ECP5 and ECP5-5G sysCONFIG Usage Guide

Technical Note

FPGA-TN-02039-2.0

September 2021

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CRC	Cyclic Redundancy Check
EBR	Embedded Block RAM
LSB	Least Significant Bit
LUT	Look Up Table
MSB	Most Significant Bit
MSPI	Master Serial Peripheral Interface
PCB	Printed Circuit Board
POR	Power On Reset
SCM	Serial Configuration Mode
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
TCP	Test Clock Pin
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select

1. Introduction

The configuration memory in ECP5™ and ECP5-5G™ FPGAs is built using volatile SRAM; therefore, an external non-volatile configuration memory is required to maintain the configuration data when the power is removed. This non-volatile memory supplies the configuration data to the ECP5 and ECP5-5G devices when it powers up or anytime the device needs to be updated. The ECP5 and ECP5-5G devices provide a rich set of features for programming and configuration of the FPGA. You have many options available to you for building the programming solution that fits your needs. Each of the options available is described in detail so that you can put together the programming and configuration solution that meets your needs. Waveforms presented in this document are for reference only; for detailed timing recommendations refer to ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012).

2. Features

Key programming and configuration features of ECP5 and ECP5-5G devices are:

- Multiple programming and configuration interfaces:
  - 1149.1 JTAG
  - Slave Parallel
  - Slave Serial
  - Slave SPI
  - Master SPI, includes SERIAL, DUAL, and QUAD read mode
  - Dual Boot and Multi Boot support
  - Daisy chaining
- Transparent programming of external memory
- Readback security and encryption for design protection
- Compression of bitstreams

3. Definition of Terms

This document uses the following terms to describe common functions:

- Programming – Programming refers to the process used to alter the contents of the external configuration memory.
- Configuration – Configuration refers to a change in the state of the SRAM memory cells.
- Configuration Mode – The configuration mode defines the method the device uses to acquire the configuration data from the volatile memory.
- Configuration Data – This is the data read from the non-volatile memory and loaded into the FPGA's SRAM configuration memory. This is also referred to as a bitstream, or device bitstream.
- BIT – The BIT file is the configuration data for the device that is stored in an external SPI Flash or other memory device. It is a binary file and is programmed unmodified into the SPI Flash by Lattice programming tool.
- HEX – The HEX file is also a configuration data file for the device. It is in HEX format and is normally requested by third party programming vendors.
- Port – A port refers to the physical connection used to perform programming and some configuration operations. Ports on the ECP5 and ECP5-5G include JTAG and SPI physical connections.
- User Mode – The ECP5 and ECP5-5G devices is in user mode when configuration is complete, and the FPGA is performing the logic functions you have programmed it to perform.
- Offline Mode – Offline mode is a term that is applied to SRAM configuration or external memory programming. When using offline mode, the FPGA no longer operates in user mode. The contents of the SRAM configuration memory or external memory device are updated. The FPGA does not perform your logic operations until offline mode programming/configuration is complete.
- Direct Mode (Foreground Mode) – The device is in a configuration mode and all the I/O pins are kept tristated.
- Background Mode – The device is in a configuration mode where all the I/O pins remain operational.
- Dual Boot – Supports two configuration patterns that reside in a SPI Flash device. Whenever loading failure occurs with the primary pattern, the FPGA device searches for and loads a so-called 'golden' or fail-safe pattern. Both patterns come from an off-chip non-volatile SPI memory.
- Multi Boot – The FPGA device determines and triggers the loading of the next pattern after a prior successful configuration. Multiple patterns (that is more than two patterns) are available for the FPGA device to choose to load on demand. All the patterns are stored in an external SPI Flash memory.
- Transparent Mode – Transparent mode is used to update the SRAM configuration 'while leaving the ECP5 and ECP5-5G devices in user mode.
- Number Formats – The following nomenclature is used to denote the radix of numbers
  - 0x: Numbers preceded by '0x' are hexadecimal
  - b (suffix): Numbers suffixed with 'b' are binary
  - All other numbers are decimal
- SPI – Serial Peripheral Interface Bus. An industry standard, full duplex, synchronous serial data link that uses a four wire interface. The interface supports a single master and single or multiple slaves.
- Master SPI – A configuration mode where the FPGA drives the master clock and issues commands to read the bitstream from an external SPI Flash device.
- Slave SPI (SSPI) – A configuration mode where the CPU drives the clock and issues commands to the FPGA for writing the bitstream into the SRAM cells.
- Refresh – The process of re-triggering a bitstream write operation. It is activated by toggling of the PROGRAMN pin or issuing a REFRESH command, which emulates the PROGRAMN pin toggling. Only the JTAG port and the Slave SPI port support the REFRESH command.

4. Configuration Details

The ECP5 and ECP5-5G SRAM memory contains the active configuration, essentially the “fuses” that define the behavior of the FPGA. The active configuration is, in most cases, retrieved from an external memory. The non-volatile memory holds the configuration data that is loaded into the FPGAs SRAM.

4.1. Bitstream/PROM Sizes

The ECP5 and ECP5-5G are SRAM based FPGAs. The SRAM configuration memory must be loaded from an external non-volatile memory that can store all of the configuration data. The size of the configuration data is variable. It is based on the amount of logic available in the FPGA, and the number of pre-initialized Embedded Block RAM (EBR) components. An ECP5 and ECP5-5G design using the largest device, with every EBR pre-initialized with unique data values, and generated without compression turned on requires the largest amount of storage.

Table 4.1. Maximum Configuration Bits

Device	All Uncompressed	SPI Mode	
	Unencrypted/Encrypted Bitstream Size (Mb)	Recommended SPI Flash Size (Mb)	Dual Boot Recommended SPI Flash
LFE5-12, No EBR	4.45	8	16
LFE5-12, Max EBR	5.42	8	16
LFE5-25, LFE No EBR	4.45	8	16
LFE5-25, LFE5UM-25, LFE25UM5G-25 Max EBR	5.42	8	16
LFE5-45, LFE5UM-45, LFE25UM5G-45No EBR	7.86	8	16
LFE5-45, LFE5UM-45, LFE25UM5G-45 Max EBR	9.74	16	32
LFE5-85, LFE5UM-85, LFE25UM5G-85 No EBR	14.71	16	32
LFE5-85, LFE5UM-85, LFE25UM5G-85 Max EBR	18.35	32	64

Note: Both unencrypted and encrypted bitstreams are the same size. Compression ratio depends on bitstream, so we only provide uncompressed bitstream data.

4.2. Device Status Register

The ECP5 and ECP5-5G devices have a 32-bit device status register, which indicates the status of the device. The READ_STATUS command shifts out this 32-bit internal status of the device. The first bit shifted out on the SO pin is bit 0, and the last bit is bit 31. The device status register can only be read by JTAG, Slave SPI and Slave Parallel port.

Table 4.2. 32-Bit Device Status Register

Bit	Function	Status Value			Comments
		Reset Value	0	1	
0	Transparent Mode	0	No	Yes	Device is currently in Transparent mode
[3:1]	Config Target Selection	000	—	—	000 SRAM array
					001 Efuse
4	JTAG Active	0	No	Yes	JTAG State Machine is currently active
5	PWD Protection	0	No	Yes	Configuration logic is password protected
6	Internal use	0	—	—	Not used
7	Decrypt Enable	0	No	Yes	Encrypted bitstreams are accepted
8	DONE	0	No	Yes	Done bit has been set
9	ISC Enable	0	No	Yes	JTAG instructions are being executed with ISC Enabled

Table 4.2. 32-Bit Device Status Register

Bit	Function	Status Value			Comments
		Reset Value	0	1	
10	Write Enable	0	Not Writable	Writable	Selected configuration target is write-protected from at least one of the following setup: • Selected configuration target security bit is set • Password protection is enabled and password is mismatch
11	Read Enable	0	Not Readable	Readable	Read-protected from at least one of the following setup: • Security bit is set • Password protection is enabled and password is mismatch
12	Busy Flag	0	No	Yes	Configuration logic is busy
13	Fail Flag	0	No	Yes	Last instruction/command execution
14	FEA OTP	0	—	—	Feature row is set to be One-Time Programmable
15	Decrypt Only	0	No	Yes	Only encrypted data is accepted
16	PWD Enable	0	No	Yes	Password protection is enabled
17	Internal use	0	—	—	Not used
18	Internal use	0	—	—	Not used
19	Internal use	0	—	—	Not used
20	Encrypt Preamble	0	No	Yes	Encrypted Preamble is detected
21	Std Preamble	0	No	Yes	Standard Preamble is detected
22	SPIm Fail 1	0	No	Yes	Failed to configure from the primary pattern
[25:23]	BSE Error Code	000	—	—	000 No error
					001 ID error
					010 CMD error - illegal command
					011 CRC error
					100 PRMB error - preamble error
					101 ABRT error - configuration aborted by the user
					110 OVFL error - data overflow error
					111 SDM error - bitstream pass the size of the SRAM array
26	Execution Error	0	No	Yes	Error occur during execution.
27	ID Error	0	No	Yes	ID mismatch with Verify_ID command
28	Invalid Command	0	No	Yes	Invalid command received
29	SED Error	0	No	Yes	SED error detected
30	Bypass Mode	0	No	Yes	Device currently in Bypass mode
31	Flow Through Mode	0	No	Yes	Device currently in Flow Through Mode

4.3. sysCONFIG™ Ports

Table 4.3. ECP5 and ECP5-5G Programming and Configuration Ports

Interface	Port	Description
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	4-wire or 5-wire JTAG Interface
sysCONFIG	SSPI	Slave Serial Peripheral Interface (SPI)
	MSPI (SERIAL, DUAL, QUAD)	Master Serial Peripheral Interface (SPI)
	SPCM	Slave-Parallel (CPU) Interface
	SCM	Slave Serial Interface for daisy chain configuration

4.4. sysCONFIG Pins

The ECP5 and ECP5-5G devices provide a set of sysCONFIG I/O pins that you use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports (such as JTAG, SSPI, MSPI) that are used to interact with the FPGA for programming, configuration, and access of resources inside the FPGA. The sysCONFIG pins in a configuration port group may be active, and used for programming the FPGA, or they can be reconfigured to act as general purpose I/O excluding the dedicated JTAG pins.

Recovering the configuration port pins for use as general purpose I/O requires you to adhere to the following guidelines:

- You must DISABLE the unused port. You can accomplish this by using the Lattice Diamond® Spreadsheet View's Global Preferences tab. Each configuration port is listed in the sysCONFIG options tree.
- You must prevent external logic from interfering with device programming.

Table 4.4 lists the shared sysCONFIG pins of the device, and the default state of these pins in user mode. After programming the ECP5 and ECP5-5G devices, the default state of the SSPI sysCONFIG pins become general purpose I/O. This means you lose the ability to program the ECP5 and ECP5-5G devices using SSPI or SPCM when using the default sysCONFIG port settings. To retain the SSPI or SPCM sysCONFIG pins in user mode, be sure to ENABLE them using the Diamond Spreadsheet View editor.

Unless specified otherwise, the sysCONFIG pins are powered by the VCCIO8 voltage. It is crucial for this to be taken into consideration when provisioning other logic attached to Bank 8.

The function of each sysCONFIG pin is described in detail.

Table 4.4. Default State of the sysCONFIG Pins

sysCONFIG Pins		Pull During Configuration	Configuration Modes			
Name	Type		Slave SPI	Slave Serial	Slave Parallel	Master SPI SERIAL(S), DUAL(D), QUAD(Q),
CFGMDN[2:0]	Dedicated	UP	=001	=101	=111	=010
PROGRAMN	Dedicated	UP	PROGRAMN			
INITN	Dedicated	UP	INITN			
DONE	Dedicated	UP	DONE			
MCLK/CCLK	Dedicated[6]	UP	CCLK	CCLK	CCLK	MCLK
D7	Shared	NO	—	—	D7[3]	—
D6	Shared	NO	—	—	D6[3]	—
D5	Shared	NO	—	—	D5[3]	—
D4	Shared	NO	—	—	D4[3]	—
D3/IO3	Shared	NO	—	—	D3[3]	(S)N/C, (D)N/C, (Q)IO3
D2/IO2	Shared	NO	—	—	D2[3]	(S)N/C, (D)N/C, (Q)IO2
D1/MISO	Shared	NO	MISO	—	D1[3]	MISO
D0/MOSI	Shared	NO	MOSI	—	D0[3]	MOSI
CSN/SN	Shared	UP	SN[1]	—	CSN	—
CS1N	Shared	UP	—	—	CS1N	—
WRITEN	Shared	UP	—	—	WRITEN	—
HOLDN/DI/BUSY/ CSSPIN/ CEN	Shared	UP	HOLDN	DI	BUSY	(S)(D)(Q)CSSPIN[2]
DOUT/CSON	Shared	UP	DOUT	DOUT	CSON	DOUT

Notes:
1. SN should have 4.7 kΩ pull-up resistor on-board for SSPI.
2. CSSPIN should have 4.7 kΩ pull-up on-board resistor for MSPI.
3. D[7:0] should have 10 kΩ pull-up resistor on-board for SPCM.
4. MOSI and MISO should have 10 kΩ pull-up resistor on-board for MSPI.
5. IO[3:2] should have 10 kΩ pull-up resistor on-board for QUAD MSPI.
6. MCLK should have a 1 kΩ pull-up. See Figure 6.1.

4.5. Dedicated sysCONFIG Pins

4.5.1. CFGMDN[2:0]

The Configuration Mode pins, CFGMDN[2:0], are dedicated inputs with weak pull-ups. The CFGMDN pins are sampled on the rising edge of INITN and are used to select the configuration mode (that is what type of device the ECP5 and ECP5-5G devices configure from). External <500 Ω pull-down resistors ensure that the CFGMDN pin senses a low (or logic *0*). 4.7 kΩ pullup resistors are recommended to assure proper operation. The JTAG TAP port remains operative at all times, independent of the CFGMDN[2:0] setting. As a consequence, the CFGMDN pins determine which groups of dual-purpose pins is used for device configuration.

Table 4.5. CFGMDN Mode Settings

Configuration Mode	Bus Size	Options	Clock	CFGMDN2	CFGMDN1	CFGMDN0
SSPI	1	—	CCLK	0	0	1
MSPI	1	Serial (SPI_Serial)	MCLK	0	1	0
	2	Dual (SPI_Dual)				
	4	Quad (SPI_Quad)				
	1, 2, 4	Dual-Boot				
	1, 2, 4	Multi-Boot				
SCM (Slave_Serial)	1	—	CCLK	1	0	1
SPCM (Slave_Parallel)	8	—	CCLK	1	1	1

4.5.2. PROGRAMN

PROGRAMN is an input used to configure the FPGA. The PROGRAMN pin is low level sensitive, and has an internal weak pull-up. When PROGRAMN is asserted low, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier. Holding the PROGRAMN pin low prevents the ECP5 and ECP5-5G devices from leaving the Initialization phase. The PROGRAMN has a minimum pulse width assertion period in order for it to be recognized by the FPGA. You can find this minimum time in the AC timing section of ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012).

Be aware of the following special cases when the PROGRAMN pin is active:

- If the device is currently being programmed through JTAG then PROGRAMN is ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration interrupts the process and restart the configuration cycle.
- PROGRAMN must not be asserted low until after all power rails have reached stable operation. This can be achieved by either placing an external pull-up resistor and tying it to the VCCIO8 voltage, or permitting the FPGA's internal pull-up resistor to pull the input high.
- PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state. PROGRAMN must be asserted for a minimum low period of tPRGMRJ in order for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.

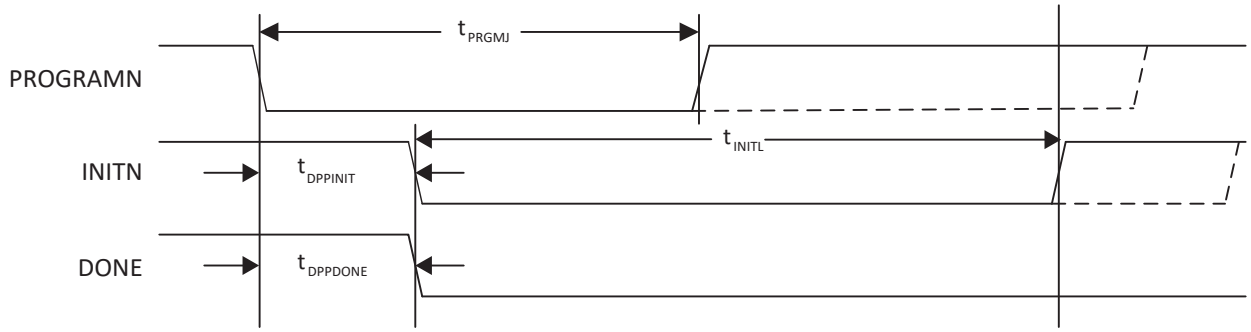


Figure 4.1. Configuration from PROGRAMN Timing

4.5.3. INITN

The INITn pin is a bi-directional open-drain control pin. It has the following functions:

- After power is applied, after a PROGRAMN assertion, or a REFRESH command it goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the t_{INTIL} parameter.
- After the t_{INTIL} time period has elapsed the INITn pin is deasserted (that is active high) to indicate the ECP5 and ECP5-5G devices are ready for its configuration bits. The ECP5 and ECP5-5G devices begin loading configuration data from an external SPI Flash.
- INITn can be asserted low by an external agent before the t_{INTIL} time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple programmable devices chained together. The programmable device with the longest t_{INTIL} time can hold all other devices in the chain from starting to get data until it is ready itself.
- The last function provided by INITn is to signal an error during the time configuration data is being read. Once t_{INTIL} has elapsed and the INITn pin has gone high, any subsequent INITn assertion signals the ECP5 and ECP5-5G devices have detected an error during configuration.

The following conditions causes INITN to become active, indicating the Initialization state is active:

- Power has just been applied
- PROGRAMN falling edge occurred
- The IEEE 1532 REFRESH command has been sent using a slave configuration port (JTAG or SSPI). If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bitstream and forcing the FPGA into the Initialization state.

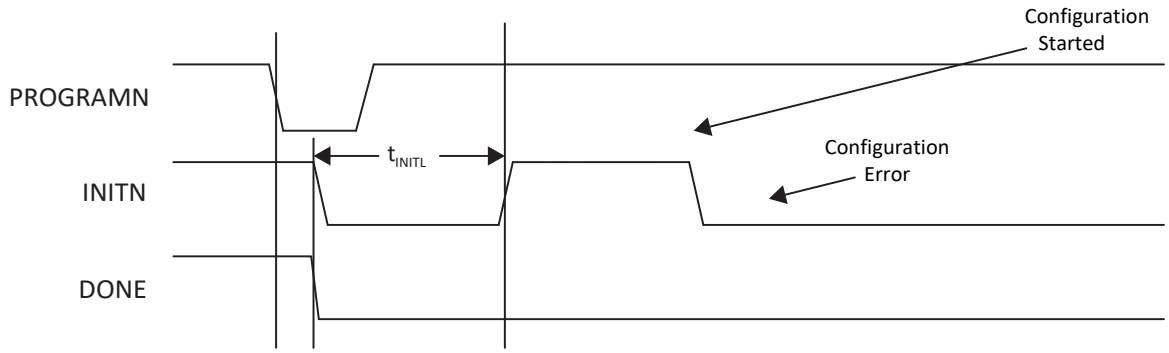


Figure 4.2. Configuration Error Notification

If an error is detected when reading the bitstream, INITN goes low, the internal DONE bit is not set, the DONE pin stays low, and the device does not wake up. The device fails configuration when the following happens:

- The bitstream CRC error is detected
- The invalid command error detected
- A time out error is encountered when loading from the external Flash. This can occur when the device is in MSPI configuration mode and the SPI Flash device is not programmed.
- The program done command is not received when the end of on-chip SRAM configuration or external Flash memory is reached

4.5.4. DONE

The DONE pin is a bi-directional open drain with a weak pull-up that signals the FPGA is in user mode. DONE is first able to indicate entry into user mode only after an internal DONE bit is asserted. The internal DONE bit defines the beginning of the FPGA Wake-Up state.

The FPGA can be held from entering user mode indefinitely by having an external agent keep the DONE pin asserted low. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each

FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison.

The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received through an active configuration port.

Sampling the DONE pin is a way for an external device to tell if the FPGA has finished configuration. However, when using IEEE 1532 JTAG to configure SRAM the DONE pin is driven by a boundary scan cell, so the state of the DONE pin has no meaning during IEEE 1532 JTAG configuration.

4.5.5. MCLK/CCLK

The MCLK/CCLK, when active, are clocks used to sequentially load the configuration data for the FPGA. As an input, CCLK is a dedicated input pin that supports all slave configuration modes. The CCLK pin is defined as an input configuration clock only to support all slave mode configuration modes.

