Logos2 Family FPGA Configuration User Guide

(UG040005, V1.3)

(24.04.2023)

Shenzhen Pango Microsystems Co., Ltd.

Revisions History

Document Revisions

Version	Date of Release	Revisions
V1.3	24.04.2023	Initial release.

About this Manual

Terms and Abbreviations

Terms and Abbreviations	Meaning
JTAG	Joint Test Action Group
SED	Soft Error Detection
CCS	Configuration Control System
SPI	Serial Peripheral Interface
AES	Advanced Encryption Standard
SEU	Single Event Upsets
ECC	Error Control Coding
SECDED	Single Error Correcting and Double Error Detecting
UID	Unique ID
TAPC	Test Access Port Controller
IR	Instruction Register
TDR	Test Data Register

Table of Contents

Tables

Figures

Chapter 1 Overview

Configuration is the process of writing the user's design data (bitstream) into the internal memory of an FPGA. Configuration data can be actively acquired by the chip from an external Flash, or downloaded into the chip via an external processor/controller. Logos2 Family FPGAs use SRAM cells to store configuration data, which is lost upon power-down. Therefore, users need to reconfigure FPGAs upon each power-up.

Logos2 Family FPGAs support the following configuration modes:

- JTAG mode, compliant with IEEE 1149.1, IEEE 1149.6 and IEEE1532 standards
- Master SPI mode, supporting 1/2/4/8-bit data width
- Slave Parallel mode, supporting 8/16/32-bit data width modes
- Slave Serial mode, supporting 1-bit width

In addition, Logos2 Family FPGAs also provide the following functions:

- Watchdog, supporting time-out detection
- SEU 1-bit error correction and 2-bit error detection through the internal slave parallel interface
- Configuration bitstream version fallback in the Master SPI mode
- Configuration bitstream compression, which effectively reduces the bitstream size, storage space, and programming time
- Configuration bitstream encryption, which protects customers' design intellectual property against malicious copying
- JTAG mode, which provides a dedicated interface that supports internal debugging and boundary scan testing
- 96-bit UID protection, which writes a UID to each device before leaving the factory
- SHA digest, RSA-2048 authentication, and AES256-GCM self-authentication for digital signature verification of bitstreams, ensuring bitstream integrity
- Key storage methods, supporting eFuse and BB-RAM, of which the latter provides chip-level security protection
- Security protection technology, which prevents bitstream reverse reading
- JTAG security management, allowing users to permanently or temporarily disable the JTAG function
- Protection against DPA attacks, which prevents encrypted keys from being hacked

Chapter 2 Descriptions of Configuration Modes

Users can select the configuration mode for Logos2 Family FPGA by setting the value of MODE[2:0], as shown in Table 2-1. The JTAG configuration mode has the highest priority, and can be configured by setting MODE[2:0] to any value (except for 3’b000). The direction of configuration clock CFG_CLK is determined by the configuration mode: In the master mode, CFG_CLK is output by the FPGA to an external device storing the bitstream, such as Flash; in the slave mode, CFG_CLK is input by an external device (such as a microprocessor, CPLD, or another FPGA) to the FPGA.

Table 2-1 Logos2 Family FPGA Configuration Modes

Series Number	Configuration Mode	MODE[2:0]	Data Width	CFG_CLK Direction
1	JTAG	xxx[1]	1	Input (TCK)
2	Master SPI	001	1,2,4,8	Output
3	Slave Parallel	110	8,16,32	Input
4	Slave Serial	111	1	Input

Notes:

1. It is prohibited to set MODE[2:0]=3'b000, as no interface is available in this mode. Except for 3'b000, set MODE[2:0] to any other value to use the JTAG mode. To use the JTAG mode only, set MODE[2:0]=3'b101, under which only JTAG is available.

Table 2-2 shows the pins used in different configuration modes. The functions of some pins may vary with the configuration mode.

Table 2-2 Configuration Pins for Serial Mode

Pin	Bank	Type	JTAG	Master SPI				Slave Serial
				X1	X2	X4	X8	
SCBV	VCCIOCFG	Dedicated	SCBV	SCBV	SCBV	SCBV	SCBV	SCBV
MODE[2:0]	VCCIOCFG	Dedicated	xxx[1]	001	001	001	001	111
TCK	VCCIOCFG	Dedicated	TCK	TCK	TCK	TCK	TCK	TCK
TMS	VCCIOCFG	Dedicated	TMS	TMS	TMS	TMS	TMS	TMS
TDI	VCCIOCFG	Dedicated	TDI	TDI	TDI	TDI	TDI	TDI
TDO	VCCIOCFG	Dedicated	TDO	TDO	TDO	TDO	TDO	TDO
RSTN	VCCIOCFG	Dedicated	RSTN	RSTN	RSTN	RSTN	RSTN	RSTN
INIT_FLAG_N	VCCIOCFG	Dedicated	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N
CFG_DONE	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE
CFG_CLK	VCCIOCFG	Dedicated	-	CFG_CLK	CFG_CLK	CFG_CLK	CFG_CLK	CFG_CLK

Pin	Bank	Type	JTAG	Master SPI				Slave Serial
				X1	X2	X4	X8	
IO_STATUS_C	VCCIOL5	Multi-function	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C
ECCLKIN	VCCIOL5	Multi-function	-	ECCLKIN	ECCLKIN	ECCLKIN	ECCLKIN	-
FCS_N	VCCIOL5	Multi-function	-	FCS_N	FCS_N	FCS_N	FCS_N	-
FCS2_N	VCCIOL4	Multi-function	-	-	-	-	FCS2_N	-
CS_N	VCCIOL5	Multi-function	-	-	-	-	-	-
RWSEL	VCCIOL5	Multi-function	-	-	-	-	-	-
D[31:16]	VCCIOL5	Multi-function	-	-	-	-	-	-
D[15:8]	VCCIOL5	Multi-function	-	-	-	-	-	-
D[7:4]	VCCIOL5	Multi-function	-	-	-	-	D[7:4]	-
D[3:2]	VCCIOL5	Multi-function	-	-	-	D[3:2]	D[3:2]	-
MISO_D1_DI	VCCIOL5	Multi-function	-	MISO	D[1]	D[1]	D[1]	DI
MOSI_D0	VCCIOL5	Multi-function	-	MOSI	D[0]	D[0]	D[0]	-
CSO_DOUT	VCCIOL5	Multi-function	-	CSO_DOUT	-	-	-	CSO_DOUT

Note:

1. It is prohibited to set MODE[2:0]=3'b000, as no interface is available in this mode. Except for 3'b000, set MODE[2:0] to any other value to use the JTAG mode. To use the JTAG mode only, set MODE[2:0]=3'b101, under which only JTAG is available.

Table 2-3 Configuration Pins for Parallel Mode

Pin	Bank	Type	Slave Parallel		
			X8	X16	X32
SCBV	VCCIOCFG	Dedicated	SCBV	SCBV	SCBV
MODE[2:0]	VCCIOCFG	Dedicated	110	110	110
TCK	VCCIOCFG	Dedicated	TCK	TCK	TCK
TMS	VCCIOCFG	Dedicated	TMS	TMS	TMS
TDI	VCCIOCFG	Dedicated	TDI	TDI	TDI
TDO	VCCIOCFG	Dedicated	TDO	TDO	TDO
RSTN	VCCIOCFG	Dedicated	RSTN	RSTN	RSTN
INIT_FLAG_N	VCCIOCFG	Dedicated	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N
CFG_DONE	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE
CFG_CLK	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE
IO_STATUS_C	VCCIOL5	Multi-function	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C
ECCLKIN	VCCIOL5	Multi-function	-	-	-
FCS_N	VCCIOL5	Multi-function	-	-	-
FCS2_N	VCCIOL4	Multi-function	-	-	-
CS_N	VCCIOL5	Multi-function	CS_N	CS_N	CS_N
RWSEL	VCCIOL5	Multi-function	RWSEL	RWSEL	RWSEL
D[31:16]	VCCIOL5	Multi-function	-	-	D[31:16]

Pin	Bank	Type	Slave Parallel		
			X8	X16	X32
D[15:8]	VCCIOL5	Multi-function	-	D[15:8]	D[15:8]
D[7:4]	VCCIOL5	Multi-function	D[7:4]	D[7:4]	D[7:4]
D[3:2]	VCCIOL5	Multi-function	D[3:2]	D[3:2]	D[3:2]
MISO_D1_DI	VCCIOL5	Multi-function	D[1]	D[1]	D[1]
MOSI_D0	VCCIOL5	Multi-function	D[0]	D[0]	D[0]
CSO_DOUT	VCCIOL5	Multi-function	CSO_DOUT	CSO_DOUT	CSO_DOUT

defines the functions of configuration pins:

Table 2-4 Function Definitions of Configuration Pins

Pin	Bank	Type	Direction	Pin Description
SCBV	VCCIOCFG	Dedicated	Input	Selects the voltage of the Banks where the configurable multiplexed pins are located. This pin determines the voltage difference between BANKCFG and the voltage of the Banks containing the configurable multiplexed pins, as well as the I/O voltage operating ranges of these Banks. This pin always affects BANKCFG, but only affects the Banks containing the configurable multiplexed pins during configuration. When the voltage of VCCIOCFG is 2.5V or 3.3V, this pin must be directly connected to VCCIOCFG. When the voltage of VCCIOCFG is 1.8V or lower, this pin must be directly connected to the ground. Note: Use this pin in conjunction with the software, that is, keep the selection of SCBV in the bitstream settings consistent with the hardware settings! For the correspondence between the pull-up and pull-down levels of the SCBV pin and the power supply for configuring related Banks, refer to "UG040012_Logos2 Single Board Hardware Design Guide";
MODE[2:0]	VCCIOCFG	Dedicated	Input	An input pin dedicated to configuration, used for selecting the configuration mode; With a built-in weak pull-up resistor; To determine the mode, use a resistor of no greater than 1kΩ to pull this pin up to VCCIOCFG or a resistor of no greater than 1kΩ to pull this pin down to VSS.
TCK	VCCIOCFG	Dedicated	Input	A test clock input pin compliant with IEEE STD 1149.1 standard; Provides a clock and a testing clock input pin for FPGA's JTAG chain; With a built-in weak pull-up resistor to pull this pin up to VCCIOCFG.

Pin	Bank	Type	Direction	Pin Description
TMS	VCCIOCFG	Dedicated	Input	An input pin dedicated to JTAG test mode selection; With a built-in weak pull-up resistor to pull this pin up to VCCIOCFG.
TDI	VCCIOCFG	Dedicated	Input	A pin dedicated to JTAG test data input; With a built-in weak pull-up resistor (always enabled) to pull this pin up to VCCIOCFG.
TDO	VCCIOCFG	Dedicated	Output	A pin dedicated to JTAG test data output; With a built-in weak pull-up resistor (always enabled) to pull this pin up to VCCIOCFG.
RSTN	VCCIOCFG	Dedicated	Input	A dedicated configurable multiplexed pin with a built-in weak pull-up resistor, which is always active; It is active-low and used to reset configuration logic and configuration memory. When this pin is set to low, the FPGA configuration memory is cleared and a new configuration process begins. Configuration logic reset starts from the falling edge of this pin, and the configuration process starts from the subsequent rising edge of this pin. This pin should be pulled up to VCCIOCFG via an external resistor of no more than 4.7kΩ. Keeping this pin low during power-up will not reset the FPGA configuration logic. After configuration is complete, users can choose to configure it as weak pull-up or floating.
INIT_FLAG_N	VCCIOCFG	Dedicated	Bidirectional (open-drain)	A dedicated pin for initialisation and configuration status. When the output is low, it indicates that the FPGA chip is either performing initialisation (clearing the configuration memory) or a configuration error has occurred. With a built-in weak pull-up resistor that enables weak pull-up during configuration; Drive this pin to a low level after FPGA is powered up. Release the drive on this pin after FPGA is initialised. During power-up and initialisation, this pin can be driven externally to a low level to delay the configuration process after initialisation. When detecting a high-level input to this pin after initialisation, the chip begins the configuration process. During the configuration process, this pin indicates the configuration error status. When it is low, it indicates an error occurred during configuration; This pin should be pulled up to VCCIOCFG via an external weak pull-up resistor (recommended resistance: no more than 4.7kΩ). After configuration is complete, users can choose to configure it as weak pull-up or floating.

Pin	Bank	Type	Direction	Pin Description
CFG_DONE	VCCIOCFG	Dedicated	Bidirectional (open-drain)	A dedicated pin for configuration status: With a built-in weak pull-up resistor (about 10kΩ); It indicates configuration completion status. When it is high, it indicates configuration is complete. It defaults to open-drain output mode. When the FPGA is powered up, this pin is driven to a low level before or during the configuration process. Once all configuration data is correctly received and the wakeup timing is initiated, release control of this pin. After configuration is complete, this pin can continue to be driven to a low level externally. Once the internal wakeup timing detects that the external pin DONE is at a low level, the internal wakeup circuit will remain active until the external pin switches to a high level before proceeding with the wakeup process. This pin should be pulled up to VCCIOCFG via an external weak pull-up resistor (recommended resistance: no more than 4.7kΩ). After configuration is complete, users can choose to configure it as weak pull-up or floating.
CFG_CLK	VCCIOCFG	Dedicated	Input/ Output	A configuration clock pin. Except for the JTAG configuration mode, other modes adopt this clock to synchronise the FPGA configuration process. Under Slave Serial and Slave Parallel modes, this pin acts as a clock input to retrieve configuration data from external sources; Under Master SPI configuration mode, this pin acts as a clock output to retrieve configuration data from external sources, and should be connected to an external pull-up resistor of 1kΩ; When this clock is not required (for example in JTAG mode), put this pin in a high-impedance state. After configuration is complete, users can choose to configure it as weak pull-up or floating.
IO_STATUS_C	VCCIOL5	Multi-function	Input	A multi-function pin that inputs signals to control whether the weak pull-up resistors of all user IOs (including the multiplexing IOs) are enabled or not from the completion of power-up to the entry into user mode. (1) "0": enabling all the internal pull-up resistors of user IOs. (2) "1": disabling all the internal pull-up resistors of user IOs. (3) It is recommended to connect this pin to the corresponding VCCIO via an external weak pull-up resistor. (4) This pin can be connected to the corresponding VCCIO or VSS directly or through a resistor of no more than 1kΩ.

Pin	Bank	Type	Direction	Pin Description
				(5) Do not leave this pin unused before or during configuration;
ECCLKIN	VCCIOL5	Multi-function	Input	A master mode configuration clock for external input. This is an optional external clock that inputs signals to the configuration logic. (1) In master mode (Master SPI), the FPGA can choose this input clock as the configuration clock for the configuration logic. (2) In other configuration modes or user mode, or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state.
FCS_N	VCCIOL5	Multi-function	Output	A configurable multiplexed pin, used in the external Master SPI configuration mode. (1) In the Master SPI X1/X2/X4 mode, this pin outputs a chip select signal to the external Flash when it is active-low, and should be connected to VCCIO of the Bank where the pin is located via an external pull-up resistor (no more than 4.7kΩ). (2) In other configuration modes or during initialisation, this pin acts as a regular user I/O in a high-impedance or weak pull-up state. (3) After configuration is complete, this pin can be used as a user I/O.
FCS2_N	VCCIOL4	Multi-function	Output	A configurable multiplexed pin, used in the external Master SPI X8 configuration mode. (1) In the Master SPI X8 mode, this pin outputs a chip select signal to the external Flash when it is active-low, and should be connected to VCCIO via an external pull-up resistor of no more than 4.7kΩ. (2) In other configuration modes or user mode, or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (3) After configuration is complete, this pin can be used as a user I/O.
CS_N	VCCIOL5	Multi-function	Input	A configurable multiplexed pin, active-low and inputs chip select signals. (1) When it is low-level, this pin enables the Slave Parallel mode configuration interface. In Slave Parallel configuration mode, the external controller can control this pin to select the FPGA's Slave Parallel bus that is about to operate, or connect this pin to the CSO_DOUT pin of the previous FPGA in the Slave Parallel configuration chain. (2) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (3) After configuration is complete, this pin can be used as a user I/O.

Pin	Bank	Type	Direction	Pin Description
RWSEL	VCCIOL5	Multi-function	Input	A configurable multiplexed pin, for selecting the read/write operation of the Slave Parallel configuration mode (high for read and low for write). (1) When this pin is high-level, the Slave Parallel configuration mode reads data from the data bus; (2) When this pin is low-level, the Slave Parallel configuration mode writes data to the data bus; (3) The read and write operations can be switched only when CS_N is at a high level. (4) After configuration is complete, this pin can be used as a user I/O. (5) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state.
D[31:16]	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin (1) In Slave Parallel mode, this pin is X32-bit width, as D[31:16] for the data bus. (2) In other configuration modes or conditions with insufficient bit width, or during initialisation, this pin is not used and acts as a regular user I/O in a high-impedance or weak pull-up state. (3) After configuration is complete, this pin can be used as a user I/O.
D[15:8]	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin (1) In Slave Parallel mode, this pin is X16- or X32-bit width, as D[15:8] for the data bus. (2) In other configuration modes or conditions with insufficient bit width, or during initialisation, this pin is not used and acts as a regular user I/O in a high-impedance or weak pull-up state. (3) After configuration is complete, this pin can be used as a user I/O.
D[7:4]	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin: (1) In Master SPI X8 mode, this pin is connected to the second Flash in the same way as D[3:0]. (2) In Slave Parallel mode, this pin acts as D[7:4] for the data bus. (3) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (4) After configuration is complete, this pin can be used as a user I/O.

Pin	Bank	Type	Direction	Pin Description
D[3:2]	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin (1) In the Master SPI X4/X8 configuration mode, this pin is connected to a data output port of the SPI flash as data input. "D2" is connected to the 3rd-bit data output of the external SPI Flash (e.g. DQ2, W#, WP#, or IO2) and "D3" is connected to the 4th-bit data output of the external SPI Flash (e.g. DQ3, HOLD#, or IO3). This pin should be connected to its corresponding VCCIO via an external weak pull-up resistor of 4.7kΩ. (2) In Slave Parallel mode, this pin acts as D[3:2] for the data bus. (3) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (4) After configuration is complete, this pin can be used as a user I/O.
MISO_D1_DI	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin (1) In Master SPI X1 mode, "MISO" is connected to a data output port of the SPI flash (such as DQ1, Q, SO, or IO1) as data input. (2) In Master SPI X2/X4/X8 mode, "D1" is connected to the second serial data output port of the SPI flash (such as DQ1, Q, SO, or IO1). (3) In Slave Parallel mode, this pin acts as D[1] for the data bus. (4) In Slave Serial mode, "DI" acts as the data input pin. (5) In other configuration modes and during initialisation, this pin serves as a general user I/O in a high-impedance or weak pull-up state. In other configuration modes such as JTAG, the state of this pin will be ignored. (6) After configuration is complete, this pin can be used as a user I/O.

Pin	Bank	Type	Direction	Pin Description
MOSI_D0	VCCIOL5	Multi-function	Input/ Output	A configurable multiplexed data pin; (1) "MOSI": This pin outputs serial data in Master SPI X1 mode, and is connected to a data input pin of the external SPI Flash (such as DQ0, D, SI, or IO0). When commands and addresses are sent to the external SPI Flash, this pin's output is in a high-impedance or weak pull-up state. (2) In Master SPI X2/X4/X8 mode, the bidirectional data port outputs commands and addresses to the external SPI Flash. The lowest bit data input receives data from the external SPI Flash. This pin is connected to a bidirectional data pin of the external SPI Flash (such as DQ0, D, SI, or IO0). (3) "D0": This pin is used as D[0] for the data bus in Slave Parallel mode; (4) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (5) After configuration is complete, this pin can be used as a user I/O.
CSO_DOUT	VCCIOL5	Multi-function	Output (open-drain)/ output	A configurable multiplexed pin, which is only used for cascading. (1) In Master SPI X1 mode, this pin can output cascaded data; in other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (2) In Slave Serial configuration mode, this pin can output cascaded data; in other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (3) Under Slave Parallel cascade configuration mode, it can be connected to the CS_N pin of a downstream device as an open-drain output for the chip select signal, and connected to its corresponding VCCIO via an external 330Ω pull-up resistor. In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state. (4) In other configuration modes or during initialisation, this pin is not used but acts as a regular user I/O in a high-impedance or weak pull-up state.

2.1 JTAG Configuration Mode

Figure 2-1 shows the application interface for the JTAG configuration mode.

