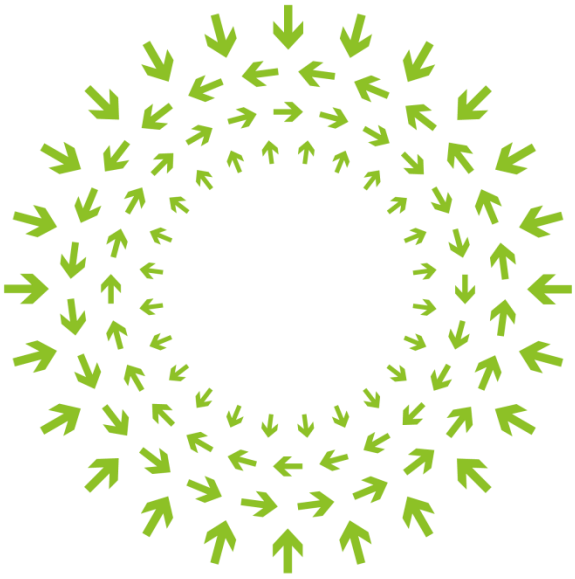

OPEN
Compute Project

DC-SCM 2.1 LVDS Tunneling Protocol & Interface Specification (LTPI)

Revision 1.1, Version 1.0

Revision 1.1 Authors:

Intel: Kasper Wszolek, Yi Zeng, Qian Wang, Katarzyna Krzewska
Dell: Tim Lambert, Shawn Dube, Rich Boyle
Lattice: Munir Ahmad, Loyce Trisha Beloso
Oracle: Joseph Ervin

Revision History

Revision	Version	Date	Notes
1.0		Dec 31st 2020	DC-SCM 2.0 Initial public review using DC-SCM 0.8 as base. First LTPI Definition Proposal.
		July 2021	DC-SCM2.0 updates using DC-SCM1.0 as base
		Sept 2021	Updates from working group decisions
	0.5	Oct 2021	Implement 0.95 pinout descriptions
	0.51	Oct 2021	LTPI Details
	0.52	Nov 9, 2021	General updates and fixes in the document. LTPI Link Initialization, Training and Configuration details added. Incorporated feedback from v0.51 review.
	0.53	Nov 20, 2021	Updates based on feedback.
	0.7	Nov 23, 2021	0.7 released
	0.9	Feb 10, 2022	Separated from DC-SCM 2.0 Base Specification into standalone document. Addressed feedback from v 0.7 review. **Major changes:** - Training Flow improvements and definition of Frame Alignment sub-stages - Operation status and CRC error notification in Data Frames - I2C/SMBus Channel Echo messages to improve reliability in case of Link errors - UART over-sampling details defined - Added new CRC Errors handling chapter - Added new LTPI Latency chapter **Improvements and clarifications:** - General updates and fixed in the documents - Sending frames back-to-back and possible inter-frame gaps considerations - How NL GPIOs are decoded and how number of NL GPIOs is derived from Configuration Frame - Tag values usage and tracking requirements - Added SDR and DDR options in clocking architecture diagram
	0.95	May 6, 2022	- Changed SMBus/I2C channel behavior to send events continuously until new event occurs; added new Data Received Echo Event - Added Capabilities type field to allow easy customization of capabilities - Minor fixes and cleanup
	0.96	June 15, 2022	- Added CRC polynomial initial value definition - Increased HPM Accept Frame Timeout - Added LTPI Link Retraining Request bit in CSR
	1.0	July 27, 2022	1.0 release

1.1	1.0	Oct 13, 2023	- Added a clarification in UART tunneling principle of operation examples (chapter 2.2.1.2) that CRC needs to be checked before signal samples received are regenerated on UART. - Marked Echo Support option as always enabled in the Default LTPI Capabilities (Table 28). Echo Messages cannot be disabled with addition of continuously sent SMBus Events introduced in v0.95 - Fixed incorrect UART Enable bit alignment in the Default LTPI Capabilities (Table 28) - Added missing UART0 Enable in place of Duplicated UART1 fields in the Default LTPI Capabilities (Table 28) - 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

Contributions to this 2.0 Revision of the Specification are made under the terms and conditions set forth in Open Web Foundation Contributor License Agreement ("052021 OWF CLA 1.0") ("Contribution License") by:

Advanced Micro Devices
AMI US Holdings Inc.
Ampere Computing
Arm, Ltd.
ASPEED Technology, Inc.
Dell, Inc.
GOOGLE LLC
Hewlett Packard Enterprise
IBM Corporation
Inspur Electronical Information Industry Co., Ltd.
Intel Corporation
Lattice Semiconductor Corporation
Lenovo, ISG
Microsoft Corporation
MiTAC Computing Technology Corporation
Oracle Americas, Inc.

Usage of this Specification is governed by the terms and conditions set forth in the Open Web Foundation Final Specification Agreement ("OWFa 1.0").

Note: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)

I2C TRADEMARK OF PHILLIPS SEMICONDUCTOR

I3C TRADEMARK OF MIPI ALLIANCE, INC

USB TRADEMARK OF USB IMPLEMENTORS FORUM, INC

PCIE TRADEMARK OF PCI-SIG

ESPI TRADEMARK OF INTEL CORP

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Acknowledgement

With the hope of making this specification useful for the entire OCP community, we acknowledge and appreciate the contributions, review, and feedback from members of DC-SCM 2.0 working group and other companies or individuals who provided feedback during public reviews and monthly meetings.

References

- Open Compute Project. Hardware Management Module Subproject https://www.opencompute.org/wiki/Hardware_Management/Hardware_Management_Module
- SMBus Management Interface Forum. System Management Bus (SMBus) Specification. System Management Interface Forum, Inc, Version 2.0, August 3rd, 2000
- Intel® Avalon® Interface Specifications https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

Trademarks

Names and brands may be claimed as trademarks by their respective companies. I2C® is a trademark of NXP Semiconductor.

Table of Contents

1 Overview 10

1.1 DC-SCM 2.0 LTPI Architecture 10

1.2 DC-SCM 2.0 LTPI Channels 11

1.2.1 Channels mapping to physical interfaces 12

1.2.2 GPIO Channel 14

1.2.3 I2C/SMBus Channel 16

1.2.4 UART Channel 17

1.2.5 OEM Channels 18

1.2.6 Data Channel 19

2 LTPI Architecture 21

2.1 General Working Principle 22

2.2 Interfaces Tunneling Principle 24

2.3 LTPI Clock Topology 38

2.4 CRC Checksum 38

2.5 CRC Errors Handling 39

2.6 8b/10b Encoding 41

2.7 LTPI Latency 41

2.7.1 I2C/SMBus End-to-End Bandwidth 44

2.7.2 UART End-to-End Integrity 45

3 LTPI Frames Definition 48

3.1.1 Link Detect and Link Speed Selection Frames 49

3.1.2 Advertise, Configure and Accept Frames 51

3.1.1 LTPI Operational Frames 55

3.2 LTPI Control and Status Registers (CSR) 57

4 LTPI Link Initialization and Operation 61

4.1.1 Link Training 64

4.1.2 Link Configuration 68

4.1.3 Operational mode 73

4.1.4 Link Training and Configuration Example 75

5 Acronyms 77

List of Figures

Figure 1 LVDS connection to SCM CPLD 10
Figure 2 Direct connection of LVDS interface to the BMC. 11
Figure 3 LTPI Channels encapsulation and serialization using TDM method. 12
Figure 4 Example of GPIO Channel mapping to physical GPIOs 13
Figure 5 Example GPIO Channel to physical GPIOs and SGPIO interface 14
Figure 6 Example of GPIO mapping to Low Latency and Normal Latency GPIOs 15
Figure 7 Example mapping of I2C Channel to physical I2C buses on HPM 17
Figure 8 Example mapping of UART Channel to physical UART interfaces on HPM and SCM 18
Figure 9 Example of OEM Channel usage for tunneling OEM-specific interface 19
Figure 10 Example Data Channel use with internal AVMM interface. 20
Figure 11 Data Channel usage to access ADC registers in HPM FPGA 21
Figure 12 LTPI Architecture 21
Figure 13 LTPI Block Diagram 23
Figure 14 NL GPIOs mapping example with Total Number of NL GPIOs equal to 100 26
Figure 15 LTPI UART Sampling Frequency 26
Figure 16 LTPI UART Sampling distribution across LTPI Frames 28
Figure 17 I2C/SMBus Relay Block Diagram 29
Figure 18 Example of I2C/SMBus Channel Controller-Target connection 30
Figure 19 I2C/SMBus Events and clock stretching for Controller and Target side 30
Figure 20 LTPI Clock Topology 38
Figure 21 LTPI Constant Latency 42
Figure 22 LTPI Random Data Access Latency 42
Figure 23 LTPI Latency impact on I2C/SMBus Events 44
Figure 24 LTPI Latency impact on UART Integrity 46
Figure 25 UART signal Jitter impact RX signal integrity 46
Figure 26 Reduced Jitter impact by limited Data Channel Frames 47
Figure 27 Example of LTPI CSR access from BMC 58
Figure 28 Link Training and Initialization Flow 63
Figure 29 LTPI Link Detect State Frames 65
Figure 30 LTPI Link Speed State Frames 66
Figure 31 LTPI Advertise State Frames 69
Figure 32 LTPI Advertise State Frames 70
Figure 33 LTPI Operational State with I/O Frames Stream 73
Figure 34 Data Frames interleaving I/O Frames Stream 74
Figure 35 Link Training and Configuration Flow 76

List of Tables

Table 1 LTPI Channels summary 11
Table 2 Low Latency and Normal Latency GPIOs 15
Table 3 Example of GPIO Channel Definition for Normal Latency GPIOs from SCM perspective 15
Table 4 UART Signals tunneled through LTPI 17
Table 5 LTPI Channels working characteristics. 24
Table 6 LTPIO GPIO Channel Encoding 25
Table 7 UART Maximum Baud Rate example for Default IO Frame 26
Table 8 UART Channels Encoding 28
Table 9 I2C/SMBus Events flow between Controller and Target LTPI I2C/SMBus Relays 31
Table 10 I2C/SMBus Events Bit Encoding 35
Table 11 I2C/SMBus Bit Fields Encoding in LTPI Frame 36
Table 12 Data Bus Command encoding. 36
Table 13 Memory Read and Write operations Encoding 37
Table 14 CRC Error impact on LTPI channels 39
Table 15 Frame Transfer Latencies Example 43
Table 16 LTPI Latency impact on LTPI channels 43
Table 17 I2C/SMBus Channel End-to-End Bandwidth 45
Table 18 Example of maximum number of consecutive Data Channel Frames 47
Table 19 LTPI Frame Types summary 48
Table 20 Link Detect and Link Speed Frame Type and Subtypes 49
Table 21 Speed Capabilities Encoding 49
Table 22 Link Detect Frame 49
Table 23 Speed Select Encoding 50
Table 24 Link Speed Frame 50
Table 25 Advertise, Configure and Accept Frame Type and Subtypes 51
Table 26 Platform Type Encoding 51
Table 27 UART Channel Max Baud Rate Encoding 52
Table 28 Default LTPI Capabilities Encoding 52
Table 29 LTPI Advertise Frame 53
Table 30 LTPI Configure Frame 53
Table 31 LTPI Accept Frame 54
Table 32 LTPI Operational Frame Type and Subtypes 55
Table 33 LTPI Default I/O Frame 55
Table 34 LTPI Default Data Frame 56
Table 35: General Control and Status Register Map 58
Table 36: LTPI Control and Status Registers 58
Table 37 Link Detect State Overview 65
Table 38 Link Speed State Overview 66
Table 39 Speed Capabilities sent by HPM 67

Table 40 Speed Capabilities sent by SCM 67
Table 41 Speed Select sent by SCM and HPM 67
Table 42 Advertise State Overview 69
Table 43 Configure and Accept States Overview 70
Table 44 SCM Capabilities presented in Advertise Frame 71
Table 45 HPM Capabilities presented in Advertise Frame 72
Table 46 Configuration requested in Configure Frame 72
Table 47 Operational States Overview 74

1 Overview

This specification provides architectural details of the LTPI (LVDS Tunneling Protocol & Interface) which is introduced in the DC-SCM 2.0 Specification. LTPI is a protocol and interface designed for tunneling various low-speed signals between the HPM and SCM. The LTPI protocol goes over the LVDS (Low Voltage Differential Signals) electrical interfaces supported by majority of the CPLDs and FPGAs. This is the next generation protocol for DC-SCM 2.0 as the replacement to two Serial GPIO (SGPIO) interfaces. The LVDS interface provides higher bandwidth and better scalability than the SGPIO interface. It allows for tunneling of not only GPIOs but also low speed serial interfaces such as SMBus, I2C, UART. It is also extensible with additional proprietary OEM interfaces and provides support for raw Data tunneling between HPM CPLD and SCM CPLD.

1.1 DC-SCM 2.0 LTPI Architecture

The LTPI interface can be implemented with two FPGA/CPLD devices for configurations with typical BMC devices (Figure 1):

1. HPM FPGA – providing a bridging of local HPM interfaces to LTPI
2. SCM CPLD – providing an LTPI bridging to local SCM interfaces

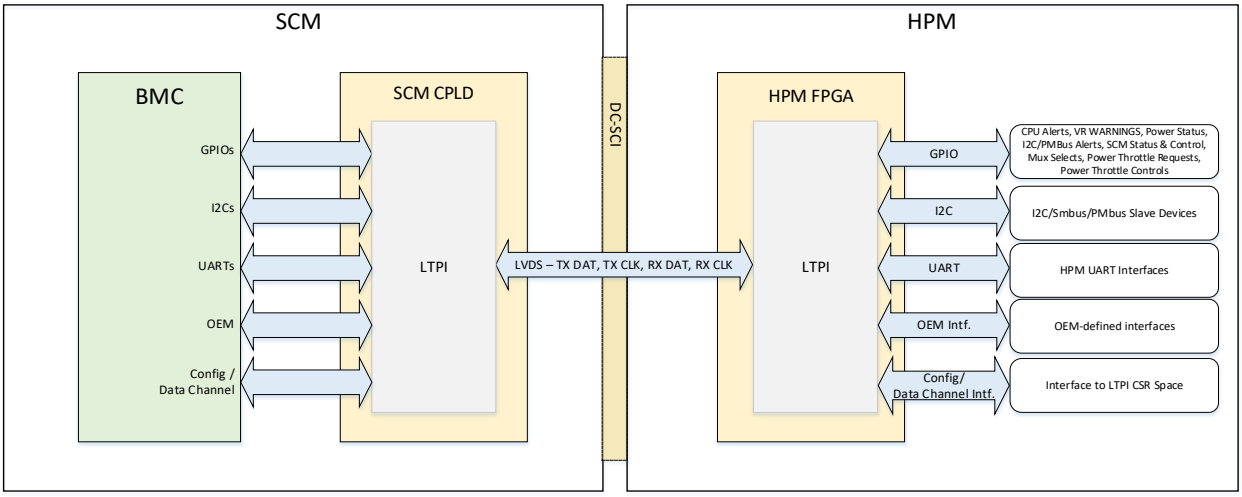

Figure 1 LVDS connection to SCM CPLD.

Future BMC SoC devices might integrate LVDS interface and LTPI protocol in the BMC ASIC or integrated FPGA if BMC SoC integrates FPGA fabric. In such solution LTPI would be directly handled by BMC HW as shown in Figure 2.

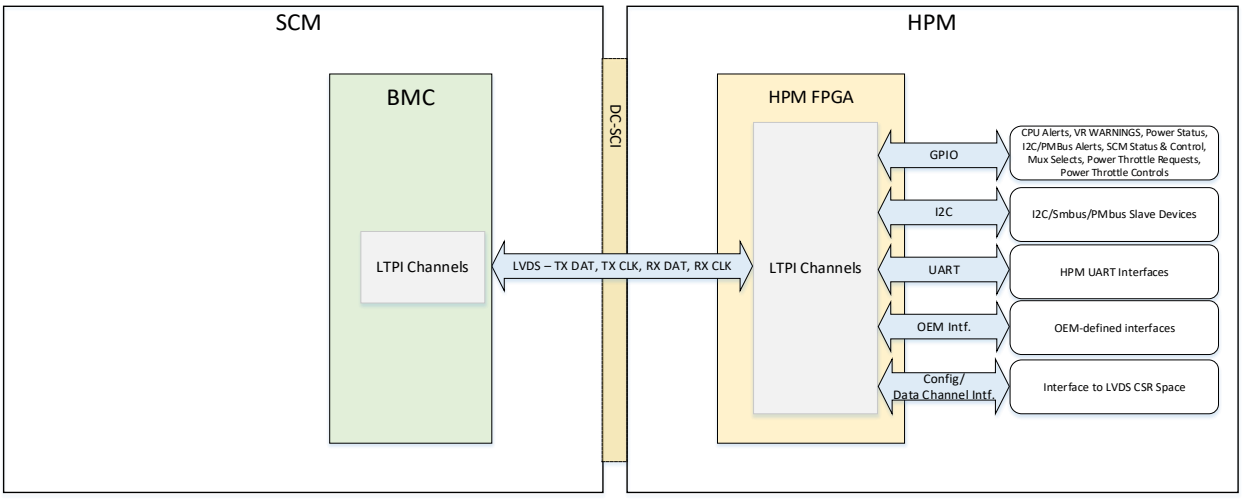

Figure 2 Direct connection of LVDS interface to the BMC.

1.2 DC-SCM 2.0 LTPI Channels

The DC-SCM LTPI interface implements a concept of channels where each channel is mapped to a specific type of physical interface on the DC-SCM and HPM. The following channels are being defined for LTPI Interface:

- GPIO Channel
- I2C/SMBus Channel
- UART Channel
- OEM Channel
- Data Channel

The summary of the LTPI Channels is listed in the **Error! Reference source not found.** below.

Table 1 LTPI Channels summary

LTPI Channel	Required	Description
GPIO	Yes	Tunnels GPIO signals from HPM to SCM and from SCM to HPM. The GPIO Channel allows for differentiation between Low Latency and Normal Latency GPIOs to allocate more bandwidth for time critical GPIO Signals and allow for scalability in extending number of tunneled GPIOs.
I2C/SMBus	No	Tunnels I2C/SMBus Links from SCM to HPM and HPM to SCM. The definition of DC-SCM LTPI I2C/SMBus channel is limited to a use case with single I2C/SMBus Controller which is located either on SCM or HPM for a given LTPI Link.
UART	No	Tunnels Full-Duplex UART interfaces with flow control support between SCM and HPM.
OEM	No	This channel is a placeholder for any proprietary OEM interfaces that can be tunneled with LTPI. This is providing a placeholder for LTPI Bandwidth allocation to be used for non-standard interfaces.

Data	No	The Data Channel allows for tunneling random access to addressable memory-mapped addressable spaces. The intention is to allow remote access to standardize memory mapped buses used to interconnect CPLD/FPGA IPs e.g., Avalon-MM or Wishbone bus. This way data can be easily tunneled between SCM CPLD and HPM CPLD to allow for flexibility in use cases of LTPI interface.

The LTPI uses Time Division Multiplexing (TDM) of high speed LVDS full-duplex link to transmit and receive LTPI channels between SCM and HPM. As shown in the Figure 3 in every equal time slot T_N on LVDS link there is one LTPI Frame being transmitted. Each LVDS channel is being provided with a portion of the LVDS Frame transported in each direction. The number of bits assigned to a specific channel in each frame sent over LTPI interface is directly proportional to the LTPI bandwidth dedicated for each channel. To minimize the latency impact on LTPI channels the LTPI frames are assumed to be transmitted back-to-back without any inter-frame gaps even if channel states are not chaining between frames. In implementations that cannot avoid inter-frame gaps the time of inter-frame gap is recommended to be constant to minimize latency variations and simplify implementation.

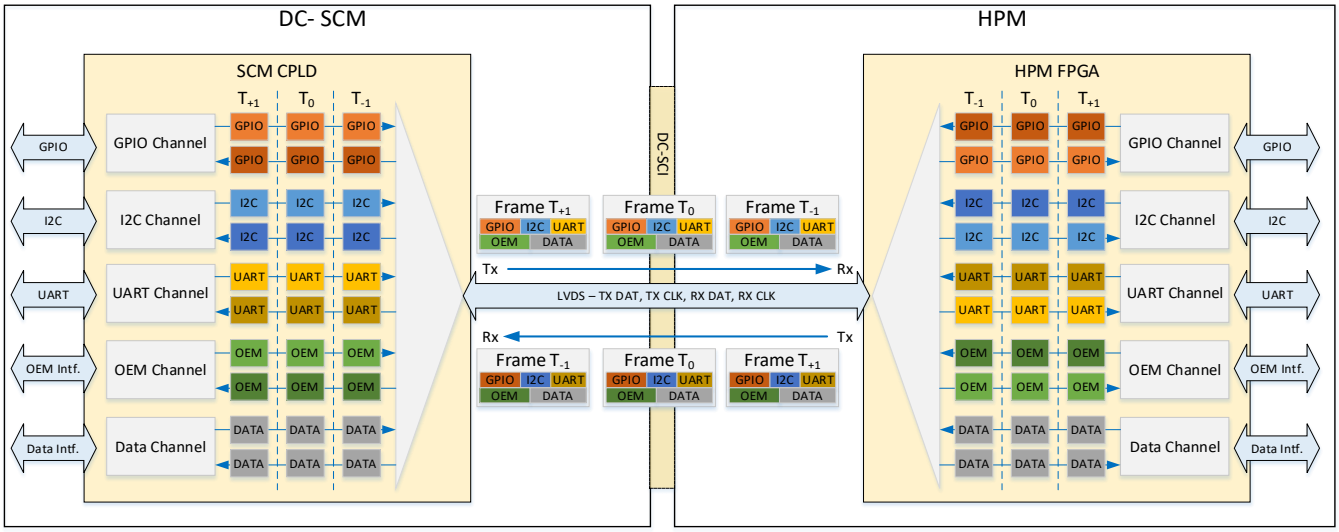

Figure 3 LTPI Channels encapsulation and serialization using TDM method.

1.2.1 Channels mapping to physical interfaces

The LTPI Channels provide a functional classification of specific types of interfaces used in communication between SCM and HPM as listed in the **Error! Reference source not found.**. The physical signals that are connected to the CPLD/FPGA on the HPM and SCM by default are matching the same generic type as in the GPIO channel example implementation showed in the Figure 4 below where physical GPIO signals are being connected to LTPI on SCM and HPM. LTPI allows for more flexibility in mapping of the interfaces. An example of design flexibility with LTPI is presented in Figure 5 below. In this example the GPIO channel is converted into SGPIO interface with additional logic within SCM CPLD.

While similar flexibility applies to all LTPI channels the DC-SCM LTPI specification examples are limited only to direct matching of the channels to physical interfaces as in example in Figure 4. It is a design choice how to eventually map the LTPI channels to physical interfaces and use the LTPI flexibility to meet the design goals.

