



Adding Scalable Power and Thermal Management to ECP5 Using L-ASC10

Technical Note

FPGA-AN-02019-1.2

August 2023

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 and with all faults, and all risk associated with such information is entirely with 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

1. Introduction	5
2. Overview	5
2.1. L-ASC10 (Hardware Management Expander)	5
3. ECP5 FPGAs	5
4. Hardware Connections for ECP5 + ASC	6
4.1. Interconnect Signals Between ECP5 and ASC	7
4.2. ECP5 Configuration Signals	8
4.3. Power Supply Requirements	9
4.4. Programming and Configuration of ECP5 + ASC	10
4.5. Power-On Reset and Configuration Behavior	10
4.6. ECP5 Power-On-Reset and Configuration	11
4.7. ASC Boot Operation	13
4.8. Mandatory/Optional ASC Designation	14
4.9. Delaying ASC Boot with ext_rstn_i	14
4.10. Special Considerations for ASC Boot	15
4.11. Device Behavior During Powering Up and Programming	15
4.12. Dual-Boot	15
5. Platform Designer – ECP5 Support Overview	17
5.1. Creating a new ECP5 + ASC Project	17
5.2. Migrating from MachXO2 + ASC or MachXO3 + ASC to ECP5 + ASC	17
5.3. Platform Designer – Unique Feature Summary for ECP5 + ASC	18
5.4. Global View: ASC & Device Options – ECP5 Specific Features	18
5.5. Device Options – External Connected Components Mode	20
6. I ² C/SPI Interface Blocks	20
6.1. Overview	20
6.2. I ² C Slave Addressing Restriction	20
6.3. Configurable I ² C and SPI Clock Rates	21
7. Fault Logging	22
7.1. Fault Log Software Configuration	22
7.2. V, I, T Fault Log Configuration	22
7.3. Full Featured Fault Log Configuration	23
7.4. Full Featured Fault Log Hardware Connections	29
8. Platform Manager 2 I ² C GUI	31
9. Special Considerations for Designing with ECP5 + ASC in Lattice Diamond	32
9.1. Use of LGB_Clock or PLD_Clock in Imported HDL	32
9.2. Place & Route Trace Evaluation – Potential Need for Hold Time Optimization	33
9.3. Post-Map Simulation – Test Bench Update	34
Appendix A. ASC Boot Flow Chart	37
Technical Support Assistance	41
Revision History	42

Figures

Figure 4.1. ECP5 + ASC System Block Diagram	7
Figure 4.2. Diamond Deployment Tool – Output File Type for Quad-SPI Setup	11
Figure 4.3. Setting Dual or Quad I/O SPI Flash Read Mode	11
Figure 4.4. Spreadsheet View Interface for Configuration Parameters	12
Figure 4.5. ECP5 Power-On-Reset with 62 MHz Quad-SPI and Compressed Bitstream	13
Figure 4.6. ASC Configuration Time for 3 ASCs using 400 kHz I ² C Clock	14
Figure 4.7. Diamond Accessory Deployment	16

Figure 4.8. Deployment Tool – Advanced SPI Flash - Multiple Boot Setup	16
Figure 5.1. New Project Device Selector	17
Figure 5.2. Device Selector	18
Figure 5.3. SPI Flash Background Programming – JTAG.....	19
Figure 5.4. SPI Flash Background Programming – SPI.....	20
Figure 7.1. Fault Logger Component Selection	22
Figure 7.2. V, I, T Fault Log Configuration	23
Figure 7.3. Full Featured Fault Log Configuration View	24
Figure 7.4. SPI Memory Map for Single SPI Memory	25
Figure 7.5. Hardware Connections for Design with Configuration SPI Memory for Fault Logging	30
Figure 7.6. Hardware Connections for Design with Alternate SPI Memory for Fault Logging	30
Figure 8.1. Passthrough Connection to Support Platform Manager 2 I ² C GUI	32
Figure 9.1. Enabling Place & Route Trace in the Diamond Process View	33
Figure 9.2. Place & Route Trace – Hold Report.....	33
Figure 9.3. Place & Route Strategy Settings.....	34
Figure 9.4. Command Line Setting for Hold Time Optimization	34
Figure 9.5. SDA Glitch in Post-Map Simulation	35

Tables

Table 4.1. ECP5 + ASC System Interconnect Signals	8
Table 4.2. ECP5 Master SPI Connections	9
Table 4.3. ECP5 + ASC - System Power Rails	10
Table 4.4. ECP5 Master SPI Configuration Options.....	11
Table 4.5. ECP5 Master SPI Configuration Time with Various Settings	12
Table 4.6. ASC Boot – Configuration Times	13
Table 4.7. Restricted ASC Features in ECP5 + ASC Systems	15
Table 4.8. ECP5 + ASC – Pin Status During Configuration	15
Table 5.1. Unique Feature Summary for ECP5 Solutions	18
Table 6.1. Tcl Commands for I2C and SPI Clock Rate Settings	21
Table 6.2. I2C Clock Rate Settings vs i2cbr Setting	21
Table 6.3. Common SPI Clock Settings vs spibr Setting	21
Table 7.1. Detailed Fault Record for Single ASC with Time Stamp	26
Table 7.2. Detailed Fault Record for Single ASC without Time Stamp	26
Table 7.3. Detailed Fault Record for Three ASCs with Time Stamp.	27
Table 7.4. Maximum Fault Record Byte Map	28
Table 9.1. ECP5 + ASC Clocks	32

1. Introduction

Complex hardware systems require a large number of resources for sensing analog signals, programmable digital control and many GPIOs to provide interface with external components. It is desired to have a flexible system where above resources can be easily scaled as per application requirement. Lattice Platform Manager™ 2 family consists of Hardware Management Expander (L-ASC10), which in combination with hardware management controller, MachXO2™, MachXO3™, and ECP5™ FPGAs enable designers to create a flexible and scalable hardware system.

Lattice Diamond® software includes the Platform Designer tool which is used to configure L-ASC10 and MachXO2, MachXO3, or ECP5 FPGA. The tool also provides a mechanism to set up the interface between L-ASC10 and the controlling FPGA. With the help of the Platform designer tool, the user can easily build solutions for Voltage, Current, Temperature monitoring, fault-logging, hot-swap, trimming and margining, power supply sequencing and supervision by adding L-ASC10 to a MachXO2, MachXO3, or ECP5 design.

This document explains the procedure to add L-ASC10 device to new or existing ECP5 designs. It describes the Platform Designer tool settings and external hardware connections to construct a system having power and thermal management using ECP5 and L-ASC10 devices. For details on adding L-ASC10 devices to new or existing MachXO2 and MachXO3 designs, please refer to [Adding Scalable Power and Thermal Management to MachXO2 and MachXO3 Using L-ASC10 \(FPGA-AN-02011\)](#).

2. Overview

2.1. L-ASC10 (Hardware Management Expander)

The L-ASC10 (Analog Sense and Control - 10 rails) is a Hardware Management (Power, Thermal, and Control Plane Management) Expander designed to be used with ECP5 FPGAs to implement the Hardware Management Control function in a circuit board. The L-ASC10 (referred to as ASC) enables seamless scaling of power supply voltage and current monitoring, temperature monitoring, sequence and margin control channels. The ASC includes dedicated interfaces supporting the exchange of monitor signal status and output control signals with ECP5. Up to eight ASC devices can be used to implement a hardware management system.

The ASC provides three types of analog sense channels: voltage (nine standard channels and one high voltage channel), current (one standard voltage and one high voltage), and temperature (two external and one internal). The device incorporates nine general purpose 5 V tolerant open-drain digital input/output pins, four high-voltage charge pumped outputs (HVOUT1-HVOUT4) and four TRIM outputs for controlling the output voltages of DCDC converters.

The dedicated ASC Interface (ASC-I/F) is a reliable serial channel used to communicate with an ECP5 FPGA in a scalable star topology. The centralized control algorithm in the FPGA monitors signal status and controls output behavior via this ASC-I/F. The ASC I2C interface is used by the FPGA for ASC background programming, interface configuration, and additional data transfer such as parameter measurement or I/O control or status.

3. ECP5 FPGAs

The L-ASC10 (ASC) hardware management expander is designed to work in conjunction with MachXO2, MachXO, and ECP5 FPGAs to provide a complete scalable hardware management solution. This application note describes specific hardware and software considerations for designing hardware management systems around the ECP5 FPGA and ASC device(s). This solution is supported for the 12K, 25K and 45K LUT variants of the ECP5.

The ECP5 + ASC hardware management solution is based on the same building blocks as the MachXO2 + ASC and MachXO3 + ASC hardware management solutions. In particular, the following components perform essentially the same function:

- ASC Devices (See [L-ASC10 Data Sheet \(FPGA-DS-02038\)](#) for specification and description of L-ASC10)
- Platform Designer Software Tool
- Building Block IPs (Background instantiated in design)
 - ASC Interface
 - FPGA Clock & Reset Management

- Hardware Management Logic Generation (Sequence and Supervisory)
- Component and Function IPs
 - Fan Controller
 - VID

Due to the architectural differences between the ECP5 and MachXO2 or MachXO3 FPGAs, several of the design components are customized for ECP5 usage. The customizations have been made to preserve functionality wherever possible, only making changes to address specific hardware mismatches. The solution features that have been customized include the following:

- Programming & Configuration
 - FPGA Programming / Configuration
 - ASC Programming / Configuration
 - Dual-Boot Support
- Full-Featured Fault Logging IP Component

There is also a short list of features whose support is not available with the ECP5 + ASC solution. The features which are unavailable in the ECP5 + ASC solution include the following:

- ASC features requiring EEPROM access
 - ASC UES Codes
 - I2C Write Protect and User Tag Memory
 - I2C Address Reprogramming
- Component and Function IPs
 - Hot Swap
 - PMBus Adapter

This application note covers the details regarding ECP5 + ASC hardware management solution including hardware connections, programming and configuration considerations, software operation, and component IP behavior.

4. Hardware Connections for ECP5 + ASC

Systems built on ECP5 and ASC should refer first and foremost to the following documents:

- [ECP5 Family Data Sheet \(FPGA-DS-02012\)](#)
- [L-ASC10 Data Sheet \(FPGA-DS-02038\)](#) – especially System Connections sub-section of Theory of Operation
- [ECP5 and ECP5-5G Hardware Checklist \(FPGA-TN-02038\)](#)
- [ECP5 and ECP5-5G sysCONFIG Usage Guide \(FPGA-TN-02039\)](#)

There are three general categories of connections for an ECP5 + ASC system:

- Interconnect signals between ECP5 + ASC
- ECP5 configuration signals
- Power for ECP5 + ASC

[Figure 4.1](#) is a system level block diagram showing the connections from all three categories. This figure only shows the device connections, there are passive components for many of the connections (both pull-up / pull-down resistors as well as capacitors) not shown. Also not shown in the figure are the connections required to implement the SPI-Flash based fault logging. See the Fault Logging section of this document for information related to those connections.

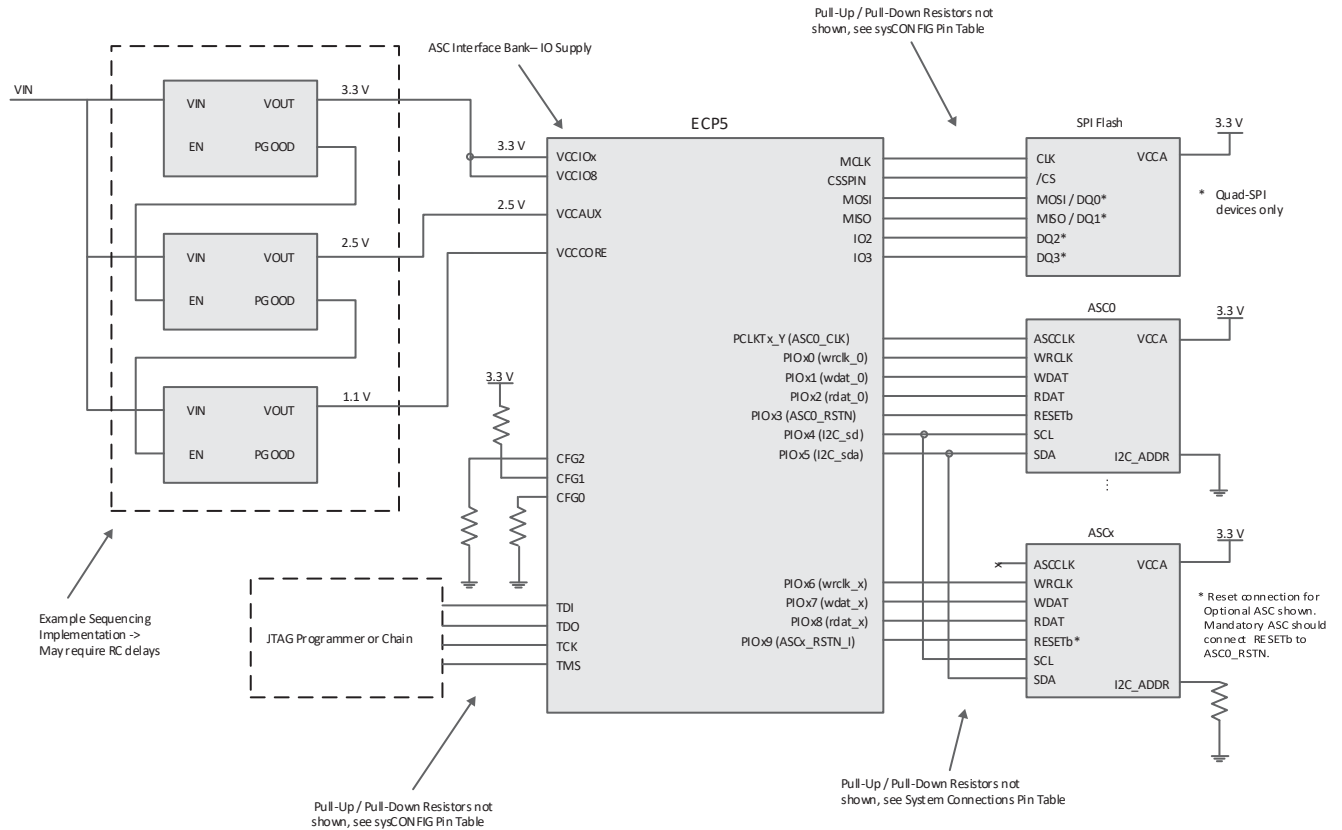


Figure 4.1. ECP5 + ASC System Block Diagram

4.1. Interconnect Signals Between ECP5 and ASC

Table 4.1 provides an overview of the ECP5 + ASC system interconnect signals. The required connections are mostly similar to the MachXO2 + ASC or MachXO3 + ASC system connections. The key difference is that the I2C signals on ECP5 are assignable to any I/O signal in the package, versus the MachXO2 and MachXO3 primary Embedded Function Block (EFB) I2C signals which are locked to a specific location on the MachXO2 and MachXO3. Table 4.1 lists each signal, with ECP5 pin details, I/O bank restrictions, ASC Pin, and any additional comments.

