



OPEN
Compute Project

LVDS TUNNELING PROTOCOL & INTERFACE (LTPI) IP- USER GUIDE

Version 1.2

Author (s):

Katarzyna Krzewska, Intel®

Kasper Wszolek, Intel®

Version History

| Version | Date | Notes |
|---------|-------------------|---|
| 1.0 | 22 August, 2023 | Initial public release, LTPI 1.0 |
| 1.05 | 24 November, 2023 | OCP 2023 LTPI Interoperability Demo version: - Added clarification regarding CRC algorithm (no inversion/reflection) - Added clarification regarding Total Number on NL GPIOs - Increased the LTPI Advertise Frame Alignment timeout to 100ms |
| 1.09 | 16 May, 2025 | - Exposed to the top LTPI modules data channel and CSR access signals - Increased data channel timeout to 10ms - Synchronize lvds phy reset signals - Changed SMBUSs timing parameters - Added gpio ltpi top module and ltpi top module parameterized unit test |
| 1.1 | 19 May, 2025 | - Added requirement for Data Echo and Data Received Echo to be sent at least 3 times and received at least once correctly Implementation compliant with 1.1 LTPI Specification. |
| 1.2 | 10 December, 2025 | LTPI IP 1v2 version updates: 1. Extend CRC and Other Errors Handling with details for all LTPI error types and handling: - Added Frame lost error - Frame CRC verification failed or Unexpected Frame (modules: mgmt_phy_controller.sv and mgmt_phy_target.sv) 2. Changed the LTPI Version name to LTPI Revision to match intended use: - Regenerate CSR packages for all LTPI CSR configurations. (rtl/modules/CSR/RDL directory) - Updated CSR generation packages script to use peakRDL version 1.4.0 (csr_gen.py) 3. Modified Timeout condition for Link Speed (modules: mgmt_phy_controller.sv and mgmt_phy_target.sv) 4. Extended the Link Speed condition for SCM with at least 1 Link Speed received from HPM (modules: mgmt_phy_controller.sv and mgmt_phy_target.sv, mgmt_ltpi_frm_rx.sv, mgmt_ltpi_frm_tx.sv) 5. Updated Link Lost conditions in main controller and target FSM (modules: mgmt_phy_controller.sv and mgmt_phy_target.sv) |

Executive Summary

This User Guide provides guidelines how to create and run LTPI IP project. Project was created in Intel® Quartus® Prime Standard Edition Version 21.1.0 software and reference implementation was made for Intel® MAX® 10 FPGA devices family.

Table of Contents

| | |
|--|----|
| Introduction | 4 |
| 1. Build project instruction | 4 |
| 2. Run unit test instruction | 5 |
| 3. LTPI IP top modules | 5 |
| 3.1. LTPI top controller (SCM FPGA) – ltpi_top_controller.sv | 7 |
| 3.2. LTPI top target (HPM FPGA) - ltpi_top_target.sv | 10 |
| 4. SMBus Relay module | 12 |
| 4.1. Relay overview. | 12 |
| 4.2. SMBus relay controller (smbus_relay_controller.sv) | 13 |
| 4.2.1. Device Behavior FSM – Controller | 13 |
| 4.2.2. Relay Controller FSM | 16 |
| 4.3. SMBus relay target (smbus_relay_target.sv) | 17 |
| 4.3.1. Device Behavior FSM – Target | 17 |
| 4.3.2. Relay Target FSM | 18 |
| 4.4. Simulation of example I2C/SMBus transaction waveforms | 20 |
| 5. Data channel module | 20 |
| 5.1. Data channel mailbox Controller registers | 21 |
| 5.2. Programing model flowchart – data channel with mailbox. | 23 |
| 6. CSR module | 24 |
| 7. Resource utilization | 25 |
| 8. Glossary | 26 |
| 9. References | 26 |
| 10. License | 27 |
| 11. About Open Compute Foundation | 27 |



Introduction

LTPI_src.zip archive contains the following folders:

- logic
- quartus
- quartus_projects
- rtl
- tests_scripts
- tests

In **quartus** folder you will find pin assignments and timing constraints for LTPI controller and target project.

In **quartus_projects** folder you will find two subfolders with Quartus ® Prime projects files: one for LTPI controller device and second one for LTPI target device.

rtl and **logic** folder contains all modules essential to build the correct design.

In **tests** folder you will find unit tests for LTPI IP which are listed in LTPI_test_plan.xlsx

In **tests_script** folder you will find script which start simulation tool.

In the next chapters there is more detailed description of the LTPI implementation structure and internal modules design, so users could easily adapt it to their own use.

1. Build project instruction

1. Run Quartus ® Prime Software (used: Standard Edition Version 22.1.0).
2. Open one of projects from quartus_project directory:
 - controller/ltpi_top_controller_quartus.qpf
 - target/ltpi_top_target_quartus.qpf
3. Assign FPGA pin allocation file in quartus directory:
 - quartus_ltpi_controller/pinout_assignment_controller.tcl
 - quartus_ltpi_target /pinout_assignment_target.tcl

Or through Quartus ® Prime Software (Assignments -> Pin Planer)
4. Start Compilation (Processing -> Star Compilation)



2. Run unit test instruction

All project unit test were written using System Verilog Unit Test (SVUnit) framework.

All tests scripts give user the option to choose a simulation tool (between VCS[®] and ModelSim[®]/Questa[®]) by setting appropriate parameters inside script. E.g. `VCS_SIM_EN="ON"`, `MODELSIM_SIM_EN="OFF"`

1. Install SVUnit test from <http://agilesoc.com/open-source-projects/svunit/>
2. Set environment variable SVUNIT_INSTALL to SVUnit installation directory (e.g. `../svunit/3.34.1`)
`$export SVUNIT_INSTALL=<path to svunit directory>`
3. Compile Intel Simulation Model Libraries :
<https://www.intel.com/content/www/us/en/docs/programmable/683870/22-1/compiling-simulation-model-libraries.html>
4. Install simulation tool (e.g. VCS[®])
5. Edit script to use installed simulation tool
6. Run script from **tests_script** directory

3. LTPI IP top modules

LTPI IP can be configured as controller device (SCM) or as target device (HPM). Reference top modules for devices are respectively:

- `ltpi_top_controller.sv`
- `ltpi_top_target.sv`

Each of top module can be parameterized by user with parameter from table below:

Table 1 LTPI IP top module parameter

| Name | Value | Default value | Description |
|-------------------|------------------------|---------------|---|
| LL_GPIO_RST_VALUE | From 0 to $2^{16}-1$ | $2^{16}-1$ | Mask defining low latency GPIO pin value after reset. Bit position in this mask corresponds to respective LL GPIO index. |
| NL_GPIO_RST_VALUE | From 0 to $2^{1024}-1$ | $2^{1024}-1$ | Mask defining normal latency GPIO pin value after reset. Bit position in this mask corresponds to respective NL GPIO index. |

| | | | |
|-------------------------|----------------|------|---|
| CSR_LIGHT_VER_EN | 0 or 1 | 0 | When 0 LTPI IP CSR fully cover CSR in LTPI specification, when 1 LTPI IP CSR are limited. Refer to the Table 5 Registers disabled in CSR versions which CSRs are disabled in light version. |
| GPIO_EN | 0 or 1 | 1 | When 1 - GPIO module is turn on, when 0 it is turn off and NL_GPIO_CNT is not relevant. |
| NUM_OF_NLGPIO | From 1 to 1024 | 1024 | Number of normal latency GPIOs. |
| UART_EN | 0 or 1 | 1 | When 1 - UART module is turn on, when 0 it is turn off and UART_DEV is not relevant. |
| NUM_OF_UART_DEV | 1 or 2 | 2 | Number of UART devices. |
| SMBUS_EN | 0 or 1 | 1 | When 1 - SMBus module is turn on, when 0 it is turn off and SMBUS_DEV is not relevant. |
| NUM_OF_SMBUS_DEV | From 1 to 6 | 6 | Number of SMBUS devices. |
| DATA_CHANNEL_EN | 0 or 1 | 1 | When 1 – data channel module is turn on, when 0 it is turn off and DATA_CHANNEL_MAILBOX_EN is not relevant. |
| DATA_CHANNEL_MAILBOX_EN | 0 or 1 | 1 | When 1 data channel controller Avalon® bus is control through register described in Table 4 Data channel mailbox register description , otherwise data channels are directly connected to external interfaces. |

