



Gowin goConfig I2C IP

User Guide

IPUG795-2.1E, 09/27/2023

Copyright © 2023 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any denotes, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
03/09/2022	1.0E	Initial version published.
10/17/2022	2.0E	● Supported devices updated. ● The descriptions of GW1N(R)-2C series improved. ● The descriptions of GW2AN-9X series improved.
09/27/2023	2.1E	● Some descriptions improved. ● Chapter 6 Example Description added.

Contents

Contents	i
List of Figures	iii
List of Tables	v
1 About This Guide	1
1.1 Purpose	1
1.2 Related Documents	1
1.3 Terminology and Abbreviations	2
1.4 Support and Feedback	2
2 Function Introduction	3
2.1 Programming Overview	4
2.2 I ² C Interface	4
2.2.1 I ² C Features	5
2.2.2 I ² C Timing Features	5
2.3 IP Features	7
2.4 Resource Utilization	7
3 Functional Description	9
3.1 goConfig I2C Structure Block Diagram	9
3.2 Gowin goConfig I2C IP External Circuit Connection	11
3.3 Gowin goConfig I2C IP Background Programming Commands	11
3.3.1 2C Series	11
3.3.2 9X Series	17
4 Signal Definition	23
4.1 Signal Definition	23
5 Instructions for IP Use	26
5.1 2C Series	26
5.1.1 Software Configuration	26

5.1.2 Flow Description	27
5.1.3 Notes	29
5.2 9X Series	29
5.2.1 Software Configuration	29
5.2.2 Flow Description	31
5.2.3 Notes	32
6 Example Description.....	34
6.1 Example for 2C Series.....	34
6.1.1 Check MCU-IP Communication for Normalcy	34
6.1.2 Overall Process of Background Programming	35
7 Interface Configuration.....	40
8 Reference Design.....	43
8.1 Reference Design	43
8.1.1 FPGA Project Description	43
8.1.2 Reference Design File	44
8.1.3 Notes	45

List of Figures

Figure 2-1 I ² C 7-bit Typical Timing	5
Figure 2-2 I ² C Interface Timing	5
Figure 3-1 Structure Block Diagram (2C Series)	10
Figure 3-2 Structure Block Diagram (9X Series)	10
Figure 3-3 External Resistor Circuit Connection.....	11
Figure 3-4 Example of Read Status Register Command	13
Figure 3-5 Example of Reset IP Command	14
Figure 3-6 Example of Sending Read-back Command	15
Figure 3-7 Example of Read FIFO Command	15
Figure 3-8 Example of Waiting Command	15
Figure 3-9 Example of Sending Data and No Read Back Flash Data Command	16
Figure 3-10 Example of Reconfig Command.....	16
Figure 3-11 IP version Command	16
Figure 3-12 Example of Using CSCR Register.....	18
Figure 3-13 Example of DICN Register Write	18
Figure 3-14 Example of DICN Register Write Read	18
Figure 3-15 Example of DOCN Register Write	19
Figure 3-16 Example of DOCN Register Read	19
Figure 3-17 Example of Using DIFR Register	20
Figure 3-18 Example of Using SRSTR Register.....	20
Figure 3-19 Example of Using SR Register	21
Figure 3-20 Example of Using DOFR Register.....	22
Figure 5-1 Open Configuration Option.....	26
Figure 5-2 Select I2C in Background Programming Option	27
Figure 5-3 Open Configuration Option.....	30
Figure 5-4 Select I2C in Background Programming Option	30
Figure 5-5 I/O Definition Example.....	33
Figure 6-1 Block Diagram	34

Figure 6-2 Flowchart of Checking MCU-IP Communication	35
Figure 6-3 Flowchart of Background Programming	36
Figure 6-4 Flowchart of Erasing Embedded Flash	37
Figure 6-5 Flowchart of Programming Embedded Flash	39
Figure 7-1 IP Core Generator	40
Figure 7-2 Open goConfig I2C IP Core	41
Figure 7-3 goConfig I2C IP Configuration Interface.....	42
Figure 8-1 Reference Design Block Diagram	44

List of Tables

Table 1-1 Terminology and Abbreviations	2
Table 2-1 Gowin goConfig I2C IP Overview	3
Table 2-2 2C Series I ² C Interface Timing	6
Table 2-3 9X Series I ² C Interface Timing	6
Table 2-4 Resource Utilization (2C Series)	7
Table 2-5 Resource Utilization (9X Series)	8
Table 3-1 2C Series Commands	12
Table 3-2 Status Register	13
Table 3-3 Register Definition	17
Table 3-4 CSCR-Data Definition	17
Table 3-5 DICN-Data Definition	18
Table 3-6 DOCN-Data Definition	19
Table 3-7 DIFR-Data Definition	19
Table 3-8 SRSTR-Data Definition	20
Table 3-9 SR-Data Definition	20
Table 3-10 DOFR-Data Definition	22
Table 4-1 I/O Definition for 2C Series	23
Table 4-2 I/O Definition for 9X Series	24
Table 5-1 Contrast Table	32
Table 8-1 File List in src Folder for 2C Series	44
Table 8-2 File List in src Folder for 9X Series	44

1 About This Guide

1.1 Purpose

Gowin goConfig I2C IP User Guide includes functional description, signal definition, usage description, example description, and interface configuration. It can help you quickly learn the features and usage of goConfig I2C IP.

Note!

Gowin goConfig I2C IP has slight differences in functions to match the characteristics of different devices. GW1N(R)-1P5C/GW1N(R)-2C series (hereinafter referred to as 2C series) are in one category and GW2AN-9X/GW2AN-18X series (hereinafter referred to as 9X series) are in another category, and this manual will describe the functions and features separately. If there is no clear distinction between devices in the document, it is stated as a common feature or function.

1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com.

- [DS100, GW1N series of FPGA Products Data Sheet](#)
- [DS117, GW1NR series of FPGA Products Data Sheet](#)
- [DS971, GW2AN-18X & 9X Data Sheet](#)
- [SUG100, Gowin Software User Guide](#)
- [SUG283, Gowin Primitive User Guide](#)
- [UG290, Gowin FPGA Products Programming and Configuration Guide](#)

1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
ACK	Acknowledge
FPGA	Field Programmable Gate Array
I ² C(I2C)	Inter-Integrated Circuit
IP	Intellectual Property
NACK	Not ACKnowledge
SRAM	Static Random Access Memory

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Function Introduction

Background programming is a function that has been in high demand by users in recent years. In the use of Gowin FPGA products, there is "Function A", if you want to program or re-program "Function A" to "Function B", and in this process, "Function A" will not stop working; then the programmed "Function B" can be immediately loaded to the chip, background programming is required.

Gowin goConfig I2C IP can realize the online programming function through I²C port. Users can program the FPGA through I²C port to update the data stored internally, realizing the background programming.

Table 2-1 Gowin goConfig I2C IP Overview

Gowin goConfig I2C IP	
IP Core Application	
Logic Resource	Please refer to Table 2-4 and Table 2-5.
Delivered Doc.	
Design Files	Verilog (encrypted)
Reference Design	Verilog
TestBench	Verilog
Test and Design Flow	
Synthesis Software	GowinSynthesis
Application Software	Gowin Software (V1.9.8.07 and above)

Note!

For the devices supported, you can click [here](#) to get the information.

2.1 Programming Overview

The programming includes two processes: Program Flash and Configure SRAM.

- Program Flash: The process of writing the bitstream file generated by Gowin Software to Flash (hereinafter referred to as Write to Flash).
- Configure SRAM: The process of loading the bitstream data from the Flash into the SRAM in the FPGA (hereinafter referred to as Load to SRAM).

Note!

- Normal programming: When programming, SRAM is erased (i.e., the running function stops) and then write to Flash.
- Background programming: When programming, the SRAM is not erased (i.e., the running function is not affected), and write to Flash directly.
- When configuring, the process of normal programming and background programming is the same.

Here is an example of background programming.

Users generally write Function B to be updated to the Flash through the running Function A; then users can load the Function B in the Flash to SRAM through a series of operations when the function needs to be updated.

Note !

- When Function A is operated according to the background programming, it must have a background programming module. In this manual, i.e. Function A must have Gowin goConfig I2C IP.
- Function B can be operated without background programming module, but it can not perform background programming subsequently.
- Function A will still run normally during Write to Flash, and it will stop running during Load to SRAM.

2.2 I²C Interface

I²C interface communication in Gowin goConfig I2C IP conforms to official I²C 7-bit address protocol; for the details of using IP, see chapter [3 Functional Description](#) and chapter [5 Instructions for IP Use](#).

2.2.1 I²C Features

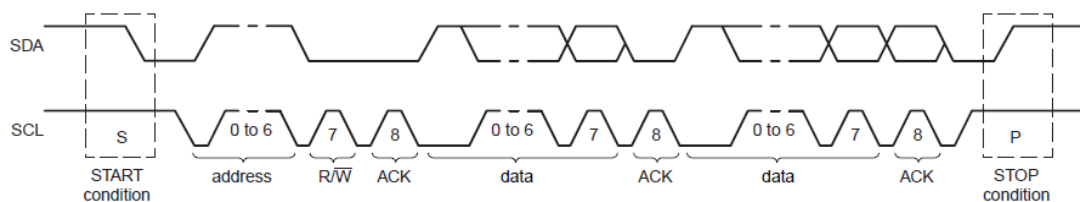
An example of typical timing of I²C 7-bit address protocol is shown in , and the descriptions are shown below.

- S: START condition
- ADDRESS: Slave address, 7 bits.
- R/W: Read and write control bit, 0-> write, 1-> read.
- ACK: Slave acknowledge bit, 0-> ACK, 1-> NACK.
- data: Read/write valid data, 8 bits.
- S: STOP condition

Note!

In case of multiple bit widths, the higher bits are sent first. For example: address=7'h58, the order of sending by bits is 101_1000. data=8'hB1, the order of sending by bits is 1011_0001.

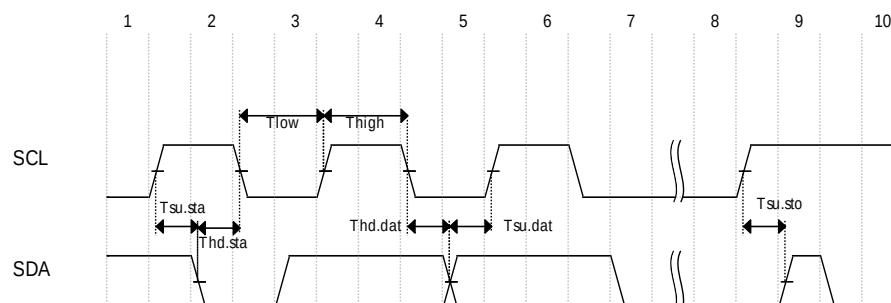
Figure 2-1 I²C 7-bit Typical Timing



2.2.2 I²C Timing Features

Gowin goConfig I2C IP has timing differences between 2C series and 9X series, as shown in Figure 2-2.

Figure 2-2 I²C Interface Timing



2C Series Interface Timing

The 2C series interface timing of Gowin goConfig I2C IP is compatible

with the definitions in the official I²C manual, as shown in Table 2-2. And verified I²C can be up to 400KHz.