When Master Configuration mode is used, MCLK is an output clock with the frequency you set. The output is used to drive to external memory device.

The MCLK/CCLK pin's default state is determined by the state of the CFGMDN[2:0] settings. The maximum CCLK frequency and the data setup/hold parameters can be found in the AC timing section of ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012).

The MCLK/CCLK pin functions as a Master Clock (MCLK) when the CFGMDN[2:0] is set to MSPI mode. The MCLK becomes an output and provides a reference clock for an SPI Flash attached to the Master SPI Configuration port of the ECP5 and ECP5-5G devices. MCLK actively drives until all of the configuration data has been received. When the ECP5 and ECP5-5G devices enter user mode the MCLK output tristates. The MCLK is always reserved for use in MSPI mode, in most post-configuration applications, as the reference clock for performing memory transactions with the external SPI PROM. See Master SPI Modes section for details.

The ECP5 and ECP5-5G devices generate MCLK from an internal oscillator. The initial frequency of the MCLK is nominally 2.4 MHz. The MCLK frequency can be altered using the MCCLK_FREQ parameter. You can select the MCCLK_FREQ using the Diamond Spreadsheet View. For a complete list of the supported MCLK frequencies, see Table 4.6.

Table 4.6. ECP5 and ECP5-5G MCLK Valid Frequencies

MCLK Frequency (MHz)
2.4
4.8
9.7
19.4
38.8
62

At startup, the lowest frequency MCLK is used by the FPGA. During the initial stages of device configuration, the frequency value specified using MCCLK_FREQ is loaded into the FPGA. Once the ECP5 and ECP5-5G devices accept the new MCLK_FREQ value the MCLK output begins driving the selected frequency. Make certain when selecting the MCLK_FREQ that you do not exceed the frequency specification of your configuration memory, or of your PCB. When making MCLK_FREQ decisions, review the ECP5 and ECP5-5G devices AC specifications in ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012).

4.6. Dual-Purpose sysCONFIG Pins

The following is a list of the dual-purpose sysCONFIG pins. If any of these pins are used for configuration and for user I/O, you must adhere to the requirements listed at the start of the Configuration pin sections. These pins are powered by VCCIO8.

4.6.1. HOLDN/DI/BUSY/CSSPIN

HOLDN – When Slave SPI mode is used, the HOLDN pin is an asynchronous active low input that tristates the serial read out data of the SPI port and sets the device to the suspend state by ignoring the clock. HOLDN is only available in SSPI mode during configuration.

DI – Data Input for Serial Configuration Mode (SCM). DI has an internal weak pull-up and captures the data on the rising edge of CCLK.

BUSY – In Slave Parallel configuration mode, the BUSY pin is a tristated output. The BUSY pin is driven low by the device only when a byte of data is ready for reading. Set the SLAVE_PARALLEL_PORT to ENABLE to retain the pin as the BUSY pin to access the Slave Parallel port in user mode.

CSSPIN – In MSPI mode, the CSSPIN becomes a low true Chip Select output that drives the SPI Serial Flash chip select. If SPI memory needs to be accessed using the SPI port while the part is in user mode (the DONE pin is high) then the MASTER_SPI_PORT= ENABLE, must be used to preserve this pin as CSSPIN. When the ECP5 and ECP5-5G devices CFGMDN[2:0] pins are not in MSPI mode, the CSSPIN is a general purpose I/O with a weak pulldown. Adding a 4.7 kΩ to 10 kΩ pull-up resistor to CSSPIN pin on the ECP5 and ECP5-5G devices is recommended. CSSPIN must ramp in tandem with the SPI PROM VCC input. It remains a general purpose I/O when the FPGA enters user mode.

4.6.2. DOUT/CSON

This is an output pin that has the following purposes.

DOUT – For both serial and parallel configuration modes. When the Bypass mode is selected, this pin becomes a data output pin for serial daisy chaining. When the device is fully configured, the Bypass instruction contained within the bitstream is executed and the data on DI, or D[0:7] in the case of a parallel configuration mode, is then routed to the DOUT pin. This allows data to be passed, serially, to the next device. In a parallel configuration mode, D0 is shifted out first followed by D1, D2, and so on.

CSON – For parallel daisy chaining implemented with the Flow-through attribute, this attribute allows the CSON pin to be driven when the done bit is set and configuration of the first device is complete. The CSON of the first device drives the CSN of the second part.

You will find this attribute in the Diamond Generate Bitstream Data property under Chain Mode or in the Diamond Bitstream Strategy options window.

The DOUT/CSON drives out a high on power-up and continues to do so until the execution of the Bypass/Flow through instruction within the bitstream, or until the I/O Type is changed by the user code.

4.6.3. CSN/SN

CSN/SN is a bi-directional pin with following functions.

CSN – This is an active low input pin with a weak pull-up used in parallel mode only. See the CS1N pins details for more information.

SN – When Slave SPI mode is used, this pin is an active low chip select input. Set the SLAVE_SPI_PORT to ENABLE to retain the pin as the SN pin to access the Slave SPI port in user mode. Lattice recommends that the SN pin be pulled high externally to augment the weak internal pull-up.

4.6.4. CS1N

CS1N – CS1N is an active low input pin. Both CSN and CS1N are OR'ed and used to enable the D[0:7] data pins to receive or output a byte of data in parallel mode.

When CSN or CS1N is high, the D[0:7], and BUSY pins are tristated. Both CSN and CS1N need to be driven low to enable the parallel port. Driving CSN high pauses the data transfer and driving CS1N high resets port. Driving both CSN and CS1N high causes the ECP5 and ECP5-5G devices to exit the Bypass or Flow-through modes and resets the Bypass register. Set SLAVE_PARALLEL_PORT to ENABLE to retain the pin as CS1N pin to access the SLAVE PARALLEL port in user mode.

4.6.5. WRITEN

This pin is used to determine the direction of the data pins D[0:7]. The WRITEN pin must be driven low in order to clock a byte of data into the device and driven high to clock data out of the device. If SRAM (configuration memory) needs to be accessed using the parallel pins while the part is in user mode (the DONE pin is high) then the SLAVE_PARALLEL_PORT must be set to ENABLE to preserve this pin as WRITEN.

Note: Pulling WRITEN low causes the master SPI to issue a 0XFF write on MOSI at the start of configuration (this is designed to force a reset of quad-SPI devices). Pulling WRITEN high causes master SPI to begin configuration immediately without the 0xFF block.

4.6.6. D0/MOSI

D0 – In parallel mode this pin is D[0] and operates in the same way as D[7:0] below.

MOSI – In slave SPI mode, the MOSI pin is the serial input for SPI command and data.

MOSI/IO0 – In MSPI configuration mode, the MOSI pin becomes an output that drives control and data to an SPI Flash. Control and data are output on the falling edge of MCLK.

Set the SLAVE_SPI_PORT to ENABLE or set the MASTER_SPI_PORT to ENABLE to retain the pin as the MOSI pin to access the SPI port in user mode.

4.6.7. D1/MISO

D1 – In parallel mode this pin is D[1] and operates in the same way as D[7:0] below.

MISO – In slave SPI mode, the MISO pin is the serial output data from FPGA.

MISO/IO1 – In MSPI configuration mode, the MISO pin is serial data input.

Set the SLAVE_SPI_PORT to ENABLE or set the MASTER_SPI_PORT to ENABLE to retain the pin as the MISO pin to access the SPI port in user mode.

4.6.8. D[7:0]

These are tristated bi-directional I/O pins. A byte of data is driven into or read from these pins. Taken together D[7:0] form the parallel data bus. If SRAM (configuration memory) needs to be accessed using the parallel pins while the part is in user mode (the DONE pin is high) then the SLAVE_PARALLEL_PORT=ENABLE must be set to preserve these pins.

When the WRITEN signal is low and CSN is low, these pins are inputs. When the WRITEN signal is driven high and CSN is low, these pins are output pins. These pins are enabled by the CSN signal. D[0:7] is always power-up tristated with no pull-up.

4.6.9. IO[3:0]

These pins are used in conjunction with advanced DUAL or Quad SPI Flash memory interfaces. Dependent on the targeted read mode, these pins provide data to and from the external memory.

4.6.10. PERSISTENT

The internal PERSISTENT control bits are used to determine whether the dual-purpose sysCONFIG pins remain sysCONFIG pins during normal operation. The ECP5 and ECP5-5G devices have several PERSISTENT physical SRAM cells that determine the existence of either the Slave parallel port, the Master SPI port or a Slave SPI port when entering the user mode.

Table 4.7. sysCONFIG Pins Global Preferences

Port Setting	Pins Affected	Details
SLAVE_PARALLEL_PORT	D[0:7], CSN, CS1N, WRITEN, BUSY	If enabled, persisted for configuration purpose.
SLAVE_SPI_PORT	MOSI, MISO, SN	If enabled, persisted for configuration purpose.
MASTER_SPI_PORT	MOSI, MISO, CSSPIN	If enabled, persisted for configuration purpose.

4.7. JTAG Configuration Port Pins

The JTAG pins provide a standard IEEE 1149.1 Test Access Port (TAP). Programming and configuration over the JTAG port uses IEEE 1532 compliant commands. In addition to the IEEE 1532 capabilities, the ECP5 and ECP5-5G devices provide all of the mandatory IEEE 1149.1 Test Access Port commands allowing printed circuit board assembly verification.

The JTAG port is always an available port in the ECP5 and ECP5-5G devices.

When the device is programmed through IEEE 1149.1 control, the sysCONFIG programming pins, such as DONE, cannot be used to determine programming progress. This is because the state of the boundary scan cell drives the pin, per the IEEE JTAG standard, rather than normal internal logic.

Table 4.8. JTAG Port Pins

Pin Name	Pin Function	Pin Direction
TDI	TDI	Input with weak pull-up
TDO	TDO	Output with weak pull-up
TCK	TCK	Input
TMS	TMS	Input with weak pull-up
VCCIO8	JTAG Power Supply	N/A

Note: JTAG is only active once the POR circuit has triggered. See Power-up Sequence section for more details.

TDO – The Test Data Output (TDO) pin is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin is in a high impedance state. The only time TDO is not in a high impedance state is when the JTAG state machine is in the Shift IR or Shift DR state. This pin should be wired to TDO of the JTAG connector, or to TDI of a downstream device in a JTAG chain. An internal pull-up resistor on the TDO pin is provided. The internal resistor is pulled up to VCCIO8.

TDI – The Test Data Input (TDI) pin is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to VCCIO8.

TMS – The Test Mode Select (TMS) pin is an input pin that controls the progression through the 1149.1 compliant state machine states. The TMS pin is sampled on the rising edge of TCK. The JTAG state machine remains in or transitions to a new TAP state depending on the current state of the TAP, and the present state of the TMS input. An internal pull-up resistor is present on TMS per the JTAG specification. The internal resistor is pulled to the VCCIO8.

TCK – The test clock pin (TCK) provides the clock used to time the other JTAG port pins. Data is shifted into the instruction or data registers on the rising edge of TCK and shifted out on the falling edge of TCK. The TAP is a static design permitting TCK to be stopped in either the high or low state. The maximum input frequency for TCK is specified in the DC and Switching Characteristics section of ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012). The TCK pin does not have a pull-up. An external pull-down resistor of 4.7 kΩ is recommended to avoid inadvertently clocking the TAP controller as power is applied to the ECP5 and ECP5-5G devices.

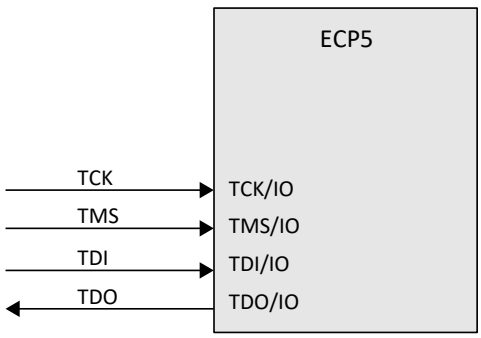


Figure 4.3. JTAG Port

5. Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration and wake-up.

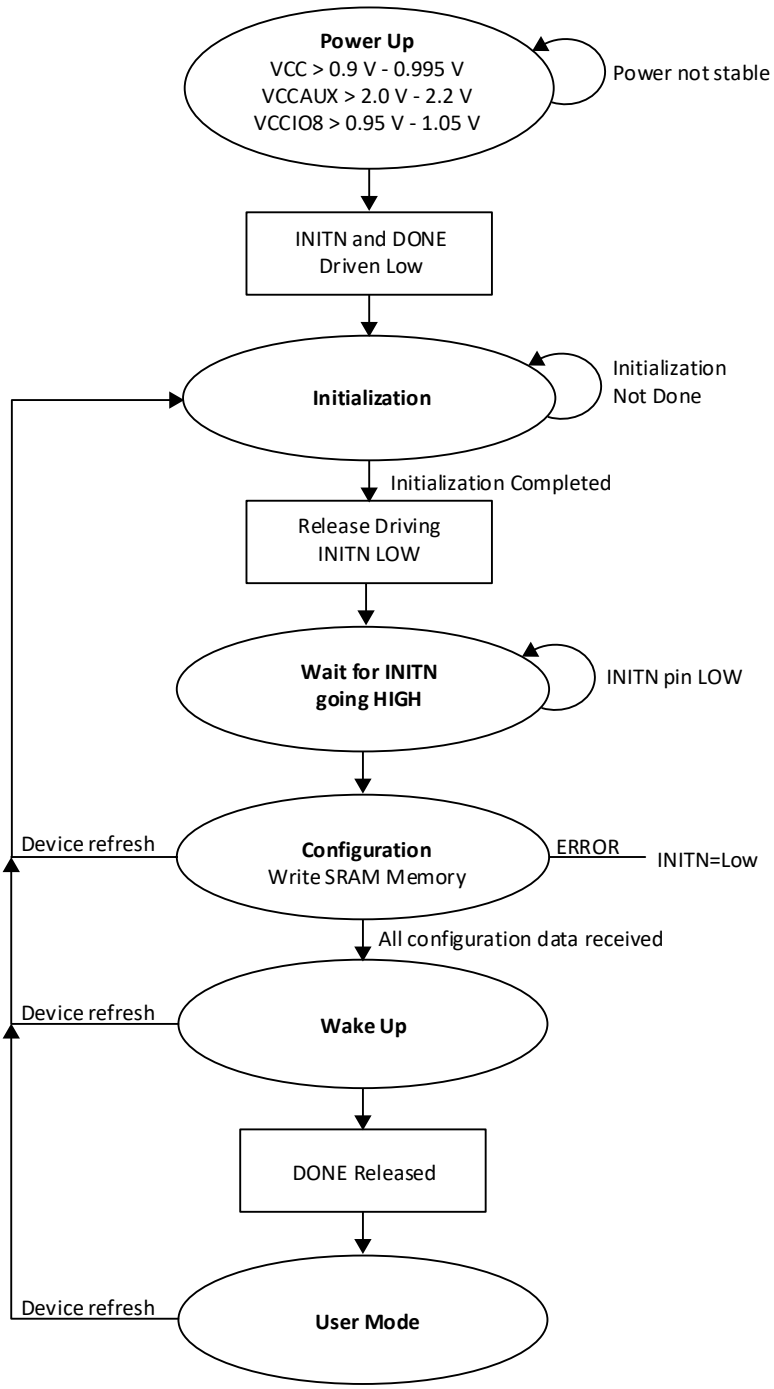


Figure 5.1. Configuration Flow

The ECP5 and ECP5-5G devices sysCONFIG ports provide industry standard communication protocols for programming and configuring the FPGA. Each of the protocols shown in Table 4.3 provides a way to access the configuration SRAM of the ECP5 and ECP5-5G devices. The Reconfiguration Priority section provides information about the capabilities of each sysCONFIG port.

5.1. Power-up Sequence

In order for the ECP5 and ECP5-5G devices to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA has an indeterminate state.

As power continues to ramp, a Power On Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the V_{CC} and V_{CCIO8} input rails. The POR circuit waits for the following conditions:

- V_{CC} > 0.9 V – 0.995 V
- V_{CCIO8} > 0.95 V – 1.05 V
- V_{CCAUX} > 2.0 V – 2.2 V

When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The ECP5 and ECP5-5G devices assert INITN active low, and drives DONE low. When INITN and DONE are asserted low the device moves to the initialization state, as shown in Figure 5.1.

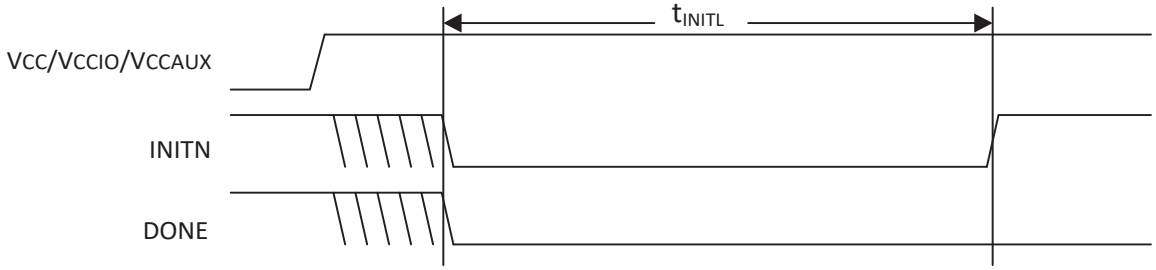


Figure 5.2. Configuration from Power-On-Reset Timing

5.2. Initialization

The ECP5 and ECP5-5G devices enter the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all of the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all of the following conditions are met:

- The t_{INITL} time period has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the t_{INITL} time period the FPGA is clearing the configuration SRAM. When the ECP5 and ECP5-5G devices are part of a chain of devices each device has different t_{INTIL} initialization times. The FPGA with the slowest t_{INTIL} parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

The GPIO of the device at power-up defaults to tristated outputs with active weak input pull-downs. After configuration, all GPIO included in the user design wakes up in the user-defined condition. GPIO not defined in the user design remains output tristated and the input has a weak pull-down enabled. This default avoids inadvertent effects of the inputs rising while powering up. In some cases, this can cause a problem if other connected devices on the board reset or trigger from an active high signal.

Before/during configuration, the D[7:0] pins have no pull-up/down. Other dedicated and dual-purpose I/O with pull-up exceptions, such as PROGRAMN, DONE, and so on are excluded from the GPIO default setting.

5.3. Configuration

The rising edge of the INITN pin causes the FPGA to enter the configuration state. The FPGA is able to accept the configuration bitstream created by the Diamond development tools.

The ECP5 and ECP5-5G devices begin fetching configuration data from non-volatile memory. The ECP5 and ECP5-5G devices do not leave the Configuration state if there is no valid configuration data.

INITN is used to indicate an error exists in the configuration data. When INITN is active high configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA does not operate.

5.4. Wake-up

Wake-up is the transition from configuration mode to user mode. The ECP5 and ECP5-5G devices' fixed four-phase wake-up sequence starts when the device has correctly received all of its configuration data. You can sequence the order of these four phases to meet specific implementation requirements. When all configuration data is received, the FPGA asserts an internal DONE status bit. The assertion of the internal DONE causes a Wake Up state machine to run that sequences four controls. The four control strobes are:

- Global Set/Reset (GSR)
- Global Output Enable (GOE)
- External DONE
- Global Write Disable (GWDISn)

One phase of the Wake-Up process is for the FPGA to release the Global Output Enable. When it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on their programmed output function. The FPGA inputs are always active. The input signals are prevented from performing any action on the FPGA flip-flops by the assertion of the Global Set/Reset (GSR).

Other phases of the Wake-Up process release the Global Set/Reset and the Global Write Disable controls.

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O flip-flops, Look Up Table (LUT) flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the *GSR enabled* attribute to be set/cleared per their hardware description language definition.

The Global Write Disable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. As mentioned previously, the inputs on the FPGA are always active. Keeping GWDIS asserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

Another phase of the Wake-Up process asserts the external DONE pin. The external DONE is a bi-directional, open-drain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the FPGA from proceeding. Only after the external DONE, if enabled, is active high does the final wake-up phase complete. Wake-Up completes uninterrupted when the external DONE pin is not enabled.

Once the final wake-up phase is complete, the FPGA enters user mode. This process is detailed later in the document.

5.5. User Mode

The ECP5 and ECP5-5G devices enter user mode immediately following the Wake-Up sequence has completed. User Mode is the point in time when the ECP5 and ECP5-5G devices begin performing the logic operations you designed. The device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received through one of the configuration ports
- Power is cycled or power supply levels drop below their specified trigger levels

5.6. Clearing the Configuration Memory and Re-initialization

The current user mode configuration of the ECP5 and ECP5-5G devices remain in operation until they are actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the ECP5 and ECP5-5G devices. The first is to remove power and reapply power. Another method is to toggle the PROGRAMN pin. Lastly you can reinitialize the memory through a Refresh command. Any active configuration port can be used to send a Refresh command.