3.1. LTPI top controller (SCM FPGA) – ltpi_top_controller.sv

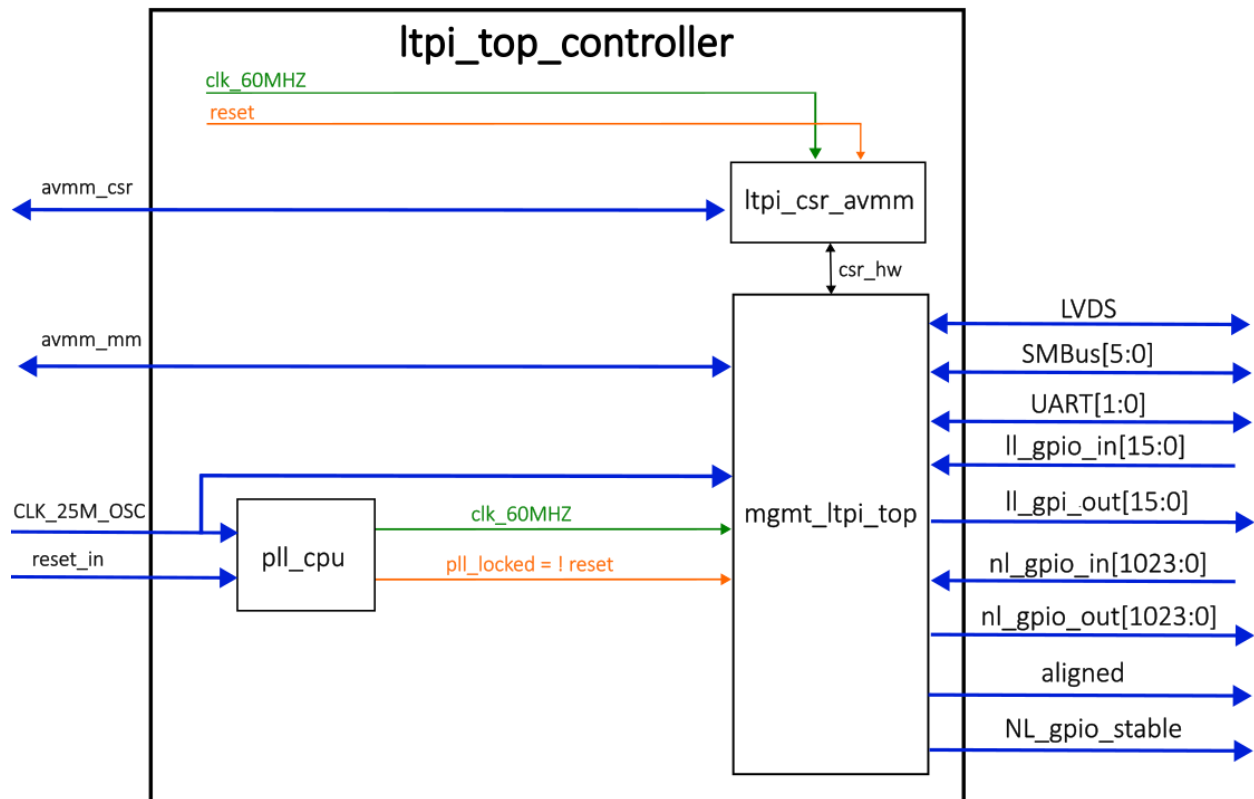


Figure 1 LTPI top controller diagram.

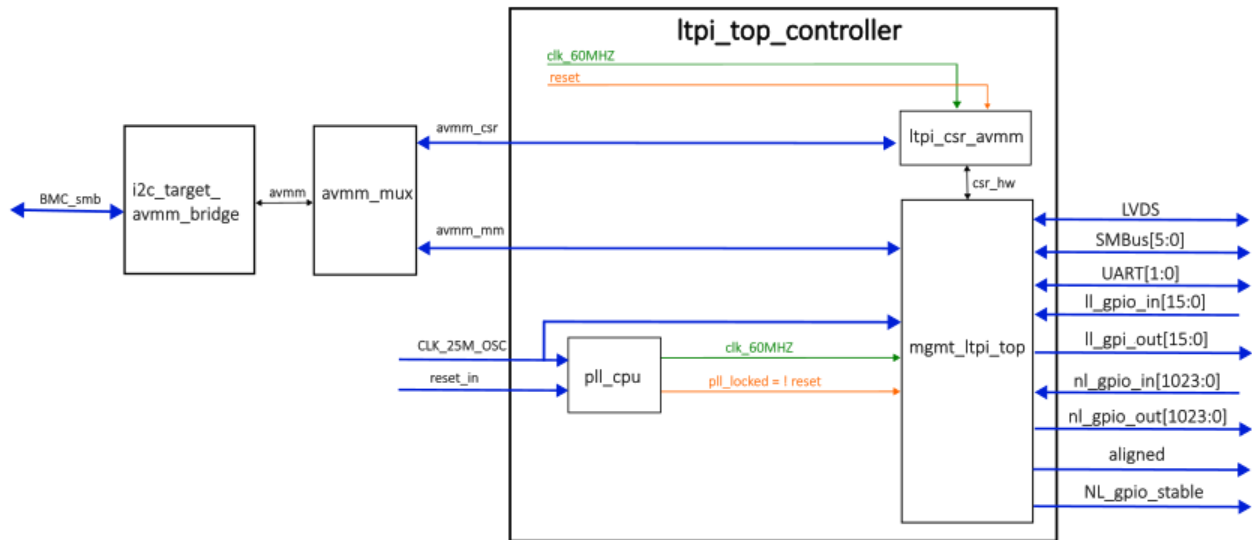


Figure 2 LTPI top controller diagram - BMC proposal connection

LTPI top controller with BMC proposal connection include:

- **pll_cpu** module - phase-locked loops with output for internal logic
- **i2c_target_avmm_bridge** module - i2c target ("slave") to Avalon® memory map bridge
- **avmm_mux** module - 1 to 4 Avalon® memory map bus multiplexer
- **ltpi_csr_avmm** module - LTPI Control and Status Register space with software access through Avalon® bus, and hardware access through **csr_hw** structure
- **mgmt_ltpi_top** module - Management LTPI phy and interfaces

Table 2 LTPI top controller module port definition

| Name | Type | Size | Description |
|--------------|--------|------|-------------------------------|
| CLK_25M_OSC | Input | 1 | 25MHZ reference clock input |
| reset_in | Input | 1 | Reset input |
| clk_60MHZ | Output | 1 | System clock output |
| pll_locked | Output | 1 | Internal pll locked |
| lvds_tx_data | Output | 1 | LVDS data transmitted signal |
| lvds_tx_clk | Output | 1 | LVDS clock transmitted signal |
| lvds_rx_data | Input | 1 | LVDS data received signal |
| lvds_rx_clk | Input | 1 | LVDS clock received signal |