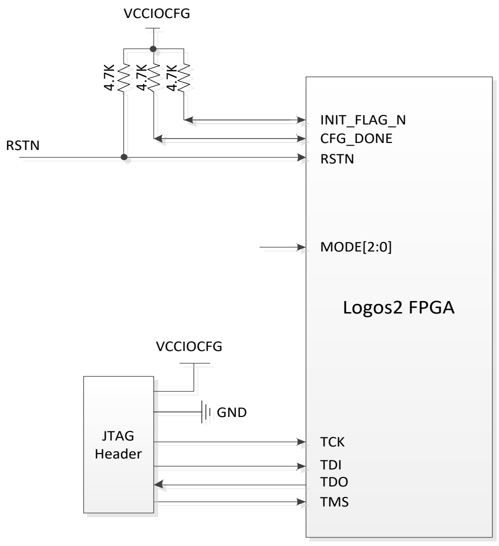


Figure 2-1 JTAG Mode Application Diagram

An RSTN low-level signal will reset the configuration logic. It is recommended to pull the signal up to VCCIOCFG (3.3V) via an external 4.7kΩ resistor. During initialisation process, if INIT_FLAG_N is used as input and connected to a low level, the Logos 2 FPGA will remain in the initialisation stage. It is recommended to pull the signal up to VCCIOCFG via an external 4.7kΩ resistor. When used as an output, INIT_FLAG_N outputs a high level to indicate the end of chip initialisation. If CFG_DONE outputs a high level, it indicates that the chip has entered user mode. Inputting an external low level will keep the Logos2 FPGA in the configuration stage. It is recommended to pull the signal up to VCCIOCFG via an external 4.7kΩ resistor.

In the JTAG mode, the TCK should be provided externally; the external can control the transition of the JTAG internal TAP state machine by changing the state of TMS, to select configuration bitstream writing (TDI) or on-chip data readback (TDO). In addition to configuration programming, the JTAG interface is also commonly used for internal debugging and boundary scan testing. In the JTAG mode, it is recommended to pull up TDI, TCK, and TMS to VCCIOCFG via a 12–15kΩ resistor to provide a stable initial input level.

Figure 2-2 shows the typical timing for the JTAG programming mode.

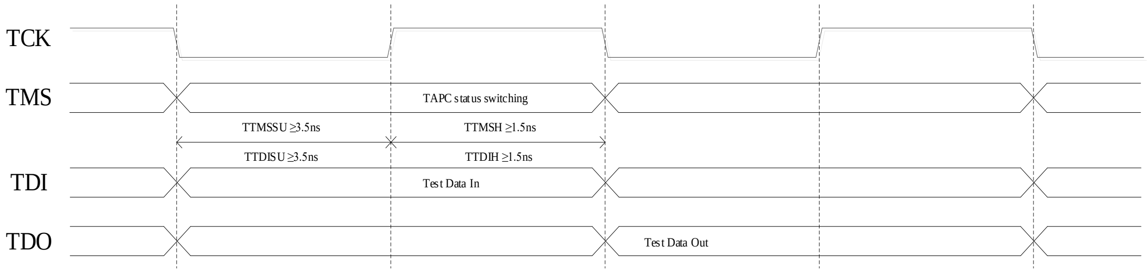


Figure 2-2 Typical Timing for JTAG Programming

As the JTAG interface can be used for programming eFuse, it is recommended to lock all eFuse functional bits after confirming the chip application scenario, so as to avoid environmental interference signals on the JTAG interface from causing erroneous eFuse programming. For details, refer to the eFuse introduction.

2.1.1 JTAG Cascade Mode

Multiple devices can be connected using a JTAG daisy chain, as shown in Figure 2-3.

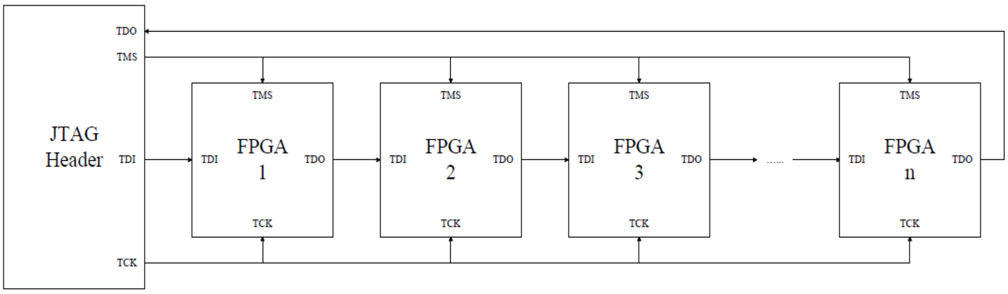


Figure 2-3 JTAG Cascading Application Diagram

In this mode, the mode pin should be set to the JTAG mode. After connecting the download cable, the PDS software on the host will scan all the devices on the JTAG chain, and users can choose to program and download the corresponding FPGA.

The TCK and TMS signals connect all the devices on the JTAG chain, so their quality will affect the maximum frequency and reliability of JTAG configuration. Note: During JTAG cascade, only the external Flash of the first-stage FPGA can be operated.

2.1.2 JTAG Boundary Scan Architecture

The device supports the IEEE1149.1 standard, and the JTAG boundary scan architecture is as shown in Figure 2-4 .

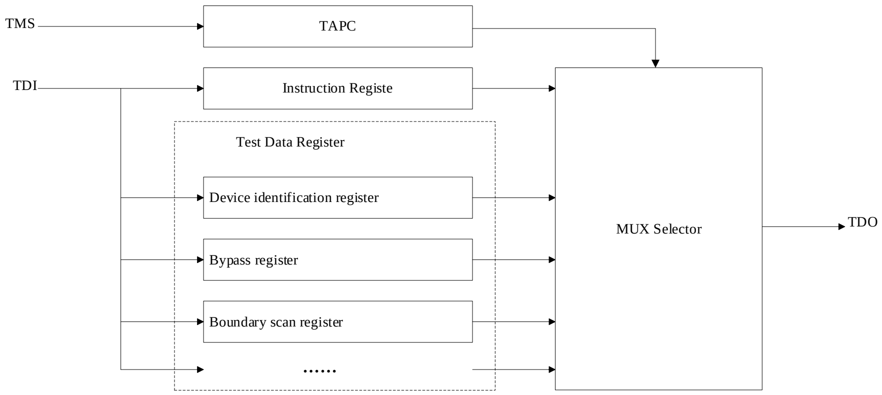


Figure 2-4 Block Diagram of JTAG Boundary Scan System

The boundary scan hardware architecture mainly consists of: Test Access Port Controller (TAPC), Instruction Register (IR), and Test Data Register (TDR).

The TAPC implements the state switching control as defined by the IEEE1149.1 standard. For the functions and usage of the IR and TDR, as well as the arrangement of boundary registers in various FPGA devices, refer to the BSDL (*.bsm) file in the PDS installation directory.

Table 2-5 lists some commonly used instructions supported by IR and their descriptions.

Table 2-5 Common JTAG Instruction Set

Instruction	Type	Op Code	Description
BYPASS	1149.1 Non-test instruction	1111111111	Bypass instruction
SAMPLE/PRELOAD	1149.1 Non-test instruction	1010000000	Sample/preload instruction
EXTEST	1149.1 Test instruction	1010000001	External test instruction
IDCODE	1149.1 Non-test instruction	1010000011	Identification instruction
HIGHZ	1149.1 Test instruction	1010000101	High-Z instruction
JRST	Only for design	1010001010	Reset instruction
CFGI	Only for design	1010001011	Configuration instruction
CFGO	Only for design	1010001100	Readback instruction
JWAKEUP	Only for design	1010001101	Wakeup instruction

2.2 Master SPI Configuration Mode

Figure 2-5 shows the interface for the Master SPI configuration mode:

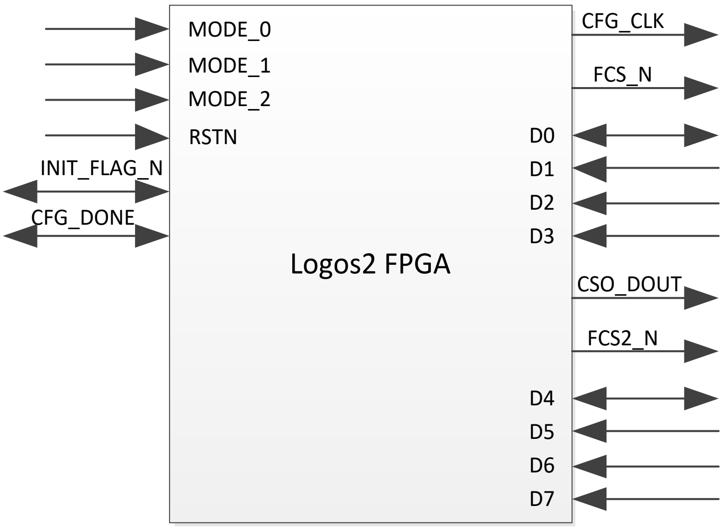


Figure 2-5 Interface for the Master SPI Configuration Mode

Table 2-6 lists the signal descriptions of the interface for the Master SPI configuration mode.

Table 2-6 Master SPI Interface Signal Descriptions

Pin	Direction	Type	Pin Description
RSTN	Input	Dedicated	Asynchronous full-chip reset signal, active low
CFG_CLK	Output	Dedicated	Configuration clock
MODE[2:0]	Input	Dedicated	Configuration mode pin 001: Master SPI configuration mode (x1/x2/x4/x8)
INIT_FLAG_N	Bidirectional (open-drain)	Dedicated	Before sampling pins MODE[2:0], INIT_FLAG_N is an input, and configuration can be delayed by keeping it at a low level. After sampling pins MODE[2:0], INIT_FLAG_N is open-drain, indicating whether an error occurred in the configuration process. 0: Wrong 1: Correct
FCS_N	Output	Multi-function	SPI Flash chip select signal, active-low
FCS2_N	Output	Multi-function	Second SPI Flash chip select signal in the x8 mode, active-low
D[7:0]	Bidirectional	Multi-function	Serial input-output data bus, sampling data from the rising or falling edge of CFG_CLK D[7:4] is the serial input-output data bus for the second SPI Flash in the x8 mode
CSO_DOUT	Output	Multi-function	Daisy chain data output, transmitted on the falling edge of CFG_CLK
CFG_DONE	Bidirectional (open-drain)	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA configured

In the Master SPI mode, bitstreams are usually stored in the external SPI Flash. To select the Master SPI mode, it is recommended to connect MODE[2:1] to the ground via a pull-down resistor and MODE[0] to the VCCIOCFG power supply via a pull-up resistor. Power FPGAs up or apply a low-level pulse to RSTN to start programming. Then Logos2 family FPGAs will actively read the bitstream from the external Flash, and the CCS will automatically fetch data from address 0 of serial NOR FLASH in fast read X1 mode, sampling data from the rising edge. After fetching the mode setting instruction at the beginning of the bitstream, CCS will switch to the mode specified by the instruction, sampling on the rising or falling edge. Once all the bitstreams are fetched from the Flash, FCS_N (FCS2_N) will be set to 1, and operation in the Master SPI mode is completed.

If the CFG_CLK frequency is set to not exceed 15.38MHz, use rising edge sampling; if it is set to exceed 15.38MHz, use falling edge sampling.

CFG_CLK is generated and output by the chip internally; monitor pins INIT_FLAG_N and CFG_DONE to judge whether programming is completed; Master SPI supports 1/2/4/8-bit data width modes, which can be set through bitstream interpretation. For different data widths, the data bit selection and data input/output direction will be different, as described below:

In the x1 bit width mode, the FPGA's pin D[0] is connected to the data input port of the SPI Flash as a command output, and pin D[1] is connected to the data output port of the SPI Flash as a data input. The application diagram is shown in Figure 2-6:

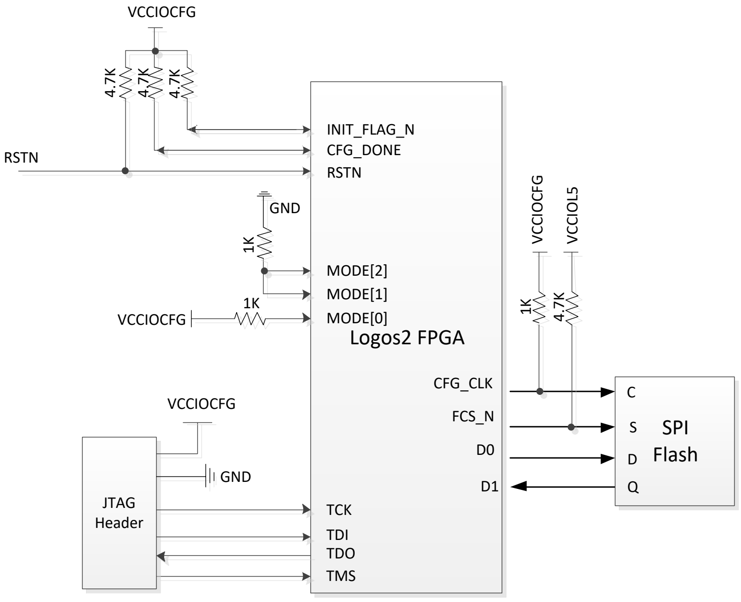


Figure 2-6 Application Diagram of the Master SPI X1 Configuration Mode

When the bit width is X2, pins D[1:0] serve as the data buses. The application diagram is shown in Figure 2-7:

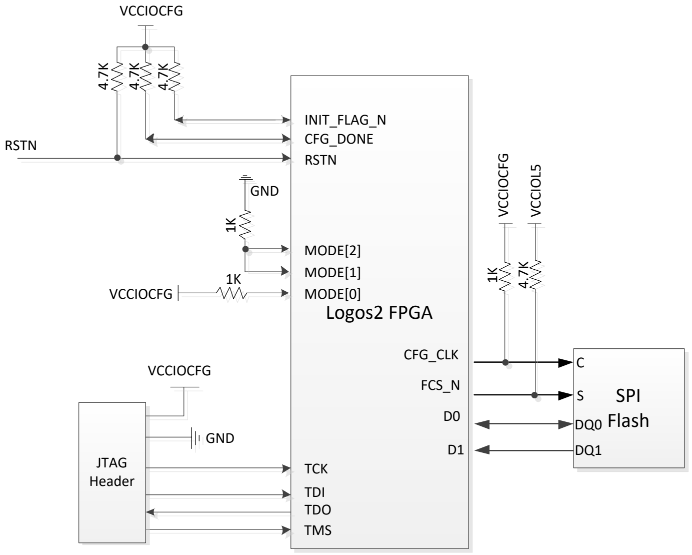


Figure 2-7 Application Diagram of the Master SPI X2 Configuration Mode

When the bit width is X4, pins D[3:0] serve as the data buses. The application diagram is shown in Figure 2-8: DQ[3:2] must be provided with a pull-up resistor.

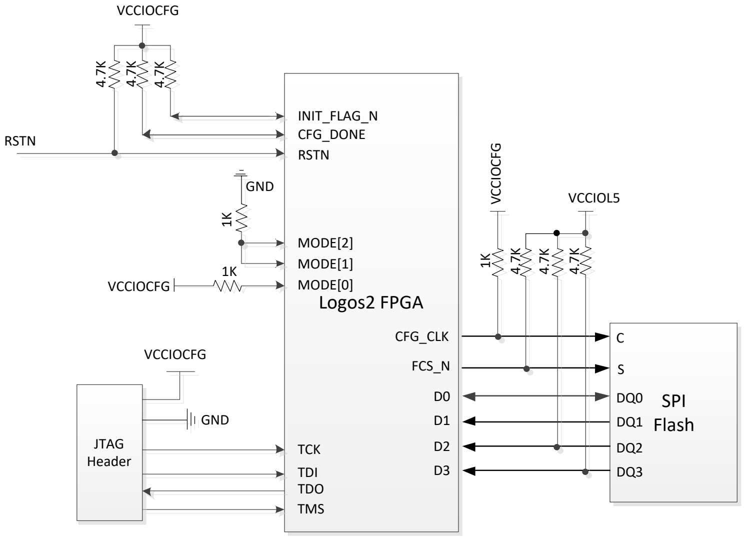


Figure 2-8 Application Diagram of the Master SPI X4 Configuration Mode

When the bit width is X8, pins D[3:0] are the data buses for the first SPI Flash, and pins D[7:4] are the data buses for the second SPI Flash. The application diagram is shown in Figure 2-9:

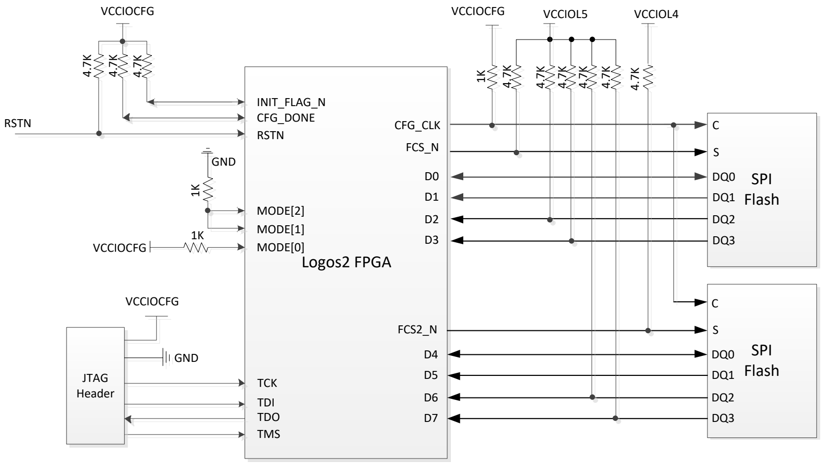


Figure 2-9 Application Diagram of the Master SPI X8 Configuration Mode

Figure 2-10 shows the typical timing for the Master SPI programming mode.

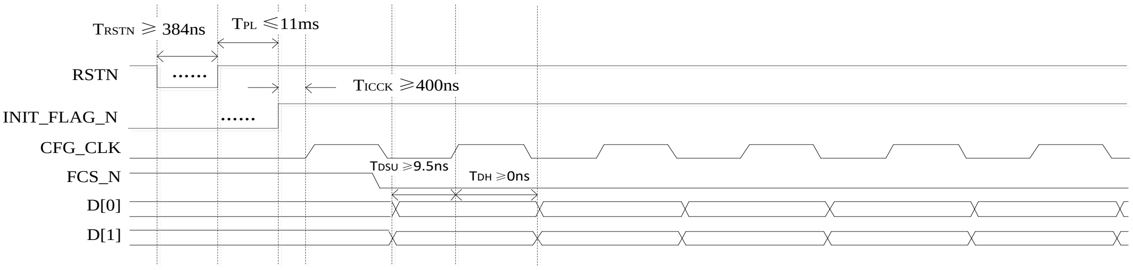


Figure 2-10 Typical Timing for 1bit Master SPI Programming

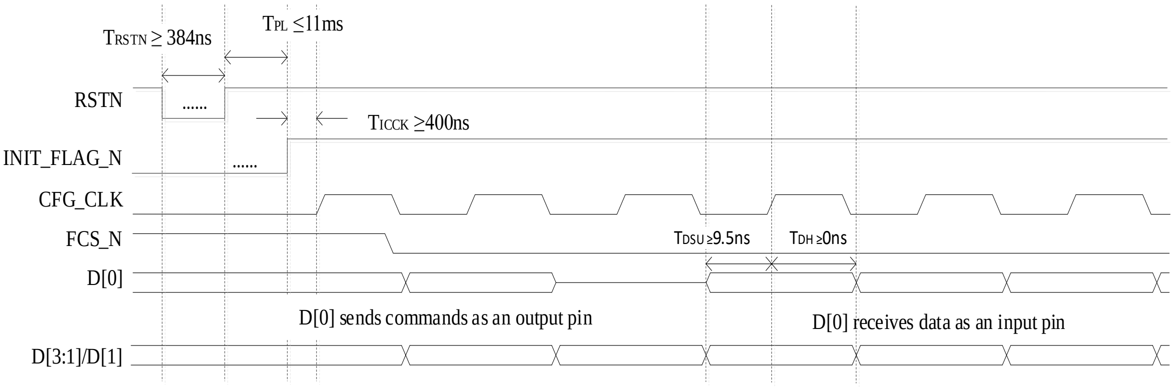


Figure 2-11 Typical Timing for 2/4bit Master SPI Programming

Table 2-7 lists some SPI Flash models supported by the Logos2 Family FPGAs.

Table 2-7 Supported SPI Flash Models

Model	Manufacturer	Density
N25Q32	Micron	32Mb
N25Q64	Micron	64Mb
N25Q128	Micron	128Mb
N25Q256	Micron	256Mb
N25Q512	Micron	512Mb
W25Q80	WINBOND	8Mb
W25Q16	WINBOND	16Mb
W25Q32	WINBOND	32Mb
W25Q64	WINBOND	64Mb
W25Q128	WINBOND	128Mb
W25Q256	WINBOND	256Mb

Note:

To select a Flash, consider the actual storage space requirements, such as the actual bitstream size. (Refer to Table 3-3 for the ordinary bitstream size of Logos2 device.) The above Flash models have passed tests. To use another Flash model not included in the table above, it must be tested first.

