ForgeFPGA Configuration Guide

This document describes how to configure the ForgeFPGA core from three different configuration bitstream sources: External SPI/QSPI Flash, Internal OTP, MCU as a host. It also discusses on how to use the Development Board, Socket Adaptor Board, and the Evaluation Board to debug

Contents

1. Terms and Definitions 1
2. References 2
3. Introduction 2
4. General SPI Interface 2
4.1 SPI Modes with Clock Polarity and Clock Phase 3
5. Development Board 4
6. Evaluation Board 7
6.1 MCU Block Description 9
6.2 Additional Board Features 10
6.3 Pin Description 10
7. OTP Read/Write 11
7.1 Writing the OTP Block 12
7.2 Reading the OTP Block 13
7.3 Read Command Structure 14
8. QSPI Programming (Master Mode) 15
8.1 Configure ForgeFPGA from External Flash Memory 16
9. MCU Programming (Slave Mode) 18
10. Conclusion 19
11. Revision History 20

1. Terms and Definitions

OTP	One Time Programmable on chip NVM
QSPI	Quad Serial Programming Interface
MCU	Micro Controller Unit
FPGA	Field-Programmable Gate Array
CPOL	Clock Polarity
CPHA	Clock Phase
EVB	Evaluation Board

2. References

[1] SLG47910, Datasheet, Renesas Electronics Corporation
[2] ForgeFPGA Designer Software, Software Download and User Guide, Renesas Electronics Corporation
[3] ForgeFPGA Dev. Board R1.1 User Guide
[4] ForgeFPGA Socket Adapter Quick Start Guide R1.0

3. Introduction

An internal Configuration Wrapper is used to configure the ForgeFPGA core. The configuration can be done from three different configuration bitstream sources:

- External SPI/QSPI Flash
- Internal OTP
- MCU as a host

The ForgeFPGA Designer Software is used to generate bitstreams. The schematic in Figure 1 shows a block diagram of the SLG47910 configuration block and the external MCU Host and QSPI Flash interface. The four Configuration pins are GPIO3 (SPI_CLK), GPIO4(SPI_SS, Chip Select), GPIO5(SPI_SI, serial input) and GPIO6(SPI_SO, serial output). GPIO9 is used as a Config Done signal. Table 1 shows which modes activate the SPI Master and SPI Slave blocks during configuration.

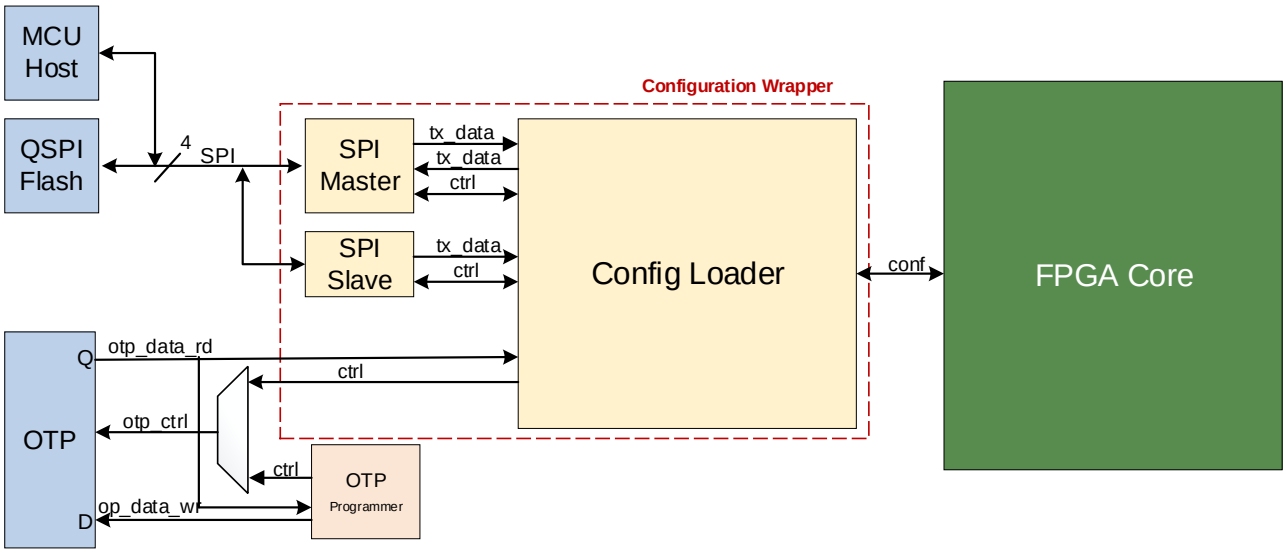

Figure 1: SLG47910 Programming Interface

Table 1: Configuration Modes

Configuration Mode	SPI Block Activated	Clock Source
QSPI/SPI	Master	FPGA
MCU	Slave	MCU
READ OTP	Slave	External to FPGA
Write OTP	Slave	External to FPGA

4. General SPI Interface

A 4-wire SPI device has four signals (see Figure 2):

1. **SCLK: Serial Clock** (output from Master). When the master communicates with the slave, the data on MOSI or MISO pin will be synchronized with the Serial Clock. In the SPI protocol, the master produces the clock. The

slave will only receive the clock, so the slave has no control over the serial clock, which is produced by the master.

2. **MOSI: Master-Out Slave-In** (data output from master). MOSI is a data pin. This pin is used to transmit data from the Master to the Slave device. Whenever the master sends data, that data will be collected over the MOSI pin by the slave.

3. **MISO: Master-In Slave-Out** (data output from slave). MISO is a data pin. This pin is used to transmit data from the slave to the master. Whenever the slave sends data, that data will be collected over the MISO pin by the master.

4. **SS: Slave-Select** (often active low, the output from master). Depending on the SPI and slave select setting, the SS pin used to select an individual slave device for communication. When there is one master and one slave device, then the SS pin is not required. This slave select pin will make sense only when the master is communicating with the different slaves. So, the master can select the slave to which the master wants to convey. For choosing the slave, the SS pin dedicated.

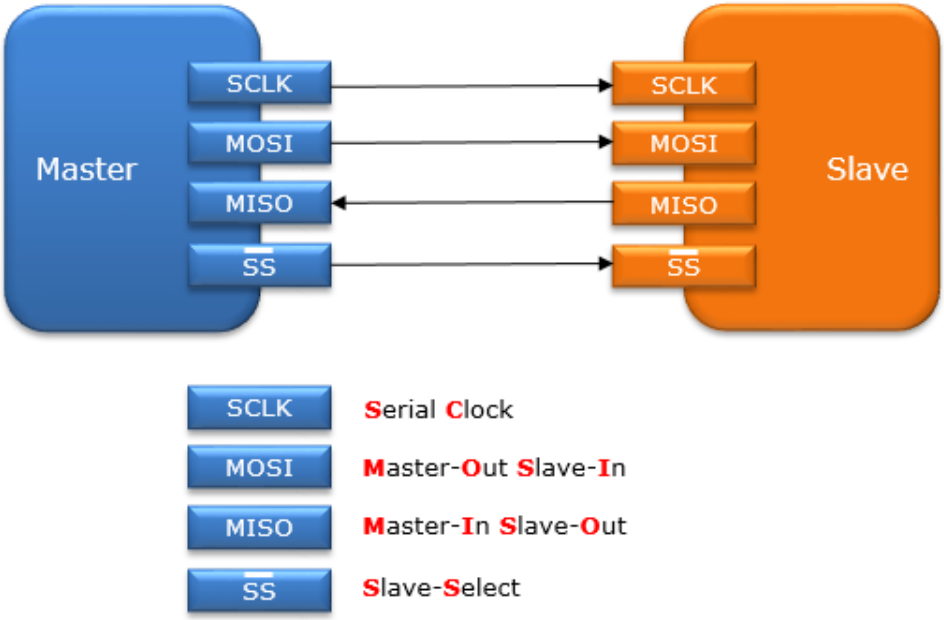

Figure 2: SPI Interface

4.1 SPI Modes with Clock Polarity and Clock Phase

In SPI, the master can select the Clock Polarity (CPOL) and Clock Phase (CPHA). The CPOL bit sets the polarity of the clock signal during the idle state. The idle state is defined as the period when SS is transitioning. The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data. Depending on the CPOL and CPHA bit selection, four SPI modes are available. (See Table 2)

Table 2: SPI Modes

SPI Modes	CPOL	CPHA	Clock Polarity in Idle State	Clock Phase Used to Sample and/or Shift the Data
0	0	0	Logic Low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic Low	Data sampled on the falling edge and shifted out on the rising edge
2	1	1	Logic High	Data sampled on the falling edge and shifted out on the rising edge
3	1	0	Logic High	Data sampled on the rising edge and shifted out on the falling edge

Figure 3 shows the data on the MOSI and MISO line. The green dotted lines show, the end and the beginning of the transmission. Also, the data sampling is shown with orange line which corresponds to the rising or falling edge depending on SPI Mode. The shifting edge of the data is depicted using the blue doted lines. Figure 3 depicts the SPI Mode 0 with CPOL = 0 and CPHA = 0 with the clk idle state = 0 and hence the data is sampled on rising edge and the shifted on falling edge according to Table 2.