Table 2-2 2C Series I²C Interface Timing

Name	Description	Min	Max	Unit	Note
F _{SCL}	Clock Frequency	-	400	KHz	—
T _{low}	LOW period of the SCL	1.3	-	μs	
T _{high}	HIGH period of the SCL	0.6	-	μs	
T _{hd.sta}	Start Hold Time	0.6	-	μs	
T _{su.sta}	Start Setup Time	0.6	-	μs	
T _{hd.dat}	Data In Hold Time	0	-	ns	
T _{su.dat}	Data In Setup Time	100	-	ns	
T _{su.sto}	Stop Setup Time	0.6	-	μs	

9X Series Interface Timing

For 9X series interface timing of Gowin goConfig I2C IP, some parameters are not compatible with the definitions in the official I²C manual, as shown in Table 2-3. And verified I²C can be up to 400KHz.

Table 2-3 9X Series I²C Interface Timing

Name	Description	Min	Max	Unit	Note
F _{SCL}	Clock Frequency	-	400	KHz	—
T _{low}	LOW period of the SCL	1.3	-	μs	
T _{high}	HIGH period of the SCL	0.6	-	μs	
T _{hd.sta}	Start Hold Time	0.6	-	μs	
T _{su.sta}	Start Setup Time	0.6	-	μs	
T _{hd.dat}	Data In Hold Time	50 ^[1]	-	ns	
T _{su.dat}	Data In Setup Time	100	-	ns	
T _{su.sto}	Stop Setup Time	0.6	-	μs	

Note!

T_{hd.dat} is not consistent with the official I²C user manual.

2.3 IP Features

The features of Gowin goConfig I2C IP (2C Series) include:

- I²C interface in slave mode
- I²C speed up to 400 Kbps
- I²C interface supports 7-bit addressing
- I²C interface supports slave address configuration
- Supports read and write of internal memory
- Supports 25 MHz clock for user design
- Supports fundamental I²C read and write
- Generates clock and reset internally without external input
- Can be synthesized

The features of goConfig I2C IP (9X Series) include:

- I²C interface in slave mode
- I²C speed up to 400 Kbps
- I²C interface supports 7-bit addressing
- I²C interface supports Slave address configuration
- Supports read and write of internal memory
- Supports 20MHz clock
- Generates clock and reset internally without external input
- Can be synthesized

2.4 Resource Utilization

Taking Gowin GW1N-2C device as an instance, its resource utilization is as shown in Table 2-4.

Table 2-4 Resource Utilization (2C Series)

Device	Programming Language	LUT	REG	BSRAM
GW1N-2C	Verilog	608	241	1

Taking Gowin GW2AN-9X device as an instance, its resource utilization is as shown in Table 2-5.

Table 2-5 Resource Utilization (9X Series)

Device	Programming Language	LUT	REG	BSRAM
GW2AN-9X	Verilog	444	247	2

3 Functional Description

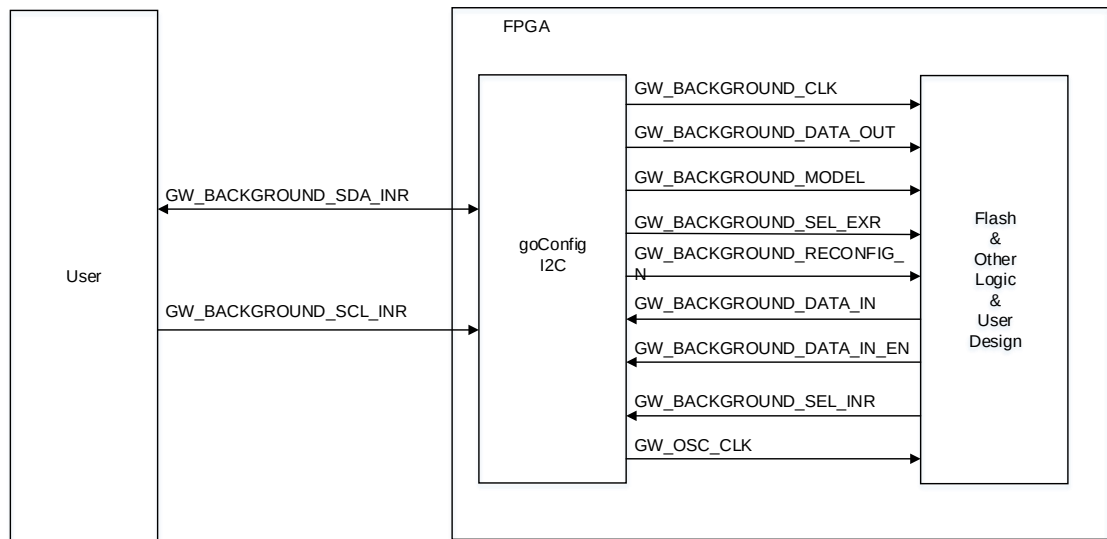
3.1 goConfig I2C Structure Block Diagram

Gowin goConfig I2C IP diagram includes User and FPGA. User is used to control the I²C interface as well as other user functions. FPGA is connected to Flash and the functions that users need to implement on FPGA. If only IP is considered, User controls the IP in the FPGA through the I²C interface, which controls the Flash to complete the background programming.

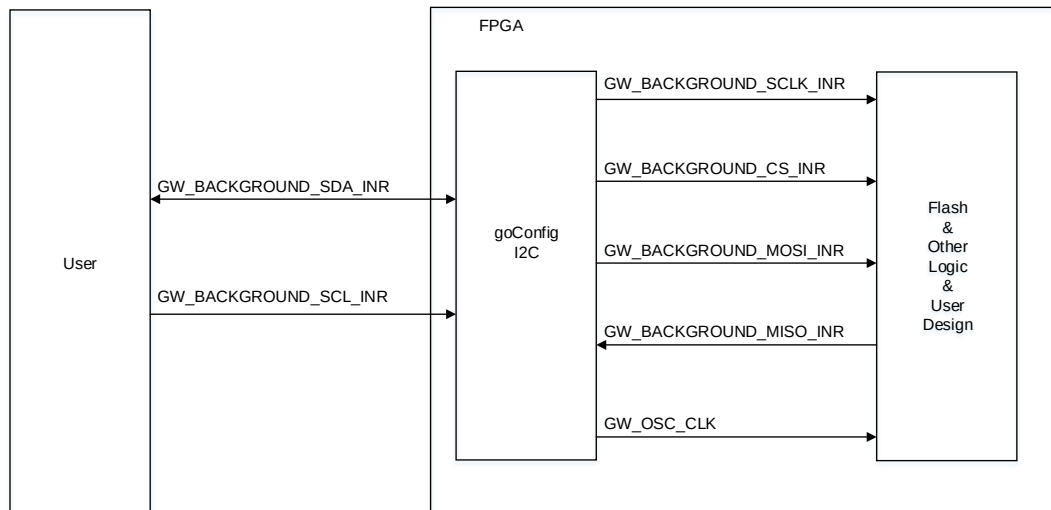
Note!

This manual mainly describes the interface operations of User, and does not cover the detailed operation and timing of the Flash interface.

The block diagram of the 2C series is shown in Figure 3-1. The I²C interface (GW_BACKGROUND_SCL_INR, GW_BACKGROUND_SDA_INR) is controlled by User; for the other signals, you can select or control them as requirements.

Figure 3-1 Structure Block Diagram (2C Series)

The block diagram of the 9X series is shown in Figure 3-2. The I²C interface (GW_BACKGROUND_SCL_INR, GW_BACKGROUND_SDA_INR) is controlled by User; for the other signals, you can select or control them as requirements.

Figure 3-2 Structure Block Diagram (9X Series)

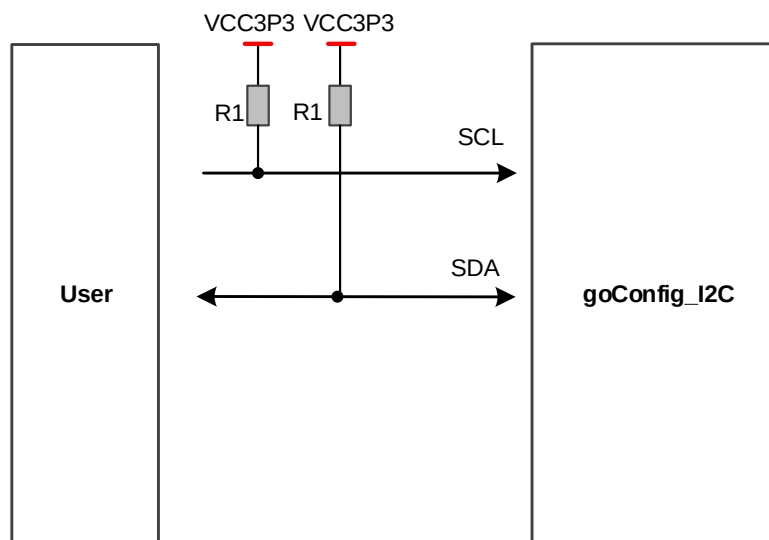
3.2 Gowin goConfig I2C IP External Circuit Connection

Note!

The descriptions in this section can be applied to both 2C series and 9X series.

Gowin goConfig I2C IP external circuit connection is as shown in Figure 3-3.

Figure 3-3 External Resistor Circuit Connection



R1:4.7K

Note!

- You can see the followings for the IO port attribute constraints.
 - GW_BACKGROUND_SCL_INR: IO_TYPE=LVC MOS33 PULL_MODE= UP
 - GW_BACKGROUND_SDA_INR: IO_TYPE=LVC MOS33 PULL_MODE= UP
DRIVE=8
- The supply voltage of I/O Bank is 3.3V.

3.3 Gowin goConfig I2C IP Background Programming Commands

Gowin goConfig I2C IP operation is different in 2C series and 9X series, the following will separately describe 2C series and 9X series.

3.3.1 2C Series

Command Definition

2C series commands are as shown in Table 3-1.

Table 3-1 2C Series Commands

No.	Name	Command Address	Description	Command Format	Note
1	Status	0x44	Read status register	CMD	—
2	Reset	0x55	Reset IP	CMD	
3	Send (data back)	0x66	Send data and read back Flash interface data	CMD + Length + Data + Check_sum	
4	Waiting	0x77	Waiting	CMD + Length + Check_sum	
5	Send (No data back)	0x88	Send data and no read back Flash interface data	CMD + Length + Data + Check_sum	
6	Reconfign	0x99	Reconfiguration	CMD	—
7	IP Version	0xAA	IP Version	CMD	

The command is defined and described as follows:

- **CMD:** See Table 3-1 for the command support.
 - Definition: Command address
 - Width: 8 bits
- **Length:** Mini. 2
 - Definition: Sent valid byte length^[1]
 - Width: 16 bits, low 8 bits (Length_L) are sent first, high 8 bits (Length_H) are sent later.
 - Formula: Length = valid data length + 2.

Note!

- [1] Valid byte length is the valid data (Data) length + 2; take 0x66 command as an example, Length = Data length + 2.
- For example, if the expected input byte length is 10, then Length = 10 + 2 = 12.
- **Data:**
 - Definition: Sent valid data
 - Width: 8 bits for one data
 - Length: The actual length of the data sent is determined according to Length-2; Data_0 indicates the first valid data; Data_1 indicates

the second valid data, ... Data_N means the last valid data.