| | | | |
|------------------|--------|------|--|
| aligned | Output | 1 | HIGH – Received frames are decoded correctly. |
| NL_gpio_stable | Output | 1 | Normal latency GPIO Stable indication. It is HIGH only for one clock cycle, when all normal latency GPIOs were received. |
| smb_scl | Inout | 6 | I2C/SMBus clock signal sent/received through LTPI |
| smb_sda | Inout | 6 | I2C/SMBus data signal sent/received through LTPI |
| ll_gpio_in | Input | 16 | Low Latency GPIO input signal sent through LTPI |
| ll_gpio_out | Output | 16 | Low Latency GPIO output signal received through LTPI |
| nl_gpio_in | Input | 1024 | Normal Latency GPIO input signal sent through LTPI |
| nl_gpio_out | Output | 1024 | Normal Latency GPIO output signal received through LTPI |
| uart_rxd | Input | 2 | UART receiver signal |
| uart_cts | Input | 2 | UART CST flow control signal |
| uart_txd | Output | 2 | UART transmitter signal |
| uart_rts | Output | 2 | UART RTS flow control signal |
| avmm_slv_addr | Input | 16 | Data channel LTPI - AVMM address signal |
| avmm_slv_read | Input | 1 | Data channel LTPI - AVMM read signal |
| avmm_slv_write | Input | 1 | Data channel LTPI - AVMM write signal |
| avmm_slv_wdata | Input | 32 | Data channel LTPI - AVMM write data signal |
| avmm_slv_byteen | Input | 4 | Data channel LTPI - AVMM byte enable signal |
| avmm_slv_rdata | Output | 32 | Data channel LTPI - AVMM read data valid signal |
| avmm_slv_rdvalid | Output | 1 | Data channel LTPI - AVMM read data valid signal |
| avmm_slv_waitrq | Output | 1 | Data channel LTPI - AVMM wait request signal |
| avmm_csr_addr | Input | 16 | LTPI CSR - AVMM address signal |
| avmm_csr_read | Input | 1 | LTPI CSR - AVMM read signal |
| avmm_csr_write | Input | 1 | LTPI CSR - AVMM write signal |
| avmm_csr_wdata | Input | 32 | LTPI CSR - AVMM write data signal |
| avmm_csr_byteen | Input | 4 | LTPI CSR - AVMM byte enable signal |
| avmm_csr_rdata | Output | 32 | LTPI CSR - AVMM read data valid signal |
| avmm_csr_rdvalid | Output | 1 | LTPI CSR - AVMM read data valid signal |
| avmm_csr_waitrq | Output | 1 | LTPI CSR - AVMM wait request signal |

3.2. LTPI top target (HPM FPGA) - ltpi_top_target.sv

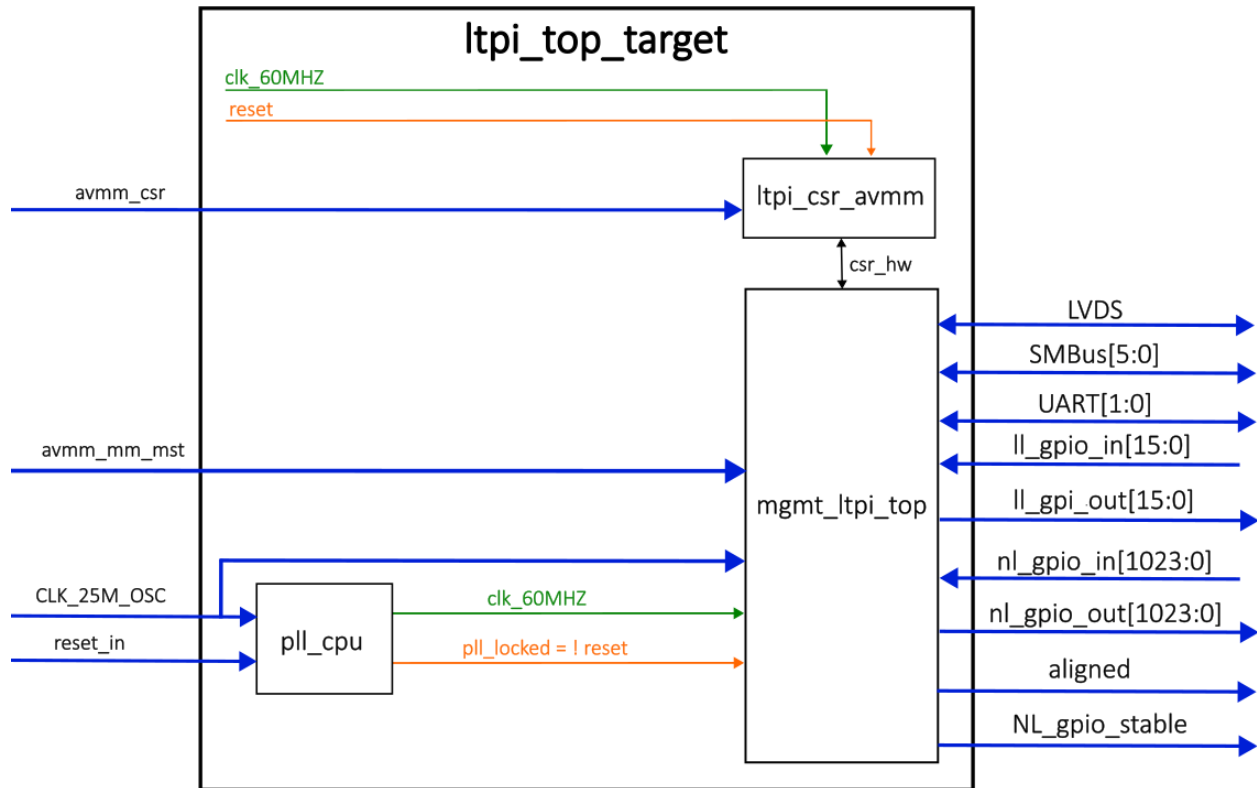


Figure 3 LTPI top target diagram.

LTPI top target include:

- pll_cpu module- phase-locked loops with output for internal logic
- ltpi_csr_avmm module – LTPI Control and Status Register space with software access through Avalon® bus, and hardware access through csr_hw structure
- mgmt_ltpi_top module - Management LTPI phy and interfaces

Table 3 LTPI top target module port definition

| Name | Type | Size | Description |
|-------------|-------|------|-----------------------------|
| CLK_25M_OSC | Input | 1 | 25MHZ reference clock input |
| reset_in | Input | 1 | Reset input |

| | | | |
|------------------|--------|------|--|
| clk_60MHZ | Output | 1 | System clock output |
| pll_locked | Output | 1 | Internall pll locked |
| lvds_tx_data | Output | 1 | LVDS data transmitted signal |
| lvds_tx_clk | Output | 1 | LVDS clock transmitted signal |
| lvds_rx_data | Input | 1 | LVDS data received signal |
| lvds_rx_clk | Input | 1 | LVDS clock received signal |
| aligned | Output | 1 | HIGH – Received frames are decoded correctly. |
| NL_gpio_stable | Output | 1 | Normal latency GPIO Stable indication. It is HIGH only for one clock cycle, when all normal latency GPIOs were received. |
| smb_scl | Inout | 6 | I2C/SMBus clock signal sent/received through LTPI |
| smb_sda | Inout | 6 | I2C/SMBus data signal sent/received through LTPI |
| ll_gpio_in | Input | 16 | Low Latency GPIO input signal sent through LTPI |
| ll_gpio_out | Output | 16 | Low Latency GPIO output signal received through LTPI |
| nl_gpio_in | Input | 1024 | Normal Latency GPIO input signal sent through LTPI |
| nl_gpio_out | Output | 1024 | Normal Latency GPIO output signal received through LTPI |
| uart_rxd | Input | 2 | UART receiver signal |
| uart_cts | Input | 2 | UART CST flow control signal |
| uart_txd | Output | 2 | UART transmitter signal |
| uart_rts | Output | 2 | UART RTS flow control signal |
| avmm_mst_addr | Output | 16 | Data channel LTPI - AVMM address signal |
| avmm_mst_read | Output | 1 | Data channel LTPI - AVMM read signal |
| avmm_mst_write | Output | 1 | Data channel LTPI - AVMM write signal |
| avmm_mst_wdata | Output | 32 | Data channel LTPI - AVMM write data signal |
| avmm_mst_byteen | Output | 4 | Data channel LTPI - AVMM byte enable signal |
| avmm_mst_rdata | Input | 32 | Data channel LTPI - AVMM read data valid signal |
| avmm_mst_rdvalid | Input | 1 | Data channel LTPI - AVMM read data valid signal |
| avmm_mst_waitrq | Input | 1 | Data channel LTPI - AVMM wait request signal |
| avmm_csr_addr | Input | 16 | LTPI CSR - AVMM address signal |
| avmm_csr_read | Input | 1 | LTPI CSR - AVMM read signal |
| avmm_csr_write | Input | 1 | LTPI CSR - AVMM write signal |
| avmm_csr_wdata | Input | 32 | LTPI CSR - AVMM write data signal |
| avmm_csr_byteen | Input | 4 | LTPI CSR - AVMM byte enable signal |

| | | | |
|------------------|--------|----|--|
| avmm_csr_rdata | Output | 32 | LTPI CSR - AVMM read data valid signal |
| avmm_csr_rdvalid | Output | 1 | LTPI CSR - AVMM read data valid signal |
| avmm_csr_waitrq | Output | 1 | LTPI CSR - AVMM wait request signal |

4. SMBus Relay module

4.1. Relay overview.

The I2C/SMBus SDL and SDA signal states are being captured and controlled by so called I2C/SMBus Relays defined by LTPI specification. The I2C/SMBus Relay block diagram is presented on the **Figure 4**.