Invoking one of these methods causes the ECP5 and ECP5-5G devices to drive INITN and DONE low. The ECP5 and ECP5-5G devices enter the initialization state as described earlier.

5.7. Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. There is a priority as to which of these sources can interrupt the configuration process depending on which of the sources initiated the original configuration. Note that if an interruption occurs, the reconfiguration occurs without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event causes a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It interrupts any current configuration other than a JTAG configuration.

8-bit slave-parallel mode (SPCM) does not interrupt JTAG or PROGRAMN pin initiated configurations.

6. Configuration Modes

The ECP5 and ECP5-5G devices provide multiple options for loading the configuration SRAM from a non-volatile memory. The previous section described the physical interface necessary to interact with the configuration logic of the ECP5 and ECP5-5G devices. This section focuses on describing the functionality of each of the different configuration modes. Descriptions of important settings required in the Diamond Spreadsheet View are also discussed.

6.1. Master SPI Modes

The Master SPI port is considered an intelligent port. It is capable of performing read and write actions based on the command shifted into the device. All commands are built inside the bitstreams. In Master SPI mode (MSPI), the ECP5 and ECP5-5G devices begin retrieving configuration data from the SPI Flash when power is applied, a REFRESH command is received, or the PROGRAMN pin is asserted and released. This occurs when the CFGMDN[2:0] pins are set to [010] respectively. One MSPI mode CFGMDN setting supports several different submodes such as SLOW or FAST SPI speed, DUAL-boot and MULTI-boot, as well as supporting DUAL and QUAD read SPI Flash devices. All the different submodes are supported by different command build inside the bitstream. Diamond flow only generates default submode bitstream, which is serial and slow read. To change to a different submode, you must alter the bitstream using Lattice Deployment Tool. The MCLK/CCLK I/O takes on the Master Clock (MCLK) function, and begins driving a nominal 2.4 MHz clock to the SPI Flash's SCLK input. CSSPIN is asserted low, commands are transmitted to the PROM over the SI/SISPI output, and data is read from the PROM on the SO/SPISO input pin. When all of the configuration data is retrieved from the PROM, the CSSPIN pin is deasserted, and the MSPI output pins are tristated.

Table 6.1. Master SPI Configuration Port Pins

Pin Name	Function	Direction	Description
MCLK/CCLK	MCLK	Output with weak pullup	Master clock used to time data transmission/reception from the ECP5 and ECP5-5G devices Configuration Logic to a slave SPI
CSSPIN[1]	CSSPIN	Output	Chip select used to enable an external SPI PROM containing configuration data.
MOSI/IO0[2]	MOSI	Output	Carries output data from the ECP5 and ECP5-5G devices Configuration Logic to the slave SPI PROM.
	IO0	Input	Input pin for bitstream data as DUAL and QUAD read mode
MISO/IO1[2]	MISO	Input	Carries input data from the slave SPI PROM to the ECP5 and ECP5-5G devices Configuration Logic.
	IO1	Input	Input pin for bitstream data as DUAL and QUAD read mode.
IO[3:2][2]	IO[3:2]	Input	This is the input pins for bitstream data from SPI Flash, used only on QUAD read mode.

Notes:

1. Use 4.7 kΩ pull-up resistor
2. Use 10 kΩ pull-up resistor

The MCLK frequency always starts downloading the configuration data at the nominal 2.4 MHz frequency. The MCCLK_FREQ parameter, accessed using Spreadsheet View, can be used to increase the configuration frequency. The configuration data in the PROM has some padding bits, and then the data altering the MCLK base frequency is read. The ECP5 and ECP5-5G devices read the remaining configuration data bytes using the new MCLK frequency.

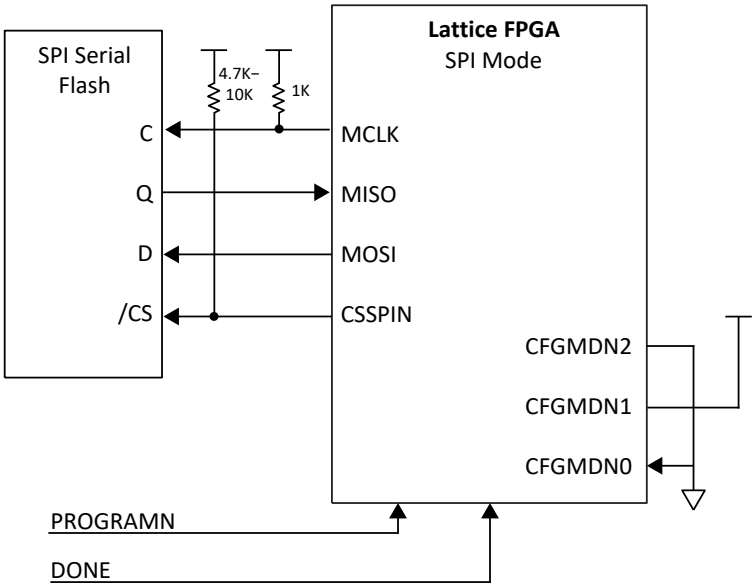


Figure 6.1. ECP5 and ECP5-5G Master SPI Port with SPI Flash

Once the SPI Flash contains your configuration data, you can test the configuration. Assert the PROGRAMN, transmit a REFRESH command, or cycle power to the board, and the ECP5 and ECP5-5G devices configures from the external SPI Flash.

6.1.1. Method to Enable the Master SPI Port

Similar to all configuration ports, the Master SPI port is enabled by two standard methods.

- Setting the CFGMDN[2:0] pin to [0, 1, 0].

  When the device is powered up, or when the PROGRAMN pin is toggled, the device checks the state of the CFGMDN pins. If they are set to [0, 1, 0], then the device chooses the Master SPI port as the configuration port. A port is said to be a configuration port when it is capable of executing both bitstream write and read commands. And this is the only method that you can perform DUAL read and QUAD read from SPI Flash.

- Enabling Master SPI port persistence.

  The configuration bitstream contains optional Master SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the Master SPI port is to remain operational once in user mode. This selection is independent of the CFG pins setting. A port enabled by persistence is a Background Mode port. It mainly used for background self-reconfiguration upon SED failure when you set SED auto correction.

  Note that both the DONE pin and the INITN pin must be high. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

6.1.2. Customized SPI Port in User Mode

After the ECP5 and ECP-5G devices enter user mode, the Master SPI configuration port pins are tristated with a weak pull-up. This allows the SPI pins to be used as user I/O except MCLK/CCLK which is tristated. You can perform SPI data transfers using the SPI pins but need to use a soft (RTL) Master SPI Controller to do this. The ECP5 and ECP5-5G devices provide a solution for you to choose any user clock as MCLK under this scenario by instantiating USRMCLK macro in your Verilog or VHDL. See Figure 6.2 below.

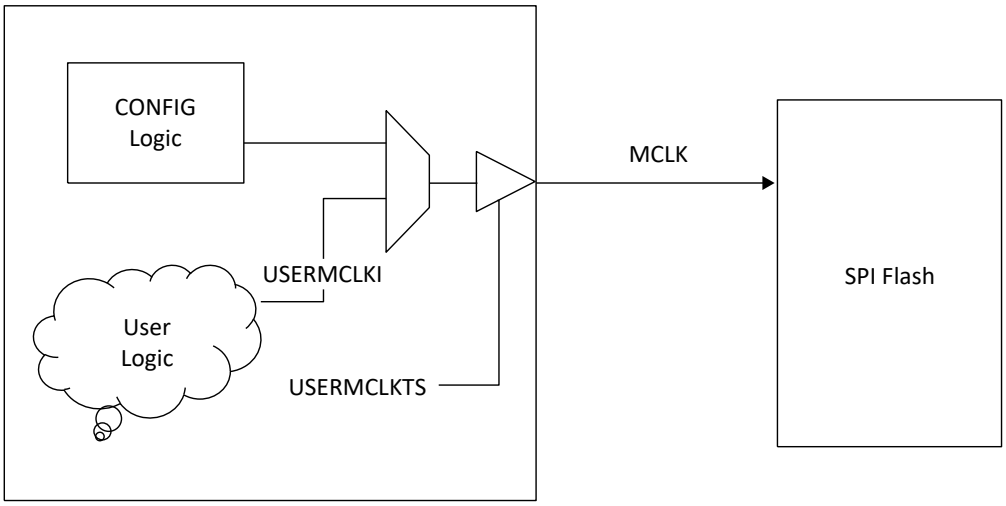


Figure 6.2. MCLK Connection

Notes:

- The USERMCLKI is selected from the MUX by instantiating the USERMCLK macro.
- USERMCLKTS is active High (this means that when it is high, it tristates the MCLK pin). USRMCLKTS must be connected to a signal (it cannot be connected to a constant value).
- As soon as the device wakes up, the USERMCLK and USERMCLKTS are activated.
- Once USERMCLK is used, the device can no longer perform background programming through SPI port.

Verilog

```
module USRMCLK (USRMCLKI, USRMCLKTS);
input USRMCLKI, USRMCLKTS;
endmodule
```

```
USRMCLK u1 (.USRMCKI(<clock_name>), .USRMCLKTS(<tristate_name> ) /* synthesis
syn_noprune=1 */;
```

VHDL

```
COMPONENT USRMCLK
   PORT(
       USRMCLKI : IN STD_ULOGIC;
       USRMCLKTS : IN STD)ULOGIC
   );
   END COMPONENT;
   attribute syn_noprune: boolean ;
   attribute syn_noprune of USRMCLK: component is true;

   begin
   u1: USRMCLK port map (
       USRMCLKI => <clock name>,
       USRMCLKTS => ,tristate_name>);
```

6.1.3. Dual-Boot and Multi-Boot Configuration Modes

Both the primary and the golden (fail-safe) configuration data is stored in external SPI memory. The fail-safe pattern is available in case the primary pattern would fail to load. The primary image can fail in one of two ways:

- A bitstream CRC error is detected
- A time-out error is encountered when loading from the primary pattern stored in the external memory

A CRC error is caused by incorrect data being written into the SRAM configuration memory. Data is read from the external Flash memory. As data enters the Configuration Engine the data is checked for CRC consistency. Before the data enters the Configuration SRAM the CRC must be correct. Any incorrect CRC causes the device to erase the Configuration SRAM and retrieve configuration data from the external golden pattern.

It is possible for the data to be correct from a CRC calculation perspective, but not be functionally correct. In this instance, the internal DONE bit is never active. The ECP5 and ECP5-5G devices count the number of master clock pulses it has provided after the Power On Reset signal was released. When the count expires without DONE becoming active the FPGA attempts to get its configuration data from the external golden pattern.

Dual boot configuration mode typically requires two configuration data files. One of the two configuration data files is a fail-safe image that is rarely, if ever, updated. The second configuration data file is a working image that is routinely updated. Both the working (primary) image and the fail-safe image are stored in the external SPI memory. One Diamond project can be used to create both the working and the fail-safe configuration data files. Configure the Diamond project with an implementation named *working*, and an implementation named *failsafe*. Read the Diamond Online Help for more information about using Diamond implementations.

The ECP5 and ECP5-5G devices support dual-boot with the MSPI mode CFGMDN[2:0] setting. If the primary pattern fails to load correctly, the ECP5 and ECP5-5G devices starts loading data from the golden sector in the SPI Flash device. In cases where the CFGMDN[2:0] setting is MSPI, a blank external Flash device causes a dual-boot event failure indicated by INITn going low. This is due to the absence of a primary or golden boot image.

The dual boot feature allows you to split a SPI Flash device into two sections, the first containing a sacred *golden boot* file, and a second updatable *primary boot* file, which can be erased and reprogrammed. By default, the FPGA loads the primary boot file in block 0 (0x000000). If the FPGA fails in configuration, it automatically loads the golden boot file in the last block (0xFFFF00). This allows your system to boot to a known operable state, so that it continues to operate if for some reason (such as a power failure) the SPI Flash fails to program correctly.

Users can dynamically switch between up to five different design revisions stored in external Flash. Multi-boot, or the ability to dynamically reconfigure from multiple design bitstreams, is similar to the dual-boot where there is one primary (working) bitstream and up to four alternate bitstreams.

For multi-boot operation, the next target address is set in memory that was loaded during the current configuration memory load. Initiating reprogramming by toggling the PROGRAMN pin or issuing a REFRESH through any sysCONFIG port causes the device to load from the defined SPI Flash address. Dual-boot can also be deployed with multi-boot allowing a golden (fail-safe) design (or sixth design) to also be available in the external Flash.

Diamond Deployment Tool allows you to assemble SPI Flash images formatted to correctly match the hardware sector mapping.

6.1.4. Dual and Quad Master SPI Read Mode

The master SPI configuration mode in the ECP5 and ECP5-5G devices are expanded to support new industry standard Quad I/O SPI Flash memory. The support of (Serial Multi I/O) Flash memory enables fast parallel read.

A typical SPI Flash interface uses either 4 or 6 interface signals to the FPGA. The Standard SPI flash uses CLK, CS, SI and SO while Quad SPI Flash uses CLK, CS, I/O0, I/O1, I/O2 and I/O3, maintaining function and pin-out compatibility with the single SPI Flash devices, while adding Dual-I/O and Quad-I/O SPI capability. All SPI modes are submodes of MASTER SPI, therefore there is no longer a separate CFGMDN pin decode for each SPI mode.

In Dual mode, the Fast-Read Dual Output (BBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on two pins, SO and SIO0, instead of just SO. This allows data to be transferred from the dual output at twice the rate of standard SPI devices. In QUAD mode, the Fast-Read Quad Output (EBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on four data pins, instead of just SO. This allows data to be transferred from the quad output at four times the rate of standard SPI devices.

To change the SPI read mode to fast read, dual read or quad read, the Deployment Tool must be used to generate the hex file used for programming the SPI flash device. Diamond flow only generate bitstream with default SPI read mode which is slow serial (03h) read mode. The CONFIG_MODE option in Diamond Global Preference is for the software to preserve appropriated pins to help customer design flow. It does not serve the purpose of enabling SPI port nor setting appropriate bits in the PROM file.

Table 6.2. Dual/Quad SPI Port Names

ECP5 and ECP5-5G Devices Pin Name	Flash Device Pin	MSPI Function DUAL(D), QUAD(Q)
MCLK	SCK	Clock output from the ECP5 and ECP5-5G devices Configuration Logic and Master SPI controller. Connect MCLK to the SCLK input of the Slave SPI device. (D)(Q)
IO[3:2]	SIO[3:2]	Data I/O between the ECP5 and ECP5-5G devices and SPI device. (Q)
IO[1:0]	SIO[1:0]	Data I/O between the ECP5 and ECP5-5G devices and SPI device. (D)(Q)
CSSPIN	CSN	Chip select output from the ECP5 and ECP5-5G devices configuration logic to the slave SPI Flash holding configuration data for the ECP5 and ECP5-5G devices.

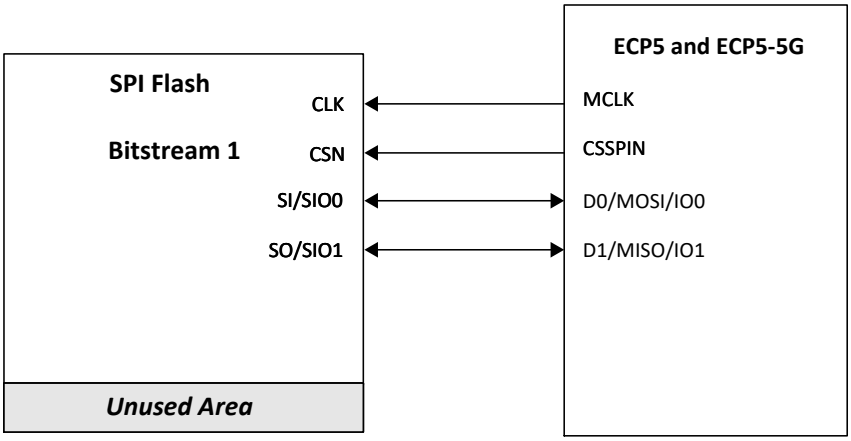


Figure 6.3. One Dual SPI Flash Interface (Dual)

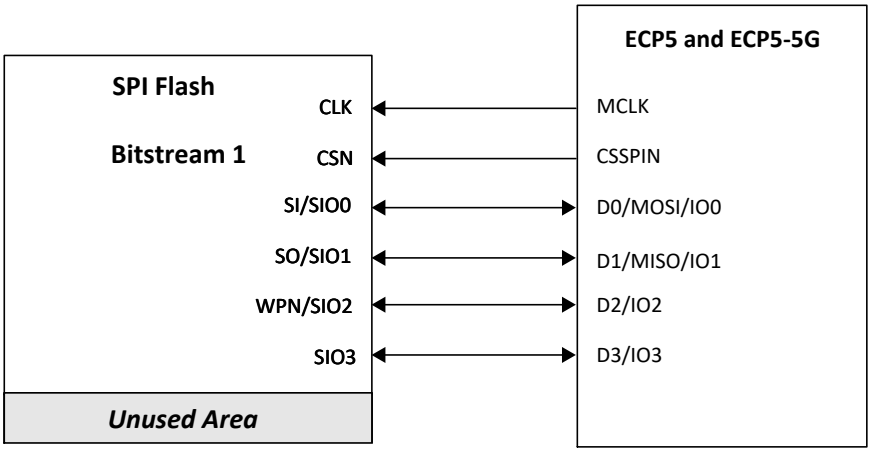


Figure 6.4. One Quad SPI Flash Interface (Quad)

6.2. Slave SPI Mode

The ECP5 and ECP5-5G devices provide a Slave SPI (SSPI) configuration port that allows you to access features provided by the Configuration Logic. You can reprogram the Configuration SRAM and access status/control registers within the Configuration Logic block. The external Flash memory updates are done using either offline or transparent operations. It is necessary to send a REFRESH command to load a new external Flash image into the configuration SRAM. When the ECP5 and ECP5-5G devices are in Background Mode, only read type commands are supported, allowing for device verification or debugging. The command set consists of 8-bit opcodes. Some of the commands have a 24-bit operand following the 8-bit opcode. The Slave SPI port is a byte bounded port; all input and output data must be byte bounded.

In the Slave SPI mode, the MCLK/CCLK pin becomes CCLK (that is Configuration clock). Input data is read into the ECP5 and ECP5-5G devices on the MOSI pin at the rising edge of CCLK. Output data is valid on the MISO pin at the falling edge of CCLK. The SN acts as the chip select signal. When SN is high, the SSPI interface is deselected and the MISO pin is tristated. Commands can be written into and data read from the ECP5 and ECP5-5G devices when SN is asserted.

The SSPI port is active when the CFGMDN[2:0] is set to [001]. Diamond's default preference for the SLAVE_SPI_PORT is to DISABLE the port. Use the Spreadsheet View to ENABLE the SLAVE_SPI_PORT preference in your design to keep the SSPI port active for configuration after the device enters user mode.

Using the SSPI port simplifies the ECP5 and ECP5-5G devices configuration process. Lattice provides *C*source code called SSPIEmbedded to insulate you from the complexity of programming the ECP5 and ECP5-5G devices. Refer to Diamond online help about SSPIEmbedded.

Table 6.3. Slave SPI Configuration Port Pins

Pin name	Function	Direction	Description
MCLK/CCLK	CCLK	Input with weak pull-up	Clock used to time data transmission/reception from an external SPI master device to the ECP5 and ECP5-5G devices Configuration Logic.
MOSI	MOSI	Input	Carries output data from the external SPI master to the ECP5 and ECP5-5G devices Configuration Logic
MISO	MISO	Output	Carries output data from the ECP5 and ECP5-5G devices Configuration Logic to the external SPI master. It is normally tristated with an internal pull-up, It becomes active only when the command is a read type command.
SN[1]	SN	Input with weak pull-up	ECP5 and ECP5-5G devices Configuration Logic slave SPI chip select input. SN is an active low input. High to Low transition: reset the device, prepare it to receive commands. Low to High transition: Completes or terminates the current command.
HOLDN[2]	HOLDN	Input	Although not a standard SPI pin, this is an industry standard pin that allows the CPU to suspend data transmission. This pin can be asserted while shifting the bitstream into the device. Do not assert the HOLDN pin while shifting commands or operand into the device.

Notes:

1. Use external 4.7 kΩ pull-up resistor.
2. By definition, the Slave SPI port is a four (4) wire port. The HOLDN pin was added by SPI Flash vendors to provide the CPU a method to support suspension. The ECP5 and ECP5-5G devices also support the HOLDN pin to provide the CPU with a method to suspend data transmission when it cannot process a large bitstream in one single burst and needs time to fetch the bitstream in fragments.

