TLRB AIB PHY DESIGN PRELIMINARY SPECIFICATION

Rev 0.9

Prepared By

INTRINSIX CORP.

Document Revision Record

Revision	Date	Author	Description
0.0	03/12/2018	Intrinsix	Preliminary, initial Work in Progress Draft outline
0.1	04/02/2018	Intrinsix	• Add Sections: Timing, Clocks, Resets, Analog, Test Bus, Timing Diagrams, Intel Spec cross-reference Table, Latency, Bandwidth • Update AS AIB spec reference to v0.95 • Update AIB IO Buffer Diagram to include DFF resets. • Update for 80 Data bumps vs. 40.
0.2	04/09/2018	Intrinsix	Corrections, clarifications, and updates made following an Intrinsix internal review.
0.3	04/17/2018	Intrinsix	Updates following Architecture specification clarifications conveyed by Intel 4/10/2018 • The RX data redundancy MUX depicted in the architecture spec "Clocking Scheme" diagrams is only for delay matching. The actual RX data redundancy MUX function is to be implemented on the "odat" outputs of the AIB buffer. Diagrams/text updated accordingly. • The redundancy spares described in the AIBAUX section of the architecture spec are now exclusively to be implemented in the AIB IO channel (not AUX channel). Diagrams/text updated accordingly. Add por_vcc inputs for analog level shifters. Add a section for AIB IO Buffer digital decode. Add a diagram for the connections to the Redundancy spare AIB IO Cells. Remove config_done from top-level I/O ports. Updates to all Analog Block sections
0.4	04/19/2018	Intrinsix	Remove separate weak pull configuration signals (JTAG only). Editing updates resulting from Customer review comments. Updates to Digital I/O Ports table resulting from initial implementation work Updates to analog blocks: details added to delay chain, electromigration, clarifications to IO specs
0.5	05/07/2018	Intrinsix	Update AIB IO Cell block diagram to reflect clarifications conveyed by Intel (5/4/18) relative to

the BSR and Redundancy connections and logic, and the relationship to spare AIB IOs.

Add note to Digital I/O Ports table (section 5) to capture that idat_selb uses the opposite polarity for a Legacy AIB implementation (TLRB is NOT Legacy).

Changes due to AIB spec version 0.99 updates:

- Add Section 5.1 for new, even/odd mapping requirement of synchronous data adapter bus bits to AIB IOs for DDR and SDR modes.
- Update AIB Base Microbump mapping (section 3.4) to reflect new AIB IO bump order
- Add section 3.5 illustrating the uBump connections between two TLRB PHYs via a silicon interposer.
- Remove JTAG from AIB AUX channel since the new AIB spec now explicitly states that it is not required for the AUX channel.
- Update AIB IO Cell diagram (section 4.3) to depict new 3 to 1 redundancy MUX required for Spare IOs as show in in Fig. 3-32 “Full Redundancy Muxs” in the v0.99 Spec.
- Apply correction from new AIB 0.99 spec of AIB IO pullup/pulldown units from M ohm to K ohm.
- DDR mode is now optional for AIB Base configurations; however DDR mode is retained and is implemented for the TLRB implementation.

Add a “Repair Setting” column to the updated Microbump mapping tables to indicate which configuration bits to set to repair each pair of uBumps.

Update AIB IO Cell block diagram to depict some logic to reset (tri-stated with weak pull-down enabled) an AIB IO that is being repaired.

Clarification added relative to the option to implement the manual programmable delay line

			with standard cell logic (Section 9.3)
0.6	07/16/2018	Intrinsix	Correct a copy-paste error for a few indices of sl_data[] not matching corresponding RX [] indices in the Table: TLRB PHY AIB Base System Microbump Mapping (RX Data/Clocks/Resets; 80 Data IO Channel). Add a column to AIB Microbump Mapping tables (AIB Base Microbump Mapping) to indicate the JTAG connection order between AIB IO Cells. For clarity, add some notes to the Digital I/O Ports Table 11 indicating the (externally driven) power up default states of the jtag_* input port signals (as the states are specified by the Intel AIB Architecture Spec). Clarification: Figure 11 updated to explicitly show both SDR and DDR delay adjusts configuration top level input ports logic. Update the figure in the "RX Clocks" section to clarify that the redundant oclk and oclk_b inputs to the AIB IO cells are tied low except for the actual RX CLK cell. Explain in this section how these tied off connections affect the expected JTAG BSR scan results. Add a diagram that shows an example AIB IO Cell layout with 80 synchronous data IOs (12.1 AIB I/O Cell Array Placement). Add a **new section** (14 TLRB AIB PHY Controller & Register Map Extensions), that specifies an extended version of the TLRB AIB PHY (called TLRB AIB PHY Extended). The extended version adds an APB register map and JTAG TAP controller.
0.7	09/24/2018	Intrinsix	Update digital I/O ports (section 5) table to add a separate reset input (adap_irstb), and separate configuration inputs for TX drive strengths for clocks and resets. Also, update the table to indicate which signals (driven by the AIB AUX channel) are VCC IO voltage levels. Add a DFT note to the "TX Clock" section clarifying the use of the JTAG clock for TX clocking when doing wafer testing of DDR JTAG loopback to RX

			uBumps.
			