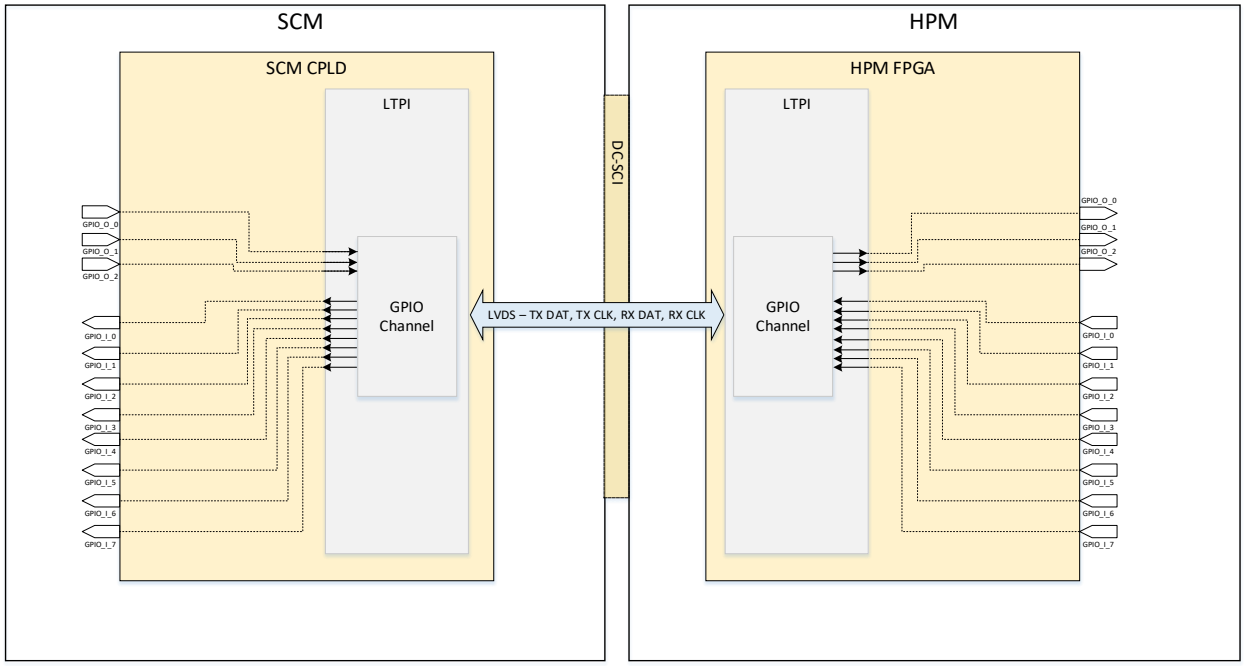

Figure 4 Example of GPIO Channel mapping to physical GPIOs

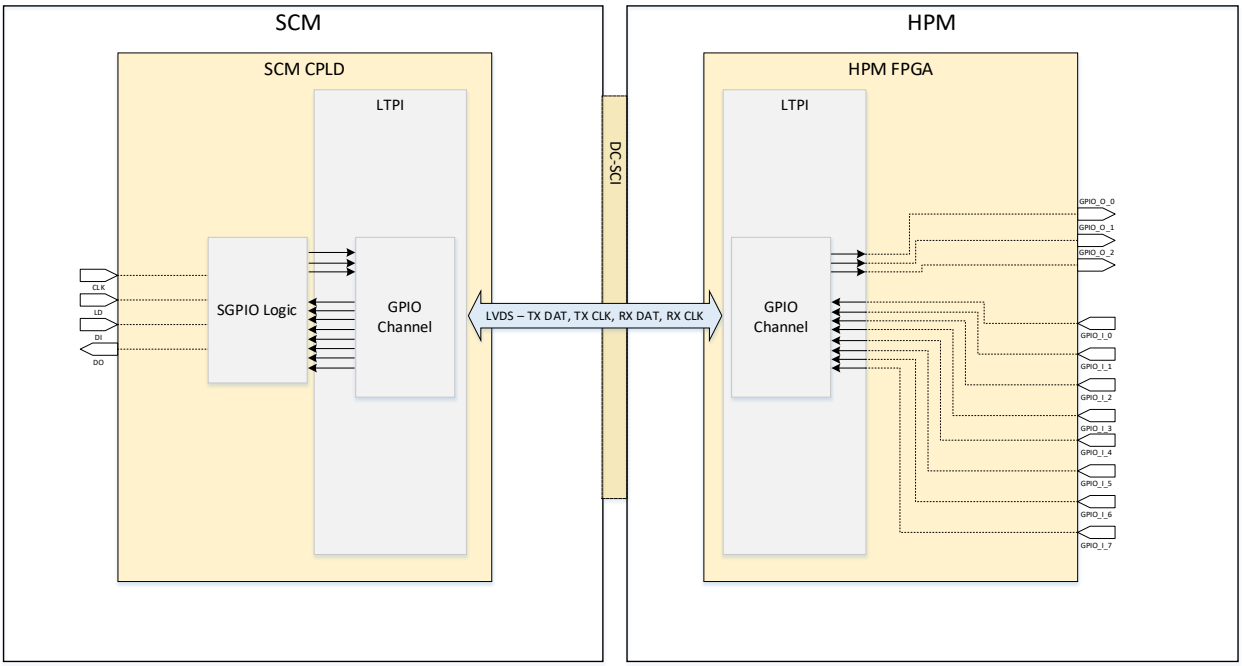

Figure 5 Example GPIO Channel to physical GPIOs and SGPIO interface

1.2.2 GPIO Channel

The GPIO Channel is used to tunnel low speed HPM and SCM GPIOs through LTPI interface. The GPIO channel defines Low Latency and Normal Latency GPIOs. It is a particular platform design choice which GPIO is mapped to Low Latency pool and which to Normal Latency pool. This is a part of CPLD/FPGA IP configuration and reprogramming and can be modified through CPLD/FPGA image update following DC-SCM Plug & Recode model. An example of the mapping of physical GPIOs to Low Latency and Normal Latency GPIOs in LTPI GPIO Channel is presented on Figure 6 below.

Note: The GPIO is a generic terminology where it can be a Software (either CPU on HPM or BMC on SCM) controlled Input/Output, or a pure hardware digital signal that behave similarly as a physical direct connection.

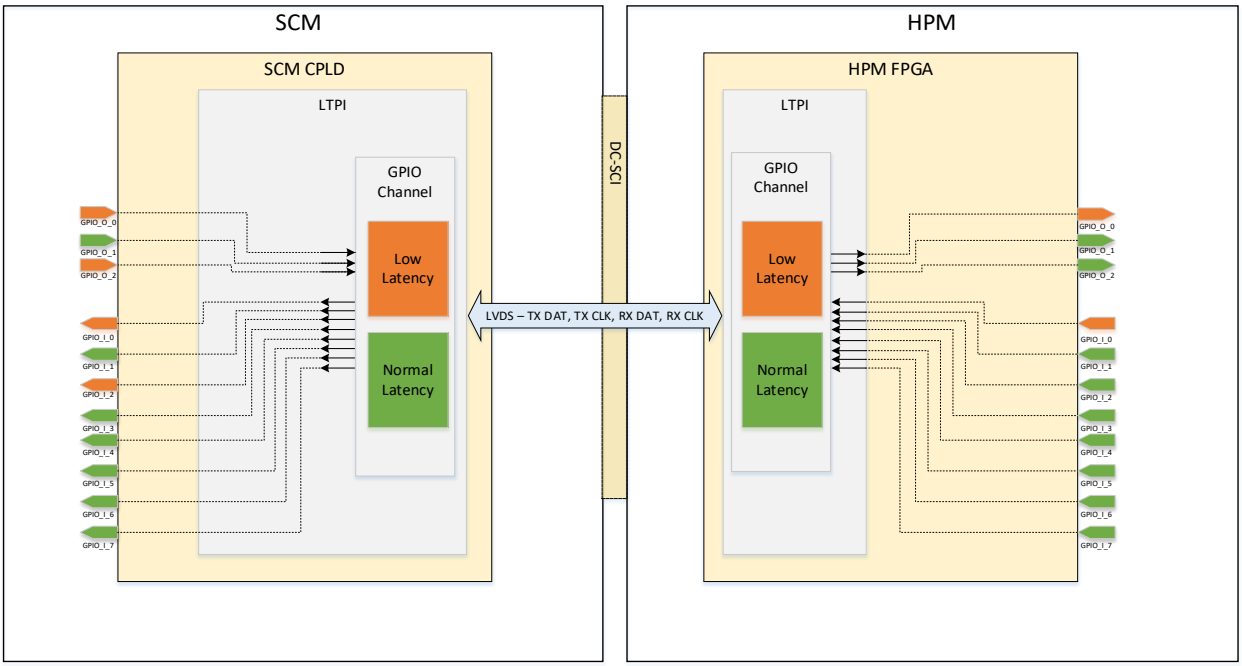

Figure 6 Example of GPIO mapping to Low Latency and Normal Latency GPIOs

The overview of Low Latency and Normal Latency GPIOs characteristics is defined in the Table 2 below

Table 2 Low Latency and Normal Latency GPIOs

Parameter	Low Latency (LL) GPIOs	Normal Latency (NL) GPIOs
Bandwidth	Maximum LTPI Bandwidth allocated for each Low Latency GPIO. Each GPIO is mapped to specific bit in the LTPI Frame and updated with every LTPI Frame.	Maximum LTPI Bandwidth is allocated on predefined number of bits in the LTPI Frame. All bits are shared between Normal Latency GPIOs. It takes multiple LTPI frames to update all NL GPIOs depending on the number of NL GPIOs and number of NL GPIO bits allocated in LTPI Frame.
Number of GPIOs	Limited by the number of LTPI Frame bits allocated for LL GPIOs.	Limited by the acceptable latency for NL GPIOs. The more NL GPIOs defined the higher latency.

An example of Normal Latency GPIOs used on x86 server platform is shown in the Table 3. It includes only CPU0 Socket signals and BMC signals. Various platform related signals like Mux Control, Debug, etc. are not listed. Depending on the HPM design a separate set of CPU-related signals might be required for every CPU Socket on the HPM.

Table 3 Example of GPIO Channel Definition for Normal Latency GPIOs from SCM perspective

GPI#	GPI Signal	Description	GPO#	GPO Signal	Description

0	FP_PWR_BTN_I_N	Power Button Input	0	FP_PWR_BTN_O_N	Power Button Output
1	FP_RST_BTN_I_N	Reset Button Input	1	FP_RST_BTN_O_N	Reset Button Output
2	FP_ID_BTN_N	ID Button Input	2	FP_ID_LED_N	ID LED Input
3	FP_NMI_BTN_N	NMI Button Input	3	FP_LED_STS_AMBER_N	Status LED Amber Output
4	PSU_PWR_OK_N	PSU Power OK	4	FP_LED_STS_GREEN_N	Status Green Output
5	CPU0_ERR_0_N	CPU Error 0	5	BMC_ONCTL_N	BMC On Control Output
6	CPU0_ERR_1_N	CPU Error 1	6	BMC_RST_IND_LED	BMC CPU in Reset LED
7	CPU0_ERR_2_N	CPU Error 2	7	BMC_SPEAKER	BMC Speaker
8	CPU0_MEMHOT_N	Memory Hot	8	CPU0_AUX_PWRGD_N	CPU AUX Power Control
9	CPU0_MEMTRIP_N	Memory Thermal Trip	9	CPU0_PLTRST_N	Platform Reset
10	CPU0_THERMTRIP_N	CPU Thermal Trip	10	CPU0_RTC_RST_N	RTC Reset
11	CPU0_MEM_VRHOT_N	CPU Memory VR Hot	11	BIOS_POST_CODE_LED0	BIOS Post Code LED 0
12	CPU0_VR_HOT_N	CPU VR Hot	12	BIOS_POST_CODE_LED1	BIOS Post Code LED 1
13	CPU0_CATERR_N	CPU Catastrophic Error	13	BIOS_POST_CODE_LED2	BIOS Post Code LED 2
14	CPU0_PKGID0	CPU Package ID 0	14	BIOS_POST_CODE_LED3	BIOS Post Code LED 3
15	CPU0_PKGID1	CPU Package ID 1	15	BIOS_POST_CODE_LED4	BIOS Post Code LED 4
16	CPU0_PKGID2	CPU Package ID 2	16	BIOS_POST_CODE_LED5	BIOS Post Code LED 5
17	CPU0_SLPS3_N	ACPI Sleep S3	17	BIOS_POST_CODE_LED6	BIOS Post Code LED 6
18	CPU0_SLPS4_N	ACPI Sleep S4	18	BIOS_POST_CODE_LED7	BIOS Post Code LED 7

The Low Latency GPIOs can be defined in a similar manner. More detailed examples of NL and LL GPIO mapping for typical HPM platforms will be defined in the DC-SCM

1.2.3 I2C/SMBus Channel

The I2C/SMBus Channel is used to tunnel I2C/SMBus buses through LTPI interface for the links where there is only one Controller either on SCM or on the HPM. The primary use case addressed by the DC-SCM LTPI I2C/SMBus tunneling is as presented in the Figure 7 where BMC on SCM acts as the controller of the I2C/SMBus links with Target devices located on the HPM. From BMC perspective the I2C/SMBus buses are accessible the same as if they were physically routed through SCI interface and connected to the BMC. Tunneling of the I2C/SMBus use the Clock Stretching method defined in the I2C/SMBus specification to compensate for the LTPI latency and turnaround time. The DC-SCM LTPI does not limit the Controller location to be only on the SCM.

Note: I2C/SMBus channel in LTPI is introduced as optional for DC-SCM 2.0 as physical I2C buses are still defined in DC-SCI. The LTPI allows to move I2C physical pins to I2C/SMBus over LTPI in the future, so that physical pins can be utilized more efficiently. The topology presented in the Figure 7 below is intended to show potential use of LTPI I2C/SMBus channels and it is presenting more channels than the Default

I/O Frame defined in this specification which would have to be added either through OEM fields in Default I/O Frame or by defining OEM I/O Frame Sub-Type.

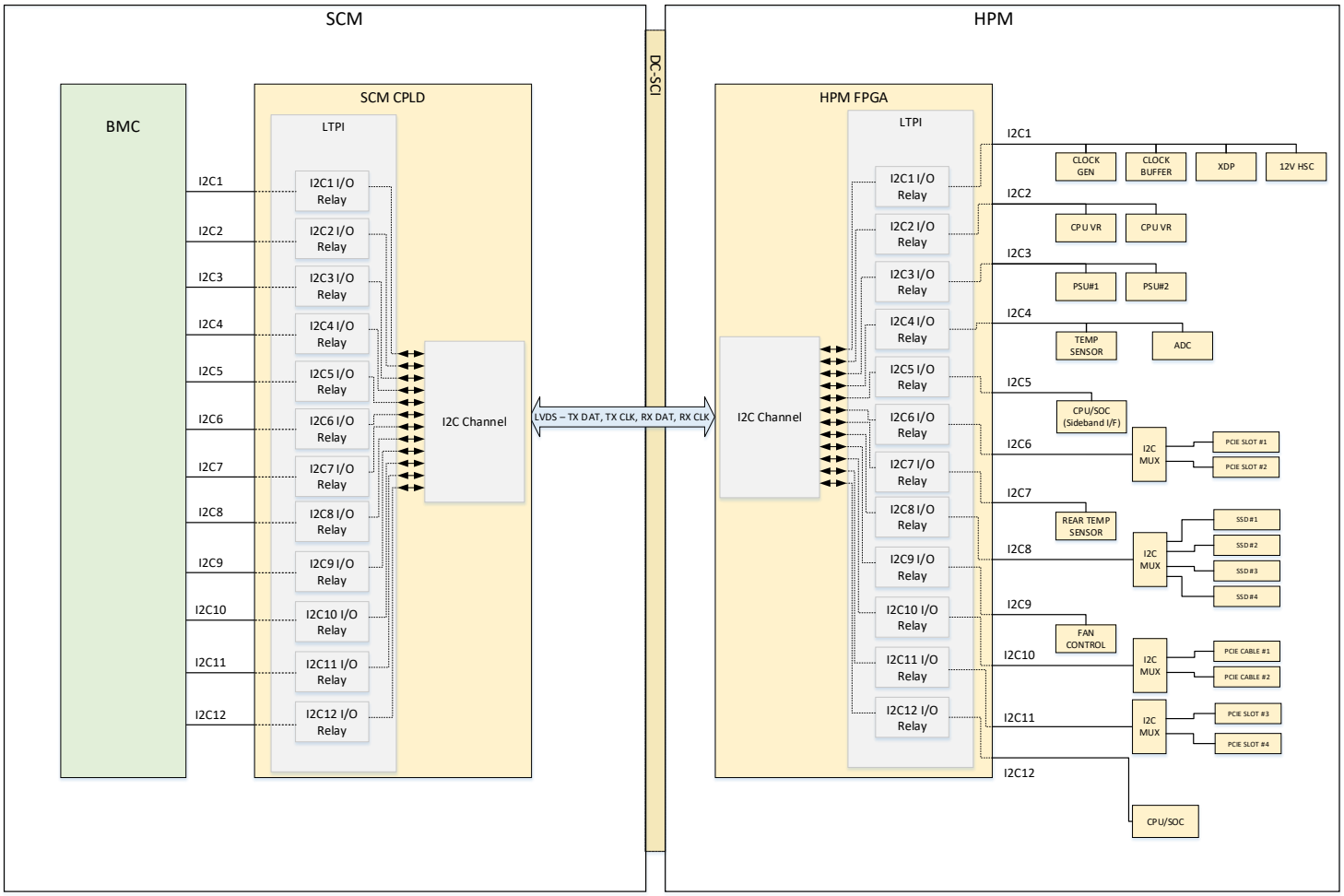

Figure 7 Example mapping of I2C Channel to physical I2C buses on HPM

1.2.4 UART Channel

The UART Channel is used to tunnel physical UART interfaces between SCM and HPM through LTPI. The UART Channel supports tunneling of multiple Full Duplex UART interfaces with Flow Control Signals. The UART signals tunneled through LTPI are listed int the Table 4 below from SCM Perspective.

Table 4 UART Signals tunneled through LTPI

UART Signal Name	Description
TXD	Transmit Data UART Output Signal from the UART Controller on SCM and HPM
RXD	Receive Data UART Input Signal from the UART Controller on SCM and HPM
CTS	Clear To Send UART Input Signal from the UART Controller on SCM and HPM

RTS	Ready To Send UART Output Signal from the UART Controller on SCM and HPM

The UART interface is tunneled through LTPI and reconstructed into physical UART interface similar to GPIO and I2C Channels. As shown in the Example usage of UART on Figure 8 below, the tunneled HPM UART interfaces can be connected directly to BMC or to other device like USB-UART Bridge. It is a design choice how the UART Channel is utilized in a specific platform.

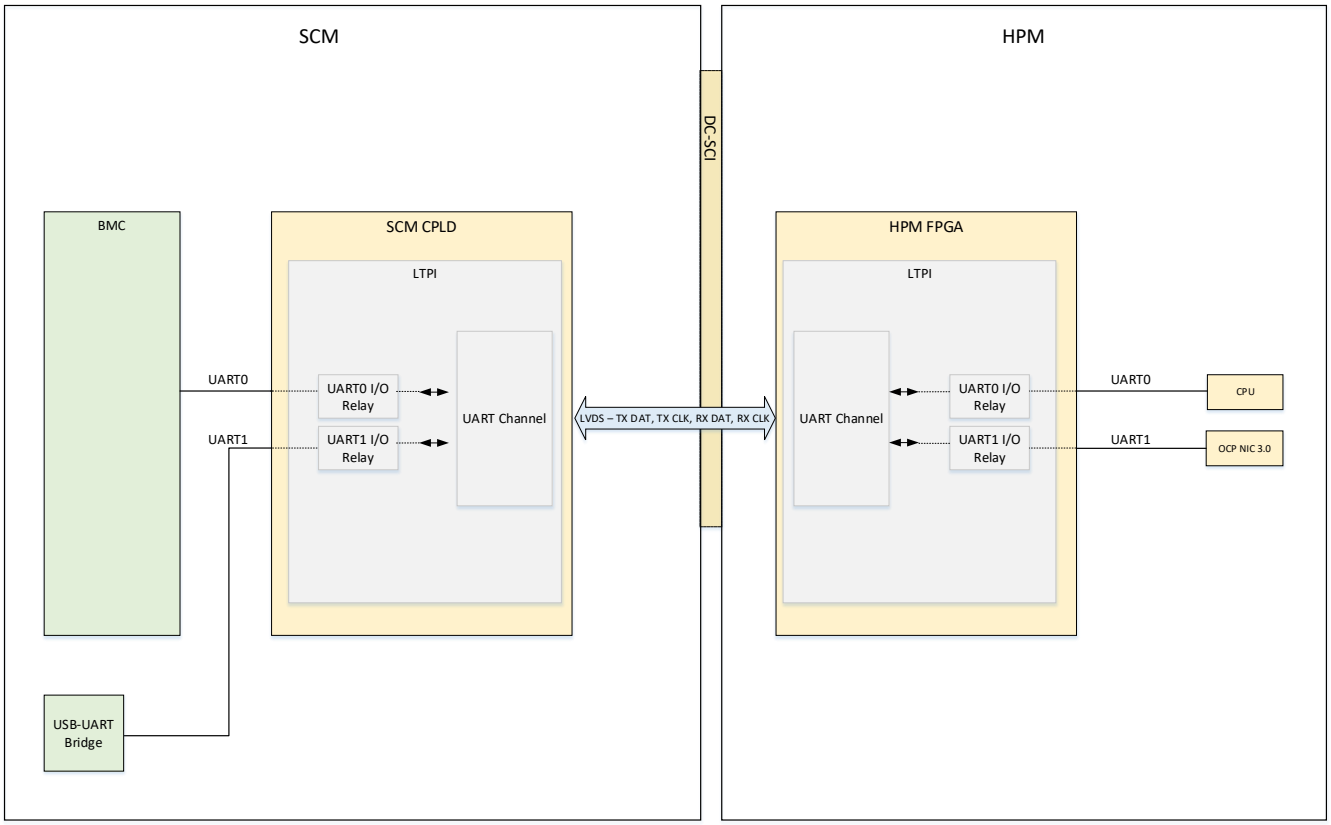

Figure 8 Example mapping of UART Channel to physical UART interfaces on HPM and SCM

1.2.5 OEM Channels

The OEM Channels allow for OEM Specific interface tunneling through LTPI. The OEM can implement proprietary interface tunneling by following the same approach as used in other Channels. It is out of scope of this specification to define the OEM-specific interface details and Channel design. OEM Channels is only defined as placeholder for OEM extension and proprietary interfaces implementation using DC-SCM LTPI.

Figure 9 Example of OEM Channel usage for tunneling OEM-specific interface

1.2.6 Data Channel

The Data Channel allows for tunneling of Data between SCM CPLD and HPM FPGA via LTPI. This adds a flexibility and possibility to implement LTPI extensions to match specific design needs. The Data Channel follows a concept of a Memory Mapped Interfaces such as Avalon Memory Mapped (AVMM) or Wishbone Bus. As shown in the example in the Figure 10 below the Data Channel implements an internal AVMM bus that allows for mapping Memory Spaces between SCM CPLD and HPM FPGA and use of standardized IP blocks that follows AVMM bus support. Data channel allows a Random-Access Data Read and Write from IP located behind the LTPI Interface. Data Channel accesses are done on demand only when there's a request to write or read the data.

Note: In current definition, the SCM is always the Memory Transaction initiator to provide a way for the BMC software on the SCM to access memory mapped register spaces defined in the HPM.

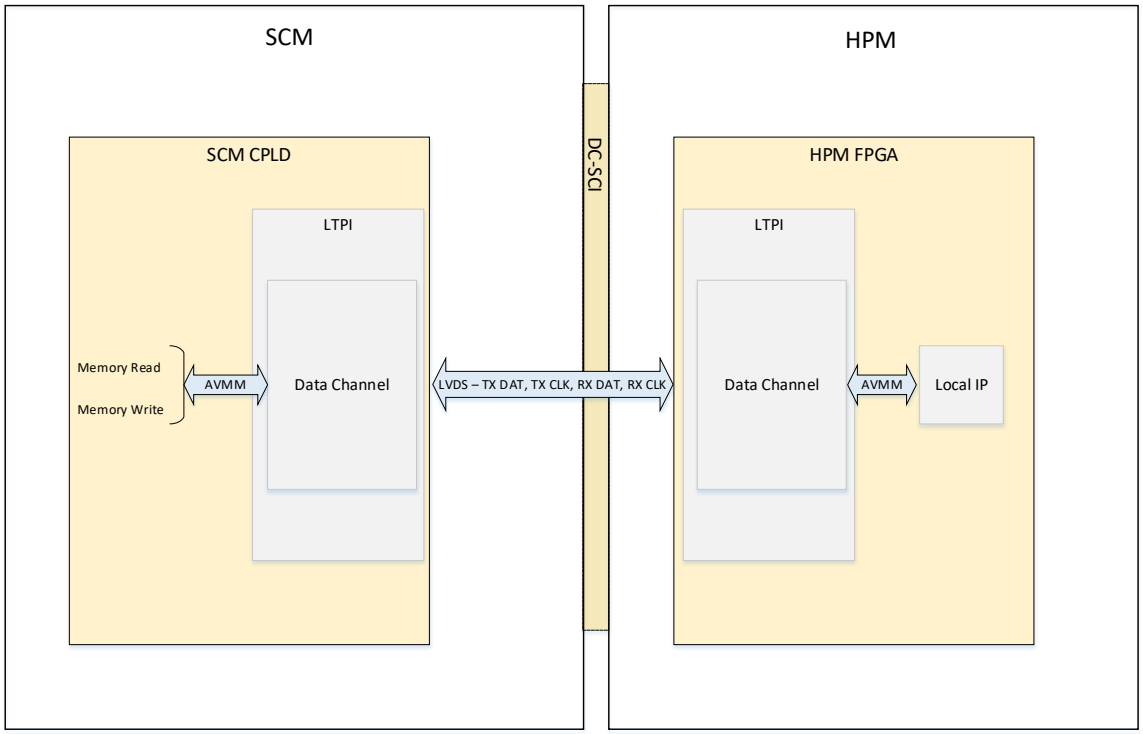

Figure 10 Example Data Channel use with internal AVMM interface.