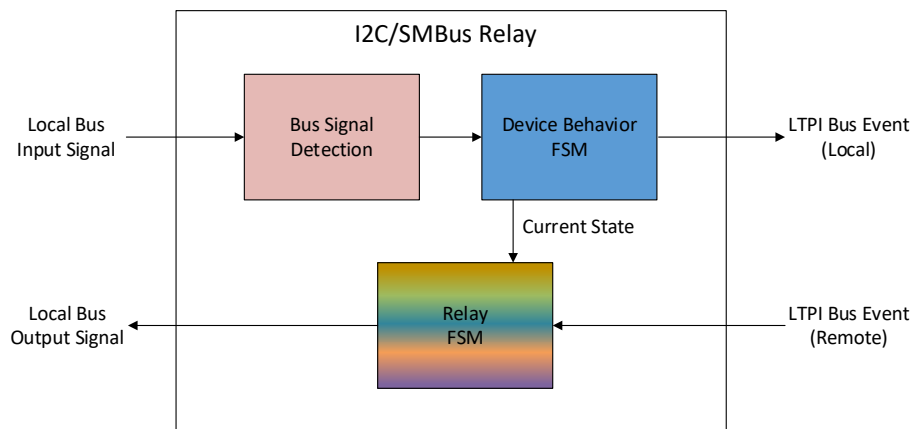


Figure 4 SMBus Relay Block Diagram

There are two types of SMBus relays – Controller and Target. Controller SMBus relay is located on SCM FPGA and is connected to controller SMBus device (mostly BMC). Target Relay is a part of HPM FPGA and it is connected to target devices (e.g. temperature sensors, I2C memory's, power management devices).

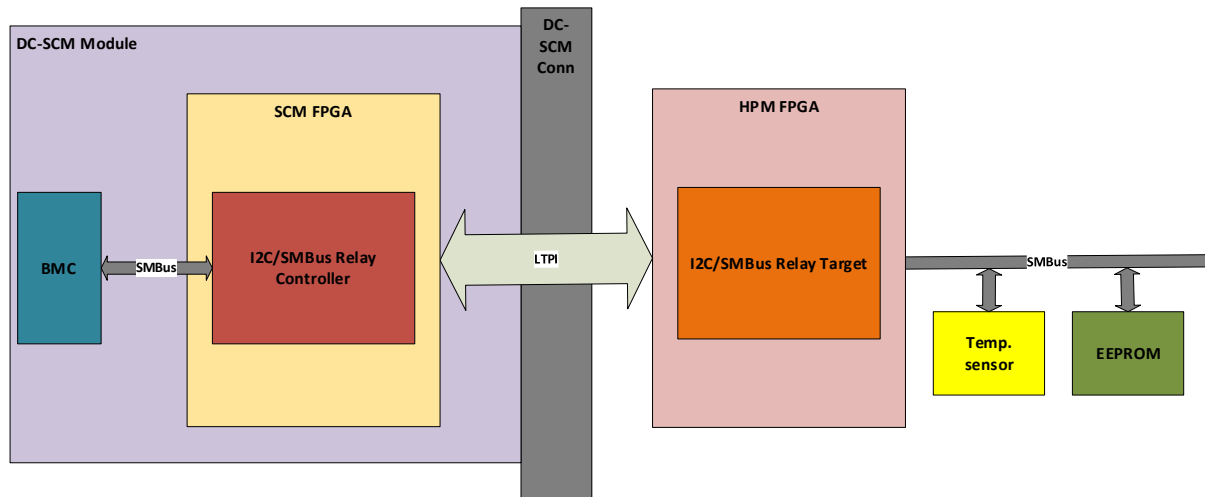


Figure 5 SMBus relays connection

There are two FSM in SMBus Relay (with different behavior for Target and Controller Devices):

- Device Behavior FSM - Determine the state of the controller and target SMBus
- Relay FSM - Determine the state of the relay, and drive the target and controller SMBus signals based on the state of the controller and target busses

Main differences between Target and Controller behavior FSM is a way to change states.

In Device Behavior FSM Controller state changes are forced by changes on SMBus line. In the Device Behavior FSM Target - state changes are mainly forced by events which are sent from SCM - Controller devices (ioc_remote).

4.2. SMBus relay controller (smbus_relay_controller.sv)

This module implements an SMBus relay between a single controller and multiple target devices.

The module uses clock stretching on the interface from the SMBus controller to allow time for the target to respond with ACK and read data.

4.2.1. Device Behavior FSM – Controller

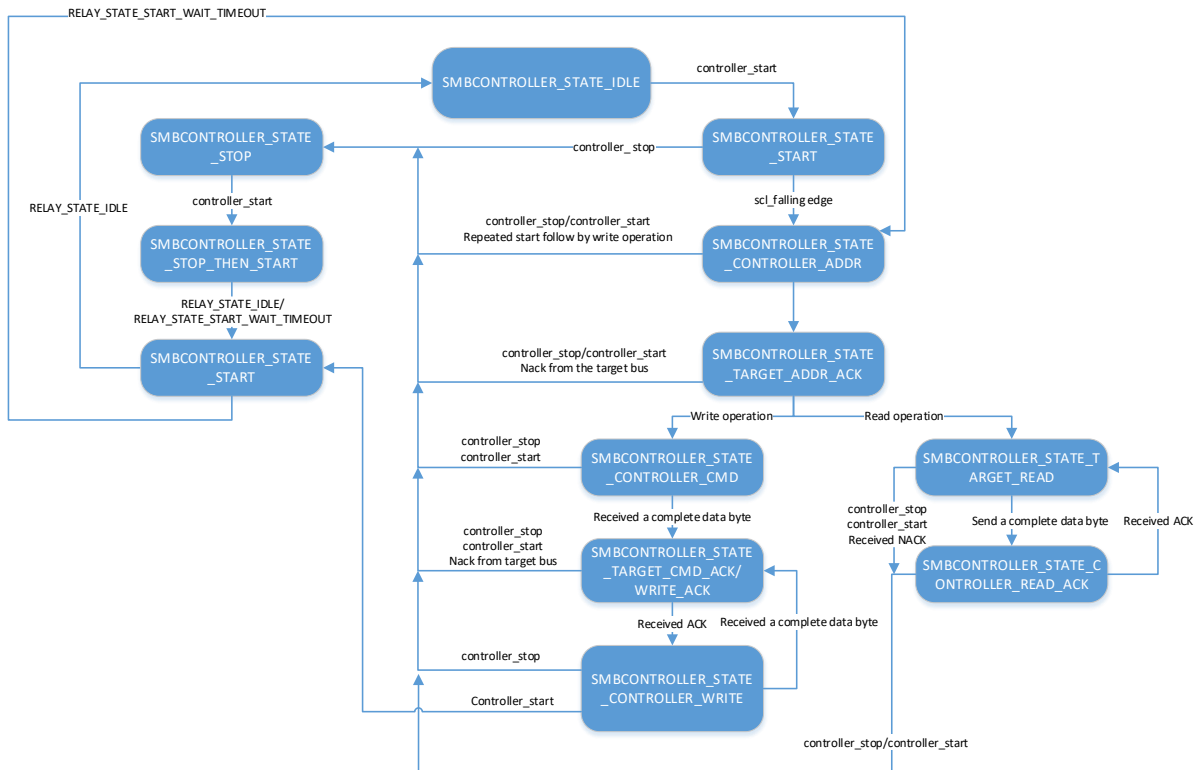


Figure 6 Device Behavior Controller FSM

SMBCONTROLLER_STATE_IDLE - This is the reset state. Wait here until a valid START condition is detected on the controller bus.

SMBCONTROLLER_STATE_START - A start condition was detected on the controller bus, the FSM stays here and clock-stretches the controller bus as required until the target bus has 'caught up' and then issued a start condition.