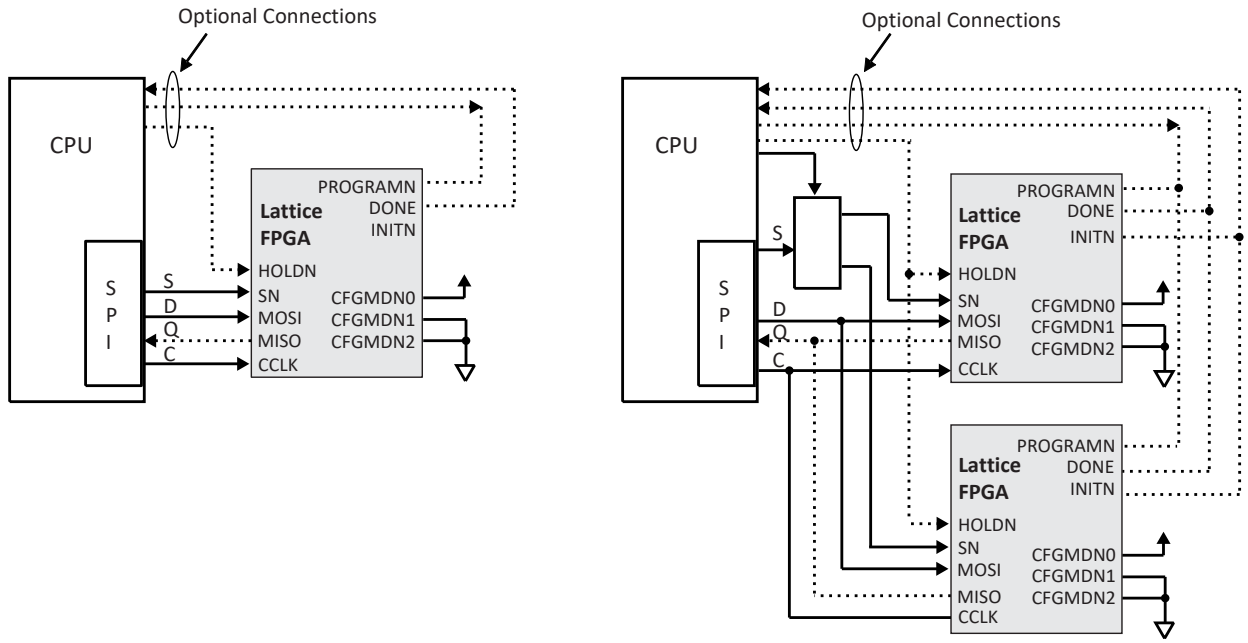


Figure 6.5. ECP5 and ECP5-5G Slave SPI Port with CPU and Single or Multiple Devices

Notes:

- The dotted lines indicate optional connections.
- The wake up time of the device does vary with the bitstream size and the speed of the SPI port. Lattice recommends connecting the DONE pin to the CPU to monitor when the configuration is complete.
- If the bitstream for the two ECP5 and ECP5-5G devices are the same, the chip select logic (S/SN) is not required.
- The MISO to Q connection is optional if read back is not needed and the DONE pin is connected.

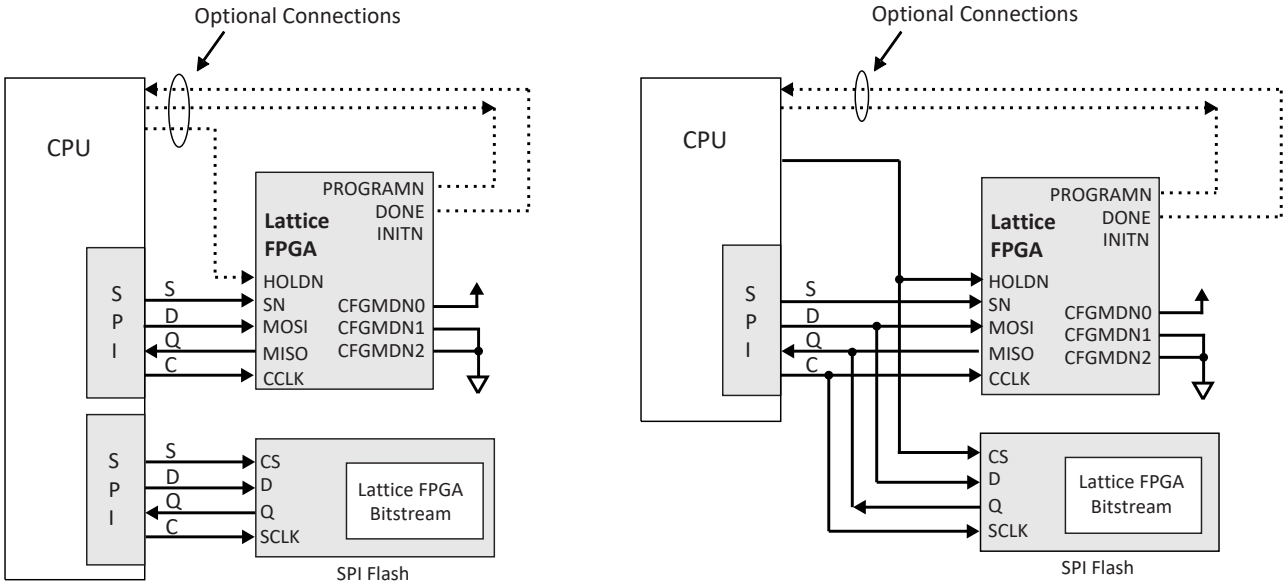


Figure 6.6. ECP5 and ECP5-5G Slave SPI Port with SPI Flash

Notes:

- The dotted lines indicate the connection is optional.
- The ECP5 and ECP5-5G devices bitstream can reside in the SPI Flash device instead of the system Flash memory. The advantage of this is that the bitstream can be easily updated without changing the system software.

6.2.1. Method to Enable the Slave SPI Port

Similar to all configuration ports, the Slave SPI port is enabled by the two standard methods:

- Setting the CFGMDN[2:0] pin to [0, 0, 1].

  When the device is powered up, or when the PROGRAMN pin is toggled, the device checks the state of the CFGMDN pins. If they are set to [0,0,1], then the device chooses the Slave SPI port as the configuration port. This is the only method to enable the Slave SPI port as the configuration port. The CONFIG_MODE option in Diamond Global Preference is for our software to preserve appropriated pins to help customer design flow, it does not serve the purpose to enable SPI port. A port is said to be a configuration port when it is capable of executing both bitstream write and read commands.
- Enabling Slave SPI port persistence.

  The configuration bitstream contains optional Slave SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the Slave SPI port is to remain operational once in user mode. This selection is independent of the CFGMDN pins setting. A port enabled by persistence is capable of read commands only. Thus, the port is a Background Mode port and is not a configuration port, and can only be used for read back operations.

  Note that both the DONE pin and the INITN pin must be high to qualify the Slave SPI port as a read back port. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

6.2.2. Specifications and Timing Diagrams – Slave SPI Port Waveforms

Data and commands shift into the MOSI pin on the rising edge of clock. Data is shifted out of the MISO pin on the falling edge of clock. Only a read command causes the MISO pin to be enabled for data read out.

The Slave SPI read and write waveforms are shown in Figure 6.7 and Figure 6.8. The Slave SPI HOLDN pin waveform is shown in Figure 6.9.

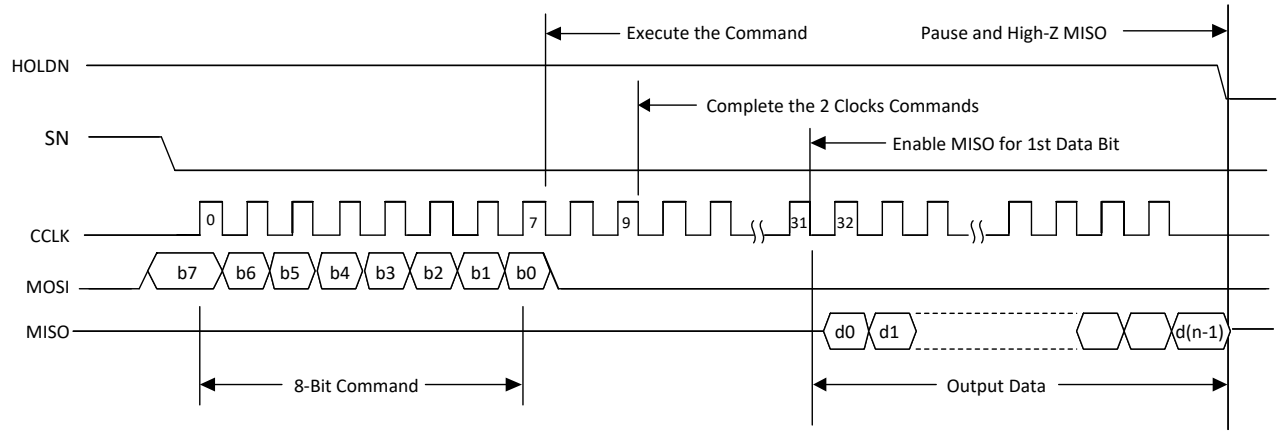
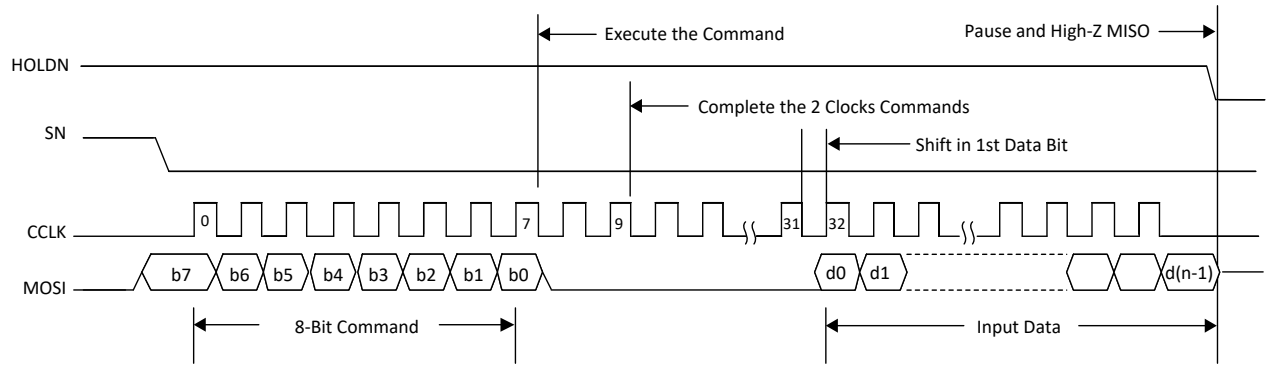


Figure 6.7. Slave SPI Read Waveforms



Note: The bitstream is transferred starting with the first byte of the data file, starting with the MSB of the byte.

Figure 6.8. Slave SPI Write Waveforms

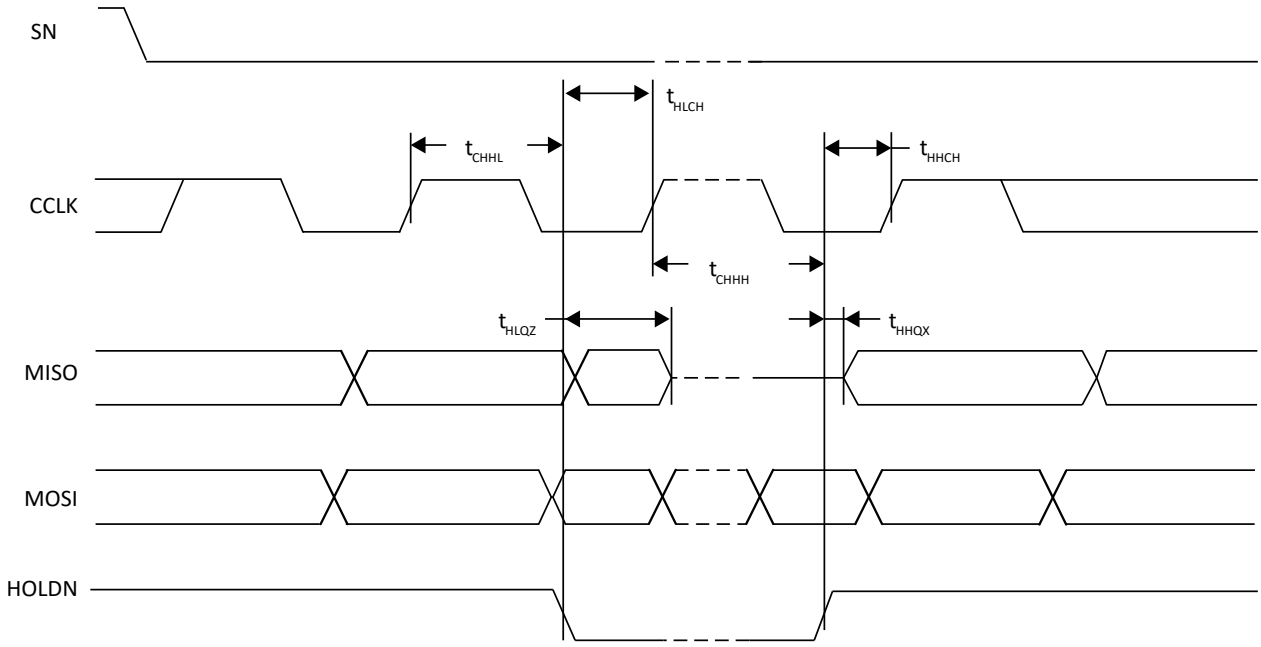


Figure 6.9. Slave SPI HOLDN Waveforms

6.2.3. Slave SPI List of Command

Table 6.4. Slave SPI Command Table

Command	Binary	Hex	Operand	Description
ISC_NOOP	11111111	FF	0 bits	Non-operation
READ_ID	11100000	E0	24 bits	Read out the 32-bit IDCODE of the device
USERCODE	11000000	C0	24 bits	Read 32-bit user code
LSC_READ_STATUS	00111100	3C	24 bits	Read out internal status
LSC_CHECK_BUSY	11110000	F0	24 bits	Read 1 bit busy flag to check the command execution status
LSC_REFRESH	01111001	79	24 bits	Equivalent to toggle PROGRAMN pin
ISC_ENABLE	11000110	C6	24 bits	Enable the Offline configuration mode
ISC_ENABLE_X	01110100	74	24 bits	Enable the Transparent configuration mode
ISC_DISABLE	00100110	26	24 bits	Disable the configuration operation
ISC_PROGRAM_USERCOD E	11000010	C2	24 bits	Write the 32-bit new USERCODE data to USERCODE register

Command	Binary	Hex	Operand	Description
ISC_ERASE	00001110	0E	24 bits	Bulk erase the memory array base on the access mode and array selection
ISC_PROGRAM_DONE	01011110	5E	24 bits	Program the DONE bit if the device is in Configuration state.
ISC_PROGRAM_SECURITY	11001110	CE	24 bits	Program the Security bit if the device is in Configuration state
LSC_INIT_ADDRESS	01000110	46	24 bits	Initialize the Address Shift Register
LSC_WRITE_ADDRESS	10110100	B4	24 bits	Write the 16 bit Address Register to move the address quickly
LSC_BITSTREAM_BURST	01111010	7A	24 bits	Program the device the whole bitstream sent in as the command operand
LSC_PROG_INCR_RTI	10000010	82	24 bits	Write configuration data to the configuration memory frame at current address and post increment the address, Byte 2~0 of the opcode indicate number of the frames included in the operand field
LSC_PROG_INCR_ENC	10110110	B6	24 bits	Encrypt the configuration data then write
LSC_PROG_INCR_CMP	10111000	B8	24 bits	Decompress the configuration data, then write
LSC_PROG_INCR_CNE	10111010	BA	24 bits	Decompress and Encrypt the configuration data, then write
LSC_VERIFY_INCR_RTI	01101010	6A	24 bits	Read back the configuration memory frame selected by the address register and post increment the address
LSC_PROG_CTRL0	00100010	22	24 bits	Modify the Control Register 0
LSC_READ_CTRL0	00100000	20	24 bits	Read the Control Register 0
LSC_RESET_CRC	00111011	3B	24 bits	Reset 16-bit frame CRC register to 0x0000
LSC_READ_CRC	01100000	60	24 bits	Read 16-bit frame CRC register content
LSC_PROG_SED_CRC	10100010	A2	24 bits	Program the calculated 32-bit CRC based on configuration bit values only into overall CRC register
LSC_READ_SED_CRC	10100100	A4	24 bits	Read the 32-bit SED CRC
LSC_PROG_PASSWORD	11110001	F1	24 bits	Program 64-bit password into the non-volatile memory (Efuse)
LSC_READ_PASSWORD	11110010	F2	24 bits	Read out the 64-bit password before activated for verification
LSC_SHIFT_PASSWORD	10111100	BC	24 bits	Shift in the password to unlock for re-configuration (necessary when password protection feature is active).
LSC_PROG_CIPHER_KEY	11110011	F3	24 bits	Program the 128-bit cipher key into Efuse
LSC_READ_CIPHER_KEY	11110100	F4	24 bits	Read out the 128-bit cipher key before activated for verification
LSC_PROG_FEATURE	11100100	E4	24 bits	Program User Feature, such as Customer ID, I2C Slave Address, Unique ID Header
LSC_READ_FEATURE	11100111	E7	24 bits	Read User Feature, such as Customer ID, I2C Slave Address, Unique ID Header
LSC_PROG_FEABITS	11111000	F8	24 bits	Program User Feature Bits, such as CFG port and pin persistence, PWD_EN, PWD_ALL, DEC_ONLY, Feature Row Lock etc.
LSC_READ_FEABITS	11111011	FB	24 bits	Read User Feature Bits, such as CFH port and pin persistence, PWD_EN, PWD_ALL, DEC_ONLY, Feature Row Lock etc.
LSC_PROG_OTP	11111001	F9	24 bits	Program OTP bits, to set Memory Sectors One Time Programmable
LSC_READ_OTP	11111010	FA	24 bits	Read OTP bits setting

Table 6.5. Slave SPI Command Usage Table

Command	OPCODE	CLASS	Flow Operation Number	Delay Time
ISC_NOOP	FF	C	—	None
READ_ID	E0	A	2, 3	None
USERCODE	C0	A	—	None
LSC_READ_STATUS	3C	A	7	None
LSC_CHECK_BUSY	F0	A	—	None
LSC_REFRESH	79	D	—	[3]
ISC_ENABLE	C6	C	4	None
ISC_ENABLE_X	74	C	—	None
ISC_DISABLE	26	C	8	None
ISC_PROGRAM_USERCODE	C2	B	—	None
ISC_ERASE	0E	D	—	[2]
ISC_PROGRAM_DONE	5E	D	—	[1]
ISC_PROGRAM_SECURITY	CE	C	—	None
LSC_INIT_ADDRESS	46	C	—	None
LSC_WRITE_ADDRESS	B4	B	—	None
LSC_BITSTREAM_BURST	7A	C	5, 6	None
LSC_PROG_INCR_RTI	82	B	—	[1]
LSC_PROG_INCR_ENC	B6	B	—	[1]
LSC_PROG_INCR_CMP	B8	B	—	[1]
LSC_PROG_INCR_CNE	BA	B	—	[1]
LSC_VERIFY_INCR_RTI	6A	A	—	None
LSC_PROG_CTRL0	22	B	—	None
LSC_READ_CTRL0	20	A	—	None
LSC_RESET_CRC	3B	C	—	None
LSC_READ_CRC	60	A	—	None
LSC_PROG_SED_CRC	A2	B	—	[1]
LSC_READ_SED_CRC	A4	A	—	None
LSC_PROG_PASSWORD	F1	B	—	[1]
LSC_READ_PASSWORD	F2	A	—	None
LSC_SHIFT_PASSWORD	BC	B	—	None
LSC_PROG_CIPHER_KEY	F3	B	—	[1]
LSC_READ_CIPHER_KEY	F4	A	—	None
LSC_PROG_FEATURE	E4	B	—	[1]
LSC_READ_FEATURE	E7	A	—	None
LSC_PROG_FEABIT	F8	B	—	[1]
LSC_READ_FEABIT	FB	A	—	None
LSC_PROG_OTP	F9	B	—	[1]
LSC_READ_OTP	FA	A	—	None

Notes:

1. Delay time depends on data frame size and clock frequency. Duration could be in seconds. Use LSC_CHECK_BUSY to check the status.
2. Delay time depends on the flash memory sector size and clock frequency. Duration could be in seconds. Please use LSC_CHECK_BUSY to check the status.

3. Delay time depends predominantly on the bitstream size and clock frequency. Duration could be in seconds. Please use LSC_CHECK_BUSY to check the status.