One of the examples of Data Channel usage is shown in the Figure 11 below where Data Channel is used to access the Analog Digital Converter readings from ADC which is built into the HPM FPGA. The BMC access to Data Channel can be provided by any standard interface that BMC SOC supports and is connected to the SCM CPLD e.g., a SPI, I2C or similar interface. In the Example below the BMC is using the I2C interface to trigger reads from the ADC. The I2C AVMM bridge interprets BMC reads and converts them to AVMM Data Reads that are passed through LTPI interface to the HPM FPGA. The definition of Data Channel is generic and it is not tied to any specific Data Bus interface.

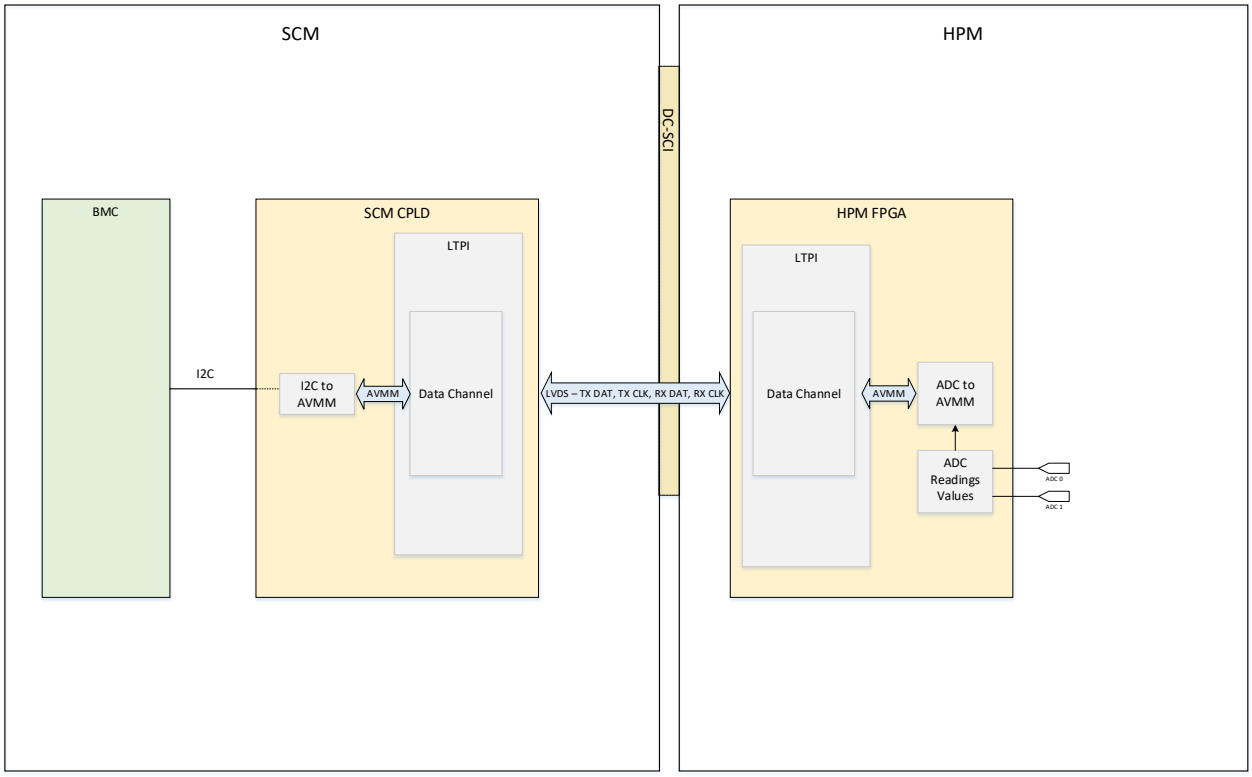

Figure 11 Data Channel usage to access ADC registers in HPM FPGA

2 LTPI Architecture

The LTPI Architecture consist major building blocks as presented in Figure 12 below.

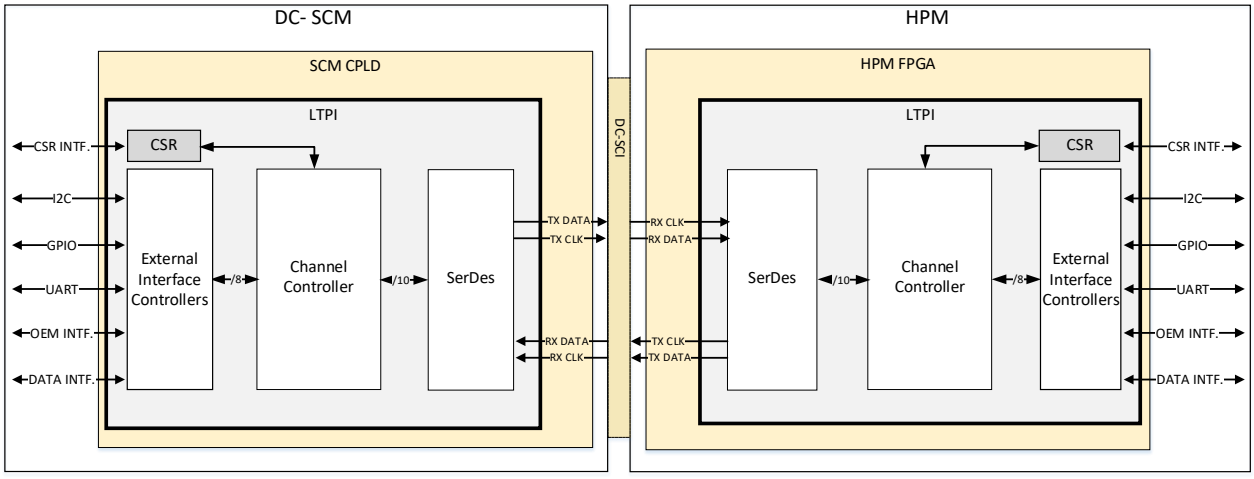

Figure 12 LTPI Architecture

- **External Interface Controllers** – also called I/O Relays are responsible for capturing and reconstructing of physical interfaces on LTPI channels:
    - GPIO
    - I2C/SMBus
    - UART
    - OEM
    - Data
- **CSR** – Configuration & Status Registers provides a way for BMC or other device on the SCM or HPM to control the operation of the LTPI Logic
- **Channel Controller** – this block is responsible for:
    - Link State Machine Control (Link Training and Interface Configuration)
    - Generation of outgoing LTPI Frames
    - Parsing of incoming LTPI Frames
    - CRC Checksum Generation and Verification
    - Comma Symbol Chasing and Locking
    - 8b/10b Encoding and Decoding
- **SerDes** – Serialization and Deserialization of the data on the LVDS Link

The SCM CPLD and HPM CPLD LTPI architectural building blocks are considered symmetric between SCM and HPM from high-level architectural perspective i.e., TX paths and RX paths on both CPLDs are following the same theory of operation in general. There are specific differences between SCM and HPM though e.g., differences in HPM and SCM training requirements or different configurations of specific channels like SMBus where one CPLD will be connected to SMBus Controller while the other to SMBus target. It is assumed that the architectural symmetry of LTPI between HPM and SCM should allow for majority of CPLD source code to be common and shared between HPM and SCM instances of the LTPI in a given platform design.

2.1 General Working Principle

The LTPI Block Diagram is presented on Figure 13 below. LTPI blocks for TX Path and RX paths are the same on both SCM and HPM CPLD from high-level architecture perspective presented on this block diagram.

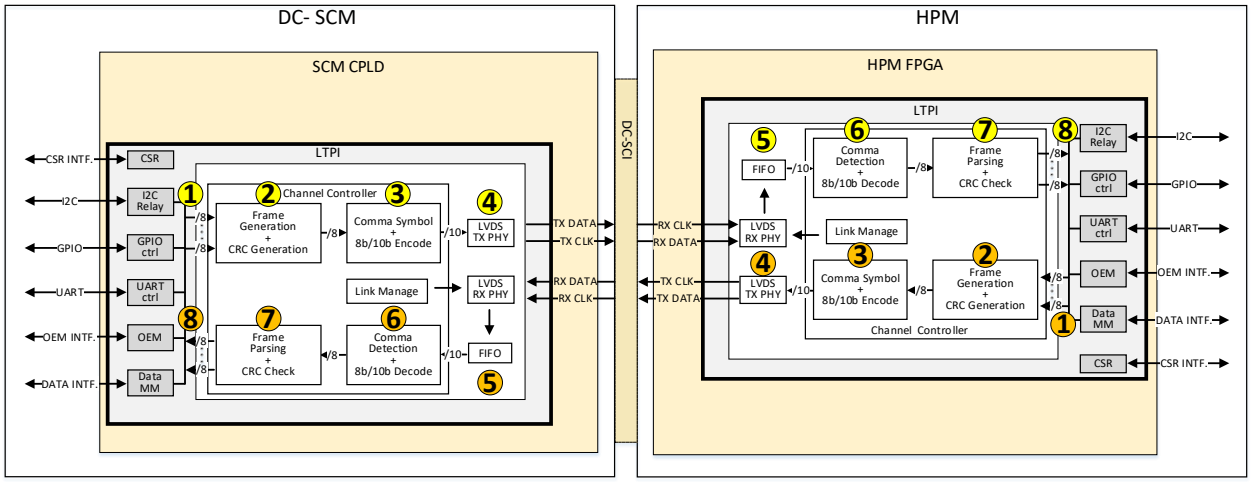

Figure 13 LTPI Block Diagram

The TX Path in both SCM and HPM works as follows:

1. Traffic on physical interfaces is captured by respective Controllers/Relays:
    a. I2C Relays – I2C/SMBus interfaces
    b. GPIO Controllers – GPIO interfaces
    c. UART Controller – UART Interfaces
    d. OEM Controller – OEM Interfaces
    e. Data MM Controller – Data Channel Bridge to external Interface
2. LTPI Channel Controller generates the LTPI Frame and CRC Checksum
    a. Each LTPI Channels is encoded on dedicated bitfields inside the LTPI Frame
3. The Frame is then encoded using 8b/10b encoding
    a. The frame is extended with Frame Comma Symbol
4. The 10b-encoded frame is serialized on the LVDS TX Link

The RX Path in both SCM and HPM works as follows:

5. The LVDS traffic is captured and Deserialized into 10b-encoded FIFO buffer
6. In the 10b data stream the Frame Comma symbol is captured as the beginning of the LTPI frame
    a. The frame is decoded to 8b encoding
7. The frame is checked again CRC checksum and decomposed into LTPI Channels
8. Channel states are being reconstructed on respective channels with channel Controllers/Relays:
    b. I2C Relays – I2C/SMBus interfaces
    c. GPIO Controllers – GPIO interfaces
    d. UART Controller – UART Interfaces
    e. OEM Controller – OEM Interfaces
    f. Data MM Controller – Data Channel Bridge to external Interface

2.2 Interfaces Tunneling Principle

The LTPI uses the following generic methods to capture and tunnel various LTPI channels:

- **Sampling** – the I/O states are sampled by LTPI and the samples are tunneled in the LTPI Frames
- **Event/State Detection** – the set of events/states are defined for channel and tunneled and tunneled in the LTPI Frames according to the interface state
- **Random Access Read/Writes Requests** – Random access memory mapped channel is triggered by external interface

The interfaces supported by the LTPI can also be classified according to the synchronization of RX and TX directions for given interface i.e., whether tunneling of the interface requires both TX and RX directions to be synchronized between each other:

- **Asynchronous** – for a given channel and interface/link LTPI TX and RX paths are independent
- **Synchronous** – for a given channel and interface/link LTPI TX and RX paths need to be synchronized to allow interface to work correctly after tunneling

Following those definitions, the LTPI Channels can be characterized as in the Table 5 below.

Table 5 LTPI Channels working characteristics.

Channel	Capture Method	TX and RX Synchronization	Channel characteristics
GPIO	Sampling	Asynchronous	- Captured signal levels are transmitted directly through LTPI - Low Latency GPIOs are updated in every LTPI frame - Normal Latency GPIOs are split across multiple frames with Frame ID/Frame - Counter used to identify the NL GPIO subset
UART			- Captured signal levels are transmitted directly through LTPI - UART signals are oversampled - Multiple samples are tunneled in every LTPI Frame
I2C/SMBus	Event/State Detection	Synchronous	- I2C/SMBus states are encoded into LTPI events and Events are tunneled through LTPI - I2C Clock stretching is used while waiting for synchronization to be completed after event is transmitted and waiting for I2C/SMBus state/event from the other side of LTPI interface
Data bus	Random Access		- Data bus transaction such as Data Read and Write are encoded into LTPI Events - Completion indication of Bus Operation is used to synchronize access to the Data Bus

2.2.1.1 GPIO Interfaces

For GPIO Channel the GPIO logical states (High and Low) are directly encoded on the LTPI Frame bits. The Low Latency (LL) GPIOs are refreshed every LTPI Frame and have dedicated bit for each GPIO in every Frame sent. The Normal Latency (NL) GPIOs are refreshed every N number of frames where N is defined as:

$$N = \left(\frac{\text{Total Number of NL GPIOs tunneled over LTPI}}{\text{Number of NL GPIO bits per LTPI IO Frame}}\right)$$

In order to decode which NL GPIOs are tunneled in given frame a Frame Counter which is incremented in every Frame is used to decode the NL GPIOs indexes from the LTPI Frame. An encoding of LTPI GPIO Channel in LTPI Frame is presented in the Table 6 below where LL and NL GPIOs are defined on 1 byte each. For LL GPIOs each bit in a byte is defined as GPIO with number from 0 to 7. For Normal Latency GPIOs in order to decode the GPIO Number for each bit position 0 to 7 an offset X needs to be added where X is defined as:

$$X = (\text{Frame Counter } \%\ N) \text{ x } (\text{Number of NL GPIO bits per LTPI Frame})$$

Table 6 LTPIO GPIO Channel Encoding

Name	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
Low Latency (LL) GPIO	LL GPIO7 0: Low 1: High	LL GPIO6 0: Low 1: High	LL GPIO5 0: Low 1: High	LL GPIO4 0: Low 1: High	LL GPIO3 0: Low 1: High	LL GPIO2 0: Low 1: High	LL GPIO1 0: Low 1: High	LL GPIO0 0: Low 1: High
Normal Latency (NL) GPIO	NL GPIO [X+7] 0: Low 1: High	NL GPIO [X+6] 0: Low 1: High	NL GPIO [X+5] 0: Low 1: High	NL GPIO [X+4] 0: Low 1: High	NL GPIO [X+3] 0: Low 1: High	NL GPIO [X+2] 0: Low 1: High	NL GPIO [X+1] 0: Low 1: High	NL GPIO [X] 0: Low 1: High

The Default I/O Frame allocates 2 bytes for the Low Latency GPIOs and 2 bytes for Normal Latency GPIOs. This means 16 NL GPIOs can be sent in each IO Frame and it will take N number of frames to transfer all NL GPIOs. If a certain implementation requires more LL or NL GPIOs defined in a single IO Frame either OEM fields can be used to extend Default I/O Frame or customized I/O Frame can be defined for those implementations. The Total number of Normal Latency GPIOs is defined in Capabilities Frames in 'Number of NL GPIOs' field. The Capabilities are exchanged during the LTPI Training phase. This way LTPI in SCM and LTPI in HPM is able to correctly decode the index of the NL GPIOs in given frame based on the Frame Counter and N parameter defined above. The NL GPIOs shall be considered as Virtual GPIOs, not physical GPIOs. Mapping of NL GPIOs to physical GPIOs is done outside of the scope of LTPI IP. An example of such mapping is presented in Figure 14 the below. The Total Number of NL GPIOs sent in both directions shall be equal between SCM and HPM and can either be fixed in specific designs or can be resolved during Link Training phase. The training phase is intended to align the Total Number of NL GPIOs between the SCM and HPM. Refer to the 4.1.2 Link Configuration for details.

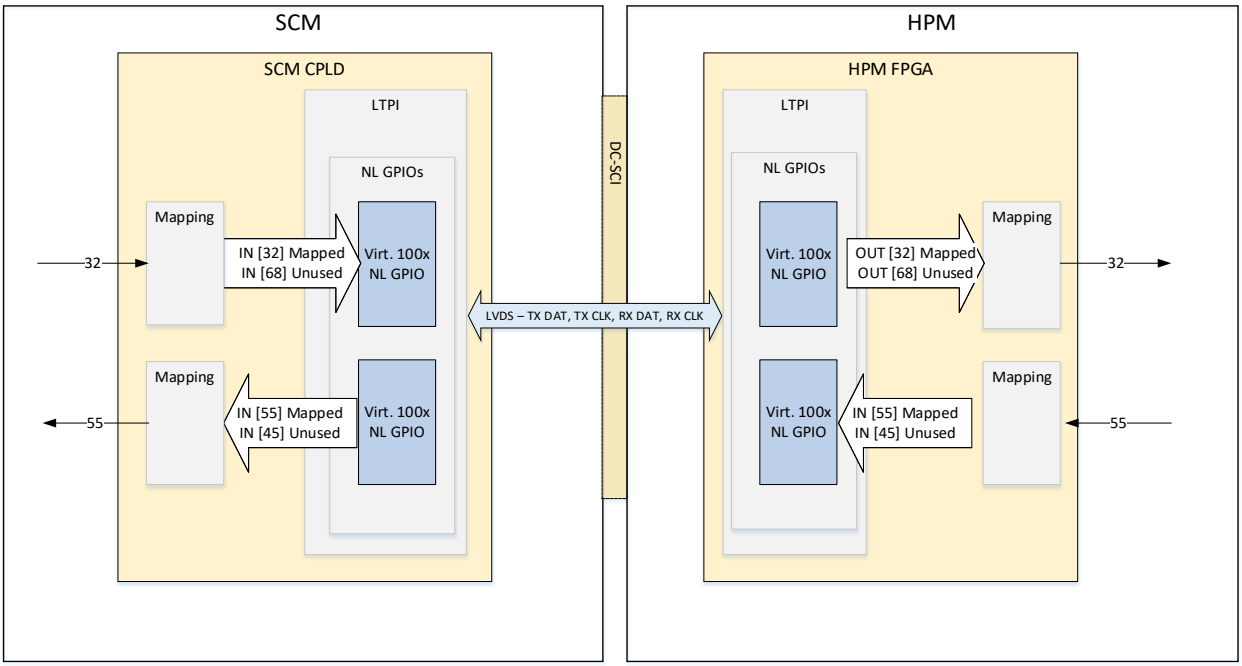

Figure 14 NL GPIOs mapping example with Total Number of NL GPIOs equal to 100

2.2.1.2 UART Interfaces

The working principle of UART channel is similar to LL GPIO channel. The difference is that UART Transmit Data (TXD) and Receive Data (RXD) lines are oversampled by a factor of 3 with regard to the LL GPIO sampling frequency that is determined by the frequency of LTPI Frames generation. The sampling frequency difference between LL GPIO and UART oversampling is outlined in the Figure 15 below.

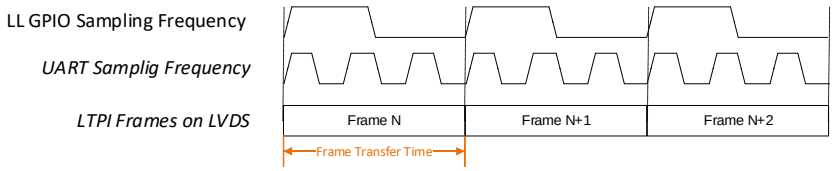

Figure 15 LTPI UART Sampling Frequency

As a result, the maximum UART baud rate supported by LTPI can be determined based on the effective UART sampling frequency.

Table 7 UART Maximum Baud Rate example for Default IO Frame

LTPI Clock Speed [MHz]	Frame Transfer Time [µs]		UART Max Baud rate [bits/s]	
	SDR	DDR	SDR	DDR
25	6.40	3.20	234,375	468,750

50	3.20	1.60	468,750	937,500
75	2.13	1.07	703,125	1,406,250
100	1.60	0.80	937,500	1,875,000
150	1.07	0.53	1,406,250	2,812,500
200	0.80	0.40	1,875,000	3,750,000
250	0.64	0.32	2,343,750	4,687,500
300	0.53	0.27	2,812,500	5,625,000
400	0.40	0.20	3,750,000	7,500,000
600	0.27	0.13	5,625,000	11,250,000
800	0.20	0.10	7,500,000	15,000,000
1000	0.16	0.08	9,375,000	18,750,000

For the UART channels 3 consecutive samples are being encoded in every LTPI Frame (TXD[0-2] and RXD[0-2]) as opposed to single sample encoded for GPIOs. The first samples are stored in TXD[0]/RXD[0] and last sample per frame is stored in TXD[2]/RXD[2]. In order to maintain the sampling/regeneration frequency duty cycle close to 50% as shown in Figure 15 the Frame Generation/Parsing time needs to be considered.

An example of how UART sampling can be aligned to the byte positions of Frame generated on TX side (SCM) and bytes position of the same frame received on RX side (HPM) is presented in the Figure 16 below. As shown in this example the UART samples need to be ready when byte number 7 is generated and transmitted over LTPI. It means that prior to that 3 samples need to be collected assuming even distribution of samples over time. For Frame N Sample 0 is actually collected when byte 10 of previous Frame (N-1) was sent. Sample 1 is collected when byte 0 is sent over LTPI and Sample 2 is collected when byte 7 with UART samples is generated and sent. This way an even distribution of samples over time is maintained between Frames. On Receiver side the same distribution needs to be maintained when received samples are used to generate signal on UART interface. When byte 7 of Frame N is received Sample 0 is used to generate signal on UART interface, the next Sample 1 from Frame N is generated when byte 12 is received. The last sample is generated on byte 1 of next Frame (N+1). For simplicity, the samples in this example are generated as soon as UART byte is received and before CRC of the whole frame in checked. The final implementation needs to verify the CRC before the received samples are considered valid and can be generated on the UART. The generation of the valid samples can be started on either byte 15 when CRC is fully received and verified or delayed by the full frame width.

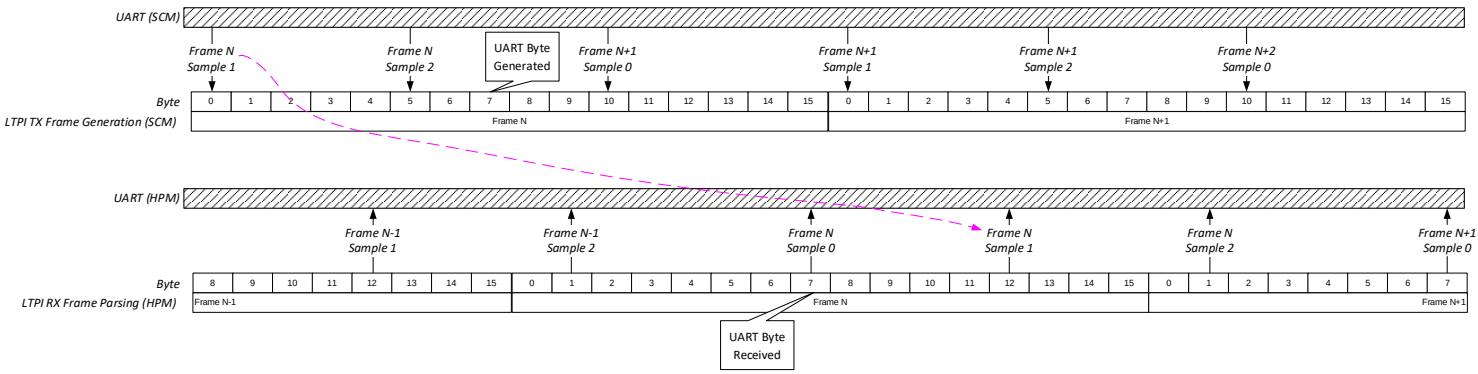

Figure 16 LTPI UART Sampling distribution across LTPI Frames

Together with RXD and TXD signals UART channel supports flow control signals tunneling with RTS and CTS support. The flow control signals are not over sampled but are encoded in every LTPI Frame similar to Low Latency GPIOs. The Table 8 below shows the encoding of 2 x UART channels. Each channel is encoded on 4 bits for TX and 4 bits for RX direction. These encodings will be tunneled separately through LTPI interface. In this example the direction is considered from SCM perspective hence UART0/1 TX encoding applies for the LTPI frames tunneled from SCM to HPM. The UART0/1 RX Encoding applies for the LTPI frames which are tunneled from HPM to SCM.