2.2.1 Master SPI Serial Daisy Chain

The Master SPI configuration mode supports SPI serial daisy chain configuration using both Master SPI and Slave Serial modes. In the SPI serial daisy chain, the first device uses the Master SPI mode, while other devices use the Slave Serial mode. The bitstream needs to be generated by clicking [Operations > Generate Daisy Chain File] in the Fabric Configuration tool. Configure from the last device to the first one in the SPI serial daisy chain. The logic block diagram is shown in Figure

2-12:

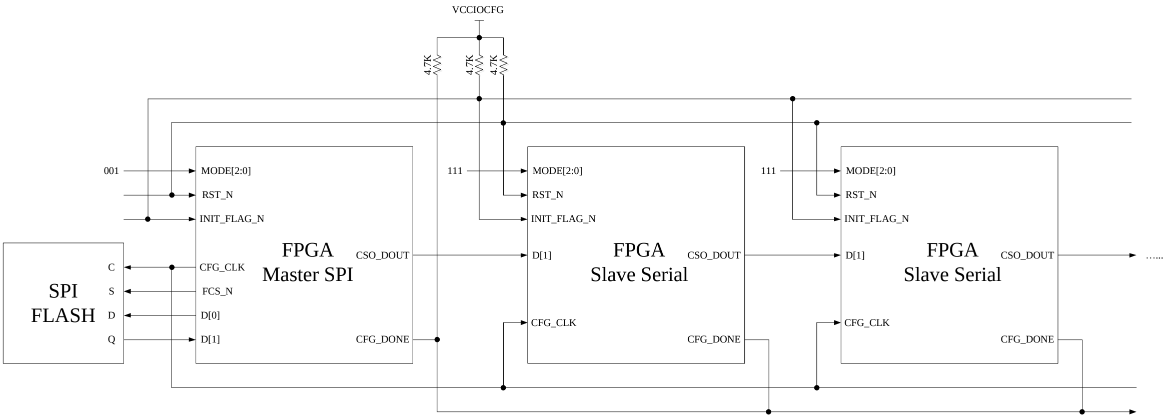


Figure 2-12 Application Diagram of the Master SPI Serial Daisy Chain

2.3 Slave Serial Configuration Mode

Figure 2-13 shows the interface for the Slave Serial configuration mode:

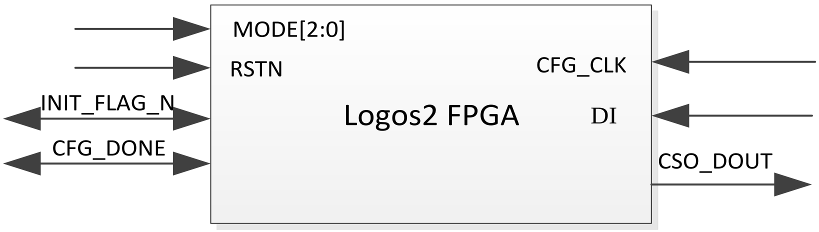


Figure 2-13 Slave Serial Configuration Interface

Table 2-8 lists the signal descriptions of the interface for the Slave Serial configuration mode.

Table 2-8 Slave Serial Configuration Interface Signal Descriptions

Pin	Direction	Type	Pin Description
RSTN	Input	Dedicated	Asynchronous full-chip reset signal, active low
CFG_CLK	Input	Dedicated	Configuration clock
MODE[2:0]	Input	Dedicated	Configuration mode pin 111: Slave Serial Mode
INIT_FLAG_N	Bidirectional (open-drain)	Dedicated	Before sampling pins MODE[2:0], INIT_FLAG_N is an input, and configuration can be delayed by keeping it at a low level. After sampling pins MODE[2:0], INIT_FLAG_N is open-drain, indicating whether an error occurred in the configuration process. 0: Wrong 1: Correct

Pin	Direction	Type	Pin Description
CFG_DONE	Bidirectional (open-drain)	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA configured
DI	Input	Multi-function	Serial input data bus, sampling data from the rising edge of CFG_CLK for self-configuration, and sampling data from the falling edge during cascade for the next-level configuration.
CSO_DOUT	Output	Multi-function	Daisy chain data output, transmitted on the falling edge of CFG_CLK

To select the Slave Serial mode, it is recommended to connect pins MODE[2:0] to VCCIOCFG via a pull-up resistor. The application diagram of the Slave Serial configuration mode is shown in Figure 2-14:

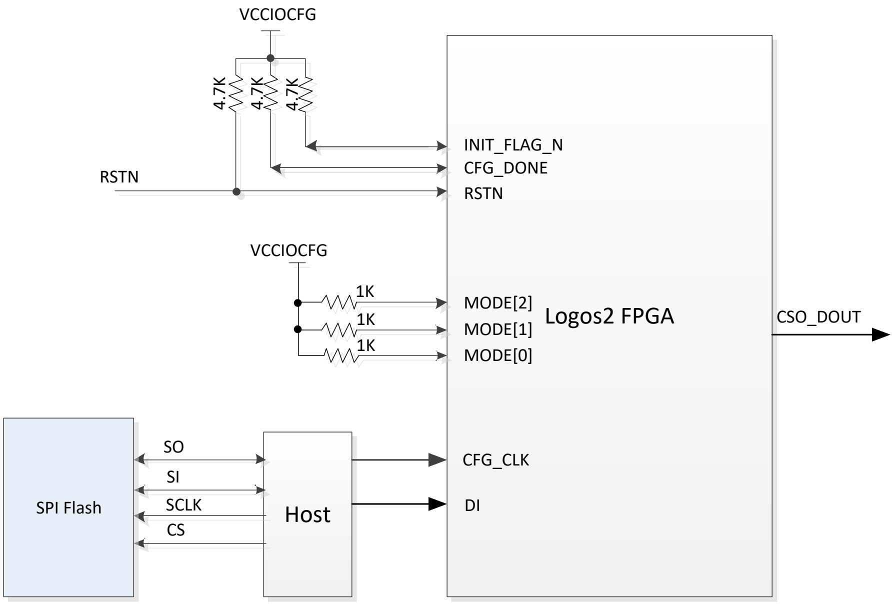


Figure 2-14 Application Diagram of the Slave Serial Mode

In the Slave Serial configuration mode, users can control the power-up, start, and data loading of multiple chips on the board via a master control chip (Host). The Host can be a microprocessor, CPLD, or another FPGA. In this mode, power the device up or apply a low-level pulse to RSTN to start programming. Monitor pins INIT_FLAG_N and CFG_DONE to judge whether programming is completed.

When the Host sends a bitstream to the Logos2 device, if wait PLL Lock is not enabled (disabled by default), the device will release control of CFG_DONE during a clock cycle between the 100 NOP type 1 packet headers at the end of the bitstream, pulling CFG_DONE high via the external pull-up resistor. After CFG_DONE is pulled high, the bitstream provides a clock for waking up the device, so it is necessary to ensure that the bitstream is completely transmitted before terminating the clock. If wait is enabled, the clock cannot be terminated before CFG_DONE is pulled high, and at least 100 clocks must be provided after CFG_DONE is pulled high.

When using serial daisy-chain configuration, each-stage device will sample DI from the rising edge of CFG_CLK for its own configuration and also sample DI from the falling edge to output configuration for the next level. Therefore, during cascaded configuration, the Host needs to consider both the rising and falling edges to meet the setup and hold time requirements. Figure 2-15 shows the typical timing for the Slave Serial programming mode.

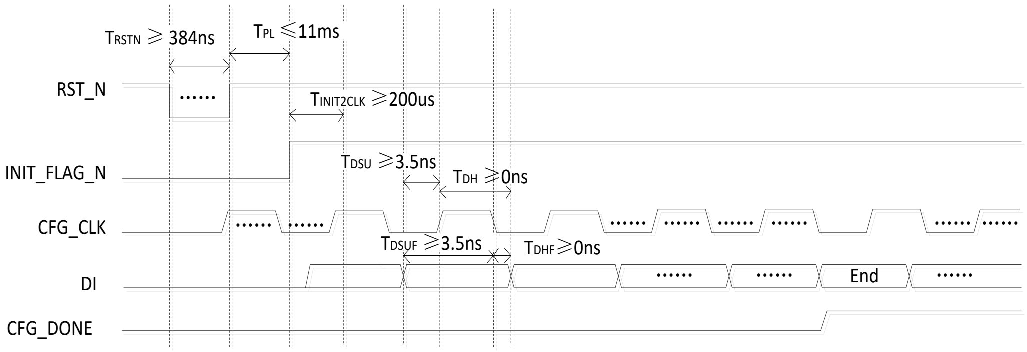


Figure 2-15 Typical Timing for Slave Serial Programming

2.3.1 Slave Serial Daisy Chain

Serial daisy chain is a typical application of the Slave Serial configuration mode. All devices can be configured in a serial cascade using the Slave Serial mode. Configure from the last (furthest) device to the first one in the chain during cascade configuration. The bitstream needs to be generated by clicking [Operations > Generate Daisy Chain File] in the Fabric Configuration tool.

Through serial cascade, users can control the configuration of multiple cascaded chips on the board via a master control chip (Host). Connect the chips as shown in Figure 2-16.

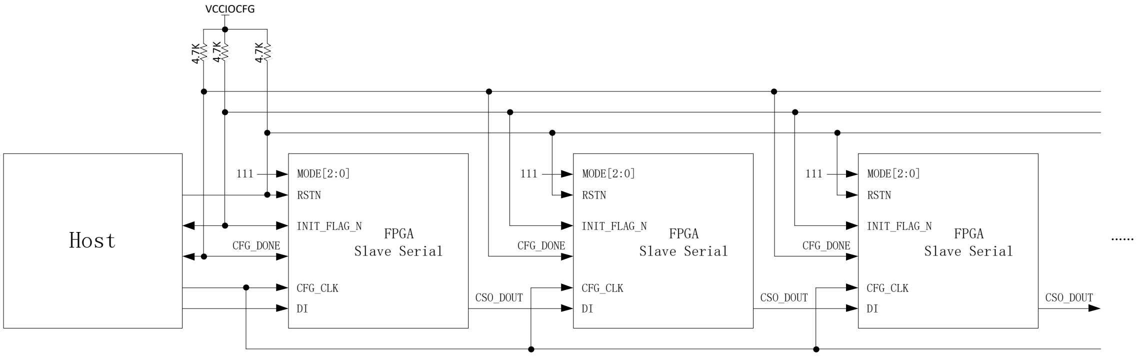


Figure 2-16 Serial Daisy Chain Connection in the Slave Serial Configuration Mode

When adopting the serial daisy chain, be sure to:

1. Meet the primary stage timing
2. As shown in Figure 2-15, the timing sent by the Host must meet the setup and hold time requirements for sampling DI from the rising and falling edges of CFG_CLK.
3. Meet the subsequent stage timing
4. Adjust the Host output to meet the primary stage timing. The subsequent stage inputs all come from the previous stage, with adjustments mainly involving delays introduced by PCB traces, etc. As shown in Figure 2-17, it is necessary to adjust delay parameters $T_{CLKDELAY}$ and $T_{DOUTDELAY}$ to meet the setup and holding time requirements of the subsequent stage devices for both edges.

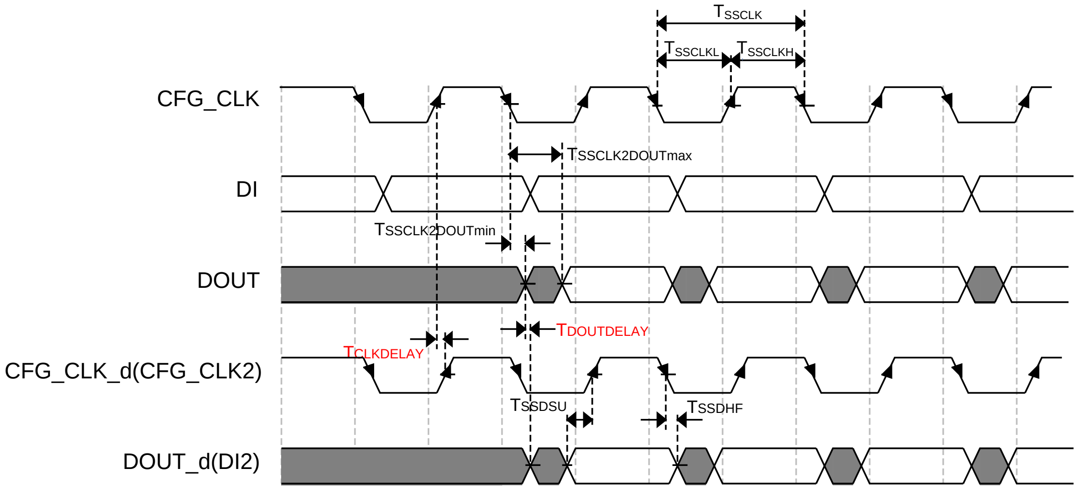


Figure 2-17 Timing Adjustment Example–Subsequent Stage

Note:

For TSSCLK2DOUT, TSSCLKL, TSSCLKH, TSSDSU, and TSSDHF, refer to "DS04001_Logos2 Family FPGA Device Datasheet".

The above figure shows that if the setup time for the rising edge is met when sending data on the falling edge, then the setup time for the falling edge will certainly be met; if the hold time for the falling edge is met, then the hold time for the rising edge will also be met. Therefore, $\mathrm{T_{CLKDELAY}}$ and $\mathrm{T_{DOUTDELAY}}$ must satisfy the following formula (unit: ns):

$$\begin{cases} T_{\mathrm{SSDSU}} = (T_{\mathrm{SSCLK}} + T_{\mathrm{CLKDELAY}}) - (T_{\mathrm{SSCLKH}} + T_{\mathrm{SSCLK2DOUTmax}} + T_{\mathrm{DOUTDELAY}}) \geq 3.5 \\ T_{\mathrm{SSDHF}} = T_{\mathrm{SSCLK2DOUTmin}} + T_{\mathrm{DOUTDELAY}} - T_{\mathrm{CLKDELAY}} \geq 0 \end{cases}$$

i.e.,

$$10.5 - T_{\mathrm{SSCLKL}} \leq T_{\mathrm{CLKDELAY}} - T_{\mathrm{DOUTDELAY}} \leq 2$$

2.4 Slave Parallel Configuration Mode

Figure 2-18 shows the interface for the Slave Parallel configuration mode:

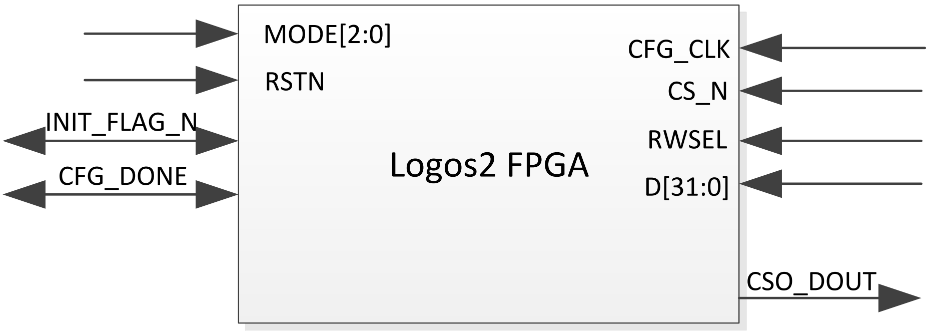


Figure 2-18 Slave Parallel Configuration Interface

Table 2-9 lists the signal descriptions of the interface for the Slave Parallel configuration mode.

Table 2-9 Slave Parallel Configuration Interface Signal Descriptions

Pin	Direction	Type	Pin Description
RSTN	Input	Dedicated	Asynchronous full-chip reset signal, active low
CFG_CLK	Input	Dedicated	Configuration clock
MODE[2:0]	Input	Dedicated	Configuration mode pin 110: Slave Parallel Mode
INIT_FLAG_N	Bidirectional (open-drain)	Dedicated	Before sampling pins MODE[2:0], INIT_FLAG_N is an input, and configuration can be delayed by keeping it at a low level. After sampling pins MODE[2:0], INIT_FLAG_N is open-drain, indicating whether an error occurred in the configuration process. 0: Wrong 1: Correct
CFG_DONE	Bidirectional (open-drain)	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA configured
CS_N	Input	Multi-function	External parallel interface chip select signal, active low. Sampling data from the rising edge of CFG_CLK.[1]
RWSEL	Input	Multi-function	External parallel interface read/write control signal. Sampling data from the rising edge of CFG_CLK. 0: Write 1: Read
D[31:0]	Bidirectional	Multi-function	Parallel data bus, sampling data from the rising edge of CFG_CLK and transmitting data
CSO_DOUT	Output	Multi-function	Daisy chain data output, transmitted on the falling edge of CFG_CLK

Notes:

1. CS_N is a synchronous signal. After INIT_FLAG_N is pulled high, before configuring the chip, CS_N must be kept high and provided at least 8 clock cycles for CFG_CLK to ensure the interface is in an initial operating state.

In the Slave Parallel mode, users can control the power-up and data loading of multiple chips on the board via a master control chip (Host). The Host can be a microprocessor, CPLD, or another FPGA. To select Slave Parallel mode, it is recommended to connect MODE[0] to the ground through a pull-down resistor and MODE[2:1] to the VCCIOCFG power supply through a pull-up resistor. The application diagram is shown in Figure 2-19:

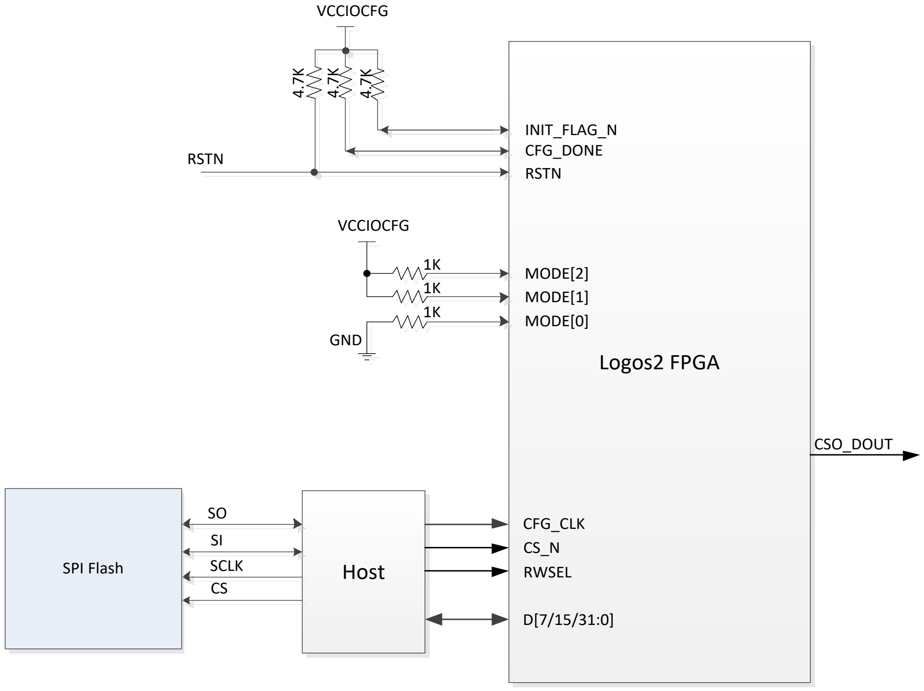


Figure 2-19 Application Diagram of the Slave Parallel Mode

In this mode, power the device up or apply a low-level pulse to RSTN to start programming;

Monitor pins INIT_FLAG_N and CFG_DONE to judge whether programming is completed; when the Host sends a bitstream to the Logos2 device, if wait PLL Lock is not enabled (disabled by default), the device releases control of CFG_DONE during a clock cycle between the 100 NOP type 1 packet headers at the end of the bitstream, pulling CFG_DONE high via the external pull-up resistor. After CFG_DONE is pulled high, the bitstream provides a clock for waking up the device, so it is necessary to ensure the bitstream is completely transmitted before pulling CS_N high. If wait is enabled, the clock cannot be terminated before CFG_DONE is pulled high, and CFG_DONE should be pulled high after CFG_DONE is pulled high. After CS_N is pulled high, at least 100 additional clocks must be provided.

The Slave Parallel mode supports 8/16/32-bit data width modes, which can be selected by parsing the bitstream. For more details, refer to Bus Bit Width Auto Detection.

The Slave Parallel interface of a Logos2 device supports x8, x16, and x32 bit widths. When configuring with the Slave Parallel interface, pay attention to the correspondence between the data buses and the bit order of data. For details, refer to Bit Orders under Different Interface Bit Widths.