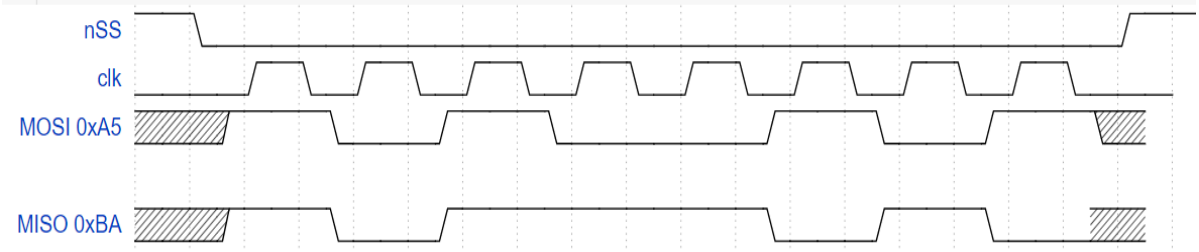

Figure 3: SPI Mode 0

5. Development Board

There are two pre-requisite steps that need to be performed before the design is sent to the device and can be further configured with the development board.

a. **RTL Synthesis**: After creating your desired Verilog Code in the HDL Editor Window of the ForgeFPGA Workshop, the next step is to create a Netlist of your design. This can be done with the help of the built-in Synthesis tool that takes input design and produces a Netlist out of it. While performing synthesis, the input design is analyzed and converted into gate-level representation.

b. **Generating Bitstream**: To prepare your design to be sent to the device you need to perform the Place-and-Route procedure, that takes the elements of the synthesized netlist and maps its primitives to FPGA physical resources. You can do this after successfully generating netlist and pressing Generate Bitstream button on the control panel. Completing these two steps would have successfully sent the design to the device.

To enter the debug controls of the development board, we need to select the correct platform on which we need to configure our device. Under the "Debug" button on the toolbar, select the ForgeFPGA Development Board as the platform (see Figure 4)

Figure 4: ForgeFPGA Development Platform Selection

The FPGA Development Board (see Figure 5) is a multi-functional tool that allows the user to develop their FPGA designs with ease by providing on board power source, digital and analog signal generation, and logic analysis capabilities. The FPGA Development Board can connect additional external boards called socket adapters (see Figure 7). The function of the socket adapter board is to implement a stable electrical connection between the pins of the chip under test and the FPGA Development Board. To implement this, the FPGA Development Board has a Dual PCIe connector. This connector has 40 differential pairs (80 digital channels), 32 analog pins, service pins, and power pins. Dual PCIe connector is universal and can be applied to multiple socket adapter boards.

Driven by the free software, the FPGA Development Board can be configured to work as any one of several traditional instruments, which include:

- Logic Analyzer
- Digital pattern generator
- 8-channel analog Arbitrary Waveform Generator (AWG)
- Precision ADC
- Three programmable power supplies (+0.6 V…+3.3 V). The maximum available output current 2 A. The same voltage is supplied to the GPIO, for keeping the logic level compatibility with the circuit under test.

Also, the board can be used as an independent unit. The chip can be powered through the EXT PWR connector and signals can be read through the through-hole 12-pin connectors (Pmod connectors).

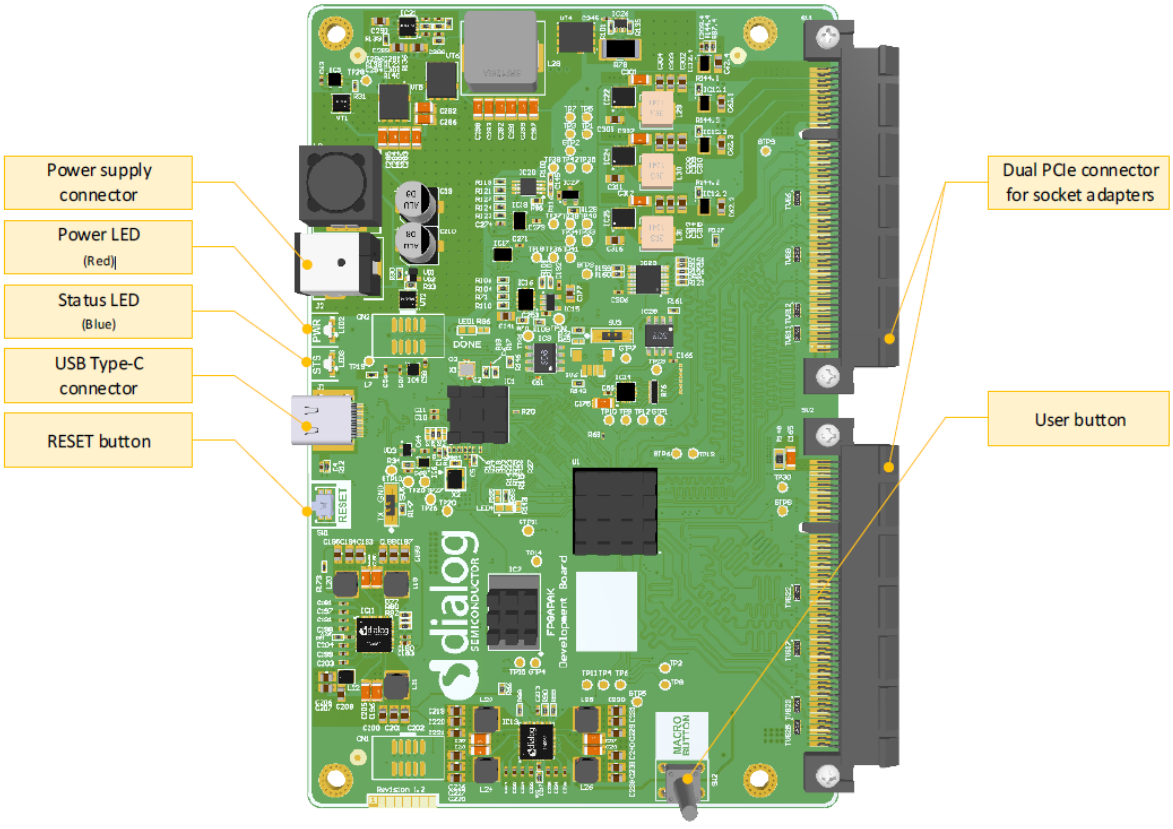

Figure 5: ForgeFPGA Development Board Overview

To configure the development board and read the desired output, connect the Development Board with the Socket Adapter through the PCIe connectors. Put the SLG47910 part in the socket. Then connect the USB cable from the laptop to the USB Type-C Connector (see Figure 8). Connect the power cord that was supplied with the development board to the power supply connection on the development board. If all the connections are correct, then the RED LED(PWR) and BLUE LED(STS) should light up and the software would have recognized the SLG47910 part in the socket.

The user can now either program the chip with the design by clicking on the "Program" button else, the user can use the "Emulation" button under the Debugging Controls Panel (see Figure 6) to observe the output and manipulate it by connecting the Development Board to Oscilloscope, or any PMOD if required.