Table 8 UART Channels Encoding

Name	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
UART0/1 TX Encoding	**UART1**				**UART0**			
	RTS 0: Low 1: High	TXD[2] 0: Low 1: High	TXD[1] 0: Low 1: High	TXD[0] 0: Low 1: High	RTS 0: Low 1: High	TXD[2] 0: Low 1: High	TXD[1] 0: Low 1: High	TXD[0] 0: Low 1: High
UART0/1 RX Encoding	**UART1**				**UART0**			
	CTS 0: Low 1: High	RXD[2] 0: Low 1: High	RXD[1] 0: Low 1: High	RXD[0] 0: Low 1: High	RTS 0: Low 1: High	RXD[2] 0: Low 1: High	RXD[1] 0: Low 1: High	RXD[0] 0: Low 1: High

2.2.1.3 I2C/SMBus Interfaces

I2C/SMBus SCL and SDA signals are bi-directional and can be actively driven by Controller or Target devices during the I2C/SMBus transaction. As a result, SCL and SDA signals cannot be tunneled directly through LTPI in similar way as GPIO or UART signals. In LTPI the I2C/SMBus Channel implements the event-based model where the encoded I2C/SMBus event/states are captured on one side and recovered on the other side of LTPI. When the I2C/SMBus Event is generated by the Controller (e.g., START

condition) the Controller's local SCL line is stretched by the LTPI I2C/SMBus Relay until Remote event is received. When the remote event is received it is recovered on Controller local bus and then the SCL is released from stretch condition (the SDA bus setup time must be maintained by LTPI I2C/SMBus Relay before SCL is released). The I2C/SMBus SDL and SDA signal states are being captured and controlled by so called I2C/SMBus Relays defined by LTPI (see Figure 13). The I2C/SMBus Relay block diagram is presented on the Figure 17 below.

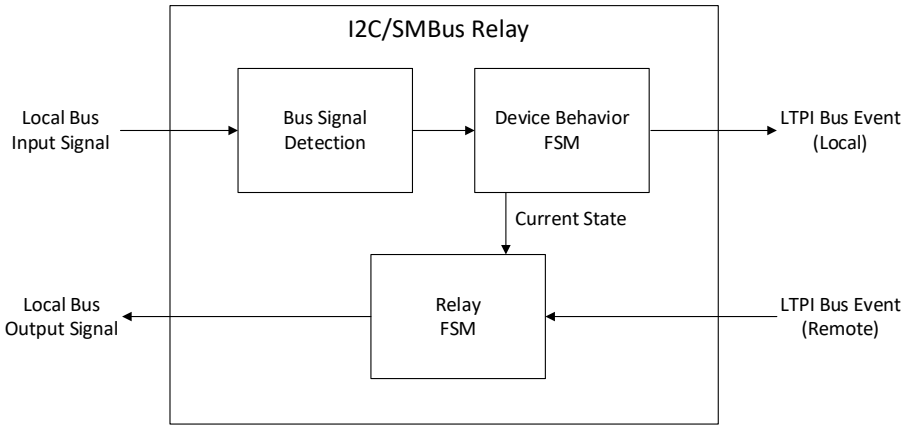

Figure 17 I2C/SMBus Relay Block Diagram

SMBus Relay block integrates the following functions:

- Detection of local bus signal
- Translation of local bus signals into local events
- Management of the clock stretching condition on local bus
- Monitoring of remote events coming from LTPI interface
- Recovery of the remote events on local signals
- Tracking of the consistancy of transaction
    - Expected state transitions (e.g. SCL driven low by controller after START condition)
    - Direction of transaction based on bit count and R/W bit at the end of Address byte
    - SMBus timings requirements and SMBus timouts
    - Bus resets for I2C and SMBus cases

Bus Signal Detection module detects bus signal event from SCL and SDA input, such as rising and falling edge. Device Behavior FSM module uses the detection results to control the bus state transition. It will generate local event based on its current state. Local event is sent on SCL falling edge on LTPI interface to remote LTPI. Relay FSM module is used to monitor remote event and stretch SCL. It will recover the enabling sigal based on its current event and remote event received. The Figure 18 below shows an

example of single I2C/SMBus Channel between Controller located on the SCM in BMC SoC and Target located on HPM in PSU Device.

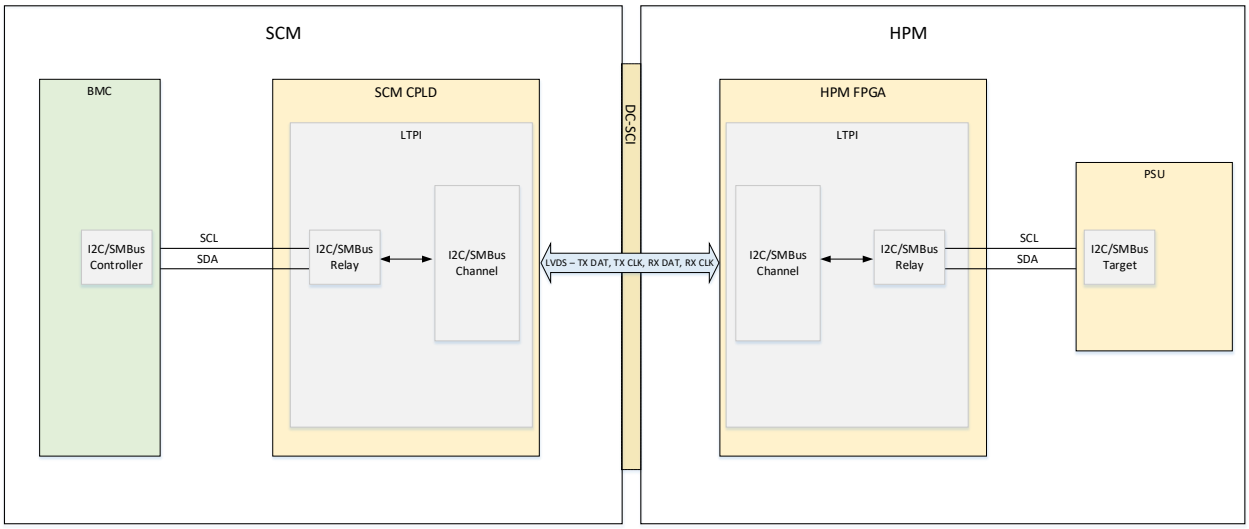

Figure 18 Example of I2C/SMBus Channel Controller-Target connection

The Bus Events/States during I2C/SMBus transactions observed by the Controller connected to SCM CPLD LTPI and Target connectored to HPM FPGA LTPI are presented on Figure 19 below. The red line is used to show clock stretching on SCL buses.

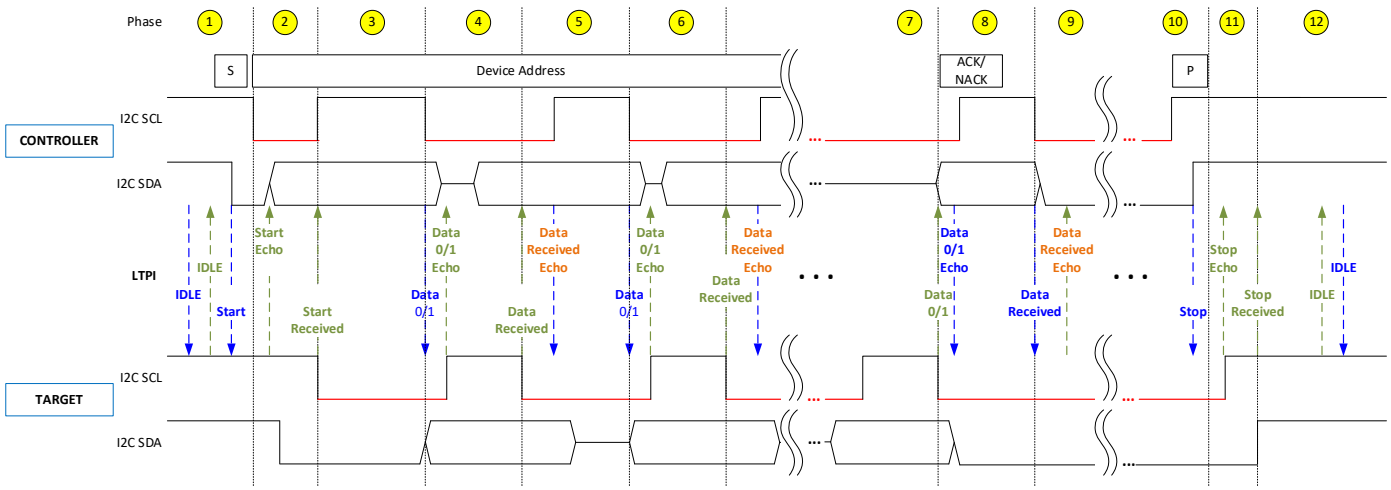

Figure 19 I2C/SMBus Events and clock stretching for Controller and Target side

Regarding the state transitions defined in the waveforn on Figure 19 above the LTPI I2C/SMBus Relays in Controller and Target side will follow the behavior and event exchange flows as defined in Table 9 below.

Table 9 I2C/SMBus Events flow between Controller and Target LTPI I2C/SMBus Relays

Phase	Controller	Target	I2C/SMBus Event	LTPI Event Direction
1	1. Signal Detection Module detects SDA Falling Edge with SCL High 2. Device Behavior FSM detects START condition and Device Behavior FSM start sending Start Event to on LTPI Interface 3. Signal Detection Module detects SCL falling edge and Relay FSM triggers SCL stretching on local SCL and waits for START Received	Bus Idle	Start	Controller to Target
2	Clock Stretching	1. Device behavior FSM receives a Start Event and enters START state. 2. Relay FSM starts sending Start Echo Event. Echo events are used to indicate that event was correctly received and will be processed. After sending Start Echo the Relay FSM starts to generate a START Condition signal on SCL and SDA local bus. 3. SCL falling edge is correctly generated on the bus. 4. Device behavior FSM starts sending Start Received through LTPI interface. 3. Relay FSM triggers SCL stretching on local SCL and waits for next event.	Start Echo Start Received	Target to Controller
3	1. Relay FSM received Start Received Event and stops clock stretching on local bus. 2. Controller starts sending the data bit by triggering raising edge on the bus. 3. Signal Detection Module detects SCL rising edge and SDA data bit 4. Signal Detection Module detects SCL falling edge and Device Behavior FSM starts sending a Data0/1 Event on LTPI Interface 5. Relay FSM triggers SCL stretching on local SCL and waits for Data Received Event	Clock Stretching	Data0/1	Controller to Target

4	Clock Stretching	1. Device behavior FSM receives a Data0/1 Event 2. Relay FSM starts sending Data Echo event and generates a Data0/1 bit value on SDA then releases the SCL from clock stretching condition. 3. SCL is pulled low after SCL high duration time 4. Device behavior FSM starts sending Data Received through LTPI interface. 3. Relay FSM triggers SCL stretching on local SCL and waits for next event.	Data Echo Data Received	Target to Controller
5	1. Relay FSM received Data Received Event stops clock stretching on local bus. 2. Device Behavior FSM starts sending Data Received Echo Event. 3. Controller starts sending the data bit by triggering raising edge on the bus. 4. Signal Detection Module detects SCL rising edge and SDA data bit 5. Signal Detection Module detects SCL falling edge and Device Behavior FSM starts sending a Data0/1 Event on LTPI Interface 6. Relay FSM triggers SCL stretching on local SCL and waits for Data Received Event	Clock Stretching	Data Received Echo Data 0/1	Controller to Target
6	Clock Stretching	1. Device behavior FSM receives a Data Received Echo first – this is indication of end of previous bit transmission. Next Device behavior FSM receives a new Data 0/1 Event. 2. Relay FSM starts sending Data Echo event and generates a Data0/1 bit value on SDA then releases the SCL from clock stretching condition. 3. SCL is pulled low after SCL high duration time 4. Device behavior FSM sends Data Received through LTPI interface. 3. Relay FSM triggers SCL stretching on local SCL and waits for next event.	Data Echo Data Received	Target to Controller
…	…	…	…	…
Change in direction of data transaction i.e., ACK/NACK in this example but it will be the same for Data Bits in Read Transaction				
…	…	…	…	…

7	Clock Stretching	1. Target sets the data bit on SDA 2. Relay FSM drives the SCL raising edge on the bus. 3. Signal Detection Module detects SCL rising edge and SDA data bit 4. Signal Detection Module detects SCL falling edge and Device Behavior FSM starts sending a Data0/1 Event on LTPI Interface 5. Relay FSM triggers SCL stretching on local SCL and waits for Data Received Event	Data0/1	Target to Controller
8	1. Device behavior FSM receives a Data0/1 Event 2. Relay FSM starts sending Data Echo event and generates a Data0/1 bit value on SDA then releases the SCL from clock stretching condition. 3. SCL is pulled low after SCL high duration time 4. Device behavior FSM starts sending Data Received through LTPI interface. 3. Relay FSM triggers SCL stretching on local SCL and waits for next event.	Clock Stretching	Data Echo Data Received	Controller to Target
9	Clock Stretching	1. Relay FSM received Data Received Event. 2. Device Behavior FSM starts sending Data Received Echo Event.	Data Received Echo	Target to Controller
…	…	…	…	…
10	1. Signal Detection Module detects SDA Rising Edge with SCL High 2. Device Behavior FSM detects STOP condition and starts sending Stop Event.	Clock Stretching	Stop	Controller to Target
11	1. Device Behavior FSM Waits for Stop Echo and Stop Received Event. 2. Signal Detection Module waits for new START condition. If new START condition is detected it should be deferred (using clock stretching) until Stop Received event is received for previous STOP condition.	1. Device behavior FSM receives a Stop Event. 2. Relay FSM sends Stop Echo event and starts to generate a STOP Condition signal on SCL and SDA local bus. 3. Device behavior FSM sends Stop Received through LTPI interface. 4. Relay FSM triggers bus Idle condition on the bus.	Stop Echo Stop Received	Target to Controller

12	Bus Idle	Bus Idle	Idle	N/A

The flow outlined in the Figure 19 above is intended to provide a high-level concept of the SMBus/I2C Relays and how clock stretching is used to compensate for LTPI turnaround time. The I2C/SMBus Relays state machine will have to handle bus errors, CRC errors, timing dependencies, track the stage and progress of transaction, etc. The full micro-architecture definition of the SMBus/I2C LTPI Relays is out of scope of the LTPI Specification and should be part of LTPI implementation design specification. The SMBus/I2C implementations needs to comprehend:

1. START and STOP condition detection and generation
2. Bus consistency between START and STOP conditions
3. Direction of operation on the bus for given phase of the transaction
4. Bit count within byte boundary
5. ACK/NACK conditions
6. Read/Write conditions
7. Repeated Start condition
8. SMBus timeouts including maximum clock stretching allowed
9. Bus recovery and termination of transaction in case of errors on LTPI link

The events sent between SCM and HPM for a given SMBus/I2C Channel are sent continuously until new event is generated. Every Non-Idle event sent requires:

1. Echo Event (as defined in the Table 10) – the purpose of Echo Event is to let the sender know that the Events was correctly received and is being processed.
2. Received Event (as defined in the Table 10) – the purpose of Received Events is to indicate that the state of the bus indicated by the Event was correctly regenerated on the I2C/SMBus bus.

The Received Events do not require to be confirmed back with Echo message with one exception for Data Received Events. An Echo is sent for this event to differentiate between subsequent Data Events within given byte transmission i.e.:

1. New Data Event is started by one end and sent continuously on LTPI
2. When data bit is received by the other end it starts sending Data Echo continuously to confirm reception of Data Event
3. When Data bit is regenerated on physical bus Data Received event starts to be sent continuously to confirm regeneration. The other end is still sending Data Event which should be ignored until Data Received Event Echo is received as differentiator between subsequent Data Events.
4. When Data Received event is received by the other end Data Received Echo is transmitted continuously until new Data Event is generated

The I2C/SMBus Events/States that are tunneled through LTPI I2C/SMBus Channel are defined in the Table 10 below with binary encodings.

Table 10 I2C/SMBus Events Bit Encoding

Bus Event	Direction	Description	Encoding Event[4b]
Idle	Bidirectional	I2C/SMBus LTPI Relay is in Idle state.	0b0000
Start	Controller to Target	Start Event detected on Controller side.	0b0001
Start Received	Target to Controller	Start Event recovered on Target side.	0b0010
Stop	Controller to Target	Stop Event detected on Controller side.	0b0011
Stop Received	Target to Controller	Stop Event recovered on Target side.	0b0100
Data Received	Bidirectional	SDA bit value recovered on remote side.	0b0101
Data 0	Bidirectional	Send SDA bit value 0.	0b0110
Data 1	Bidirectional	Send SDA bit value 1.	0b0111
Start Echo	Target to Controller	Event used to indicate that Start Event was correctly received by Target side. If the Echo is not received within expected time i.e., Frame turnaround time (typically 1-2 LTPI Frames), the event can be repeated immediately without waiting for Start event timeout.	0b1000
Stop Echo	Controller to Target	Optional Stop Event correctly received on Target side. If the Echo is not received within expected time i.e., Frame turnaround time (typically 1-2 LTPI Frames), the event can be repeated immediately without waiting for Start event timeout.	0b1001
Data 0 Echo	Bidirectional	Send SDA bit value 0 received correctly. If the Echo is not received within expected time i.e., Frame turnaround time (typically 1-2 LTPI Frames), the event can be repeated immediately without waiting for Start event timeout.	0b1010
Data 1 Echo	Bidirectional	Send SDA bit value 1 received correctly. If the Echo is not received within expected time i.e., Frame turnaround time (typically 1-2 LTPI Frames), the event can be repeated immediately without waiting for Start event timeout.	0b1011
Data Received Echo	Bidirectional	Event used to indicate that Data Received Event was correctly received. It is used to differentiate between consecutive Data 0/1 events during byte transmission.	0b1100
Reserved	Reserved	Reserved for future use.	0b1101-0b1111

The encoding of I2C/SMBus Events in LTPI frame mapping is defined in the Table 11 below. As I2C/SMBus Events are encoded using 4-bit encoding, a single 8-bit LTPI Frame field can fit 2 I2C/SMBus interfaces.

Table 11 I2C/SMBus Bit Fields Encoding in LTPI Frame

Name	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
I2C/SMBus 0/1 Encoding	I2C/SMBus1				I2C/SMBus0			
	I2C/SMBus Event as defined in Table 10				I2C/SMBus Event as defined in Table 10			

2.2.1.4 Data Bus Interfaces

The purpose of the Data Channel is to allow for tunneling standardized Read and Write Requests to Addressable Memory Spaces used to interconnect IP blocks within FPGA/CPLD such as Avalon Memory Mapped interface (Avalon-MM), Wishbone interface or similar. For other Memory Mapped interfaces, a mapping and encoding needs to be defined for the purpose of the specific design using it or added as extension to DC-SCM LTPI specification.

Data bus interface is used to issue Read and Write Requests to addressable memory interface. The LTPI converts those requests to encoding in LTPI Frames and tunnel them via LVDS link to the CPLD/FPGA device where they are decoded and targeted at the IP Block within FPGA/CPLD that supports standardized memory mapped interface. The use of data channel provides a lot of flexibility and ability to interface remote IP without the need to duplicate the same IP in both CPLD/FPGA devices on SCM and HPM.

The supported memory access operations with LTPI tunneling are 1-4 byte Writes and 1-4 byte Reads. The following parameters are associated with Read and Write Requests:

- Command – Identifies the Memory Operation
- Address – Address of Read or Write operation
- Data – 1 to 4 bytes of Data
- BE – Byte Enable mask indicates number of bytes (1 to 4)

The encoding of Data Channel Commands that are passed through LTPI interface are defined in the Table 12 below.

Table 12 Data Bus Command encoding.

Command	Parameters	Description	Encoding Event[8b]
Read Request	Address	Command indicates a request to read data from specific address. Byte Enable mask indicates the number of bytes to read (1 – 4).	0h00
Write Request	Address Data	Command indicates a request to write data to a specific address. Byte Enable mask indicates the number of bytes to write (1 – 4).	0h01
Read Completion	Address Data	Command indicates a read operation is completed. Byte Enable mask indicates the number of bytes valid in a read completion frame (1 – 4).	0h02

Write Completion	Address	Command indicates a write operation result.	0h03
CRC Error	N/A	Command indicates a Data Frame was received with CRC error and operation won't be completed.	0h04
Reserved	Reserved	Reserved for future use	0h03-0hFF

The encoding of the Memory Read and Write Transaction is defined in the Table 13 below.

Table 13 Memory Read and Write operations Encoding

Name	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
Memory Read	Command = 0h00							
	Address Byte 3							
	Address Byte 2							
	Address Byte 1							
	Address Byte 0							
	Reserved				**Byte Enable**			
	Reserved				Byte 3	Byte 2	Byte 1	Byte 0
Memory Read Completion	Command = 0h02							
	Address Byte 3							
	Address Byte 2							
	Address Byte 1							
	Address Byte 0							
	Read Operation Status				**Byte Enable**			
	0h: Success 1h: Invalid Access 1h – Fh: Reserved				Byte 3	Byte 2	Byte 1	Byte 0
	Data Byte 3							
	Data Byte 2							
	Data Byte 1							
	Data Byte 0							
Memory Write	Command = 0h01							
	Address Byte 3							
	Address Byte 2							
	Address Byte 1							
	Address Byte 0							
	Byte Enable							
	Reserved	Reserved	Reserved	Reserved	Byte 3	Byte 2	Byte 1	Byte 0
	Data Byte 3							
	Data Byte 2							
	Data Byte 1							
	Data Byte 0							
Memory Write Completion	Command = 0h03							
	Address Byte 3							
	Address Byte 2							
	Address Byte 1							
	Address Byte 0							
	Write Operation Status				Reserved			
	0h: Success				Reserved			

	1h: Invalid Access 1h – Fh: Reserved	

2.2.1.5 OEM Interfaces

Definition of methods used to tunnel proprietary OEM interfaces in LTPI is out of scope. OEM can follow the examples of how other channels are tunneled in LTPI and follow similar methods if applicable to capture and recover OEM proprietary interfaces. The OEM channel is a placeholder for innovation and customization of LTPI.

2.3 LTPI Clock Topology

The LTPI interface is using 2 LVDS links in each direction. One of the links is a clock used to synchronize the TX and RX PHY on both sides of the link. For a given LTPI in either SCM CPLD or HPM FPGA the local TX and RX clocks are independent. The clock source synchronous mode is use for the TX clock source to align the phase shift on the LVDS IO Links. RX side will use the incoming clock as sampling clock for the data link. The example topology of how LTPI clock topology can be implemented is presented in the Figure 20 below. Local System clock is used to drive the LTPI logic except for TX and RX parts hence CDC FIFO is used in this example.

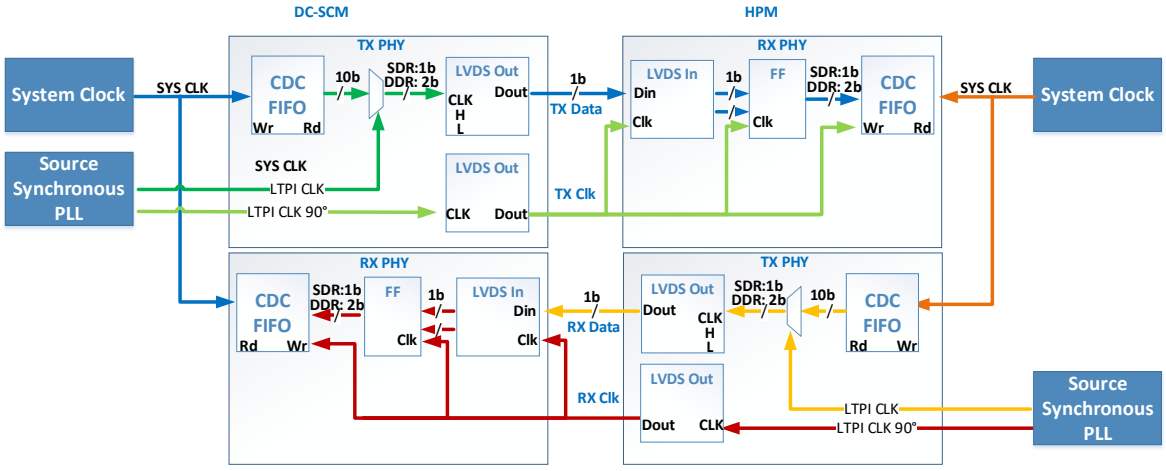

Figure 20 LTPI Clock Topology

2.4 CRC Checksum

Each LTPI Frame contains the CRC checksum. This is CRC-8 with following polynomial:

$$x^8 + x^2 + x^1 + 1$$

The polynomial initial value is defined as '0'. The CRC is calculated for entire payload fields after Comma Symbol of the LTPI Frame. CRC shall be calculated without input bits inversion and without result inversion/reflection.