SMBCONTROLLER_STATE_CONTROLLER_ADDR - In this state the 7 bits target address and the read/write bit are received. Leave this state when all 8 bits have been received and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_ADDR_ACK - SCL signal is driven low during this state to clock-stretch while awaiting an ACK from the target bus. Always enter this state after a SCL falling edge. Leave this state on SCL falling edge and when the ack/nack bit has been sent or when unexpected stop/start condition occurred.

SMBCONTROLLER_STATE_CONTROLLER_CMD - Received the 8 bits SMBus command (the first data byte after the address is called the 'command' in SMBus Specification). Always enter this state after an SCL falling edge. Leave this state when all 8 bits have been received and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_CMD_ACK, SMBCONTROLLER_STATE_TARGET_WRITE_ACK - SCL signal is driven low during this state to clock-stretch while awaiting an ACK from the target bus. Always enter this state after an SCL falling edge. Leave this state when the ack/nack bit has been sent and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_CONTROLLER_WRITE - Enter the state after write command - received a byte to write to the target device. Always enter this state after a SCL falling edge, leave this state when 8 bytes were sent to target devices and clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_READ - Enter this state after read command and after a SCL falling edge, leave this state after received byte from target device.

SMBCONTROLLER_STATE_CONTROLLER_READ_ACK - Enter this state after a SCL falling edge, leave this state when the ack/nack bit has been received and the controller drive SCL low again.

SMBCONTROLLER_STATE_STOP - Enter this state to indicate a STOP condition should be sent to the target bus. Once the STOP has been sent on the target bus, we return to the idle state and wait for another start condition. We can enter this state if a stop condition has been received on the controller bus, or if we expect a start condition on the controller busses. We do not wait to see a stop condition on the controller bus before issuing the stop on the target bus and proceeding to the IDLE state

SMBCONTROLLER_STATE_STOP_THEN_START - While waiting to send a STOP on the target bus, we receive a new start condition on the controller busses. Must finish sending the stop condition on the target bus, then send a start condition on the target bus.

4.2.2. Relay Controller FSM

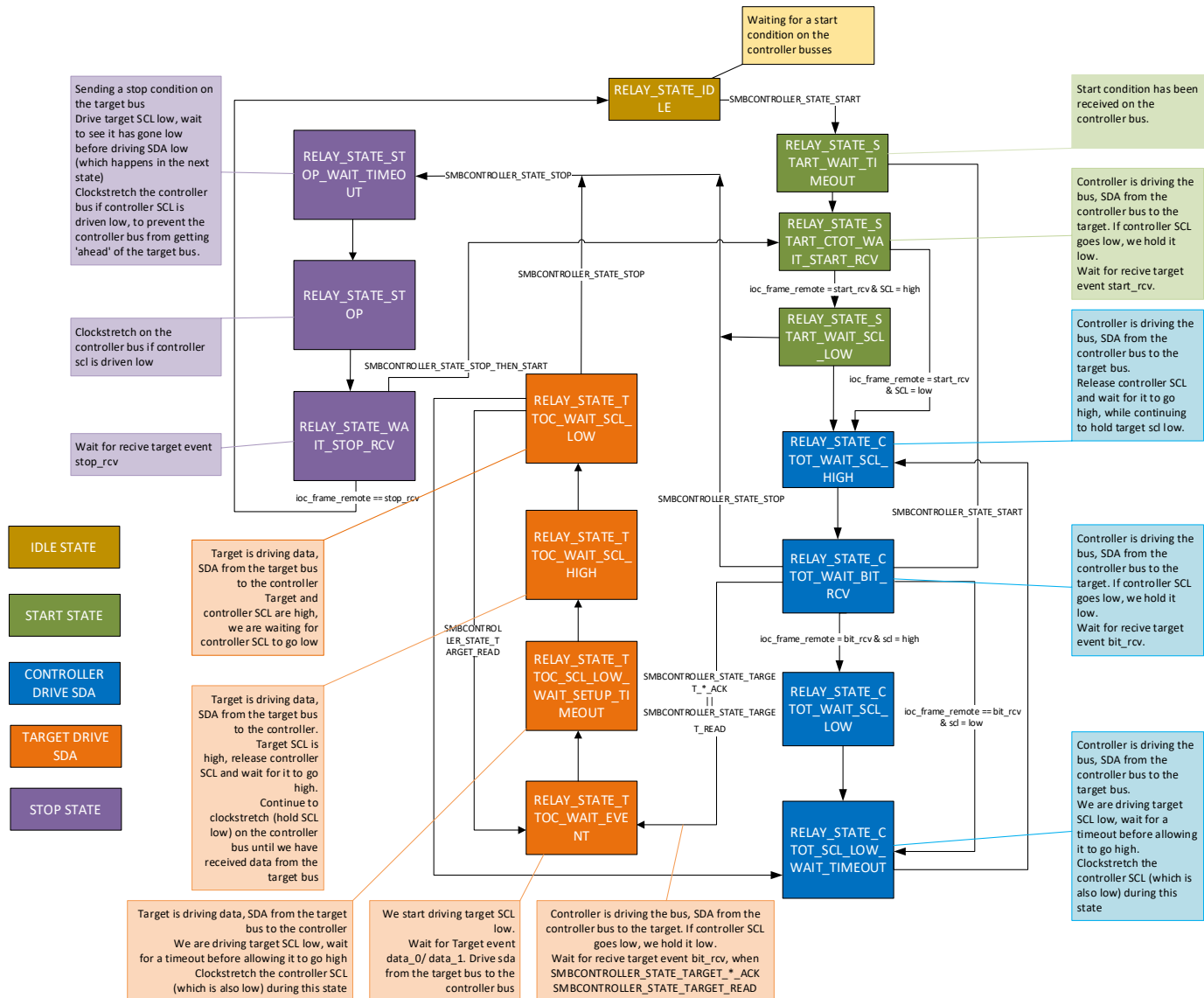


Figure 7 Relay Controller FSM

4.3. SMBus relay target (smbus_relay_target.sv)

4.3.1. Device Behavior FSM – Target

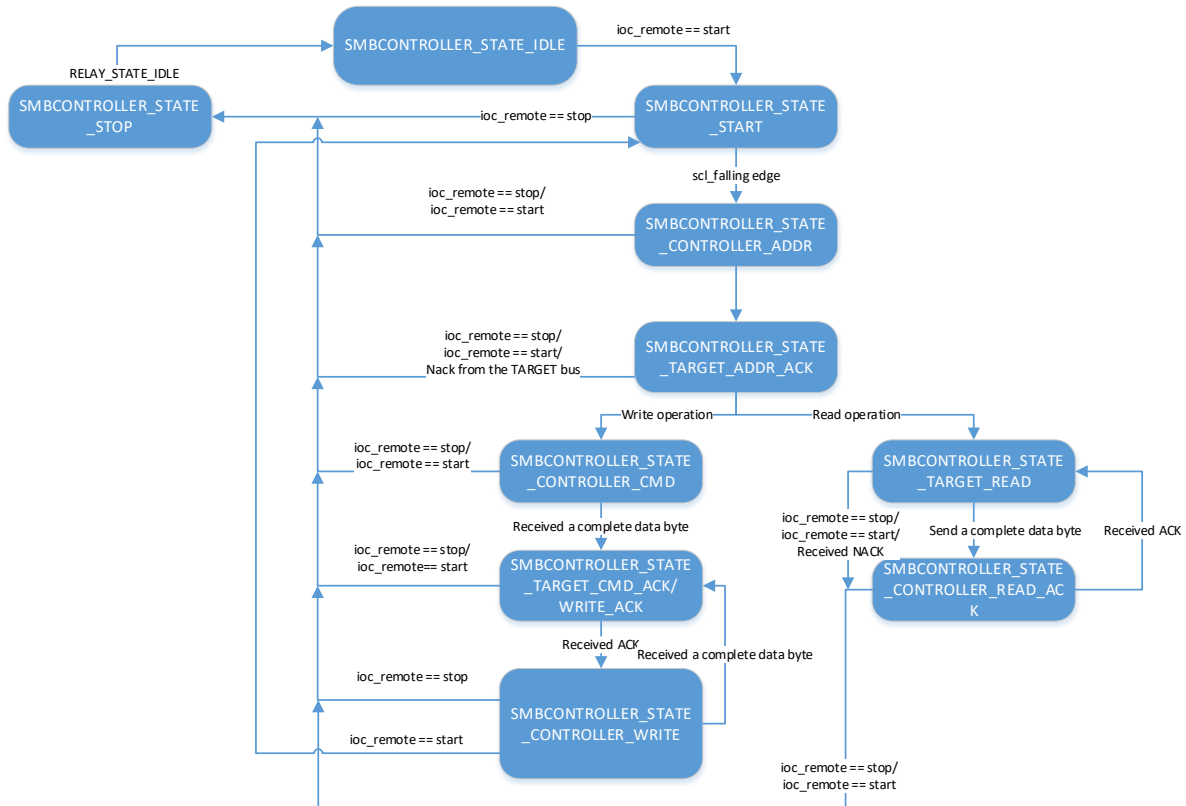


Figure 8 Device Behavior Target FSM

SMBCONTROLLER_STATE_IDLE - This is the reset state. Wait here until a valid START condition is detected on the controller bus.