Table 4.1. ECP5 + ASC System Interconnect Signals

Interconnect Signals	ECP5 Pin	ECP5 I/O Bank Restrictions	ASC Pin	Comments
ASC0 to ECP5				
ASC0_CLK	Any PCLK	VCCIO = VCCA (3.3 V) of ASC0; All ASC0 signals in same bank	ASCCLK (ASC0)	N/A
ASC0_RSTN	Any PIO	—	RESETb (ASC0)	10k Pull-up to VCCA
wrclk_0	Any PIO	—	WRCLK (ASC0)	—
wdat_0	Any PIO	—	WDAT (ASC0)	—
rdat_0	Any PIO	—	RDAT (ASC0)	—
I2C_sda	Any PIO ¹	VCCIO = VCCA (3.3 V) of ASC0;	SCL (All ASCs)	Open Drain I/O on ECP5, pull-up to VCCA (2.2 kOhm typical)
I2C_scl	Any PIO ¹	—	SDA (All ASCs)	
I2C_ADDR	N/A	—	I2C_ADDR (ASC0)	Connect to GND On ASC0
ASCx to ECP5				
ASCx_CLK	N/A	—	ASCCLK (ASCx)	Leave ASCCLK pin open for all ASC except ASC0
ASCx_RSTN_I2	Any PIO	VCCIO = VCCA (3.3 V) of ASCx; All ASCx signals in same bank	RESETb (ASCx)	10k Pull-up to VCCA
wrclk_x	Any PIO	—	WRCLK (ASCx)	—
wdat_x	Any PIO	—	WDAT (ASCx)	—
rdat_x	Any PIO	—	RDAT (ASCx)	—
I2C_sda	ASC0 SDA PIO1	VCCIO = VCCA (3.3 V) of ASC0;	SCL (All ASCs)	Open Drain I/O on ECP5, pull-up to VCCA (2.2 kOhm typical)
I2C_scl	ASC0 SCL PIO1	—	SDA (All ASCs)	
I2C_ADDR	N/A	—	I2C_ADDR (ASCx)	Connect to resistor to GND, per Table 7.4 of DS1042, L-ASC10 Data Sheet.

Notes:

1. There is a single I2C_sda and single I2C_scl pin assignment on the ECP5, all ASC devices should connect to these common pins.
2. ASC devices designated as optional will have their own ASCx_RSTN_I signal connection. ASC devices designated as mandatory should connect their RESETb signal to ASC0_RSTN.

4.2. ECP5 Configuration Signals

The ECP5 provides several options for configuration and programming the device, as described in [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#). The key differences in the ECP5 + ASC system versus the MachXO2 + ASC or MachXO3 + ASC system are the following:

- MachXO2 and MachXO3 include on-chip configuration flash as well as SRAM, ECP5 only includes configuration SRAM
- MachXO2 and MachXO3 provide a hardened JTAG to I2C bridge for programming (used in PTM Programming mode of MachXO2 and MachXO3), this is not present in ECP5
- MachXO2 can be programmed over I2C, ECP5 does not support I2C programming
- ECP5 Master SPI mode supports Quad-SPI Flash devices, MachXO2 and MachXO3 only work with standard SPI Flash devices

More details on the programming of ECP5 can be found in the Programming and Configuration of ECP5 + ASC section of this document. The ECP5 + ASC system has been designed to use the ECP5 Master SPI configuration mode. [Table 4.2](#) shows the signals related to ECP5 programming and Master SPI configuration mode, including information on which pins can be shared as user I/O, required external components, and general comments.

Table 4.2. ECP5 Master SPI Connections

sysCONFIG Pins ¹		External Component Connection	Comments
Name	Type		
CFG[2:0]	Dedicated	CFG0 – 500 Ohm Pull-Down CFG1 – 4.7k Pull-Up (VCCIO8) CFG2 – 500 Ohm Pull-Down	010 Setting for Master SPI mode (ECP5 + ASC supported mode)
DONE	Dedicated	10k Pull-Up (VCCIO8)	—
PROGRAMN	Dedicated	10k Pull-Up (VCCIO8)	—
INITN	Dedicated	10k Pull-Up (VCCIO8)	—
TDI	Dedicated	4.7k Pull-Up (VCCIO8)	—
TMS	Dedicated	4.7k Pull-Up (VCCIO8)	—
TDO	Dedicated	4.7k Pull-Up (VCCIO8)	—
TCK	Dedicated	4.7k Pull-Down	—
MCLK/CCLK	Dedicated ²	1k Pull-Up (VCCIO8)	—
D3/IO3	Shared	10k Pull-Up (VCCIO8); Quad-SPI DQ3	Quad-SPI Only
D2/IO2	Shared	10k Pull-Up (VCCIO8); Quad-SPI DQ2	Quad-SPI Only
D1/MISO	Shared	10k Pull-Up (VCCIO8); SPI Flash MISO/Quad-SPI DQ1	—
D0/MOSI	Shared	10k Pull-Up (VCCIO8); SPI Flash MOSI/Quad-SPI DQ0	—
CSSPIN	Shared	4.7k Pull-Up (VCCIO8)	—

Notes:

1. See [ECP5 SysCONFIG Usage Guide \(FPGA-TN-02039\)](#) and [ECP5 Hardware Checklist \(FPGA-TN-02038\)](#) for the most up-to-date configuration information.
2. MCLK pin can be driven in user mode, see [ECP5 SysCONFIG Usage Guide \(FPGA-TN-02039\)](#), USRMCLK macro discussion.

The ASC devices are configured after the ECP5 Master SPI configuration has completed. The ASC configuration is described in more detail in the ASC Boot Operation section of this document.

The Master SPI mode for ECP5 also supports dual boot configuration as well as single/dual/and quad-SPI flash memories. This information is detailed in the ECP5 Power-On-Reset & Configuration section of the document.

4.3. Power Supply Requirements

The ECP5 + ASC solution includes several voltage rails as shown in [Table 4.3](#). The ECP5 FPGA requirements are detailed in the [ECP5 and ECP5-5G Family Data Sheet \(FPGA-DS-02012\)](#) and [ECP5 and ECP5-5G Hardware Checklist \(FPGA-TN-02038\)](#). The most critical voltage rail requirements for ECP5 are the following:

- Minimum 3 Different Voltage Rails Required (VCC_CORE = 1.1 V, VCC_AUX = 2.5 V, VCCIOx)
- Sequence Requirement: VCCIO8 > VCC_AUX > VCC_CORE (more combinations acceptable, see device data sheet)

Table 4.3. ECP5 + ASC - System Power Rails

Voltage Rail	Voltage	Sequence Requirement	Comments
VCC_AUX	2.5 V	VCCIO8 = GOOD prior to VCC_AUX and VCC_CORE. Example: VCCIO8 > VCC_AUX > VCC_CORE	—
VCC_CORE	1.1 V		—
VCCIO8	3.3 V ²		ECP5 supports multiple configuration voltages for ECP5 + ASC systems, recommend using = 3.3 V
VCCIOx ¹	3.3 V	N/A	Tie to ASC(s) VCCA
VCCA (ASC)	3.3 V	—	—
VCCA _x (ECP5)	1.1 V	—	Connect to GND if SERDES unused
VCCAUXA _x	2.5 V	—	Connect to GND if SERDES unused
VCCHTX	1.2 V	—	Connect to GND if SERDES unused
VCCHRX	1.2 V	—	Connect to GND if SERDES unused
SPI VCC	3.3 V ²	—	SPI VCC must match VCCIO8

Notes:

1. All IO banks connected to ASC devices should be supported by the common 3.3 V rail of VCCA of ASC. Other non ASC IO banks can support a variety of voltages.
2. VCCIO8 supports multiple configuration voltage selections - for ASC systems, recommend using 3.3 V.

The ECP5 + ASC solution uses the Master SPI Boot Mode as detailed in [Table 4.2](#). Recommend using a 3.3 V SPI Flash, with a VCCIO8 (ECP5 configuration voltage bank) of 3.3 V.

In the ECP5 + ASC system, the VCCIO of any IO bank used to connect to the ASC should be connected to VCCA of that ASC. This ensures proper startup and reliable communication between the devices.

ECP5 also includes a set of SERDES voltage rails. Per [ECP5 Hardware Checklist \(FPGA-TN-02038\)](#), these voltages can all be grounded if the SERDES is not used in the application.

4.4. Programming and Configuration of ECP5 + ASC

The ECP5 + ASC system has special considerations for programming compared to the MachXO2 + ASC or MachXO3 + ASC systems. Most importantly, there is no programming path from the ECP5 JTAG to the ASC I2C (called PTM Programming in Diamond Programmer). Instead, the ASC configuration information is loaded into EBR memory of the ECP5, as part of the ECP5 bitstream. After ECP5 configuration, the ASC Boot component configures the ASC devices over the user specified I2C pins. This operation is similar to the Dual Boot behavior of the MachXO2 + ASC or MachXO3 + ASC systems, and is described in the ASC Boot Operation section of this document.

The ECP5 bitstream (including the ASC configuration information) is stored in external SPI Flash. The SPI Flash can be programmed via the ECP5 JTAG interface. This is the SPI Flash Background Programming mode found in Lattice Diamond® programmer. This method is only available when the configuration SPI Flash interface is preserved – that is, not used by the user logic which includes fault logging. If the user logic design includes a SPI Flash interface, the ECP5 must be erased (using JTAG 1532 mode, Erase) prior to executing a SPI Flash Background program from Diamond Programmer.

The Fault Logger IP component also provides an interface for background programming the configuration SPI Flash using an external SPI master (see the Device Options – Operation Mode programming area of the Platform Designer section for more details.)

4.5. Power-On Reset and Configuration Behavior

Understanding of the ECP5 + ASC system power-on-reset and configuration is critical for system design. The following section includes information on the ECP5 Power-On-Reset and configuration behavior, the ASC Boot Operation, and the I/O behavior during powering up and programming.

4.6. ECP5 Power-On-Reset and Configuration

The ECP5 + ASC system uses the ECP5 Master SPI configuration mode. The Master SPI configuration mode is detailed in [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#). The ECP5 will begin its configuration process when all device power-on-reset requirements are met (see the power supply section above).

The time required for the ECP5 to complete configuration from the external SPI memory is dependent on several configurable parameters. The settable options are shown in [Table 4.4](#).

Table 4.4. ECP5 Master SPI Configuration Options

Parameter	Settings	Default Setting	Configuration Interface
SPI Mode	Single, Dual, Quad	Single	Deployment Tool
MCLK Frequency (MHz)	2.4, 4.8, 9.7, 19.4, 38.8, 62	2.4	Spreadsheet View
Bitstream Compression	Off, On	Off	Spreadsheet View
# of Initialized EBRs	0 to 108 ¹	0	HDL Source/Clarity Designer ²

Notes:

1. ECP5-12k includes 32 EBRs, ECP5-25k includes 56 EBRs, ECP5-45k includes 108 EBRs.
2. Clarity Designer is not supported with Diamond. This is to limit unexpected tool interactions between Clarity Designer and Platform Designer.

The SPI Mode is set using Diamond Deployment Tool (found in Lattice Diamond > Accessories menu). Select **Function Type – External Memory** and **Output File Type – Advanced SPI Flash**, as shown in [Figure 4.2](#).

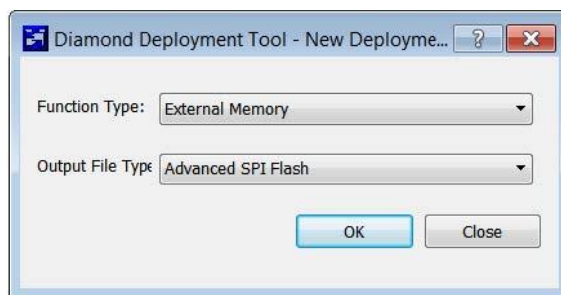


Figure 4.2. Diamond Deployment Tool – Output File Type for Quad-SPI Setup

In the Step 1 dialog of deployment tool, set the input bitstream to the ECP5 + ASC system bitstream generated by Platform Designer. In the Step 2 Dialog, the SPI Flash Size and SPI Flash Read Mode should be set. Quad or Dual I/O SPI Flash Read can be set as the SPI Flash Read Mode in this screen (see [Figure 4.3](#)). Follow the rest of the prompts through the Step 3 & Step 4 Dialogs to generate an MCS file which will support the Dual or Quad I/O SPI Flash.

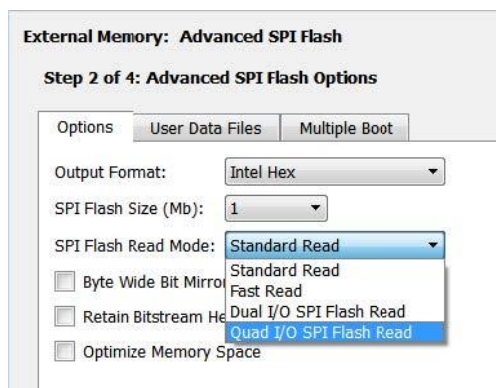
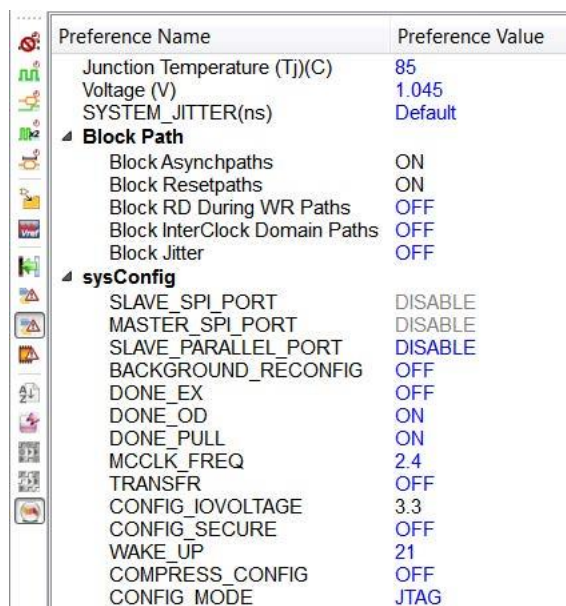


Figure 4.3. Setting Dual or Quad I/O SPI Flash Read Mode

The MCLK frequency is set in the spreadsheet view or in the .lpf file. Figure 4.4 shows the MCCLK_FREQ parameter in the global preferences area of spreadsheet view. The configuration compression is also enabled and disabled in the global preferences area. This is the COMPRESS_CONFIG option also shown in Figure 4.4.