2.5 CRC Errors Handling

When the CRC Error is detected the LTPI Frame is dropped. The CRC Error and Frame dropped impact on LTPI channels will differ depending on the channel. The Table 14 provides a summary of how LTPI channels are impacted by CRC errors and what is the recovery method.

Table 14 CRC Error impact on LTPI channels

<table>
<tr><th>Channel</th><th>Impact</th><th>Recovery</th></tr>
<tr><td>Low Latency GPIO</td><td>The state of GPIOs is maintained from the previous Frame. The next Frame with correct CRC will provide updated states of Low Latency GPIOs.</td><td>Not needed if CRC error is not permanent. If it is permanent link is lost and requires re-initialization.</td></tr>
<tr><td>Normal Latency GPIO</td><td>The state of subset of Normal Latency GPIOs is maintained based on previous update. The next update will happen after (N-1) number of Frames with correct CRC where N is as defined in 2.2.1.1</td><td>Not needed if CRC error is not permanent. If it is permanent link requires re-initialization.</td></tr>
<tr><td>SMBus</td><td>If the impacted Frame do not contain any active Bus state i.e., Bus is in Idle State there's no impact. If the transaction is in progress the impact of CRC errors will result in the transaction termination through SMBus timeout.

If all the devices on the Bus on Controller Side (e.g., BMC on DC-SCM) and Target Side (e.g., Sensors on HPM) are SMBus Devices then the LTPI SMBus channel Relays will rely on SMBus Timeouts as the mechanism to terminate the ongoing transaction on the bus and resume to bus idle state. For any active SMBus state the CRC Error and Frame dropped will cause the timeout on the Bus on respective side i.e., Controller and Target as presented in the table below.

<table>
<tr><th>Event Lost</th><th>SMBus Timeout</th></tr>
<tr><td>Idle</td><td>-</td></tr>
<tr><td>Start</td><td>Controller</td></tr>
<tr><td>Start Received</td><td>Controller & Target</td></tr>
<tr><td>Stop</td><td>Target</td></tr>
<tr><td>Stop Received</td><td>-</td></tr>
<tr><td>Data 0</td><td>Controller & Target</td></tr>
<tr><td>Data 1</td><td>Controller & Target</td></tr>
<tr><td>Data Received</td><td>Controller & Target</td></tr>
</table>
</td><td>LTPI SMBus Relays should keep track of the expected conditions on the bus and expected events and reset the Relay state if timeout is triggered on Controller side and Target Side. The LTPI CSR can also provide a way to reset the SMBus Relays explicitly by the BMC FW.

Special considerations:
1. Some simple SMBus Devices e.g., temperature sensors might not handle the SMBus Timeouts correctly. This needs to be comprehended by SMBus Relay implementation and validation of particular system design.
2. Alternatively, if SMBus timeouts are not supported by all the devices on the bus the same approach should be considered as for I2C-based devices below.
3. When transaction is terminated on one side e.g., on the Controller side then the LTPI SMBus Relay on this side needs to wait for transaction termination on the other side before it allows for new transaction to be started. It can be achieved by using clock stretching mechanism when CLK goes low after new START condition is triggered by the Controller. This might result in another timeout if remote side is taking long to recover.</td></tr>
</table>

I2C	If the impacted Frame do not contain any active Bus state i.e., Bus is in Idle State there's no impact. If the transaction is in progress the impact of CRC errors will be termination of the transaction through explicit NACK or STOP condition.	When the CRC error is observed on the Controller side e.g., SCM it needs to terminate the transaction to controller on the bus by generating NACK conditions on any subsequent bytes on the bus until STOP condition is achieved. For I2C bus on the Target devices side e.g., on the HPM the I2C Relay needs to issue a STOP condition to the device. Special considerations: 1. When transaction is terminated on one side e.g., on the Controller side then the LTPI I2C Relay on this side needs to wait for transaction termination on the other side before it allows for new transaction to be started. It can be achieved by using clock stretching mechanism when CLK goes low after new START condition is triggered by the Controller. 2. When on the Target side the I2C Relay triggers the STOP condition due to CRC errors it needs to first make sure that the Target device is not actively driving the bus during the transaction or is not stretching the clock.
UART	In similar way to GPIO UART will keep the previous state of the interface when Frame is dropped. The next frame with correct CRC will provide update on the UART. From the UART interface perspective this condition would cause jitter if the dropped frame contained the UART signal transition. The correct signal transition will happen with the next correct LTPI frame.	Not needed if CRC error is not permanent. If it is permanent link requires re-initialization.
Data Channel	1. CRC Errors When the CRC error is received the operation on the Data bus cannot be executed. The LTPI side which detected the CRC error should drop the frame and inform the remote side of the CRC error received by sending the dedicated Data Channel Command indicating CRC error. The other LTPI side can make a decision of retransmission or return error to the originator of Data Channel access. 2. Completion Timeouts Additional Error Cases in Data Channel accesses are covered by timeouts i.e., when completion is not received withing given timeout (depends on the data bus used and the IPs behind it) the originator of Data Access request should be notified about timeout error. The LTPI Data Channels should implement the same timeout handling for such cases and completions after timeouts should not be generated. When unexpected completion is	Not needed if CRC error is not permanent. If it is permanent link requires re-initialization.

	received however it should be dropped. 3. Completions with Errors For Read or Write failures such as Invalid Access a Status field in the completion should be used to indicate the type of error.	

For all the cases when CRC error is observed there are dedicated CRC error counters defined to let BMC monitor the health status of the LTPI link. BMC policies can be implemented in the BMC FW to react in case of increased CRC error rates on the link. It is out of the scope of LTPI specification to define what actions BMC should take but those might include resetting of LTPI Link, lowering the LTPI Operational Speed, Disabling of some of the channels, etc.

2.6 8b/10b Encoding

8b/10b encoder translates 8-bit Characters into 10-bit Symbols. Three of the most important features of 10b encodings are:

- Embedding a Clock into the Data.
- Maintaining DC Balance.
- Enhancing Error Detection.

The LTPI uses AC coupling between SCM and HPM on LVDS links hence the 10b encoding is required to maintain DC balance on the link. In LTPI Frame data encoding is represented in 8-bit and needs to be converted into 10-bit with 8b/10b encoding to ensures that the encoded data stream has at least a certain amount of data transitions. The definition of 8b/10b encoding and de-coding is out of scope of LTPI specification and is considered a standard practice in differential links implementation. The recommended encoding scheme for 8b/10b encoding is IBM definition. The LTPI for DC-SCM 2.0 is defined with dedicated clock signal but the implementations using clock recovery from the data signal are not precluded and might be defined in the future versions of the specification.

2.7 LTPI Latency

There are two types of latencies that can be identified in the LTPI that affect LTIP Channels tunneled with IO Frames:

1. Constant Latency – this is a constant latency value for a given LTPI frequency and inter-frame gap (the inter-frame gap is expected to be close to 0 in most LTPI implementations). As a result, the Constant Latency is defined as time between the beginning of consecutive LTPI IO Frames as presented in the Figure 21 below.

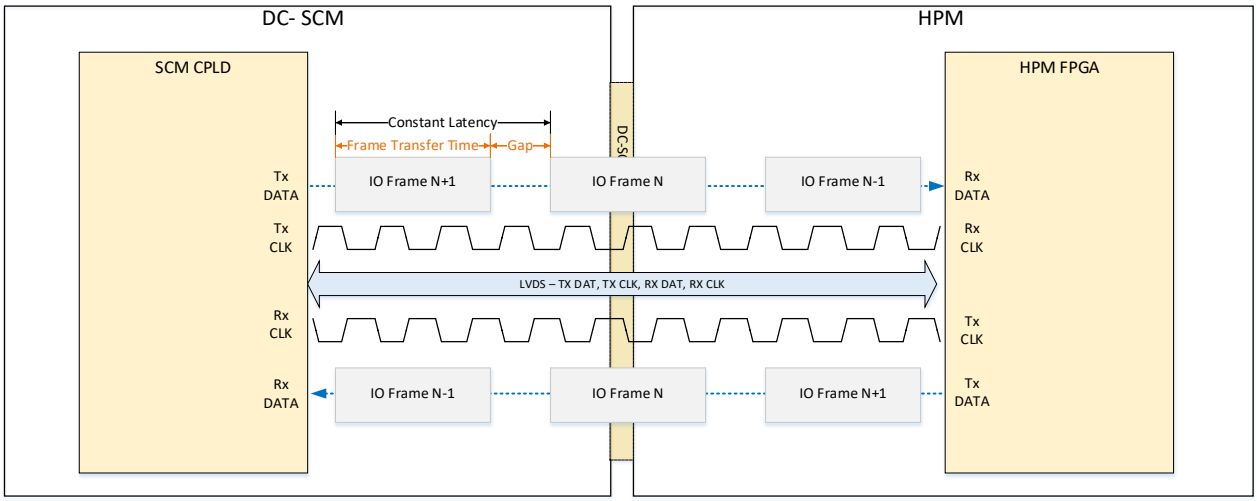

Figure 21 LTPI Constant Latency

2. Random Latency – Additional Latency in IO Channels can be introduced by Data Channel Access Frames interleaving the IO Frames. Each Random Access to the Data Channel will introduce an additional latency needed to Transfer the Data Frame. As a result, the total Latency for the IO Frame will intermittently be extended by the time needed to transfer the Data Frame as presented in the Figure 22 below.

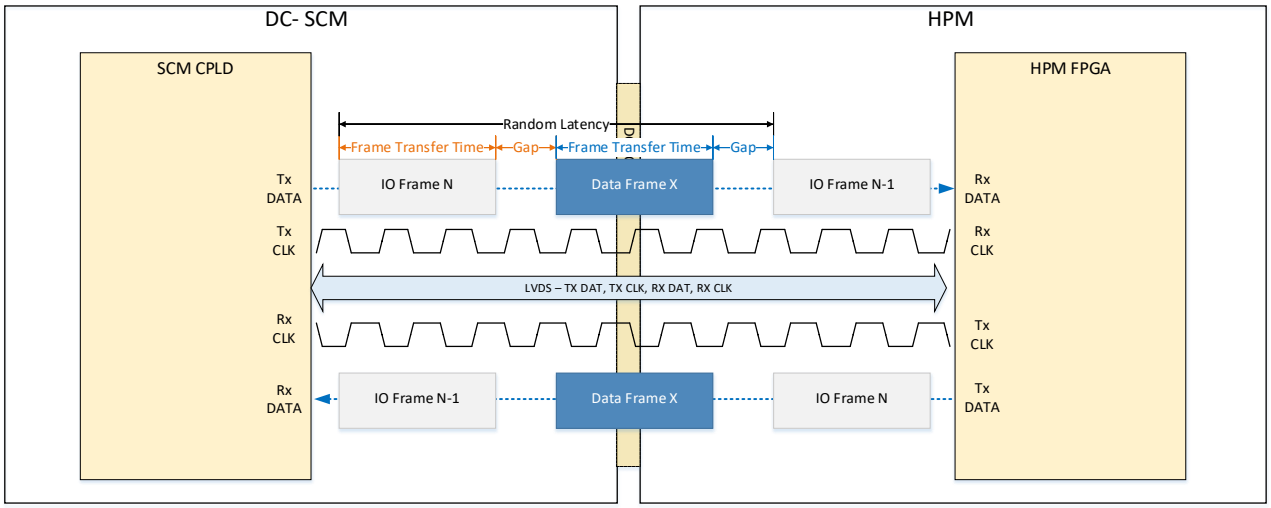

Figure 22 LTPI Random Data Access Latency

Both types of latencies directly depend on the LTPI clock frequency. The higher the LTPI clock frequency is the lower value of the latency will be. The Random Latency will depend on how many consecutive

Data Access Frames will be allowed to interleave the IO Frames. It is a design decision to optimize LTPI usage and determine the limit.

An example of Latencies expected for different LTPI clock frequencies are presented in the Table 15 with assumptions that the frames are sent back-to-back with no inter-frame gap and a maximum number of consecutive Random Data Access Frames limit set to 2 Data Frames.

Table 15 Frame Transfer Latencies Example

LTPI Clock Speed [MHz]	Frame Transfer Time [µs]		Constand Latency [us]		Random Latency [us]	
	SDR	DDR	SDR	DDR	SDR	DDR
25	6.40	3.20	6.40	3.20	19.20	9.60
50	3.20	1.60	3.20	1.60	9.60	4.80
75	2.13	1.07	2.13	1.07	6.40	3.20
100	1.60	0.80	1.60	0.80	4.80	2.40
150	1.07	0.53	1.07	0.53	3.20	1.60
200	0.80	0.40	0.80	0.40	2.40	1.20
250	0.64	0.32	0.64	0.32	1.92	0.96
300	0.53	0.27	0.53	0.27	1.60	0.80
400	0.40	0.20	0.40	0.20	1.20	0.60
600	0.27	0.13	0.27	0.13	0.80	0.40
800	0.20	0.10	0.20	0.10	0.60	0.30
1000	0.16	0.08	0.16	0.08	0.48	0.24

The impact of LTPI Latency on IO channels will be different depending on specific IO channel. It is outlined in the Table 16 below.

Table 16 LTPI Latency impact on LTPI channels

Channel	Constant Latency	Random Latency
Low Latency GPIO	It defines minimum latency supported for Low Latency GPIOs as a result the shortest GPIO state change which can be supported by Low Latency GPIOs.	No impact. Data Frame supports Low Latency GPIOs and random access to Data Channel should not have impact on Low Latency GPIOs Channel.
Normal Latency GPIO	It defines minimum latency supported for Normal Latency GPIOs. Each Normal Latency GPIO will have a latency of N times higher than Constant Latency where N is as defined in 2.2.1.1 GPIO Interfaces.	Random Latency will increase the Latency of Normal Latency GPIOs depending on the number of Data Frames allowed to be sent in a row over LTPI. Since Normal Latency GPIOs are distributed across N IO Frames as defined in 2.2.1.1 GPIO Interfaces the total impact will be determined by how many Data Channels access can be interleaved within N Frames.
I2C/SMBus	It will impact the time needed to transfer the I2C/SMBus events through LTPI. As a result, the latency will lower the end-to-end I2C/SMBus	Random Latency will increase the Latency of I2c/SMBus channels depending on the number of Data Frames allowed to be sent in a row over LTPI. As a

	Bandwidth. The latency will not have an impact on integrity of the I2C/SMBus interface.	result, the end-to-end I2C/SMBus Bandwidth will be lower.
UART	It will create a delay in UART end-to-end signal. Since UART is an asynchronous interface, the constant latency will not impact the integrity or bandwidth of the UART interface.	It can have an impact on the integrity of the UART signal by creating the jitter on the edge of UART signal transition. To eliminate this impact there needs to be a limit set for the maximum number of consecutive Data Channel Frames that are allowed to be sent over LTPI for a given LTPI speed and maximum UART baud rate supported.
Data Channel	It defines the minimum Data Channel access latency by means of time required to transfer the Data Channel Frame across LTPI link and transfer the response Data Channel Frame back. Additional latency introduced by the design-specific logic which handles the Data Channel requests on SCM and HPM will add to the end-to-end Data Channel access latency. It is out of scope of LTPI specification to define this latency.	N/A

2.7.1 I2C/SMBus End-to-End Bandwidth

Depending on the design and implementation of LTPI I2C/SMBus Relay and the integration with LVDS PHY the latency between the time of the I2C/SMBus Event on the bus e.g., START condition and the time it is correctly received on the other side of LTPI link can be up to 2 Frames as shown in the Figure 23 '*Constant Latency Impact*' below. However, in the optimized CPLD design it's possible to capture the events as the Frame is being generated thus minimizing the latency impact on the I2C/SMBus interface close to single Frame latency. As for the Random Latency impact in a worst-case scenario the Data Channel Frames might interleave the frame when the SMBus Event was detected and the Frame when it is sent as shown on Figure 23 '*Random Latency Impact*' below.

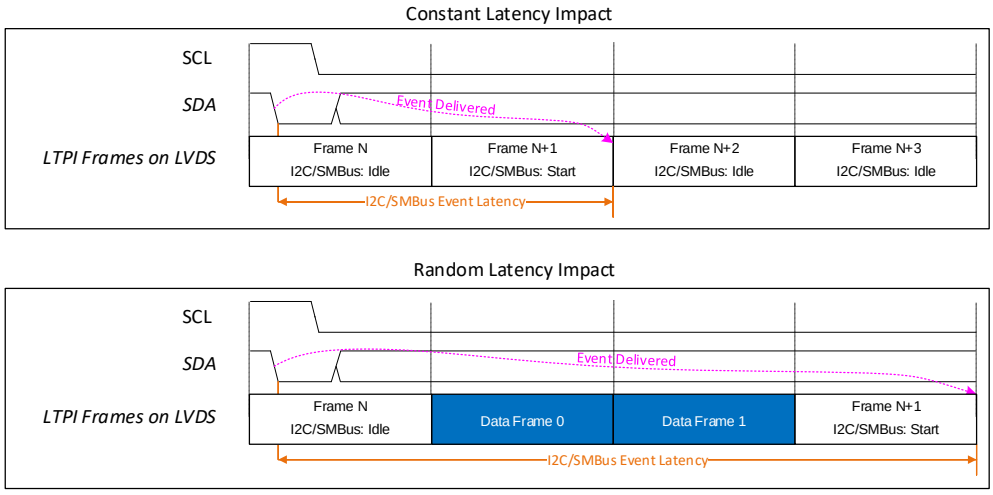

Figure 23 LTPI Latency impact on I2C/SMBus Events

For the examples provided in the Figure 23 and the estimated end-to-end bandwidth can be calculated based on the fact that the I2C/SMBus clock signal needs to be stretched at least by the LTPI latency time including Frame turnaround time. The Table 17 below shows the estimated end-to-end bandwidth for 100kHz and 400kHz I2C/SMBus configurations. The 'Min' value is defined assuming that 2 Data Channel access Frames interleave the I2C/SMBus Event change transfer in both directions. The 'Max' value is defined as the case when no Data Channel access Frame is interleaving the I2C/SMBus Event change transfer. The calculation below is only an example of theoretical worst-case scenario, in optimized implementations the latency can be reduced in many ways e.g., by generating the frame synchronously as it goes on LVDS link or by prioritizing SMBus events updates, which are infrequent, over Data Channel accessed to minimize Random Latency impact on the I2C/SMBus channel. Other optimizations might be applied in specific LTPI Designs.

Table 17 I2C/SMBus Channel End-to-End Bandwidth

LTPI Clock Speed [MHz]	I2C/SMBus @100kHz End-to-End Bandwidth[kHz]				I2C/SMBus @400kHz End-to-End Bandwidth [kHz]			
	SDR		DDR		SDR		DDR	
	Min	Max	Min	Max	Min	Max	Min	Max
25	16.3	35.6	28.1	43.9	18.6	35.6	35.6	65.4
50	28.1	65.4	43.9	61.0	35.6	65.4	65.4	112.4
75	37.0	90.7	53.9	70.0	51.1	90.7	90.6	147.5
100	43.9	112.4	61.0	75.8	65.4	112.4	112.4	175.4
150	53.9	147.5	70.1	82.5	90.6	147.5	147.9	216.5
200	61.0	175.4	75.8	86.2	112.4	175.4	175.4	243.9
250	66.1	197.6	79.6	88.7	131.2	197.6	197.6	264.6
300	70.1	216.5	82.4	90.3	147.9	216.5	215.5	279.3
400	75.8	243.9	86.2	92.6	175.4	243.9	243.9	303.0
600	82.4	279.3	90.4	95.1	215.5	279.3	280.9	331.1
800	86.2	303.0	92.6	96.2	243.9	303.0	303.0	344.8
1000	88.7	318.5	94.0	96.9	264.6	318.5	318.5	354.6

2.7.2 UART End-to-End Integrity

Since the UART is asynchronous interface, the Constant Latency will not have an impact on the integrity of the UART signal between UART TX and UART RX as presented in the Figure 24 '*Constant Latency Impact*' below. Random Latency however can impact the UART signal integrity especially if the Data Channel frames are aligned with the UART signal bit transition. In such case, as presented in the Figure 24 '*Random Latency Impact*' below, the Random Latency will create a Jitter in the signal received on the RX.

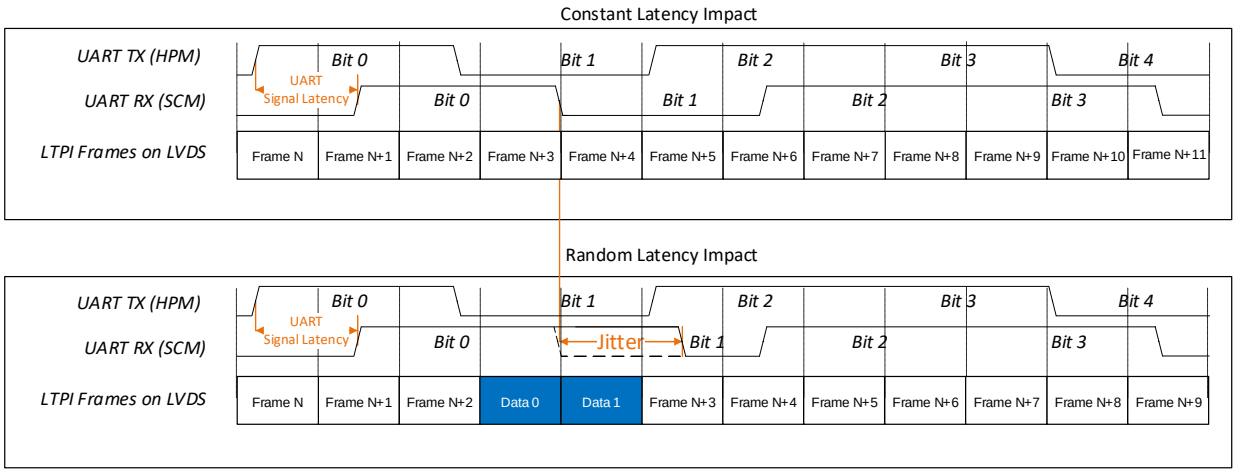

Figure 24 LTPI Latency impact on UART Integrity

As presented on the Figure 25 below the jitter introduced by the Random Latency can lead to reception of a wrong bit level and loss of UART signal. Typical UART HW Receiver oversamples the UART signal 16 times comparing to the UART baud rate as presented on the Figure 25 below. The RX Bit Value is interpreted based on 'Middle Of bit Samples' which refers to samples 7, 8 and 9 below. When the bit transition is delayed due to jitter introduced by Data Channel Frames interleaving IO Frames the UART Signal level is maintained based on previous UART signal state. As a result, the 'Middle Of Bit Samples' are set to incorrect value and the bit affected by Jitter is received with error.