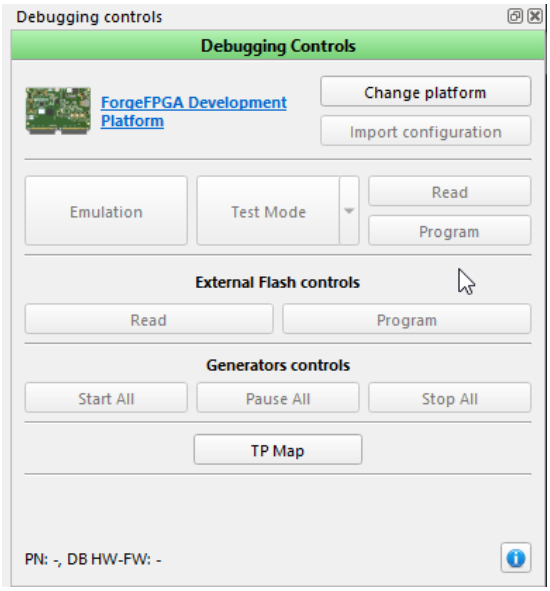

Figure 6: Debugging Controls Panel

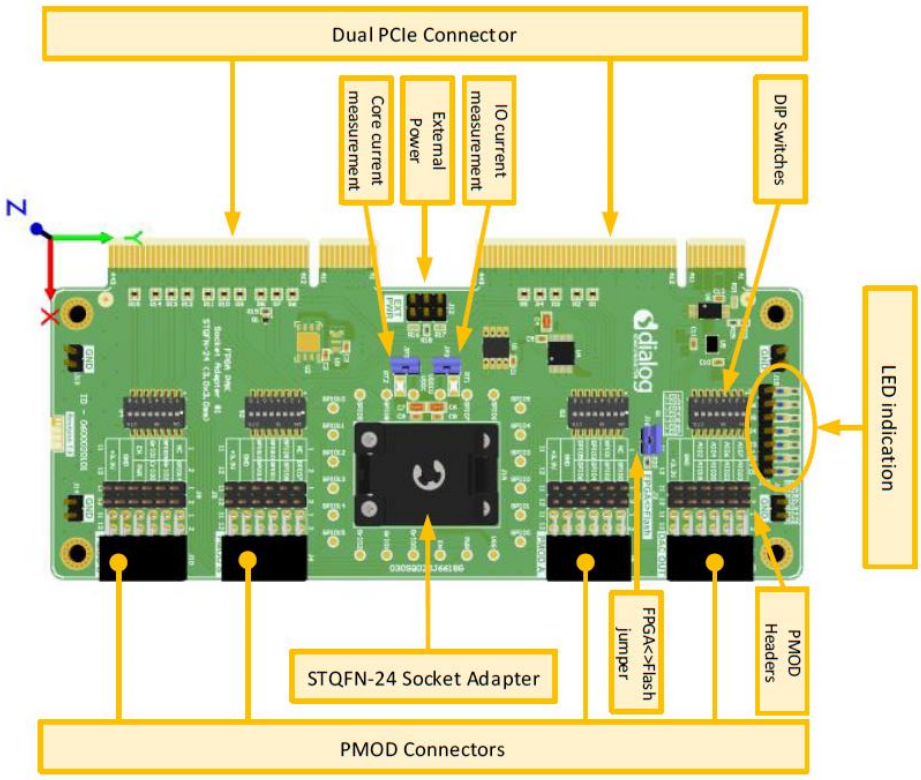

Figure 7: ForgeFPGA Socket Adapter, Top View

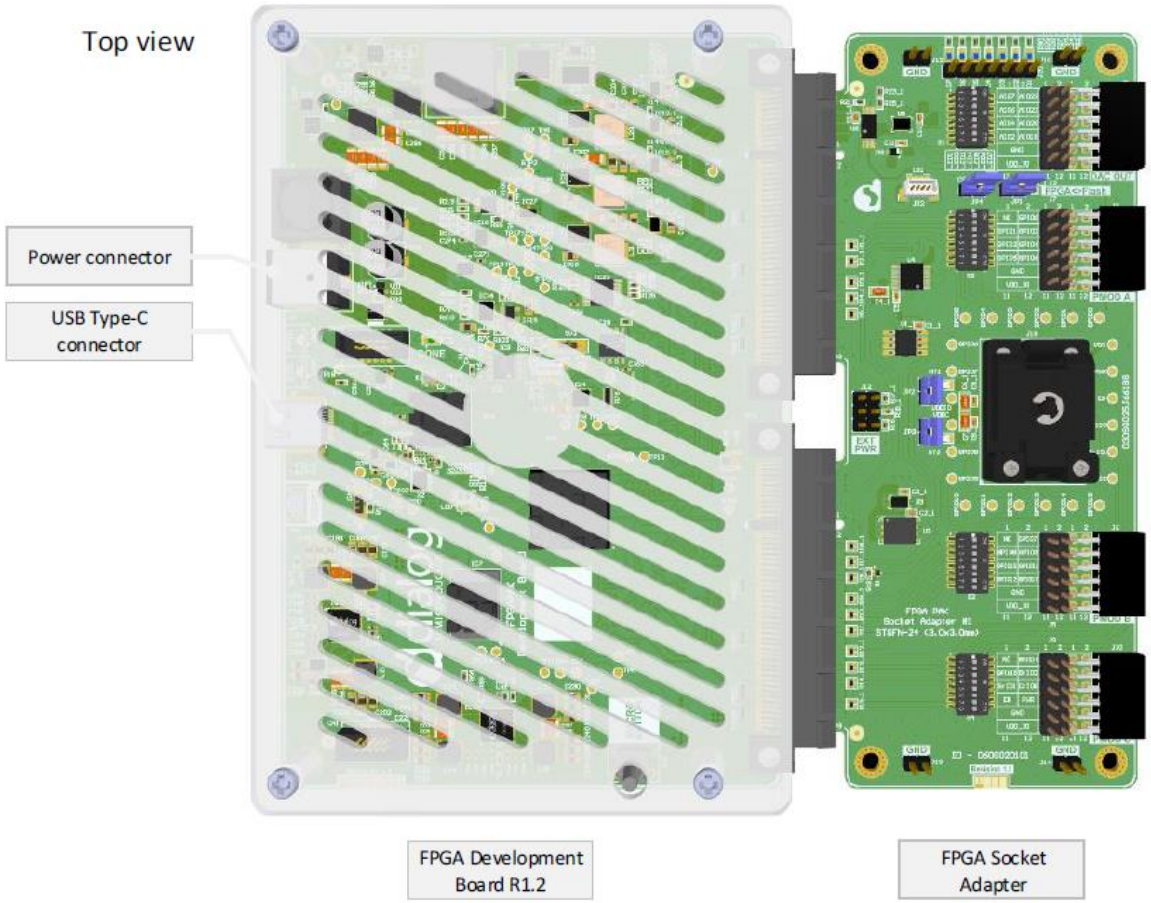

Figure 8: Assembled Equipment for Working with the Chip

6. Evaluation Board

To enter the debug controls of the Evaluation Board, we need to select the correct platform on which we need to configure our device. Under the "Debug" button on the toolbar, select the ForgeFPGA Evaluation Board as the platform (see Figure 9)

Figure 9: ForgeFPGA Development Platform Selection

ForgeFPGA Evaluation board provides SLG47910 IC hardware support for design emulation and real time testing. The board consists of 2 main blocks – Programmer and SLG47910 IC with external connectors.

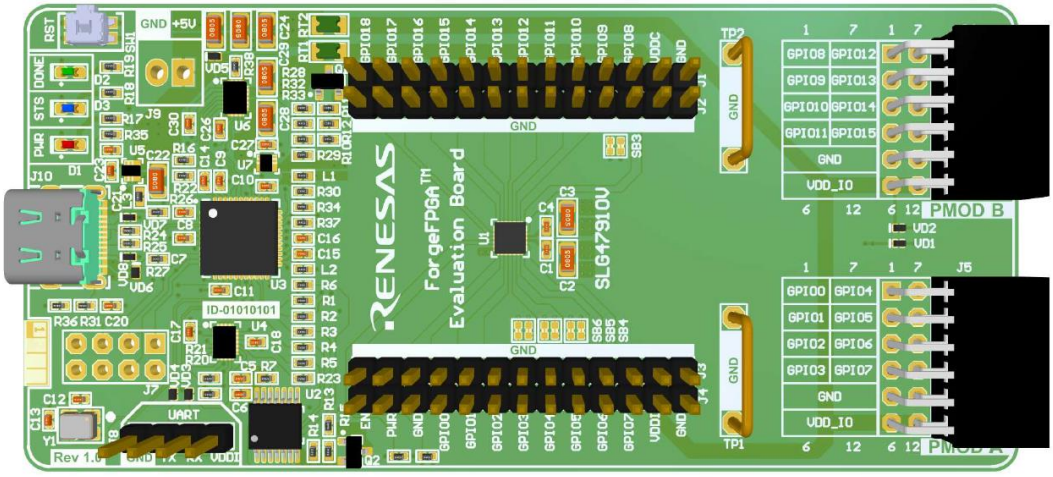

Figure 10: ForgeFPGA Evaluation Board (Top View)

ForgeFPGA Evaluation board is a standalone USB powered and USB controlled system. The design emulation and peripheral control are provided by ATMEGA32U4 MCU, that woks as USBSPI/ I2C/UART/GPIO Bridge. Below is the table specifying the voltage range for which the board operates.