Preference Name	Preference Value
Junction Temperature (Tj)(C)	85
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
BACKGROUND_RECONFIG	OFF
DONE_EX	OFF
DONE_OD	ON
DONE_PULL	ON
MCCLK_FREQ	2.4
TRANSFR	OFF
CONFIG_IOVOLTAGE	3.3
CONFIG_SECURE	OFF
WAKE_UP	21
COMPRESS_CONFIG	OFF
CONFIG_MODE	JTAG

Figure 4.4. Spreadsheet View Interface for Configuration Parameters

The ECP5 configuration times for several different values of these settings are shown in Table 4.5 below. These are all based on initializing a single EBR (adding more initialized EBR will increase the configuration time). The ECP5 configuration time includes a power-on-reset time which takes between 15 ms to 33 ms (see Figure 4.5 below, this is the time between INITN low to MCLK active). The MCLK active time is the time to read the image from the external SPI Flash – this time is influenced directly by the settings in Table 4.5 below. The time in Table 4.5 includes both the power-on-reset time and the MCLK active time.

Table 4.5. ECP5 Master SPI Configuration Time with Various Settings

SPI Read Mode	MCLK	Compression	No. of EBR	Typical ECP5 Configuration Time
Quad	62 MHz	Yes	1	23.3 ms
Quad	62 MHz	No	1	50.4 ms
Quad	38.8 MHz	Yes	1	26.3 ms
Quad	38.8 MHz	No	1	70.5 ms
Single	62 MHz	Yes	1	40.3 ms
Single	62 MHz	No	1	150 ms



Figure 4.5. ECP5 Power-On-Reset with 62 MHz Quad-SPI and Compressed Bitstream

4.7. ASC Boot Operation

The ASC Boot IP component is used to configure the ASC devices in the ECP5 + ASC system. At the conclusion of a successful ECP5 configuration, the ASC Boot IP component will begin configuring the ASC devices. The ASC Boot IP component executes a special sequence of I²C commands (similar to the dual boot component from the MachXO2 + ASC or MachXO3 + ASC systems) based on information stored in the ECP5 EBR memories. The EBRs are loaded during the ECP5 device configuration step described above, eliminating the need for the ASC Boot component to access the external SPI memory during the configuration of the ASC devices.

The ASC Boot process runs each time a power-on-reset or global reset occurs. The ASC Boot configures the shadow register memory in the ASCs, and is therefore a volatile configuration. For the ASC Boot to function correctly, the ASC devices EEPROM memory should always be in the erased state (this is also the shipped state). The ASC Boot behavior is described by the flow chart shown in Appendix A: ASC Boot Flow Chart.

The overall time to complete the ASC configuration depending on the # of ASC devices and the selected I²C speed (see the I²C interface block section below) is shown in Table 4.6 below. There is a short delay between the configuration of each ASC (~22 us).

Table 4.6. ASC Boot – Configuration Times

No. of ASC	I ² C Speed	ASC Configuration Time
1	100 kHz	10.9 ms
1	400 kHz	2.73 ms
3	100 kHz	33.1 ms
3	400 kHz	8.6 ms
8	400 kHz	23 ms

Figure 4.6 below shows the ASC boot configuration time, as measured for a 3 ASC configuration using a 400 kHz I²C clock. This configuration starts up immediately following the completion of the ECP5 configuration (or following the release of ext_rstn_i, see section below on Delaying ASC Boot with ext_rstn_i).



Figure 4.6. ASC Configuration Time for 3 ASCs using 400 kHz I²C Clock

By combining the time to complete the ECP5 configuration (Table 4.5 – depending on parameter settings) and ASC configuration (Table 4.6 – depending on system), the full configuration time from power-on-reset release to system active can be estimated.

4.8. Mandatory/Optional ASC Designation

ASC devices in the ECP5 + ASC system may be designated as mandatory or optional, just as in the MachXO2 + ASC or MachXO3 + ASC systems. Mandatory ASC devices are devices which are always expected to be present in the design. Optional ASC devices may not be present in all design variations, these are most often found on plug-in cards.

The ASC boot component treats mandatory and optional ASC devices differently. If the component fails to configure a mandatory ASC device, either due to an I²C communication error or other issue, the AB_Busy signal (ASC Boot Busy signal) will stay high until a global reset or power cycle. This signal will hold the Platform designer generated logic in reset and should be polled by other user logic which depends on the ASC device input and output data.

If the ASC Boot component does not detect that an optional ASC is present (using the ASC_x_RSTN_I signal state of the given optional ASC), it will skip the configuration step of that device. If the ASC Boot component does detect the optional ASC, it will attempt to configure the device. If the configuration fails, this will be treated as a critical error, and AB_Busy will be held high until a global reset or power cycle. See the flow chart in Appendix A: ASC Boot Flow Chart for more details.

4.9. Delaying ASC Boot with ext_rstn_i

Platform Designer provides the possibility to delay the ASC Boot configuration using an internal signal called ext_rstn_i. This active low signal acts as a reset input to the ASC Boot component (and several other components, see below for comments on handling this signal). By default, this signal is disabled – held high. The signal can be enabled by using the following tcl command (input to the tcl console while Platform Designer is open):

```
psb_global set -ext_rstn_enable
```

This command will activate the `ext_rstn_i` signal in the design, it will be shown in the nodes tab of the Ports & Nodes view. The signal can be tied to user logic (so that it can be set or cleared using a user-defined I2C register for example) or connected via the Logic view, supervisory tab to an external port.

This signal must be treated with care. It is provided as a mechanism for delaying the ASC Boot operation, however it serves to place all IP components and the Platform Designer generated logic into its reset state. It is not recommended to set this signal low during normal operation, as the logic sequences will be reset, rather than move through a safe power-down routine. The signal should only be used immediately after power-on-reset to delay the ASC boot operation, or to re-try the operation after a failure (when `AB_Busy` remains high).

4.10. Special Considerations for ASC Boot

The ASC Boot process only configures the shadow register portion of the ASC configuration memory, it does not access the configuration EEPROM. The following ASC features (Table 4.7) can only be programmed in EEPROM and are therefore fixed to their default setting in the ECP5 + ASC system.

Table 4.7. Restricted ASC Features in ECP5 + ASC Systems

Feature	Default Setting
I ² C Write Protect	I2C Write Enabled
User Tag Enable	Disabled (ASC Fault Log Enabled)
ASCCLK Output	Enabled
I ² C Base Address	0x60
UES Bits	0xFFFFFFFF

For the ECP5 + ASC system to function properly, the ASC devices should be in their erased/default state. This ensures that the ASCCLK will be output from the ASC0 device in the system.

4.11. Device Behavior During Powering Up and Programming

The power-on-reset behavior of the ECP5 + ASC system is a two-stage process – first the ECP5 device configures from external SPI Flash, then the ASC devices are configured from EBR memory in the ECP5. The I/O behavior during these two states is shown in Table 4.8 below.

Table 4.8. ECP5 + ASC – Pin Status During Configuration

Pin Type	Pin Status During ECP5 Configuration	Pin Status During ASC Configuration
ECP5 PIO	Weak pull-down	Logical Reset State
ASC GPIO1-6 ¹	Low/Active pull-down	Low/Active pull-down
ASC GPIO 8-9	Hi-Z	Hi-Z
HVOUT	Low /Active pull-down	Low /Active pull-down
TRIM	Hi-Z	Hi-Z

Note:

- GPIO7 is not bonded out in L-ASC10.

The ECP5 I/O have a weak pull-down which is active during the configuration stage. Upon completion of the ECP5 configuration, the ECP5 PIO will transition to the reset state defined by the user logic. The ASC GPIO are held in their safe state (see [L-ASC10 Data Sheet \(FPGA-DS-02038\)](#) for details) until the `AB_Busy` signal is released indicating a successful configuration of the ASC devices.

4.12. Dual-Boot

The ECP5 FPGA supports both Dual-Boot and Multi-Boot configurations. For details see [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#). The ASC configuration images are stored in initialized EBR memory, which are part of both the Golden and Primary Dual Boot images. Since the ASC images are always loaded after ECP5 configuration (via the ASC Boot Operation), there is no need for a separate ASC Dual Boot operation.

ECP5 Dual-Boot is configured in the Diamond Deployment Tool (an Accessory installed under the Diamond folder, see [Figure 4.7](#)). There are two available methods to setup the dual-boot image. The first is by selecting Function Type – External Memory and Output File Type – Dual Boot. This is a very simple interface to generate the dual boot memory image for the external SPI Flash. However this method does not support the Quad-SPI read method for the configuration process.

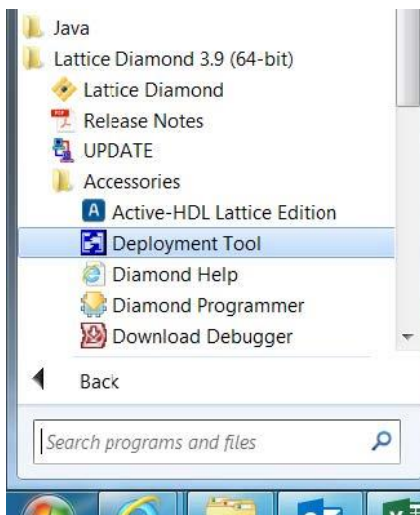


Figure 4.7. Diamond Accessory Deployment

The second method is to select Function Type – External Memory and Output File Type – Advanced SPI Flash (this is shown in [Figure 4.2](#) above in the ECP5 Power-on-reset and Configuration section). This method supports Quad-SPI read. The Multiple Boot tab is used to enable the dual Boot deployment and select the Golden Pattern (see [Figure 4.8](#) below). The Diamond tool requires the user to assign an additional Alternate Pattern at Starting Address – 0x00280000, a total of 3 boot images. This takes additional space in the SPI Flash and must be accounted for if the Fault Logger IP will use the same flash.

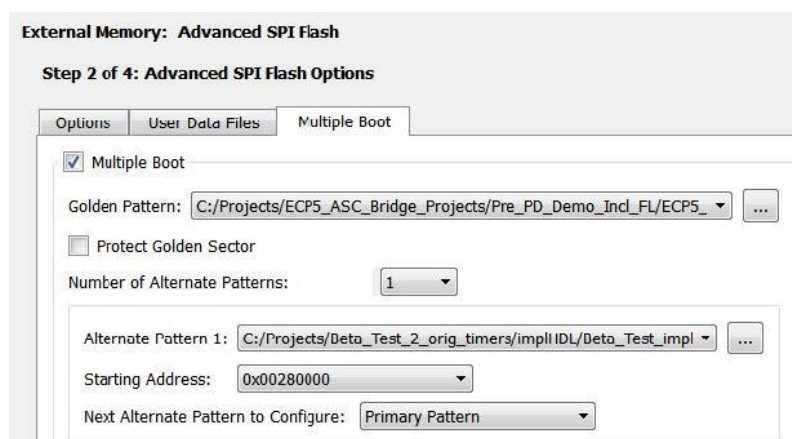


Figure 4.8. Deployment Tool – Advanced SPI Flash - Multiple Boot Setup

5. Platform Designer – ECP5 Support Overview

5.1. Creating a new ECP5 + ASC Project

1. From the Diamond Start Page, select **Project > New**.
2. Choose **Next**.
3. Select a Project Name, Location, and Implementation. Then choose **Next**.
4. Add source files if desired. Files can also be added to the project later in the design flow.
5. Choose any of the following ECP5 Devices:
 - a. ECP5U: LFE5U-12F, LFE5U-25F, LFE5U-45F
 - b. ECP5UM: LFE5UM-25F, LFE5UM-45F
6. Choose the Package Type
7. Add ASC Devices by increasing the Numbers in the Select ASC Device section (See [Figure 5.1](#) Below)

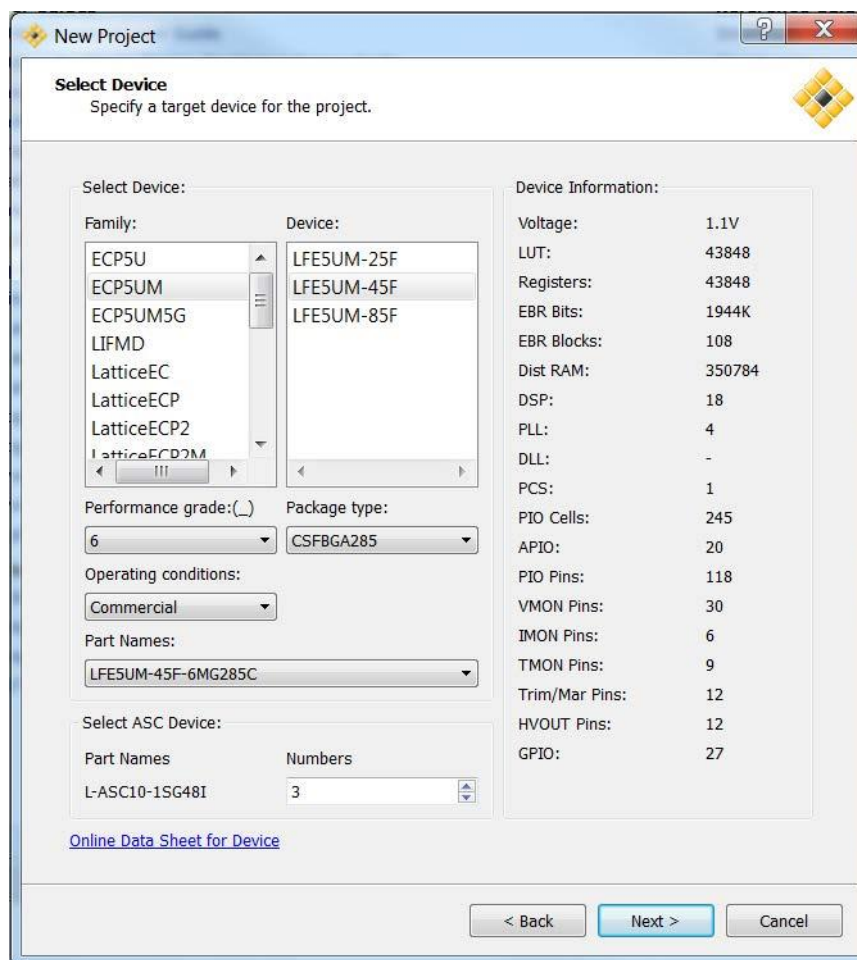


Figure 5.1. New Project Device Selector

5.2. Migrating from MachXO2 + ASC or MachXO3 + ASC to ECP5 + ASC

Existing MachXO2 + ASC or MachXO3 + ASC projects can be migrated to projects based on ECP5 + ASC. Simply double-click the Device Selector icon in the file list view in Diamond. The device selector is highlighted in [Figure 5.2](#) below. This

will open the same dialog window shown in [Figure 5.1](#). Change the device from MachXO2 or MachXO3 to one of the supported ECP5 variants.