Figure 2-20 shows the typical timing for the Slave Parallel configuration interface.

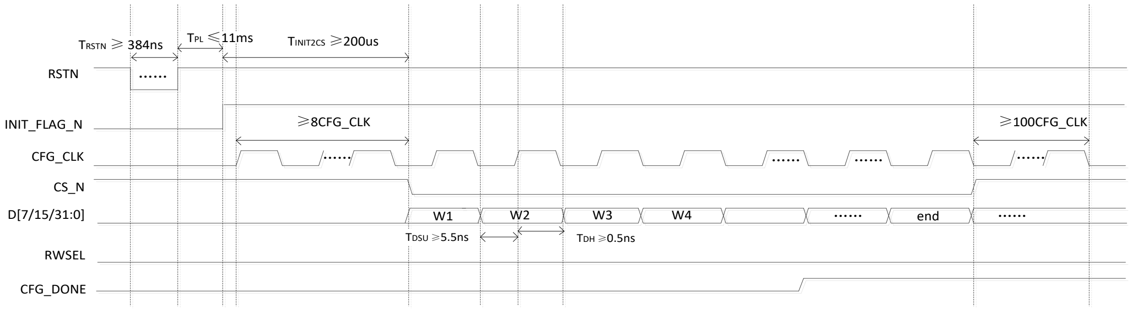


Figure 2-20 Slave Parallel Configuration Timing

When the Host cannot continuously send the data stream, control the CS_N or CFG_CLK of the Slave Parallel configuration interface to achieve discontinuous loading, as shown in Figure 2-21.

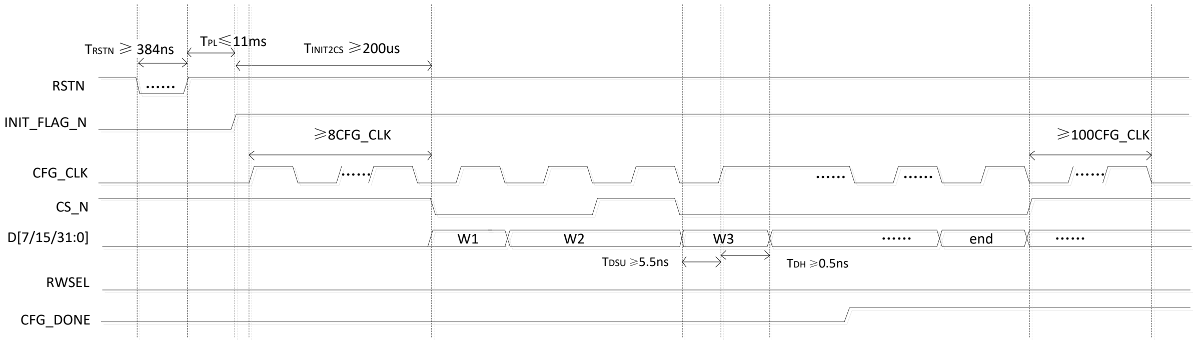


Figure 2-21 Slave Parallel Discontinuous Configuration Timing

After sending W2, pull CS_N high to pause data loading. It is important to keep the data unchanged, that is, keep D[31:0] unchanged while CS_N is high to avoid loading failure. After sending W3, stop the toggling of CFG_CLK to pause data loading, in which case there is no need to keep D[31:0] unchanged.

Additionally, the Slave Parallel interface also supports readback, with the typical timing shown in Figure 2-22. For the specific readback process, refer to Readback via the Slave Parallel Interface.

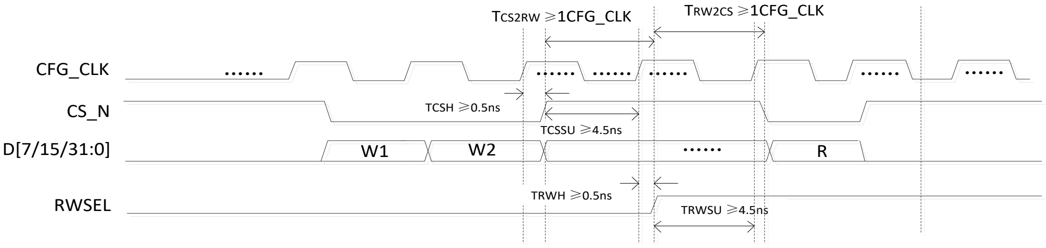


Figure 2-22 Slave Parallel Readback Timing

2.4.1 Slave Parallel Daisy Chain

Parallel daisy chain is a typical application of the Slave Parallel configuration mode. All devices can be configured in a parallel cascade using the Slave Parallel mode. Configure from the last (furthest) device to the first one in the chain during cascade configuration. The bitstream needs to be generated by clicking [Operations > Generate Daisy Chain File] in the Fabric Configuration tool.

Through parallel cascade, users can control the configuration of multiple cascaded chips on the board via a master control chip (Host). Connect the chips as shown in Figure 2-23.

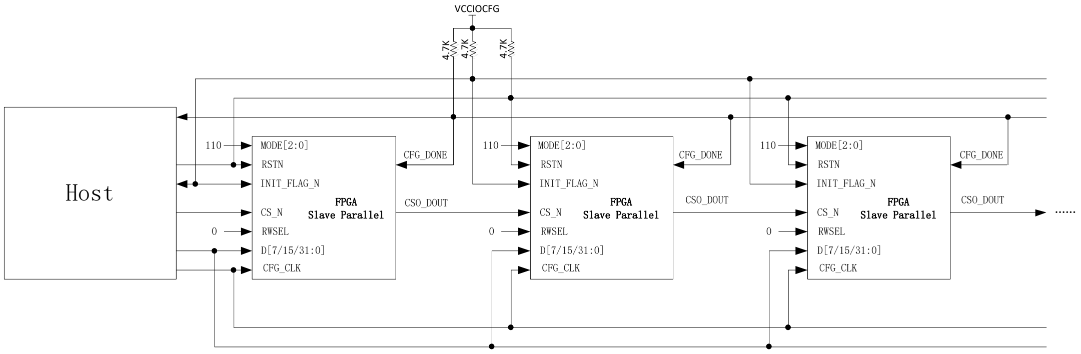


Figure 2-23 Daisy Chain Connection in the Slave Parallel Configuration Mode

Chapter 3 Download and Configuration

3.1 Configuration Process

Figure 3-1 shows the download and configuration process for the Logos2 Family FPGAs:

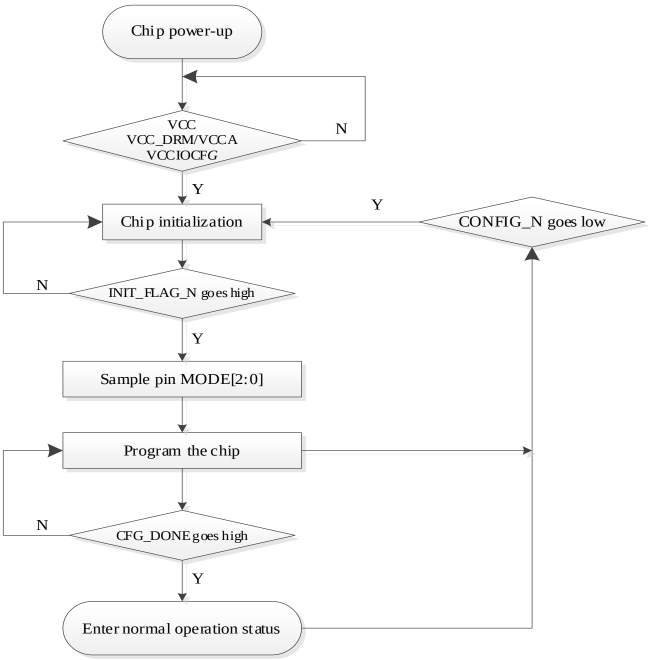


Figure 3-1 Download and Configuration Flowchart

For all configuration interfaces, the basic configuration steps are the same, including setup, bitstream loading, and wakeup.

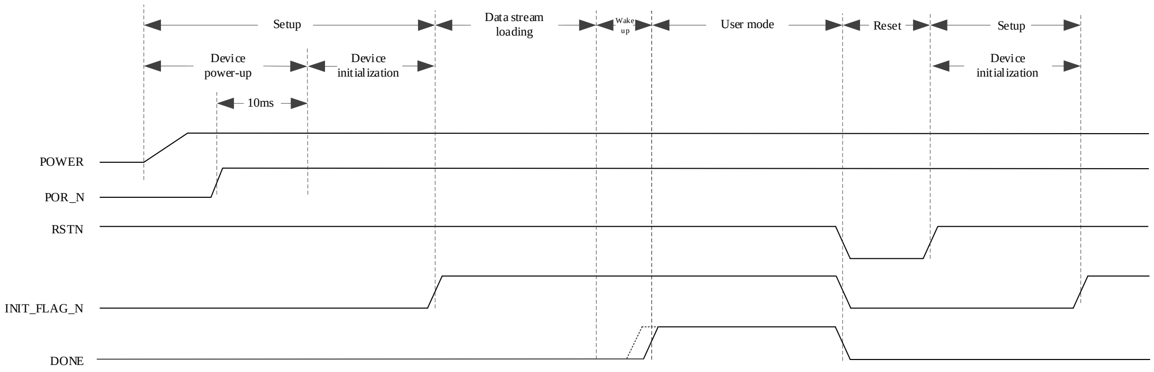


Figure 3-2 Download and Configuration Timing

The configuration process for the Logos2 Family FPGAs includes the following stages.

3.1.1 Setup

In the setup stage, complete the operations for FPGA device initialisation and configuration mode determination, including device power-up, device initialization, and configuration mode selection.

3.1.1.1 Device Power-Up

After the device is powered up, the power-up reset circuit of the CCS begins to operate, while the other circuits remain in the reset state. The power-up reset circuit counts using the master clk, which is generated by the device's internal crystal oscillator. After counting for 10ms, the clock generated by the internal crystal oscillator becomes stabilized, the power-up reset of the CCS is released, and the CCS starts to operate.

3.1.1.2 Device Initialisation

After the device is powered up and the CCS reset is released, the CCS starts to operate, performing device initialisation.

In the device initialisation stage, two operations are completed: clearing the configuration memory and reading the eFuse content. After initialisation is complete, the flag signal INIT_COMPLETE goes high.

To keep the CCS in the device initialisation process, maintain the external pin INIT_FLAG_N at a low level.

The CCS re-enters the device initialisation process when any of the following functions is triggered: reset, warmboot, or version fallback.

3.1.1.3 Configuration Mode Selection

After the INIT_FLAG_N signal goes high, configuration mode pins MODE[2:0] are sampled to select the device's configuration mode. After mode sampling, the CCS selects the configuration function of the multi-function pins.

3.1.2 Bitstream Loading

Different configuration modes have different configuration interfaces. After configuration mode selection, the pins corresponding to the modes are set as configuration pins. This stage includes operations such as bus bit width check, synchronization, device ID check, pre-load option settings, data loading, CRC, and post-load option settings.

3.1.2.1 Bus Bit Width Auto Detection

In the Slave Parallel configuration mode, the data bus bit width is automatically detected by the CCS, which performs bit width matching by detecting the two-word bus bit width auto detection sequence 000000AA 08100020.

The bus bit width auto detection sequence is at the beginning of the bitstream, before the synchronization word. The CCS only detects the lower 8 bits of the parallel data bus. During detection, the CCS first detects the lower 8 bits of the data bus for the 0xAA byte. After 0xAA is detected, the CCS continues to check whether the next byte is 0x08(x8), 0x10(x16), or 0x20(x32). If the next byte is not one of the aforementioned three, the CCS will re-detect 0xAA until it detects the sequence 0xAA + 0x08/0x10/0x20. Afterward, the CCS sets the relevant data pins as configuration data pins, switches the data bus to the detected bit width, and begins searching for the synchronization word.

8-bit bus

FF	Padding word
FF	
FF	
FF	
00	Bus bit width auto detection sequence
00	
00	
AA	
08	
10	
00	
20	
FF	Padding word
FF	
FF	
FF	
...	

16-bit bus

FFFF	Padding word
FFFF	
0000	Bus bit width auto detection sequence
00AA	
0810	
0020	
FFFF	Padding word
FFFF	
...	

32-bit bus

FFFFFFFF	Padding word
000000AA	Bus bit width auto detection sequence
08100020	
FFFFFFFF	Padding word
...	

3.1.2.2 Synchronization

The synchronization word 0x01332D94 is used for 32-bit word boundary alignment. Only after synchronization can the subsequent deframing operations be performed.

Only when the correct synchronization word is received does the CCS consider the data to be valid; then the CCS's packet processor performs the unpacking operation; any data before the synchronization word, other than the bus width auto detection sequence, will be ignored.

3.1.2.3 Master Mode Read Operation Selection

In the Master SPI configuration mode, select the read operation.

After selection, the device sends a read command to the SPI Flash again and readjusts the data bus bit width according to the read command.

3.1.2.4 Reset CRC

Reset the CRC register.

3.1.2.5 Device ID Check

Check whether the device ID in the bitstream matches the hardware; if not, the INIT_COMPLETE signal goes low, the ID error flag is stored in the status register, and the configuration process is exited.

3.1.2.6 Pre-load Options Settings

Before the configuration data is loaded, specify the operating state of the device in the configuration register:

Watchdog settings

Decryption initial vector settings

Key selection

Decryption selection

Set whether to continue using the configurable multiplexed port as a configuration port after configuration is completed

Wakeup clock selection

Wakeup timing settings

Whether to wait for PLL upon wakeup

Version fallback settings

Master mode clock frequency settings

3.1.2.7 Loading Configuration Data

Write the configuration data frames into the configuration memory.

3.1.2.8 Post-load CRC

After the configuration data is loaded, perform a CRC. The CRC value generated by the software is stored in the CRC register and compared with the CRC value calculated by the CCS. If the two values match, it indicates the CRC has been passed and all configuration information has been correctly written into the configuration memory. If the two values do not match, it indicates the CRC has failed, causing the INIT_N signal to go low and the CRC error flag to be stored in the status register.

3.1.2.9 Post-load Option Settings

After the configuration data is loaded, specify the operating state of the device in the configuration register:

Set whether to allow user logic to turn off the OSC

External port security level settings

3.1.3 Wakeup

After the bitstream is loaded and passes the CRC, the FPGA enters the wakeup stage. The CCS first enables the logical outputs of all functional modules inside the FPGA. Before wakeup, a CRC is performed again followed by a desynchronization operation, indicating the end of configuration, and subsequent operations require resynchronization. Finally, the wakeup is completed, and the corresponding global signals are released gradually.

3.1.3.1 Enabling Global Logic

After loading, once the CRC is passed, the logical outputs of all functional modules within the chip are enabled.

3.1.3.2 Initiate the Wakeup

Initiate the wakeup operation after the global logic is enabled.

3.1.3.3 Pre-wakeup CRC

A CRC is performed before executing the wakeup operation. The CRC value generated by the software is stored in the CRC register and compared with the CRC value calculated by the CCS. If the two values match, it indicates the CRC has passed and the pre-wakeup preparation is completed. If the two values do not match, it indicates the CRC has failed, causing the INIT_FLAG_N signal to go low and the CRC error flag to be stored in the status register.

3.1.3.4 Desynchronization

This indicates the end of the configuration process, requiring resynchronization for subsequent operations.

3.1.3.5 Wakeup

After desynchronization is completed, the wakeup circuit begins to operate according to the set timing, gradually releasing global signals. Users can set the wakeup timing in PDS by clicking [Project > Project Setting > Generate Bitstream > Startup]. For details, refer to ***"Pango_Design_Suite_User_***

Guide". The following text introduces each global signal based on the wake-up sequence of DONE-T1, GOUTEN-T2, GWEN-T3, and GRS_N-T4, with the default wake-up timing specified in PDS.

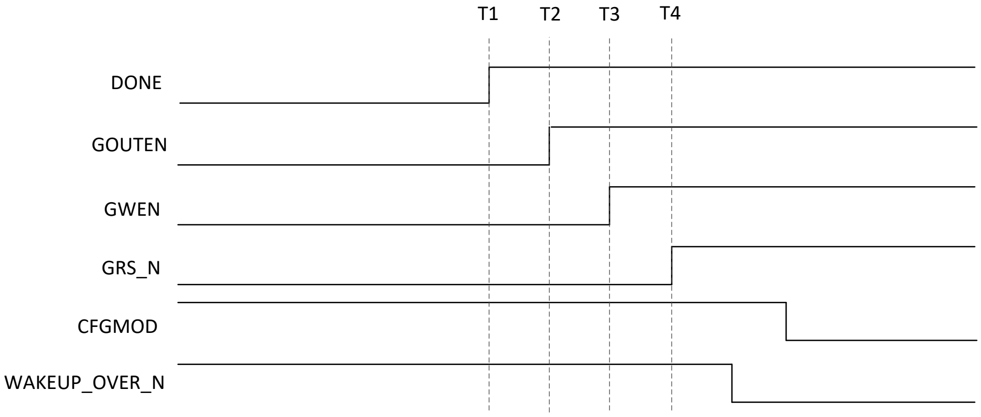


Figure 3-3 Wakeup Timing

The global logic enable signal GLOGEN is output to the fabric, enabling all logic blocks; then the signal is fed back to the CCS from the farthest end. After the CCS receives the feedback signal GLOGEN_FB, it enters the wakeup timing.

Before configuration is completed, the global register set/reset signal GRS_N is at a low level, controlling the GRS_N ports of all modules in the chip, putting the registers of modules in a set/reset state. After configuration is completed, the GRS_N signals of all modules in the chip are released, putting the registers of modules to a non-global set/reset state.

Before configuration is completed, the global IO output enable signal GOUTEN is at a low level, and all IO outputs are high-impedance. After configuration is completed, all IO outputs are enabled.

Table 3-1 GOUTEN Logical Functions

GOUTEN	**0**	**1**
IO used by users	High-Z	User logic control
I/O unused by users	High-Z	Bitstream settings[1]

Note:

1. Click [Project > Project Setting > Generate Bitstream > General > Unused IO Status] in PDS to set the bitstream. The default value is UNUSED, i.e., High-Z.

The global write enable signal GWEN can enable write for the global internal storage resources of the FPGA chip. Before configuration is completed, this signal is at a low level, disabling the write operation for the FPGA's storage resources.

The multi-function pin control signal CFGMOD releases the multi-function pins used during the configuration process for users.

The wakeup completion indicator signal WAKEUP_OVER_N indicates the successful completion of the wakeup process.

After wakeup, the FPGA switches from the configuration mode to the user mode.

To reload bitstreams, pull RSTN low for a period of time and then release it, and the FPGA will repeat the above process.

3.1.3.6 Delayed Wakeup

Users can set the global wakeup signals GOUTEN, GRS_N, GWEN, and DONE to 1 at time T1, T2, T3, or T4.

Users can set the number of cycles between two times.

The configuration completion indicator signal DONE drives pin CFG_DONE. After CFG_DONE is set to 1, the drive control of the pin is released. Pin CFG_DONE is an open-drain output pin, which is usually connected to an external pull-up resistor. If the DONE signal goes high but pin CFG_DONE does not go high (external drive is low), the wakeup process will be paused and not resumed until pin CFG_DONE goes high.

After DONE is set to 1, subsequent operations can be carried out only after the external signal and drive CFG_DONE is high. Signals that are set to 1 after DONE is set to 1 need to wait for CFG_DONE to be high. Signals that are set to 1 concurrently with DONE or before DONE is set to 1 are not affected by the external drive CFG_DONE.

For example, set GOUTEN to 1 at T1, DONE to 1 at T2, GWEN to 1 at T3, and GRSN to 1 at T4. Then GOUTEN and DONE are set to 1 according to the set timing. At time T3, if the external drive CFG_DONE is 1, then GWEN is set to 1; otherwise, GWEN is delayed by a time slot (the number of cycles between two times is settable), with CFG_DONE checked again in the next time slot. If CFG_DONE is high, then GWEN is set to 1; otherwise, it is delayed by another time slot. The time when GRSN is set to 1 is delayed sequentially based on the time when GWEN is set to 1.

3.2 Configuration Files

3.2.1 Bitstream Generation

PDS generates configuration files with different extensions to accommodate different configuration schemes, as described in Table 3-2.

Table 3-2 Descriptions of PDS Configuration Files

Configuration File Extension	**Description**
.sbit	Binary configuration data that includes header information (bitstream name, date, etc.). The configuration tool recognizes the header information but does not write it into the FPGA. Users can directly write the .sbit files into the FPGA through Cable using the Fabric Configuration tool.
.bin	Binary configuration data without header information (pure bitstream), suitable for user configuration schemes, such as microprocessor-based FPGA configuration.
.sfc	Bitstream files written into Flash, which are converted from the .sbit files.