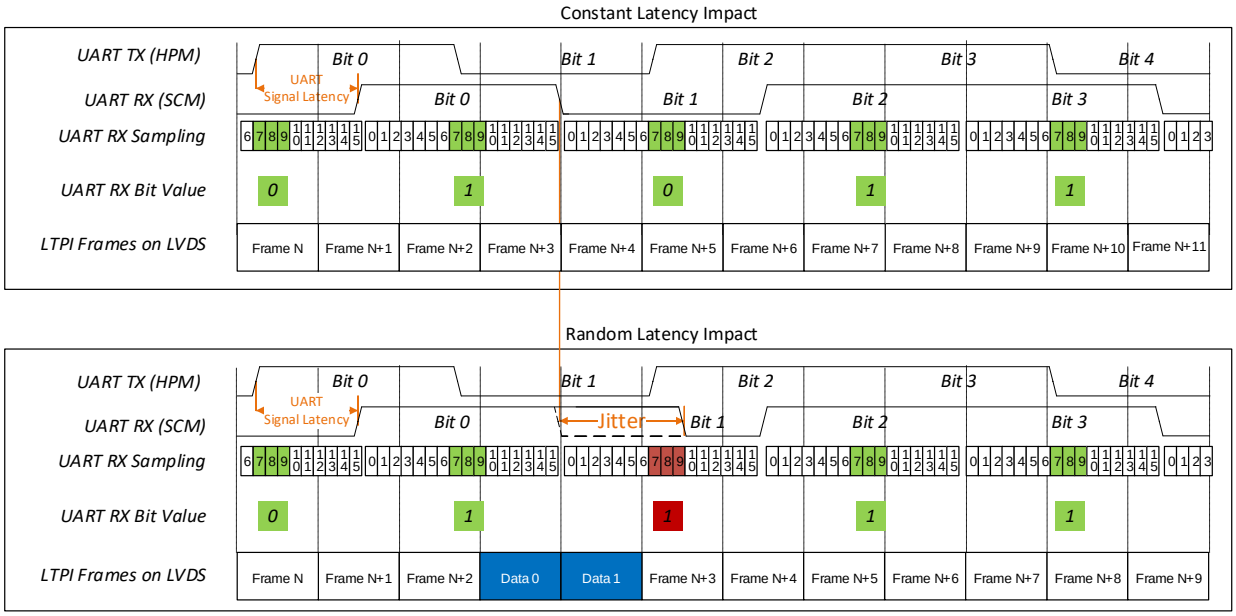

Figure 25 UART signal Jitter impact RX signal integrity

In order to eliminate Jitter impact on the UART signal integrity the LTPI implementation shall limit the number of consecutive Data Channel Frames that can interleave the IO Frames on the LTPI link. The maximum Jitter that is introduced by the Data Channel Frames shall be lower than the time between RX Sample 0 and the beginning of 'Middle Of Bit Samples'. This is presented in the example on the Figure 26 below where the number of consecutive Data Channel Frames was limited to 1. This is just an example to demonstrate the Jitter impact and how it can be avoided. The final implementation and decision of maximum consecutive Data Frames allowed will depend on the following parameters:

1. LTPI Clock Frequency
2. DDR or SDR LTPI Mode
3. UART Baud Rate

Additionally, some LTPI designs might follow specific optimizations targeted to maximize the number of Consecutive Data Frames e.g., interrupt Data Frames transfer when the change of the UART signal is observed. It is out of scope to LTPI specification to define such optimizations.

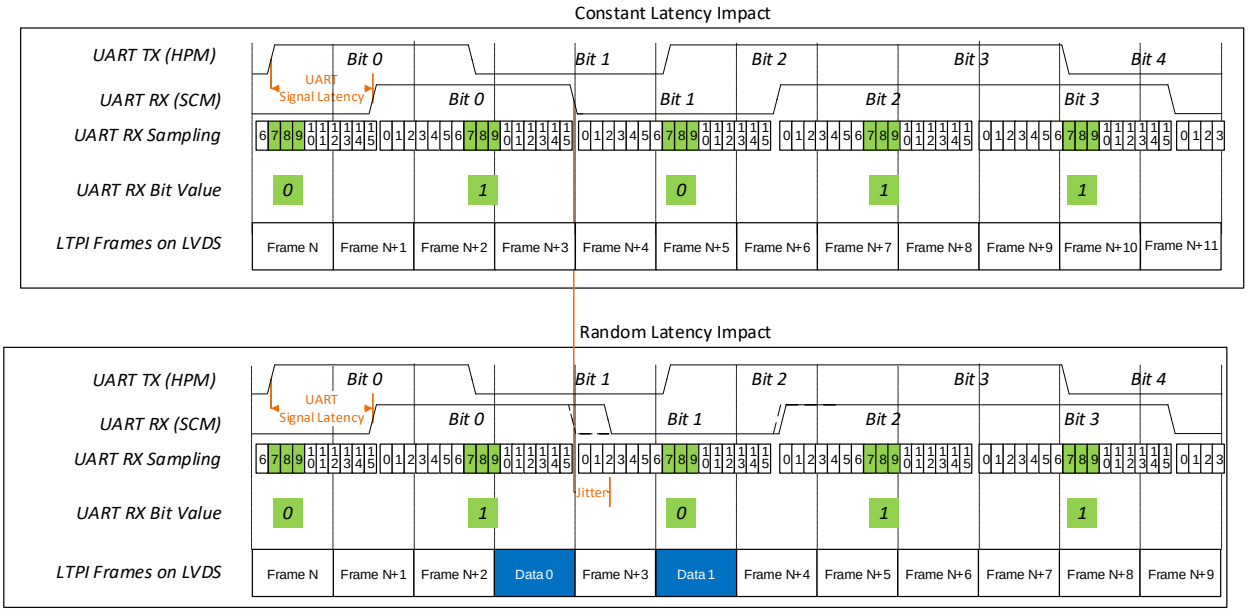

Figure 26 Reduced Jitter impact by limited Data Channel Frames

The Table 18 below presents an example for maximum number of consecutive Data Frames that can be interleaved between IO Frames without affecting UART integrity. The example assumes worst-case and no specific optimizations in the Design of LTPI UART and Data Channel.

Table 18 Example of maximum number of consecutive Data Channel Frames

	9600 [bits/s]	57600 [bits/s]	115200 [bits/s]

LTPI Clock Speed [MHz]	SDR	DDR	SDR	DDR	SDR	DDR
25	7	14	1	2	0	1
50	14	28	2	4	1	2
75	21	42	3	7	1	3
100	28	56	4	9	2	4
150	42	85	7	14	3	7
200	56	113	9	18	4	9
250	71	142	11	23	5	11
300	85	170	14	28	7	14
400	113	227	18	37	9	18
600	170	341	28	56	14	28
800	227	455	37	75	18	37
1000	284	569	47	94	23	47

3 LTPI Frames Definition

The LTPI defined several Frame Types that are used during different phases of LTPI operation. All frames used on the LTPI interface are aligned to 16 symbols including Comma Symbol (160 bit using 10b encoding). The Table 19 below contains a summary of the LTPI Frame types with respect to a different comma symbol used.

Table 19 LTPI Frame Types summary

Comma Symbol	Size	Frame Type	Description
K28.5	16 symbols	Link Detect and Link Speed Selection	Initial frame transmitted during LTPI bring-up flow to provide information about supported LTPI speed capabilities. Frame is also used to achieve DC balance on the link and select target link speed.
K28.6	16 symbols	Capabilities Advertise and Configure/Accept	Frame is used to advertise supported capabilities (Channels and features supported by LTPI) and configure enabled capabilities supported by DC-SCM and HPM LTPI endpoints.
K28.7	16 symbols	LTPI Operational Frame	Frame use during normal LTPI operation after interface initialization and configuration is completed. Frame carries all the channels supported by given LTPI implementation.

3.1.1 Link Detect and Link Speed Selection Frames

The Link Detect and Link Speed Selection are initial frames transmitted during LTPI bring-up flow to provide information about supported LTPI speed capabilities. Frame is also used to achieve DC balance on the link and select target link speed. Refer to the 4 LTPI Link Initialization and Operation for more details. The Frame Subtype field is used to identify Link Detect and Link Speed Fame as defined in the Table 20 below.

Table 20 Link Detect and Link Speed Frame Type and Subtypes

Frame Type	Comma Symbol	Frame Subtype Field Value
Link Detect	K28.5	0h00
Link Speed		0h01

3.1.1.1 Link Detect Frame

The Link Detect Frame is used to exchange information about supported LTPI version and operating speed capabilities. The Speed Capabilities is defined in the Table 21 below. Bitfields in the Speed capabilities WORD are referring to supported frequencies by LTPI implementation defined as multipliers of Base Frequency which is defined as 25MHz at SDR (Single Data Rate) mode for DC-SCM LTPI. The highest bit is indicating whether a Dual Data Rate is supported by LTPI I/O. A given speed is supported when a bit is set to 1. LTPI implementations can support any combination of speed capabilities. All implementations need to support Base Frequency 25MHz.

Table 21 Speed Capabilities Encoding

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	Base Freq. x12 300MHz	Base Freq. x10 250MHz	Base Freq. x8 200MHz	Base Freq. x6 150MHz	Base Freq. x4 100MHz	Base Freq. x3 75MHz	Base Freq. x2 50MHz	Base Freq. x1 25 MHz
1	Dual Data Rate 0: SDR 1: DDR	Reserved	Reserved	Reserved	Base Freq. x40 1GHz	Base Freq. x32 800MHz	Base Freq. x24 600MHz	Base Freq. x16 400MHz

The Link Detect Frame is defined in the Table 22 below.

Table 22 Link Detect Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 20
1	8	Frame Subtype								Frame Subtype as defined in Table 20

2	8	LTPI Version Major	LTPI Version Minor	BCD Encoded LTPI Version [LTPI Version Major].[LTPI Version Minor]
3	8	Speed Capabilities		Speed Capabilities as defined in Table 21
4	8			
5	8	Reserved		Reserved
6	8	Reserved		Reserved
7	8	Reserved		Reserved
8	8	Reserved		Reserved
9	8	Reserved		Reserved
10	8	Reserved		Reserved
11	8	Reserved		Reserved
12	8	Reserved		Reserved
13	8	Reserved		Reserved
14	8	Reserved		Reserved
15	8	CRC		CRC8 Checksum

3.1.1.2 Link Speed Frame

The Link Speed Frame is used to select the desired operating speed. The Speed Select is defined in Table 23 below. Bitfields in the Speed capabilities WORD are referring to Speed Capabilities used in Link Detect Frame. Selected operational speed is indicated by a bit is set to 1 in one of the Speed Select fields.

Table 23 Speed Select Encoding

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	Base Freq. x12 300MHz	Base Freq. x10 250MHz	Base Freq. x8 200MHz	Base Freq. x6 150MHz	Base Freq. x4 100MHz	Base Freq. x3 75MHz	Base Freq. x2 50MHz	Base Freq. x1 25 MHz
1	Dual Data Rate 0: SDR 1: DDR	Reserved	Reserved	Reserved	Base Freq. x40 1GHz	Base Freq. x32 800MHz	Base Freq. x24 600MHz	Base Freq. x16 400MHz

The Link Speed Frame is defined in the Table 24 below.

Table 24 Link Speed Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 20
1	8	Frame Subtype								Frame Subtype as defined in Table 20
2	8	LTPI Version Major				LTPI Version Minor				BCD Encoded LTPI Version [LTPI Version Major].[LTPI Version Minor]
3	8	Speed Select								Speed Select as defined in Table 23

4	8		
5	8	Reserved	Reserved
6	8	Reserved	Reserved
7	8	Reserved	Reserved
8	8	Reserved	Reserved
9	8	Reserved	Reserved
10	8	Reserved	Reserved
11	8	Reserved	Reserved
12	8	Reserved	Reserved
13	8	Reserved	Reserved
14	8	Reserved	Reserved
15	8	CRC	CRC8 Checksum

3.1.2 Advertise, Configure and Accept Frames

The Advertise and Configure and Accept Frames are used to exchange the supported capabilities of LTPI (Channels and features supported) and configure which capabilities supported by DC-SCM and HPM LTPI endpoints should be enabled. Refer to the 4 LTPI Link Initialization and Operation for more details. The Frame Subtype field is used to identify Advertise, Configure and Accept Fames as defined in the Table 25 below.

Table 25 Advertise, Configure and Accept Frame Type and Subtypes

Frame Type	Comma Symbol	Frame Subtype Field Value
Advertise	K28.6	0h00
Configure		0h01
Accept		0h02

3.1.2.1 Advertise Frame

The Advertise Frame is used to manifest which LTPI capabilities are supported by SCM LTPI and HPM LTPI. The Advertise Frame is manifesting the following information:

1. OEM-defined SCM/HPM platform ID in Table 26 below
2. LTPI Capabilities defined in Table 28 below
3. OEM Capabilities

Table 26 Platform Type Encoding

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	SCM/HPM ID (OEM-defined ID of the SCM/HPM)							
1								

Table 27 UART Channel Max Baud Rate Encoding

Baud Rate [bits per second]	Max Baud Rate Encoding
300	0h00
600	0h01
1200	0h02
1800	0h03
2400	0h04
4800	0h05
9600	0h06
19200	0h07
38400	0h08
57600	0h09
115200	0h0A
230400	0h0B
460800	0h0C
576000	0h0D
921600	0h0E
Reserved	0h0F

Table 28 Default LTPI Capabilities Encoding

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	**Supported Channels**							
	Reserved			OEM Channel	Data Channel	UART Channel	I2C/SMBus Channel	GPIO Channel
1	**GPIO Channel Capabilities**							
	Number of NL GPIOs [7:0]							
2	Reserved						Number of NL GPIOs [9:8]	
3	**I2C Channel Capabilities**							
	Reserved	Echo Support 0: Echo not supported 1: Echo supported (Always set to 1 if I2C/SMBus Channel is Supported)	I2C/SMBus5 Enable 0: Disabled 1: Enabled	I2C/SMBus4 Enable 0: Disabled 1: Enabled	I2C/SMBus3 Enable 0: Disabled 1: Enabled	I2C/SMBus2 Enable 0: Disabled 1: Enabled	I2C/SMBus1 Enable 0: Disabled 1: Enabled	I2C/SMBus0 Enable 0: Disabled 1: Enabled
4	Reserved		I2C/SMBus5 Speed 0: 100kHz 1: 400kHz	I2C/SMBus4 Speed 0: 100kHz 1: 400kHz	I2C/SMBus3 Speed 0: 100kHz 1: 400kHz	I2C/SMBus2 Speed 0: 100kHz 1: 400kHz	I2C/SMBus1 Speed 0: 100kHz 1: 400kHz	I2C/SMBus0 Speed 0: 100kHz 1: 400kHz
5	**UART Channel Capabilities**							

	Reserved	UART1 Enable 0: Disabled 1: Enabled	UART0 Enable 0: Disabled 1: Enabled	Flow Control	Max Baud Rate as defined in Table 27
6	**OEM Capabilities**				
7					

The Advertise Frame is defined in the Table 29 below.

Table 29 LTPI Advertise Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 25
1	8	Frame Subtype								Frame Subtype as defined in Table 25
2	8	Platform Type								Platform Type as defined in Table 26
3	8									
4	8	Capabilities Type = 0h00 – Default Capabilities as defined in Table 28 = 0h01:0h80 – Reserved = 0h81:0fFF – OEM Defined								Capabilities Type allows for use of non-default capabilities to customize the LTPI implementations.
5	8	LTPI Capabilities								LTPI Capabilities according to Capabilities Type. Default Capabilities are defined in Table 28
6	8									
7	8									
8	8									
9	8									
10	8									
11	8									
12	8									
13	8	Reserved								Reserved
14	8									
15	8	CRC								CRC8 Checksum

3.1.2.2 Configure Frame

The Configure Frame is used to select the desired capabilities based on Advertise Frame. It is used by SCM to Request specific Configuration of LTPI Features based on the Advertise Definition. The LTPI Configuration Frame is defined in the Table 30 below.

Table 30 LTPI Configure Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 25
1	8	Frame Subtype								Frame Subtype as defined in Table 25

2	8	Capabilities Type = 0h00 – Default Capabilities as defined in Table 28 = 0h01:0h80 – Reserved = 0h81:0fFF – OEM Defined	Capabilities Type allows for use of non-default capabilities to customize the LTPI implementations.
3	8	Request LTPI Capabilities	LTPI Capabilities as defined in Table 28
4	8		
5	8		
6	8		
7	8		
8	8		
9	8		
10	8		
11	8	Reserved	Reserved
12	8	Reserved	Reserved
13	8	Reserved	Reserved
14	8	Reserved	Reserved
15	8	CRC	CRC8 Checksum

3.1.2.3 Accept Frame

The Accept Frame is used to confirm Requested LTPI Configuration. It is used by HPM to Accept specific Configuration of LTPI Features that SCM Requested in Configure Frame. The LTPI Accept Frame is defined in the Table 30 below.

Table 31 LTPI Accept Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 25
1	8	Frame Subtype								Frame Subtype as defined in Table 25
2	8	Capabilities Type = 0h00 – Default Capabilities as defined in Table 28 = 0h01:0h80 – Reserved = 0h81:0fFF – OEM Defined								Capabilities Type allows for use of non-default capabilities to customize the LTPI implementations.
3	8	Accept LTPI Capabilities								LTPI Capabilities as defined in Table 28
4	8									
5	8									
6	8									
7	8									
8	8									
9	8									
10	8									

11	8	Reserved	Reserved
12	8	Reserved	Reserved
13	8	Reserved	Reserved
14	8	Reserved	Reserved
15	8	CRC	CRC8 Checksum

3.1.1 LTPI Operational Frames

The Operational Frames are used in normal operation of the link after Link Training and Configuration Phase is completed. Refer to the 4 LTPI Link Initialization and Operation for more details. The Frame Subtype field is used to identify different Frame Types as defined in the Table 32 below. The Default I/O and Data Frames are defined but particular platform design can define OEM/Custom Frame Types to match specific design requirements. A combination of OEM Capabilities defined in Table 28 and OEM Frame Subtypes can be used to define custom Operational Frames.

Table 32 LTPI Operational Frame Type and Subtypes

Frame Type	Comma Symbol	Frame Subtype Field Value
Default I/O	K28.7	0h00
Default Data		0h01
Reserved		0h02-0h0F
OEM Defined		0h10-0hFF

3.1.1.1 Default I/O Frame

The Default I/O Frame is used in LTPI Operational mode to tunnel LTPI Channels. If certain Channels are not enabled in Capabilities or Configure phase those channels are ignored by RX side. The Default I/O Frame is defined in the Table 33 below.

Table 33 LTPI Default I/O Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 32
1	8	Frame Subtype								Frame Subtype as defined in Table 32
2	8	Frame Counter								Frame Counter used to decode NL GPIOs (see 2.2.1.1 GPIO Interfaces)
3	8	Low Latency GPIO 0								Low Latency GPIOs as defined in Table 6
4	8	Low Latency GPIO 1								Low Latency GPIOs as defined in Table 6
5	8	Normal Latency GPIO 0								Normal Latency GPIOs as defined in Table 6
6	8	Normal Latency GPIO 1								Normal Latency GPIOs as defined in Table 6
7	8	UART0 and UART1								UART0/1 as defined in Table 8

8	8	I2C/SMBus0 and I2C/SMBus1	I2C/SMBus0/1 as defined in Table 11
9	8	I2C/SMBus2 and I2C/SMBus3	I2C/SMBus2/3 as defined in Table 11
10	8	I2C/SMBus4 and I2C/SMBus5	I2C/SMBus4/5 as defined in Table 11
11	8	OEM Reserved	OEM Reserved
12	8		
13	8		
14	8		
15	8	CRC	CRC8 Checksum

3.1.1.2 Default Data Frame

The Default Data Frame is in LTPI Operational mode to tunnel LTPI Data Channel. It is considered a Random-Access Frame and it is only sent on demand when Data Read/Write Access is triggered e.g., asynchronously by BMC Request. The Data Frame interleaves a stream of I/O Frames and also supports Low Latency GPIOs tunneling not to impact the most critical low latency signals. The default Data Frame is defined in the Table 34 below.

The Tag field in the Data Frame is intended to be used by the BMC or any other logic within the CPLD that is making use of the LTPI Data Channel. The Tag field however needs to be maintained by the LTPI Data Channel implementation to meet the following requirements:

1. LTPI Data Channel keeps track of Tag values assigned to outstanding LTPI Data Channel requests linked with Request type and address
2. LTPI Data Channel handles cases when multiple requests with the Tag are received while operation is outstanding on the Data Chanel or is currently being completed. The repeated requests with the same Tag might be a result of BMC timeout while waiting for response. In this case repeated requests shall be dropped until initial request is completed or LTPI Data Channel Error shall be returned to BMC or another initiator of the request.
3. The Tag field shall not be interpreted by LTPI Data Channel as a counter or order of operations. LTPI Data Channel on both sides of LTPI Link shall send and execute the Data Channel operations in order of received Data Channel access requests.
4. The Tag shall be retired if no response was received

Open: Shall the detailed Tag handling above be defined withing the scope of LTPI Specification or it should be part of LTPI Design-level specification?

Table 34 LTPI Default Data Frame

Frame Offset [bytes]	Size [bits]	Bit Field								Description
		7	6	5	4	3	2	1	0	
0	8	Comma Symbol								Comma Symbol as defined in Table 32
1	8	Frame Subtype								Frame Subtype as defined in Table 32
2	8	Low Latency GPIO 0								Low Latency GPIOs as defined in Table 6

3	8	Low Latency GPIO 1	Low Latency GPIOs as defined in Table 6
4	8	Tag	Tag field which can be used by the user of the Data Channel (e.g., other CPLD/FPGA Logic, BMC FW, etc.) to identify responses if multiple outstanding Data Request were sent to LTPI.
5	8	Data Channel Payload	Data Channel Payload as defined in the Table 13
6	8		
7	8		
8	8		
9	8		
10	8		
11	8		
12	8		
13	8		
14	8		
15	8	CRC	CRC8 Checksum

3.2 LTPI Control and Status Registers (CSR)

The LTPI interface provides a Configuration Space Registers. The configuration registers are primarily defined for the BMC to have control over LTPI operation. The basic usage of LTPI CSR is to allow BMC to access:

1. LTPI Link Status information
2. LTPI Link Capabilities Information
3. LTPI Link Configuration Control
4. LTPI Channel Operational Telemetry Information
5. LTPI Link/Channel Reset Capabilities

For the flexibility of BMC and CPLD integration it is out of scope of LTPI to define what interface is used by BMC to access LTPI CSR space. LTPI CSR only defines the default register space and OEM extensions placeholders. A specific platform design might use any interface with appropriate mapping in CPLD to expose the LTPI CSR to BMC such as I2C, I3C, SPI, UART etc. If BMC needs to access CSR on the HPM Side a Data Channel Can be utilized for this purpose. An example of BMC access to CSR through I2C interface is presented on Figure 27 below.

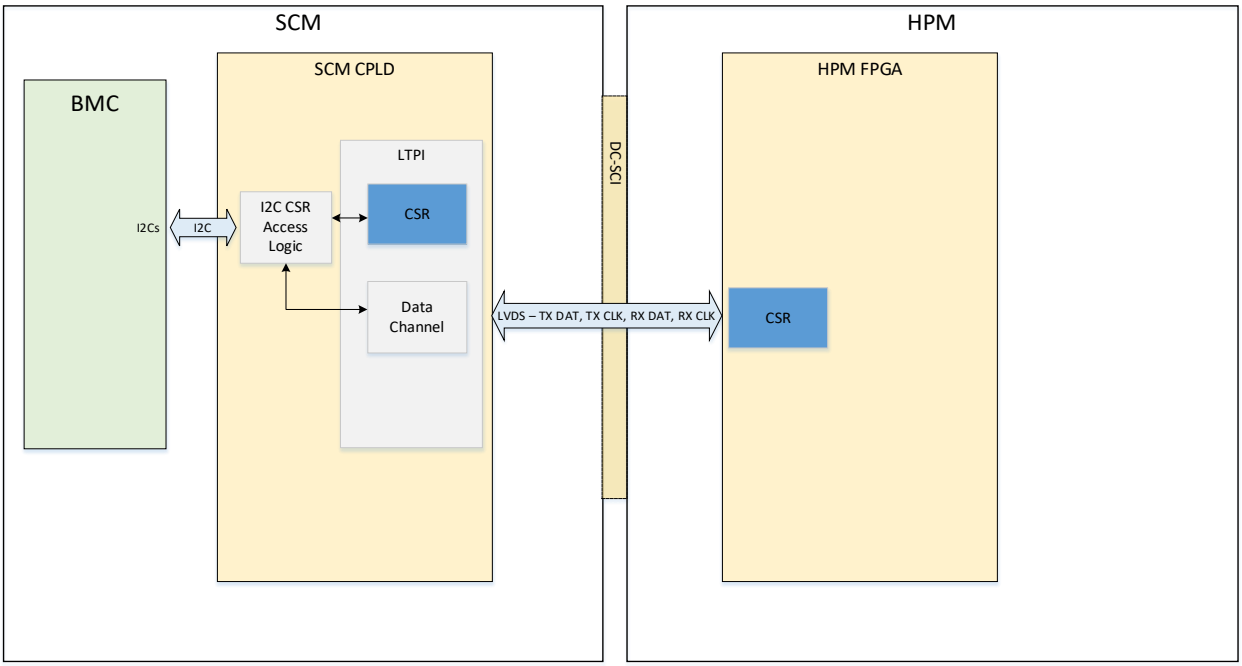

Figure 27 Example of LTPI CSR access from BMC

The general register map with predefined offsets is defined in the Table 35 below. The LTPI specification defines a placeholder for OEM-defined information related to FPGA/CPLD Device, Image build number and tracking data, etc. The general CSR Map also defines only a placeholder for Memory Mapped Channel definition with assumption that if Data Channel is used the content will be OEM-specific.