6.2.4. Slave SPI Configuration Flow Diagrams

The Slave SPI port supports the regular ECP5 and ECP5-5G devices bitstream and the encrypted bitstream (SSPI Mode).

The regular bitstream format is the same for all configuration modes. However, the encrypted bitstream format is configuration mode dependent. When generating an encrypted bitstream, you must select the Slave SPI configuration mode (SSPI mode) for use with the Slave SPI port.

The Slave SPI configuration flow diagram is shown in Figure 6.10. Highlights are listed below.

- The bitstream file is a stand-alone file. It is not part of the driver or system code. This provides seamless system integration and flexible file management. For example, the bitstream can be switched on the fly without changing a single line of system code.
- The ECP5 and ECP5-5G devices wakes up and enter user mode once it reads in the entire bitstream. If it is necessary to delay the wake-up, the simplest method is by using the DONE pin. Wake-up can be delayed by holding the open-drain DONE pin low until the wake-up is desired.

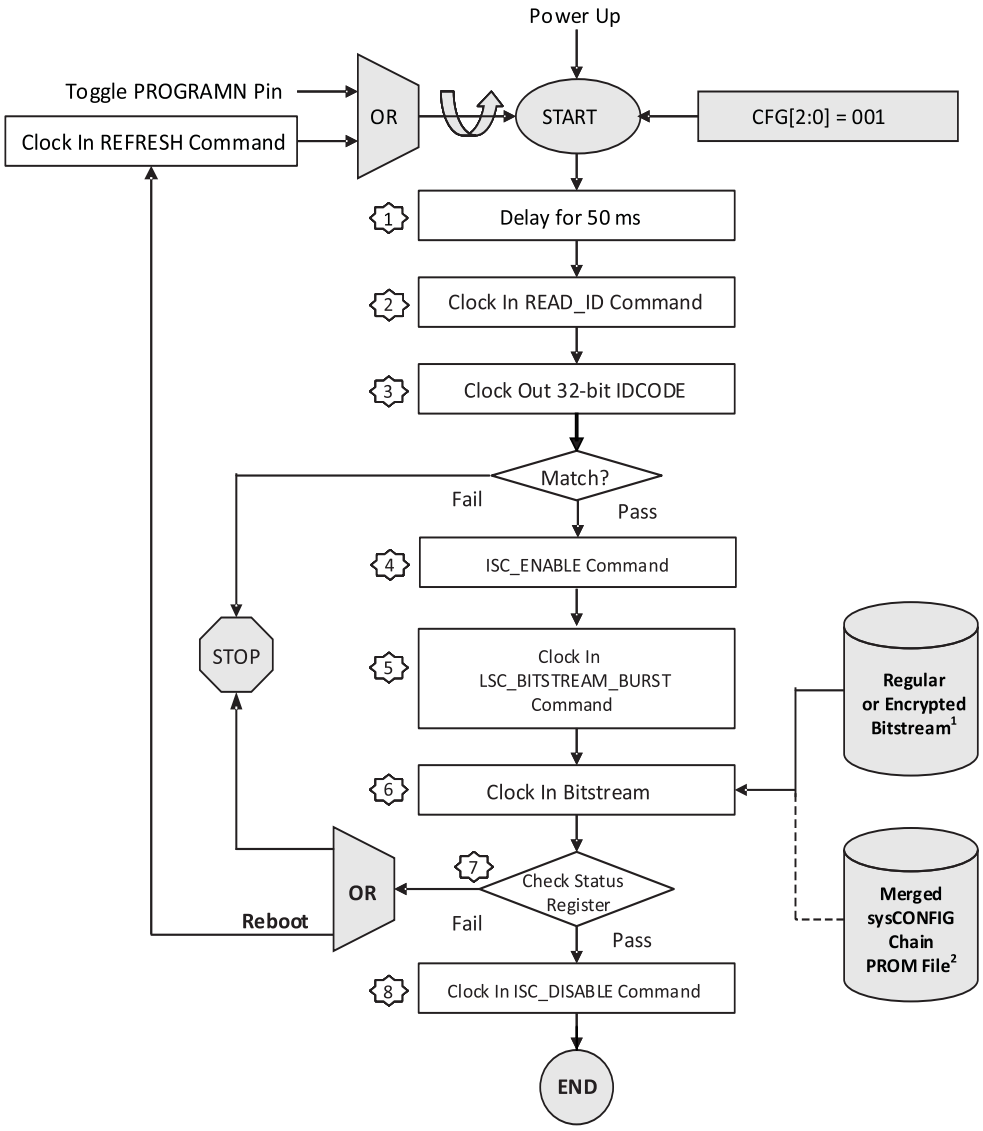


Figure 6.10. Slave SPI Configuration Flow

Notes:

1. For a single ECP5 or ECP5-5G device, the input file is a bitstream, which may be a standard or encrypted bitstream.
2. For a sysCONFIG chain of devices, the input file can be a merged PROM file. See the Appendix E. Advanced Applications – Slave SPI sysCONFIG Daisy Chaining for more details.

6.2.5. Command Waveforms

Class A Command Waveforms

The Class A commands are ones that read data out from the ECP5 and ECP5-5G devices. Bit 0 of the data or bitstream is read out first. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command.

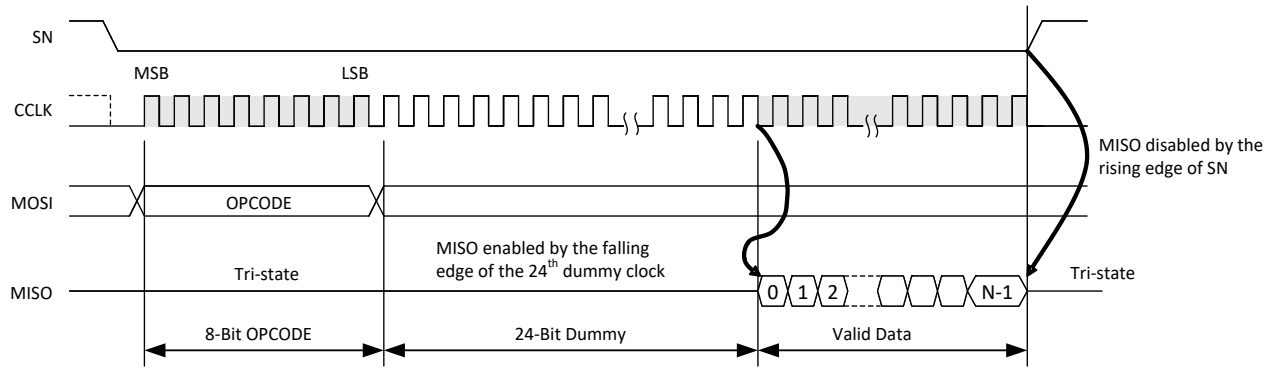


Figure 6.11. Class A Command Waveforms

Class B Command Waveforms

The Class B commands are used to shift data into the port. Bit 0 of the data or bitstream is shifted in first. The twenty four (24) dummy clocks provide the device the necessary delay time to execute the command properly.

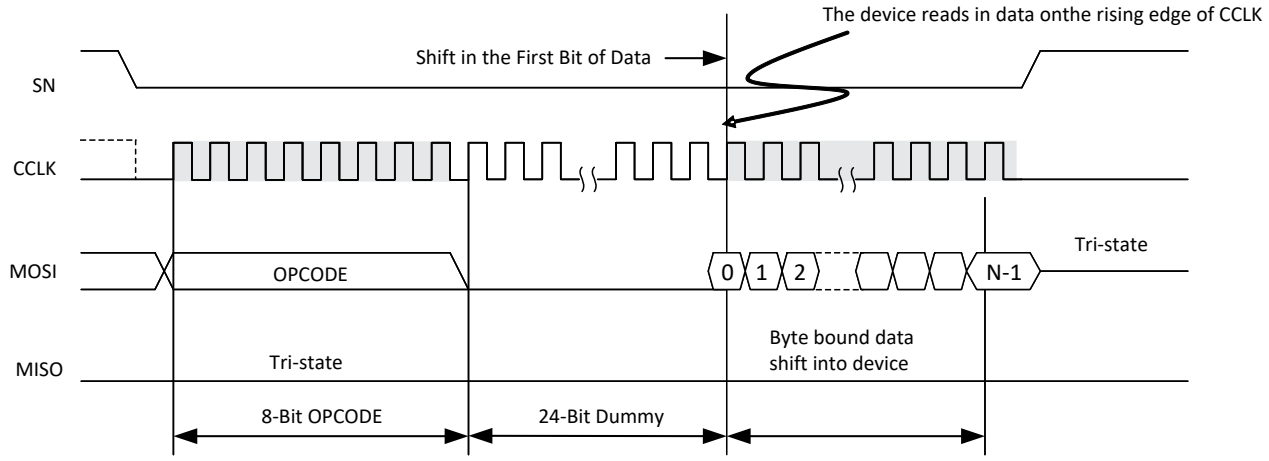


Figure 6.12. Class B Command Waveforms

Class C Command Waveforms

The Class C commands do not require any data to be shifted in or out. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command. Even if extra dummy clocks are presented, the device ignores them.

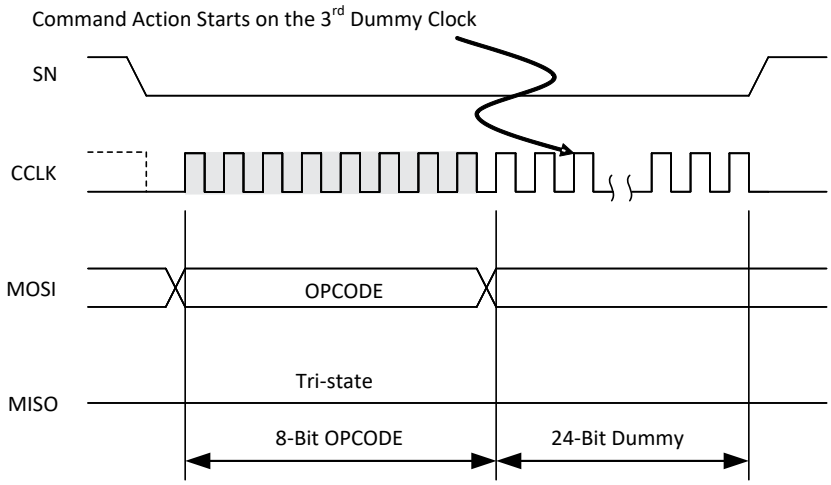


Figure 6.13. Class C Command Waveforms

Class D Commands Waveforms

The Class D commands do not need to shift data in or out but still require a delay to execute the action associated with the command. This type of command cannot terminate the action of any commands including itself. After the 24th dummy clock, continuing to clock or suspending the clock or driving the SN pin high does not terminate the action. The action ends when it is complete. This class of commands is defined particularly for the benefit of the two unique commands: CLEAR and REFRESH.

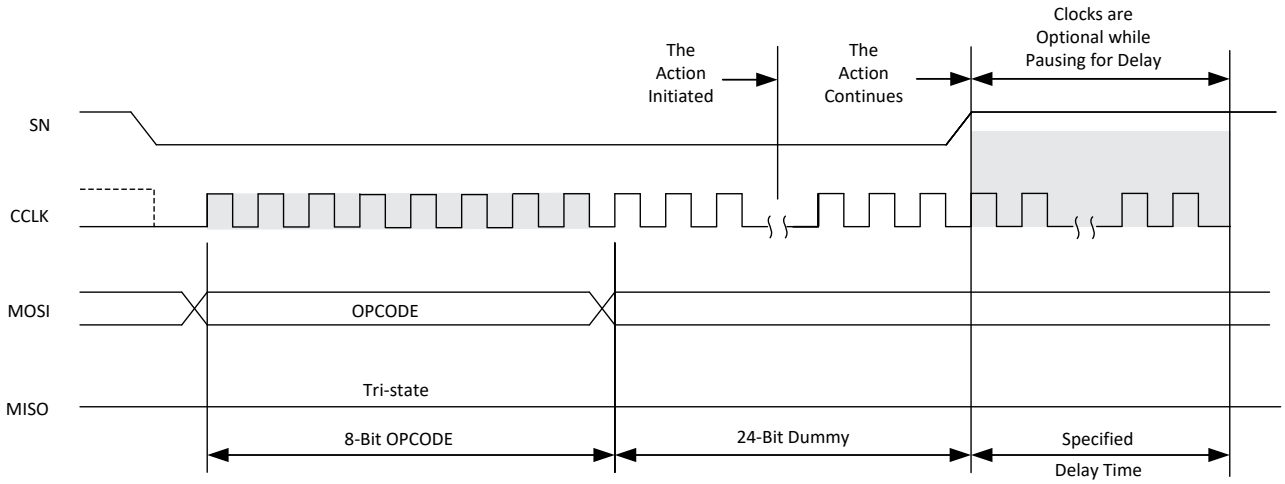


Figure 6.14. Class D Command Waveforms

6.3. Slave Parallel Mode (SPCM)

The Slave Parallel interface supports 8-bit wide buses to configure or read back the SRAM. In Slave Parallel mode, a host system sends the configuration data in a byte-wide stream to the ECP5 and ECP5-5G devices. The CCLK, CSN, CS1N, and WRITEN pins are driven by the host system. WRITEN, CSN, and CS1N must be held low to write to the device; data is input from D[7:0]. D7 lines up with the very first bit (MSB of the first byte) in the bitstream. No internal pull-up or pull-down resistors are on the Data[7:0] pins during configuration, therefore the PCB design should include them.

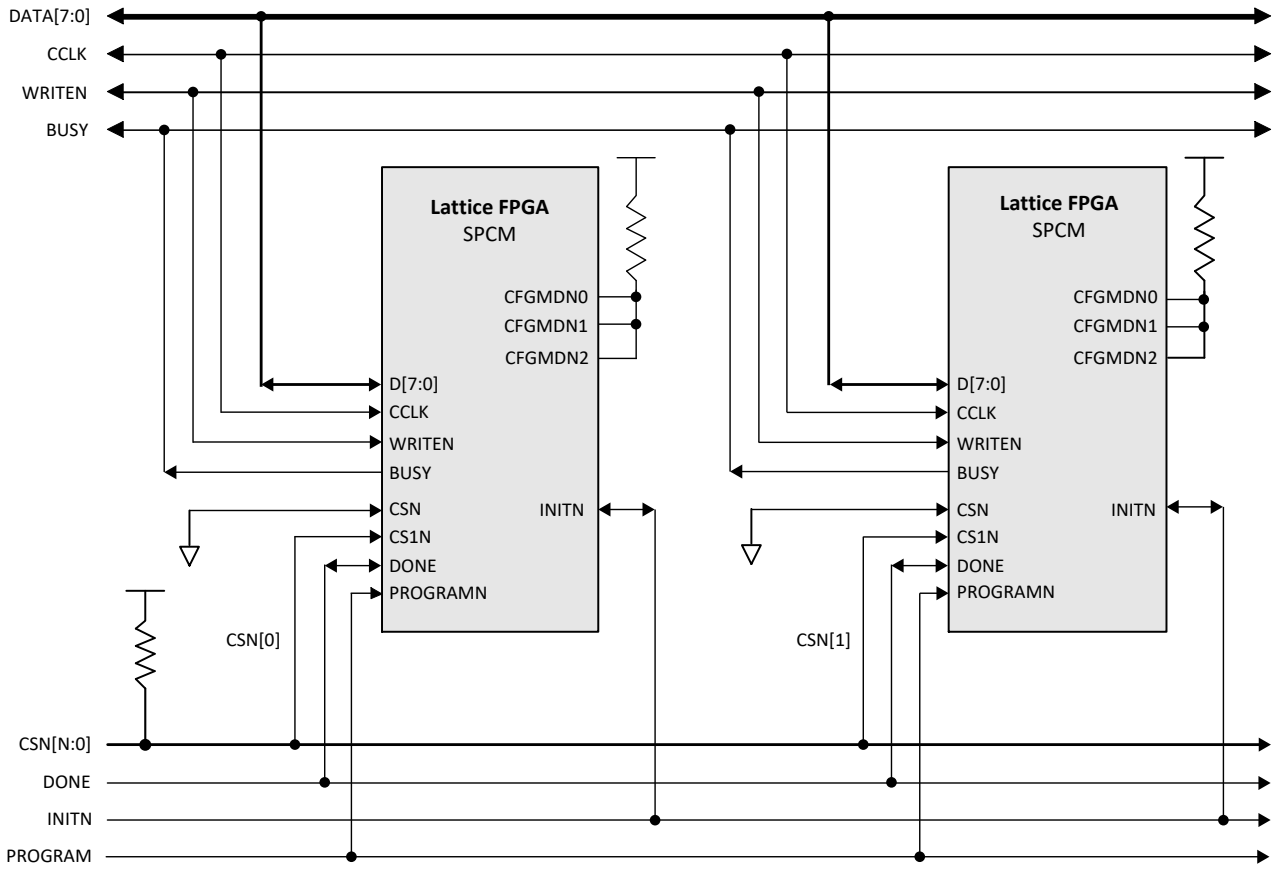


Figure 6.15. Slave Parallel Interface

In slave-parallel mode, multiple devices can be chained in parallel. The D[7:0], CCLK, WRITEN, BUSY, DONE, PROGRAMN, and INITN may be connected in parallel between devices. CSN is connected separately to allow individual devices to be selected. For example to select the first device in the chain, CSN[0] is set low while CSN[1] to CSN[n] is set high. After configuring, the first device, CSN[0] is set to high and CSN[1] is set to low to select the second device. CSN acts as a clock enable signal and CS1N starts and executes individual commands.

Figure 6.16 shows a slave-parallel write sequence. To send configuration data to a device, the WRITEN signal has to be asserted. During the write cycle, the BUSY signal provides handshaking between the host system and the device. When BUSY signal is low, the device is ready to read a byte of data at the next rising edge of CCLK. The BUSY signal is set high when the device reads the data and the device requires extra clock cycles to process the data. The CSN signal is used to temporary stop the write process by setting it to high, if the host system is busy. The device resumes the configuration when the CSN signal is set to low again. After the last byte of configuration data is sent, the WRITEN signal is set to high.

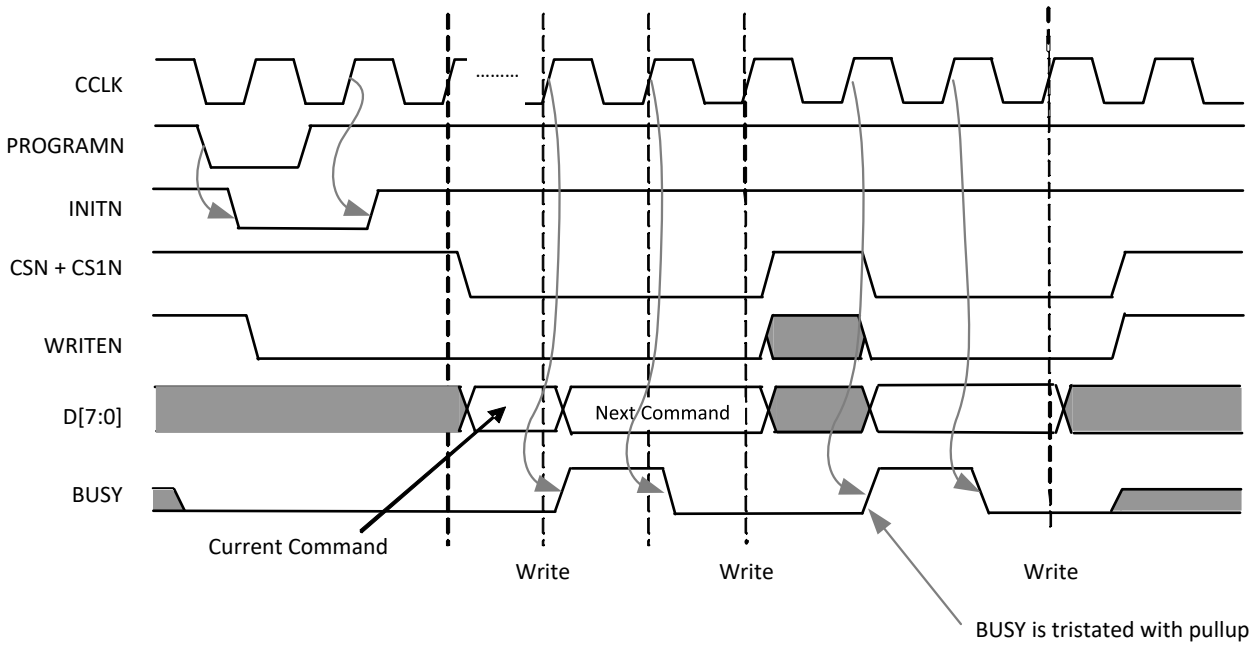


Figure 6.16. Slave Parallel Write

Slave Parallel mode can also be used for readback of the internal configuration. By driving the WRITEN pin low, and CSN and CS1N low, the device inputs the readback instructions on the D[7:0] pins; WRITEN is then driven high and read data is output on D[0:7] In order to support readback, the SLAVE PARALLEL PORT in the Diamond Spreadsheet View must be set to ENABLE.