- Check_sum:

- Definition: Check
- Width: 16 bits, high 8 bits (Check_sum_H) are sent first, low 8 bits (Check_sum_L) are sent later.
- Formula: Check_sum = CMD + Length_L + Length_H + Data_0 + ... + Data_N

Note!

When Check_sum calculation overflows, only the lower 16 bits are taken and the rest are discarded. Suppose Check_sum=0x10F11, then Check_sum_H=0x0F; Check_sum_L=0x11, and the rest are discarded.

Status(0x44)

Read status register command, read only. The data read back is the value of the status register, and the status register is defined as shown in Table 3-2.

Table 3-2 Status Register

Bit(s)	Default	Description	Note
Bit[7:3]	0	Reserved	—
Bit [2]	0	1: Flash interface operating 0: Flash interface idle	
Bit[1]	0	1: Check error 0: Check correct	
Bit[0]	0	1: I ² C status error 0: I ² C status normal	—

An example of read status register command is as shown in Figure 3-4.

Figure 3-4 Example of Read Status Register Command

S	Address	W	A	0x44	Sr	Address	R	A	Data	N	P
---	---------	---	---	------	----	---------	---	---	------	---	---

Note!

In the following, unless otherwise specified, the names in similar diagrams have the same meanings as this example.

- For white background in the figure, it means I²C master sends to I²C slave; for gray background, it means I²C slave sends to I²C master.

- The names shown in the figure are defined as follows:
 - S: START in I²C bus
 - Address: Slave Address in I²C bus
 - W: WRITE in I²C bus
 - A: ACK in I²C bus
 - Sr: Repeated START in I²C bus
 - R: READ in I²C bus
 - Data: Transferred valid data in I²C bus
 - N: NACK in I²C bus
 - P: STOP in I²C bus
 - Length_L: Lower 8 bits of Length
 - Length_H: High 8 bits of Length
 - Data_0: First valid data
 - Data_N: Last valid data
 - Check_sum_H: High 8 bits of Check_sum
 - Check_sum_L: Low 8 bits of Check_sum

Reset(0x55)

Reset IP command. It can reset internal status of the IP, as well as clear the internal FIFO.

Note!

This process only involves the IP, not the background programming.

Figure 3-5 Example of Reset IP Command

S	Address	W	A	0x55	A	P
---	---------	---	---	------	---	---

Send(0x66, data back)

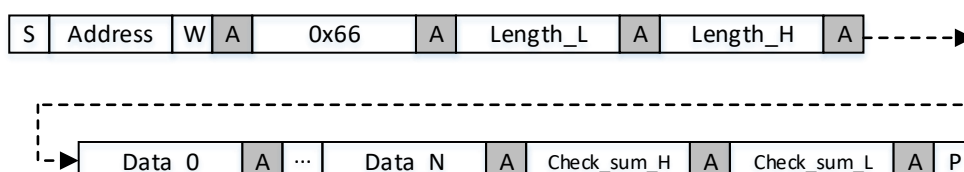
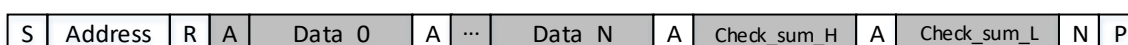
Send data and read back Flash data command. The I²C interface sends the byte length of the data to be read back, and sends the data at the same time; if there is no check error, the IP will be converted to the Flash interface and the data will be read back into the FIFO, as shown in Figure 3-6. At this time, if users need this data, a read command can be sent through the I²C interface, you can read the data in the FIFO, as shown in Figure 3-7.

In brief, the process of read back includes two steps.

1. Read the data into the FIFO of the IP.
2. Read the data in the FIFO through the I²C bus.

Note!

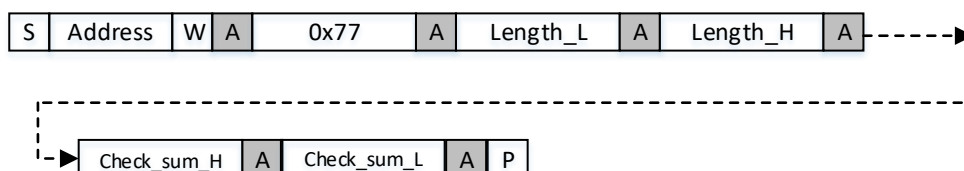
- Some commands can be inserted between step 1 and step 2, such as read status command.
- In Figure 3-6, Data_0~Data_N can be any value, but it is recommended to write 0x00 or 0xFF.
- In Figure 3-7, Data_0~Data_N is the actual read back value.

Figure 3-6 Example of Sending Read-back Command**Figure 3-7 Example of Read FIFO Command****Waiting(0x77)**

Waiting command. The I²C interface sends the byte length of waiting; if there is no check error, Flash interface will keep sending the idle clock (no data transmission) until the bytes waiting count is completed.

Note!

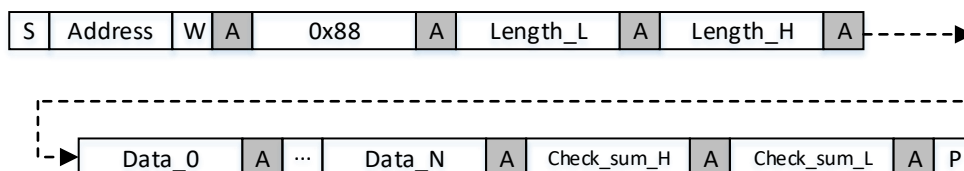
In 2C series, Flash interface clock is 2.5MHz, which means one cycle is 400ns. 1 byte is $8 \times 400 \text{ ns} = 3200 \text{ ns} = 3.2 \mu\text{s}$. For example, if waiting 150ms, the number of valid bytes is calculated as $46875 = 0xB71B$; the number of input bytes is $0xB71B + 2 = 0xB71D$, then Length_L = 0x1D; Length_H = 0xB7.

Figure 3-8 Example of Waiting Command**Send(0x88, No data back)**

Send data and no read back Flash data command. The I²C interface first stores the data to be sent into the FIFO, after the data count is

completed and there is no check error, the data in the FIFO is sent out once through the Flash interface.

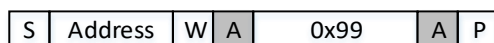
Figure 3-9 Example of Sending Data and No Read Back Flash Data Command



Reconfig(0x99)

Reconfig command. When the I²C interface sends this command, "Reconfig_N" pin can be pulled down.

Figure 3-10 Example of Reconfig Command



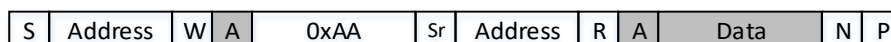
IP Version(0xAA)

IP version command. The I²C interface sends this command and read-back data is a fixed value.

Note!

Version_Data=0x20 for the current version.

Figure 3-11 IP version Command



3.3.2 9X Series

There are significant differences between the 9X series and 2C series devices. Note the differences when you use them. 9X series communicates in the register mode.

Register Definition

The definition of the related register is shown in Table 3-3.

Table 3-3 Register Definition

No.	Name	Address	Access	Description	Format	Note
1	CSCR	0x00	Write	$\overline{CS}$ signal control register	ADDR+Data	–
2	DICN	0x01	Write/Read	Write operation byte number register	ADDR+Data	
3	DOCN	0x02	Write/Read	Read operation byte number register	ADDR+Data	
4	DIFR	0x03	Write	Write FIFO Register	ADDR+Data	
5	SRSTR	0x08	Write	Soft reset register	ADDR+Data	
6	SR	0x80	Read	Status Register	ADDR+Data	
7	DOFR	0x81	Read	Read FIFO register	ADDR+Data	

Format: register address (ADDR) + data (Data); the length of the data (in Byte) varies depending on the register address.

In the following, the register data is distinguished by "register name-Data". For example, CSCR-Data means the Data of $\overline{CS}$ signal control register.

CSCR(0x00)

$\overline{CS}$ signal control register in the format of ADDR + CSCR-Data (1 byte), and the length of CSCR-Data is 1 byte, as shown below.

Table 3-4 CSCR-Data Definition

Bit(s)	Default	Access	Description	Note
Bit [7:0]	0x00	Write	0xFF: CS signal pulls down (Flash interface will enter operating state) 0x00: CS signal pulls up (Flash interface will enter idle state) Others: reserved	–

Example of using $\overline{CS}$ signal control register is as shown in Figure 3-12; pull down cs, CSCR-Data=0xFF.

Figure 3-12 Example of Using CSCR Register

S	Address	W	A	0x00	A	Data	A	P
---	---------	---	---	------	---	------	---	---

DICN(0x01)

Write operation byte number register in the format of ADDR + DICN-Data (2 bytes). The length of DICN-Data is 2 bytes, the valid range of the value is 1~300, where 0x0001 means 1 byte, 0x00FF means 255 bytes, and so on for other values.

Table 3-5 DICN-Data Definition

Bit(s)	Default	Access	Description	Note
Bit[15:0]	0x0000	Write/Read	1~300: Byte Number of write operation Others: reserved	—

When two bytes of DICN-Data are sent, the low byte Bit[7:0] is sent first and then the high byte Bit[15:8] is sent. Therefore, when DICN-Data=0x0001, 0x01 is sent first and then 0x00.

An example of DICN register write is shown in Figure 3-13. If DICN-Data=0x0001, then Data_0 = 0x01 and Data_1 = 0x00.

Figure 3-13 Example of DICN Register Write

S	Address	W	A	0x01	A	Data_0	A	Data_1	A	P
---	---------	---	---	------	---	--------	---	--------	---	---

An example of DICN register read is shown in Figure 3-14. If DICN-Data=0x0001, then Data_0 = 0x01 and Data_1 = 0x00.

Figure 3-14 Example of DICN Register Write Read

S	Address	W	A	0x01	Sr	Address	R	A	Data_0	A	Data_1	N	P
---	---------	---	---	------	----	---------	---	---	--------	---	--------	---	---

DOCN(0x02)

Read operation byte number register in the format of ADDR + DOCN-Data (2 bytes). The length of DOCN-Data is 2 bytes, the valid range of the value is 1~300, where 0x0001 means 1 byte, 0x00FF means 255 bytes, and so on for other values.

Table 3-6 DOCN-Data Definition

Bit(s)	Default	Access	Description	Note
Bit [15:0]	0x0000	Write/Read	1~300: Byte Number of write operation Others: reserved	—

When two bytes of DOCN-Data are sent, the low byte Bit[7:0] is sent first and then the high byte Bit[15:8] is sent. Therefore, when DOCN-Data=0x0001, 0x01 is sent first and then 0x00.

An example of DOCN register write is shown in Figure 3-15. If DOCN-Data=0x0001, then Data_0 = 0x01 and Data_1 = 0x00.

Figure 3-15 Example of DOCN Register Write

S	Address	W	A	0x02	A	Data_0	A	Data_1	A	P
---	---------	---	---	------	---	--------	---	--------	---	---

An example of DOCN register read is shown in Figure 3-16. If DOCN-Data=0x0001, then Data_0 = 0x01 and Data_1 = 0x00.