SMBCONTROLLER_STATE_START - A start condition was detected on the controller bus.

SMBCONTROLLER_STATE_CONTROLLER_ADDR - In this state the 7 bits target address and the read/write bit are received. Leave this state when all 8 bits have been received and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_ADDR_ACK - Enter this state after a SCL falling edge Target send ACK on the bus, leave this state when the ack/nack bit has been sent and the clock has been driven low.

SMBCONTROLLER_STATE_CONTROLLER_CMD - Received the 8 bits SMBus command (the first data byte after the address is called the 'command' in SMBus Specification). Always enter this state after an SCL falling edge. Leave this state when all 8 bits have been received and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_CMD_ACK, SMBCONTROLLER_STATE_TARGET_WRITE_ACK - Enter this state on clock falling edge and after target received command byte. Target send ACK on bus, leave this state when the ack/nack bit has been sent and the clock has been driven low.

SMBCONTROLLER_STATE_CONTROLLER_WRITE - Enter the state after write command - received a byte to write to the target device. Always enter this state after a SCL falling edge, leave this state when 8 bites were sent to target devices and clock has been driven low again by the controller.

SMBCONTROLLER_STATE_TARGET_READ - Enter this state after read command, and after a SCL falling edge, leave this state after sent data byte from target device.

SMBCONTROLLER_STATE_CONTROLLER_READ_ACK - Enter this state after a SCL falling edge, leave this state when the ack/nack bit has been received and the clock has been driven low again by the controller.

SMBCONTROLLER_STATE_STOP - - Enter this state to indicate a STOP condition should be sent to the target bus. Once the STOP has been sent on the target bus, we return to the idle state and wait for another start condition. We can enter this state if a stop condition has been received on the controller bus, or if we expect a start condition on the controller busses. We do not wait to see a stop condition on the controller bus before issuing the stop on the target bus and proceeding to the IDLE state

4.3.2.Relay Target FSM

Duration each of *_WAIT_TIMEOUT states depends from SMBus SPEED (100k/400k). Timing parametr (timeout) determine how long to hold target SCL low or high before proceeding to next state.

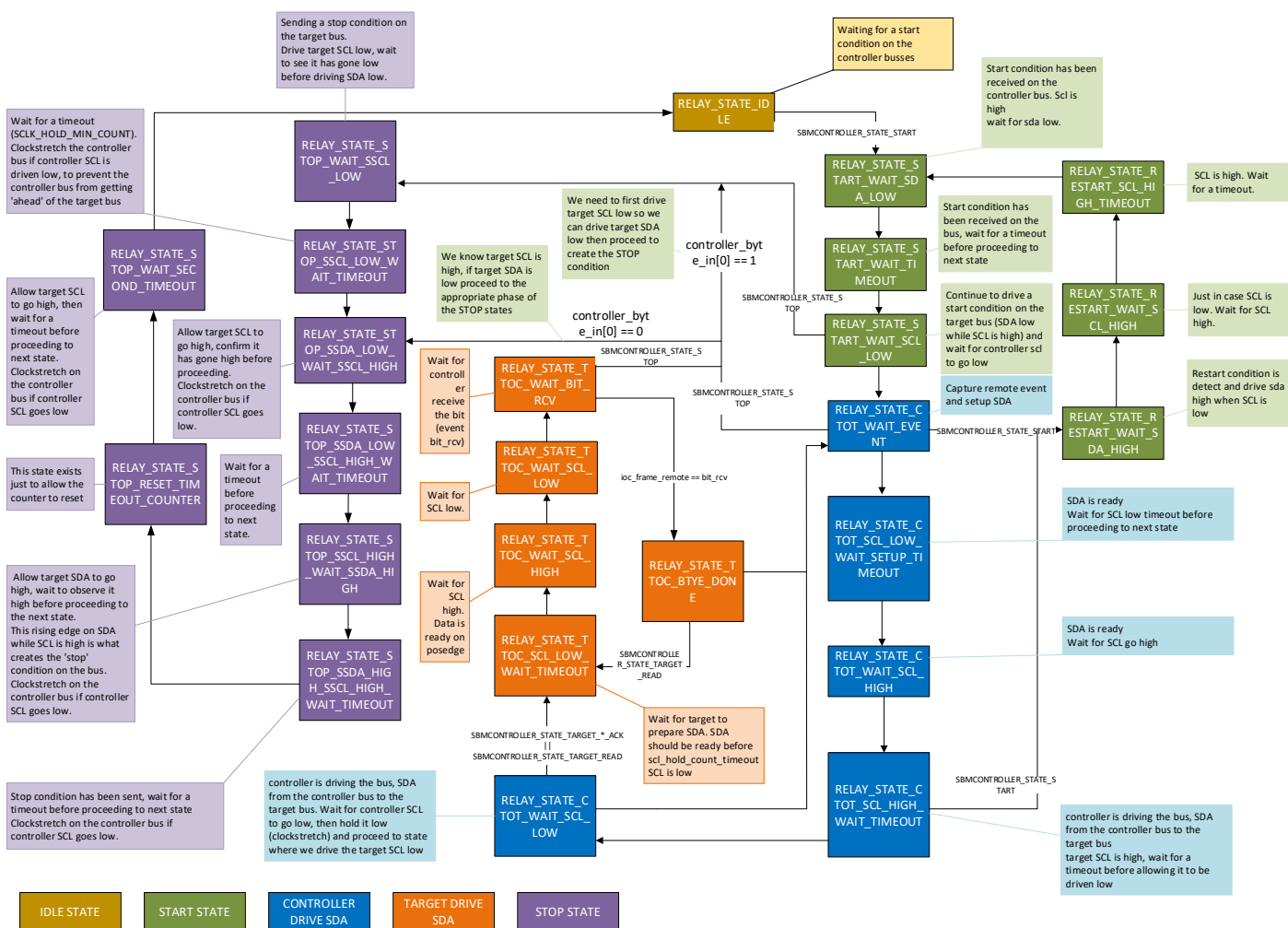


Figure 9 Relay Target FSM

4.4. Simulation of example I2C/SMBus transaction waveforms



Figure 10 I2C/SMBus transaction waveforms

5. Data channel module

To use data channel module user must turn it on (parameter DATA_CHANNEL_EN = 1). There are two data channel options implemented – with or without mailbox (parameter DATA_CHANNEL_MAILBOX_EN). Parametrization make the LTPI IP more flexible by allowing the user to decide, how data channel is connected and used.

When mailbox option is turned on, data channel controller module is guided by register from **Table 4 Data channel mailbox register description** otherwise data channels are directly connected to external interfaces – simplest solution with less Logic Element utilization.