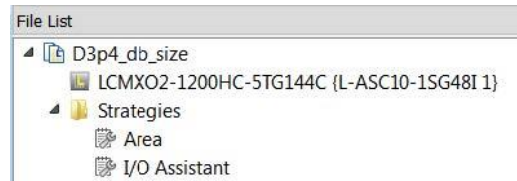


Figure 5.2. Device Selector

After changing the device, Platform Designer will automatically disable unsupported features for ECP5. You may be required to make additional changes to the design in order to fully remove references to the unsupported features. For example, if your project uses the Hot Swap component, Platform Designer will remove it when you migrate to ECP5. This may leave undriven nodes which are referenced by other parts of the code, which have to be corrected before proceeding.

The .lpf file/Spreadsheet view may also contain device specific information (such as Port assignments which do not apply to the new device). In this case you must also update the .lpf file to proceed to the programming file generation step.

5.3. Platform Designer – Unique Feature Summary for ECP5 + ASC

The Platform Designer environment for ECP5 based designs is mostly identical to the environment for MachXO2 or MachXO3 based designs. The views from the tool are listed in [Table 5.1](#), with a brief comment on each section which is unique.

Table 5.1. Unique Feature Summary for ECP5 Solutions

View	Differences for ECP5 Support	Comment
Global	Multiple Differences	Described in the Global View: ASC & Device Options – ECP5 Specific Features section.
Fan Control	None	—
Fault Logger	Multiple Differences	Described in the Fault Logging section.
Hot Swap	Support Not Supported	—
PMBus Adapter	Support Not Supported	—
Current	None	—
Voltage	VID Write Protect Support Not Supported	Described in the Global View: ASC & Device Options – ECP5 Specific Features section.
Temperature	None	—
Ports & Nodes	None	—
Logic	Timer Clock Source – 62.5 kHz only; Different sequence clocking scheme	EFB Prescaler not available in ECP5; Clocking Updates described in the Special Considerations for Designing with ECP5 + ASC in Lattice Diamond section.
Build	None	—

5.4. Global View: ASC & Device Options – ECP5 Specific Features

The Global View settings for ECP5 + ASC designs are mostly identical to the settings for MachXO2 + ASC or MachXO3 + ASC designs. Only the settings which differ from MachXO2 or MachXO3 based designs are discussed in this document.

ASC Options

I2C Base Address (ASC0) – The I2C Base Address is locked to the default base address setting of 1100XXX. The method used to configure the ASC devices in the ECP5 + ASC solution is the ASC Boot IP Component (described earlier in this

document). The ASC Boot only updates the shadow register configuration space of the ASC device (which does not include the I2C Base Address setting area). For this reason, the base address is locked to the default setting.

UES Bits – The UES Bits are unavailable in Platform Designer for ASC devices in the ECP5 + ASC solution. This is due to the same ASC Boot IP restriction mentioned above.

Device Options – Operation Mode

Programming Interface – The programming interface is locked to JTAG in Platform Designer. ECP5 does not support I2C programming. Other programming modes can be configured in Diamond Spreadsheet View – see [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#) for more details.

Background Programming – The Background Programming setting provides two options: JTAG or SPI. For ECP5, background programming refers to the method used to access the external SPI flash memory for programming. [Figure 5.3](#) below shows the JTAG background programming interface while [Figure 5.4](#) shows the SPI background programming interface.

JTAG: This setting is used to preserve the JTAG-to-SPI path through ECP5 for background programming the ECP5 configuration SPI Flash. The JTAG pins are always available, this setting will ENABLE the Master SPI Configuration setting to preserve the SPI pins for configuration on ECP5. This setting cannot be used in conjunction with the Fault Logger Setting Log to Configuration SPI Memory. See the Customized SPI Port in User Mode portion of the Master SPI Modes configuration section in [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#) for more details. This setting is shown in [Figure 5.3](#) below.

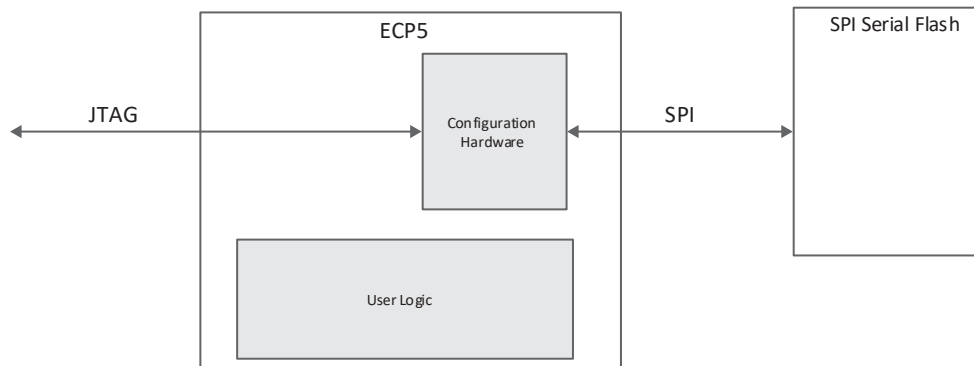


Figure 5.3. SPI Flash Background Programming – JTAG

SPI: This setting will not preserve the JTAG-to-SPI path through ECP5. This setting can be used in conjunction with the Fault Logger Setting Log to Configuration SPI Memory, but will not support SED automatic background reconfiguration. When this setting is selected, the external configuration SPI Flash can only be background updated using the External SPI Memory interface included with the Fault Logger component. This setting is shown in [Figure 5.4](#) below. See the Fault Logging section for full discussion on this mode.

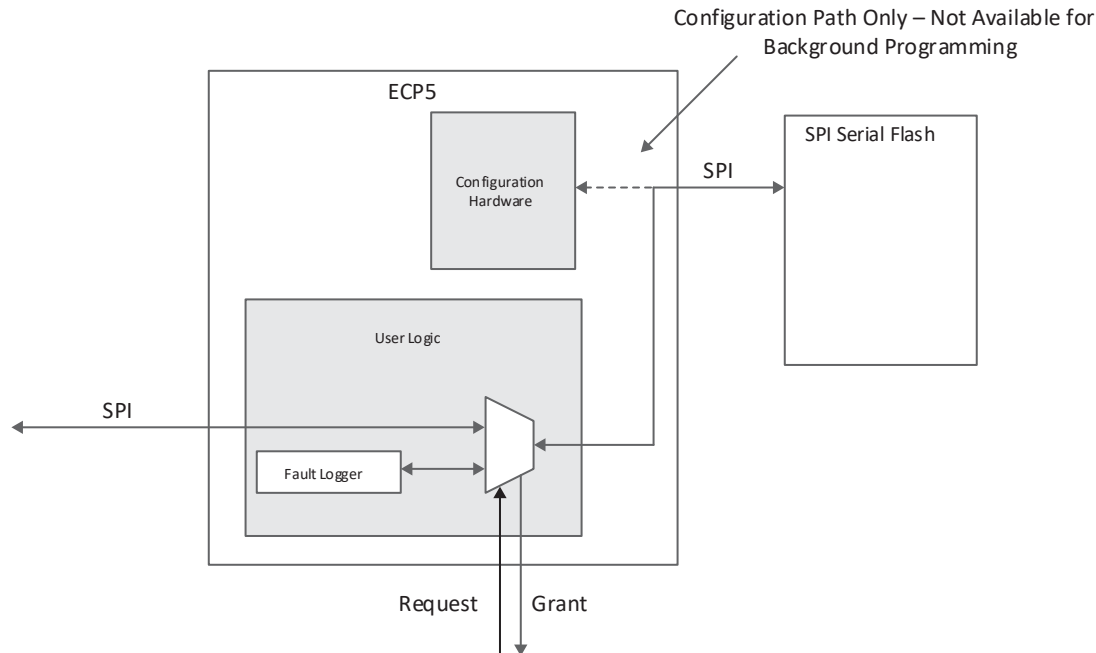


Figure 5.4. SPI Flash Background Programming – SPI

5.5. Device Options – External Connected Components Mode

Boot Mode – In the ECP5 based solution, the Boot Mode setting is locked to Normal. The ECP5 based solution supports Single, Dual, and Multi-Boot, however the Boot Mode is configured using Diamond Deployment tool rather than any setting in Platform Designer. See the ASC Boot section of this document and the [ECP5 sysCONFIG Usage Guide \(FPGA-TN-02039\)](#) for more details on setting up different boot modes in ECP5.

ASC I2C Write Feature – In the ECP5 based solution, the ASC I2C Write Feature is locked to Enabled. This is due to the same ASC Boot restriction (shadow register configuration only) described in the I2C base address section above.

6. I²C/SPI Interface Blocks

6.1. Overview

The design components provided for the ECP5 + ASC solution through the Platform Designer software (VID, Fault Logging, ASC Boot) require either I²C or SPI master blocks. In the MachXO2 or MachXO3 based solutions, these components utilize the EFB, however the same EFB is not available in the ECP5 device family. Instead, Platform Designer will automatically instantiate a soft I²C or SPI interface block, depending on the design.

These blocks will be included for synthesis as .ngo files (binary-format FPGA pre-map) and for simulation as black box components. The interfacing to these blocks is handled entirely by the automatically instantiated IP components from Platform Designer, the user is not required to add any additional logic. These components are based on the hardened IP components included in the ICE40LM device, which are described in [ICE40 SPI/I2C Hardened IP Usage Guide \(FPGA-TN-02010\)](#).

6.2. I²C Slave Addressing Restriction

The I²C interface block generated by Platform Designer is a flexible IP block re-used from previous Lattice design projects. This block supports by default both slave and master operations, although Platform Designer only uses the master function of the I²C interface block. The slave function is still active, using I²C address 0x71. This address must be avoided by other devices on the bus or unexpected behavior may occur.

6.3. Configurable I²C and SPI Clock Rates

The I²C and SPI blocks support configurable clock rates. The clock rates can be updated from their default values via Tcl command (using the Tcl console in Diamond). Table 6.1 below shows the interfaces with their associated Tcl commands and default clock divider values.

Table 6.1. Tcl Commands for I2C and SPI Clock Rate Settings

Block	Tcl Command	Default Value	Comment
I ² C	<code>psb_global set -i2cbr <0x0000-0x03FF></code>	0x04 ¹ or 0x0E ² (400 kHz)	Sets I ² C clock divider according to equation below
SPI	<code>psb_faultlogger set -spibr <0x00-0x3F></code>	0x18 (~0.95 MHz)	Sets SPI clock divider according to equation below

Notes:

1. System Clock = 8 MHz, no SPI Flash Fault Logging Enabled
2. System Clock = 24 MHz, SPI Flash Fault Logging Enabled

The I²C clock frequency is determined by the following setting:

$$\text{I}^2\text{C Clock} = (\text{System Clock}) / (\text{Divider} * 4)$$

The System Clock is set automatically by Platform Designer to either 8 MHz (No SPI Flash Fault Logging Enabled) or 24 MHz (SPI Flash Fault Logging Enabled). Table 6.2 shows the settings for i2cbr which correspond to the common I²C clock rates of 50 kHz, 100 kHz, and 400 kHz.

Table 6.2. I2C Clock Rate Settings vs i2cbr Setting

I ² C Clock Rate	i2cbr Setting – No fault Log Enabled (System Clock = 8 MHz)	i2cbr Setting – Fault Log Enabled (System Clock = 24 MHz)
400 kHz	0x05	0x0F
100 kHz	0x14	0x3C
50 kHz	0x28	0x78

The SPI clock rate is determined by the following equation:

$$\text{SPI Clock} = (\text{System Clock}) / (\text{Divider} + 1)$$

The System Clock is set to 24 MHz (using PLL1 of ECP5) whenever SPI Flash Fault Logging is enabled. Table 6.3 shows some common SPI Clock rates and their associated spibr divider setting.

Table 6.3. Common SPI Clock Settings vs spibr Setting

SPI Clock Rate	Spibr divider setting
24 MHz	0x00
12 MHz	0x01
8 MHz	0x02
4 MHz	0x05
0.95 MHz	0x18 (default)

Note that the spibr setting affects the SPI Flash Fault Logging clock rate and not the Master SPI configuration clock rate. The configuration clock rate is only set through the spreadsheet view global setting, as described in the configuration section of this document.

The current I²C clock rate setting can be read back using the following Tcl Command `psb_global get -i2cbr`. The current SPI clock rate setting can be read back with `psb_faultlogger get -spibr`.

7. Fault Logging

The fault logging component has been updated significantly for the ECP5 + ASC system solution. The key updates for the full-featured Fault Log component as compared to the MachXO2 or MachXO3 based hardware management system include the following:

- Memory type restricted to external SPI (no User Flash Memory available in ECP5)
- Enhanced Timestamp Features
- Enhanced Record Counter/Fault Location Management

The following section details the software configuration and hardware connections for the fault log component.

7.1. Fault Log Software Configuration

Click on the Fault Logger Component (shown in [Figure 7.1](#)) to display the V, I, T and Full Featured configuration views. Click on the corresponding tab at the top of the view to select between the V, I, T and Full Featured configuration views.

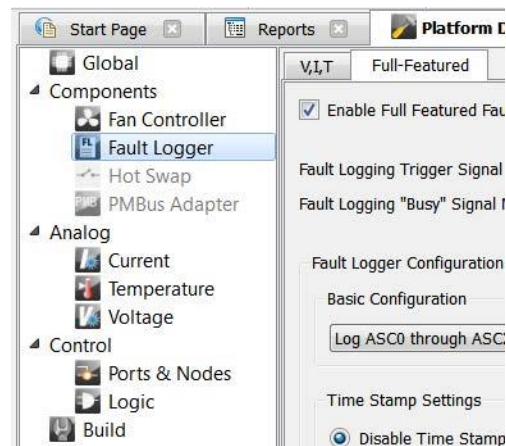


Figure 7.1. Fault Logger Component Selection

7.2. V, I, T Fault Log Configuration

Check Box: Enable V, I, T Fault Log/User Tag Operation Disabled

This check box is located in the upper left of the view, as seen in [Figure 7.2](#). Check this box to enable V, I, T fault logging. The V, I, T and Full-Featured fault log are enabled independently, and can be used at the same time. The V, I, T fault log will record the status of the analog sense and control signals within the ASC device at the time the trigger signal becomes true.

Drop Down List: Fault Logging Trigger Signal Name

Click to open the list, then scroll up or down to locate and select the user signal to trigger a fault log. The trigger signal is active high and can be a node, port, GPIO, or analog sense signal.

Basic Configuration

Two radio buttons are available to select how the V, I, T fault log will be stored. The choice is either in volatile memory (SRAM) or non-volatile memory (EEPROM). See the [L-ASC10 Data Sheet \(FPGA-DS-02038\)](#) for more details on the log memory.

User Signals to Log

This section of the configuration view is a drag-and-drop interface where signals can be dragged from the LOGIC SIGNAL POOL and dropped into the desired location of the respective ASC. The filter under the LOGIC SIGNAL POOL can be used to limit the number of signals shown. For example, in [Figure 7.2](#) only the group shutdown signals are

shown. Note the ? single character wild card is used so all group shutdown signals are listed. Bit locations without a signal assigned will be recorded with a zero when the fault log is triggered.