Figure 3-16 Example of DOCN Register Read

S	Address	W	A	0x02	Sr	Address	R	A	Data_0	A	Data_1	N	P
---	---------	---	---	------	----	---------	---	---	--------	---	--------	---	---

DIFR(0x03)

Write FIFO register in the format of ADDR + DIFR-Data (1~300 bytes), and the DIFR-Data length ranges from 1~300 bytes. Send one byte at a time; the actual TX byte number needs to match the value of the write operation byte number register.

Table 3-7 DIFR-Data Definition

Bit(s)	Default	Access	Description	Note
Bit [7:0]	—	Write	0x00~0xFF	—

For example, DICN-Data=0x0011; then DIFR will send data of 0x0011=17 bytes.

An example of using DIFR register is as shown below; Data_0 indicates the first byte of write data and Data_N indicates the last write data. The number of write byte should be the same as the DICN-Data value.

Figure 3-17 Example of Using DIFR Register

S	Address	W	A	0x03	A	Data_0	A	...	Data_N	A	P
---	---------	---	---	------	---	--------	---	-----	--------	---	---

SRSTR(0x08)

Soft reset register in the format of ADDR + SRSTR-Data (1 byte). The length of SRSTR-Data is 1 byte; the reset is valid when SRSTR-Data =0xFF.

Table 3-8 SRSTR-Data Definition

Bit(s)	Default	Access	Description	Note
Bit[7:0]	0x00	Write	0xFF: soft reset valid Others: reserved	—

An example of using soft reset register is as shown in Figure 3-18; SRSTR-Data=0xFF.

Figure 3-18 Example of Using SRSTR Register

S	Address	W	A	0x08	A	Data	A	P
---	---------	---	---	------	---	------	---	---

SR(0x80)

Status register in the format of ADDR + SR-Data (3 bytes), and the SR-Data length is three bytes.

Table 3-9 SR-Data Definition

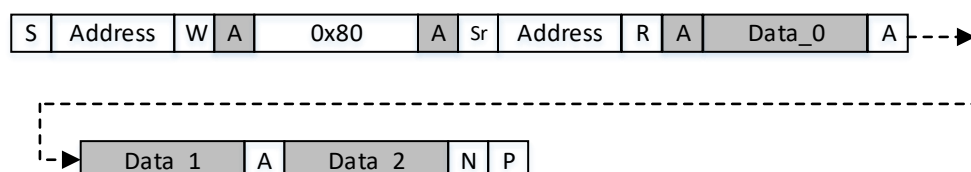
Bit(s)	Name	Access	Description	Note
Bit [11:15 PM]	TXLVL	Read	Number of valid data of TX FIFO Value range: 1 ~ 300	—
Bit [14:6]	RXLVL	Read	Number of valid data of RX FIFO Value range: 1 ~ 300	
Bit [5:4]	FIFO status	Read	Bit [5]: 1: Data in TXLVL 0: No data in TXLVL Bit [4]: 1: Data in RXLVL 0: No data in RXLVL	
Bit [3:0]	Flash	Read	Bit [3:2]: Reserved	—

Bit(s)	Name	Access	Description	Note
Bit [11:15 PM]	TXLVL	Read	Number of valid data of TX FIFO Value range: 1 ~ 300	—
Bit [14:6]	RXLVL	Read	Number of valid data of RX FIFO Value range: 1 ~ 300	
Bit [5:4]	FIFO status	Read	Bit [5]: 1: Data in TXLVL 0: No data in TXLVL Bit [4]: 1: Data in RXLVL 0: No data in RXLVL	
	interface operation status		Bit [1:0]: 11: reserved 10: Flash interface operation completed 01: Flash interface operation in progress 00: Flash interface idle	

When 3 bytes of SR-Data are sent, first is the low byte Bit[7:0], then the middle byte Bit[15:8], and then the high byte Bit[23:15]. Therefore, when SR-Data=0x000102, first send 0x02, then 0x01, and then 0x00.

An example of using status register is shown in Figure 3-19. If SR-Data=0x000102, then Data_0 = 0x02, Data_1 = 0x01, Data_2= 0x00.

Figure 3-19 Example of Using SR Register



DOFR(0x81)

Read FIFO register in the format of ADDR + DOFR-Data (1~300 bytes), and the DOFR-Data length ranges from 1~300 bytes. Send one byte at a time; the actual number of read byte needs to match the value of the read operation byte number register.

Table 3-10 DOFR-Data Definition

Bit(s)	Default	Access	Description	Note
Bit [7:0]	—	Read	0x00~0xFF	—

An example of using DOFR register is as shown below; Data_0 indicates the first byte of read data and Data_N indicates the last read data. The number of read byte should be the same as the DOCN-Data value.

Figure 3-20 Example of Using DOFR Register

S	Address	W	A	0x81	A	Data_0	A	...	Data_N	N	P
---	---------	---	---	------	---	--------	---	-----	--------	---	---

4 Signal Definition

4.1 Signal Definition

For Gowin goConfig I2C IP in the 2C series and 9X series, there are some differences in the IO interface definition, the following will describe 2C series and 9X series separately. For the 2C series, you can see Table 4-1.

Table 4-1 I/O Definition for 2C Series

No.	Signal Name	I/O	Description	I/O Constraint	Note
1	GW_BACKGROUND_SCL_INR	Input	I ² C bus clock	Name constraint	The I/O of all the signals takes IP as reference
2	GW_BACKGROUND_SDA_INR	Inout	I ² C bus Data	Name constraint	
3	GW_BACKGROUND_SEL_INR	Input	Background programming enable input 1: Enable (default recommended) 0: Disable	-	
4	GW_OSC_CLK	Output	25 MHz output Clock	-	
5	GW_BACKGROUND_CLK	Output	IP-Flash interface clock	Automatically	
6	GW_BACKGROUND_MODEL	Output	IP-Flash interface mode selection	Automatically	
7	GW_BACKGROUND_DATA_OUT	Output	IP-Flash interface data output	Automatically	
8	GW_BACKGROUND_DATA	Input	IP-Flash interface	Automatically	

No.	Signal Name	I/O	Description	I/O Constraint	Note
	A_IN		data input		
9	GW_BACKGROUND_DAT A_IN_EN	Input	IP-Flash interface input data valid signal	Automatically	
10	GW_BACKGROUND_SEL_ EXR	Output	Background programming enable output	Automatically	
11	GW_BACKGROUND_REC ONFIG_N	Output	RECONFIG_N signal	Automatically	

Note!

- The pin constraints are defined as follows:
 - "-": no restriction, free to name, and free to constrain the location.
 - Name constraint: You can't name freely, but can constrain the location. That is, that is, the top-level IO name needs to be named as specific IO names, including signal name and direction.
 - "Automatically": Automatically constrain the location. You can't name freely or constrain the location. That is, the top-level I/O name needs to be exactly the same as those IP defined; and the location cannot be constrained, Gowin Software will automatically constrain the location.
- "GW_BACKGROUND_SEL_INR": When there is no enable change, input should be 1, high.
- For 2C Series, the specific IO names are the names of IP IO interfaces. That is, the top-level IO name needs to be exactly the same as those of IP defined.
- The I/O constraints in 2C series are also applicable to 9X series.

9X series I/O definition is as shown in Table 4-2.

Table 4-2 I/O Definition for 9X Series

No.	Signal Name	I/O	Description	I/O Constraint ^[1]	Note
1	GW_BACKGROUND_S CL_INR	Input	I ² C bus clock	-	The I/O of all the signals takes IP as reference
2	GW_BACKGROUND_S DA_INR	Inout	I ² C bus Data	-	
3	GW_OSC_CLK	Output	20MHz output Clock	-	
4	GW_BACKGROUND_S CLK_INR	Output	IP-Flash interface clock	Automatically ^[2]	

No.	Signal Name	I/O	Description	I/O Constraint ^[1]	Note
5	GW_BACKGROUND_CS_INR	Output	IP-Flash interface CS output	Automatically ^[2]	
6	GW_BACKGROUND_MOSI_INR	Output	IP-Flash interface data output	Automatically ^[2]	
7	GW_BACKGROUND_MISO_INR	Input	IP-Flash interface data input	Automatically ^[2]	

Note!

[1] For the definition of each I/O constraint, please refer to the description in 2C Series.

[2] When used at the top level, some IO interface names need to be modified to the specific IO names of 9X series, meaning the dedicated names of Flash interface; and the dedicated names of Flash interface are defined as follows:

IP IO name > dedicated names of Flash interface (IO)

GW_BACKGROUND_SCLK_INR > **O_flash_ck(Output)**

GW_BACKGROUND_CS_INR > **O_flash_cs_n(Output)**

GW_BACKGROUND_MOSI_INR > **IO_flash_do(InOut)**

GW_BACKGROUND_MISO_INR > **IO_flash_di(Output)**

5 Instructions for IP Use

Note!

The content of this chapter is focused on describing the IP, and the descriptions are limited to IP operations.

There are differences between 2C series and 9X series in using Gowin goConfig I2C IP, and this chapter will describe Gowin Software configuration, recommended IP process, and notes respectively.

5.1 2C Series

5.1.1 Software Configuration

Before synthesize the IP project with background programming, you need to modify Gowin Software configuration first, then Gowin goConfig I2C IP can work properly.

In Gowin Software, select 2C series devices, click "Project > Configuration > BitStream > Background Programming > I2C" as shown in Figure 5-1.