Table 35: General Control and Status Register Map

Offset	Name	Description
0x0000 – 0x01FF	CPLD/FPGA Information	The offset is reserved for OEM-defined CPLD/FPGA information such as CPLD/FPGA image version numbers, device identifiers, OEM specific information, etc.
0x0200 – 0x02FF	LTPI Control and Status	LTPI Status and Link Training Control Registers defined in the []
0x0300 – 0x03FF	Reserved	Reserved for future use
0x0400 – 0xFFFFFFFF	LTPI Memory Mapped Channel	Address Space reserved for memory mapped channel.

The LTPI Control and Status Registers are defined in the Table 36 below. The Register types are defined as follows:

RO – *Read Only Register: Write to this register do not change the register value.*
RW – *Read/Write Register: Write to this register will change the register value.*
RWC – *Read/Write-Clear Register: Write all '1' to clear register value to 0.*

Table 36: LTPI Control and Status Registers

<table>
<tr><th>Offset</th><th>Name</th><th>Bitfield</th><th>Type</th><th>Bit Field Definition</th></tr>
<tr><td rowspan="20">0x00</td><td rowspan="20">LTPI Link Status</td><td>19:16</td><td>RO</td><td>Local LTPI Link State (SCM)
<table><tr><th>State</th><th>Encoding</th></tr><tr><td>Link Detect</td><td>0x0</td></tr><tr><td>Link Speed</td><td>0x1</td></tr><tr><td>Advertise</td><td>0x2</td></tr><tr><td>Configuration</td><td>0x3</td></tr><tr><td>Operational</td><td>0x4</td></tr><tr><td>Reserved</td><td>0x5-0xF</td></tr></table></td></tr>
<tr><td>15:12</td><td>RO</td><td>Remote LTPI Link State (HPM)
<table><tr><th>State</th><th>Encoding</th></tr><tr><td>Link Detect</td><td>0x0</td></tr><tr><td>Link Speed</td><td>0x1</td></tr><tr><td>Advertise</td><td>0x2</td></tr><tr><td>Accept</td><td>0x3</td></tr><tr><td>Operational</td><td>0x4</td></tr><tr><td>Reserved</td><td>0x5-0xF</td></tr></table></td></tr>
<tr><td>11:8</td><td>RO</td><td>LTPI Link Speed
<table><tr><th>Link Speed</th><th>Encoding</th></tr><tr><td>Base Freq x1</td><td>0x0</td></tr><tr><td>Base Freq x2</td><td>0x1</td></tr><tr><td>Base Freq x3</td><td>0x2</td></tr><tr><td>Base Freq x4</td><td>0x3</td></tr><tr><td>Base Freq x6</td><td>0x4</td></tr><tr><td>Base Freq x8</td><td>0x5</td></tr><tr><td>Base Freq x10</td><td>0x6</td></tr><tr><td>Base Freq x12</td><td>0x7</td></tr><tr><td>Base Freq x16</td><td>0x8</td></tr><tr><td>Base Freq x24</td><td>0x9</td></tr><tr><td>Base Freq x32</td><td>0xA</td></tr><tr><td>Base Freq x40</td><td>0xB</td></tr><tr><td>Reserved</td><td>0xC-0xF</td></tr></table></td></tr>
<tr><td>7</td><td>RO</td><td>DDR Mode:
0: SDR Mode
1: DDR Mode</td></tr>
<tr><td>6</td><td>-</td><td>Reserved</td></tr>
<tr><td>5</td><td>RWC</td><td>Link Configure/Accept Timeout Error</td></tr>
<tr><td>4</td><td>RWC</td><td>Link Speed Timeout Error</td></tr>
<tr><td>3</td><td>RWC</td><td>Unknown Comma Symbol Error</td></tr>
<tr><td>2</td><td>RWC</td><td>Frame CRC Error</td></tr>
<tr><td>1</td><td>RWC</td><td>LTPI Link Lost Error</td></tr>
<tr><td>0</td><td>RO</td><td>LTPI Link Aligned
0: Link not Aligned
1: Link Aligned</td></tr>
<tr><td rowspan="2">0x04</td><td rowspan="2">LTPI Detect Capabilities Local</td><td>31:24</td><td>-</td><td>Reserved</td></tr>
<tr><td>23:8</td><td>RW</td><td>Link Speed Capabilities
Speed capabilities as in Table 21</td></tr>
</table>

				(BMC can use the capabilities register to override Link Speed capabilities)
		7:4	RO	**Local LTPI Major Version** Major version BCD encoded as in Table 22
		3:0	RO	**Local LTPI Minor Version** Minor version BCD encoded as in Table 22
0x08	LTPI Detect Capabilities Remote	31:24	-	**Reserved**
		23:8	RO	**Link Speed Capabilities** Speed capabilities as in Table 21
		7:4	RO	**Remote LTPI Major Version** Major version BCD encoded as in Table 22
		3:0	RO	**Remote LTPI Minor Version** Minor version BCD encoded as in Table 22
0x0C	LTPI Platform ID Local	31:16	-	**Reserved**
		15:0	RO	**Local LTPI Platform ID** Platform ID supported by Local LTPI as defined in Table 26
0x10	LTPI Platform ID Remote	31:16	-	**Reserved**
		15:0	RO	**Remote LTPI Platform ID** Platform ID as advertised by Remote LTPI as defined in Table 26
0x14	LTPI Advertise Capabilities Local Low	31:0	RW	**Local LTPI Capabilities Low – Bytes 3:0** LTPI Capabilities as defined in Table 28
0x18	LTPI Advertise Capabilities Local High	31:0	RW	**Local LTPI Capabilities High – Bytes 7:4** LTPI Capabilities as defined in Table 28
0x1C	LTPI Advertise Capabilities Remote Low	31:0	RO	**Remote LTPI Capabilities Low – Bytes 3:0** LTPI Capabilities as defined in Table 28
0x20	LTPI Advertise Capabilities Remote High	31:0	RO	**Remote LTPI Capabilities High – Bytes 7:4** LTPI Capabilities as defined in Table 28
0x24	LTPI Default Configuration Low	31:0	RO	**LTPI Default Configuration Low – Bytes 3:0** LTPI Capabilities as defined in Table 28
0x28	LTPI Default Configuration High	31:0	RO	**LTPI Default Configuration Low – Bytes 7:4** LTPI Capabilities as defined in Table 28
0x2C	LTPI Link Alignment Error Counter	31:0	RWC	**LTPI Link Alignment Error Counter**
0x30	LTPI Link Lost Error Counter	31:0	RWC	**LTPI Link Lost Error Counter**
0x34	LTPI CRC Error Counter	31:0	RWC	**LTPI CRC Error Counter**
0x38	Unknown Comma Error Counter	31:0	RWC	**LTPI Unknown Comma Error Counter**
0x3C	Link Speed Timeout Error Counter	31:0	RWC	**LTPI Link Speed Timeout Error Counter**
0x40	Link Configure/Accept Timeout Error Counter	31:0	RWC	**LTPI Link Configure/Accept Timeout Error Counter**
0x44	Link Training RX Frames Counter Low	31:24	RWC	**Link Configure/Accept Frames Received Counter**
		23:16	RWC	**Link Speed Frames Received Counter**
		15:0	RWC	**Link Detect Frames Received Counter**
0x48	Link Training RX Frames Counter High	31:0	RWC	**Link Advertise Frames Received Counter**
0x4C		31:24	RWC	**Link Configure/Accept Frames Sent Counter**

	Link Training TX Frames Counter Low	23:16	RWC	**Link Speed Frames Sent Counter**
		15:0	RWC	**Link Detect Frames Sent Counter**
0x50	Link Training TX Frames Counter High	31:0	RWC	**Link Advertise Frames Sent Counter**
0x54	Operational RX Frames Counter	31:0	RWC	**Operational Frames Received**
0x58	Operational TX Frames Counter	31:0	RWC	**Operational Frames Transmitted**
0x5C – 0x7F	Reserved	-	-	**-**
0x80	LTPI Link Control	31:12	-	**Reserved**
		11	RW	**Trigger Configuration State**
		10	RW	**Automatically Move to Configuration State**
		9	RW	**Data Channel Reset**
		8:2	RW	**LTPI I2C Channel Reset** Write I2C Channel Link Number
		1	RW	**LTPI Link Retraining Request**
		0	RW	**LTPI Link Software Reset**
0x84 – 0xFF	Reserved	-	-	**-**

4 LTPI Link Initialization and Operation

The LTPI Training and Link initialization needs to be executed in the following scenarios:

- After SCM CPLD and HPM FPGA power lost events
- After SCM CPLD and HPM FPGA reset de-assertion
- On request from external entities such as BMC
- After Link Lost Error occurred

The main purpose of LTPI Link Training and initialization is to:

- Achieve DC balance condition on the link
- Exchange link speed capabilities
- Switch to operational link frequency
- Exchange LTPI features capabilities
- Enable the selected operational configuration of the LTPI
- Switch to Operational Mode

The LTPI Link Training and switch to Operational mode is defined as a State Machine that can be divided into 3 major phases:

1. Link Training – Initial state of Link Initialization
2. Link Configuration – Configuration of LTPI features
    a. Exchange LTPI features capabilities

    b. Enable the selected operational configuration of the LTPI
    c. Switch to Operational Mode
3. Link Operational – State when link is for the most of the LTPI lifecycle

The state machine is presented in the *Figure 28* below.

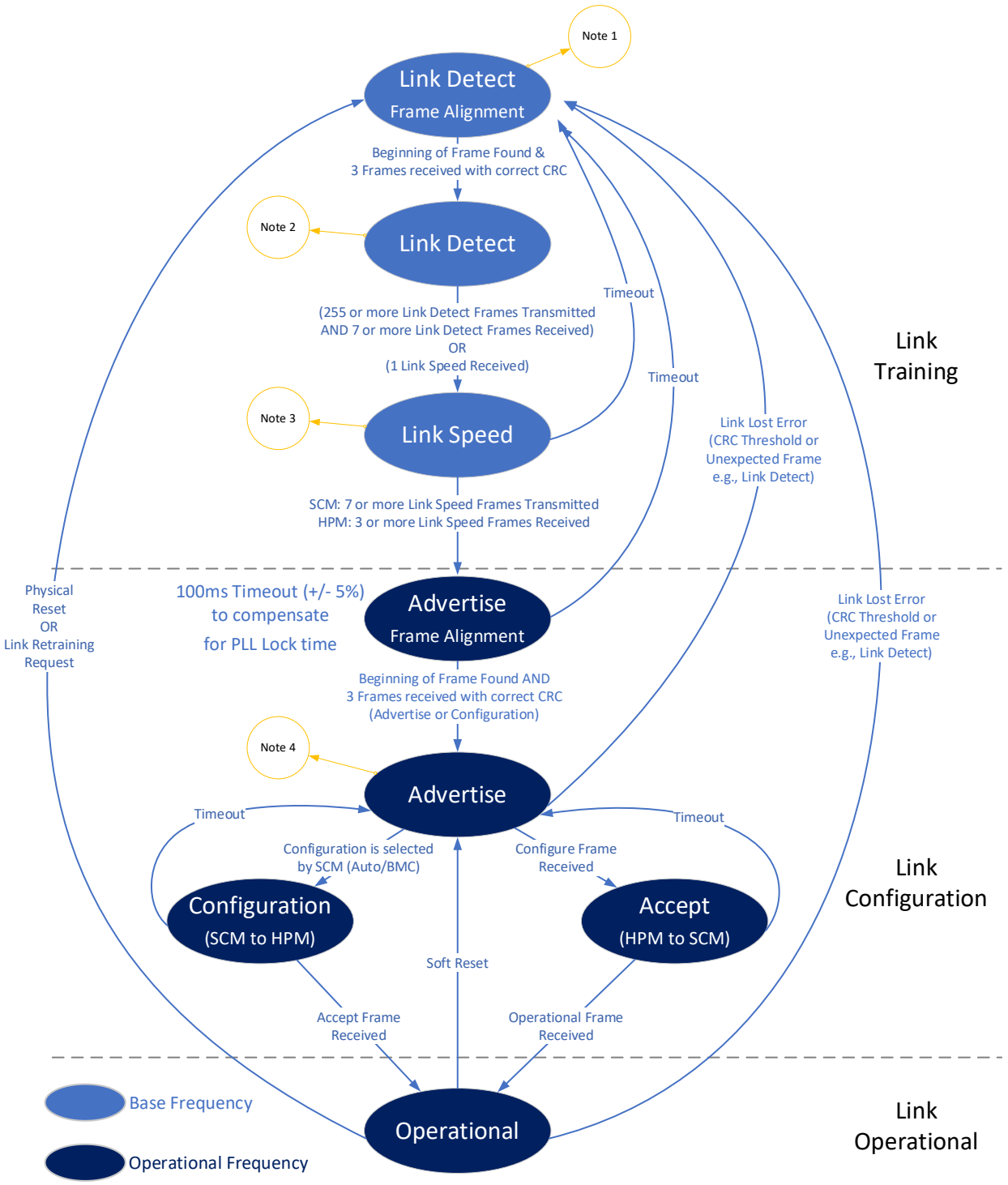

Figure 28 Link Training and Initialization Flow

Note 1: *The time it takes to align to the beginning of the frame depends on time needed to achieve the DC-balance and how efficiently the comma symbol can be found (depends on SERDES design). This stage*

is the source of initial misalignment between SCM LTPI and HPM LTPI by means of number of Frames Sent and Received. Number of Frames sent shall not be included in the minimum Frame Detect frames transmitted (255).

Note 2: *This stage is expected to be entered by SCM and HPM at different timings due to initial misaligned in the first stage. This means that one side can complete this stage earlier and move to the Link Speed stage. If it happens then the 'slower' part shall switch to the Link Speed even if the required number of TX and RX Frames is not achieved.*

Note 3: *Both sides keep sending Link Speed. SCM is required to send minimum 7 Link Speed Frames while HPM is required to receive 3 Link Speed Frames.*

Note 4: *In advertise since the LTPI clock is reconfigured to higher speed the LTPI needs to re-align to the beginning of the frames again similarly to the Link Detect stage. Typically, PLL is going to be re-configured and requires some time to lock. To compensate for that 1ms timeout is used on both sides.*

4.1.1 Link Training

The Link Training phase starts after SCM CPLD and HPM FPGA comes out of reset. It is executed on Base Frequency of 25MHz in Single Data Rate mode. The major purpose of Link Training phase is to:

a. Achieve DC balance condition on the link
b. Exchange link speed capabilities
c. Switch to operational link frequency

The Link Training Phase is based on 2 States:

1. Link Detect
2. Link Speed

4.1.1.1 Link Detect

In the Link Detect state both SCM CPLD and HPM FPGA start sending the Link Detect Frame defined in Table 22. As presented in Figure 29 frames are being transmitted in both directions. The main purpose is to establish link DC balance and manifest supported Operational Frequency. This stage can be divided into 2 parts (as presented in *Figure 28*):

1. Link Detect – Frame Alignment: In this part the SERDES logic is trying to lock to the beginning of the Frame by identifying the Comma Symbol and processing the Frame to verify the CRC. In order to move to the next part at least 3 Frames need to be received with correct CRC.
2. Link Detect: This is the main part of Link Detect where Frames are being used to interpret the Operational Speed capabilities of the other side. In this part TX and RX counters are being used. Due to the timing differences between SCM and HPM in achieving DC-balance finding beginning of the frame it is expected that SCM and HPM TX/RX counters will be misaligned.

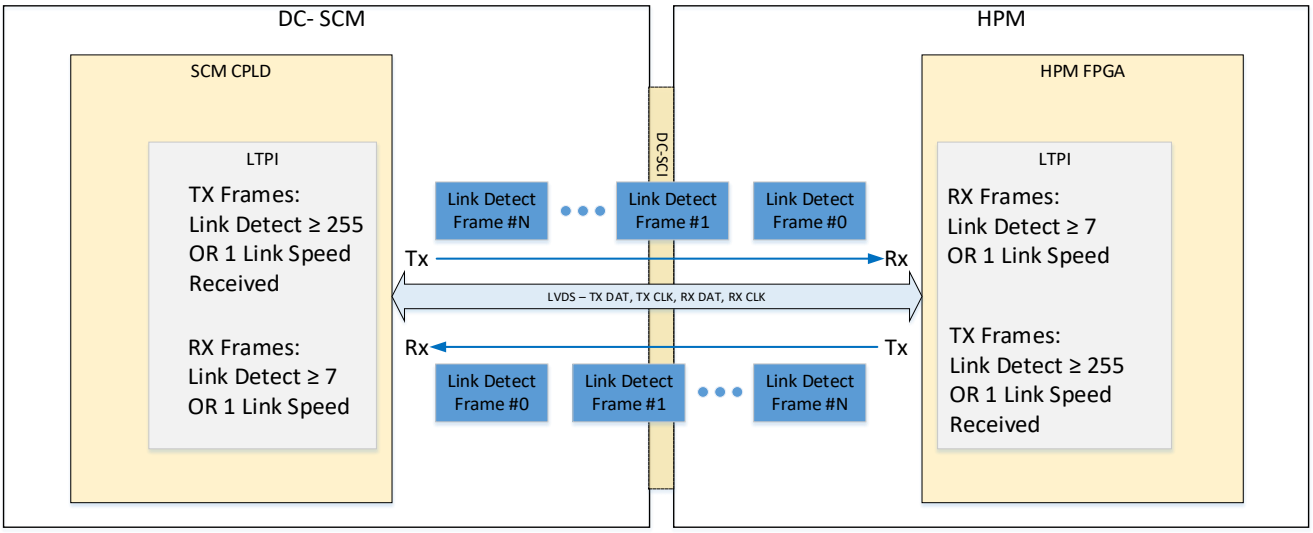

Figure 29 LTPI Link Detect State Frames

Both SCM CPLD and HPM FPGA shall:

- Send at least 255 Link Detect Frames on TX Link
- Receive at least 7 consecutive frames with correct CRC checksum on RX Link

If one side moves to the next stage faster and starts sending Link Speed Frame the 'slower' side shall move to next state immediately completing required number of TX and RX Frames.

SCM LTPI and HPM LTPI can move to the next state independently provided both conditions are met for given LTPI i.e., it's not required to wait for more than 7 consecutive correct RX frames.

Link Detect State summary is provided in the **Error! Reference source not found.** below.

Table 37 Link Detect State Overview

Parameter	SCM	HPM
State Entry Condition	Physical Reset Link Lost Error Link Speed Timeout Advertise Timeout	Physical Reset Link Lost Error Link Speed Timeout Advertise Timeout
Link Frequency	Base Frequency	Base Frequency
LTPI Frame Transmitted (TX)	Link Detect as defined in Table 22	Link Detect as defined in Table 22
Minimum Number of Frames Transmitted (TX)	255	255
LTPI Frame Received (RX)	Link Detect or Link Speed	Link Detect or Link Speed
Minimum Number of Correct Frames Received (RX)	Link Detect: 7 Link Speed: 1	Link Detect: 7 Link Speed: 1
Next State	Link Speed	Link Speed
Timeout	N/A	N/A
Next State after timeout	Link Detect	Link Detect

4.1.1.2 Link Speed

In the Link Speed state both SCM CPLD and HPM FPGA start sending the Link Speed Frame defined in Table 24. As presented in Figure 30 frames are being transmitted in both directions. SCM shall pick up the highest common target frequency based on the Link Detect capabilities received and its own capabilities. In some cases, like system integration, system debugging or recovery mode it might be needed to lower the Operational Frequency below maximum capabilities. In Order to do that LTPI can expose a setting to BMC or other controller on the SCM to allow modification of the SCM LTPI Link Detect capabilities.

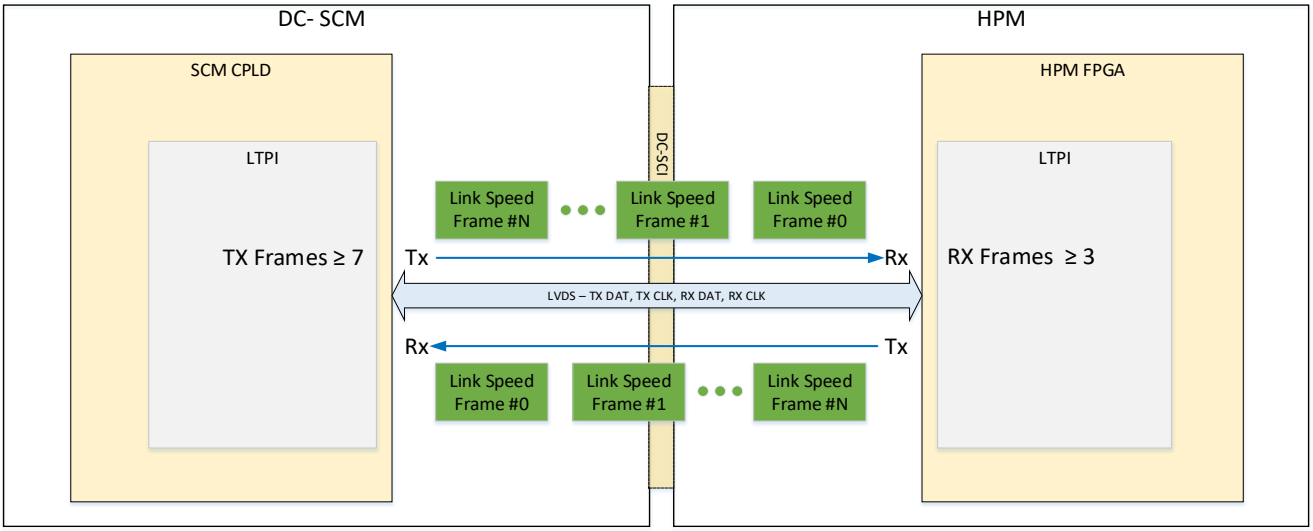

Figure 30 LTPI Link Speed State Frames

The following conditions shall be met in order to proceed to next stage:

- SCM: Send at least 7 Link Speed Frames on TX Link
- HPM: Receive at least 3 Link Speed Frames

Link Speed State summary is provided in the Table 38 below.

Table 38 Link Speed State Overview

Parameter	SCM	HPM
State Entry Condition	Link Detect Completion	Link Detect Completion
Link Frequency	Base Frequency	Base Frequency
LTPI Frame Transmitted (TX)	Link Speed as defined in Table 24.	Link Speed as defined in Table 24
Minimum Number of Frames Transmitted (TX)	7	N/A

LTPI Frame Received (RX)	Link Speed	Link Speed
Minimum Number of Correct Frames Received (RX)	N/A	3
Next State	Advertise	Advertise
Timeout	255 Frames Transmitted	255 Frames Transmitted
Next State after timeout	Link Detect	Link Detect

4.1.1.3 Link Training Example

After the reset exit SCM and HPM start sending the Detect Frames. When the DC balance is established on the links:

1. SCM Received the Speed Capabilities from HPM as defined in the Table 39 below. Following Speed Capabilities defined in Table 21 the HPM supports 25MHz, 100 MHz and 250MHz. It also supports Dual Data Rate I/O.

Table 39 Speed Capabilities sent by HPM

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	0	1
1	1	0	0	0	0	0	0	0

2. HPM Received the Speed Capabilities from SCM as defined in the Table 40 below. Following Speed Capabilities defined in Table 21 the SCM supports 25MHz, 50MHz, 75MHz and 100MHz. It also supports Dual Data Rate I/O.

Table 40 Speed Capabilities sent by SCM

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0

3. HPM and SCM send highest common speed setting using Link Speed Frames. The Speed Select setting sent is as listed in the Table 41 below. The Operational Frequency selected is set to 100MHz and DDR enabled.

Table 41 Speed Select sent by SCM and HPM

Byte	Bit Fields Encoding							
	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0	0

4.1.2 Link Configuration

The Link Configuration starts after Link Training phase is completed. It is executed on Operational Frequency selected during Link Speed stage. The major purpose of Link Configuration phase is to:

a. Switch to operational link frequency
b. Exchange LTPI features capabilities
c. Enable the selected operational configuration of the LTPI
d. Switch to Operational Mode

The Link Training Phase is based on 3 States:

1. Advertise
2. Configure
3. Accept

4.1.2.1 Advertise

The Advertise state is the first state when link switches from Base Frequency to the Operational Frequency. In the Advertise state both SCM CPLD and HPM FPGA start sending the Advertise Frame defined in Table 25. As presented in Figure 31, frames are being transmitted in both directions. After switching to Operational Frequency, the RX SERDES needs to re-adjust to the beginning of the Frame similarly as in Link Detect stage. This stage can also be divided into 2 parts (as presented in *Figure 28*):

1. Advertise – Frame Alignment: In this part the SERDES logic is trying to lock to the beginning of the Frame by identifying the Comma Symbol and processing the Frame to verify the CRC. In order to move to the next part at least 3 Frames need to be received with correct CRC.
2. Advertise: This is the main part of Advertise where Frames are being used to interpret the LTPI capabilities of the other side.