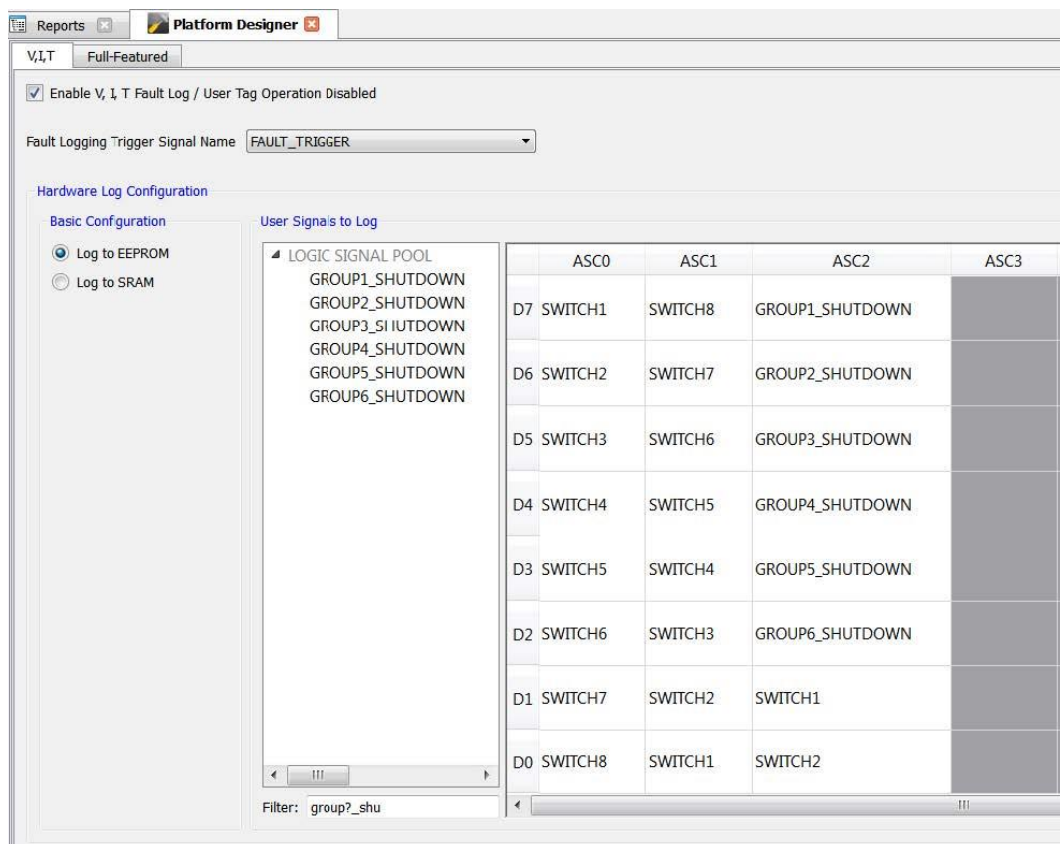


Figure 7.2. V, I, T Fault Log Configuration

7.3. Full Featured Fault Log Configuration

Check Box: Enable Full Featured Fault Log

This check box is located in the upper left of the view, as seen in Figure 7.3. Check this box to enable full featured fault logging. The V, I, T and Full-Featured fault log are enabled independently, and can be used at the same time. The full featured fault log will record the status of the analog sense and control signals to an external SPI device at the time the trigger signal becomes true.

Drop Down List: Fault Logging Trigger Signal Name

Click to open the list, then scroll up or down to locate and select the user signal to trigger a fault log. The trigger signal is active high and can be a node, port, GPIO, or analog sense signal.

Drop Down List: Fault Logging Busy Signal Name

Click to open the list, then scroll up or down to locate and select the user signal to flag when the fault logger is busy writing to the SPI memory. The busy signal is active high and can be a node, port, or GPIO. This signal is provided so that the logic design or external circuitry can monitor the fault logger activity. It must be assigned in the design to ensure the fault log functions properly.

Basic Configuration

Click on the drop down list that is under Basic Configuration to select how many of the ASC devices in the design from which to log signals from. The options are limited to consecutive ASCs starting with ASC0. For example in a design with three ASC devices the options would be as follows:

- Log ASC0 Only
- Log ASC0 through ASC1
- Log ASC0 through ASC2

Time Stamp Settings

Three radio buttons are provided to configure the fault log time stamp.

If the *Disable Time Stamp* is selected then no time stamp information will be recorded in the fault log and time stamp ports and nodes will be removed from the design (if one of the other time stamps was previously used).

If *Internally Generated Time Stamp* is selected then a 32 bit counter is included in the design that counts up once every second the system is powered up. A global reset or power cycle will restart the counter at zero. The accuracy of the time base is about 10%. This internally generated time stamp is a cost effective method of determining approximately when the fault occurred.

If a different format or accuracy is desired, then the *Externally Assigned Time Stamp* should be selected. When this is selected a 32-bit group of signals called FL_TIMESTAMP will be added to the Nodes tab of the Ports & Nodes View for external assignment from a separate timebase or clock.

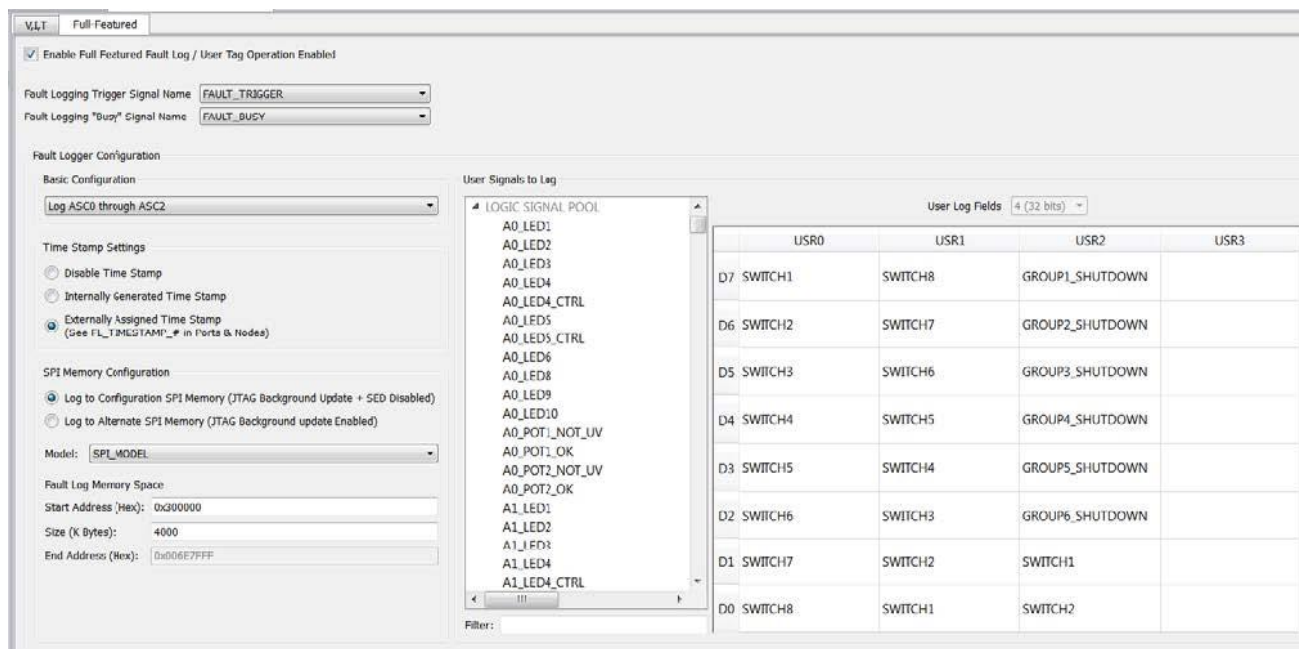


Figure 7.3. Full Featured Fault Log Configuration View

SPI Memory Configuration

Two radio buttons are provided to select the type of SPI memory connection used.

If only a single external SPI memory is used in the design then the *Log to Configuration SPI Memory...* should be selected. When using fault logging and a single SPI memory, the JTAG background programming and SED features will be disabled.

A second external SPI memory can be used for fault logging; in this case the *Log to Alternate SPI Memory...* should be selected. Then JTAG background programming and SED features can be used in the design.

This setting works in conjunction with the Background Programming setting described in the Global View update section. Invalid combinations between these settings will be flagged during the project DRC step.

Drop Down List: Model

Click on this list to select or edit the SPI model used in the design. For most designs the default SPI_MODEL has the correct flash opcodes to read and record the faults into the SPI memory.

Fault Log Memory Space

Two edit fields are provided to enter the Starting Address (Hex) of the fault log memory and the Size (K-Bytes) of the fault log memory. The End Address (Hex) is calculated and displayed by the software. When logging to the configuration memory (only one SPI memory used in the design) the start address should be above the space reserved for the boot or dual-boot image. [Figure 7.3](#) shows the default start address (0x300000) for an ECP5 – 45 device in a memory map. When using a secondary SPI memory that is dedicated to fault logging the start address can be zero (0x000000).

User Signals to Log

This section of the configuration view is a drag-and-drop interface where signals can be dragged from the LOGIC SIGNAL POOL and dropped into the desired bit location of one of the four user bytes. The filter under the LOGIC SIGNAL POOL can be used to limit the number of signals shown. For example, in [Figure 7.3](#) only the SWITCH signals are shown by using sw as the filter. The four user bytes are always included in the recorded fault logs. Bit locations without a signal assigned will be recorded with a zero when the fault log is triggered.

Fault Log Record Counter

The Full-Featured Fault Log component includes a 16-bit fault record counter. This record counter is set each time the external SPI memory is scanned (after power-up and following an external SPI master access). The record counter will be incremented each time a fault is triggered and logged. The record counter is made available to the design through a 16-bit node group called FL_RECORDCOUNTER. This group is automatically added to the nodes tab of the ports and nodes view whenever the Full-Featured Fault Log component is enabled. This group can be passed to other logic blocks or logged in the user signal area. For details on the SPI memory scan behavior, see the hardware connections section below.

Fault Log Record Storage

Each fault record will occupy half a page or 128 bytes. This organization allows easy indexing and prevents writing fault records on a page boundary. This is illustrated in [Figure 7.4](#) where each fault record is separated by 0x80 bytes.

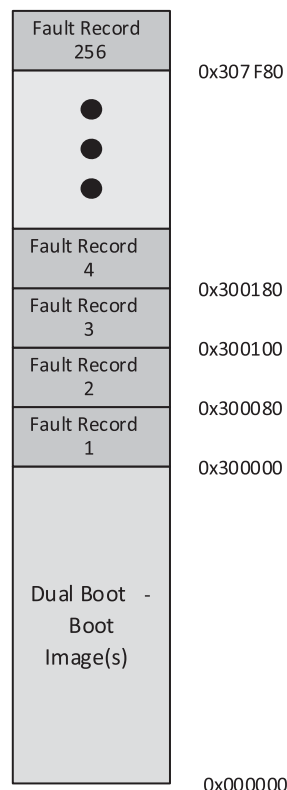


Figure 7.4. SPI Memory Map for Single SPI Memory

Fault Log Record Format

The actual number of bytes used within the fault record depends on the number of ASCs in the design and if a time stamp is included or not. Each fault record begins with the 0x3C byte to differentiate between erased memory and a valid fault record. The second byte in the fault record contains the actual number of bytes recorded (Size). The third byte is reserved for future use and serves as a placeholder so the following bytes are in a similar location as they appear within the ASC device. The bytes for the ASCs are recorded next, followed by the User Signals, and optional time stamp. The last byte contains an end of record byte (0x2A).

Three tables are shown to illustrate how the organization of the fault record depends on the design parameters.

[Table 7.1](#) shows a fault record bit-map for a single ASC design with a timestamp. [Table 7.2](#) shows a fault record bitmap for a single ASC design without a timestamp, and [Table 7.3](#) shows a fault record bit-map for a design with three ASCs and includes a timestamp. The extra column in [Table 7.3](#) is not part of the fault log, rather it is included only to illustrate the ASC order.

Table 7.1. Detailed Fault Record for Single ASC with Time Stamp

BYTE	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	1	1	1	1	0	0
1	Size[7]	Size[6]	Size[5]	Size[4]	Size[3]	Size[2]	Size[1]	Size[0]
2	Reserved	Reserved	0	0	1	Reserved	1	Reserved
3	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4
4	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B
5	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B
6	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B
7	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B
8	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1
9	USR0[7]	USR0[6]	USR0[5]	USR0[4]	USR0[3]	USR0[2]	USR0[1]	USR0[0]
10	USR1[7]	USR1[6]	USR1[5]	USR1[4]	USR1[3]	USR1[2]	USR1[1]	USR1[0]
11	USR2[7]	USR2[6]	USR2[5]	USR2[4]	USR2[3]	USR2[2]	USR2[1]	USR2[0]
12	USR3[7]	USR3[6]	USR3[5]	USR3[4]	USR3[3]	USR3[2]	USR3[1]	USR3[0]
13	Timer[7]	Timer[6]	Timer[5]	Timer[4]	Timer[3]	Timer[2]	Timer[1]	Timer[0]
14	Timer[15]	Timer[14]	Timer[13]	Timer[12]	Timer[11]	Timer[10]	Timer[9]	Timer[8]
15	Timer[23]	Timer[22]	Timer[21]	Timer[20]	Timer[19]	Timer[18]	Timer[17]	Timer[16]
16	Timer[31]	Timer[30]	Timer[29]	Timer[28]	Timer[27]	Timer[26]	Timer[25]	Timer[24]
17	0	0	1	0	1	0	1	0

Table 7.2. Detailed Fault Record for Single ASC without Time Stamp

BYTE	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	1	1	1	1	0	0
1	Size[7]	Size[6]	Size[5]	Size[4]	Size[3]	Size[2]	Size[1]	Size[0]
2	Reserved	Reserved	0	0	1	Reserved	1	Reserved
3	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4
4	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B
5	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B
6	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B
7	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B
8	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1
9	USR0[7]	USR0[6]	USR0[5]	USR0[4]	USR0[3]	USR0[2]	USR0[1]	USR0[0]
10	USR1[7]	USR1[6]	USR1[5]	USR1[4]	USR1[3]	USR1[2]	USR1[1]	USR1[0]
11	USR2[7]	USR2[6]	USR2[5]	USR2[4]	USR2[3]	USR2[2]	USR2[1]	USR2[0]
12	USR3[7]	USR3[6]	USR3[5]	USR3[4]	USR3[3]	USR3[2]	USR3[1]	USR3[0]
13	0	0	1	0	1	0	1	0

Table 7.3. Detailed Fault Record for Three ASCs with Time Stamp.