To read back the configuration data or register contents, WRITEN is first set low to send the read instruction into the device. The device reads in the command from the CPU and execute the command. If the device cannot have the data ready by the next clock cycle, it pulls the busy signal up. When BUSY is high, the device continues to execute the command regardless of the CSN pin. The device pulls the BUSY pin low when the data is ready but does not drive the D[7:0] until the CSN pin is pulled low by the CPU. The WRITEN pin is pulled high after sending in the command. Both the CSN and WRITEN signals are latched in and switches the read-write mode on the rising edge of CCLK. If the device needs more than one clock cycle to switch the bus around, BUSY is kept high until the D[7:0] is ready for read by CPU. As in the Write sequence, CSN signal is also used to temporary pulse read sequence in case the host system is busy. The data is read at the next rising CCLK edge, after CSN is set to low and BUSY is low. Slave-parallel reading is not available once configuration begins through the SPCM port. Reading is only available once the configuration is completed.

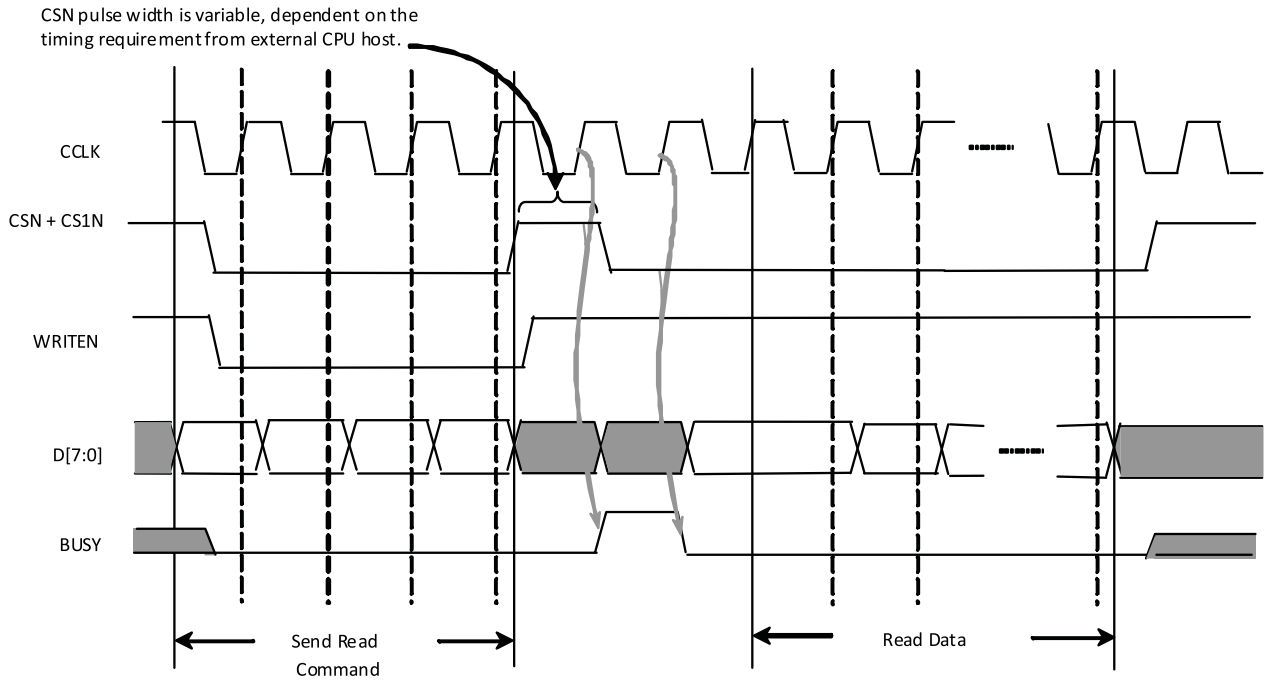


Figure 6.17. Slave Parallel Read

6.4. Slave Serial Configuration Mode

Slave Serial Configuration Mode (SCM) provides a simple, low pin count method for configuring one or more FPGAs. Data is presented to the FPGA on the Data Input pin DI at every CCLK rising edge.

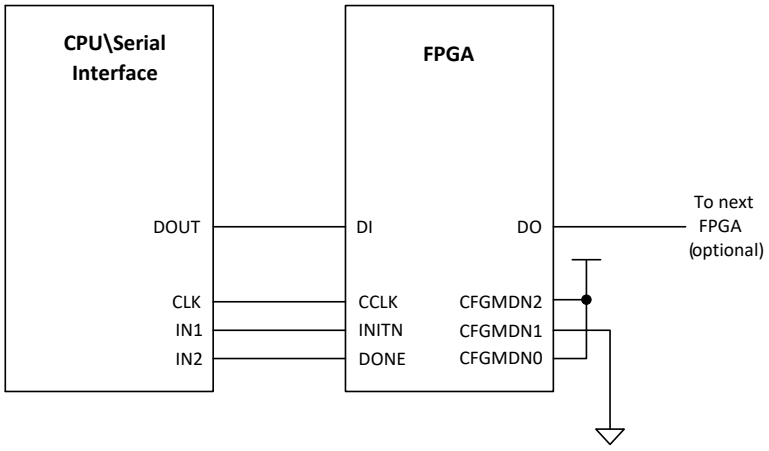


Figure 6.18. Slave Serial Block Diagram

The bitstream data generated by Lattice Diamond is formatted so that it is ready to shift into the FPGA. Left shift the data out of the file in order for it to be correctly received by the FPGA.

The FPGA synchronizes itself on either a 0xBDB3 or 0xBAB3 code word. It is critical that any data presented on DIN not be recognized as one of these two synchronization words early. To guarantee proper recognition of the synchronization word it is recommended that the synchronization word always be preceded by a minimum of 128 '1' bits. Presenting any other bitstream data, Programmer generated header information for example, risks the being misinterpreted due to bit slippage.

Slave Serial Configuration Mode can be used to configure a chain of FPGAs. Details about configuring a chain of devices is discussed in Daisy Chaining section.

6.5. JTAG Mode

The JTAG port provides:

- Offline external Flash memory programming
- Background external Flash memory programming
- Direct SRAM configuration
- Full access to the ECP5 and ECP5-5G devices Configuration Logic
- Device chaining
- IEEE 1149.1 testability
- IEEE 1532 Compliant programming

As a dedicated port on the ECP5 and ECP5-5G devices, the JTAG port is always available. The JTAG port pins are dedicated to performing the IEEE 1149.1 TAP function.

The advantages of keeping the JTAG port include:

- Multi-chain Architectures – The JTAG port is the only configuration and programming port that permits the ECP5 and ECP5-5G devices to be combined in a chain of other programmable logic.
- Reveal Debug – The Lattice Reveal debug tool is an embed-able logic analyzer tool. It allows you to analyze the logic inside the ECP5 and ECP5-5G devices in the same fashion as an external logic analyzer permits analysis of board level logic. Reveal access is only available through the JTAG port.
- SRAM Readback – The JTAG port is able to directly access the configuration SRAM. It is occasionally necessary to perform failure analysis for SRAM based FPGAs. A key component to failure analysis can involve reading the configuration SRAM.
- Boundary Scan Testability – Board level connectivity testing performed using IEEE 1149.1 JTAG is a key capability for assuring the quality of assembled printed-circuit-boards. Lattice provides Boundary Scan Description Language files for the ECP5 and ECP5-5G devices on the Lattice website.

6.6. TransFR Operation

The ECP5 and ECP5-5G, like other Lattice FPGAs, provides for the TransFR™ capability. TransFR is described in Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198).

7. Software Selectable Options

The operation of the ECP5 and ECP5-5G devices configuration logic is managed by options selected in the Diamond design software. The ECP5 and ECP5-5G devices use dedicated I/O pins to select the configuration mode. You use the Diamond Spreadsheet View to make the changes to the operation of the ECP5 and ECP5-5G devices programming which alters the operation of the configuration logic.

The configuration logic preferences are accessed using Spreadsheet View. Click on the Global Preferences tab, and look for the sysCONFIG tree. The sysCONFIG section is shown in Figure 7.1.

Preference Name	Preference Value
Junction Temperature (Tj)(C)	125.000
Voltage (V)	1.045
SYSTEM_JITTER(ns)	Default
Block Path	
Block Asynchpaths	ON
Block Resetpaths	ON
Block RD During WR Paths	OFF
Block InterClock Domain Paths	OFF
Block Jitter	OFF
sysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_PARALLEL_PORT	DISABLE
DONE_EX	OFF
DONE_OD	ON
DONE_PULL	ON
MCCLK_FREQ	2.4
TRANSFR	OFF
CONFIG_IOVOLTAGE	2.5
CONFIG_SECURE	OFF
WAKE_UP	21
COMPRESS_CONFIG	OFF
CONFIG_MODE	JTAG
User Code	
UserCode Format	Binary
UserCode	00000000000000000000000000000000
Unique ID	
UniqueID	0000
Global Set/Reset Net	
Bank VCCIO	
Bank0 (V)	Auto
Bank1 (V)	Auto
Bank2 (V)	Auto
Bank3 (V)	Auto
Bank6 (V)	Auto
Bank7 (V)	Auto
Bank8 (V)	Auto

Figure 7.1. sysCONFIG Preferences in Global Preferences Tab, Diamond Spreadsheet View

Table 7.1. sysCONFIG Options

Option Name	Default Setting	All Settings
SLAVE_SPI_PORT	DISABLE	DISABLE, ENABLE
MASTER_SPI_PORT	DISABLE	DISABLE, ENABLE
SLAVE_PARALLEL_PORT	DISABLE	DISABLE, ENABLE
CONFIG_MODE*	JTAG	JTAG, SSPI, SPI_SERIAL, SPI_DUAL, SPI_QUAD, SLAVE_PARALLEL, SLAVE_SERIAL
DONE_EX	OFF	OFF, ON
DONE_OD	ON	OFF, ON
DONE_PULL	ON	OFF, ON
MCCLK_FREQ	2.4	See MCCLK Frequency section on the next page.
TRANSFR	OFF	OFF, ON
CONFIG_IOVOLTAGE	2.5	1.2, 1.5, 1.8, 2.5, 3.3
CONFIG_SECURE	OFF	OFF, ON
WAKE_UP	21 [DONE_EX = Off]	[4, 21]
	4 [DONE_EX = On]	
COMPRESS_CONFIG	OFF	OFF ON
USERCODE Format	Binary	Binary, Hex, ASCII, Auto
USERCODE	0 (32 binary zeros)	32-bit
UNIQUE_ID	0	Upper 16-bit user-defined code for above Auto USERCODE.

*__Note__: The CONFIG_MODE option in Diamond Global Preference is for the software to preserve appropriated pins to help customer design flow. It does not serve the purpose of enabling SPI port nor setting appropriate bits in the PROM file.

Slave SPI Port

The SLAVE_SPI_PORT allows you to preserve the Slave SPI configuration port after the ECP5 and ECP5-5G devices enter user mode. There are two states to which the SLAVE_SPI_PORT preference can be set:

- ENABLE – This setting preserves the SPI port I/O when the ECP5 and ECP5-5G devices are in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to the port pins.
- DISABLE – This setting disconnects the SPI port pins from the configuration logic. By itself it does not make the port pins general purpose I/O. Both the SLAVE_SPI_PORT and MASTER_SPI_PORT must be in the DISABLE state for the SPI port pins to be general purpose I/O.

The SLAVE_SPI_PORT cannot be enabled at the same time as the MASTER_SPI_PORT. It is necessary to guarantee that the internal Master SPI controller, if the customer has created one, does not perform SPI transactions at the same time as an external SPI master. It is your responsibility to prevent two SPI masters from operating simultaneously.

Master SPI Port

The MASTER_SPI_PORT allows you to preserve the SPI configuration port after the ECP5 and ECP5-5G devices enter user mode. There are two states to which the MASTER_SPI_PORT preference can be set:

- ENABLE – This setting preserves the SPI port I/O when the ECP5 and ECP5-5G devices are in user mode. The preference also prevents you from over-assigning I/O to the port pins.
- DISABLE – This setting disconnects the SPI port pins from the configuration logic. By itself it does not make the port pins general purpose I/O. Both the SLAVE_SPI_PORT and MASTER_SPI_PORT must be in the DISABLE state for the SPI port pins to be general purpose I/O. Select this setting if you prefer to generate your own Master SPI Controller in user mode.

7.1.1. Slave Parallel Port

The SLAVE_PARALLEL_PORT allows you to preserve the SPCM configuration port after the ECP5 and ECP5-5G devices enter user mode. There are two states to which SLAVE_PARALLEL_PORT preference can be set:

- ENABLE – This setting preserve the SPCM port when the ECP5 and ECP5-5G devices are in user mode. It allows you to read all of the configuration data, except the EBR and the distributed RAM contents in user mode.
- DISABLE – This setting disconnects the SPCM port pins from the configuration logic. It allows SPCM ports pins to be general purpose I/O.

7.1.2. MCCLK Frequency

The MCLK_FREQ preference allows you to alter the MCLK frequency used to retrieve data from an external SPI Flash when using EXTERNAL or Dual Boot configuration modes. The ECP5 and ECP5-5G devices use a nominal 2.4 MHz (±15%) clock frequency to begin retrieving data from the external SPI Flash. The MCLK_FREQ value is stored in the incoming configuration data. The ECP5 and ECP5-5G devices read a series of padding bits, a “start of data” word (0xBDB3) and a control register value. The control register contains the new MCLK_FREQ value. The ECP5 and ECP5-5G devices switch to the new clock frequency shortly after receiving the MCLK_FREQ value. The MCLK_FREQ has a range of possible frequencies available from 2.4 MHz up to 62 MHz (see Table 4.6). Take care not to exceed the maximum clock rate of your SPI Flash, or of your printed circuit board.

7.1.3. TRANSFR

The TransFR function used by the ECP5 and ECP5-5G devices require the configuration data loaded into the configuration SRAM, and any future configuration data file loaded into the external Flash memory have the TRANSFR set to the ENABLE state. See the TransFR Operation section and Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198) for more information about using TransFR.

7.1.4. COMPRESS_CONFIG

The COMPRESS_CONFIG preference alters the way files are generated. The COMPRESS_CONFIG default setting is to be ON.

7.1.5. UNIQUE_ID

The ECP5 and ECP5-5G devices contain a 16-bit register for storing a user-defined value. The default value stored in the register is 0x0000. UniqueID can only be set when USERCODE_FORMAT is Auto.

7.1.6. USERCODE

The ECP5 and ECP5-5G devices contain a 32-bit register for storing a user-defined value. The default value stored in the register is 0x00000000. Using the USERCODE preference you can assign any value to the register you desire. Suggested uses include the configuration data version number, a manufacturing ID code, or the date of assembly.

The format of the USERCODE field is controlled using the USERCODE_FORMAT preference. Data entry can be performed in either Binary, Hex, ASCII or Auto formats.

7.1.7. USERCODE_FORMAT

The USERCODE_FORMAT preference selects the format for the data field used to assign a value in the USERCODE preference. The USERCODE_FORMAT has four options:

- Binary – USERCODE is set using 32 *1* or *0* characters
- Hex – USERCODE is set using eight hexadecimal digits, that is 0-9A-F
- ASCII – USERCODE is set using up to four ASCII characters
- Auto – USERCODE is automatically created by the software. The upper 16 bits is UniqueID and the lower 16 bits are sequentially increased automatically for every bitstream generation.

7.1.8. CONFIG_SECURE

When this preference set to ON, the read-back of the SRAM memory is blocked. The device must be reprogrammed in order to reset the security setting. Once the security fuses are reset, the device can be programmed again.

8. Device Wake-up Sequence

When configuration is complete (the SRAM has been loaded), the device wakes up in a predictable fashion. If the ECP5 or the ECP5-5G device is the only devices in the chain, or the last device in a chain, the wake-up process should be initiated by the completion of configuration. Once configuration is complete, the internal DONE bit is set and then the wake-up process begins. Figure 8.1 shows the wake-up sequence 21 using the internal clock.

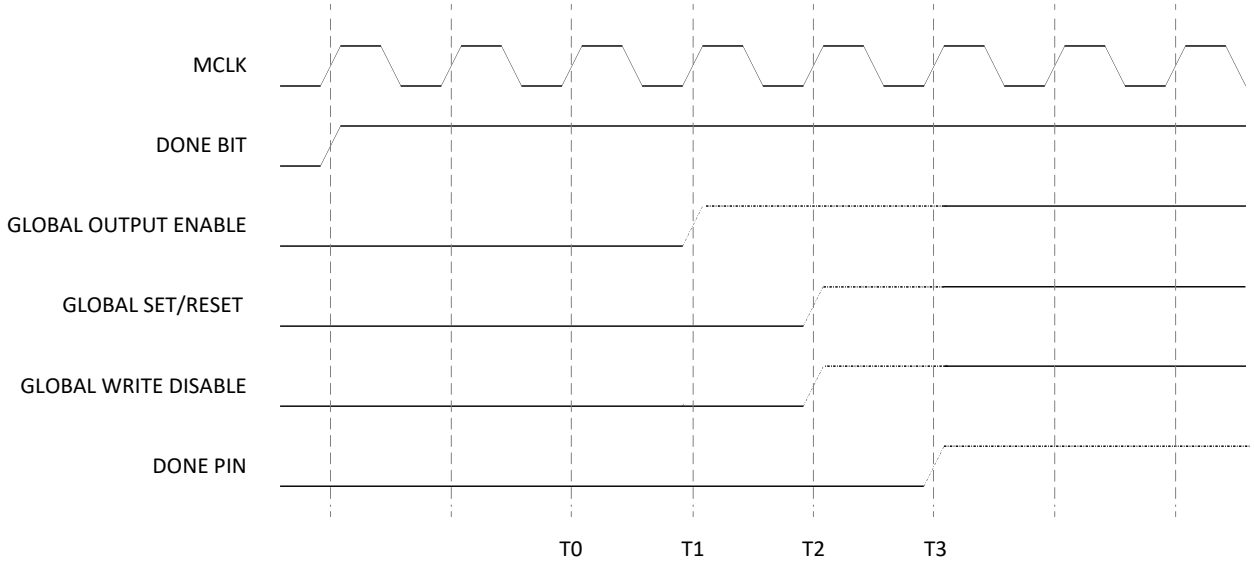


Figure 8.1. Wake-up Sequence Using Internal Clock

8.1. Wake-up Signals

Three internal signals, GSR, GWDIS, and GOE, determine the wake-up sequence.

- GSR is used to set and reset the core of the device. GSR is asserted (low) during configuration and de-asserted (high) in the wake-up sequence.
- When the GWDIS signal is low it safeguards the integrity of the RAM Blocks and LUTs in the device. This signal is low before the device wakes up. This control signal does not control the primary input pin to the device but controls specific control ports of EBR and LUTs.
- When low, GOE prevents the device's I/O buffers from driving the pins. The GOE only controls output pins. Once the internal DONE is asserted the ECP5 and ECP5-5G devices responds to input data.
- When high, the DONE pin indicates that configuration is complete and that no errors were detected.
- Before DONE pin goes high, all signals going to EBR should hold steady, otherwise, it might impact the EBR initialization.

The wake-up sequences available to you are shown in Table 8.1. A wake-up sequence is the order in which the signals change. The phases transition based on a wakeup clock, as discussed below. The exact timing relationship between the internal signals and the wakeup clock varies and is not specified.

Table 8.1. Wake-up Sequences

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
4 (Default if DONE_EX = ON)	DONE	GOE	GWDIS, GSR	—
21 (Default if DONE_EX = OFF)	—	GOE	GWDIS, GSR	DONE

8.2. Wake-up Clock Selection

The clock source used to complete the four state transitions in the wake-up sequence is user-selectable. Once the ECP5 and ECP5-5G devices are configured, it enters the wake-up state, which is the transition between the configuration mode and user mode. This sequence is synchronized to a clock source, which defaults to internal clock. The start-up clock can be user-defined and brought into the device. This user clock cannot exceed 100 MHz and instantiated as shown below in the user design.

You can change the clock used by instantiating the START macro in your Verilog or VHDL. The clock must be supplied on an external input pin, because the ECP5 and ECP5-5G devices do not begin internal operations until the Wake-up sequence is complete. There is no external indication the device is ready to perform the last four state transitions. You must either provide a free running clock frequency, or you must wait until the device is guaranteed to be ready to wake up. Using the START macro provides another mechanism for holding off configuring one or more programmable devices and then starting them in lock step.