Figure 5-1 Open Configuration Option

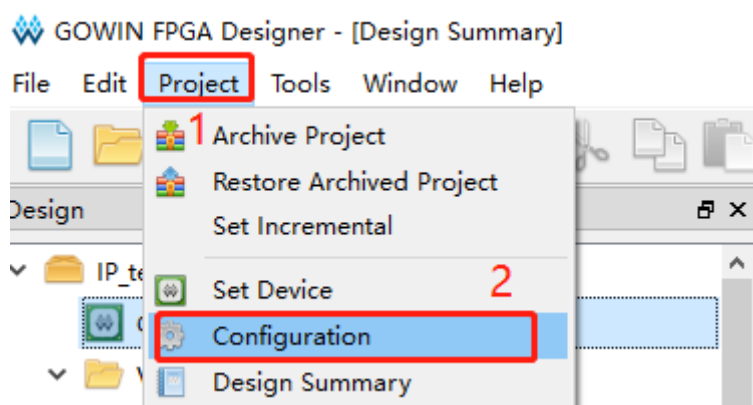
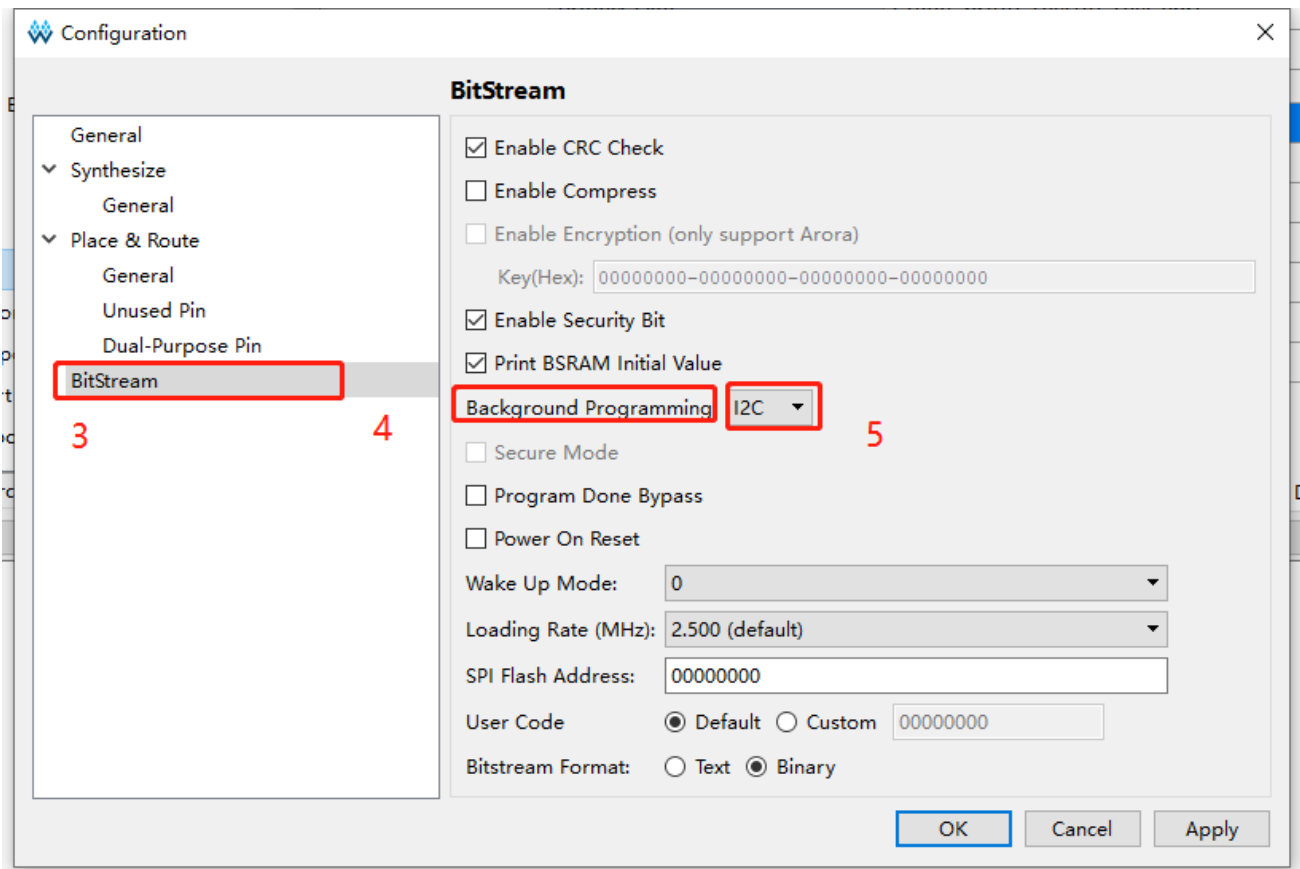


Figure 5-2 Select I2C in Background Programming Option



5.1.2 Flow Description

For the 2C series, the command includes read/write, and operation.

- Read/write command: Send CMD, and send length to control read/write command, such as "Send(data back)", "Waiting", "Send(No data back)".
- Operation command: Only send CMD, not length, such as "Reset", "Reconfig".

2C series operations include the followings:

1. Preparation
2. Write flow
3. Read flow
4. Trigger RECONFIG_N

Note!

IP will operate Flash interface directly after read/write command is completed. The operation command mainly controls/views the status of the IP.

Preparation

IP preparation can verify/ensure that the IP is working properly, and you can follow the steps below.

1. Send "Reset" (0x55) to ensure that the IP is in initial state.
2. Send "IP Version" (0xAA) to check the read back version value (Version_Data=0x20) to ensure that the IP version is as expected.
3. Send "Status" (0x44) to check the read back status value to ensure that it is 0x00.

When the operation is completed and the read back data is as expected, it means that communication with the IP can be started.

Write Command Flow

Once the preparation is done, you can start the write command flow. The write command flow generally follows the following steps.

1. Send "Send (No data back)" (0x88) to write the data.
2. Send "Status" (0x44) to ensure that the IP operation is complete and other operations can be performed.

When the operation is completed and the read back status data is as expected, it means that communication with the IP can be started.

Read-back Command Flow

Once the preparation is done, you can start read-back command flow. The read-back command flow generally follows the following steps.

1. Send "Send(data back)" (0x66) to write the length of the data to be read back.
2. Send "Status" (0x44) to ensure Flash operation is complete.
3. Read the data through the I²C interface, you can refer to Send(0x66, data back).
4. Send "Status" (0x44) to ensure that the IP operation is complete and other operations can be performed.

When the operation is completed and the read-back status data is as expected, it means that communication with the IP can be started.

Trigger RECONFIG_N

Trigger RECONFIG_N, and the function of "RECONFIG_N" is that after the first step of background programming (write Flash) is completed, the second step (load Flash data to SRAM) can be executed by triggering "RECONFIG_N" to complete the background programming.

5.1.3 Notes

When using Gowin goConfig I2C IP, you should note the followings:

I/O Constraint

As described in I/O definition, the IO signals of 2C series includes two types: unconstrained I/O and constrained I/O (including name constraint and auto).

- Unconstrained I/O: Same as regular use, you can choose the name, whether to place to the top level or not, etc.
- Constrained I/O: Constrain I/O names and must be input or output through the top layer.

Note!

Gowin Software will detect the Constrained IO; if I2C is selected in the background programming, but the I/O does not exist or does not correspond, the software will report an error when executing "Place & Route".

OSC Output

Gowin goConfig I2C IP uses the OSC module in the device. Also, IP can provide a 25MHz clock output, and you can use as requirements.

Cache Limit

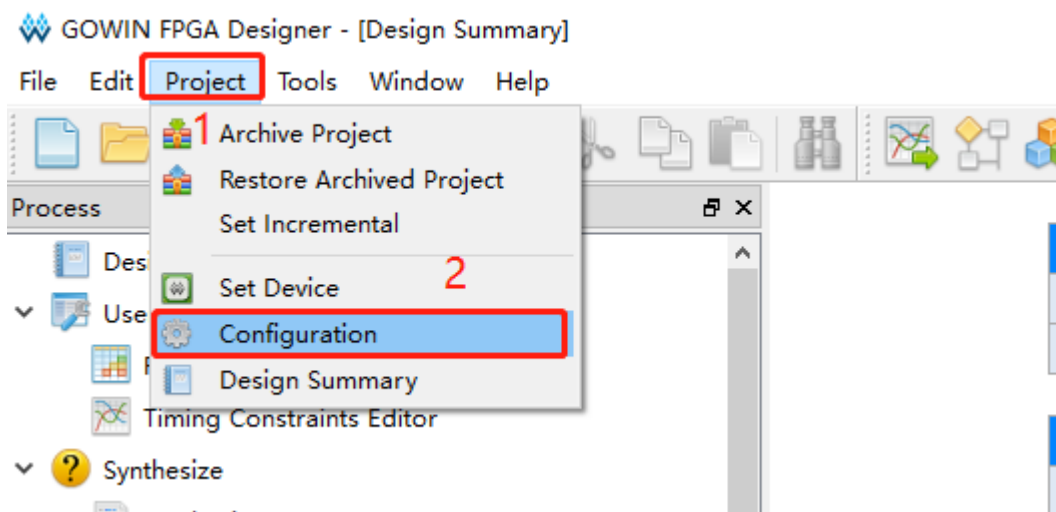
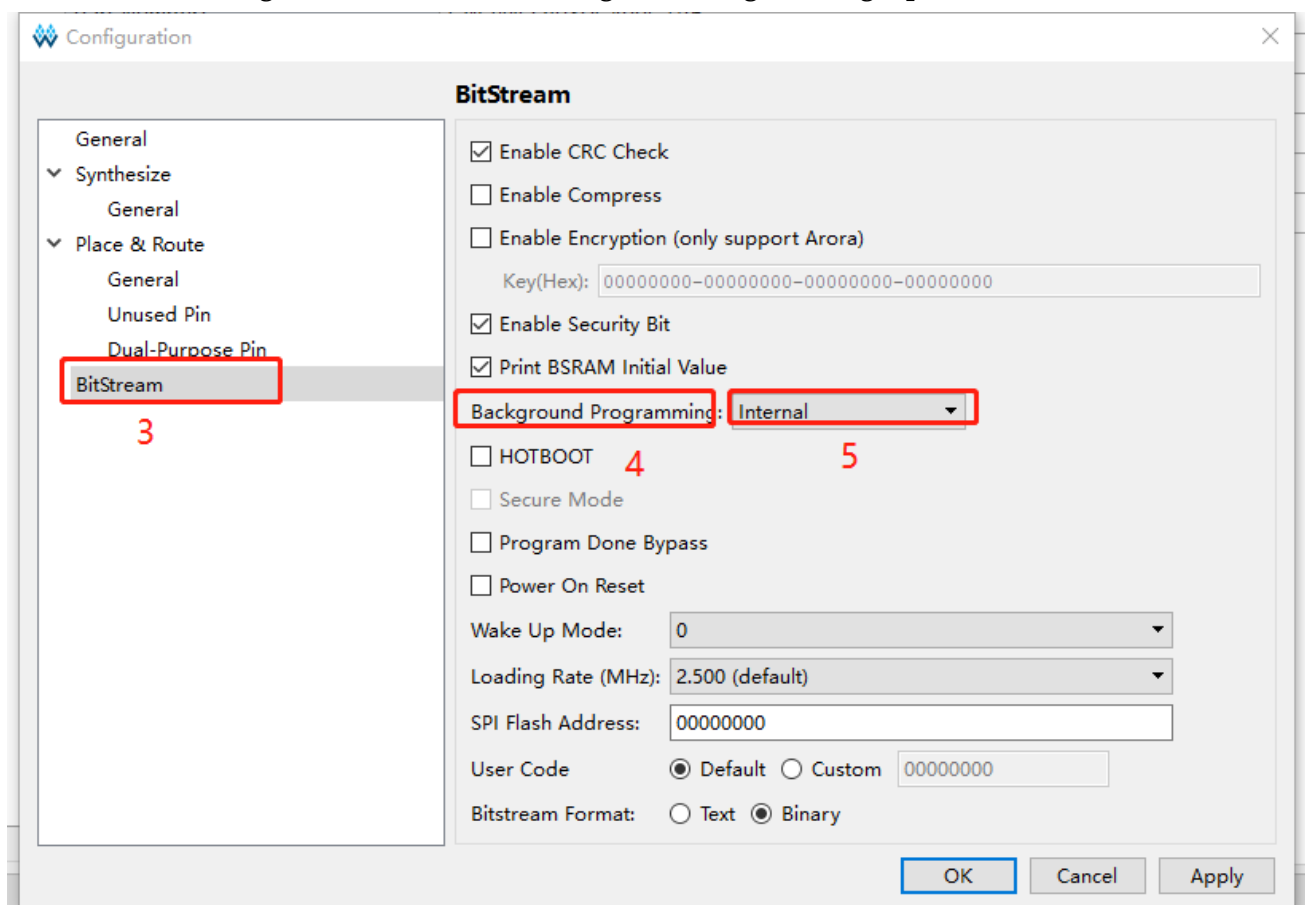
Gowin goConfig I2C IP uses one BSRAM for data caching, so the data transferred at one time cannot exceed one BSRAM.