BYTE	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ASC
0	0	0	1	1	1	1	0	0	n/a
1	Size[7]	Size[6]	Size[5]	Size[4]	Size[3]	Size[2]	Size[1]	Size[0]	n/a
2	Reserved	Reserved	0	0	1	Reserved	1	Reserved	0
3	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4	0
4	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B	0
5	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B	0
6	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B	0
7	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B	0
8	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1	0
9	Reserved	Reserved	0	0	1	Reserved	1	Reserved	1
10	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4	1
11	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B	1
12	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B	1
13	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B	1
14	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B	1
15	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1	1
16	Reserved	Reserved	0	0	1	Reserved	1	Reserved	2
17	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4	2
18	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B	2
19	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B	2
20	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B	2
21	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B	2
22	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1	2
23	USR0[7]	USR0[6]	USR0[5]	USR0[4]	USR0[3]	USR0[2]	USR0[1]	USR0[0]	n/a
24	USR1[7]	USR1[6]	USR1[5]	USR1[4]	USR1[3]	USR1[2]	USR1[1]	USR1[0]	n/a
25	USR2[7]	USR2[6]	USR2[5]	USR2[4]	USR2[3]	USR2[2]	USR2[1]	USR2[0]	n/a
26	USR3[7]	USR3[6]	USR3[5]	USR3[4]	USR3[3]	USR3[2]	USR3[1]	USR3[0]	n/a
27	Timer[7]	Timer[6]	Timer[5]	Timer[4]	Timer[3]	Timer[2]	Timer[1]	Timer[0]	n/a
28	Timer[15]	Timer[14]	Timer[13]	Timer[12]	Timer[11]	Timer[10]	Timer[9]	Timer[8]	n/a
29	Timer[23]	Timer[22]	Timer[21]	Timer[20]	Timer[19]	Timer[18]	Timer[17]	Timer[16]	n/a
30	Timer[31]	Timer[30]	Timer[29]	Timer[28]	Timer[27]	Timer[26]	Timer[25]	Timer[24]	n/a
31	0	0	1	0	1	0	1	0	n/a

The maximum number of ASC devices in a design is eight. [Table 7.4](#) is a byte map showing the location of the ASC data for a maximum sized design with the time stamp.

Table 7.4. Maximum Fault Record Byte Map

Byte	Description
0	Fault Flag (0x3C)
1	Number of bytes used in Fault Record
2	ASC0 – Reserved (place holder)
3	ASC0 – GPIO Status
4	ASC0 – GPIO/HVOUT Status
5	ASC0 – VMON Status
6	ASC0 – VMON Status
7	ASC0 – VMON Status
8	ASC0 – TMON Status
9	ASC1 – Reserved (place holder)
10	ASC1 – GPIO Status
11	ASC1 – GPIO/HVOUT Status
12	ASC1 – VMON Status
13	ASC1 – VMON Status
14	ASC1 – VMON Status
15	ASC1 – TMON Status
16	ASC2 – Reserved (place holder)
17	ASC2 – GPIO Status
18	ASC2 – GPIO/HVOUT Status
19	ASC2 – VMON Status
20	ASC2 – VMON Status
21	ASC2 – VMON Status
22	ASC2 – TMON Status
23	ASC3 – Reserved (place holder)
24	ASC3 – GPIO Status
25	ASC3 – GPIO/HVOUT Status
26	ASC3 – VMON Status
27	ASC3 – VMON Status
28	ASC3 – VMON Status
29	ASC3 – TMON Status
30	ASC4 – Reserved (place holder)
31	ASC4 – GPIO Status
32	ASC4 – GPIO/HVOUT Status
33	ASC4 – VMON Status
34	ASC4 – VMON Status
35	ASC4 – VMON Status
36	ASC4 – TMON Status
37	ASC5 – Reserved (place holder)
38	ASC5 – GPIO Status
39	ASC5 – GPIO/HVOUT Status
40	ASC5 – VMON Status
41	ASC5 – VMON Status
42	ASC5 – VMON Status
43	ASC5 – TMON Status
44	ASC6 – Reserved (place holder)
45	ASC6 – GPIO Status
46	ASC6 – GPIO/HVOUT Status
47	ASC6 – VMON Status

Byte	Description
48	ASC6 – VMON Status
49	ASC6 – VMON Status
50	ASC6 – TMON Status
51	ASC7 – Reserved (place holder)
52	ASC7 – GPIO Status
53	ASC7 – GPIO/HVOUT Status
54	ASC7 – VMON Status
55	ASC7 – VMON Status
56	ASC7 – VMON Status
57	ASC7 – TMON Status
58	User Byte 0
59	User Byte 1
60	User Byte 2
61	User Byte 3
62	Time Stamp Byte 0
63	Time Stamp Byte 1
64	Time Stamp Byte 2
65	Time Stamp Byte 3
66	End of Record Flag (0x2A)

7.4. Full Featured Fault Log Hardware Connections

For the full featured fault logging an external SPI memory has to be connected to the FPGA I/O pins. Furthermore, an external SPI master has to be connected in order to read the fault logs out of the SPI memory and to erase the fault logs when they are no longer needed. This section describes the two different connections; a single SPI memory and two SPI memories.

Single SPI Memory Connections

Figure 7.5 illustrates the system connections when using a single SPI device both for FPGA configuration and for fault logging. When the full featured fault logging is enabled, Platform Designer will add three groups of port connections to the design:

- Six faultloggerextspi_x signals to connect with the external SPI master.
- Three faultloggerflash_y signals to connect with the external SPI memory.
- One faultloggerusp_memerr_o error signal.

When the external SPI master needs control of the SPI memory, it must first raise the faultloggerextspi_req signal. If the Platform Manager is not actively recording a fault, then the faultloggerextspi_grant will be true; otherwise the external SPI master must wait for the grant signal to be true before reading or erasing the SPI memory. The fault logger will drive the faultloggerusp_memerr_o true if it encounters an error such as finding something besides 0xFF or 0x3C at the beginning of a record (when scanning the SPI to find the next available record location) or reaching the maximum number of fault records. The external SPI master (or internal logic) can monitor this signal to let the system know if a problem has occurred. The other faultloggerextspi_x signals are the standard MOSI, MISO, CLK, and CS. The last signal to discuss on the left side of the diagram in Figure 7.5 is the user defined signal FAULT_BUSY_P. The optional signals are shown with dashed lines. The FAULT_BUSY_P can optionally be connected to an interrupt input of the external SPI master, so it knows when a fault has occurred. FAULT_BUSY_P can also be connected to internal logic, to notify other logic blocks that a fault is in progress or has occurred.

The faultloggerflash_y signals connect to the configuration SPI memory. They must be mapped in the spreadsheet tool to the default SPI pins; CS, MOSI, and MISO. The SPI memory MCLK pin is mapped using the special User Clock macro for the ECP5 device – this is described in ECP5 sysCONFIG Usage Guide (FPGA-TN-02039).

When the ECP5 powers up, it will read the configuration from the SPI memory and load the fault logger IP, arbitration logic, and mux into the fabric. After power up configuration, the SPI connections are passed directly to the mux.

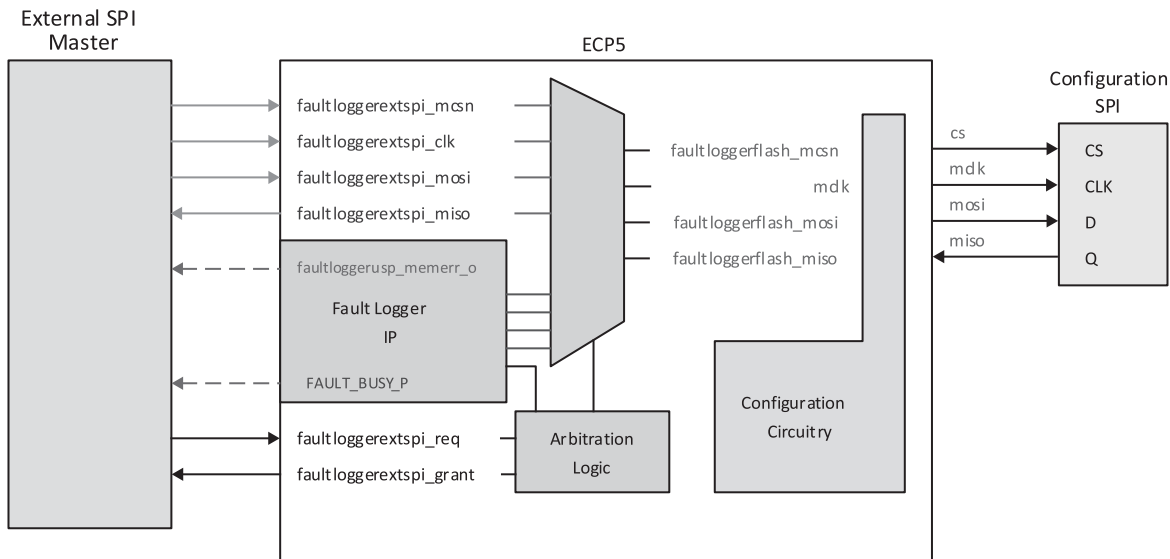


Figure 7.5. Hardware Connections for Design with Configuration SPI Memory for Fault Logging

Alternate SPI Memory Connections

Figure 7.6 illustrates the connections when using an alternate SPI memory to store the fault logs. The key difference between Figure 7.5 and Figure 7.6 is the faultloggerflash_y signals are mapped to four generic I/O pins to the alternate memory (using the spreadsheet tool). While the configuration SPI is still connected to the default SPI pins; CS, MCLK, MOSI, and MISO.

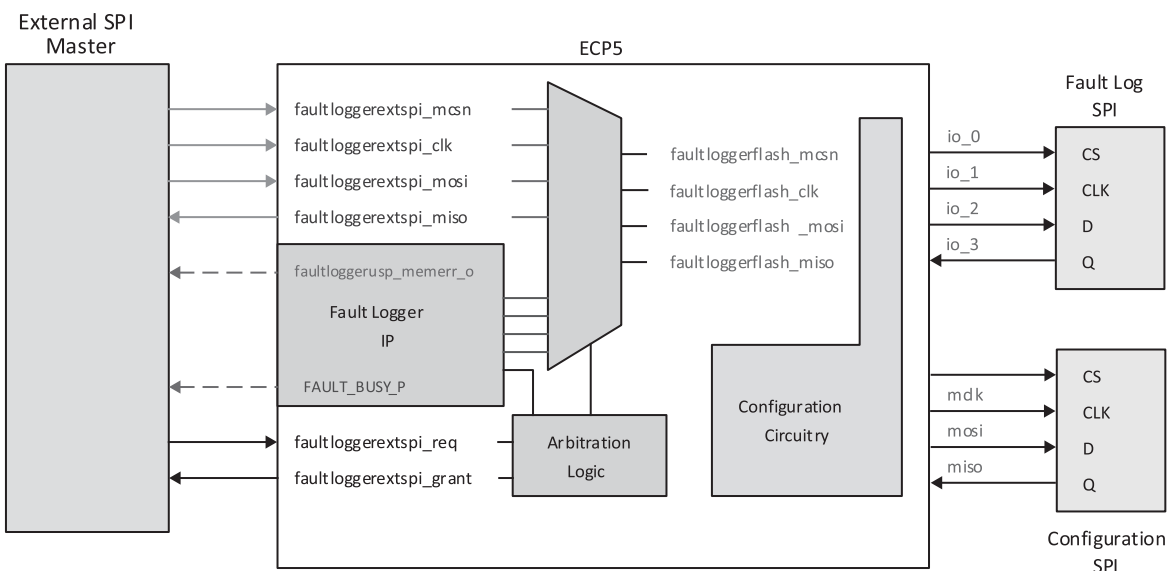


Figure 7.6. Hardware Connections for Design with Alternate SPI Memory for Fault Logging

Full Featured Fault Log – Accessing the SPI Memory

As described in the hardware connections section, the Full Featured Fault Log provides an interface for reading and erasing the external SPI memory from an external SPI master. The arbitration logic block is provided to ensure the fault log IP and external SPI master do not interfere with each other.

In order for the external SPI master to read or erase the fault log area of the SPI memory, it must set the `faultloggerextspi_req` = 1. As soon as the fault log component is not recording a fault, it will set the `faultloggerextspi_grant` = 1. During this time, no new faults will be recorded, even if the `fault_trigger` signal is set to 1. (A single fault log event will be snapshotted in volatile memory if a trigger occurs while `faultloggerextspi_grant` = 1. This log will be written out the next time `faultloggerextspi_grant` = 0.)

Once `faultloggerextspi_grant` = 1, the external SPI master may read or erase the SPI memory. After the external SPI master has completed the memory access commands, it should set `faultloggerextspi_req` = 0 (see next section for special considerations related to erase commands). The arbitration logic will then set `faultloggerextspi_grant` = 0.

Whenever `faultloggerextspi_grant` transitions from 1 to 0, the fault logger IP will perform a scan of the SPI memory, to determine if an erase operation has been performed. After the scan, the fault logger IP will reset the fault address pointer to the first memory location reading 0xFF (erased) versus 0x3C (fault log present). The fault logger IP will also reset the record counter register to match the current address location. If the scan reads any value other than 0xFF or 0x3C, the `faultloggerusp_memerr_o` signal will be set to 1. The same scan operation is performed by the fault logger IP following a power-on or global reset event. If the `faultloggerextspi_req` = 1 following reset, the fault logger IP will wait to scan the memory, and the record counter will be held to 0, until the `faultloggerextspi_grant` transitions from 1 to 0.

SPI Memory Erase Operation – Safe Handling

When the SPI master issues an erase command, it must poll the SPI memory status register to know when the operation is complete before releasing the request (`faultloggerextspi_req` = 0). Otherwise the fault logger will scan the SPI memory during the erase operation and may read back invalid data and set the error signal (`faultloggerusp_memerr_o` = 1) and subsequent faults will not be recorded.

The external SPI master must also take care to erase the entire available fault logging area of the SPI memory. The fault logger IP assumes that there is empty memory space between the last recorded/scanned fault and the end of the assigned fault logging area. The IP does not read the SPI memory before every trigger operation. If the external SPI master does not complete a full erase of the fault logging area, undefined behavior will occur that may result in data corruption in the SPI memory.

8. Platform Manager 2 I²C GUI

The Platform Manager 2 I²C GUI (also known as PowerDebug) can be used to read out measurements and settings from the ASC device over I²C. The GUI was originally designed to work with the Platform Manager 2 Evaluation Board and Demo Program. In order for the GUI to be used with the ECP5 + ASC development kit (based on the ECP5 Versa Board and the ASC Evaluation Boards) the ECP5 configuration must contain I²C passthrough logic and an additional set of external connections must be made.

The I²C passthrough logic provided in the ECP5 + ASC demo design uses signals `EXPCON_IO0` and `EXPCOON_IO2` from J1 of the ECP5 + ASC Bridge Board as SCL and SDA respectively. These signals need to be connected externally to the SCL and SDA test points on one of the ASC Evaluation Boards used in the development kit. This connection is shown in [Figure 8.1](#) below.