Table 3: Voltage Supply Range

Parameter	Min	Max	Unit
Supply Voltage (VDDC)	-0.3	1.5	V
Supply Voltage (VDDIO)	-0.3	3.6	V
Voltage at Digital Input Pins	-0.3	3.6	V

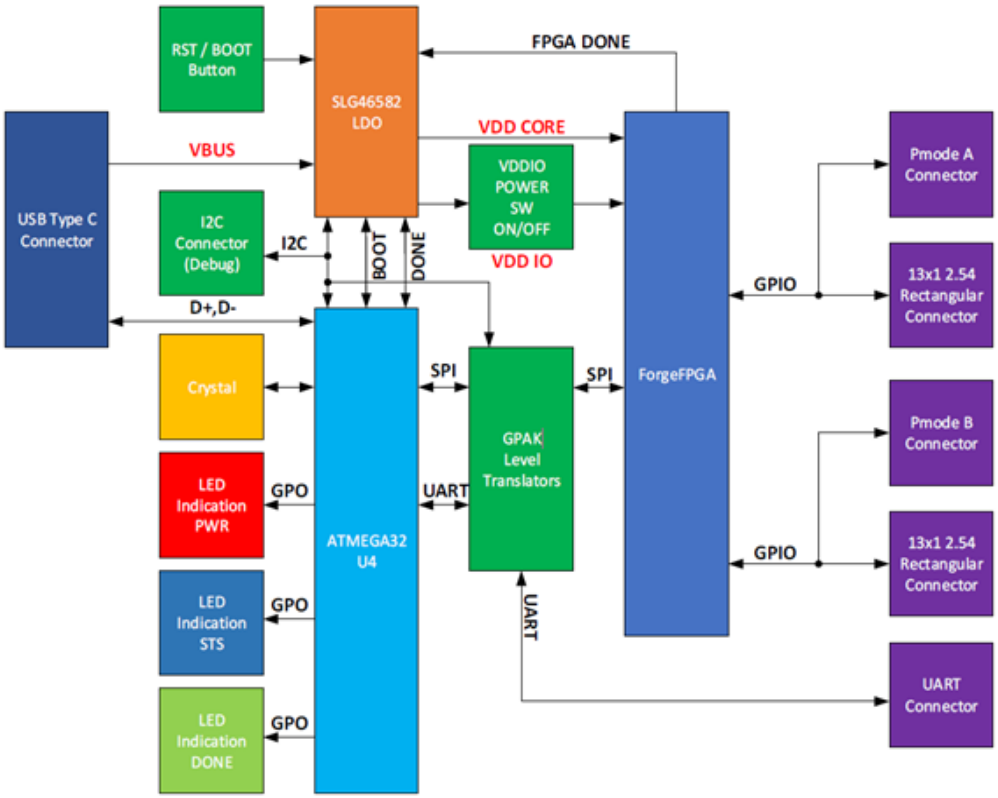

Figure 11: Board Block Diagram

6.1 MCU Block Description

MCU block is based on ATMEGA32U4 Microcontroller, that works as CDC USB with software and hardware DFU enable feature. MCU main functions are indication, SPI transfer, UART transfer, Board self-test (by ADC) and I2C Master configurations. There are three onboard indication LEDs their functions describe in table below:

Table 4: LED Functions

LED Type	Function
PWR	Indicates MCU Configuration and Self-Test Status If MCU firmware is not loaded or self-test failed led won't come up
STS	Indicates USB Package Transfers
DONE	Indicates ForgeFPGA configuration If Bit stream is loaded correctly - Led will come up

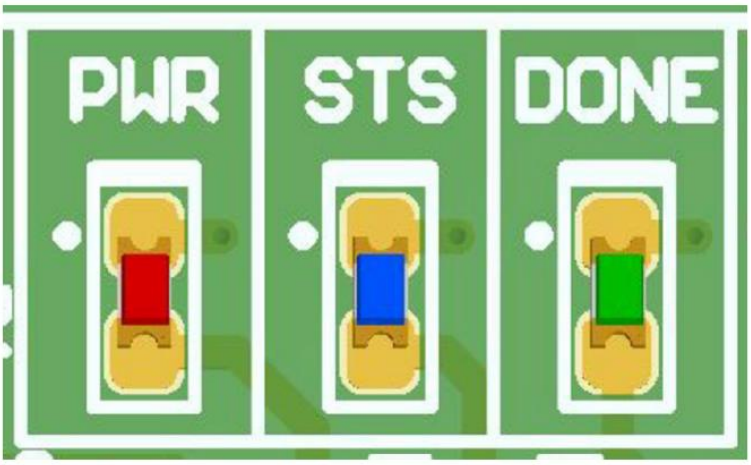

Figure 12: Indication LEDs

SPI Transfer functionality is required for ForgeFPGA programming. MCU performs full programming sequence via SPI and checks configuration status using Config Done Pin pulse detection. After successful configuration MCU will turn "DONE" LED on.

UART transfer is option that have a place if user will configure UART block inside of ForgeFPGA. It's possible to debug UART block by sending and receiving data from UART terminal inside GoConfigure software.

Self-Test feature is provided by MCUs internal ADC. Measurements are available on VUSB, VDDIO and VDDC power nets. All power lines attenuators have the same resistor values, so regarding that ADC Vref is 2560mV and resolution - 10b conversion formula is:

$$Vpower = ADC\_code * 2.5 * 3.55 (mV)$$

6.2 Additional Board Features

Evaluation board has an option of connecting the power externally as well. It can be connected through the 2.54mm Block terminal J9 on the board which is connected to +5V internal bus Recommended part number for J9 Is OSTVN02A150 (see Figure 13)