5.2 9X Series

5.2.1 Software Configuration

Before synthesize the IP project with background programming, you need to modify Gowin Software configuration first, then Gowin goConfig I2C IP can work properly.

In Gowin Software, select 9X series devices, click "Project > Configuration > BitStream > Background Programming > Internal".

Figure 5-3 Open Configuration Option**Figure 5-4 Select I2C in Background Programming Option**

5.2.2 Flow Description

Gowin goConfig I2C IP 9X series read/write Flash through registers.
The flow is as follows.

Write Operation Flow

For 9X series, some register configurations need to be completed first, and write operation flow is as follows.

1. Check SR status register (0x80); when the SPI port is idle (SR-Data[1:0]=00), initiate the operation.
2. Write the number of bytes to be sent to DICN (0x01).
3. Write the data to the DIFR (0x03) register in sequence.
4. Set CSCR (0x00) register value to 0xFF to initiate the operation (Flash interface will initiate the operation).
5. Check SR (0x80) status register to read the operation status.
6. When the Flash interface operation is completed (SR-Data[1:0]=10), set CSCR (0x00) register value to 0x00; SPI interface will enter the idle status and end this operation.

Read Operation Flow

For 9X series, some register configurations need to be completed first, and the read operation flow is as follows.

1. Check SR status register (0x80); when the SPI port is idle (SR-Data[1:0]=00), initiate the operation.
2. Write the number of bytes to be sent to DICN (0x01).
3. Write the number of bytes to be received to DOCN (0x02).
4. Write the data to the DIFR (0x03) register in sequence.
5. Set the CSCR (0x00) register value to 0xFF to initiate the operation.
6. Query SR (0x80) status register to read the operation status.
7. When the Flash interface operation is completed (SR-Data[1:0]=10); read data through Flash (0x81) register.
8. Set CSCR (0x00) register value to 0x00; SPI interface will enter the idle status and end this operation.

5.2.3 Notes

When using Gowin goConfig I2C IP, you should note the followings.

I/O Constraint

As explained in Table 4-2, the I/O signals of the Gowin goConfig I2C IP 9X series includes unconstrained I/O and constrained I/O (automatically).

- Unconstrained I/O: Use as usual; for example, you can choose the name, place it to the top level or not, etc.
- Constrained I/O: The I/O name is constrained and must be input or output through the top level.

When using constrained I/O in the top-level module, the I/O interface name of 9X series should be modified to specific IO name, namely the exclusive name of Flash interface, as shown in Table 5-1.

Table 5-1 Contrast Table

No.	IP Interface Name	Flash Interface		Description	Note
		Exclusive Name	I/O		
1	GW_BACKGROU ND_SCLK_INR	O_flash_ck	Output	IP-Flash interface clock	For top-level use, the interface I/O needs to be based on the Flash interface.
2	GW_BACKGROU ND_CS_INR	O_flash_cs_n	Output	IP-Flash interface CS output	
3	GW_BACKGROU ND_MOSI_INR	IO_flash_do	Inout	IP-Flash interface data output	
4	GW_BACKGROU ND_MISO_INR	IO_flash_di	Output	IP-Flash interface data input	

When the top-level module defines constrained I/O, the I/O direction should be the same as the Flash interface direction, as shown in Figure 5-5.

Figure 5-5 I/O Definition Example

```

module goConfig_I2C_Top_wrap
(
    //i2c
    inout  GW_BACKGROUND_SDA_INR,
    input  GW_BACKGROUND_SCL_INR,
    //flash IO
    output O_flash_ck,      //
    output O_flash_cs_n,    //
    inout  IO_flash_do,     //
    output IO_flash_di,     //
    //Others
    output GW_OSC_CLK

);

goConfig_I2C_Top inst_goConfig_I2C(
    .GW_BACKGROUND_SDA_INR(GW_BACKGROUND_SDA_INR),
    .GW_BACKGROUND_SCL_INR(GW_BACKGROUND_SCL_INR),
    .GW_BACKGROUND_SCLK_INR(O_flash_ck),
    .GW_BACKGROUND_CS_INR(O_flash_cs_n),
    .GW_BACKGROUND_MISO_INR(IO_flash_do),
    .GW_BACKGROUND_MOSI_INR(IO_flash_di),

    .GW_OSC_CLK(GW_OSC_CLK)
);
endmodule

```

Note!

Gowin Software will detect this constrained I/O; if Internal is selected in the background programming, but the I/O does not exist or does not correspond, after Place & Route, the result is not consistent with the expectation.

OSC Output

Gowin goConfig I2C IP uses the OSC module in the device. Also, IP can provide a 20MHz clock output, and you can use it as requirements.

Buffer Limit

Gowin goConfig I2C IP uses two BSRAMs for RX/TX data buffer respectively, and the data limit for one read/write is 1~300 bytes.

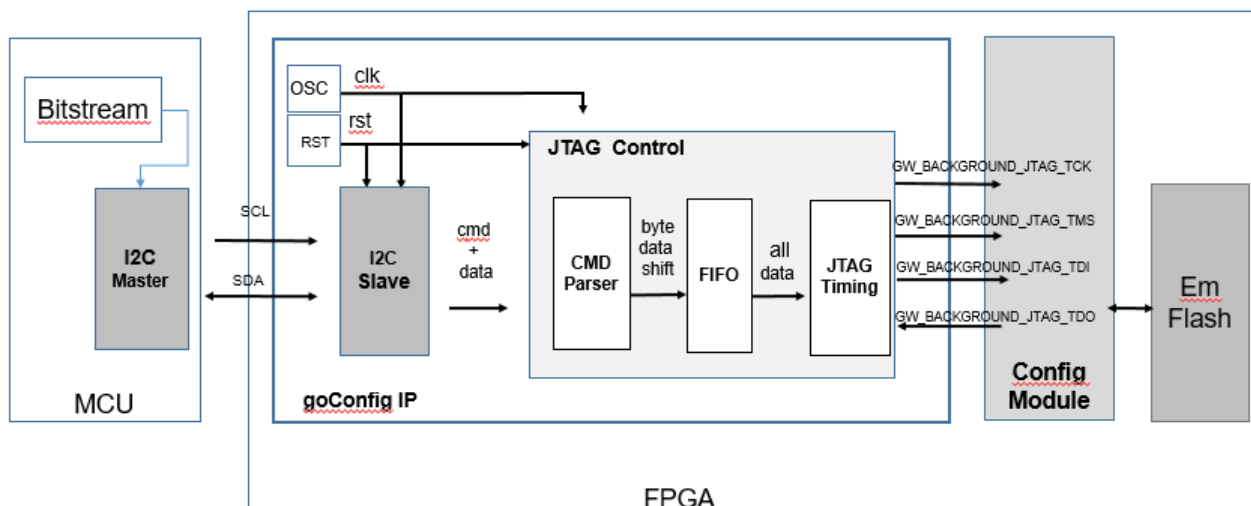
6 Example Description

For more detailed usage, you can refer to [*TN715, Gowin goConfigIP\(I2C\) Example Code Application Note*](#), and you can also contact our FAE to obtain the example code.

6.1 Example for 2C Series

This example code is executed in an MCU, and with the help of goConfigIP through the I²C interface, it can perform erasing and programming operations on the embedded Flash. It can also read the FPGA's ID code, user code, and status code.

Figure 6-1 Block Diagram



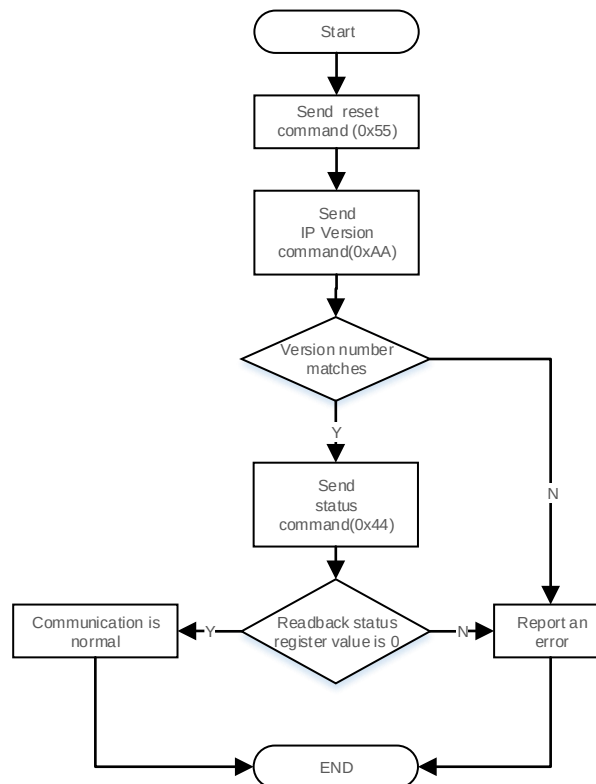
6.1.1 Check MCU-IP Communication for Normalcy

Before starting the background programming, it is important to ensure that communication between the MCU and IP is functioning properly. The check process is as follows:

1. Send a reset command (0x55) to ensure that the IP is in its initial status (API: gw_i2c_send_reset).
2. Send an IP Version command (0xAA) to check the read-back version value (e.g., Version Data=0x20) to ensure that the IP version matches the expected one (API: gw_i2c_read_version).
3. Send a Read Status Register command (0x44) and check the read-back status value (API: gw_i2c_read_reg_status).
4. When the operation is completed and the read-back data matches the expectation, it indicates that the communication with the IP is ready to start.

The flowchart of checking MCU-IP communication is shown in Figure 6-2.

Figure 6-2 Flowchart of Checking MCU-IP Communication



6.1.2 Overall Process of Background Programming

The background programming process is as follows:

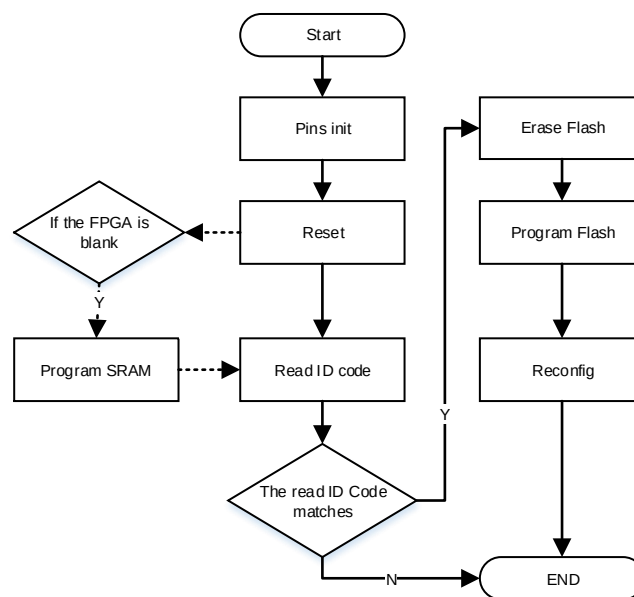
1. Initialize MCU platform pins (I2C, debug_uart).
2. Reset the IP (API: gw_i2c_send_reset).
3. If the chip is not blank, skip this step. If the chip is blank, configure

SRAM (API: `i2c_configmode_program_sram`).