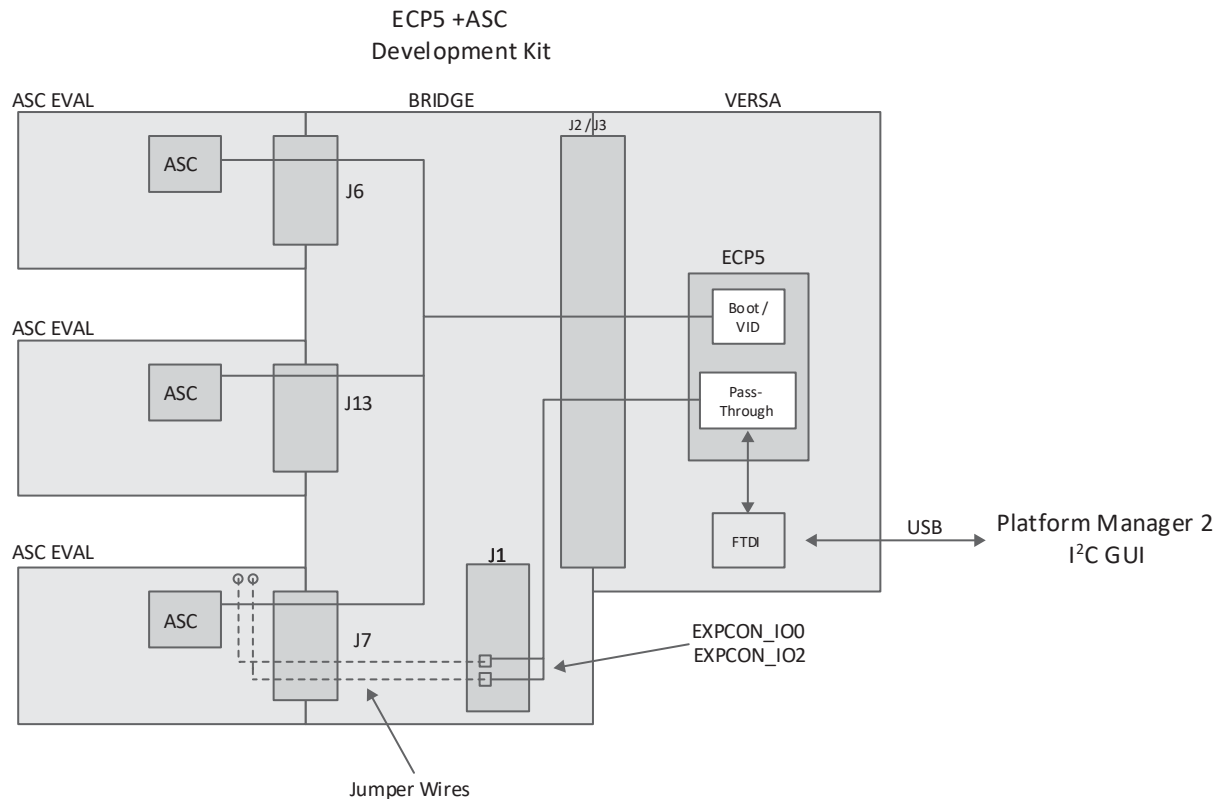


Figure 8.1. Passthrough Connection to Support Platform Manager 2 I²C GUI

9. Special Considerations for Designing with ECP5 + ASC in Lattice Diamond

In addition to the updated system features described in this technical note, there are several additional considerations to be aware of when designing with the ECP5 + ASC in Diamond. These considerations are primarily driven by the increase in logical resources and I/O count when working with the ECP5 device. The increase in overall size of the FPGA generates potential issues related to timing and simulation which should be reviewed during the design process. Three key areas that should be reviewed include:

1. Use of LGB_Clock or PLD_Clock in imported HDL
2. Place & Route Trace Evaluation – Potential Need for Hold Time Optimization
3. Post-Map Simulation – Required Test Bench Update

9.1. Use of LGB_Clock or PLD_Clock in Imported HDL

The ECP5 + ASC solution relies on a set of clock signals generated by an automatically included IP module – the Clock & Reset module. This module generates several clocks from the ASC0_CLK signal (8 MHz) as well as several reset signals. The clock signals are shown in Table 9.1 below.

Table 9.1. ECP5 + ASC Clocks

Clock Net	Frequency	Source	Comment
ASC0_CLK	8 MHz	ASC0 – ASCCLK Pin	Source clock for ECP5 + ASC System
SYS_Clock	8 MHz	ASC0 – ASCCLK Pin	Same as ASC0_CLK signal
PLD_Clock	250 kHz	Divided down SYS_CLK	Divider will induce delay between SYS_CLK and PLD_CLK edges
LGB_Clock	62.5 kHz	Divided down SYS_CLK	Divider will induce delay between SYS_CLK and LGB_CLK edges
PLL_Clk	24 MHz	PLL0 from SYS_CLK	Only included with Full-Featured Fault Log is enabled in the design

Care must be taken when using the generated clock signals in user hdl files. Specific attention should be paid to any modules which will use both the SYS_Clock and the PLD_Clock or LGB_Clock. Hold time violations can occur when both clocks are used together.

The HDL code generated by the Logic – Sequence & Supervisory tabs of Platform Designer has been updated for the ECP5 based solution. Specifically, the clocking scheme has been simplified to operate on a single clock domain (SYS_Clock), using an enable signal generated by the Clock & Reset module for state machine and D Flip-Flop transitions. This removes the hold time violation concern from the generated HDL code for the ECP5 solution. The functional behavior of the sequence and supervisory code is identical to the behavior of the MachXO2 solution.

9.2. Place & Route Trace Evaluation – Potential Need for Hold Time Optimization

Place & Route Trace Analysis is recommended for customers designing with the ECP5 + ASC system. Figure 9.1 below shows the Diamond Process View, where Place & Route Trace can be enabled.

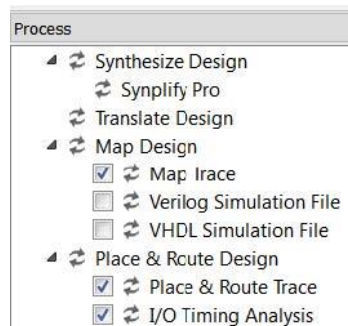


Figure 9.1. Enabling Place & Route Trace in the Diamond Process View

The ECP5 + ASC system includes soft IP for I²C/SPI interface (as described in the section on I²C/SPI interface blocks). These interface blocks include several circuits which are asynchronous in nature and may generate hold time violations during place & route. Upon completion of the Place & Route Design step, the Place & Route Trace report should be evaluated for hold time violations. The Hold area of the Place & Route Trace report is shown in the report tree in Figure 9.2.

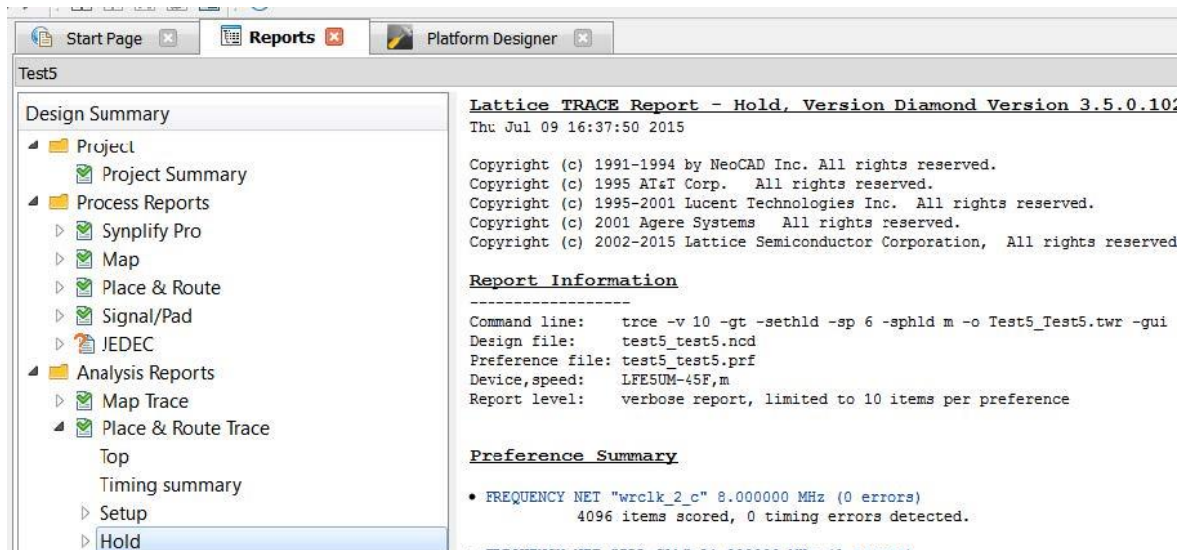


Figure 9.2. Place & Route Trace – Hold Report

The preference summary area will display any clock preferences with hold errors in red. The place & route tool will by default perform a standard level of optimization to remove hold violations. This may not be sufficient to remove the

reported violations primarily associated with the I²C/SPI interface blocks. In order to completely remove the hold violations, the hold time optimization may need to be set to a higher level. It is recommended to update this setting after completing place & route and evaluating the trace report.

To update, the setting, double-click the Strategy1 icon in the File List, as shown in Figure 9.3. Under the *Place & Route Design* menu (also shown in Figure 9.4), the Command Line Options must be updated to increase the optimization level for hold time.

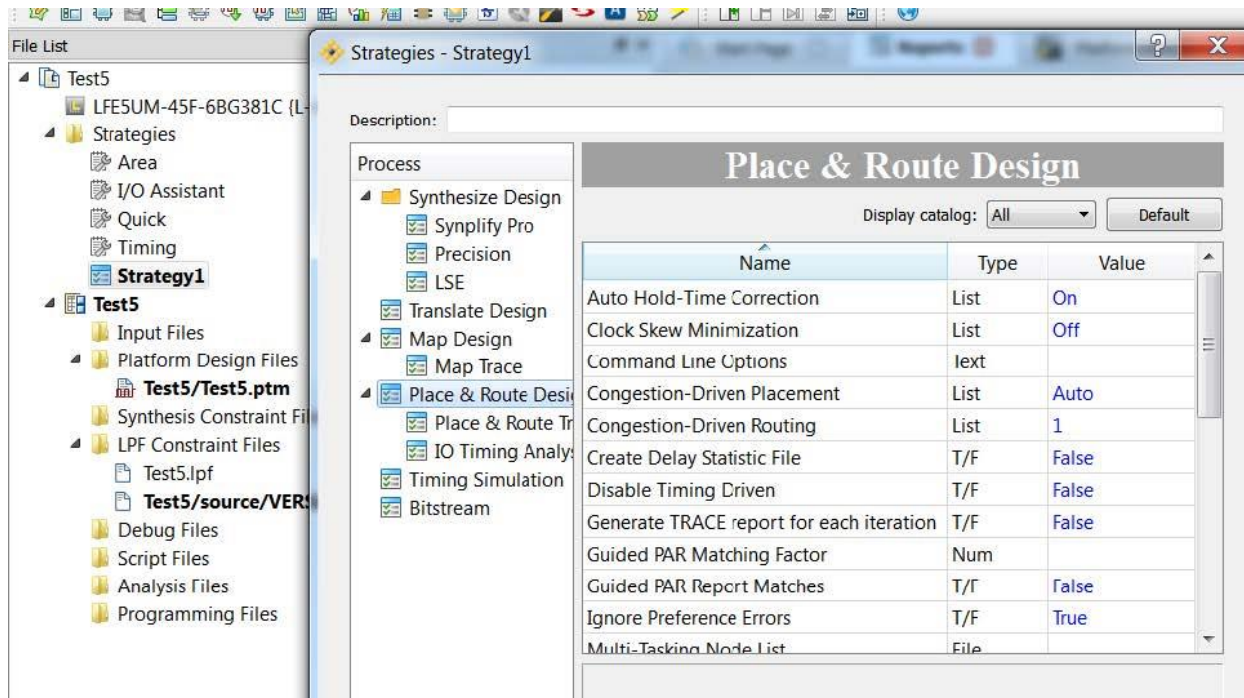


Figure 9.3. Place & Route Strategy Settings

The command line option used for improving the hold optimization is:

`"-exp parHold=2:parHoldLimit=5000"`

This is shown in Figure 9.4. The HoldLimit number may be reduced, depending on the number of errors observed in the Place & Route Trace Report. After the command line option has been updated, Place & Route Design should be repeated. The Hold area of the Place & Route Trace Report should show no errors following the optimization.

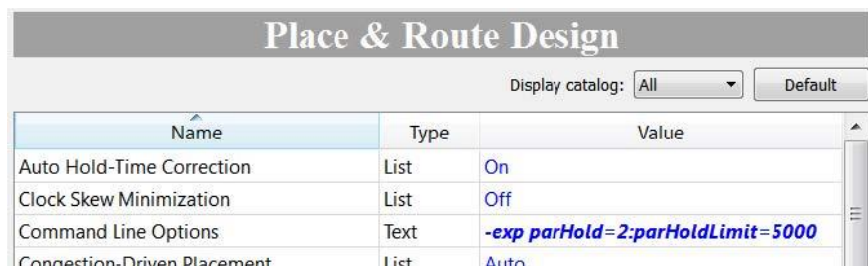


Figure 9.4. Command Line Setting for Hold Time Optimization

9.3. Post-Map Simulation – Test Bench Update

Platform Designer supports the automatic generation of a simulation test bench file including both the ECP5 design and digital models of the ASC devices. The generate stimulus button in Platform Designer is used to generate two types of files: `project_name_design` and `project_name_test`.

There are two versions of these files, the rtl version and the post-map version (called `_post`). The `_post` files can be used with exported post-map Verilog (generated in the Diamond process view) to perform post map simulation of the ECP5 + ASC system.

The post-map simulation may show a glitch in the I²C traffic (shown in Figure 9.5, this is during the initial transaction of the ASC Boot IP). This glitch will cause the simulation to fail, as the ASC model will interpret this glitch as an invalid condition and NACK the corresponding transaction.

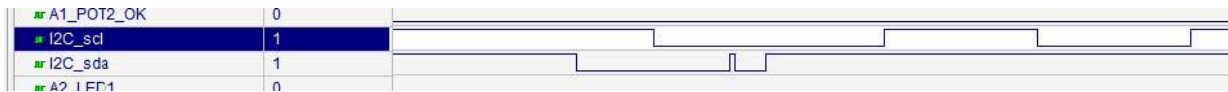


Figure 9.5. SDA Glitch in Post-Map Simulation

This post-map simulation behavior does not match the characterized ECP5 + ASC system behavior in hardware. There are two reasons why this behavior should be ignored and suppressed if present in the post-map simulation.

#1 – The glitch shown in simulation is ~40 ns. The simulation shows sharp digital transitions of the glitch, however in hardware, the rising edges are controlled by the I2C pull-up resistors and are not as sharp. Oscilloscope observation shows that the glitch only rises to about 0.5 V, well below V_{IH} for the ASC hardware.