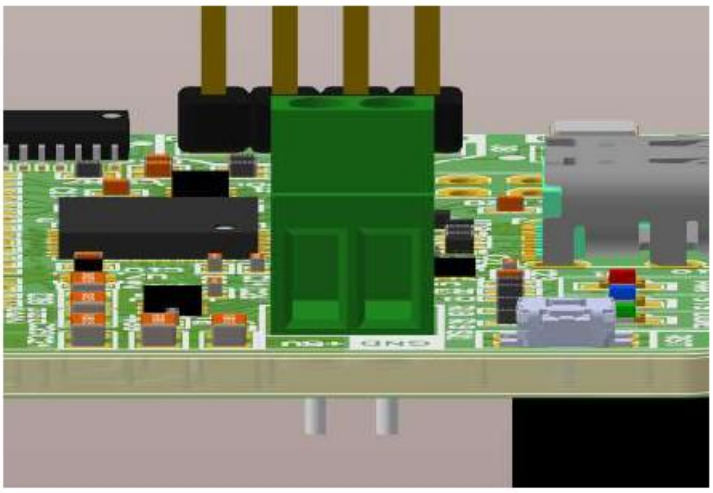

Figure 13: Power Output Terminal

6.3 Pin Description

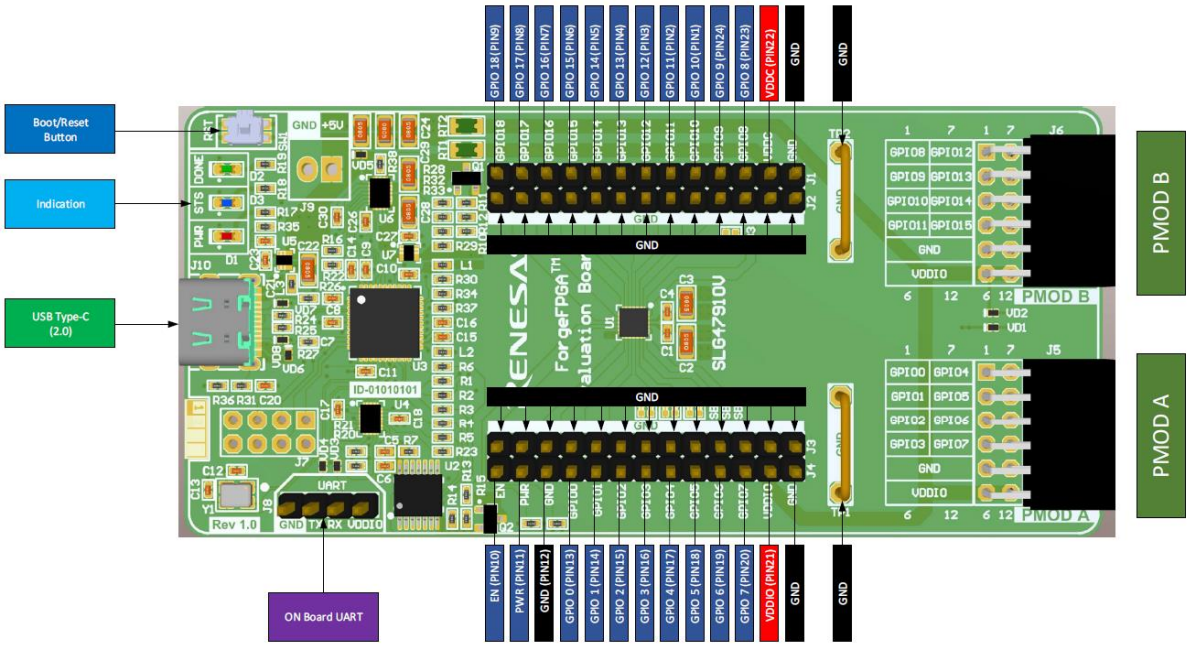

Figure 14: Board Signals Top View

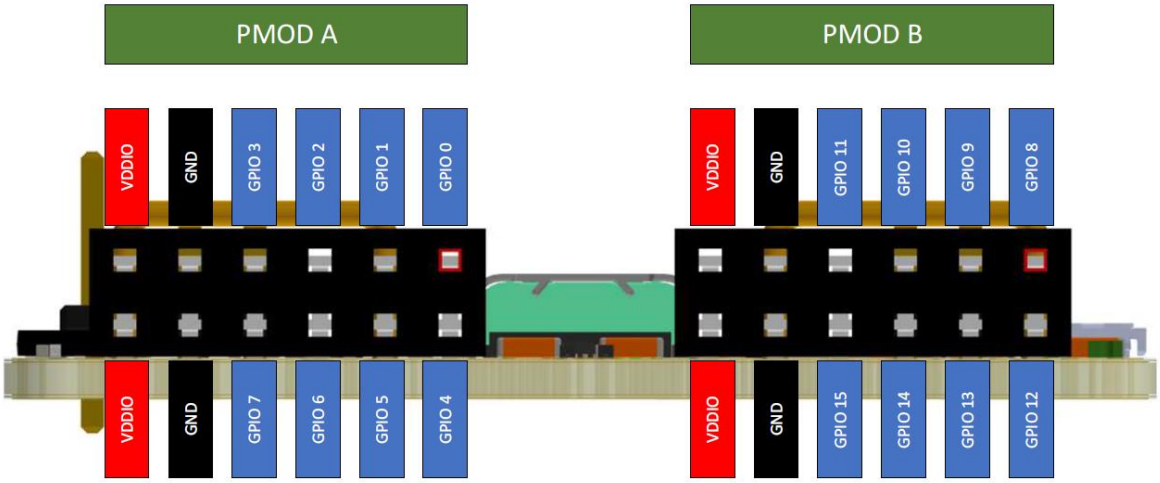

Figure 15: PMOD Connectors

7. OTP Read/Write

The configuration for the FPGA is stored in the Configuration RAM. The Configuration RAM is a volatile memory that stores the FPGA design. The OTP memory loads the Configuration RAM. The SLG47910 contains three blocks of 4k x 32-bit One-Time Programmable (OTP) Non-Volatile Memory (NVM), which are interfaced via the dedicated SPI Slave circuit block (Figure 16).

The user can read or write the OTP block through the SPI interface pins on the SLG47910. If the OTP block of the SLG47910 has been programmed, upon POR, the contents of the NVM will be loaded into the device's internal configuration RAM.

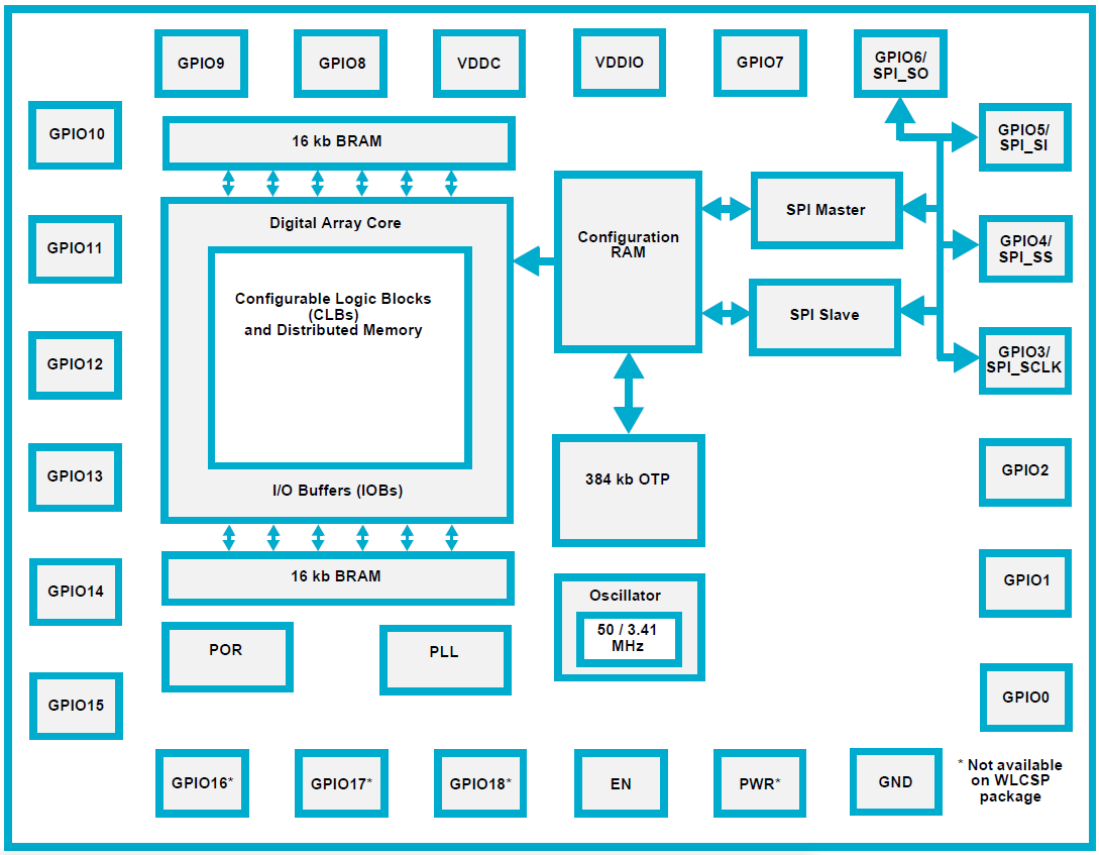

Figure 16: SLG47910 Block Diagram

The loading of the data through different bitstream sources follow a particular flow and different values of the signal help in determining the mode of operation. The Figure 17 below showcases a part of that flow.