Verilog

```
module START (STARTCLK);
input     STARTCLK;
endmodule

START u1 (.STARTCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

VHDL

```
COMPONENT START
    PORT(
        STARTCLK       :      IN STD_ULOGIC
    );
END COMPONENT;
attribute syn_noprune: boolean ;
attribute syn_noprune of START: component is true;

begin
u1: START port map (STARTCLK =><clock name>);
```

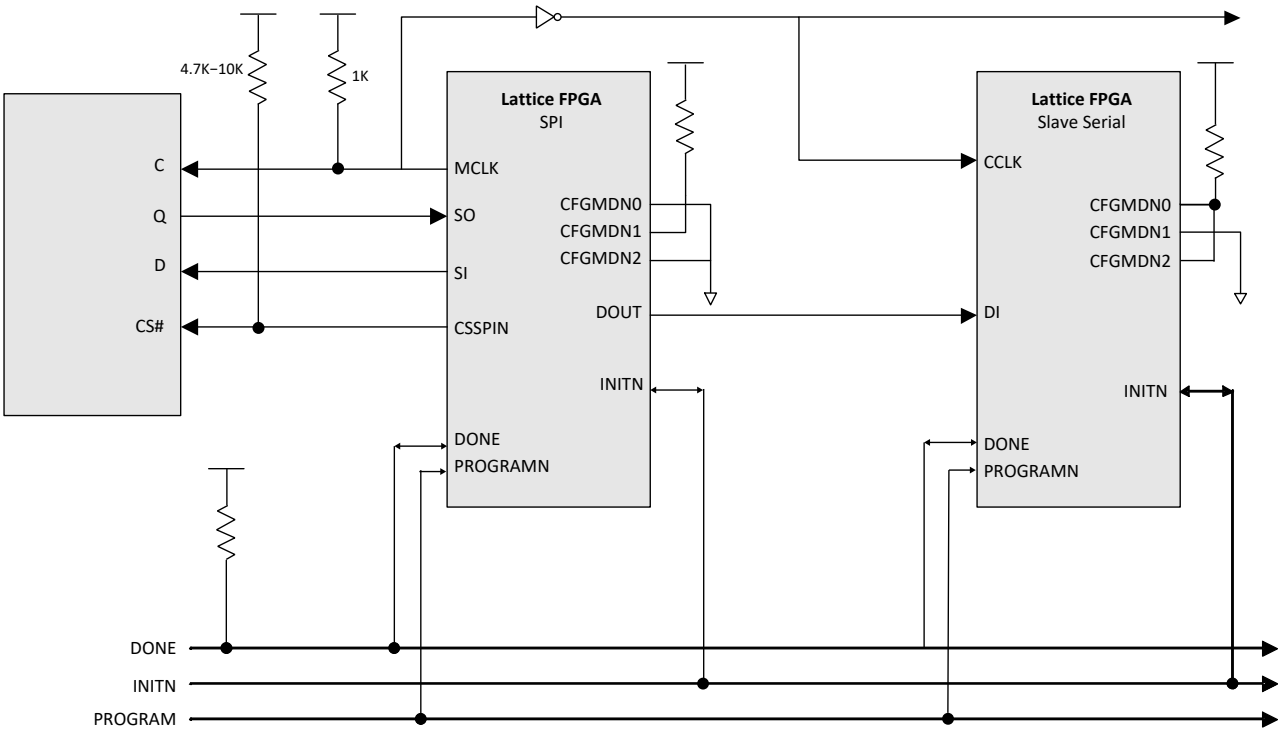
9. Daisy Chaining

Typically, there is one configuration bitstream per FPGA in a system. Today's systems have several FPGA devices. If all the FPGAs in the application utilize the same device and use the same bitstream, only a single bitstream is required. Using a ganged configuration loads multiple, similar FPGAs with the same bitstream at the same time.

However, to save PCB space and use external storage device more efficiently, several different FPGA bitstreams from various devices and designs can share a single configuration mechanism by using a daisy chain method.

ECP5 and ECP5-5G devices support two distinct daisy chaining methods. Bypass Option or Flow through option must be set for all Lattice FPGA devices in the sysCONFIG chain when using daisy chaining. The 'Synchronous to External DONE pin' option (DONE_EX) must be enabled in Lattice Diamond Spreadsheet View.

The first method supports multiple FPGAs from a single configuration source such as a SPI Flash device or the serial port of a microprocessor. In this scenario, the storage device contains all the configuration data to program all the FPGAs. The data comes into the first FPGA by its sysCONFIG port programming the first part. After the first part completes its configuration, it serially sends the remaining data from its DOUT output to the daisy chained device which receives the data into the SCM sysCONFIG port. Examples are shown in Figure 9.1 and Figure 9.2. Bypass Option must be set in this scenario.



Note: Inverter from MCLK is to guarantee DI hold time of the daisy chained FPGA device.

Figure 9.1. Serial Daisy Chaining

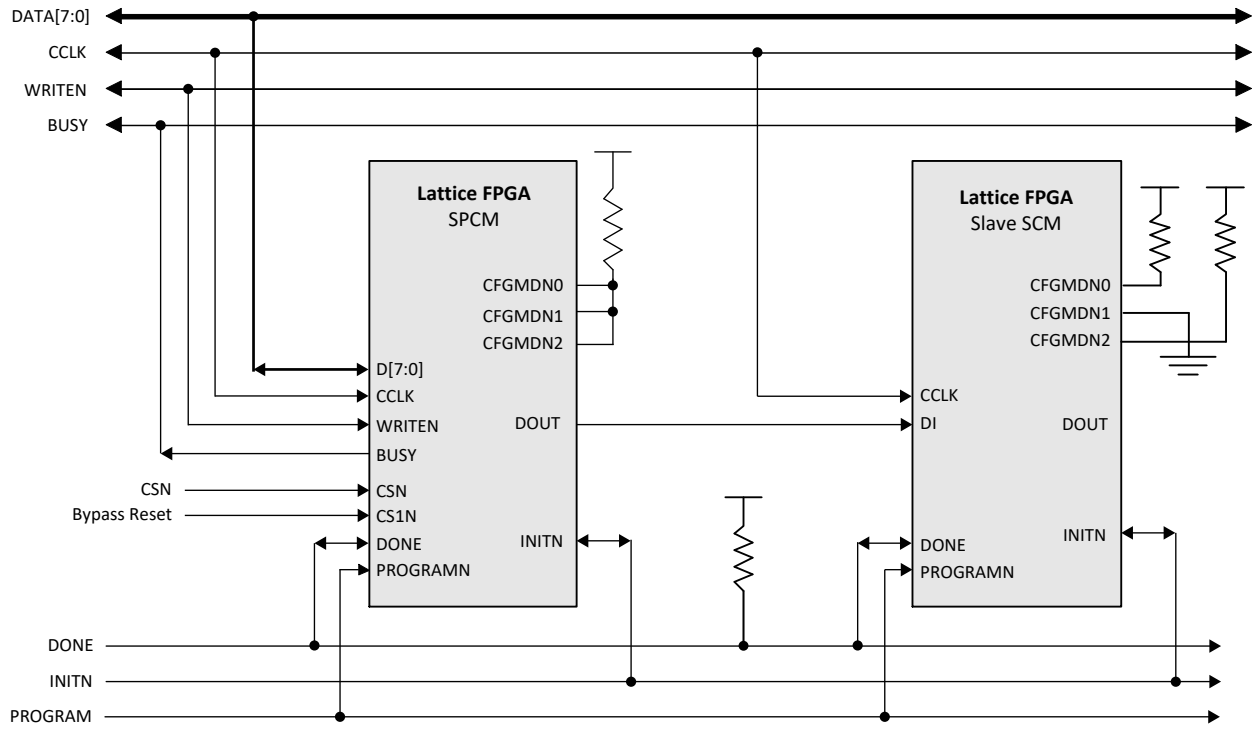


Figure 9.2. SPCM with Serial Daisy Chain

The second method permits sharing an 8-bit data bus from a storage device or CPU. The FPGAs use handshaking to signal which FPGA on the bus is reading the configuration data. An example is shown in Figure 9.3. Flow through option must be set in this scenario.

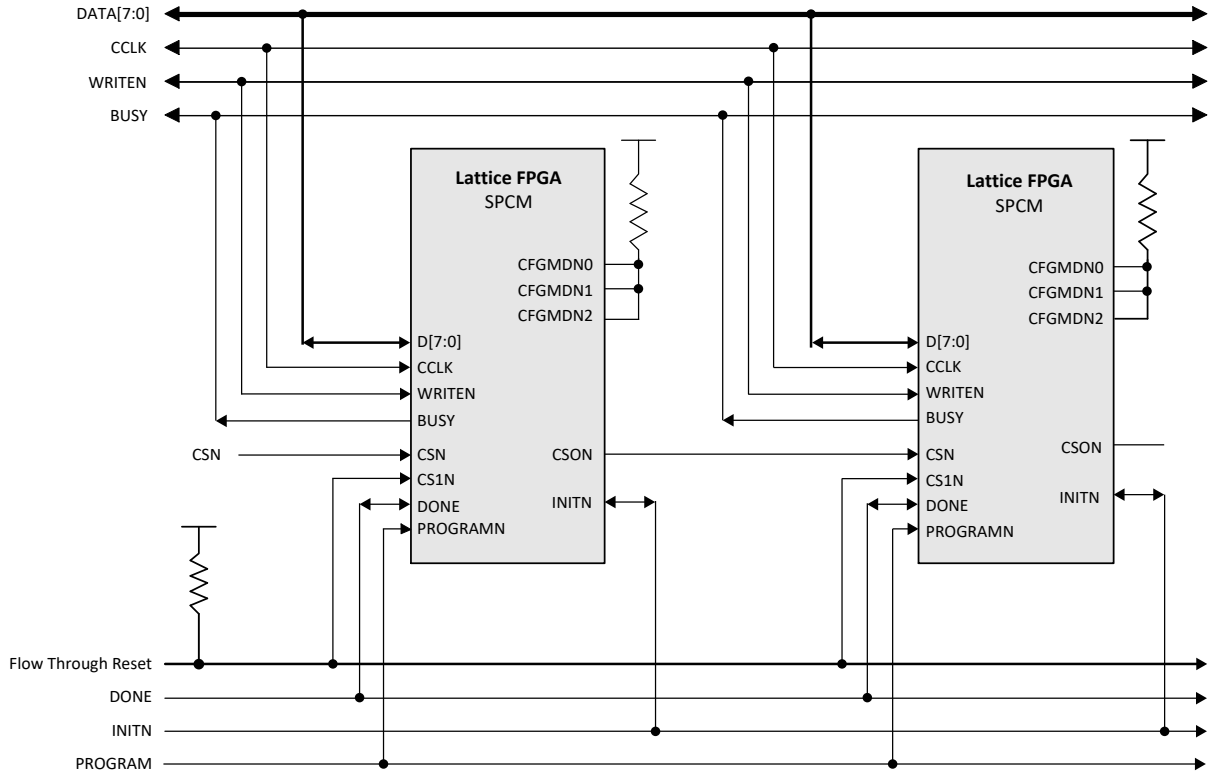


Figure 9.3. SPCM Configuration with Daisy Chaining

9.1. Bypass Option

The Bypass option can be set by using the Diamond strategy properties, or a chain of bitstreams can be assembled and Bypass set using the Diamond Deployment Tool. The Bypass option is not supported when using dual-, or multi-boot configurations. The Bypass option is supported when using encrypted bitstream files with ECP5 and ECP5-5G devices.

When the first device completes configuration, and a Bypass command is included within the bitstream, any additional data coming into the FPGA configuration port overflows serially on DOUT. This data is applied to the DI pin of the next device (downstream devices must be set to Slave Serial mode).

In Serial Configuration mode, the Bypass option connects DI to DOUT through a bypass register. The bypass register is initialized with a '1' at the beginning of configuration and stays at that value until the Bypass command is executed.

In parallel configuration modes, the Bypass option causes the excess data coming in on D[15:0] to be serially shifted to DOUT. The serialized data is shifted to DOUT through a bypass register. D0 is shifted out first followed by D1, D2, and so on. Once the Bypass option starts, the device remains in Bypass until the wake-up sequence completes. In parallel mode, if Bypass needs to be aborted, drive both CSN and CS1N high. This acts as a Bypass reset signal.

9.2. Flow through Option

The flow through option can be used with parallel daisy chains only. The flow through option is not supported when using encrypted bitstream files with ECP5 and ECP5-5G devices. Flow through does not support SCM, SSPI, SPI, and SPI modes.

When the first device completes configuration and a flow through command is included with the bitstream, the CSON pin is driven low. In addition to driving CSON low, Flow through also tristates the device's D[15:0] and BUSY pins in order to avoid contention with the other daisy chained devices. Once the flow through option starts, the device remains in flow through until the wake-up sequence completes. If flow through needs to be aborted, drive both CSN and CS1N high. This acts as a flow through reset signal.

Appendix A. ECP5 Slave SPI Programming Guide

The SPI port of the ECP5 and ECP5-5G devices can be used for device configuration. After configuration, the SPI pins can be used as user I/O except MCLK/CCLK which is tristated. If these pins are not used by user logic, they are tristated with a weak pull-up. The Slave SPI port must be enabled in order to support device configuration from an external host or download cable using SPI protocol. This is done by setting the SLAVE_SPI_PORT preference to ENABLE in the bitstream through the Lattice Diamond Spreadsheet View. Slave SPI mode supports single device configuration.

The Lattice Diamond Programmer supports the Slave SPI programming mode as one of the device access options. Selecting this option allows you to perform device erase, program, verify, readback, refresh, and more. The connections of Slave SPI pins to the Lattice programming cable are:

- TDI -> SISPI
- ISPEN -> SN
- TCK -> CCLK
- TDO -> SPISO

The Slave SPI chip select pin (SN) is held low during the command sequences. You can generate a SVF file from their bitstream (.bit) with the Lattice Deployment Tool software to show the details of the command sequences for Slave SPI programming mode.

Appendix B. ECP5 and ECP5-5G Bitstream File Format

Configuration Bitstream Format

The base binary file format is the same for all non-encrypted, non-1532 configuration modes. Different file types (hex, binary, ASCII, and so on) may ultimately be used to configure the device, but the data in the file is the same. Table B.1 lists the format of a non-encrypted and compressed bitstream. The bitstream consists of a comment string, a header, the preamble, and the configuration setup and data.

Only the frame data can be compressed. The EBR data cannot be compressed. The data frame padding for compression is as follows:

- Before compression, pad the leftmost bits of the frame data with zeroes (0) to make the data frame 64-bit bounded. For example, if the original uncompressed data frame length is 125-bit, such as 01.....10, the data frame has to be padded with all 0 (3-bit 0s) at leftmost to make it 64-bit bounded (128-bit), 00001.....10.
- After compressing the frame data, pad the rightmost bits of the frame data with zeroes (0) to make the data frame byte bounded. For example, if the 128-bit data frame is compressed to become 101-bit (not byte bounded), such as 10.....01, the compressed data frame has to be padded with all 0 (3-bit 0s) at the rightmost to make it byte bounded (104-bit), 10……01000

Table B.1. ECP5 and ECP5-5G Compressed Bitstream Format

Frame	Contents (D0..D7)	Description	CRC
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator.	None
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Bitstream Preamble.	
	1011110110110011	Second 16 bits of the 32-bit Bitstream Preamble (0xFFFFBDB3).	
	11111111…11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000…0000	23-bit Command Information (23 zeros).	Exclude
Frame (Verify ID) See Note 6	11100010	VERIFY_ID Command.	Start: Include if [6]
	0	CRC Comparison Flag (0 = no) for Verify ID command.	Include if [6]
	000…0000	23-bit Command Information (23 zeros).	Include if [6]
	(DeviceID[31..0])	32-bit Device ID.	Include if [6]
Frame (Store Compress)	00000010	LSC_WRITE_COMP_DIC Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
	(Pattern8[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
	(Pattern7[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
	•	•	•
	•	•	•
	(Pattern1[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
	(CtlReg0[31..0])	Control Register 0 Data (See Note 8).	Include

Frame	Contents (D0..D7)	Description	CRC
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
Frame (Write Increment)	10111000	LSC_PROG_INCR_CMP Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame).	Include
	0	Include dummy bits in bitstream.	Include
	1	Dummy definition (1 = use Bit[3:0] as dummy byte count).	Include
	0001	Dummy definition.	Include
	(16-bit Frame Size)	Number of configuration data frames included in operand (see Table B.4). For example: 2819 Frames = 0x0B03 or b000101100000011.	—
Frame (Data 0)	(Frame 0 Data)	x Data bits for Frame 0 (See Table B.4 and Notes 1 and 2).	Include: End
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	x Data bits for Frame 1 (See Table B.4 and Notes 1 and 2).	Include: End
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
• • •	• • •	• • •	• • •
Frame (Data n-1)	(Frame n-1 Data)	x Data bits for Frame n-1 (See Table B.4 and Notes 1 and	Include: End
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (SED CRC) (Optional) See Note 3	10100010	LSC_PROG_SED_CRC Command.	Include if [3]
	0	CRC Comparison Flag (0 = no).	Include if [3]
	000…0000	23-bit Command Information (23 zeros).	Include if [3]
	(CRC[31..0])	32-bit SED CRC (See Note 3).	Include if [3]
Frame (Program Security) (Optional) See Note 7	11001110	ISC_PROGRAM_SECURITY Command (See Note 7).	Include if [7]
	0	CRC Comparison Flag (0 = no).	Include if [7]
	000…0000	23-bit Command Information (23 zeros).	Include if [7]
Frame (Usercode)	11000010	ISC_PROGRAM_USERCODE Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
	(U[31..0])	32-bit Usercode Data.	Include: End
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
Frame (EBR Address) (Optional) See Note 5	11110110	EBR Address Command.	Start: Include if [5]
	0	CRC Comparison Flag (0 = no).	Include if [5]
	000…0000	23-bit Command Information (23 zeros).	Include if [5]
	00000000000	11-bit dummy (11 zeros).	Include if [5]
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if [5]
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if [5]

Frame	Contents (D0..D7)	Description	CRC
Frame (EBR Write) (Optional) See Note 5	10110010	LSC_EBR_WRITE Command.	Include if [5]
	1	CRC Comparison Flag (1 = yes).	Include if [5]
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if [5]
	0	Exclude dummy bytes from bitstream.	Include if [5]
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if [5]
	0000	Dummy definition.	Include if[5]
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if [5]
	(Data)	Data frame for one EBR. For 256 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0],	Include End if [5]
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
• • •	• • •	• • •	• • •
Frame (EBR Address) (Optional) See Note 5	11110110	EBR Address Command.	Start: Include if [5]
	0	CRC Comparison Flag (0 = no).	Include if [5]
	000…0000	23-bit Command Information (23 zeros).	Include if [5]
	00000000000	11-bit dummy (11 zeros).	Include if [5]
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if [5]
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if [5]
Frame (EBR Write) (Optional) See Note 5	10110010	LSC_EBR_WRITE Command.	Include if [5]
	1	CRC Comparison Flag (1 = yes).	Include if [5]
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if [5]
	0	Exclude dummy bytes from bitstream.	Include if [5]
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if [5]
	0000	Dummy definition.	Include if [5]
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if [5]
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0],	Include End if [5]
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000…0000	23-bit Command Information (23 zeros).	Exclude
Frame (DUMMY)	1111…1111	4 bytes DUMMY command (all ones).	Exclude

Table B.2. ECP5 and ECP5-5G Uncompressed Bitstream Format

Frame	Contents (D0..D7)	Description	CRC
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator.	None
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Bitstream Preamble.	
	1011110110110011	Second 16 bits of the 32-bit Bitstream Preamble (0xFFFFBDB3).	
	11111111…11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000…0000	23-bit Command Information (23 zeros).	Exclude
Frame (Verify ID) See Note 6	11100010	VERIFY_ID Command.	Start: Include if [6]
	0	CRC Comparison Flag (0 = no) for Verify ID command.	Include if [6]
	000…0000	23-bit Command Information (23 zeros).	Include if [6]
	(DeviceID[31..0])	32-bit Device ID.	Include if [6]
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
	(CtlReg0[31..0])	Control Register 0 Data (See Note 8).	Include
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR_RTI Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame).	Include
	0	Include dummy bits in bitstream.	Include
	1	Dummy definition (1 = use Bit[3:0] as dummy byte count).	Include
	0001	Dummy definition.	Include
	(16-bit Frame Size)	Number of configuration data frames included in operand (see Table B.4). For example: 2819 Frames = 0x0B03 or b000101100000011.	Include
Frame (Data 0)	(Frame 0 Data)	x Data bits for Frame 0 (See Table B.4 and Note 2).	—
Include: End	—	—	—
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	x Data bits for Frame 1 (See Table B.4 and Note 2).	—
Include: End	—	—	—
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
• • •	• • •	• • •	• • •
Frame (Data n-1)	(Frame n-1 Data)	x Data bits for Frame n-1 (See Table B.4 and Note 2).	
Include: End			
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include

Frame	Contents (D0..D7)	Description	CRC
Frame (SED CRC) (Optional) See Note 3	10100010	LSC_PROG_SED_CRC Command.	Include if [3]
	0	CRC Comparison Flag (0 = no).	Include if [3]
	000…0000	23-bit Command Information (23 zeros).	Include if [3]
	(CRC[31..0])	32-bit SED CRC (See Note 3).	Include if [3]
Frame (Program Security) (Optional) See Note 7	11001110	ISC_PROGRAM_SECURITY Command (See Note 7).	Include if [7]
	0	CRC Comparison Flag (0 = no).	Include if [7]
	000…0000	23-bit Command Information (23 zeros).	Include if [7]
Frame (Usercode)	11000010	ISC_PROGRAM_USERCODE Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	000…0000	23-bit Command Information (23 zeros).	Include
	(U[31..0])	32-bit Usercode Data.	Include: End
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