4. Compare the ID code read by the MCU (I2C) with the ID code in the data stream file (API: `mach_code`). If they match, continue; if not, terminate the background programming.
5. Erase Flash (API: `gw_i2c_erase_flash_tsmc`).
6. Program Flash (API: `gw_i2c_prog_flash_tsmc`).
7. Reconfigure (`gw_i2c_send_reconfig_n`).

The flowchart of background programming is shown in Figure 6-3.

Figure 6-3 Flowchart of Background Programming



Read FPGA Code

Use API: `gw_i2c_read_code` to read FPGA code. Once communication between the MCU and IP is normal, you can use I2C to read FPGA code.

- ID code, also known as JEDEC ID Code, is a fundamental identifier for the FPGA device.
- User code is a user's identifier for the FPGA device they are using.
- Status code is a status identifier for the FPGA device after loading the data stream file.

For example:

- To read the ID Code, use the API: `gw_i2c_read_code(0x11)`.
- To read the User Code, use the API: `gw_i2c_read_code(0x13)`.
- To read the Status Code, use the API: `gw_i2c_read_code(0x41)`.

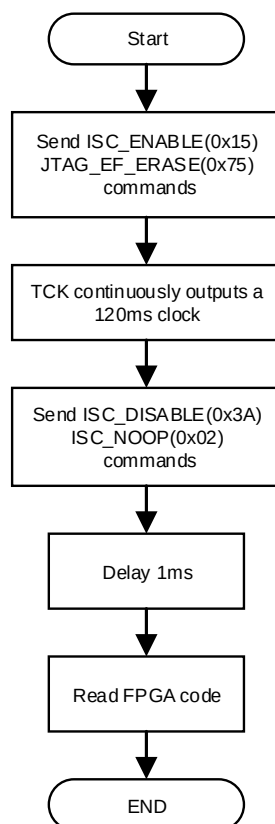
Erase Embedded Flash

Use the API: `gw_i2c_erase_flash_tsmc` to erase the embedded Flash. The process for erasing the embedded flash is as follows:

1. Send `ISC_ENABLE (0x15)` and `JTAG_EF_ERASE (0x75)` commands. These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
2. TCK continuously outputs 120ms clock (API: `gw_i2c_send_runtest_xus`).
3. Send `ISC_DISABLE (0x3A)` and `ISC_NOOP (0x02)` commands. These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
4. Delay 1ms (API: `gw_i2c_send_runtest_xus`).
5. Read FPGA code (API: `gw_i2c_read_code`).

The flowchart of erasing embedded Flash is shown in Figure 6-4.

Figure 6-4 Flowchart of Erasing Embedded Flash

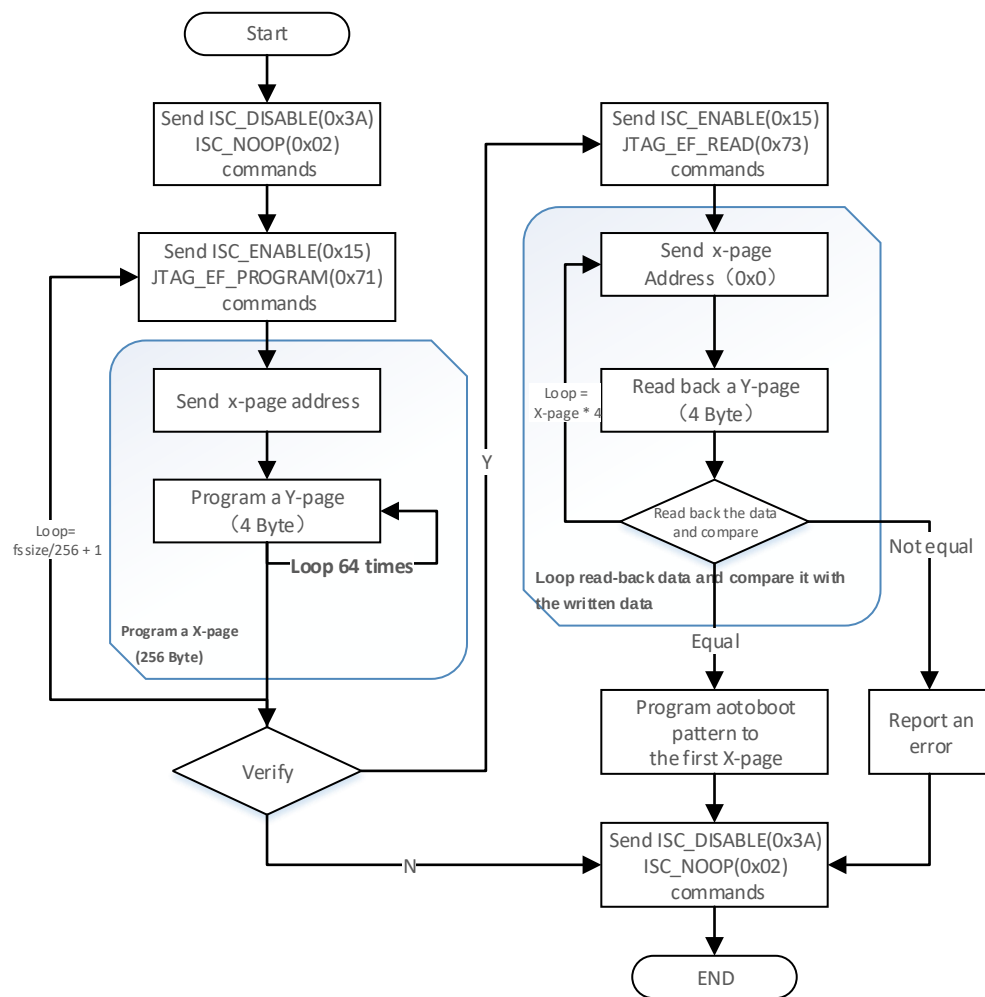


Program Embedded Flash

Use the API: `gw_i2c_prog_flash_tsmc` to program the embedded Flash. The process for programming the embedded flash is as follows:

1. Send `ISC_DISABLE` (0x3A) and `ISC_NOOP` (0x02). These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
2. Send `ISC_ENABLE` (0x15) and `JTAG_EF_PROGRAM` (0x71). These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
3. Send the X-page address. This address data is packaged using the API (`write_one_x_address`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
4. Program a Y-page. One X-page contains 64 Y-pages, and this Y-page data is packaged using the API (`write_one_y_page`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
5. If read-back verification is enabled, send `ISC_ENABLE` (0x15) and `JTAG_EF_READ` (0x73). These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
6. Send the X-page address data 0x00. This address data is packaged using the API (`write_one_x_address`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
7. Loop the read-back data from Flash and compare it with the data written in the data stream file (API: `recv_one_y_page_and_verify`).
8. After the read-back loop comparison, write the autoboot-pattern data to the first X-page. This autoboot-pattern data is packaged using the API (`i2c_prog_xpage`) and sent it to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).
9. Send `ISC_DISABLE` (0x3A) and `ISC_NOOP` (0x02). These two commands are packaged using the API (`write_inst_and_data`) and sent to the FPGA through the MCU's I2C low-level interface (API: `i2c_write_buffer`).

The flowchart of programming embedded Flash is shown in Figure 6-5.

Figure 6-5 Flowchart of Programming Embedded Flash

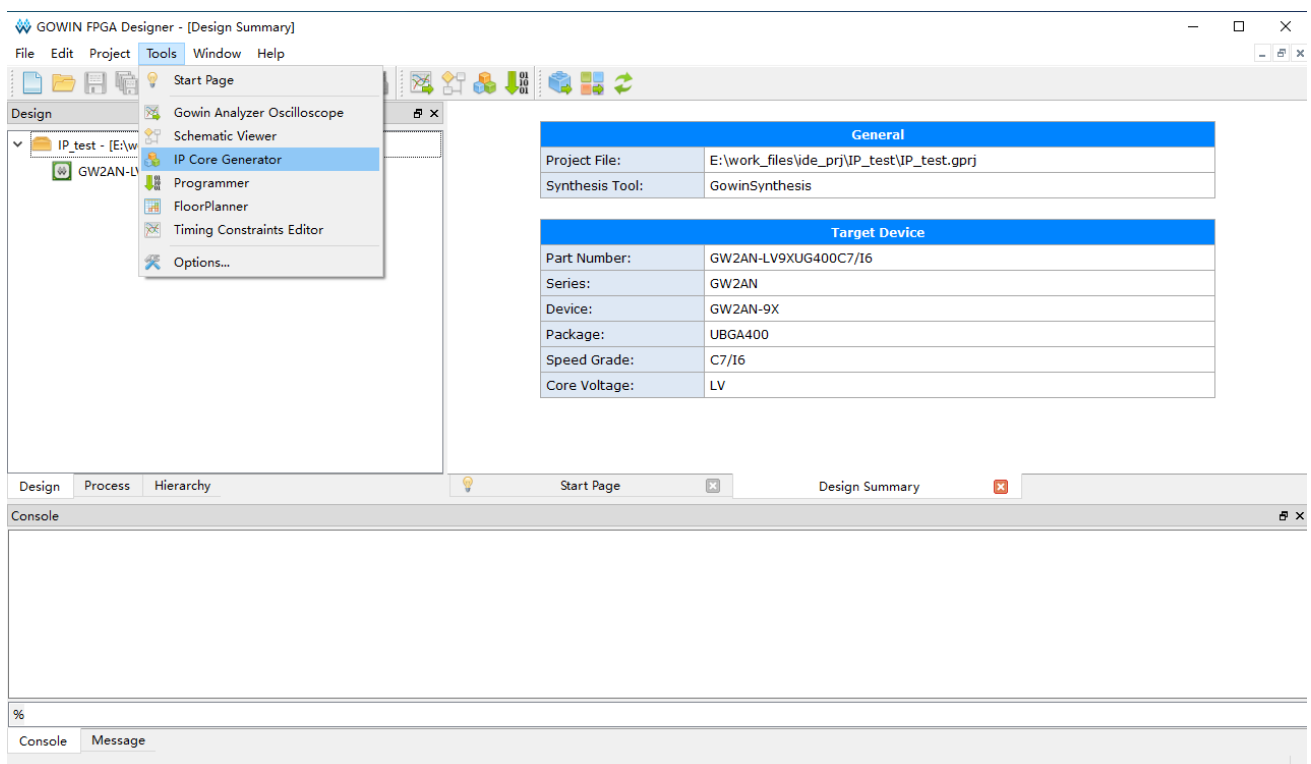
7 Interface Configuration

In Gowin Software, you can click Tools > IP Core Generator > goConfig I2C to call and configure Gowin goConfig I2C IP.

1. Open IP Core Generator

After creating the project, you can click the “Tools” tab in the upper left, then click the IP Core Generator via the drop-down list, as shown in Figure 7-1.