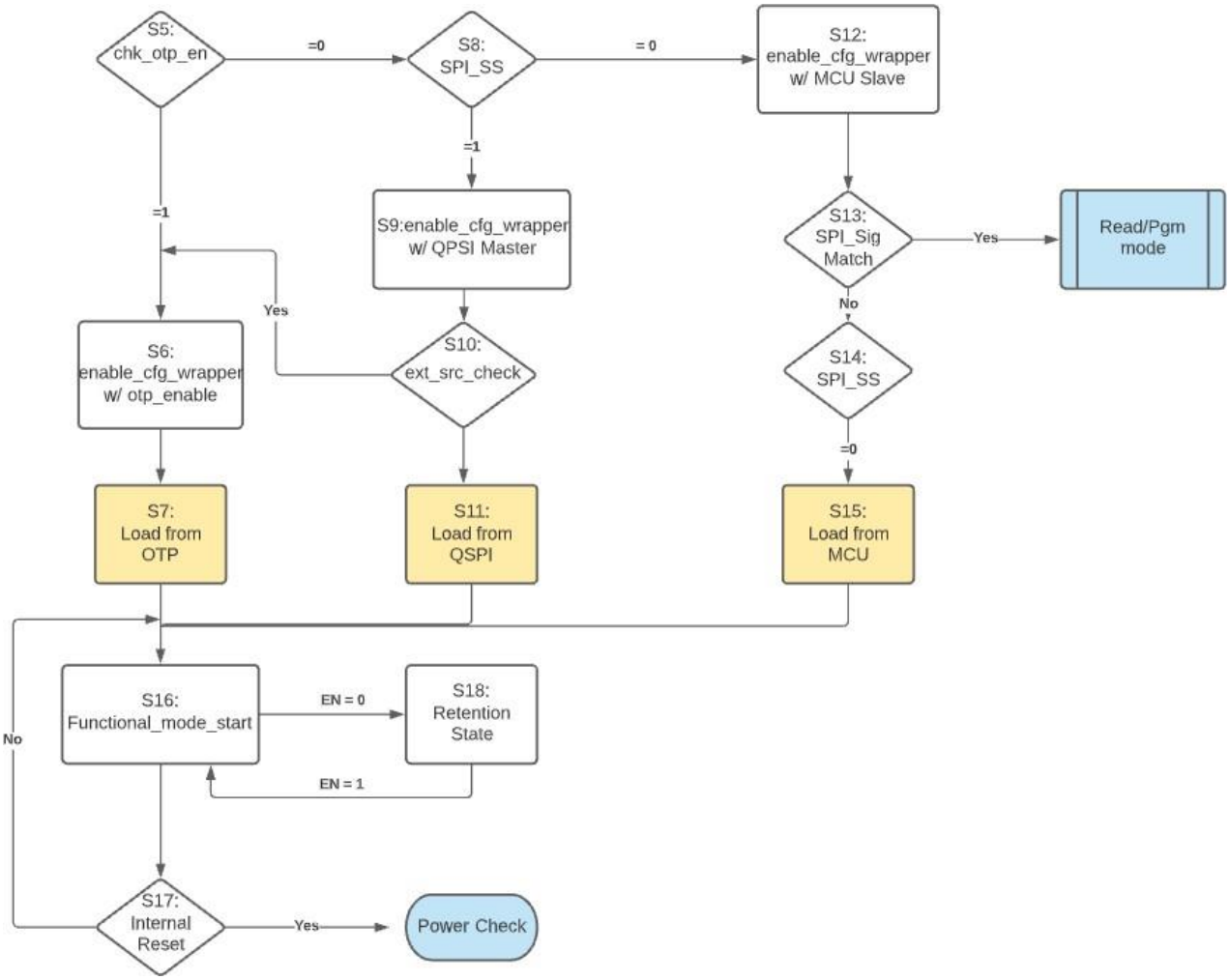

Figure 17: Power Sequencing through Different Modes

7.1 Writing the OTP Block

The OTP is written using the SPI Slave interface. The OTP write starts with ramp up of the voltage signals VDDC to 1.1 V and VDDIO to 2.9 V which sets the NVM into write program mode. There is an otp_controller that sequences the internal signals to enable the OTP write.

Once the overdrive and program mode inputs are active, the otp_controller will decode the spi_slave data, and give, data address and control information to the otp_write_controller block to initiate the OTP write. Table 5 shows the OTP Write Packet format. Table 6 shows the Write/Read options bits. The last Write packet is indicated by setting Byte8 bit[6]. Reserve bits are indicated by R and the parity bit by P. After writing the OTP the write data should be checked using the OTP read command. Once the SLG47910 OTP has been written and after POR or bringing the PWR pin low, MCU, QSPI, OTP write, and OTP read will be disabled. The SLG47910 will only load program data from the OTP. This is a design security feature.

Writing to the OTP has the following steps:

1. Wait for POR and PLL lock time (1300 us), then send the Signature Bytes through SPI_MOSI (GPIO_5) by keeping SPI_SS (GPIO_4) low.

2. After the Signature bytes match, GPIO_9 (Config-Sig match) goes high and giving delay of 80 us and then sending the OTP write command packets with delay gap between 1st and 2nd packet is 18 us delay and consecutive packets is 10.11us delay. (S13)

3. In the last write packet Byte8[6] =1 which indicates the last write.

4. When done writing OTP bring PWR =L which resets the device, then at S5 chk_otp_en will be one.

Table 5: OTP Write Packet Format (Incoming)

Bits	0	1	2	3	4	5	6	7
Byte 1	W/Rn (1)	SP/AP (0)	R	R	R	O1_A[15]	O1_A[14]	O1_A[13]
Byte 2	O1_A[12]	O1_A[11]	O1_A[10]	O1_A[8]	O1_A[7]	O1_A[6]	O1_A[5]	O1_A[4]
Byte 3	O1_A[3]	O1_A[2]	O1_A[1]	O1_A[0]	O1_D[3]	O1_D[2]	O1_D[1]	O1_D[0]
Byte 4	O2_A[15]	O2_A[14]	O2_A[13]	O2_A[12]	O2_A[11]	O2_A[10]	O2_A[8]	O2_A[7]
Byte 5	O2_A[6]	O2_A[5]	O2_A[4]	O2_A[3]	O2_A[2]	O2_A[1]	O2_A[0]	O2_D[3]
Byte 6	O2_D[2]	O2_D[1]	O2_D[0]	O3_A[15]	O3_A[14]	O3_A[13]	O3_A[12]	O3_A[11]
Byte 7	O3_A[10]	O3_A[8]	O3_A[7]	O3_A[6]	O3_A[5]	O3_A[4]	O3_A[3]	O3_A[2]
Byte 8	O3_A[1]	O3_A[0]	O3_D[3]	O3_D[2]	O3_D[1]	O3_D[0]	Last	P

Address and Data

OTP1
OTP2
OTP3

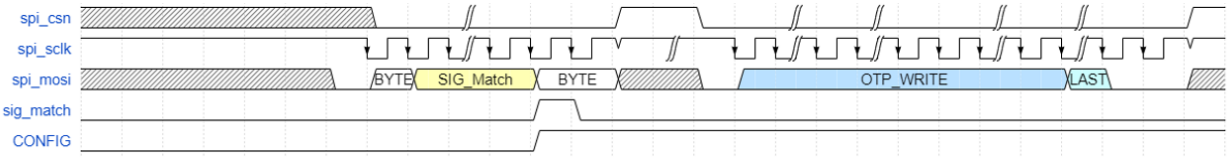

Figure 18: OTP Write Waveforms

Table 6: Read/Write Option Bits

Byte 1 [0:1] of OTP Packet format	Comments
2'b00: Read mode	Follows format in Table 7
2'b01: Reserved	Not Used
2'b10: Write mode	Follows format in Table 5
2'b11: Return	Exit Read/Write OTP

7.2 Reading the OTP Block

Reading of the NVM via SPI is done by sending a three-byte command packet, which is met with a five-byte response packet containing the requested NVM data. A diagram of the read command packet structure is given in Figure 19.

Reading to the OTP has the following steps:

1. Waiting for POR and PLL lock time (1300 us), after that we are sending the Signature bytes through SPI_MOSI (GPIO_5) by keeping SPI_SS (GPIO_4) low.

2. After Signature bytes matches, GPIO_9 (Config-Sig match) goes high and giving delay of 80 us and then sending the OTP Read command packet on SPI_MOSI (GPIO_5) by keeping low SPI_SS (GPIO_4) and OTP read data will come on SPI_MISO (GPIO_6).