Frame (EBR Address) (Optional) See Note 5	11110110	EBR Address Command.	Start: Include if5
	0	CRC Comparison Flag (0 = no).	Include if [5]
	000…0000	23-bit Command Information (23 zeros).	Include if [5]
	00000000000	11-bit dummy (11 zeros).	Include if [5]
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if [5]
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if [5]
Frame (EBR Write) (Optional) See Note 6	10110010	LSC_EBR_WRITE Command.	Include if [5]
	1	CRC Comparison Flag (1 = yes).	Include if [5]
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if [5]
	0	Exclude dummy bytes from bitstream.	Include if [5]
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if [5]
	0000	Dummy definition.	Include if [5]
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if [5]
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], …	Include: End if [5]
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
• • •	• • •	• • •	• • •
Frame (EBR Address) (Optional) See Note 6	11110110	EBR Address Command.	Start: Include if [5]
	0	CRC Comparison Flag (0 = no).	Include if [5]
	000…0000	23-bit Command Information (23 zeros).	Include if [5]
	00000000000	11-bit dummy (11 zeros).	Include if [5]
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if [5]
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if [5]

Frame	Contents (D0..D7)	Description	CRC
Frame (EBR Write) (Optional) See Note 5	10110010	LSC_EBR_WRITE Command.	Include if [5]
	1	CRC Comparison Flag (1 = yes).	Include if [5]
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if [5]
	0	Exclude dummy bytes from bitstream.	Include if [5]
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if [5]
	0000	Dummy definition.	Include if [5]
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if [5]
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], …	Include: End if [5]
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000…0000	23-bit Command Information (23 zeros).	Exclude
Frame (DUMMY)	1111…1111	4 bytes DUMMY command (all ones).	Exclude

Notes:

1. Data is byte bounded and must be padded with ones (1). For example, a 6-bit data frame of b111010 is written in the bitstream as b11111010 to make it byte bounded.
2. Only the data frame bits, highlighted in yellow, can be compressed.
3. If SED option is not chosen, place command DUMMY (NOOP) (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option you chose. It must be replaced with the LSC_PROG_SED_CRC frame by Bitgen if you implement SED. Include in CRC calculation if LSC_PROG_SED_CRC command. Do not include in CRC calculation if NOOP.
4. The CRC must be flipped so that CRC[0] is the LSB and CRC[15] is the MSB. For example, CRC = 0x0823 (CRC[15]…CRC[0]) in the bit file above is b1100010000010000. Refer to Appendix Generic Configuration Specification for CRC calculation details.
5. Insert a Write EBR Frame for each initialized EBR, or for each group of EBR Frames that is initialized to the same value. Omit if there is no EBR initialization data.
6. By default, this frame must be the Verify ID command. Bitgen shall have a switch to allow the Verify ID frame to be replaced with NOOP (all 1's). Include in CRC calculation if Verify ID command. Do not include in CRC calculation if NOOP.
7. If security option is not chosen, place command NOOP (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option you chose. Include in CRC calculation if Program Security command. Do not include in CRC calculation if NOOP.
8. The Control Register 0 must be set to the SW Default value, setting Control Register 0 bits [31..30] = [0..0].

Read Back Bitstream Format

If required, Diamond generates a binary read back file. The read back file contains an ASCII comment string, device ID, control register 0 data, frame data, dummy bits, LUT, EBR, and user code data. It does not contain a header, commands, or CRC. The data is ready to be shifted into the devices as required by the programming flow. The data is listed from MSB to LSB, where the MSB is the first bit shifted into TDI and the LSB the last bit. The MSB is also the first bit shifted out of TDO. The data for each frame is byte bounded. The byte must be padded with all ones (1). For example, a 6-bit data frame of b111010 would be written in the readback file as b11111010. The format is described in Table B.3 on the next page.

Table B.3. ECP5 and ECP5-5G Read Back File Format

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String
Comment Terminator	11111110	Read Back File Comment Terminator = 0xFE (binary file (.rbk) only)
Device ID	(ID[0..31])	Expected 32-bit Device ID (See Note 1)
Control Register 0	(CtlReg0[0..31])	Control Register 0 Data
Device Specific Padding	1111…1111	Terminator bytes (all 1's). See Table B.4 for number of bytes.
Frame 0	1111…1111	Dummy (Stop) bytes (all 1's). See Table B.4 for number of bytes.
	(Frame 0 Data)	Data for Frame 0 (byte bounded, padded with zeros)
Frame 1	1111…1111	Dummy (Stop) bytes (all 1's). See Table B.4 for number of bytes.
	(Frame 1 Data)	Data for Frame 1 (byte bounded, padded with zeros)
• • •	• • •	• • •
Frame n-1	1111…1111	Dummy (Stop) bytes (all 1's). See Table B.4 for number of bytes.
	(Frame n-1 Data)	Data for Frame n-1 (byte bounded, padded with zeros)
Usercode	(U[0..31])	32-bit Usercode Data

ECP5 and ECP5-5G Device Specific

Table B.4. ECP5 and ECP5-5G Device Specifics

Device Name	Configuration Frames (n)	Byte Bound Padding Bits	Data Bits per Frame (w)	Total Data Bits per Frame (x)	CRC Bits
LFE5-12	7562	0	592	592	16
LFE5-25, LFE5UM-25, LFE5UM5G-25	7562	0	592	592	16
LFE5-45, LFE5UM-45, LFE5UM5G-45	9470	2	846	848	16
LFE5-85, LFE5UM-85, LFE5UM5G-85	13294	0	1136	1136	16

Table B.5. ECP5 and ECP5-5G Device ID

Device Name	32-bit IDCODE
LFE5U-12	0x21111043
LFE5U-25	0x41111043
LFE5U-45	0x41112043
LFE5U-85	0x41113043
LFE5UM-25	0x01111043
LFE5UM-45	0x01112043
LFE5UM-85	0x01113043
LFE5UM5G-25	0x81111043
LFE5UM5G-45	0x81112043
LFE5UM5G-85	0x81113043

Appendix C. Advanced Applications – The Slave SPI Port and SPI Flash Interface

The system diagram shown in Figure C.1 illustrates an application of the Slave SPI interface. In this example, the CPU can independently access the external SPI flash for CPU boot-up code or for the FPGA bitstream. The FPGA selects Slave SPI as the primary boot source. When the CPU needs to configure the FPGA, it can drive CS0N low then shift the WRITE_EN into the device command to set the FGPA device into the Slave SPI configuration mode.

In this example, the CPU reads the configuration data from the SPI Flash and shifts it into the FPGA. There are two methods this can be done.

- The CPU reads the entire bitstream from the SPI Flash before shifting it into the FPGA. This method is much simpler to support, but requires sufficient RAM to store the entire bitstream.
- The CPU can read a frame of the bitstream from the SPI Flash device, shift the frame into the FPGA, and repeat for each frame. The method requires less RAM, and it does not require the CPU to have knowledge of the bitstream format. However, it does require use of the HOLDN pin.

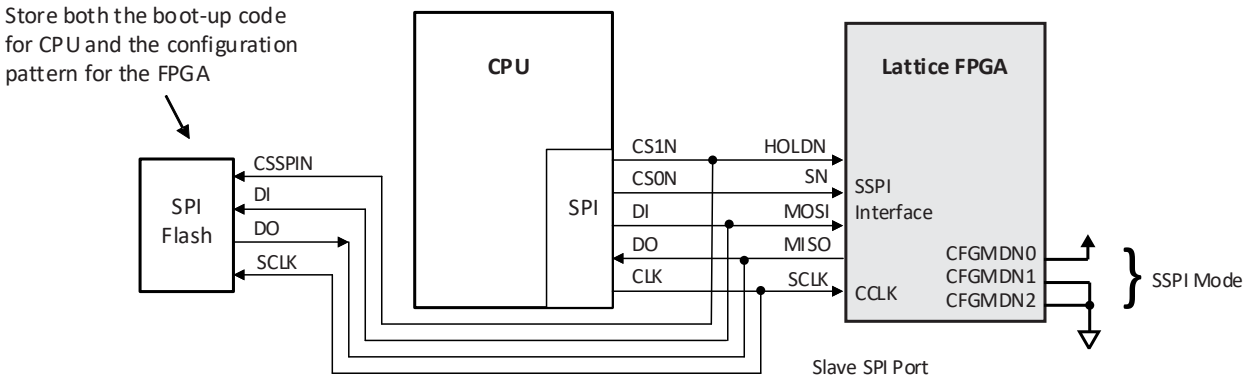
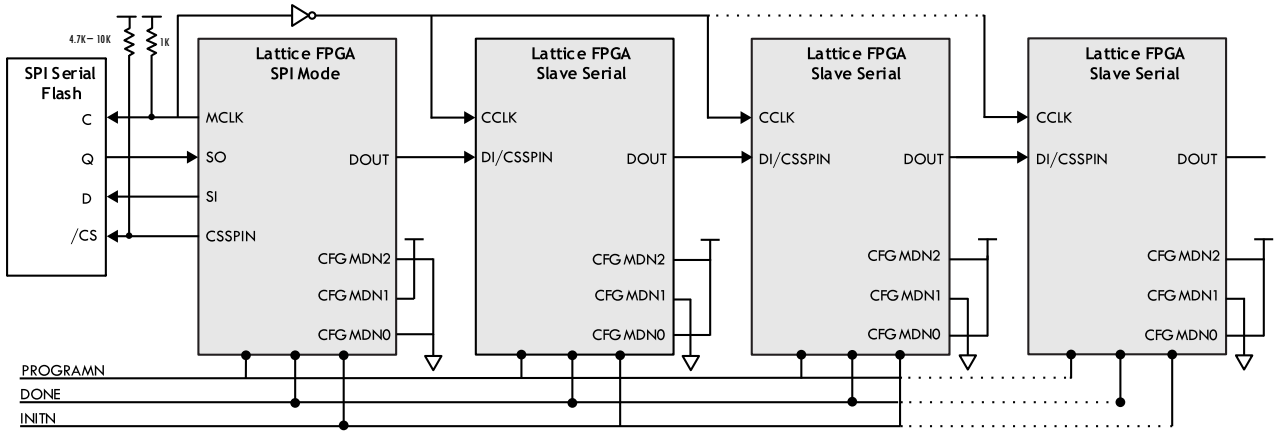


Figure C.1. Example Slave SPI System Diagram (Slave SPI Mode)

Appendix D. Advanced Applications – Master SPI sysCONFIG Daisy Chaining

With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure D.1. The requirements are as follows:

- The first device must be an ECP5 or an ECP5-5G device in Master SPI mode.
- The rest of the FPGA devices must be in Slave Serial Configuration (SCM) mode. The bitstream files for the daisy chain can be merged using the Deployment Tool.
- The DONE pin of all the FPGA devices must be connected together. This allows the devices to detect when the last device is done configuring.
- The 'Synchronous to External DONE Pin' option (DONE_EX) must be enabled in the Lattice Diamond Spreadsheet view for all Lattice FPGA devices in the sysCONFIG chain.
- The Bypass option must be set by using the Diamond 'Bitgen' properties for all Lattice FPGA devices in the sysCONFIG chain.



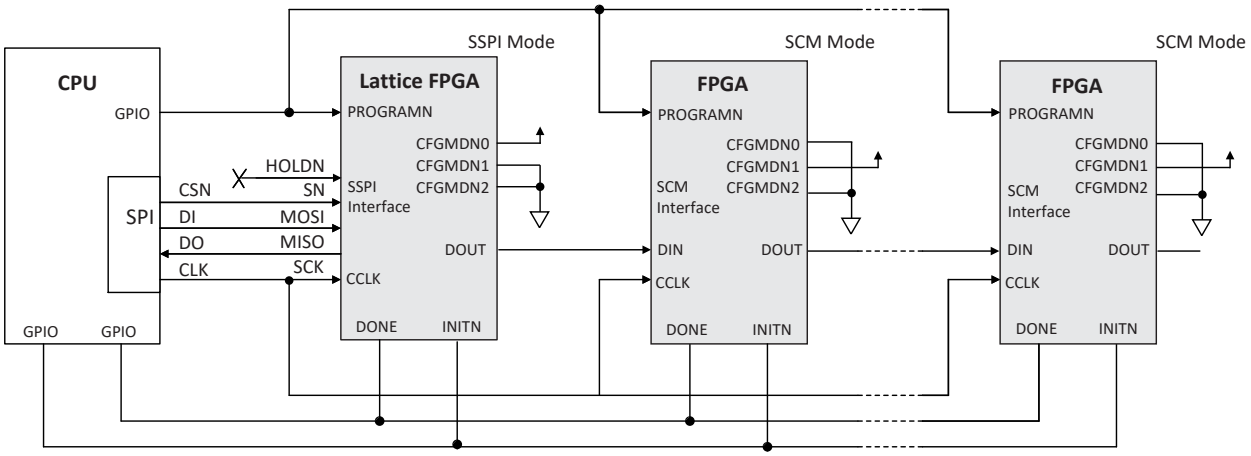
Note: The dashed lines indicate the connection is optional. The inverter for MCLK guarantees the hold time for data input to the daisy chained FPGAs.

Figure D.1. Master SPI sysCONFIG Daisy Chain

Appendix E. Advanced Applications – Slave SPI sysCONFIG Daisy Chaining

Even though the Slave SPI port does not explicitly support daisy chaining, a sysCONFIG chain of other FPGA devices can be configured using the SPI port of a CPU. A block diagram is shown in Figure E.1. The requirements are as follows:

- The first device must be an ECP5 or an ECP5-5G device in Slave SPI mode.
- The rest of the FPGA devices must be in Slave Serial Configuration (SCM) mode. The bitstream files for the daisy chain can be merged using the Deployment Tool.
- The DONE pin of all the FPGA devices must be connected together. This allows the devices to detect when the last device is done configuring.
- The *Synchronous to External DONE Pin* option (DONE_EX) must be enabled in the Lattice Diamond Spreadsheet view for all Lattice FPGA devices in the sysCONFIG chain.
- The Bypass option must be set by using the Diamond 'Bitgen' properties for all Lattice FPGA devices in the sysCONFIG chain.



Note: The dashed lines indicate the connection is optional.

Figure E.1. Slave SPI sysCONFIG Daisy Chain

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 2.0, September 2021

Section	Change Summary
Configuration Modes	• Removed '5, 6' from 'LSC_PROG_INCR_RTI' and added '5, 6' to 'LSC_BITSTREAM_BURST' in Table 6.5. Slave SPI Command Usage Table. • Removed Table 6.4. AC Timing Requirements. • Changed the title of Section 6.2.3 from "Slave SPI Port AC Timing Requirements" to "Slave SPI List of Command.
Configuration Details	Changed 'read be' to 'be read by' in Section 4.2. Device Status Register.
Software Selectable Options	• Removed 'JEDEC file checksum' from Section 7.1.6. USERCODE. • Changed 'three options' to 'four options' in Section 7.1.7. USERCODE_FORMAT.

Revision 1.9, June 2021

Section	Change Summary
Appendix C	Updated Figure C.1. Example Slave SPI System Diagram (Slave SPI Mode).

Revision 1.8, September 2020

Section	Change Summary
Disclaimers	Added this section.
Configuration Details	Updated CS1N section content to add this statement: *Both CSN and CS1N need to be driven low to enable the parallel port. Driving CSN high pauses the data transfer and driving CS1N high resets port*.
Appendix D. Advanced Applications – Master SPI sysCONFIG Daisy Chaining	Updated Figure D.1 to remove SPIFASTN.
Revision History	Updated format.

Revision 1.7, March 2018

Section	Change Summary
All	• Changed document number from TN1260 to FPGA-TN-02039. • Updated document template. • Updated linked references
Configuration Details	Updated Configuration Details section. • Updated Bitstream/PROM Sizes section. Revised Table 4.1. Maximum Configuration Bits. Added two rows for LFE5-12. • Updated sysCONFIG Pins section. Revised footnote 6 in Table 4.4. Default State of the sysCONFIG Pins. • Updated Table 4.7. sysCONFIG Pins Global Preferences in Dual-Purpose sysCONFIG Pins section.
Configuration Process and Flow	Updated text in Wake-up section.
Configuration Modes	Updated Configuration Modes section. • Removed a text in Method to Enable the Master SPI Port section. • Replaced text in Customized SPI Port in User Mode section. • Revised and added six rows at the end of Table 6.5. Slave SPI Command Table. • Revised and added six rows at the end of Table 6.6. Slave SPI Command Usage Table. Updated footnote. • Updated Figure 6.10. Slave SPI Configuration Flow. • Updated Figure 6.17. Slave Parallel Read
Appendix A, ECP5 Slave SPI Programming Guide	Revised Appendix A, ECP5 Slave SPI Programming Guide
Appendix B, ECP5 and ECP5-5G Bitstream File Format	Revised Table B.1, Table B.2, Table B.4, and Table B.5 in Appendix B, ECP5 and ECP5-5G Bitstream File Format.

Revision 1.6, February 2016

Section	Change Summary
Configuration Process and Flow	Updated Configuration Process and Flow section. Revised Figure 5.1, Configuration Flow.

Revision 1.5, November 2015

Section	Change Summary
All	• Added support for ECP5-5G device. • Changed document title to ECP5 and ECP5-5G sysCONFIG Usage Guide.
sysCONFIG Pins	Updated sysCONFIG Pins section. Revised table note 6 in Table 4.4. Default State of the sysCONFIG Pins.
Configuration Details	• Updated JTAG Configuration Port Pins section. Revised Table 4.8. JTAG Port Pins. Added table note. • Changed VCCIO8 Pin Function. • Changed VCCJ to VCCIO8 in TDO, TDI and TMS descriptions. • Updated WRITEN section. Added note.
Configuration Modes	Updated Customized SPI Port in User Mode section. Revised note on USERMCLKTS.
Technical Support Assistance	Updated Technical Support Assistance section.

Revision 1.4, February 2015

Section	Change Summary
Device Wake-up Sequence	Updated Wake-up Signals section. Added information on DONE pin.

Revision 1.3, December 2014

Section	Change Summary
Configuration Modes	Updated Dual and Quad Master SPI Read Mode section. Added information on SPI read mode.

Revision 1.2, August 2014

Section	Change Summary
Software Selectable Options	Updated MCCLK Frequency section. Revised statement to indicate that MCLK_FREQ range of possible frequencies is from 2.4 MHz to 62 MHz.

Revision 1.1, July 2014

Section	Change Summary
Configuration Details	• Updated Table 4.4. Default State of the sysCONFIG Pins. Added table note to MCLK/CCLK Pin Type. • Updated Table 4.6. ECP5 and ECP5-5G MCLK Valid Frequencies. Added MCLK frequency 62. • Corrected USERMCLKI in Figure 6.2. MCLK Connection notes.
Configuration Modes	• Updated Master SPI Modes section. Added information on generating submode. • Updated Customized SPI Port in User Mode section. Added Figure 6.2. MCLK Connection and notes. • Updated Dual and Quad Master SPI Read Mode section. Removed statement on MASTER SPI support for all bus widths and submodes being controlled by control register settings.
Configuration Process and Flow	• Updated Figure 5.1. Configuration Flow. • Update Power-up Sequence section. Changed conditions for POR circuit to release internal reset strobe.
Software Selectable Options	Updated Table 7.1. sysCONFIG Options. Added note.
All	Corrected table and figure numbering.

Revision 1.0, March 2014

Section	Change Summary
All	Initial release.

www.latticesemi.com