As shown in Figure 3-4, the.sbit files are default bitstream files generated upon compilation by the PDS tool, and can be generated by running "Generate Bitstream".

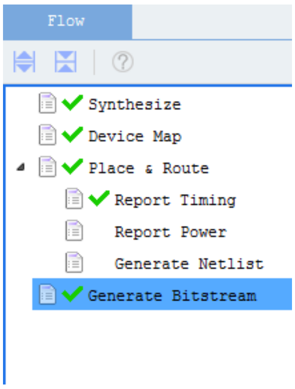
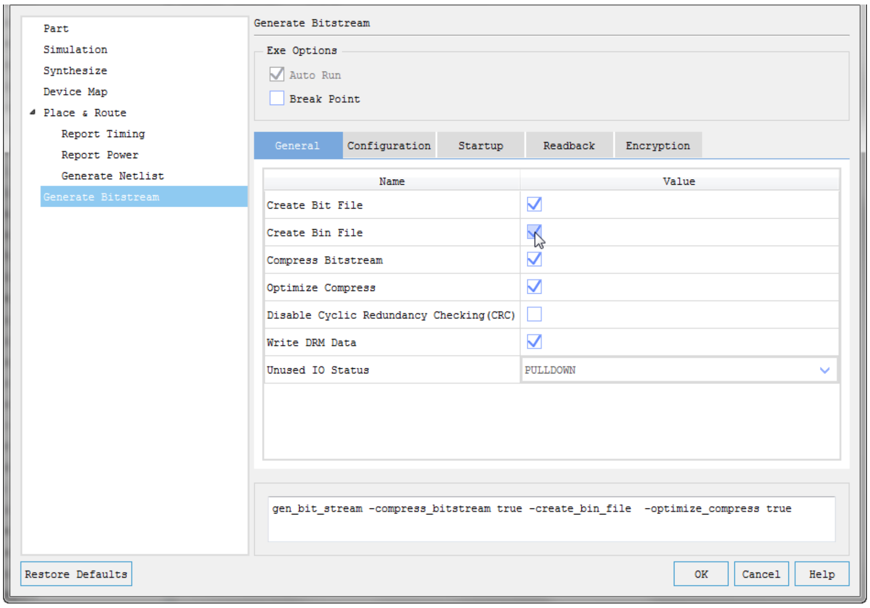


Figure 3-4 Bitstream Generation Diagram

As shown in Figure 3-4, right-click on [Generate Bitstream], select [configure], and a window as shown in Figure 3-5 will pop up. Check the [Create Bin File] option, click [OK], and rerun [Generate Bitstream] to generate both .sbit and .bin files simultaneously.



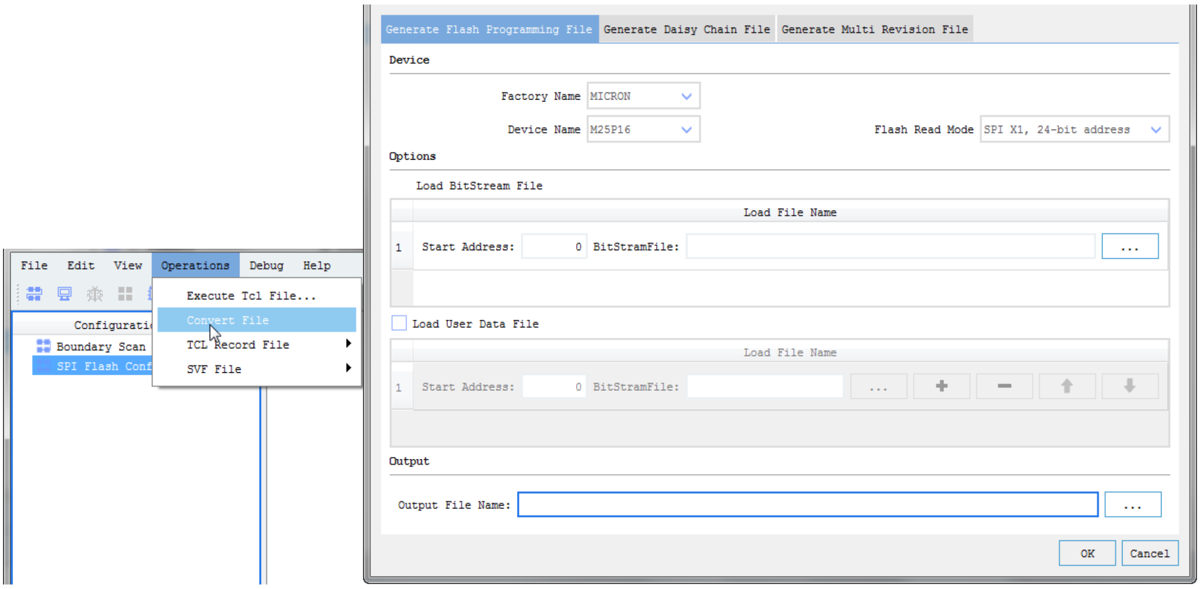
☑ Auto Run
☐ Break Point

Name	Value
Create Bit File	☑
Create Bin File	☑
Compress Bitstream	☑
Optimize Compress	☑
Disable Cyclic Redundancy Checking(CRC)	☐
Write DRM Data	☑
Unused IO Status	PULLDOWN

```
gen_bit_stream -compress_bitstream true -create_bin_file  -optimize_compress true
```

Figure 3-5 Configure Generate Bitstream

In the [Fabric Configuration] tool interface, click on [Operations] in the menu bar, and select [Generate Flash Programming File] from the dropdown menu to display the interface as shown in Figure 3-6.



Factory Name: MICRON

Device Name: M25P16

Flash Read Mode: SPI X1, 24-bit address

1 Start Address: 0 BitStreamFile:

□ Load User Data File

1 Start Address: 0 BitStreamFile:

Output File Name:

Figure 3-6 Generate Flash Programming File

Users should select the appropriate Flash Device based on their actual applications, choose the data width and address bit width of the Flash in the [Flash Read Mode] dropdown box, and enter the path of the sbit. file and the storage path for the sfc. file, and click [OK] to generate the Flash configuration file—the sfc. file.

3.2.2 Bitstream Size

The bitstream size for the Logos2 Family devices is shown in Table 3-3. The compressed bitstream files are smaller than those uncompressed. The size of the compressed files depends on the design, which is not explained here.

Table 3-3 Bitstream Size for Logos2 Family Devices

Device	**File Name**	**Uncompressed File Size (MByte)**
PG2L100H	*.sbit	3.61

3.2.3 Bit Orders under Different Interface Bit Widths

The Logos2 Family FPGA configuration supports multiple bit widths. Different configuration modes with the same bit width have a consistent bit order, i.e., the correspondence between the high & low bits of the data bus and the bitstream sequence are consistent. For example, the bit orders for Master SPI (x8) and Slave Parallel (x8) modes are consistent. Taking the synchronization word (32’h01332D94) in the bitstream as an example, the high and low bits of the data bus correspond to the bitstream sequence, as shown in Table 3-4.

Table 3-4 Synchronization Word Transmission Sequence under Different Bit Widths

Clock Cycle	**Numeral System**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
D[1:0](x2)	2'b	00	00	00	01	00	11	00	11	00	10	11	01	10	01	01	00
D[3:0](x4)	4'h	0	1	3	3	2	D	9	4								
D[7:0](x8)	8'h	01	33	2D	94												
D[15:0](x16)	16'h	0133	2D94														
D[31:0](x32)	32'h	01332D94															

Note:

In the table, D[x:0] only represents the high and low bits of the data bus. For specific corresponding pins, refer to the list of ports under different interface modes.

3.3 Configuration Data Packets

3.3.1 Packet Types

There are two types of packets for the Logos2 Family FPGA configuration bitstreams: Type 1 and Type 2.

Type 1 packets are used for register read and write operations. A type 1 packet consists of a header and data.

The header of a type 1 packet is a 32-bit word. Data follows the header. The data unit is 32-bit word.

If the number of 32-bit words following the header is 0, then the packet is empty, with a header but without data.

The format of the type 1 packet header is shown in Table 3-5.

Table 3-5 Type 1 Packet Header Format

[31:29]	**[28:27]**	**[26:22]**	**[21:0]**
Packet Header Type	Op Code	Register address	32-bit word count
101	00: No operation; 01: Write; 10: Read; 11: Reserved	Register address	The number of 32-bit words following the header

Type 2 packets are used for reading and writing to the configuration memory. A type 2 packet consists of a header and data.

The header of a type 2 packet is a 32-bit word. Data follows the header. The data unit is 32-bit word.

Type 2 packets must follow a type 1 empty packet. Type 2 packets do not specify a register address, but follow the address specified by the preceding type 1 empty packet.

The format of the type 2 packet header is shown in Table 3-6.

Table 3-6 Type 2 Packet Header Format

[31:29]	**[28:27]**	**[26:0]**
Packet Header Type	Op Code	32-bit word count
010	00: No operation; 01: Write; 10: Read; 11: Reserved	The number of 32-bit words following the header

3.3.2 Configuration register

Table 3-7 lists the configuration registers for the Logos2 Family FPGAs.

Table 3-7 Descriptions of Configuration Registers

Item	**R/W**	**Address**	**Description**
CRCR	R/W	00000	CRC Registers
IDR	R/W	00001	Device identification register
CMDR	R/W	00010	Command register
CTRL0R	R/W	00011	Control register 0
CTRL1R	R/W	00100	Control register 1
CMEMIR	W	00101	Frame data input register
MFWRITER	W	00110	Multi-frame write register
CMEMOR	R	00111	Frame data output register
IVR	W	01000	Initial vector register
STATUSR	R	01001	Status register
CHAINR	W	01010	Cascade register
ADRR	R/W	01011	Frame address register
SBPIR	R/W	01100	SBPI register
SEUR	R/W	01101	SEU control register
SEUSTATUSR	R	01110	SEU status register
IRSTCTRLR	R/W	01111	Warmboot control register
IRSTADRR	R/W	10000	Warmboot address register
WATCHDOGR	R/W	10001	Watchdog status register
HSTATUSR	R	10010	Historical status register
CMASKR	R/W	10111	Control master register

Item	R/W	Address	Description
OPTION0R	R/W	11001	Option register 0
OPTION1R	R/W	11010	Option register 1
SEUADDR	R	11101	SEU frame address register
SEUNADDR	R	11111	SEU next frame address register

The configuration registers are detailed as follows.

3.3.2.1 CRC Register (CRCR)

Each write to the CRCR corresponds to a CRC for the bitstream. If the value written to the CRCR matches the current CRC value calculated by the CCS, then the CRC is passed. Otherwise, init_complete is pulled low.

3.3.2.2 Device ID Register (IDR)

Device ID. Before loading the bitstream, it is required to write the IDCODE to the IDR. Subsequent operations can be performed only if the written IDCODE matches the device IDCODE. Each write to the IDR corresponds to a device ID check. If the lower 28 bits of the value written to the IDR match those of the device's ID, then the ID check is passed. Otherwise, init_complete is pulled low.

Table 3-8 Logos2 Family FPGA Device Models

Device Model	Device ID
PG2L100H	X0602899

3.3.2.3 Command Register (CMDR)

The CMDR determines the next operation to be performed by the CCS.

Write at most one command to the CMDR per operation. A single packet header cannot be followed by multiple commands.

Table 3-9 CCS Commands and Their Descriptions

Command	Code	Description
NOP	00000	No Op
RSTCRC	00001	Reset CRC Reset CRC register

Command	Code	Description
SWITCH	00010	Switch the master mode configuration clock frequency Initiate the operation to update the master mode configuration clock frequency, with the value determined by oscfsel in CTRL0R
WCMEM	00100	Write configuration data Used before writing configuration data into the configuration memory through CMEMIR
MFWRITE	00101	Multi-frame write Write a currently written frame of data into the subsequent consecutive frames. The number of frames is determined by MFWRITER
RCMEM	00110	Read configuration data Used before reading back configuration data from the configuration memory through CMEMOR
SWAKEUP	00111	Start wakeup operation
SWAKEDOWN	01000	Disable wakeup operation
GUP	01001	Enable the internal logic
GDOWN	01010	Disable the internal logic
DESYNC	01011	Desynchronization Used at the end of configuration
RWD	01100	Restart the watchdog
RRBCRC	01101	Reset readback CRC
RBCRC	01110	Readback CRC
IRST	01111	Warmboot Warmboot is invalid during version fallback
WCMEMDIS	10000	Disable configuration memory writes Disable configuration memory writes after configuration data has been written.
RCMEMDIS	10001	Disable configuration memory reads Disable configuration memory reads after the configuration data has been read.

3.3.2.4 Control Register 0 (CTRL0R)

Table 3-10 Descriptions of CTRL0R

Bit	**Item**	**Initial Value**	**Description**
[31:23]	Reserved		
[22]	otsd_en	1'b0	Enable over temperature shutdown 0: Not enabled 1: Enabled When over temperature occurs (the over_temperature signal from ADC to CCS changes from 0 to 1) and Over Temperature Shutdown is enabled, CCS counts with the internal master configuration clock imclk (if the clock frequency is 50M, the counting time is 10ms). Users should complete all configuration interface operations (including the internal slave parallel interface) within this counting period. After counting is complete, initiate the shutdown process (shut down the global signals according to the shutdown timing and then the global logic enable signal glogen). After the temperature resumes normal (the over_temperature signal from the ADC to the CCS changes from 1 to 0), initiate the wakeup process (set the global logic enable signal glogen to 1 first, then wake up the global signals according to the wakeup timing). Both the shutdown and wakeup processes operate in the clock domain of the internal master configuration clock imclk.
[21:5]	Reserved		
[4]	fallback_en	1’b1	Enables version fallback
[3]	Reserved		
[2]	persist	1'b0	Select whether to reserve the configuration interfaces (Master SPI interface, Slave Parallel interface, Slave Serial interface) in the user mode 1’b0: Released for users to use 1’b1: Reserved as configuration interfaces In user mode, when the internal slave parallel interface is enabled, the Slave Parallel interface and the Slave Serial interface cannot be reserved as configure interfaces.
[1]	Reserved		
[0]	dec_en	1‘b0	Enable decryption

3.3.2.5 Control Register 1 (CTRL1R)

Table 3-11 Descriptions of Control Register 1

Bit	Item	Initial Value	Description
[31:5]	Reserved		
[4]	mask_en	1'b0	Enables variable storage unit readback mask 0: Not masked, with readback supported 1: Masked
[3]	mfg_por_off	1‘b0	0: Normally monitors VCC, VCCA, VCCIOCFG, VCC_DRM voltages 1: When it is required to test SRAM retention voltage, with gwen being set to 1, turn off the fine detection function of VCC and VCC_DRM. At this time, POR will reassert only when VCC<0.65V or VCC_DRM<0.5V.
[2]	osc_off_en	1‘b0	Controls whether user logic is allowed to shut down OSC 0: Disallowed 1: Allowed
[1:0]	wrctrl	2‘b00	External port security level. Controls shutdown reconfiguration, partial reconfiguration, partial dynamic reconfiguration, and readback configuration memory. Once configured to disabled, it cannot be changed back to enabled unless reset. 00: Reconfiguration enabled, readback enabled (default) 01: Reconfiguration enabled, readback disabled 1x: Reconfiguration disabled, readback disabled

3.3.2.6 Control Mask Register (CMASKR)

The CMASKR is used to mask corresponding bits in control register 0 and control register 1, with a default value of 32'd0. 0 indicates masking. For example, when the value of the CMASKR is 32'h0000_0003, the lowest 2 bits of control register 0 and control register 1 can be written to, while the other bits cannot.

3.3.2.7 Option Register 0 (OPTION0R)

Table 3-12 Descriptions of Option Register 0

Bit	**Item**	**Initial Value**	**Description**
[31:24]	Reserved		
[23]	done_syn	1‘b0	Enables external done synchronization. Indicates whether the external input done is the synchronous signal for the output done. If so, the wakeup module of the CCS uses the signal directly; If not, the signal needs to be synchronized before use, typically for the delayed done function. 0: Asynchronous 1: Synchronous
[22:21]	t_sel	2‘d0	Selects the wakeup cycle length, i.e., the time interval between T1, T2, and T3 2‘b00: 1 2‘b01: 2 2‘b10: 4 2‘b11: 1
[20:19]	done_sel	2‘d0	Selects the time when the done signal is pulled high 2‘d0: T3 2‘d1: T1 2‘d2: T2 2‘d3: T4
[18:17]	gwen_sel	2‘d0	Selects the time when the gwen signal is pulled high 2‘d0: T2 2‘d1: T1 2‘d2: T3 2‘d3: T4
[16:15]	grsn_sel	2‘d0	Selects the time when the grsn signal is pulled high 2‘d0: T2 2‘d1: T1 2‘d2: T3 2‘d3: T4
[14:13]	gouten_sel	2‘d0	Selects the time when the gouten signal is pulled high 2‘d0: T1 2‘d1: T2 2‘d2: T3 2‘d3: T4
[12]	Reserved	1‘b0	
[11]	wait_pll	1‘b0	Enables wait PLL lock during wakeup 0: Not enabled 1: Enabled
[10]	Reserved	1'b0	
[9]	startup_uclk_en	1‘b0	Enables user clock wakeup 0: Use the clock used during configuration for wakeup 1: Use the user clock for wakeup
[8:7]	Reserved	3‘d0	
[6:0]	oscfsel	7‘d0	Selects the frequency of the master mode configuration clock CFG_CLK

3.3.2.8 Option Register 1 (OPTION1R)

Table 3-13 Descriptions of Option Register 1

Bit	Item	Initial Value	Description
[31:2]	Reserved		
[1]	rbcrc_disable	1‘b0	Disables readback CRC error indication 0: Enabled 1: Disabled
[0]	crc_disable	1‘b0	Disables CRC 0: Enabled 1: Disabled

3.3.2.9 Frame Data Input Register (CMEMIR)

The CMEMIR is the configuration data interface for the configuration memory. Data is written into the configuration memory through this interface.

3.3.2.10 Multi-Frame Write Register (MFWRITER)

It indicates the number of compressed frames, i.e., the data content of several consecutive frames following is the same as that of the just-written frame. For instance, after configuring a frame of data, if the data content of the next three consecutive frames is the same as that frame, then the MFWRITER should be written with a 3.

3.3.2.11 Frame Data Output Register (CMEMOR)

The CMEMOR is the readback data interface for the configuration memory. Data is read back from the configuration memory through this interface.

3.3.2.12 Initial Vector Register (IVR)

This register shows the initial vector of the encrypted data.

3.3.2.13 Status Register (STATUSR)

Table 3-14 Descriptions of Status Register

Bit	**Item**	**Description**
[31:30]	Reserved	
[29]	prcfg_over	Partial reconfiguration completion flag
[28]	prcfg_err	Partial reconfiguration error flag
[27]	over_temp	Overtemperature flag
[26]	flg_x32	Indicates the 32-bit data width in the Slave Parallel mode
[25]	flg_x16	Indicates the 16-bit data width in the Slave Parallel mode
[24]	flg_x8	Indicates the 8-bit data width in the Slave Parallel mode
[23:22]	ipal_m[1:0]	Selects the data width of the internal slave parallel interface
[21]	fallback	Fallback indication flag
[20]	Reserved	
[19]	pll_lock	PLL lock flag
[18]	gwen	Global write enable
[17]	grsn	Global register set/reset
[16]	gouten	Global IO output enable
[15]	glogen_fb	Global logic enable feedback
[14]	glogen	Enabling Global Logic
[13]	done_i	DONE pin input
[12]	done	Device wake-up success flag
[11]	init_n	INIT_FLAG_N pin input
[10]	init_complete	Indicates initialisation completion and configuration error
[9:7]	m[2:0]	Mode selection
[6]	wakedown_over	Wakeup shutdown over
[5]	wakeup_over	Wakeup over
[4]	timeout	Watchdog timeout
[3]	rbcrc_err	Readback CRC results 0: CRC correct 1: CRC error
[2]	aut_err	Authentication result 0: Authentication pass 1: Authentication failure
[1]	crc_err	CRC results 0: CRC correct 1: CRC error
[0]	id_err	ID detection results 0: Correct 1: Wrong

3.3.2.14 Watchdog Register (WATCHDOGR)

It is used for timeout detection in the configuration mode and user mode.

Table 3-15 Descriptions of Watchdog Register

Bit	Item	Initial Value	Description
[31]	wd_user_en	1‘b0	Enables the watchdog in the user mode
[30]	wd_cfg_en	1‘b0	Enables the watchdog in the configuration mode
[29:0]	wd_value	30‘h3FFF_FFFF	Watchdog timeout value

3.3.2.15 Frame Address Register (ADDR)

It provides the starting addresses for configuration and readback data. It can transmit only valid addresses, rather than those beyond the range.