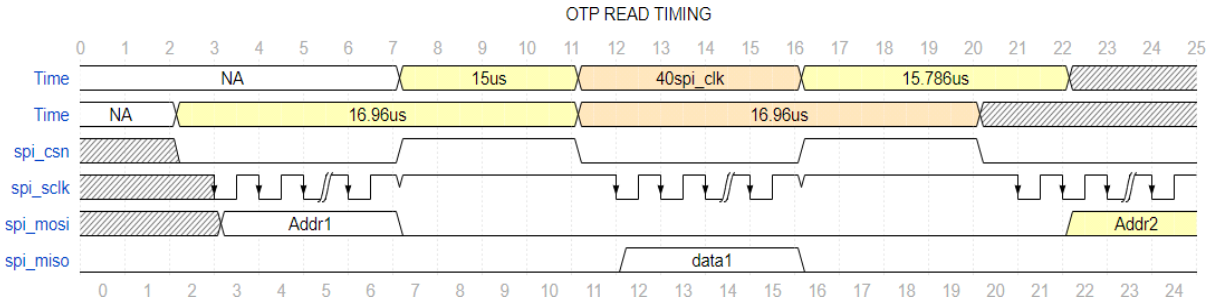

Figure 19: OTP Read Timing

7.3 Read Command Structure

Table 7: NVM OTP Read Command Packet Format

Bits	0	1	2	3	4	5	6	7
Byte 1	nRead (0)	0	R	R	R	R	R	R
Byte 2	A[18]	A[17]	A[16]	A[15]	A[14]	A[13]	A[12]	A[11]
Byte 3	A[10]	A[4]	A[3]	A[2]	A[1]	A[0]	0	P

Byte 1 of the packet will remain the same across all read operations and contains only the read command bit (Byte1[0]). Byte1 Bits[7:2] are reserved (R) bits. A read address is then provided across Bytes 2 and 3, before a parity bit (P) is provided at the MSB of Byte 3. This read address is separated into two sections. A[18:17] determines which of the three 4 k x32 NVM blocks to read from (named OTP1, OTP2, and OTP3) and A[16:10] A[4:0] determines the address within the specified NVM block to read. The parity bit to be provided is calculated by performing an AND operation of all incoming bytes, excluding the parity bit = (^Byte1)^(^Byte2)^(^[Byte3[0:6]]).

Table 8: NVM Block Selection

NVM Block Selection		Address to Read											
A[18]	A[17]	A[16]	A[15]	A[14]	A[13]	A[12]	A[11]	A[10]	A[4]	A[3]	A[2]	A[1]	A[0]

Table 9: OTP Parameters in Write & Read Mode

Symbol	Parameter	Condition	Min	Typ	Max	Unit
Iddio_otp_prog_pk	Peak Current Consumption-Program Mode VDDIO Supply	3 x OTP CSB = L; CLK = H SAP = [00]	48.2	72.3	100.77	mA
Iddc_otp_prog_pk	Peak Current Consumption Program Mode VDDIO Supply	3 x OTP CSB = L; CLK = H SAP = [00]	5.2	6.15	7.5	uA
Iddio_otp_rd_pk	Peak Current Consumption Read Mode -VDDIO Supply	3 x OTP CSB = L, CLK = H, SAP = [00]	TBD	TBD	TBD	mA
		3 x OTP CSB = L, CLK = H, SAP = [10]	TBD	TBD	TBD	
		3 x OTP CSB = L, CLK = H, SAP = [11]	TBD	TBD	TBD	
Iddc_otp_rd_pk	Peak Current Consumption Read Mode- VDDIO Supply	3 x OTP CSB = L, CLK = H, SAP = [00]	3.54	4.71	7.48	mA

		3 x OTP CSB = L, CLK = H, SAP = [10]	3.49	4.64	7.37	
		3 x OTP CSB = L, CLK = H, SAP = [11]	3.47	4.61	7.35	

8. QSPI Programming (Master Mode)

In SPI Master mode, the communication starts by driving SPI_SS Low, and then sends a Release from Power-down command to SPI Flash, 0xAB. The timing diagram below provides an example waveform. This initial command wakes up the SPI Flash if it is already in Deep Power-down Mode (see Figure 20). The SPI Master transmits data on the SPI_SO output, on the falling edge of the SPI_CLK output. No data is sent on SPI_SI currently. After sending the last command bit, the SPI_SS is de-asserted High, completing the command. Minimum of 10 us buffer is given before sending the next SPI Flash command.

Table 10 represents the pins in SLG47910 corresponding the pins in SPI

Table 10: SPI Pin Numbers

QFN Pin No	Pin Name	QSPI/SPI Mode
16	GPIO3	SPI_CLK
17	GPIO4	SPI_SS
18	GPIO5	SPI_SI (MISO)
19	GPIO6	SPI_SO (MOSI)

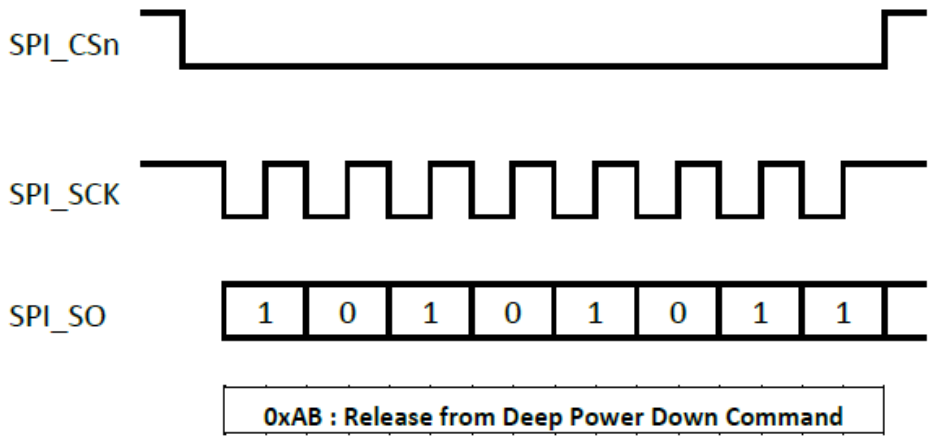

Figure 20: SPI Release from Deep Power Down Command

After the 10 us is up, the SPI Master sends a Fast Read Command with 24-Address on the SPI_SO pin and receives the data on the SPI_SI port as shown (see Figure 21). The first data byte (Data Byte 0) read from the QSPI flash is the lowest byte of a 32-bit data word.

After transferring the required number of configuration data bits, the **wrapper** ends the Fast Read command by de-asserting its SPI_SS select output. To conserve power, the **wrapper** then issues a final Deep Power-down command, 0xB9.

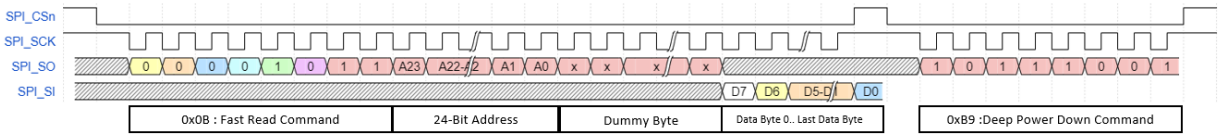

Figure 21: SPI Read Fast Command & Deep Power-Down Command

The SLG47910 device configures using a single data pin SPI_SI.

The procedure to enable QSPI Configuration are given below:

1. Hold GPIO4 (SPI_SS) High for a minimum delay of 1055 us. To enable QSPI mode. (S9)

2. After entering QSPI mode, internal logic will release GPIO4 (SPI_SS) from driving through the GPIO. (S9)

3. The internal Config-Wrapper which acts as master will control the SPI interface (GPIO3, GPIO4, GPIO5 & GPIO6) throughout the FPGA configuration. (S9 -> S11)

4. The wrapper sends a wake-up command (AB) to QSPI flash device first and then it sends fast read command (QB). After the configuration completes it sends the sleep command (B9). (S9 -> S11)

5. Config wrapper will generate internal Config done = 1 (S16)

8.1 Configure ForgeFPGA from External Flash Memory

The ForgeFPGA Socket Adapter Board has an onboard 4 Mbit (2M x 2) SPI serial flash memory(external). It is used for uploading bitstream into the ForgeFPGA externally. The Go Configure Software Hub software allows the user to program and debug the external flash memory on ForgeFPGA Socket Adapter.