Both SCM and HPM shall keep sending Advertise Frames for at least 1ms to allow the link to stabilize at the Operational Frequency. The Adverstise—Frame Alignment timeout is 100ms.If SCM completed Frame Alignment phase (already receives correct HPM Advertise frames) and also completed 1ms transmission requirement first it will automatically start sending Configure Frame; in this case HPM shall immediately switch to Accept without completing the 1ms transmission requirement. In Advertise state SCM and HPM keep sending advertise Frames. This state does not have a timeout for proceeding to the next state. This is to allow BMC to read the Advertise information and select the LTPI configuration. BMC can use the CSR registers to verify that LTPI Advertise state is active, read the advertise information and write the target configuration of the LTPI.

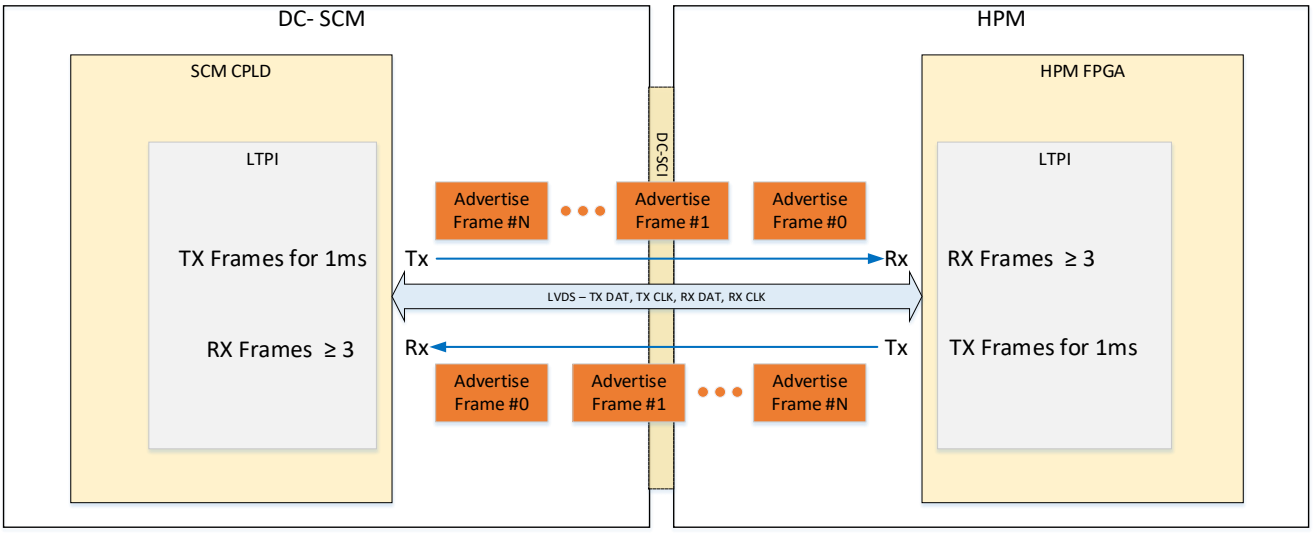

Figure 31 LTPI Advertise State Frames

The next states are different for SCM and HPM. SCM enters Configure state and HPM Accept state. To move to the next state SCM needs to start sending the Configure Frame:

1. Automatically based on SCM LTPI configuration after the Advertise state is entered correctly
2. On request from BMC following configuration applied by BMC

If 3 consecutive Advertise frames are lost in Advertise state, then the LTPI shall return to Link Detect state. Advertise state summary is provided in the Table 42 below.

Table 42 Advertise State Overview

Parameter	SCM	HPM
State Entry Condition	Link Speed Completion Accept RX Timeout	Link Speed Completion Operational State Entry Timeout
Link Frequency	Operational Frequency	Operational Frequency
LTPI Frame Transmitted (TX)	Advertise as defined in Table 29	Advertise as defined in Table 29
Minimum Time of Frame Transmission (TX)	1ms	1ms
LTPI Frame Received (RX)	N/A	N/A
Minimum Number of Correct Frames Received (RX)	3	3
Next State	Configure	Accept
Timeout	RX Frames not Aligned after 100ms	RX Frames not Aligned after 100ms
Next State after timeout	Link Detect	Link Detect
Link Lost Condition	3 Consecutive Lost Frames (CRC Error, Unexpected Frame Type, Frame Alignment Lost)	3 Consecutive Lost Frames (CRC Error, Unexpected Frame Type, Frame Alignment Lost)
Next state after Link Lost	Link Detect	Link Detect

4.1.2.2 Configure & Accept

In this state there's a differentiation between SCM and HPM States:

1. SCM enters Configure State
2. HPM enters Accept sate

As presented in Figure 32 Configure frames are being transmitted by SCM only and Accept Frames are being transmitted by HPM only. In this state SCM starts sending the Configure Frame defined in the Table 30 and waits for HPM to send back Accept Frames as defined in the Table 31.

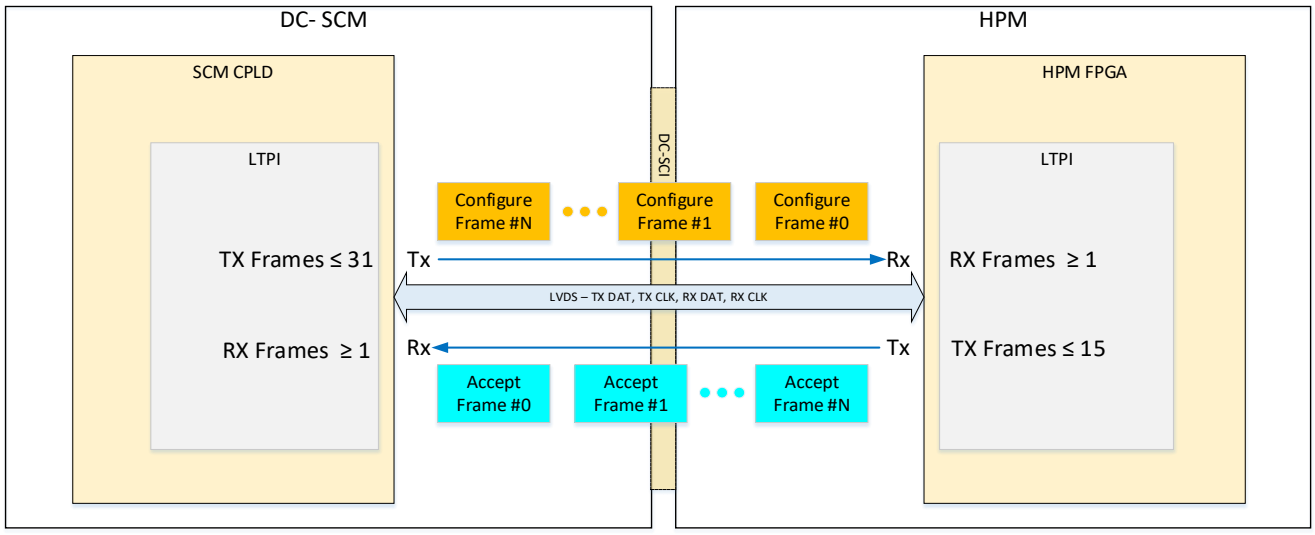

Figure 32 LTPI Advertise State Frames

The Configure state is meant to request the desired Configuration of the LTPI Link from HPM as defined in the Table 28. To move to the next state:

1. SCM starts sending Configure Frames to HPM
2. When HPM received at least 1 Configure Frame and it is matching HPM capabilities it sends the Accept frame back to SCM with the same capabilities as requested by SCM.
3. When Accept Frame is received, and it is matching Configure Frame capabilities SCM moves to next State by sending the first Operation Frame on the link.

If SCM do not receive the matching Accept Frame after sending 32 Configure Frame it moves back Advertise State. Configure and Accept states summary is provided in the Table 42 below.

Table 43 Configure and Accept States Overview

Parameter	SCM	HPM
State Entry Condition	Advertise Completion	Advertise Completion Operational State Entry Timeout
Link Frequency	Operational Frequency	Operational Frequency

LTPI Frame Transmitted (TX)	Configure as defined in Table 30	Link Detect as defined in Table 31
Maximum Number of Frames Transmitted (TX)	31	15
Number of Frames Received (RX)	N/A	N/A
Minimum Number of Correct Frames Received (RX)	1	1
Next State	Operational	Operational
Timeout	31 Frames Transmitted	15 Frames Transmitted
Next State after timeout	Advertise	Advertise
Link Lost Condition	3 Consecutive Lost Frames (CRC Error, Unexpected Frame Type, Frame Alignment Lost)	3 Consecutive Lost Frames (CRC Error, Unexpected Frame Type, Frame Alignment Lost)
Next state after Link Lost	Link Detect	Link Detect

4.1.2.3 Link Configuration Example

After Link Training phase SCM and HPM enter Advertise state:

1. SCM Sends the Advertise Frame with Capabilities presented in Table 44 below. The Capabilities are defined in the Table 28.

Table 44 SCM Capabilities presented in Advertise Frame

Byte	Bit Fields Encoding							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	**Supported Channels**							
	Reserved **= 0**			OEM Channel **= 0**	Data Channel **= 1**	UART Channel **= 1**	I2C/SMBus Channel **= 1**	GPIO Channel **= 1**
1	**GPIO Channel Capabilities**							
	Number of NL GPIOs [7:0] **= 0h64 (100 GPIOs)**							
2	Reserved **= 0**						Number of NL GPIOs [9:8] **= 0h0**	
3	**I2C Channel Capabilities**							
	Reserved **= 0**	Echo Support **=1: Echo supported** (Always set to 1 if I2C/SMBus Channel is Supported)	I2C/SMBus5 Enable **= 1 Enabled**	I2C/SMBus4 Enable **= 1 Enabled**	I2C/SMBus3 Enable **= 1 Enabled**	I2C/SMBus2 Enable **= 1 Enabled**	I2C/SMBus1 Enable **= 1 Enabled**	I2C/SMBus0 Enable **= 1 Enabled**
4	Reserved **= 0**		I2C/SMBus5 Speed **= 1(400kHz)**	I2C/SMBus4 Speed **= 1(400kHz)**	I2C/SMBus3 Speed **= 1(400kHz)**	I2C/SMBus2 Speed **= 1(400kHz)**	I2C/SMBus1 Speed **= 1(400kHz)**	I2C/SMBus0 Speed **= 1(400kHz)**
5	**UART Channel Capabilities**							
	Reserved **= 0**	UART1 Enable **= 1 Enabled**	UART0 Enable **= 1 Enabled**	Flow Control **= 1**	Max Baud Rate as defined in Table 27 **= 0h06 (9600)**			
6	**OEM Capabilities** **= N/A**							
7								

2. HPM Sends the Advertise Frame with Capabilities presented in Table 45 below.

Table 45 HPM Capabilities presented in Advertise Frame

Byte	Bit Fields Encoding							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	**Supported Channels**							
	Reserved **= 0**			OEM Channel **= 0**	Data Channel **= 0**	UART Channel **= 1**	I2C/SMBus Channel **= 0**	GPIO Channel **= 1**
1	**GPIO Channel Capabilities**							
	Number of NL GPIOs [7:0] **= 0h64 (100 GPIOs)**							
2	Reserved **= 0**						Number of NL GPIOs [9:8] **= 0h0**	
3	**I2C Channel Capabilities**							
	Reserved **= 0**	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)
4	Reserved **= 0**		N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)
5	**UART Channel Capabilities**							
	Reserved **=0**	UART1 Enable **= 1 Enabled**	UART0 Enable **= 1 Enabled**	Flow Control **= 1**	Max Baud Rate as defined in Table 27 **= 0h06 (9600)**			
6	**OEM Capabilities** **= N/A**							
7								

3. Based on the capabilities in SCM and HPM the target configuration is selected which needs to be supported by both HPM and SCM. The definition of target Configuration is selected by either SCM automatically selecting highest common configuration or by BMC.
4. SCM sends Configure Frame to HPM with the requested configuration as defined in the Table 46 below.

Table 46 Configuration requested in Configure Frame

Byte	Bit Fields Encoding							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	**Supported Channels**							
	Reserved **= 0**			OEM Channel **= 0**	Data Channel **= 0**	UART Channel **= 1**	I2C/SMBus Channel **= 0**	GPIO Channel **= 1**
1	**GPIO Channel Capabilities**							
	Number of NL GPIOs [7:0] **= 0h64 (100 GPIOs)**							
2	Reserved **= 0**						Number of NL GPIOs [9:8] **= 0h0**	
3	**I2C Channel Capabilities**							

	Reserved **= 0**	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)
4	Reserved **= 0**		N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)	N/A (Channel not supported)
5	**UART Channel Capabilities**							
	Reserved =0	UART1 Enable **= 1 Enabled**	UART0 Enable **= 1 Enabled**	Flow Control **= 1**	Max Baud Rate as defined in Table 27 **= 0h06 (9600)**			
6	**OEM Capabilities = N/A**							
7								

5. HPM waits for at least 1 Configure Frame and when received it verifies that the configuration is valid and matches the HPM capabilities. HPM accepts the configuration by sending back the Accept Frame with the same configuration as requested by SCM as defined in the Table 46 above.

4.1.3 Operational mode

Operational Mode is the main operating condition for the majority of LTPI lifetime. In Operational Mode the SCM and HPM are both constantly sending the I/O Frames in both directions as presented in Figure 33.

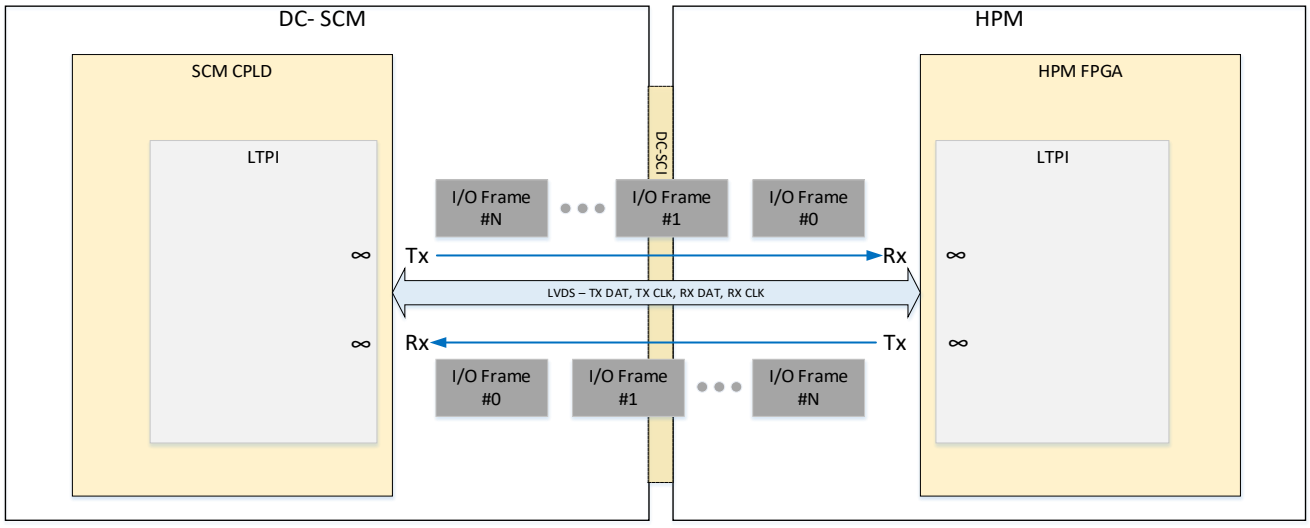

Figure 33 LTPI Operational State with I/O Frames Stream

When Data Channel is used the Data Frames will be generated on demand when access to Data Channel is triggered by:

1. Standalone IP connected in SCM or HPM FPGA directly to Data Channel
2. IP that is exposing access to Data Channel to external entity e.g., BMC

When Data Channel Frame is generated, it will interleave the I/O Frame Stream as presented on the Figure 34 below.

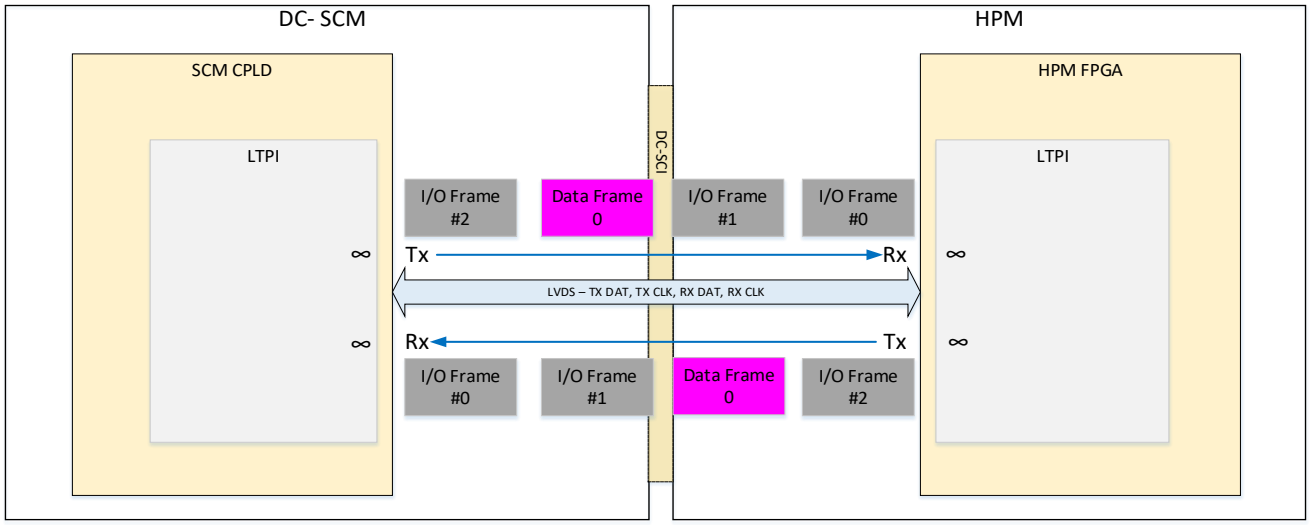

Figure 34 Data Frames interleaving I/O Frames Stream

The SCM and HPM remain in the Operational state until following conditions occur:

1. Soft Reset is triggered: switch to Advertise State
2. Hard Reset is triggered (CPLD Reset, Power Lost): start in Link Detect State
3. Link is lost after 7 consecutive Frames are lost: start in Link Detect State

Operational state summary is provided in the Table 47 below.

Table 47 Operational States Overview

Parameter	SCM	HPM
State Entry Condition	Configure & Accept Completion	Configure & Accept Completion
Link Frequency	Operational Frequency	Operational Frequency
LTPI Frame Transmitted (TX)	Operational Frames as defined in Table 33 and Table 34	Operational Frames as defined in Table 33 and Table 34
Maximum Number of Frames Transmitted (TX)	N/A	N/A
Number of Frames Received (RX)	N/A	N/A
Minimum Number of Correct Frames Received (RX)	1	1
Next State	N/A	N/A

Timeout	N/A	N/A
Next State after timeout	N/A	N/A
Link Lost Condition	7 Lost Frames	7 Lost Frames
Next state after Link Lost	Link Detect	Link Detect

4.1.4 Link Training and Configuration Example

An end-to-end flow from Link Reset to Link Operational example is presented on Figure 35 below.

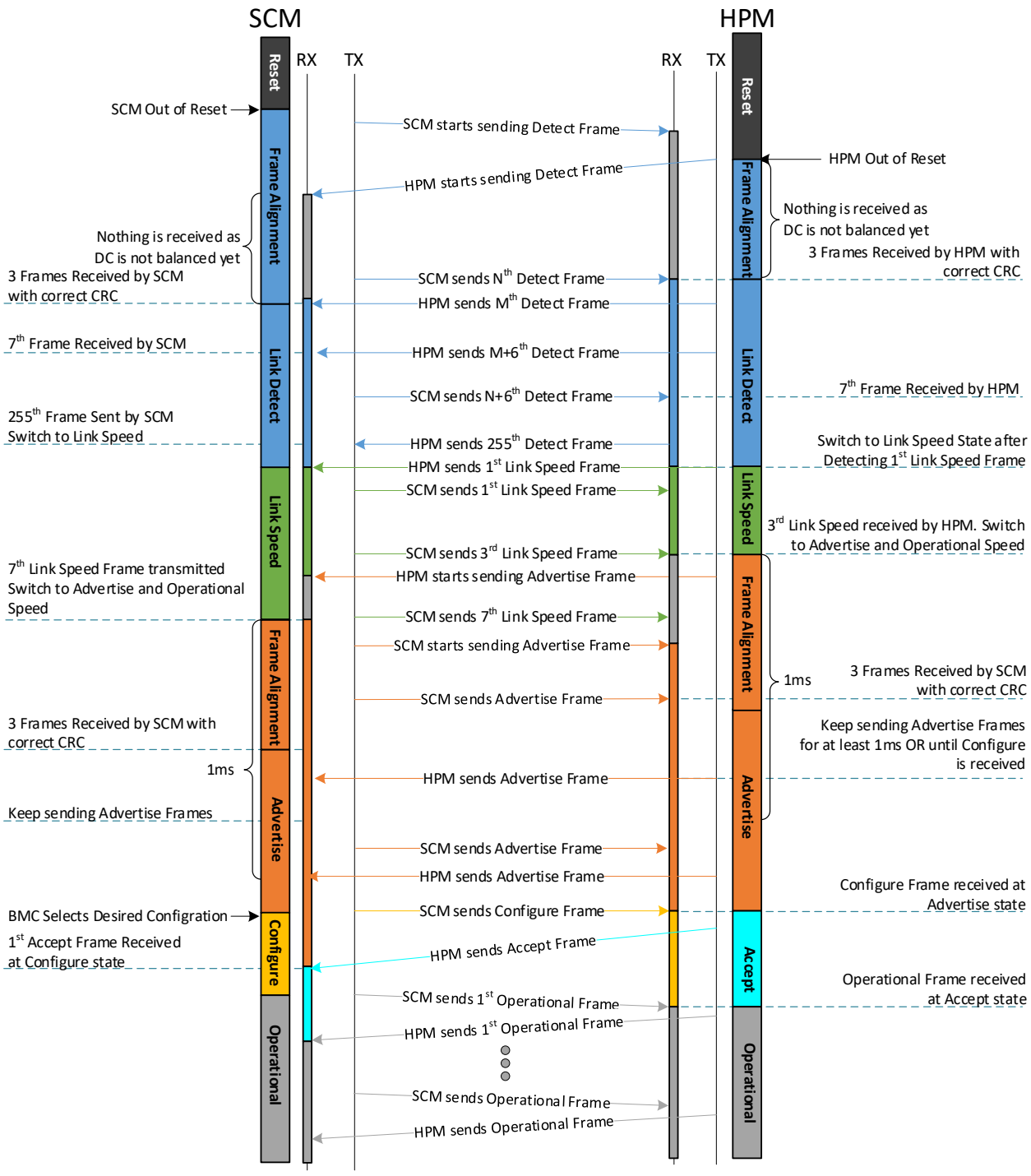

Figure 35 Link Training and Configuration Flow

5 Acronyms

For the purposes of the DC-SCM specification, the following acronyms apply:

Acronym	Definition
ASIC	Application Specific Integrated Circuit
BMC	Baseboard Management Controller
CBB	Compliance Base Board
CFM	Cubic Feet per Minute
DC-SCM	Data Center Secure Control Module
DC-SCI	Data Center Secure Control Interface
DRAM	Dynamic Random Access Memory
EDSFF	Enterprise and Datacenter SSD Form Factor
EMI	Electro Magnetic Interference
ESPI	Enhanced Serial Peripheral Interface
FRU	Field Replaceable Unit
I/O	Input / Output
HFF	Horizontal Form Factor
HPM	Host Processor Module
I2C	Inter-Integrated Circuit - two wire serial protocol
I3C	MIPI Alliance Improved Inter-Integrated Circuit – two wire serial protocol
LED	Light Emitting Diode
LFF	Large Form Factor
LFM	Linear Feet per Minute
LPC	Low Pin Count bus
MAC	Media Access Control
NC	No Connect
NC-SI	Network Controller Sideband Interface
NIC	Network Interface Card
OCP	Open Compute Project
SCM	Secure and Control Module
SERDES	SERializer DESerializer
SRIS	Separate Reference Clocks with Independent Spread-Spectrum Clocking
SRNS	Separate Reference Clocks with No Spread-Spectrum Clocking
VFF	Vertical Form Factor