Table 3-16 Descriptions of Frame Address Register

Bit	Item	Initial Value	Description
[31:27]	Reserved		
[26:25]	type	2‘d0	Configures memory content type 00: Non-DRM content 01: DRM content 10: Reserved 11: Reserved
[24:20]	addr_region	5‘d0	Region address
[19:18]	Reserved		
[17:10]	addr_column	8‘d0	Column address
[9:8]	Reserved		
[7:0]	addr_frame	8‘d0	Frame address

3.3.2.16 SBPI Register (SBPIR)

It provides control and options of the Master SPI mode.

Table 3-17 Descriptions of SBPI Register

Bit	**Item**	**Initial Value**	**Description**
[31]	emclk_en	1‘b0	Enables external master clock 1'b0: Use the internal clock as the master mode configuration clock 1'b1: Use the external clock as the master mode clock
[30]	sbpi_rf_sel	1‘b0	Selects the clock edge from which data is sampled 0: Rising edge 1: Falling edge In the Master SPI mode, sampling on the falling edge is in the fast mode
[29:11]	Reserved		
[10]	addrwidth	1‘b0	SPI address bit width 0: 24-bit address 1: 32-bit address
[9:8]	datawidth	2‘d0	SPI data width 00:X1 01:X2 10:X4 11:X8
[7:0]	opcode	8‘h0B	SPI master mode operation code

3.3.2.17 SEU Control Register (SEUR)

It provides SEU error detection and correction, and readback CRC control.

Table 3-18 Descriptions of SEU Control Register

Bit	**Item**	**Initial Value**	**Description**
[31:2]	Reserved		
[1]	seu_en	1‘b0	Enables SEU
[0]	rbcrc_en	1‘b0	Enables readback CRC

3.3.2.18 SEU Status Register (SEUSTATUSR)

Table 3-19 Descriptions of SEU Status Register

Bit	Item	Description
[31:17]	Reserved	
[16]	drcfg_over	Dynamic reconfiguration completion flag
[15]	drcfg_err	Dynamic reconfiguration error flag
[14:3]	seu_index	Indicates the position of SEU single-bit error within the frame
[2]	seu_ded	SEU double-bit error flag
[1]	seu_sec	SEU single-bit error flag
[0]	seu_over	SEU completion flag

3.3.2.19 SEU Frame Address Register (SEUADDR)

Table 3-20 Descriptions of SEU Frame Address Register

Bit	Item	Description
[31:21]	Reserved	
[20:16]	seu_region_naddr	Region address of the current SEU frame
[15:8]	seu_column_naddr	Column address of the current SEU frame
[7:0]	seu_frame_naddr	Frame address of the current SEU frame

3.3.2.20 SEU Next Frame Address Register (SEUNADDR)

Table 3-21 Descriptions of SEU Next Frame Address Register

Bit	Item	Description
[31:21]	Reserved	
[20:16]	seu_region_naddr	Region address of the next SEU frame
[15:8]	seu_column_naddr	Column address of the next SEU frame
[7:0]	seu_frame_naddr	Frame address of the next SEU frame

3.3.2.21 Warmboot Control Register (IRSTCTRLR)

Table 3-22 Descriptions of Warmboot Control Register

Bit	Item	Initial Value	Description
[31:1]	Reserved		
[0]	vback_en	1‘b0	Enables version fallback. Controls fallback to either the golden bitstream or the previous bitstream 0: Disabled 1: Enabled

3.3.2.22 Warmboot Address Register (IRSTADDR)

Flash start address during warmboot, byte addressing.

Table 3-23 Descriptions of Warmboot Address Register

Bit	Item	Initial Value	Description
[31:24]	Flash start address	8‘d0	Flash start address [31:24] for 4-byte mode
[23:0]	Flash start address	24‘d0	Flash start address

3.3.2.23 Historical Status Register (HSTATUSR)

Table 3-24 Descriptions of Historical Status Register

Bit	Item	Description
[31:15]	Reserved	
[14]	timeout1	Previous state timeout 0: Watchdog not timed out 1: Watchdog timeout
[13]	rbcrc_err1	Previous state readback CRC error 0: Correct 1: Wrong
[12]	crc_err1	CRC error of the last status 0: Correct 1: Wrong
[11]	id_err1	Previous state ID error 0: Correct 1: Wrong
[10]	irst1	Indicates if the previous state's version is a warmboot upgrade version 0: Not a warmboot upgrade version 1: Warmboot upgrade version
[9]	fallback1	Indicates if the previous state was a fallback operation 0: No 1: Yes
[8]	valid1	Last status valid 0: Invalid 1: Valid

Bit	Item	Description
[7]	1‘b0	Reserved
[6]	timeout0	Current state timeout 0: Watchdog not timed out 1: Watchdog timeout
[5]	rbcrc_err0	Current state readback CRC error 0: Correct 1: Wrong
[4]	crc_err0	CRC error of the current status 0: Correct 1: Wrong
[3]	id_err0	ID error of the current status 0: Correct 1: Wrong
[2]	irst0	Indicates if the current state's version is a warmboot upgrade version 0: Not a warmboot upgrade version 1: Warmboot upgrade version
[1]	fallback0	Indicates if the current state was a fallback operation 0: No 1: Yes
[0]	valid0	Current status valid 0: Invalid 1: Valid

3.4 Bitstream Formats

General bitstream formats are listed in this section.

Table 3-25 General Bitstream Formats

Content	Description
FFFFFFFF	Padding words (100)
……	
FFFFFFFF	
000000AA	Bus Bit Width Auto Detection
08100020	
FFFFFFFF	Padding words (10)
……	
FFFFFFFF	
01332D94	Synchronization
AB000001	Type 1 packet header: Write to SBPIR
xxxxxxxx	Data: SBPI Master mode configuration control options
A0000000	10 NOP type 1 packet headers
……	
A0000000	
ABC00001	Type 1 packet header: Write to IRSTCTRLR

Content	Description
xxxxxxxx	Data: Contents of IRSTCTRLR
AC000001	Type 1 packet header: Write to IRSTADRR
xxxxxxxx	Data: Contents of IRSTADRR
A8800001	Type 1 packet header: Write to CMDR
00000000	Data: NOP command
A8800001	Type 1 packet header: Write to CMDR
00000001	Data: RSTCRC command
A8400001	Type 1 packet header: Write to IDR
xxxxxxxx	Data: IDCODE
AC400001	Type 1 packet header: Write to WATCHDOG register
xxxxxxxx	Data: WATCHDOG
ADC00001	Type 1 packet header: Write to CMASKR
xxxxxxxx	Data: Contents of CMASKR
A8C00001	Type 1 packet header: Write to CTRL0R register
xxxxxxxx	Data: Contents of CTRL0R
AE400001	Type 1 packet header: Write to OPTION0R
xxxxxxxx	Data: Contents of OPTION0R
AE800001	Type 1 packet header: Write to OPTION1R
xxxxxxxx	Data: Contents of OPTION1R
A8800001	Type 1 packet header: Write to CMDR
00000002	Data: SWITCH command
A0000000	10 NOP type 1 packet headers
……	
A0000000	
AAC00001	Type 1 packet header: Write to ADRR
xxxxxxxx	Data: Contents of ADRR
A8800001	Type 1 packet header: Write to CMDR
00000004	Data: WCMEM command
A9400000	Type 1 packet header: Write to CMEMIR
{3‘b010, 2‘b01, 27‘dx}	Type 2 packet header: Write operation
xxxxxxxx	Data: Contents written to configuration memory
……	
xxxxxxxx	
A0000000	50 NOP type 1 packet headers
……	
A0000000	
A8800001	Type 1 packet header: Write to CMDR
00000010	Data: WCMEMDIS command
A8800001	Type 1 packet header: Write to CMDR

Content	Description
00000012	Data: VDDTUP command
A8000001	Type 1 packet header: Write to CRCR
xxxxxxxx	Data: CRC value
ADC00001	Type 1 packet header: Write to CMASKR
xxxxxxxx	Data: Contents of CMASKR
A9000001	Type 1 packet header: Write to CTRL1R
xxxxxxxx	Data: Contents of CTRL1R
A8800001	Type 1 packet header: Write to CMDR
00000009	Data: GUP command
A8800001	Type 1 packet header: Write to CMDR
00000007	Data: SWAKEUP command
A8000001	Type 1 packet header: Write to CRCR
xxxxxxxx	Data: CRC value
A8800001	Type 1 packet header: Write to CMDR
0000000B	Data: DESYNC command
A0000000	100 NOP type 1 packet headers
……	
A0000000	

3.5 Download Cable Speed

For JTAG download and SPI Flash programming via a USB cable, the default speed is 10 MHz.

Users can set the download speed in [Fabric Configuration] by clicking the [Connect To Server] icon.

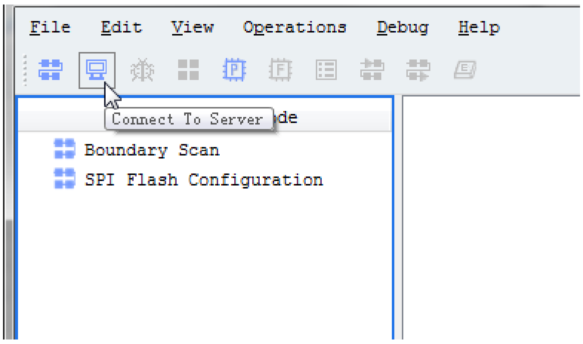
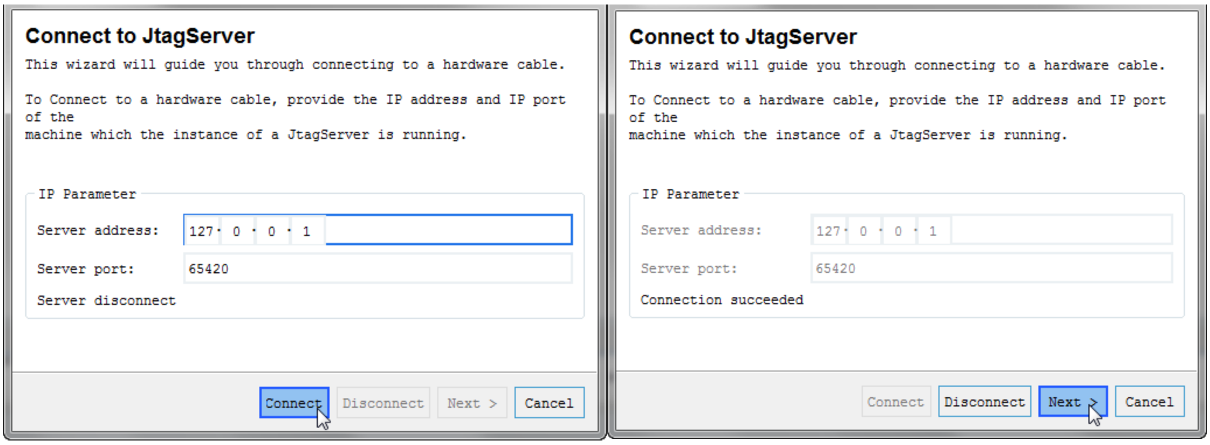


Figure 3-7 Fabric Configuration

If not connected, connect first as shown in the left of Figure 3-8; if already connected, click [Next] as shown in the right of Figure 3-8:



Connect to JtagServer

This wizard will guide you through connecting to a hardware cable.

To Connect to a hardware cable, provide the IP address and IP port of the
machine which the instance of a JtagServer is running.

IP Parameter

Server address: 127 · 0 · 0 · 1

Server port: 65420

Server disconnect

Connect to JtagServer

This wizard will guide you through connecting to a hardware cable.

To Connect to a hardware cable, provide the IP address and IP port of the
machine which the instance of a JtagServer is running.

IP Parameter

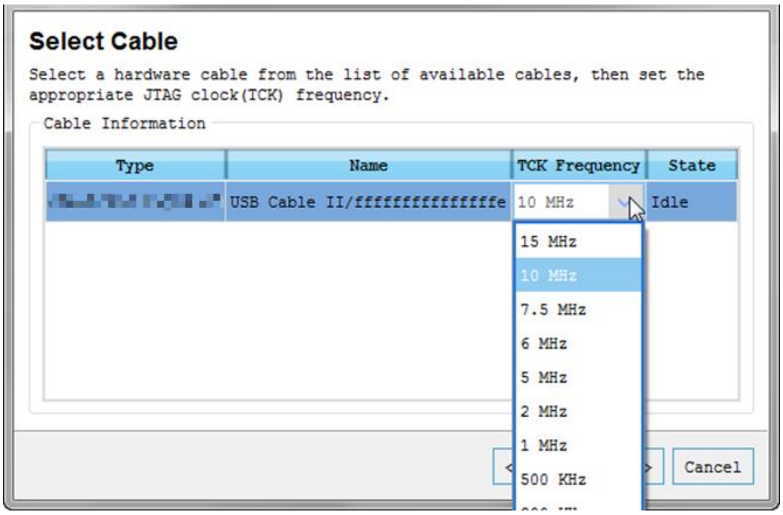
Server address: 127 · 0 · 0 · 1

Server port: 65420

Connection succeeded

Figure 3-8 Connect To Cable

Users can select the USB Cable frequency in the [TCK Frequency] dropdown menu.



Select Cable

Select a hardware cable from the list of available cables, then set the appropriate JTAG clock(TCK) frequency.

Cable Information

Type	Name	TCK Frequency	State
[redacted]	USB Cable II/fffffffffffffffe	10 MHz	Idle

Figure 3-9 USB Cable Speed Settings

3.6 Configuration Speeds

3.6.1 JTAG Mode

For download in the JTAG mode, the maximum speed is 50MHz.

Due to the download speed limitation of PANGO USB CABLE II, the maximum configuration frequency is 15MHz when downloading via the download cable.

3.6.2 Master SPI Mode

In the Master SPI mode, the CFG_CLK clock frequency is configurable, with a default value of 2.99MHz. The maximum configuration frequency is 15.38MHz and 40MHz in the low-speed mode and high-speed mode, respectively. The Master SPI serial daisy chain supports a configuration frequency of up to 22.22MHz.

3.6.3 Slave Parallel Mode

In the Slave Parallel mode, the download clock comes from an external Host device, with a maximum clock rate of 80MHz.

3.6.4 Slave Serial Mode

In the Slave Serial mode, the download clock comes from an external Host device, with a maximum clock rate of 80MHz. The Slave Serial daisy chain supports a configuration frequency of up to 50 MHz.

3.7 Multi-boot

When using the multi-boot function, the external Flash must support the storage of at least 2 sets of bitstreams. Bitstream 0 is either the golden bitstream version or the application bitstream version, with a starting address of 0, while other bitstreams are application bitstreams. If any errors occur during the configuration process, reset circuits other than the version fallback circuit, and reload the golden bitstream or the previous-version application bitstream. If the golden bitstream or the previous-version application bitstream also contains errors, there is no need to reset. Instead, while setting INIT_FLAG_N to 0, set FCS_N to 1. Then the SPI master mode operation ends.

When using the multi-boot function, it is necessary to enable the watchdog.

During configuration, the following errors will trigger a version fallback.

1. Incorrect device ID
2. CRC error
3. Watchdog timeout
4. Authentication failure

3.7.1 Golden Bitstream Initialisation System

In this application scenario, the FPGA will first start from the golden bitstream and enter user mode. Then, it completes bitstream update of the Flash through JTAG or user logic, and users can jump to the application bitstream operation step through a warmboot. For a generation in the software, the difference with the application bitstream initialisation system below is the absence of the embedded warmboot insertion option. (To insert embedded warmboot: Click [Fabric Configuration > Operations > Convert File > Generate Multi Revision File > Data Stream Type: Multi Boot Data Stream > Insert IRST CMD to Data Stream])

1. Load the golden bitstream from Flash address 0.
2. Flash operations are completed via JTAG or by the user. (Optional) If the application bitstream for multi-boot is already in the Flash, Flash operations are not necessary.
3. Perform JTAG warmboot.
4. Load the application bitstream from the Flash address specified by IRSTCTRLR.
5. If there is an error while loading the application bitstream, load the golden bitstream from Flash address 0 or fall back to the previous-version application bitstream.

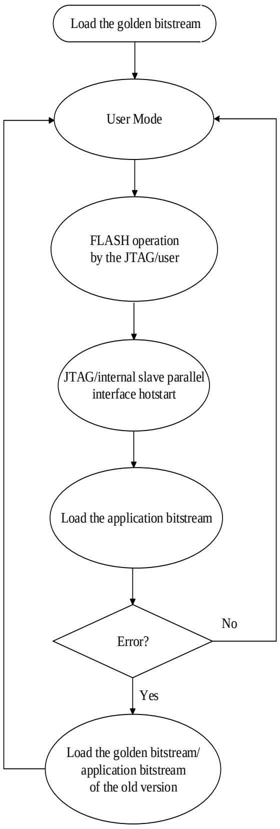

Load the golden bitstream
User Mode
FLASH operation by the JTAG/user
JTAG/internal slave parallel interface hotstart
Load the application bitstream
Error?
No
Yes
Load the golden bitstream/ application bitstream of the old version

Figure 3-10 Golden Bitstream Loading Process

3.7.2 Application Bitstream Initialisation System

In this application scenario, the FPGA will first start from the specified application bitstream and enter user mode. Compared with the golden bitstream initialisation system, no additional operation is required to start from the application bitstream, which is more convenient. For generation in the software, the difference with the golden bitstream initialisation system is the presence of the embedded warmboot insertion option. (To insert embedded warmboot: Click [Fabric Configuration > Operations > Convert File > Generate Multi Revision File > Data Stream Type: Multi Boot Data Stream > Insert IRST CMD to Data Stream])

1. Load the golden bitstream from Flash address 0.
2. Load the application bitstream from the Flash address specified by IRSTCTRLR through the golden bitstream's embedded warmboot.

3. If there is an error while loading the application bitstream, load the golden bitstream from Flash address 0 or fall back to the previous-version application bitstream.

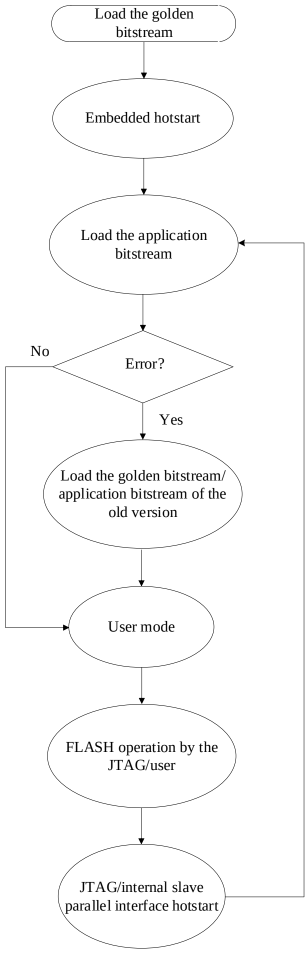

Load the golden bitstream
Embedded hotstart
Load the application bitstream
Error?
No
Yes
Load the golden bitstream/ application bitstream of the old version
User mode
FLASH operation by the JTAG/user
JTAG/internal slave parallel interface hotstart

Figure 3-11 Application Bitstream Loading Process

3.7.3 Without Golden Bitstream

1. If the Flash supports storage of up to 2 sets of bitstreams, when the version fallback function is enabled, both bitstreams are application bitstreams.
2. Load the application bitstream from Flash address 0.
3. Flash operations are completed via JTAG or by the user. (Optional) If the application bitstream for multi-boot is already in the Flash, Flash operations are not necessary.
4. Perform a warmboot via JTAG or internal slave parallel interface.
5. Load the application bitstream from the Flash address specified by IRSTCTRLR.

6. If there is an error while loading the application bitstream, fall back to the previous-version application bitstream.