- **Programming into external flash memory:**

After generating a bitstream of the desired design, the user can program the external flash memory by pressing the button "**PROGRAM**" in the External Flash control section of the Debugging control panel in the main window (See Figure 22, Figure 23). After uploading the bitstream to the external flash memory, the user receives a notification.

The uploaded bitstream data can be read back from the "**READ**" button.

Figure 22: Debugging Controls Panel: Program & Read

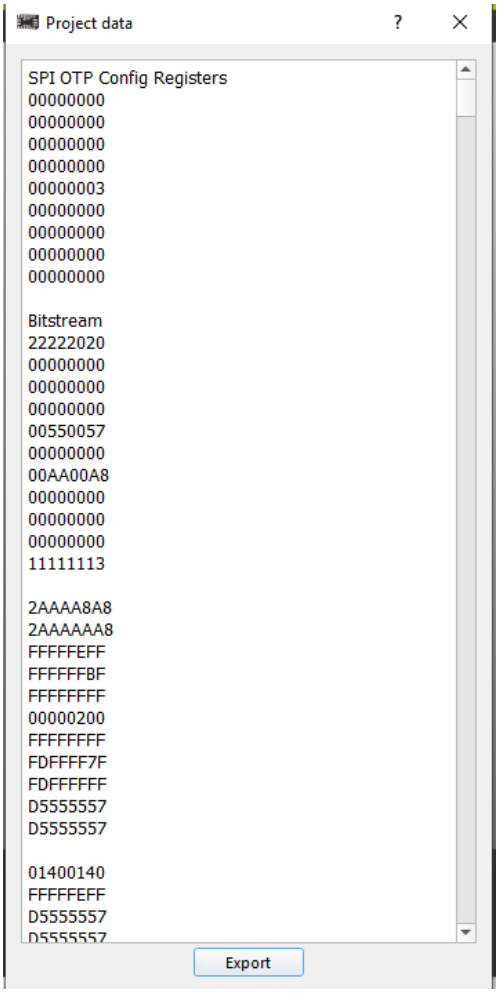

Figure 23: Reading from External Flash

Once the bitstream has been programmed into the external flash memory, its time to upload it to the FPGA chip. To do that, the user needs to disconnect the jumper "*FPGA<>Flash*" (marked with Red circle) in order to connect to the onboard external flash memory (see Figure 24)

- **Uploading into ForgeFPGA:**

Method 1:

The ForgeFPGA Socket Adapter needs to be disconnected from the ForgeFPGA Development Board. To power the Socket Adapter, use external power source and connect it to the headers (marked with purple circle) namely VDDIO and VDDC. User needs to check the pin configuration of these headers drawn at the back side of the Socket Adapter Board.

Figure 24: Jumpers and External Power Connection Points on the board

Method 2:

From the Debugging controls Panel in the software, choose the Test Mode option. Under the Test Mode option, the user needs to choose the Test Mode (*) option to upload the bitstream from the external flash memory to ForgeFPGA (see Figure 25)

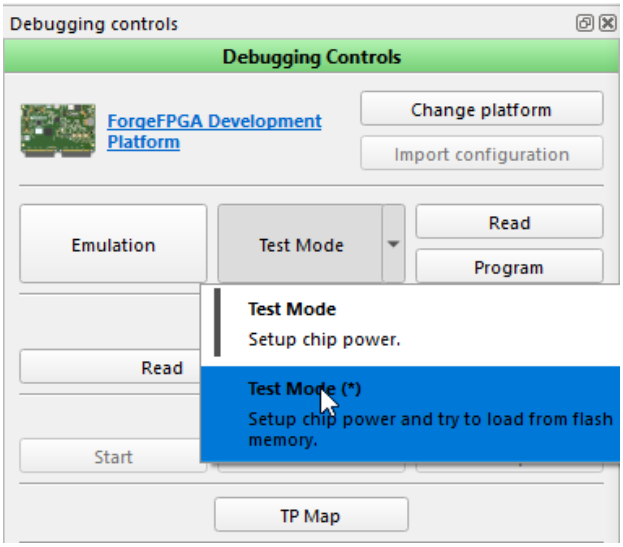

Figure 25: Test Mode (*) option

9. MCU Programming (Slave Mode)

After POR the CONFIG (GPIO9) signal will go low signaling the host (MCU) that it can start sending data to the device. After the reset is de-asserted, the host sends 10242 SPI_SCK cycles (preamble) while holding SPI_CSn low and SPI_SI data set to 0. The preamble is used to flush the SLG47910array before configuring the array. After the preamble, the MCU sends a sync word (32-bits) followed by 288-bits data used to configure other SOC registers. After the 288-bits are sent the MCU will then send the configuration bits. Once the configuration bitstream is sent, the host keeps SPI_SS low and keeps sending SPI_SCKs (postamble) until it sees the CONFIG signal go high. During the postamble, the SLG47910 generates the CHIP_RST signal resetting the array. Figure shows the SPI slave timing.

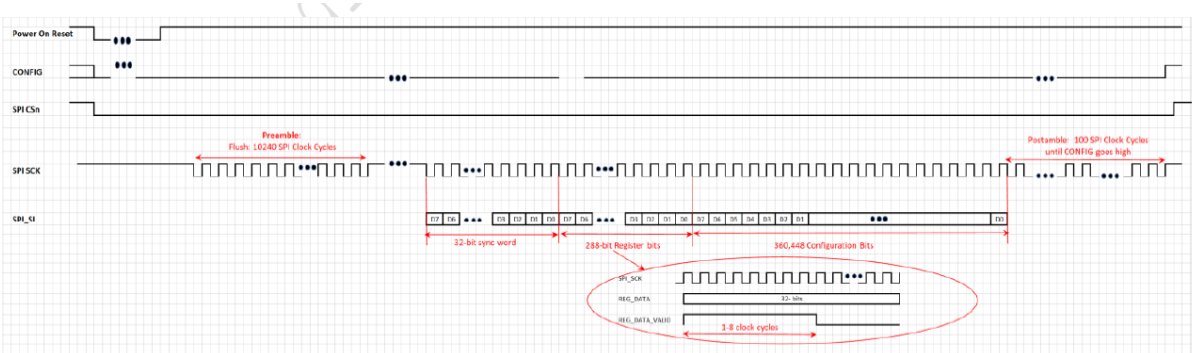

Figure 26: SPI MCU Mode Timing

A 32-bit synchronization word will be inserted in the beginning of the bitstream by the ForgeFPGA Compiler. The SLG47910 will check this synchronization word to determine if the transfer is targeting this device. If the synchronization word does not match this device, then the configuration bitstream will be discarded. This synchronization word is checked on both SPI slave and SPI master.

The following steps are used to program the SLG47910 in MCU mode.

1. Set CONFIG pin to INPUT PULL-DOWN

2. Set PWR and EN pins to "1"

3. Set Power On (VddC= 1.1 V, VDDIO = 1.8 V)

4. Wait 102.1 ms

5. Send SIGNATURE (1 word, MOSI = 0)

6. Set CS Pin to Pull-Up

7. Wait 2 ms

8. Set CS pin to "0"

9. Send PREAMBLE (321 word, MOSI = 0)

10. Send SYNC (1-word LSB 0x11FF22AA Transmission starts from LSB)

11. Send REGS (9 word. Default all zeros)

12. Send BITSTREAM (11264 words) *****

13. Send POSTAMBLE (6 words, MOSI = 0) *****

14. Set CS pin to Pull-Up (Or ALL pins to Hi-Z)

***** During POSTAMBLE phase the CONFIG pin will strobe for 50ns. If the bitstream is invalid the CONFIG pin will strobe for 1.5us. If a load error occurred, the Config pin will stay LOW.

****** SPI Frequency could be from kHz to MHz

10. Conclusion

This configuration manual focus on three configuration options: OTP, MCU and QSPI/SPI Flash. The document explains how if the part is configured using OTP, then other configuration options will be disabled. It also explains the designing and the operation of the Development Board and the Evaluation Board to test & debug designs. If interested, please contact the ForgeFPGA Business Support Team

11. Revision History

Revision	Date	Description
1.0	10-Mar-2022	Initial Version
2.0	03-Oct-022	Added contents related to Evaluation Board Added OTP Parameters Table
2.1	17-Oct-2022	Added section for how to configure from external flash memory