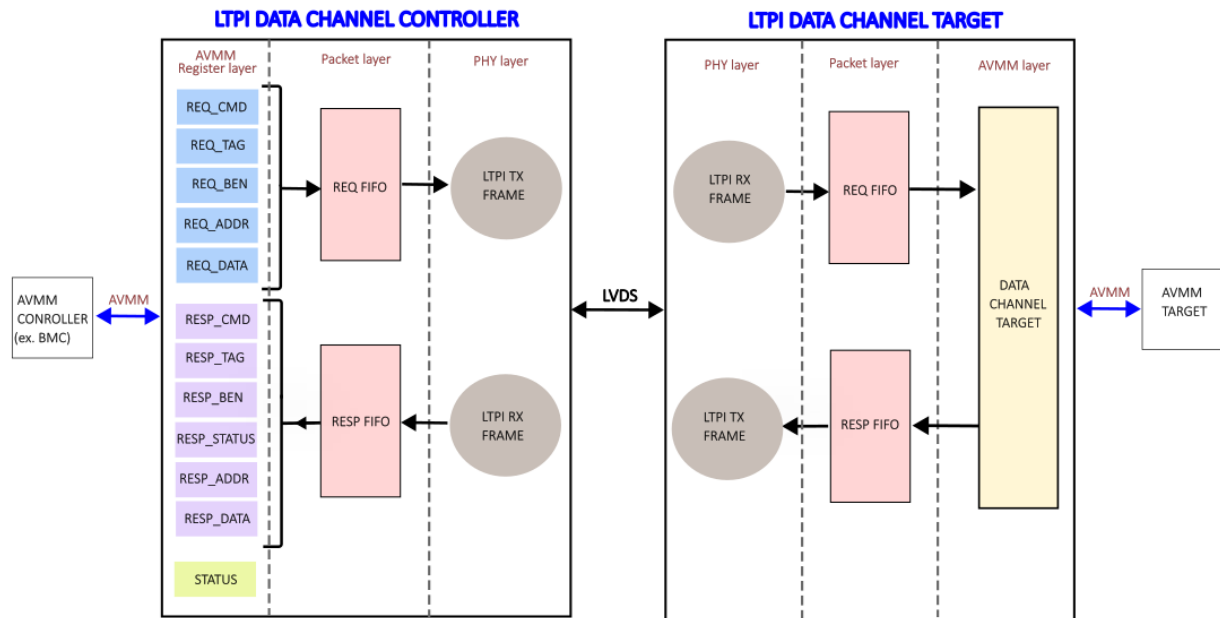


Figure 11 LTPI Data Channel with mailbox option Overview

5.1. Data channel mailbox Controller registers

Data channel mailbox Controller registers are available through Avalon[®] interface connected to LTPI IP used as Controller device.

Table 4 Data channel mailbox register description

| Offset | Name | Type | Bit field Definition |
|--------|-----------|------|-------------------------|
| 0x400 | REQ_CMD | RW | Request Command |
| 0x401 | REQ_TAG | RW | Request TAG |
| 0x402 | REQ_BEN | RW | Request Byte Enable |
| 0x404 | REQ_ADDR0 | RW | Request Address [7:0] |
| 0x405 | REQ_ADDR1 | RW | Request Address [15:8] |
| 0x406 | REQ_ADDR2 | RW | Request Address [23:16] |
| 0x407 | REQ_ADDR3 | RW | Request Address [31:24] |
| 0x408 | REQ_DATA0 | RW | Request Data [7:0] |
| 0x409 | REQ_DATA1 | RW | Request Data [15:8] |

| | | | | |
|-------|-------------|----|--------------------------|---|
| 0x40A | REQ_DATA2 | RW | Request Data [23:16] | |
| 0x40B | REQ_DATA3 | RW | Request Data [31:24] | |
| 0x40C | RESP_CMD | R | Response Command | |
| 0x40D | RESP_TAG | R | Response TAG | |
| 0x40E | RESP_BEN | R | Response Byte Enable | |
| 0x40F | RESP_STATUS | R | Response Status | |
| 0x410 | RESP_ADDR0 | R | Response Address [7:0] | |
| 0x411 | RESP_ADDR1 | R | Response Address [15:8] | |
| 0x412 | RESP_ADDR2 | R | Response Address [23:16] | |
| 0x413 | RESP_ADDR3 | R | Response Address [31:24] | |
| 0x414 | RESP_DATA0 | R | Response Data [7:0] | |
| 0x415 | RESP_DATA1 | R | Response Data [15:8] | |
| 0x416 | RESP_DATA2 | R | Response Data [23:16] | |
| 0x417 | RESP_DATA3 | R | Response Data [31:24] | |
| 0x418 | STATUS | R | Response Ready | 0 |
| | | RW | Response Read | 1 |
| | | R | Request Ready | 2 |
| | | RW | Request Write | 3 |

5.2. Programing model flowchart – data channel with mailbox.

The following flowchart illustrates the recommended programing models for sending a request and receiving response when the AVMM mailbox is being used for example by the BMC through I2C to AVMM bridge.

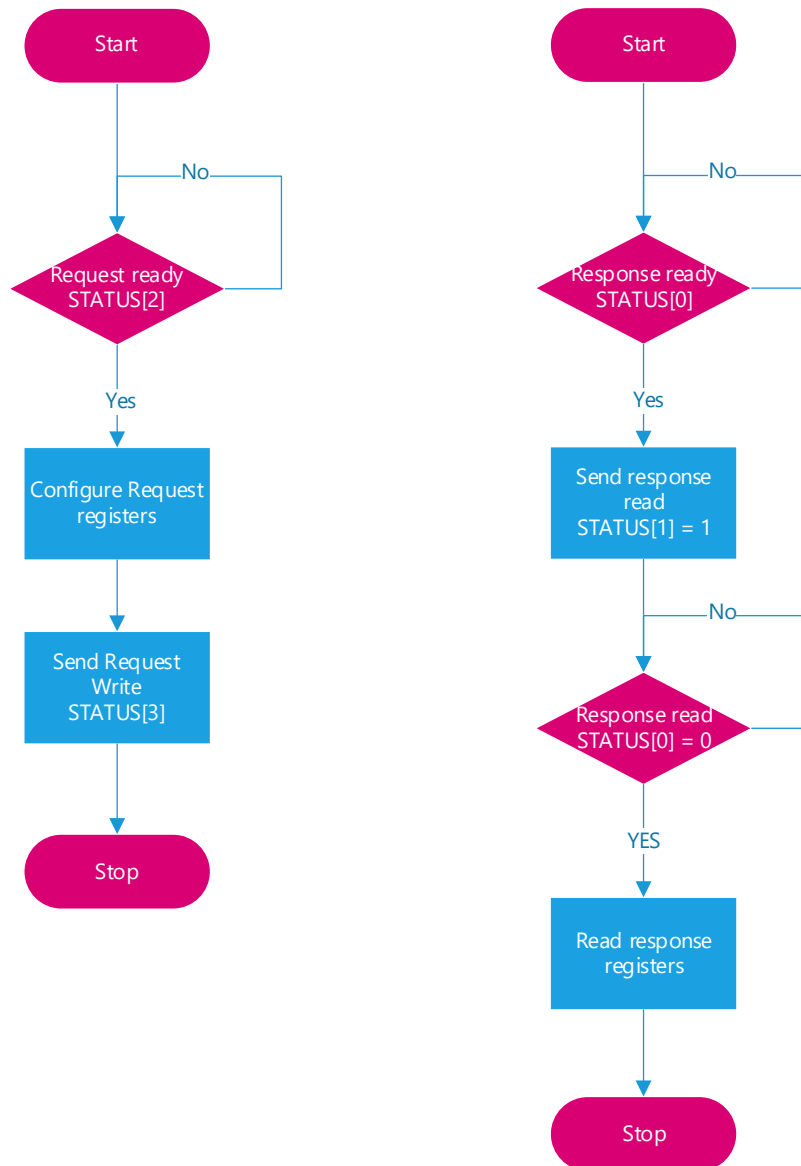


Figure 12 Request/Response programing model flowchart

6. CSR module

Parametrization make the LTPI IP more flexible by allowing the user to decide, how to use CSR module.

There are two versions of LTPI CSR module: full – with access to all registers in CSR memory space and light – with limited access to CSR memory space and less Logic Element utilization.

CSR files for both versions were generated by using PeakRDL-regblock toolchain and written in SystemRDL language - <https://accelera.org/downloads/standards/systemrdl>.

User can easily change the contend of CSR registers (e.g. default value or excluded registers) by generating RDL file or editing predefined files.

PeakRDL-regblock installing instruction: <https://peakrdl-regblock.readthedocs.io/en/latest/index.html#>

In rtl -> modules ->RDL directory there is script which can be used to generate CSR SystemVerilog package files (csr_gen.py) and there are two predefine RDL files:

- CSR_regs_full.rdl
- CSR_regs_light.rdl

In **Table 5 Registers disabled in CSR versions** you can find description which registers are included in each predefined CSR versions.