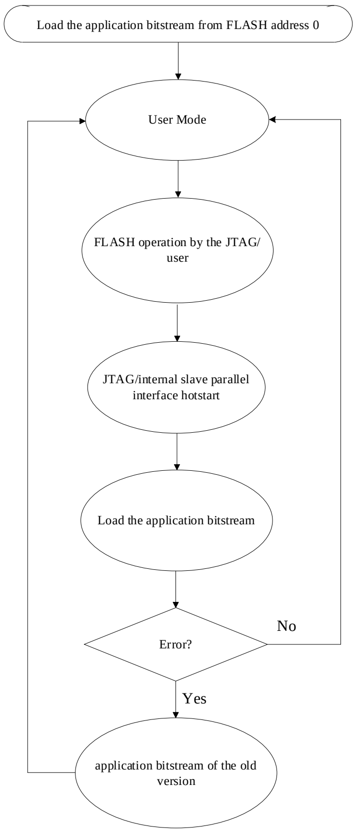

Load the application bitstream from FLASH address 0
User Mode
FLASH operation by the JTAG/ user
JTAG/internal slave parallel interface hotstart
Load the application bitstream
Error?
No
Yes
application bitstream of the old version

Figure 3-12 Loading Process Without Golden Bitstream

3.7.4 Warmboot

Before sending instructions to the device, perform synchronization to ensure the device can correctly parse the instructions. If sending instructions via the Slave Parallel interface, a bus bit width detection instruction must also be sent. In a multi-boot application, the golden bitstream includes an instruction to write the starting address of the application bitstream to IRSTADRR. IRSTADRR write operation is not required during a warmboot. If the multi-boot does not include the golden bitstream, the user needs to write the starting address of the bitstream to be loaded during warmboot into the IRSTADRR as part of the warmboot process. If a warmboot instruction is embedded within the golden bitstream, it is not possible to upgrade to the golden bitstream via warmboot within the application bitstream initialisation system.

Warmboot can also be used to clear the configuration memory, such as when lifting the temporary disablement of JTAG, in which case it is not necessary to write the starting address to the IRSTADRR. The complete warmboot process is shown in Table 3-26.

Table 3-26 Warmboot Process

Content	**Description**
FFFFFFFF	Padding words (100)
……	
FFFFFFFF	
000000AA	Bus Bit Width Auto Detection
08100020	
FFFFFFFF	Padding words (10)
……	
FFFFFFFF	
01332D94	Synchronization
AC000001	Type 1 packet header: Write to IRSTADRR (optional)
xxxxxxxx	Data: Contents of IRSTADRR (optional)
A8800001	Type 1 packet header: Write to CMDR
0000000F	Data: IRST command
A8800001	Type 1 packet header: Write to CMDR
0000000B	Data: DESYNC command
A0000000	100 NOP type 1 packet headers
……	
A0000000	

3.7.5 Watchdog

The watchdog is used to detect timeouts during configuration.

In configuration mode, the watchdog clock acts as the configuration clock. The watchdog count decreases by 1 for each clock cycle. When the watchdog count reaches 0, a watchdog timeout flag is generated.

If version fallback is enabled in the bitstream settings by clicking [Project Setting > Generate Bitstream > Configuration > Enable Watchdog in Configuration Mode] and an appropriate watchdog value is set by clicking [Load Watchdog], a watchdog timeout during the configuration process in master mode (Master SPI) will trigger a version fallback.

The watchdog is disabled during and after the version fallback process. After the version fallback is successfully completed, users can re-enable the watchdog via a warmboot.

Watchdog values

The watchdog works and uses the corresponding clocks in different configuration modes. Therefore, the watchdog value can be estimated based on the actual bitstream size ($L_{bitstream}$) and configuration bit width (W).

i.e.,

Watchdog (Hex) > $L_{bitstream}/W$

Chapter 4 Readback Operation

Logos2 FPGAs support reading back both standard and compressed bitstreams through JTAG and Slave Parallel interfaces.

4.1 Readback via the JTAG Interface

Before reading back through the JTAG interface, it is necessary to detect the wakeup_over signal in the IR or the status register to ensure that the device is awake before proceeding with readback. Typically, readback occurs long after the wakeup process is completed.

4.2 Readback via the Slave Parallel Interface

To perform readback using the Slave Parallel interface, ensure that the interface is enabled or reserved.

If using this interface for readback in configuration mode, ensure that the Slave Parallel mode is set to enabled, i.e., MODE[2:0] = 3’b110.

If using this interface for readback in user mode, ensure that "reserve configuration interface" is enabled in the valid bitstream (click [Project Setting > Generate Bitstream > Readback > Persist Pin] in PDS, not reserved by default), and this is configured through the Slave Parallel interface. Additionally, the data width used for readback must match that used during configuration; that is, if configured with Slave Parallel x8 and the configuration interface is reserved, readback must be performed with Slave Parallel x8 and not with x16/x32 bit widths.

Before readback operations via the Slave Parallel interface, it is necessary to detect the wakeup_over signal in the status register to ensure that the device is awake before proceeding with readback. Typically, readback occurs long after the wakeup process is completed.

When performing readback via the Slave Parallel interface, it is necessary to send some instructions to the device according to the timing requirements of the Slave Parallel interface. Refer to Readback via the Slave Parallel Interface for details. The instruction stream is presented in 32-bit form and must be sent according to Bit Orders under Different Interface Bit Widths. Once the device correctly receives the instructions, it will output the data from the rising edge of the clock once the data is ready, and then the user can sample the data. The processes to read back bitstreams from a configuration register and configuration memory differ, as described below. For the timing requirements of readback, refer to Typical Timing Diagram of the Slave Parallel Configuration Interface in Chapter 2.

4.2.1 Configuration Memory Readback

Table 4-1 Configuration Memory Readback Process

Process	**Instruction Stream (Hex)/Operation**
100 padding words	FFFFFFFF …… FFFFFFFF
Bus Bit Width Auto Detection	000000AA 08100020
10 padding words	FFFFFFFF …… FFFFFFFF
SYNC WORD	01332D94
Type 1 packet header: No operation	A0000000
Write to the ADRR	AAC00001 00000000[1]
Write an RCMEM command to the CMDR	A8800001 00000006
Write a Type 1 packet header to read the CMEMOR	B1C00000
Write a Type 2 packet header for reading data	5xxxxxxx[2]
Pause operations for at least 40 configuration clock cycles (CS_N is 1, and the clock CFG_CLK continues)	Pull CS_N high, and change RWSEL to read[3] when CS_N is 1
Read the CMEMOR	Pull CS_N low, sample the data, pull CS_N high after reading, and change RWSEL to write[3].
Write an RCMEMDIS command to the CMDR	A8800001 00000011
Write an NOP command to the CMDR	A8800001 A0000000
Write a DESYNC command to the CMDR	A8800001 0000000B
100 NOP Type 1 packet headers	A0000000 …… A0000000

Notes:

1. Write the readback start address to the ADRR.
2. The 32-bit number resulted from readback.

3. For the timing requirements for changes to RWSEL, refer to Slave Parallel Readback Timing.

4.2.2 Configuration Register Readback

The process of reading back bitstreams from a configuration register is shown as follows:

Table 4-2 Configuration Register Readback Process

Process	Instruction Stream (Hex)/Operation
100 padding words	FFFFFFFF …… FFFFFFFF
Bus Bit Width Auto Detection	000000AA 08100020
10 padding words	FFFFFFFF …… FFFFFFFF
Synchronization	01332D94
Type 1 packet header: No operation	A0000000
Write an NOP command to the CMDR	A8800001 A0000000
Write the header of a type 1 packet to read a specified register	Bxxxxxxx[1]
Wait for at least 20 configuration clock cycles	Pull CS_N high and change RWSEL to read[2] when CS_N is 1
Read the specified register	Pull CS_N low, sample the data, pull CS_N high after reading, and change RWSEL to write[2].
Write a DESYNC command to the CMDR	A8800001 0000000B
100 NOP Type 1 packet headers	A0000000 …… A0000000

Notes:

1. According to Packet Types, fill in the Configuration register address to read, as well as the 32-bit number resulted from readback.

2. For the timing requirements for changes to RWSEL, refer to Slave Parallel Readback Timing.

Chapter 5 SEU Detection

Logos2 FPGAs support SEU 1-bit error correction and 2-bit error detection. The ECC algorithm uses the SECDED algorithm, grouped by frame.

SEU error detection and correction can be performed through the internal slave parallel interface.

During SEU error detection and correction, the user logic should save the current readback frame. When a single-bit error occurs, the user logic toggles the erroneous bit and dynamically reconfigures this frame to correct the error.

When a double-bit error occurs, the error cannot be located. Therefore, error correction can only be achieved by dynamically reconfiguring the correct bitstream frame. Errors can occur in any frame. To perform double-bit error correction, the correct bitstream must be accessible, such as being stored in SPI FLASH. In this way, when a double-bit error occurs, the correct bitstream frame can be read and dynamically reconfigured to correct the error.

The SEU function of Logos2 can be implemented with IP; for details, refer to "***UG042001_SEU_IP***".

Chapter 6 Design Protection

6.1 Bitstream Encryption

Malicious cloning of user designs by competitors is one of the common issues during the use of FPGAs. To address this issue, the Logos2 Family FPGAs incorporate a bitstream encryption function to prevent malicious cloning of user designs.

6.1.1 Scenarios without bitstream encryption protection

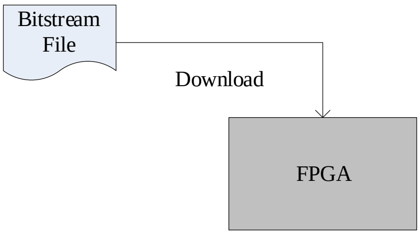


Figure 6-1 Scenario Without Bitstream Encryption

As shown in Figure 6-1, the Logos2 Family FPGA is based on SRAM, requiring the bitstream files to be loaded from an external source every time it is powered on. The user's bitstream file is usually stored in an external storage unit of the FPGA, such as Flash.

Malicious cloners can obtain the bitstream file in various ways. After obtaining the bitstream file, malicious cloners can directly load it onto their own purchased identical FPGA, thereby replicating the exact same FPGA functionality as the user.

6.1.2 Scenarios with bitstream encryption protection

To effectively protect users' intellectual property, the Logos2 Family FPGAs offer the bitstream encryption function.

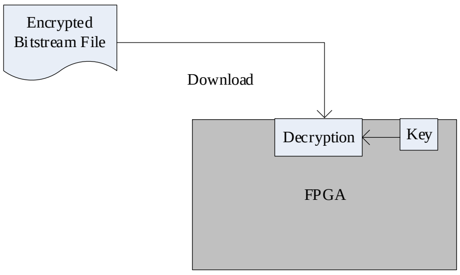


Figure 6-2 Scenario with Bitstream Encryption

As shown in Figure 6-2, the bitstream file is encrypted, and when the encrypted bitstream file is loaded onto the Logos2 Family FPGA, it must be decrypted with the key pre-stored in the chip first. The key of the encrypted bitstream file must match the key stored in the Logos2 FPGA; otherwise, the Logos2 FPGA cannot enter its working state normally.

The key stored in the Logos2 FPGA is pre-written by the user via JTAG to a location such as eFuse, BBRAM or a temporary key register used for testing. eFuse is one-time programmable (OTP) and non-volatile, meaning that the key information will be permanently retained once programmed. Furthermore, the key of Logos2 FPGA features a read protection function; once read protection is enabled, the key that is written cannot be read back. When writing the key into eFuse or BBRAM, the PDS download tool will enable key read protection by default.

Bitstream encryption protection protects users from malicious cloning. Even if a malicious cloner obtains the bitstream file (encrypted), they do not know the key and thus cannot load the bitstream file into the Logos2 FPGA that does not contain the corresponding key.

6.1.3 Encryption Algorithm

The Logos2 Family FPGAs use the AES256-GCM encryption algorithm to encrypt bitstreams, and provide block encryption for non-compressed bitstreams. Block encryption is to divide a bitstream into multiple blocks, each encrypted with a different key. Apart from the first block's key, which is stored normally in the FPGA, each block's key is encrypted along with the previous block. Block encryption improves security, effectively preventing bypass attacks such as DPA attacks. Users can set the number of blocks for block encryption in PDS. A higher number of blocks provides stronger security but also increases the bitstream size, affecting the configuration time.

The AES256-GCM encryption algorithm also supports self-authentication. When self-authentication is enabled, the bitstream will not be loaded without the correct key.

6.1.4 Bitstream Encryption Protection Usage Process

The main usage process for bitstream encryption protection is as follows:

1. Select a key. Users must properly safeguard the key;
2. Generate an encrypted bitstream file with the selected key in PDS;
3. Utilize the PDS's built-in Fabric configuration tool to program the key into the Logos2 Family product. Afterward, users can confidently use the encrypted bitstream file to load the Logos2 FPGA that has undergone the process described in 3).

The steps for generating an encrypted bitstream file (sbit) are as follows: right-click on the [Generate Bitstream] option in PDS and select the [Configure] option for setup, as shown in Figure 6-3. Users can also select the [Process] in the menu for setup.

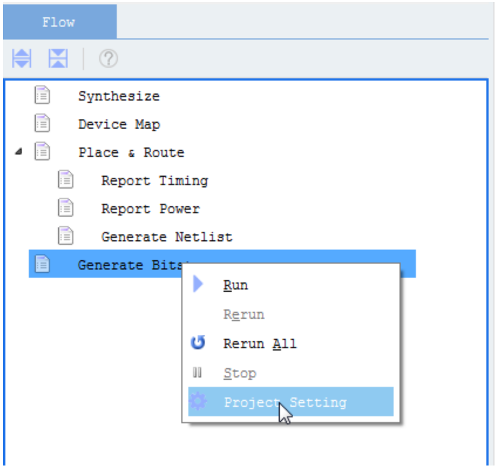
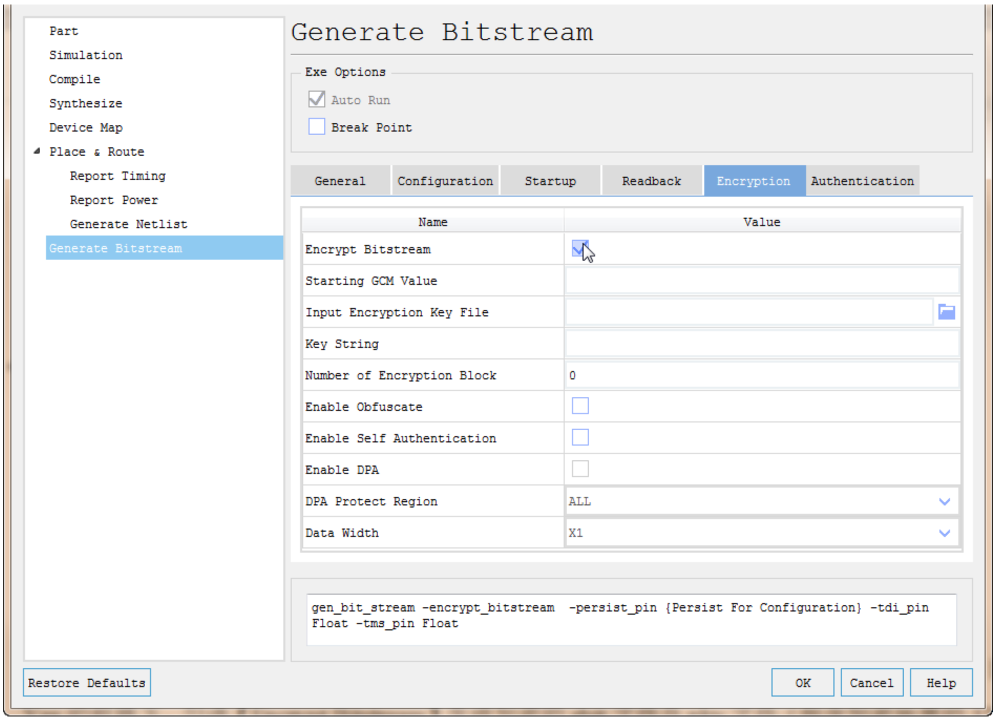


Figure 6-3 Configure Interface

The pop-up window is shown in:



☑ Auto Run
☐ Break Point

Name	Value
Encrypt Bitstream	☑
Starting GCM Value	
Input Encryption Key File	
Key String	
Number of Encryption Block	0
Enable Obfuscate	☐
Enable Self Authentication	☐
Enable DPA	☐
DPA Protect Region	ALL
Data Width	X1

```
gen_bit_stream -encrypt_bitstream  -persist_pin {Persist For Configuration} -tdi_pin
Float -tms_pin Float
```

Figure 6-4 Configure Setup Options

Among these, the [Encrypt Bitstream] option determines whether to encrypt the bitstream file; with [Create Bit File] under [General] configured to Yes, selecting [Encrypt Bitstream] will generate an encrypted sbit file and nky file (containing a 96-bit initial GCM value and a 256-bit key required for encryption). Users can edit the GCM and key string or choose a nky file; if not specified, a nky file is generated randomly.

[Starting GCM Value] allows users to manually enter the initial GCM string;

[Input Encryption Key File] allows users to select a nky file; if a key file is chosen here, the software will encrypt using the key file, regardless of whether the user has entered a key string in the options;

[Key String] allows users to manually enter the key string;

After the settings are complete, click [OK] to close the configuration window. Run [Generate Bitstream] again to generate encrypted bitstream files.

[Enable Obfuscate] allows users to perform optimization on the basis of meeting the industry AES256 algorithm standard to enhance the security of the data stream (since AES is open-source).

[Number of Encryption Blocks] allows users to set the block size for block encryption, measured in the unit of 128 frames (413,696 bits). This value defaults to 0, which means an integral block. The larger this value, the fewer number of blocks.

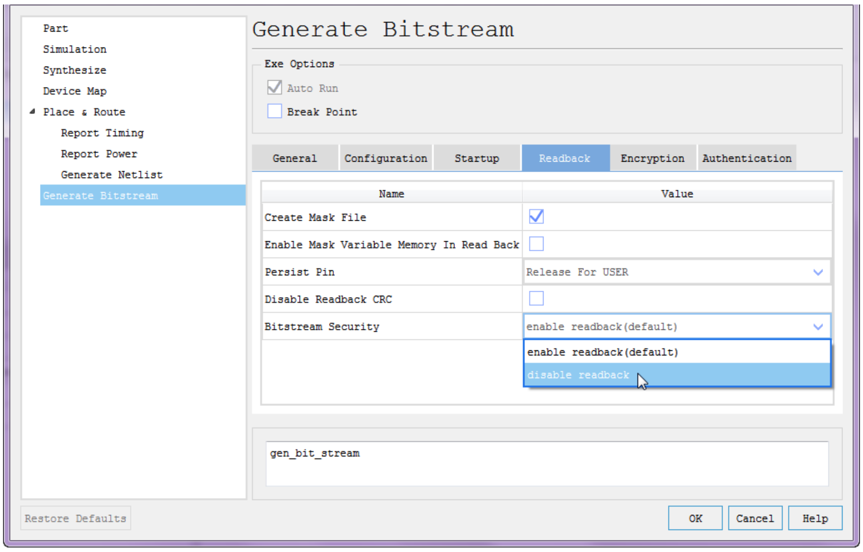
[Enable Self Authentication] Self-authentication enable provided by the AES256-GCM encryption algorithm;

[Data Width] The encryption algorithm unit is 128 bits. When configuring with different bit widths, this value must be set as needed to ensure the device can decrypt normally.

6.2 Readback Protection

By default, Logos2 devices can be reconfigured and read back in user mode via the JTAG interface. If the internal slave parallel interface is instantiated in user design or not used with the Slave Parallel interface reserved, reconfiguration and readback can be done via either the internal slave parallel interface or the Slave Parallel interface. To prevent unauthorised access to the configuration memory, users can choose not to reserve the external configuration interface and not to connect the internal slave parallel interface to the device pins. On the other hand, Logos2 devices also support the readback protection function. Once readback protection is enabled, readback can only be re-enabled through device reset.

Figure 6-5 shows how to disable readback.



Generate Bitstream

Exe Options

☑ Auto Run

□ Break Point

Name	Value
Create Mask File	☑
Enable Mask Variable Memory In Read Back	□
Persist Pin	Release For USER
Disable Readback CRC	□
Bitstream Security	enable readback(default)

gen_bit_stream

Figure 6-5 Disabling Readback

6.3 Bitstream Authentication

In addition to the self-authentication provided by the AES256-GCM encryption algorithm, Logo2 devices also support RSA authentication using the RSA-2048 authentication algorithm.

SHA3-384 is the digest algorithm for the RSA-2048 public key. The digest length of the RSA-2048 public key is 384 bits.

SHAKE256 is the digest algorithm of bitstream. The digest length of bitstream is 2047 bits.

Users can authenticate bitstreams, including normal bitstreams, encrypted bitstreams, and block-encrypted bitstreams.

Authentication lasts from the completion of writing the authentication register to the completion of desynchronisation.

While the bitstreams are being authenticated, the payload is stored in the configuration memory. Users can perform subsequent wakeup operations only after the bitstreams pass the authentication. If authentication fails, an authentication failure flag is generated, with INIT_FLAG_N set to 0.

While the encrypted bitstreams and block-encrypted bitstreams are being authenticated, Logo2 devices perform decryption and store the payload in the configuration memory. Users can perform subsequent wakeup operations only after the bitstreams pass the authentication. If authentication fails, an authentication failure flag is generated, with INIT_FLAG_N set to 0.