Figure 7-1 IP Core Generator

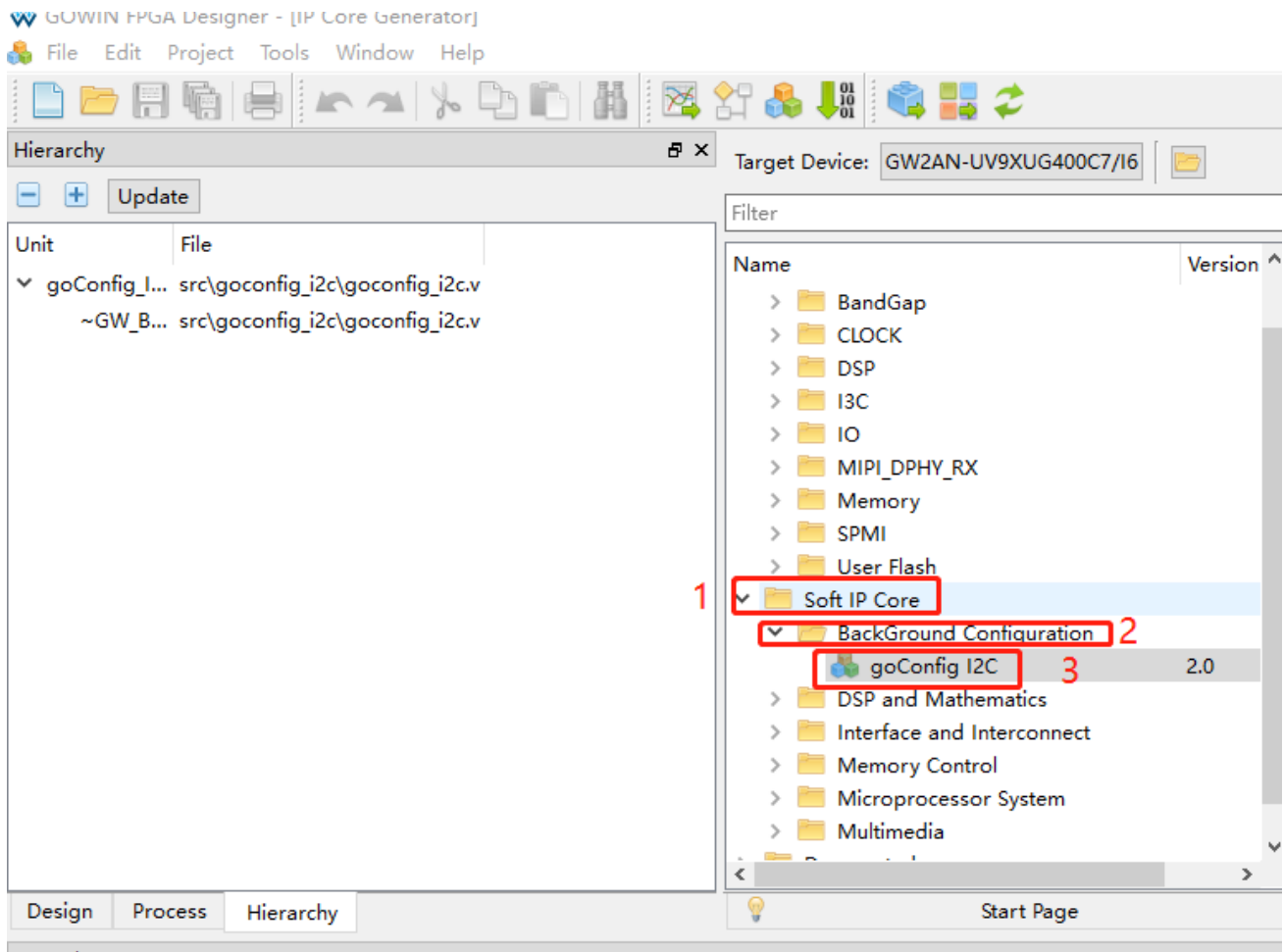


2. Open Gowin goConfig I2C IP core.

Select “Soft IP Core> BackGround Configuration > goConfig I2C”, as

shown in Figure 7-2. Double click to open the configuration interface.

Figure 7-2 Open goConfig I2C IP Core

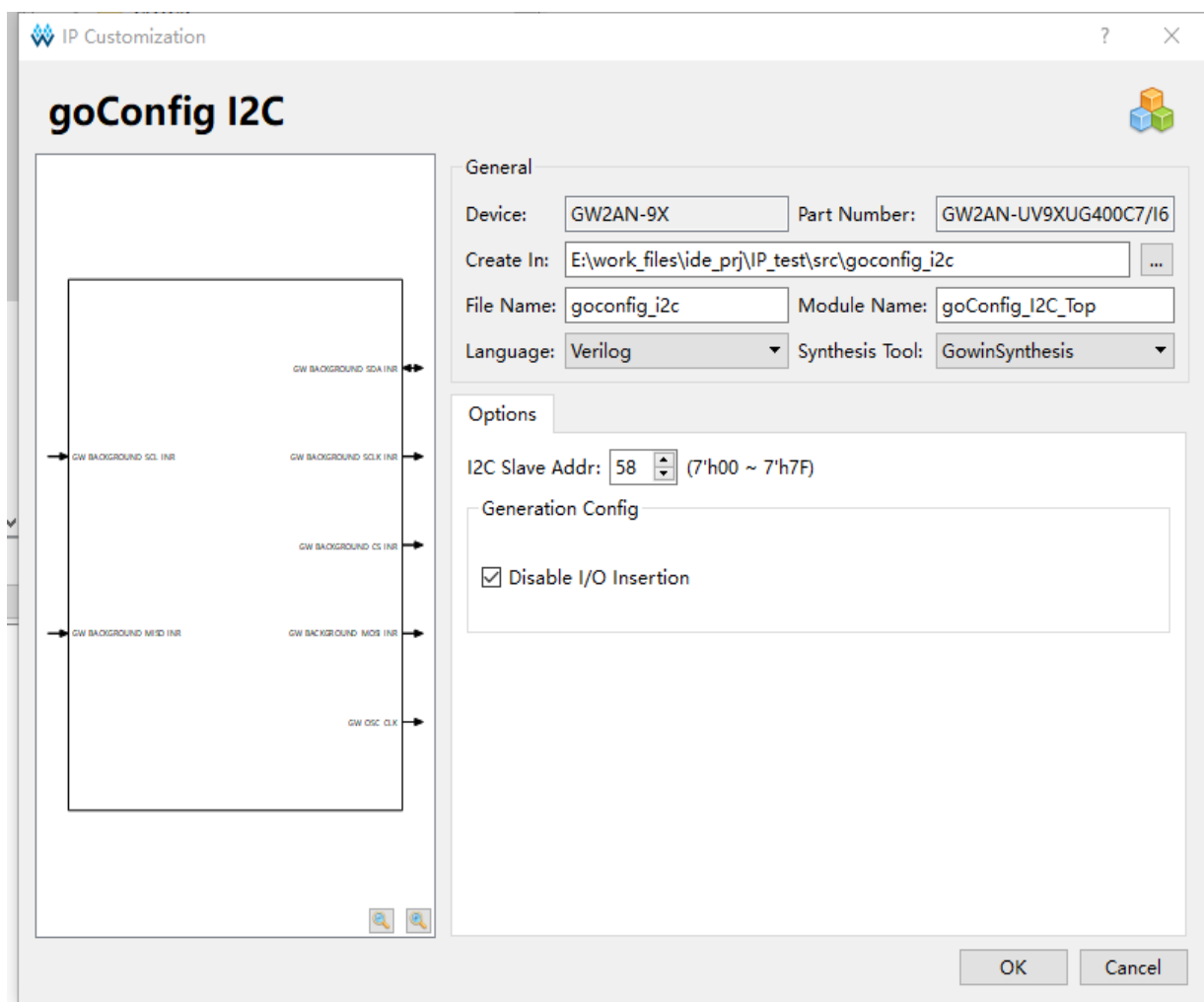


3. Gowin goConfig I2C IP Core Configuration Interface

The configuration interface of Gowin goConfig I2C IP is as shown in Figure 7-3. The interface diagram is on the left, and Options are on the right.

- You can configure the file name in File Name.
- You can configure the top module name in Module Name.
- You can configure the Slave address in I2C Slave Addr.

Figure 7-3 goConfig I2C IP Configuration Interface



8 Reference Design

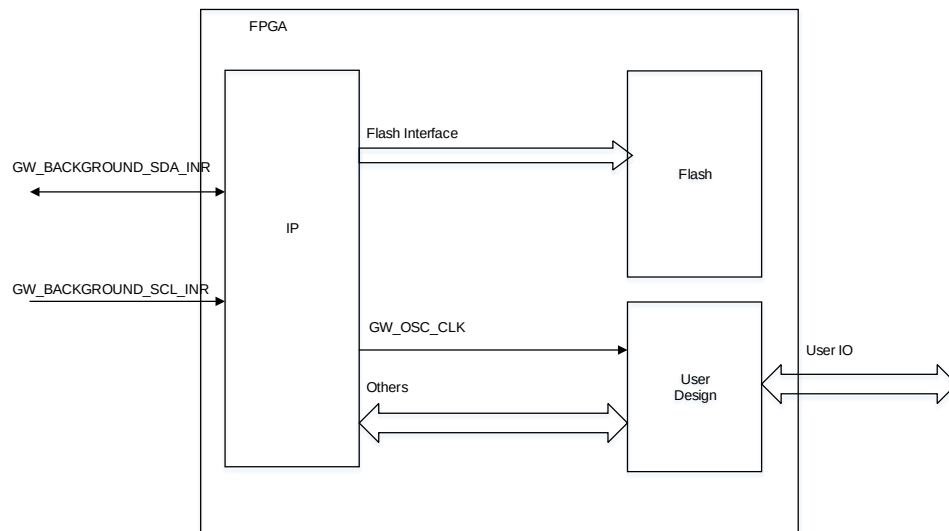
This chapter is intended to introduce the usage and structure of Gowin goConfig I2C IP reference design. Please see the Reference Design for details at Gowinsemi website.

8.1 Reference Design

8.1.1 FPGA Project Description

The FPGA reference design mainly includes two modules: IP module and user module.

- IP module: Call IP and provide an instance. The module descriptions are as follows:
 - You need modify Gowin Software configuration to use this IP, and there is an example in the reference design; see Chapter [5 Instructions for IP Use](#) for detailed descriptions.
 - There are some I/O constraints of the IP, you can see chapter [4 Signal Definition](#) and Notes in chapter [5 Instructions for IP Use](#) for the use details.
- User module: A simple LED test is used to represent the user design.

Figure 8-1 Reference Design Block Diagram

8.1.2 Reference Design File

The reference design file descriptions are shown below. For the 2C series, see Table 8-1; For 9X series, see Table 8-2.

Table 8-1 File List in src Folder for 2C Series

Name	Description
Top.v	Top-level module including LED test
goConfig_I2C_1p5c.cst	Project physical constraints file
goconfig_i2c	IP folder
----goconfig_i2c.v	Generated IP file, encrypted.
goconfig_i2c.vo	Generated IP netlist file for simulation

Table 8-2 File List in src Folder for 9X Series

Name	Description
goConfig_I2C_9X.cst	Project physical constraints file
LED_test	LED application file
----LED_TOP.v	Top module
----LED.v	LED test module
goconfig_i2c	IP folder
----goconfig_i2c.v	Generated IP file, encrypted.
----goconfig_i2c.vo	Generated IP netlist file for simulation

8.1.3 Notes

This reference design contains only FPGA project and some documentation; the simulation environment and the I²C bus driver example are not provided.

Please contact Gowin FAE for I²C bus driver example.