Table 5 Registers disabled in CSR versions

| Address Offset | Registers Name | CSR Version | |
|----------------|---|-------------|-------|
| | | Full | Light |
| 0x00 | LTPI Link Status | ✓ | ✓ |
| 0x04 | LTPI Detect Capabilities Local | ✓ | ✓ |
| 0x08 | LTPI Detect Capabilities Remote | ✓ | ✓ |
| 0x0C | LTPI Platform ID Local | ✓ | ✓ |
| 0x10 | LTPI Platform ID Remote | ✓ | ✗ |
| 0x14 | LTPI Advertise Capabilities Local Low | ✓ | ✓ |
| 0x18 | LTPI Advertise Capabilities Local High | ✓ | ✓ |
| 0x1C | LTPI Advertise Capabilities Remote Low | ✓ | ✓ |
| 0x20 | LTPI Advertise Capabilities Remote High | ✓ | ✓ |
| 0x24 | LTPI Default Configuration Low | ✓ | ✓ |

| | | | |
|------|---|---|---|
| 0x28 | LTPI Default Configuration High | ✓ | ✓ |
| 0x2C | LTPI Link Alignment Error Counter | ✓ | ✗ |
| 0x30 | LTPI Link Lost Error Counter | ✓ | ✗ |
| 0x34 | LTPI CRC Error Counter | ✓ | ✗ |
| 0x38 | Unknown Comma Error Counter | ✓ | ✗ |
| 0x3C | Link Speed Timeout Error Counter | ✓ | ✗ |
| 0x40 | Link Configure/Accept Timeout Error Counter | ✓ | ✗ |
| 0x44 | Link Training RX Frames Counter Low | ✓ | ✗ |
| 0x48 | Link Training RX Frames Counter High | ✓ | ✗ |
| 0x4C | Link Training TX Frames Counter Low | ✓ | ✗ |
| 0x50 | Link Training TX Frames Counter High | ✓ | ✗ |
| 0x54 | Operational RX Frames Counter | ✓ | ✗ |
| 0x58 | Operational TX Frames Counter | ✓ | ✗ |
| 0x80 | LTPI Link Control | ✓ | ✓ |

7. Resource utilization

Table 6 Resource utilization - LTPI IP Controller

| MODULES | Submodule | Submodule options | LTPI IP settings | LC | Reg. | LTPI IP settings | LC | Reg. | LTPI IP settings - without dynamic PLL | LC | Reg. |
|---------------|-------------------|--------------------|------------------|------|------|------------------|------|------|--|------|------|
| ltpi_csr | | full version | EN | 1954 | 1109 | | | | | | |
| | | light version | | | | EN | 497 | 360 | EN | 497 | 360 |
| mgmt_ltpi_top | mgmt_data_channel | with mailbox | EN | 1318 | 1084 | | | | | | |
| | | direct mm | | | | EN | 378 | 296 | EN | 380 | 296 |
| | mgmt_gpio | max:1024 NL, 16 LL | EN - 1024 NL | 36 | 14 | EN - 16 NL | 37 | 14 | EN - 16 NL | 36 | 14 |
| | mgmt_phy_top | | | 3066 | 1623 | | 1840 | 1098 | | 1590 | 950 |
| | mgmt_smbus | max: 6 smbus | EN - 6 SMBUS | 1870 | 715 | EN - 1 SMBUS | 307 | 119 | EN - 1 SMBUS | 307 | 119 |
| | mgmt_uart | max: 2 uart | EN - 2 UART | 31 | 16 | EN - 2 UART | 35 | 22 | EN - 2 UART | 35 | 22 |
| | | | | 8275 | 4561 | | 3094 | 1909 | | 2845 | 1761 |

Table 7 Resource utilization - LTPI IP Target

| MODULES | Submodule | Submodule options | LTPI IP settings | LC | Reg. | LTPI IP settings | LC | Reg. | LTPI IP settings - without dynamic PLL | LC | Reg. |
|---------------|-------------------|--------------------|------------------|-------------|-------------|------------------|------|-------------|--|-------------|-------------|
| ltpi_csr | | full version | EN | 1967 | 1113 | | | | | | |
| | | light version | | | | EN | 507 | 364 | EN | 508 | 364 |
| mgmt_ltpi_top | mgmt_data_channel | with mailbox | EN | 683 | 591 | | | | | | |
| | | direct mm | | | | EN | 378 | 293 | EN | 376 | 293 |
| | mgmt_gpio | max:1024 NL, 16 LL | EN - 1024 NL | 39 | 15 | EN - 16 NL | 39 | 15 | EN - 16 NL | 39 | 15 |
| | mgmt_phy_top | | | 3066 | 1626 | | 1955 | 1162 | | 1698 | 1014 |
| | mgmt_smbus | max: 6 smbus | EN - 6 SMBUS | 2009 | 745 | EN - 1 SMBUS | 326 | 124 | EN - 1 SMBUS | 328 | 124 |
| | mgmt_uart | max: 2 uart | EN - 2 UART | 30 | 16 | EN - 2 UART | 35 | 22 | EN - 2 UART | 37 | 22 |
| | | | | 7794 | 4106 | | | 3240 | 1980 | 2986 | 1832 |

8. Glossary

LVDS - Low-Voltage Differential Signaling

LTPI - LVDS Tunneling Protocol & Interface

OCF - Open Compute Project

DC-SCM - Data Center Secure Control Module

SCM - Secure Control Module

HPM - Host Processor Module

BMC – Board Management Controller

FSM – Finite State Machine

CSR – Control and Status Register

AVMM - Avalon® Memory-Mapped

9. References

DC-SCM 2.0 LVDS Tunneling Protocol & Interface Specification (LTPI):

https://drive.google.com/file/d/1PUJctmwSspVF3j01LZGyhchBmK8YM_Cu/view

SystemRDL: <https://accelera.org/downloads/standards/systemrdl>



PeakRDL-regblock: <https://peakrdl-regblock.readthedocs.io/en/latest/index.html#>

SVUnit : <http://agilesoc.com/open-source-projects/svunit/>

10. License

OCP encourages participants to share their proposals, specifications and designs with the community. This is to promote openness and encourage continuous and open feedback. It is important to remember that by providing feedback for any such documents, whether in written or verbal form, that the contributor or the contributor's organization grants OCP and its members irrevocable right to use this feedback for any purpose without any further obligation.

It is acknowledged that any such documentation and any ancillary materials that are provided to OCP in connection with this document, including without limitation any white papers, articles, photographs, studies, diagrams, contact information (together, "Materials") are made available under the Creative Commons Attribution-ShareAlike 4.0 International License found here: <https://creativecommons.org/licenses/by-sa/4.0/>, or any later version, and without limiting the foregoing, OCP may make the Materials available under such terms.

As a contributor to this document, all members represent that they have the authority to grant the rights and licenses herein. They further represent and warrant that the Materials do not and will not violate the copyrights or misappropriate the trade secret rights of any third party, including without limitation rights in intellectual property. The contributor(s) also represent that, to the extent the Materials include materials protected by copyright or trade secret rights that are owned or created by any third-party, they have obtained permission for its use consistent with the foregoing. They will provide OCP evidence of such permission upon OCP's request. This document and any "Materials" are published on the respective project's wiki page and are open to the public in accordance with OCP's Bylaws and IP Policy. This can be found at <http://www.opencompute.org/participate/legal-documents/>. If you have any questions please contact OCP.

11. About Open Compute Foundation

At the core of the Open Compute Project (OCP) is its Community of hyperscale data center operators, joined by telecom and colocation providers and enterprise IT users, working with vendors to develop open innovations that, when embedded in product are deployed from the cloud to the edge. The OCP Foundation is responsible for fostering and serving the OCP Community to meet the market and shape the future, taking hyperscale led innovations to everyone. Meeting the market is accomplished through open designs and best practices, and with data center facility and IT equipment embedding OCP Community-developed innovations for efficiency, at-scale operations and sustainability. Shaping the future includes





investing in strategic initiatives that prepare the IT ecosystem for major changes, such as AI & ML, optics, advanced cooling techniques, and composable silicon. Learn more at www.opencompute.org.