Cmpl = 1
7. 向命令寄存器写入 0x1，以发起传输。
8. 等待完成中断。检查控制存器中的 DataCnt 字段，以确认所有数据是否已成功传输。

6.2.4 多主机模式的仲裁丢失

在 I2C 总线的多主机环境中，可能会出现多个主设备同时发起传输的情况。为了解决总线争用，I2C 协议定义了一套仲裁机制。在仲裁过程中，只有一个主设备会赢得总线使用权，其他主设备将失去仲裁。当控制器失去仲裁时，会触发仲裁丢失中断。此时，应按照以下步骤重新启动传输：

1. 如果使用了 DMA，应中止 DMA 传输，并通过将设置寄存器的 DMAEn 字段设置为 0 来禁用 I2C 的 DMA 请求，然后向命令寄存器写入 0x4 以清除 FIFO。
2. 通过轮询状态寄存器的 BusBusy 字段，等待总线变为空闲状态。
3. 重新启动传输。

6.3 从模式

以下示例演示了如何在从模式下操作 I2C 控制器。

6.3.1 数据传输（不使用 DMA）

1. 将地址寄存器设置为从设备控制器 I2C 总线地址。
2. 如有必要，通过编程时序参数倍增寄存器来设置时序参数倍增系数。
3. 通过编程设置寄存器来设置控制器：
   - Master = 0
   - IICEn = 1
   - 以及其他时序参数
4. 在中断使能寄存器中启用地址命中中断和完成中断：
   - AddrHit = 1
   - Cmpl = 1
5. 等待地址命中中断：读取控制寄存器的 Dir 字段以确定传输方向。如果需要更多传输信息，可读取状态寄存器，例如，通过 GenCall 字段识别通用呼叫传输。
   - 接收器：启用 FIFO 满中断，然后进入步骤 6。
   - 发送器：启用 FIFO 空中断，然后进入步骤 7。
6. 接收器：
   - 等待 FIFO 满中断，直到完成中断触发。
   - 如果 FIFO 满中断触发：通过读取数据寄存器，来实现从 FIFO 中获取数据，直至 FIFO 为空，然后进入（I）。
7. 发送器：
   - 等待 FIFO 空中断，直到完成中断触发。
   - 如果 FIFO 空中断触发：通过向数据寄存器写入数据，来实现向 FIFO 中推送数据，直至 FIFO 满，然后进入（I）。
8. 在步骤 5 或步骤 6 期间，如果完成中断触发，则清除 FIFO，并进入步骤 8。
9. 检查控制寄存器中的 DataCnt 字段，以确定传输了多少数据。当软件准备好接收下一次传输时，清除状态寄存器中完成中断的状态。

6.3.2 数据传输（使用 DMA）

仅在已知下一次传输的方向和数据计数时，才应使用 DMA。

1. 将地址寄存器设置为控制器的地址。
2. 如有必要，通过编程时序参数倍增寄存器来设置时序参数倍增系数。
3. 通过编程设置寄存器来设置控制器：
   - Master = 0
   - IICEn = 1
   - 以及其他时序参数
4. 在中断使能寄存器中启用地址命中中断和完成中断：
   - AddrHit = 1
   - Cmpl = 1
5. 等待地址命中中断：读取控制寄存器的 Dir 字段以检查传输方向。如果该方向不是预期方向，则退回到不使用 DMA 的正常传输。如需更多传输信息，例如通过 GenCall 字段识别通用呼叫传输，则读取状态寄存器。
6. 配置 DMA 控制器。
7. 在控制寄存器中设置传输的数据计数：
   - DataCnt = 数据字节数
8. 在设置寄存器中设置 DMAEn 位以启用 DMA，并等待完成中断。
9. 当完成中断触发后，检查控制寄存器中的 DataCnt，以确定传输了多少数据。
10. 禁用 DMA。
11. 当软件准备好接收下一次传输时，清除状态寄存器中完成中断的状态。

7 参考设计

详细信息请参见高云半导体网站 Gowin APB I2C IP 相关参考设计：

- 硬件参考设计：

  …\ref_design\FPGA_RefDesign\DK_START_GW5AST138_V1.0\apb_i2c

- 软件参考设计：

  …\ref_design\MCU_RefDesign\apb_i2c

注！

I2C 主机开发板与从机开发板通信时，两板必须共地！


GOWIN高云
智慧逻辑 定制未来