#2 – The ASC device includes an analog filter at the SCL and SDA pins, designed to reject glitches of up to 50 ns. This analog filter is not a part of the digital simulation model. Even if the glitch did rise to V_{IH} , it would be rejected by the analog glitch filter.

It may be necessary to add code to the `project_name_design_post.v` file to suppress this glitch at the ASC devices. The following section of Verilog code may be added to the file in order to support post-map simulation.

This line should be added to the top of the `_design_post.v` file.

```
`timescale 1ns/100ps
```

The following code can be added before all instantiations in the `_design_post.v` file.

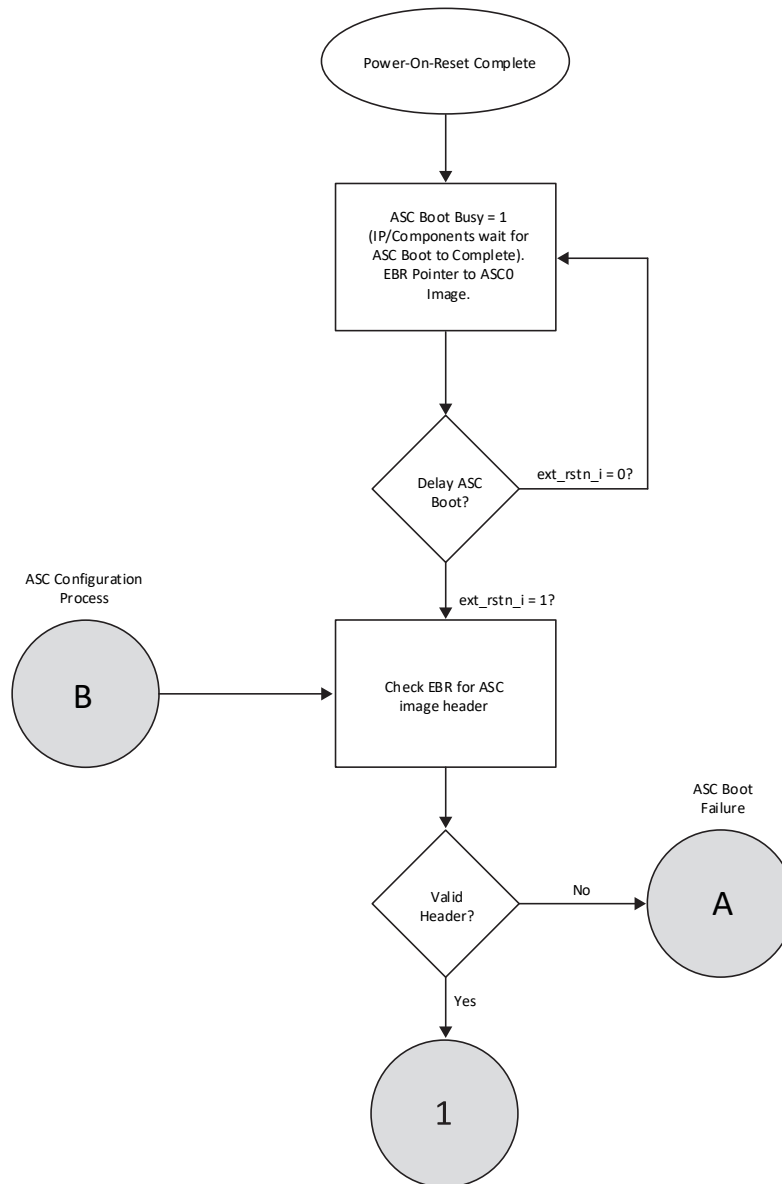
Listing 1. Glitch Suppression Code for Post-Map Simulation.

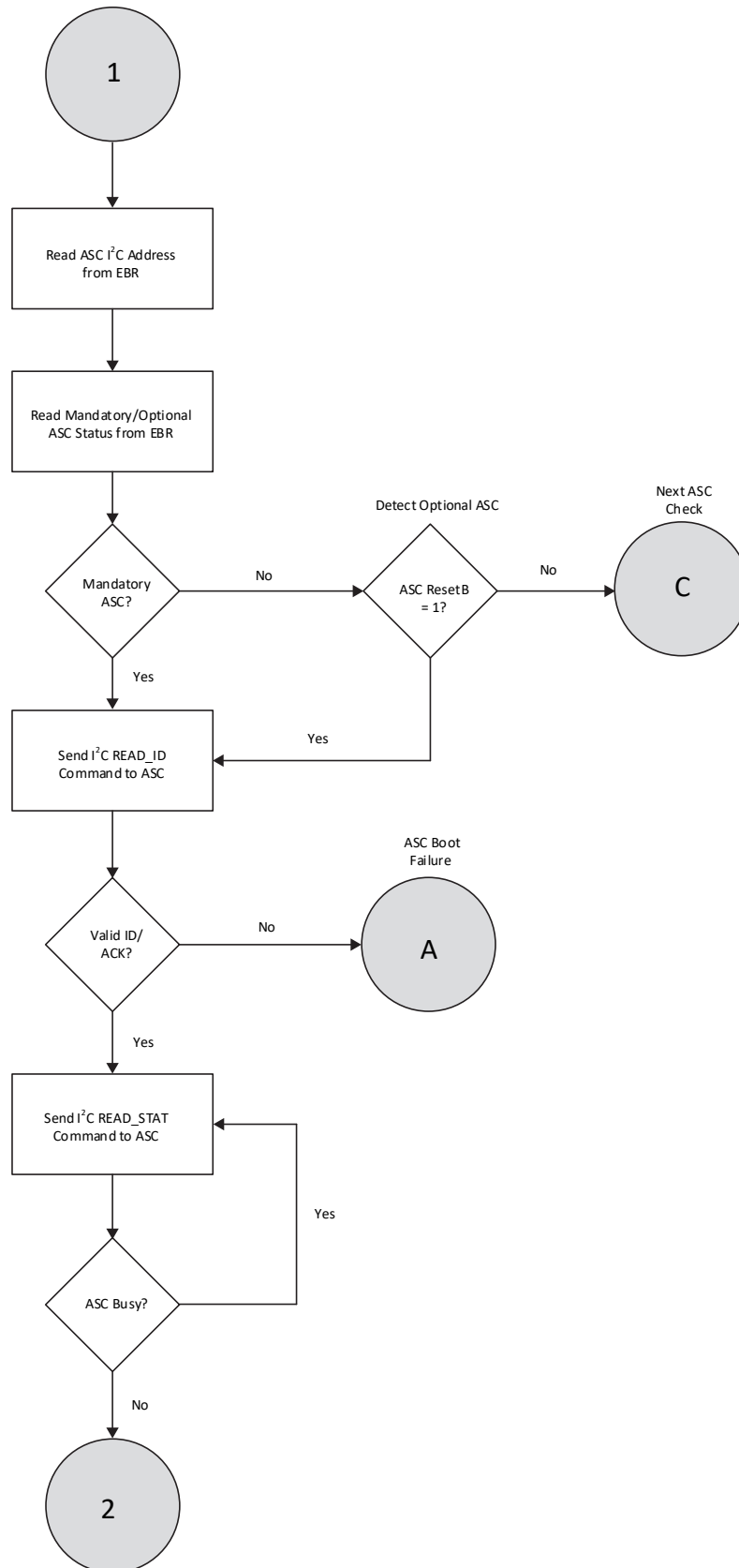
```
reg temp_sig = 0;
reg start_con = 0;
// -- detect the start condition
always @ (negedge I2C_sda or posedge temp_sig)
begin
    if (temp_sig)
        start_con = 0;
    else if(I2C_scl)
        start_con = 1;
    end
// -- generate the start detect pulse
reg start_con_ff1, start_con_ff2;
wire start_detect;
always @(posedge ASC0_CLK)
begin
    start_con_ff1 <= start_con;
    start_con_ff2 <= start_con_ff1;
end
assign start_detect = start_con_ff1 & !start_con_ff2;
// -- apply the logic to remove the pulse
initial
begin
    force ASC1_RSTN_I = 1;
    force ASC2_RSTN_I = 1;
    forever
```

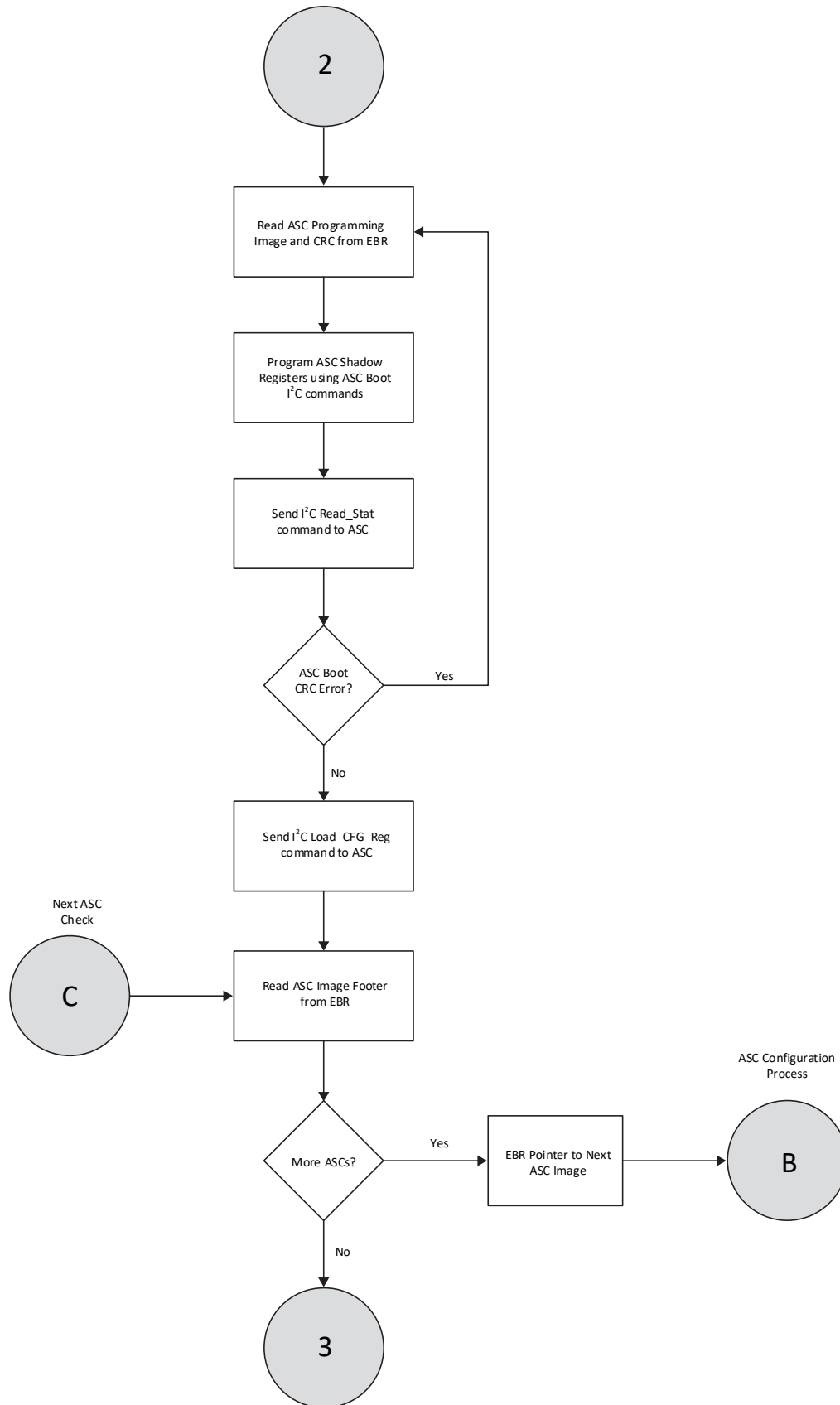
```
begin
@(posedge start_detect);
@(negedge I2C_scl)
force I2C_sda = 0;
$display ("%t, start of force", $time); //only for debugging, can be removed
temp_sig = 1;
#750 release I2C_sda; // #750 for 400kHz i2c frequency, use #2600 for 100KHz
$display ("%t, end of force", $time); // only for debugging, can be removed
temp_sig = 0;
#2000;
$display ("%t, exit the loop", $time); // only for debugging, can be removed
end
end
```

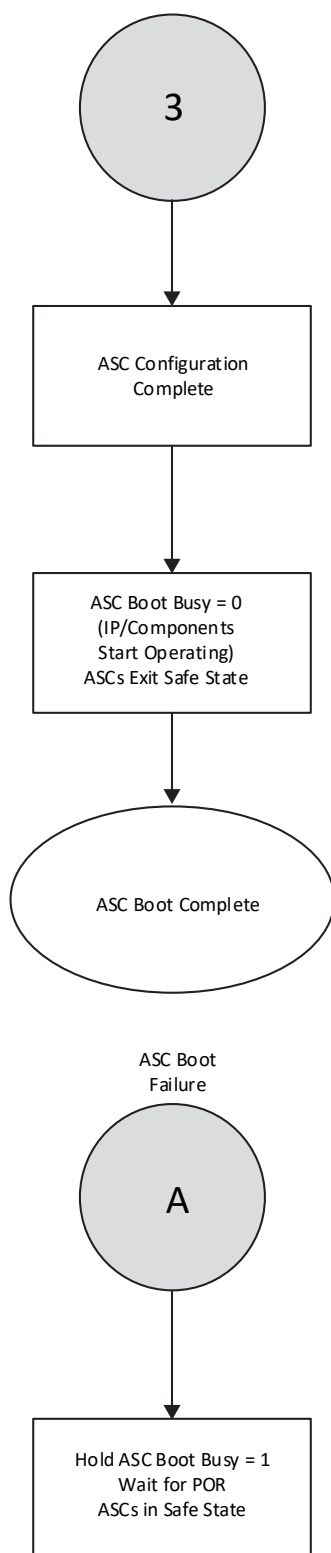
With these additional lines of code added to the simulation testbench, the simulation will more accurately represent the hardware behavior of the ECP5 + ASC system. These lines are only needed for post-map simulation, they are not required for RTL simulation.

Appendix A. ASC Boot Flow Chart









Technical Support Assistance

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

For frequently asked questions, refer to the Lattice Answer Database at <https://www.latticesemi.com/Support/AnswerDatabase>.

Revision History

Revision 1.2, August 2023

Section	Change Summary
All	Corrected document number to <i>FPGA-AN-02019</i> from <i>FPGA-TN-02019</i> in the footer.
Technical Support Assistance	Newly added the link to Lattice Answer Database.

Revision 1.1, January 2020

Section	Change Summary
All	- Changed document number from AN6095 to FPGA-AN-02019. - Updated document template.
Disclaimers	Added this section.

Revision 1.0, June 2017

Section	Change Summary
All	Initial release.



www.latticesemi.com