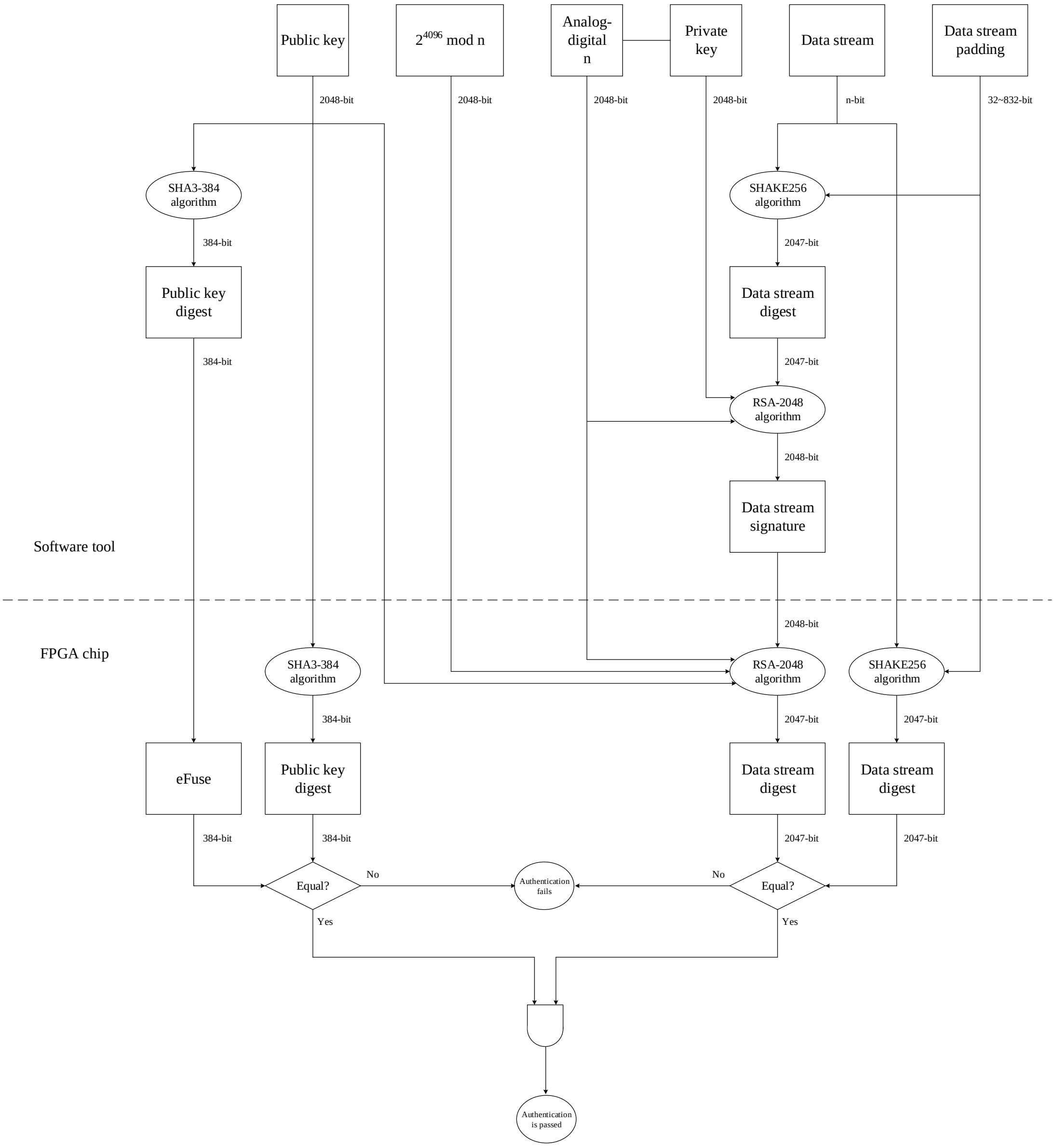


Figure 6-6 Authorization Process Diagram

6.4 DPA Protection

Logos2 devices support AES256-GCM self-authentication and RSA authentication. While bitstreams are encrypted using these two authentication methods, decryption occurs simultaneously. To further prevent DPA attacks, an effective approach is to write the bitstreams to the configuration memory without decrypting them during authentication and decryption, and decrypt and configure bitstreams only after they pass the authentication.

Logos2 devices support DPA protection authentication. When DPA protection is enabled, the encrypted payload is written to the configuration memory while bitstreams are encrypted via DPA protection authentication. After authentication is successful, the encrypted payload is read back frame by frame, decrypted, and written back to the configuration memory, and then the wakeup operation is performed. If authentication fails, an authentication failure flag is generated, with INIT_FLAG_N set to 0.

6.5 UID

A significant risk in the reconfigurable device industry is design piracy and illegal overproduction. Therefore, to prevent such illegal activities, the UID was introduced. Each device has a unique identifier, which is determined at the time of the device's manufacture. Users can read UID through the UID interface (instantiating GTP_UDID) and JTAG interface, and incorporate the results processed with their unique encryption algorithms into the programming bitstream. After each data stream reloading, the device enters user mode, where the user logic reads the UID, processes it with the user's unique encryption algorithm, and compares it with the results in the previous programming bitstream; if there is any difference, then the device will not operate properly.

The 96-bit UID is stored in eFuse and is programmed when the device is manufactured. Each time the device is powered on, the UID from the eFuse is automatically read into the register for user access at any time.

6.6 JTAG Security Management

The JTAG interface provides users with a convenient and comprehensive method for chip configuration, readback, and testing. Users can disable the JTAG interface to keep the chip operating stably without malicious interference.

Logos2 devices support JTAG interface security management. Users can temporarily disable JTAG by instantiating the user JTAG interface (corresponding to GTP_JTAGIF) in the user logic, and this temporary disablement can be lifted through chip power-up or other configuration interfaces. Alternatively, JTAG can be permanently disabled by writing the JTAG disable bit into the eFuse.

After JTAG is temporarily disabled, it can be re-enabled through other interfaces. If GTP_IPAL_E2 is instantiated in the user logic, the user can send a warmboot instruction through the internal slave parallel interface to reset the chip's configuration memory and re-enable the external JTAG interface. If GTP_IPAL_E2 is not instantiated in the user logic and the external Slave Serial and Slave Parallel interfaces are reserved, the external JTAG interface can be enabled by sending a warmboot instruction through the external interface.

6.7 eFuse

As a OTP non-volatile memory, eFuse is used to store device setup information, UID, and other user-programmable bits in Logos 2 Family devices.

1. 256-bit AES-GCM encryption key
2. 384-bit public key digest
3. 32-bit user Fuse

Security settings (authentication enable, DPA protection enable, key lock flag, public key digest lock flag, user Fuse lock flag).

The fuse value is 0 before programming, and 1 after programming. Users can configure all programmable bits through PDS. For detailed instructions, refer to "***Fabric_Configuration_User_Guide***".

Since the JTAG interface can be used to program the eFuse, users should avoid the JTAG interface from interference signals caused by operations such as hot plugging, which may result in erroneous eFuse writing. If interference with the JTAG interface cannot be avoided, lock all eFuse function bits after determining the chip application scenario. Use the TCL instruction "cfg_efuse_lock -lock FF -password 000000" in the Fabric Configuration tool to lock all eFuse function bits.

The 96-bit UID is programmed at the time of device manufacture and can be read out via the JTAG interface or UID interface. Refer to UID. The 32-bit user Fuse can be read via the JTAG interface or the User Fuse interface. Refer to User Fuse Interface.

Chapter 7 User Logic Interfaces

7.1 Internal slave parallel interface

GTP_IPAL_E2 provides an internal slave parallel interface, as well as readback CRC and SEU interfaces. Users can operate the internal slave parallel interface by instantiating this GTP, and conveniently perform readback CRC and SEU operations using the Readback CRC and SEU interfaces. This section mainly introduces the internal slave parallel interface. The SEU function can be implemented with the IP. For details, refer to "***UG042001_SEU_IP***".

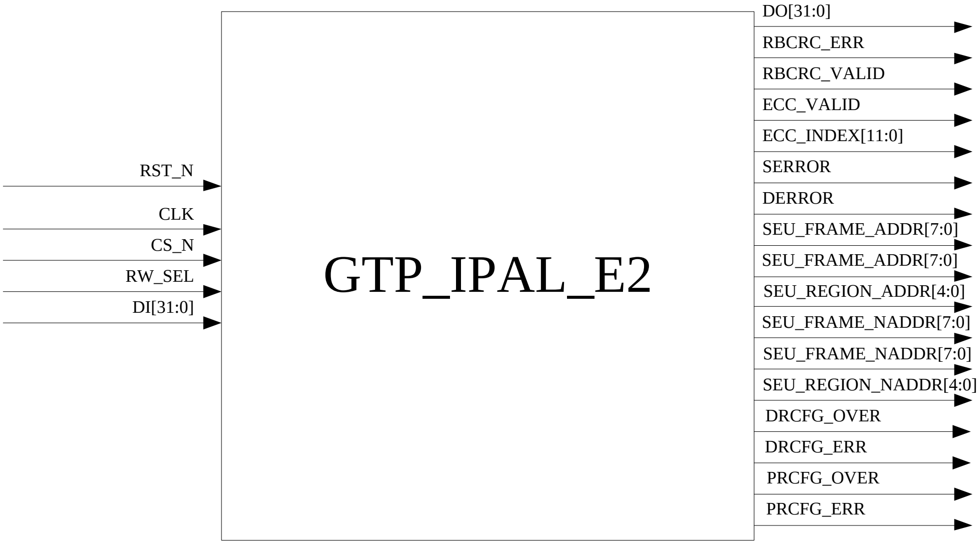


Figure 7-1 GTP_IPAL_E2 Port Diagram

7.1.1 Port List

Table 7-1 GTP_IPAL_E2 Port List

Port	**I/O**	**Bit width**	**Description**
CLK	I	1	Internal slave parallel interface clock
RST_N	I	1	Reset signal, active low, used only for simulation[1]
Internal slave parallel interface			
CS_N	I	1	Chip select signal, active-low
RW_SEL	I	1	Read/write selection: 0 for write, 1 for read;
DI	I	32	Data Input
DO	O	32	Data output
Readback CRC interface			
RBCRC_ERR	O	1	Readback CRC error flag, active-high.

Port	I/O	Bit width	Description
RBCRC_VALID	O	1	Readback CRC valid flag, active-high, lasts for 1 clock cycle.
SEU Detection Interface			
ECC_VALID	O	1	ECC valid flag, active-high.
ECC_INDEX	O	12	Single-bit error address index
SERROR	O	1	SEU detection single-bit error flag
DERROR	O	1	SEU detection double-bit error flag
SEU_FRAME_ADDR	O	8	Current frame address of SEU detection
SEU_COLUMN_ADDR	O	8	Current column address of SEU detection
SEU_REGION_ADDR	O	5	Current region address of SEU detection
SEU_FRAME_NADDR	O	8	Next frame address of SEU detection
SEU_COLUMN_NADDR	O	8	Next column address of SEU detection
SEU_REGION_NADDR	O	5	Next region address of SEU detection
PRCFG_OVER	O	1	Partial reconfiguration completion flag, indicating partial reconfiguration is complete when it is set to 1.
PRCFG_ERR	O	1	Partial reconfiguration error flag, indicating the partial reconfiguration result when PRCFG_OVER is active.
DRCFG_OVER	O	1	Dynamic reconfiguration completion flag, indicating dynamic reconfiguration is complete when it is set to 1.
DRCFG_ERR	O	1	Dynamic reconfiguration error flag, indicating the dynamic reconfiguration result when DRCFG_OVER is active.

Note:

1. This signal is only used to initialise the interface during simulation; there is no such port on the actual hardware, and operating this port is neither necessary nor will it have any effect.

7.1.2 Parameter Definitions

Table 7-2 GTP_IPAL_E2 Parameter List

Parameter Name	Parameter Type	Valid Values	Function Description
IDCODE	Binary	0~32’hffffffff	Device IDCODE, used only for simulation
DATA_WIDTH	String	"X8", "X16", "X32"	Parallel data width

7.1.3 Interface Timing

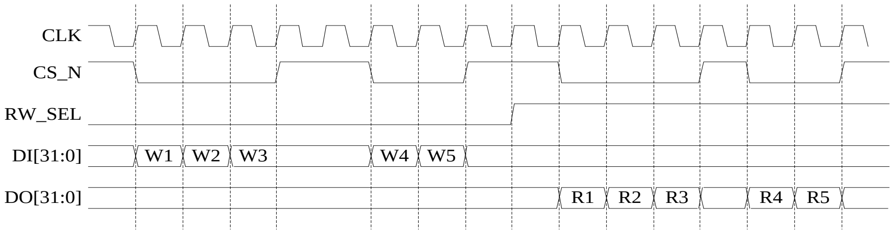


Figure 7-2 Read/Write Timing of GTP_IPAL_E2 Internal Slave Parallel Interface

Note:

The RW_SEL signal transition can only occur when CS_N is high.

7.1.4 Instantiation template

```
GTP_IPAL_E2 #(
        .DATA_WIDTH("X8"),
        .IDCODE('b10101010101010100101010101010101)
) <InstanceName> (
        .DO(),
        .ECC_INDEX(),
        .DI(),
        .DERROR(),
        .ECC_VALID(),
        .RBCRC_ERR(),
        .RBCRC_VALID(),
        .SERROR(),
        .CLK(),
        .CS_N(),
        .RST_N(),
        .RW_SEL(),
        .SEU_FRAME_ADDR(),
        .SEU_COLUMN_ADDR(),
        .SEU_REGION_ADDR(),
        .SEU_FRAME_NADDR(),
        .SEU_COLUMN_NADDR(),
        .SEU_REGION_NADDR(),
        .PRCFG_OVER(),
        .PRCFG_ERR(),
        .DRCFG_OVER(),
        .DRCFG_ERR()
);
```

7.2 UID Interface

7.2.1 Port List

Table 7-3 GTP_UDID Port List

Port	**I/O**	**Bit width**	**Function Description**
DI	I	1	Serial data input
DO	O	1	Serial data output
SE	I	1	Enables data shift
LOAD	I	1	Data registers parallel load UID CODE
CLK	I	1	Clock

7.2.2 Parameter List

Table 7-4 GTP_UDID Parameter List

Parameter Name	**Parameter Type**	**Valid Values**	**Function Description**
UDID_WIDTH	Integer	96	UID length, for simulation only[1]
UDID_CODE	Binary	0~96’hffffffffffffffffffffffff	Chip identity code, for simulation only

Note:

1. The UID length varies between different families. To maintain consistency between simulation and board operation, it should be set to 96 for Logos2 devices.

7.2.3 Interface Timing

The UID is 96-bit long.

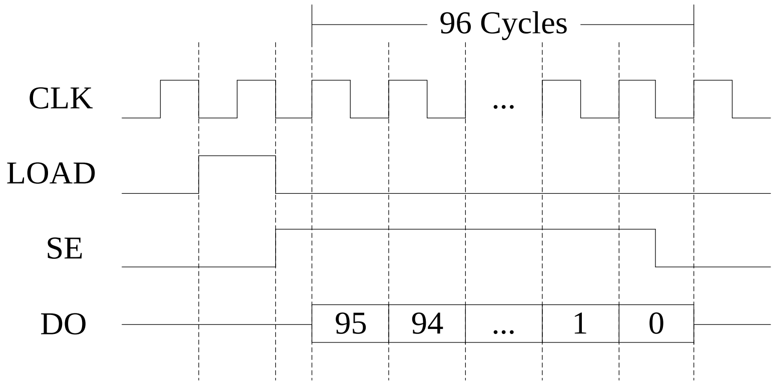


Figure 7-3 Read UID Timing

Users can extend the bit width of the UID.

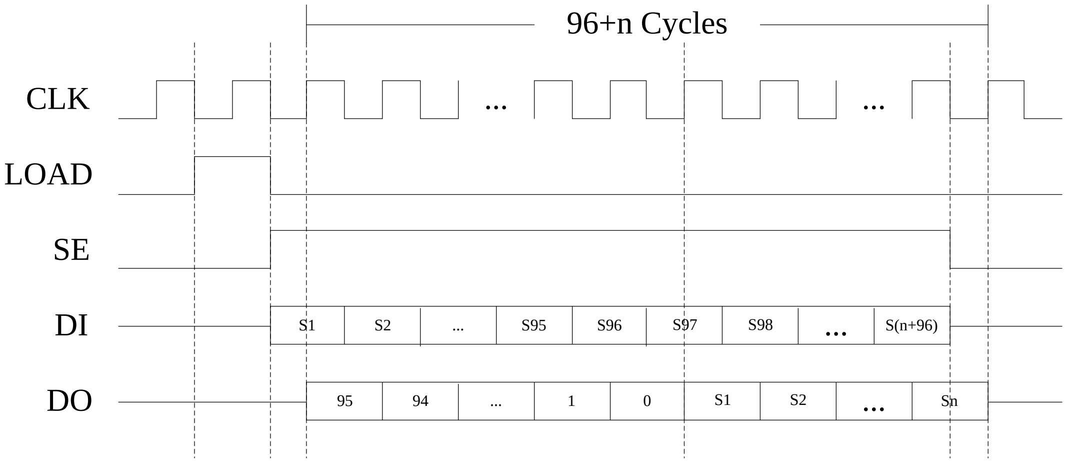


Figure 7-4 Extended UID Bit Width

7.2.4 Instantiation Template

The Verilog instantiation template for GTP_UDID is shown as follows.

```
GTP_UDID #(

        .UDID_WIDTH(96)
        .UDID_CODE('b000000000000000000000000000000000000000000000000000000000000000
        00)

) <InstanceName> (

        .DO(),

        .CLK(),

        .DI(),

        .LOAD(),

        .SE()

);
```

7.3 User JTAG Interface

When the user JTAG interface is instantiated in user logic, the user logic fully takes over the external JTAG interface. The timing and operation method of the user JTAG interface are consistent with those of the external JTAG interface.

7.3.1 Port List

Table 7-5 GTP_JTAGIF Port List

Port	**I/O**	**Bit width**	**Function Description**
TCK	I	1	User JTAG clock, with a maximum frequency of 50M
TMS	I	1	Selects the user JTAG test mode. Controls the state switching of the TAP controller on the rising edge of the clock TCK.
TDI	I	1	Inputs user JTAG test data. Serves as the serial input for JTAG instructions and data registers on the rising edge of the clock TCK.
TDO	O	1	Outputs user JTAG test data. Serves as the serial output for JTAG instructions and data on the falling edge of the clock TCK.

7.3.2 Parameter List

Table 7-6 GTP_JTAGIF Parameter List

Parameter Name	**Parameter Type**	**Valid Values**	**Function Description**
USERCODE	Binary	32’h0~32’hFFFF_FFFF	USERCODE

7.3.3 Instantiation Template

The Verilog instantiation template for GTP_JTAGIF is shown as follows.

```
GTP_JTAGIF #(

      .USERCODE('b11111111111111111111111111111111)

) <InstanceName> (

      TDO(),

      TCK(),

      TDI(),

      TMS()

);
```

7.4 User Fuse Interface

The user logic can read the Fuse user bits by instantiating the user Fuse interface (GTP_EFUSECODE), which outputs a 32-bit user value in an asynchronous parallel manner.

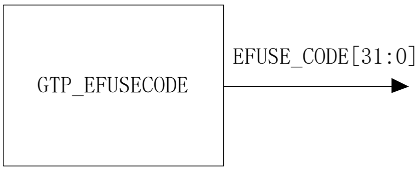


Figure 7-5 GTP_EFUSECODE Port Diagram

7.4.1 Port List

Table 7-7 GTP_EFUSECODE Port List

Port	I/O	Bit width	Function Description
EFUSE_CODE	O	32	Efusecode output data bus

7.4.2 Parameter List

Table 7-8 GTP_EFUSECODE Parameter List

Parameter Name	Parameter Type	Valid Values	Function Description
SIM_EFUSE_VALUE	Binary	32’h0~32’hFFFF_FFFF	EFUSE_CODE simulation value

7.4.3 Instantiation Template

The Verilog instantiation template for GTP_EFUSECODE is shown as follows.

```
GTP_EFUSECODE #(
    .SIM_EFUSE_VALUE('b00010010001101000101011001111000)
) <InstanceName> (
    .EFUSE_CODE()
);
```

Disclaimer

Copyright Notice

Disclaimer

1. This document only provides information in stages and may be updated at any time based on the actual situation of the products without further notice. The Company assumes no legal responsibility for any direct or indirect losses caused by improper use of this document.
2. This document is provided "as is" without any warranties, including but not limited to warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranties mentioned in proposals, specifications, or samples. This document does not grant any explicit or implied intellectual property usage license, whether by estoppel or otherwise.
3. The Company reserves the right to modify any documents related to its family products at any time without prior notice.