			Update Intel AIB Spec reference from r0.99 to r1.0. Miscellaneous editing updates from customer feedback.
			
			Updates to section (14 TLRB AIB PHY Controller & Register Map Extensions):
			• More detailed top-level block diagram
			• Digital I/O ports table update to add separate config_done port and omitted dig_test_bus port [].
			  ○ JTAG reset input added.
			  ○ APB Slave signals added.
			• JTAG BYPASS instruction encoding added.
			• Update diagram of POR Control to show more detail.
			• Update APB Slave Register addresses and paddr [] signal width to allow more space for IO channel registers.
			• Add extra fields in TX_CONFIG and RX_ENABLE Registers to account reduced AIB IO uBumps, and for each 40-bit data group.
			• Update DD_POR register to use more descriptive read/write field names, and update the associated field descriptions.
			• Add a sub-section (Bring Out of Reset Sequence) to describe the startup programming sequence.

0.8	11/12/2018	Intrinsix	Add a few (5) top-level output ports to Table 29 Digital I/O Ports (TLRB AIB PHY Extended). These new outputs are driven with the values of APB register fields, and offer a feature intended for system use. Update Digital IO Port Tables to indicate which AIB AUX channel signals are VCC_IO levels vs. VCC_DIG. Fix a few typos in the APB RESETs register description (APB register map) relative to the names of internal port signals associated with some of the register fields. For the same APB register, add detail describing that the status bits are synchronized.

			Add a translation table (Table 16 Translation of uBump Names & ID #s from this Spec to the Open Intel AIB Spec), to document the some differences in the AIB uBump names and ID numbers used in this specification vs. the newer, open Intel AIB specification.
0.9	05/12/2018	Intrinsix	Add new sections: A section is added describing AIB ID re-map features: **12.1.1 RTL Re-map of uBump AIB IDs using `defines** A section is added that summarizes the JTAG BSR infrastructure: **15 TLRB** A section is added that describes a new feature to support **multiple** AIB IO channels and 1 AIB AUX channel in a single AIB PHY: **16 Multiple AIB IO Channels.** A section is added that describes the selection of an alternate, logically equivalent JTAG TAP FSM: **14.2.1** Selection of alternate RTL code for the JTAG TAP FSM using a `define macro

Reference Documents

Table 1 Reference Document Items

Reference Item	Revision	Document
1	1.0	Intel AIB Architectural Overview (DARPA-Intel) "AIB Architecture Specification" *Intel Confidential, Copyright 2018 Intel Corporation* *Limited Use Rights to DARPA and other CHIPS program participants awarded by DARPA*
2	0.78	Chiplet Protocol Interface (CPI) Specification - Revision 0.78 Micron Technology, Inc. *Limited Distribution to DARPA CHIPS Participants* *Copyright 2018 Micron, Technology, Inc.*
3	v1.0	ARM AMBA 3.0 APB Specification, ARM IHI 0024B

Contents

Figures

Tables

1. Scope

This document describes the micro architecture of a TSMC 16FFC Long Reach Base configuration (TLRB) AIB PHY implementation.

The sections in this document cover any considerations necessary to specify the design of the TLRB AIB PHY digital logic such as block diagrams, block descriptions, logical timing diagrams, protocol specifications, interface descriptions, descriptions of integration, etc.

1.1. Acronyms, Abbreviations and Glossary

The following acronyms, abbreviations and terms are used in this document:

Table 1-1: Acronyms, Abbreviations and Terms

Term	Definition
AIB	Advanced Interface Bus (Chiplet to Chiplet bus interface)
AIB Channel	An AIB Channel includes an array of identical AIB IO cells, a DLL and DCC (optional for a Base configuration), and redundancy MUXs
AIB IO Channel	An AIB channel (1-24 per column) that is used for the main AIB synchronous data transfer.
AIB AUX Channel	An AIB channel (1 per AIB column) used auxiliary functions such as device detect signal , power-on reset signals, redundancy repair info non-volatile storage, and JTAG TAP controller.
AIB Column	An AIB Column includes and AIB AUX Channel and 1-24 AIB IO Channels
CHIPS	Common Heterogeneous Integration and Intellectual Property (IP) *Reuse* Strategies (DARPA program)
CPI	Chiplet Protocol Interface
DCC	Duty Cycle Correction module
DCD	Duty Cycle Distortion
SR	AIB Shift Register interface
TLRB	Short name for the AIB PHY implementation described by this specification (TSMC16FFC Long Reach Base AIB configuration)

1.2. Purpose

This document provides architecture and the implementation details of the TLRB AIB PHY design. This is a living document that will continue to capture the requirements as more details are flushed out, and as the AIB Architecture specification evolves.

This specification is primarily concerned with *front-end deliverables* such as TLRB AIB PHY digital logic design implementation documentation, and guidance for simulation, synthesis, and integration.

1.3. Goals of the Project

Intrinsix plans to design and deliver the TLRB AIB PHY. Chip level implementation is not included in this specification.

Back-end deliverables, including but not limited to the list below, are *not* documented by this specification.

- Cell models for custom driver and receiver (.lib: STA, Gate sims, ATPG)
- Bounding box with proposed port locations (LEF: if required early)
- AIB Channel macro GDS2
- AIB Channel macro netlist
- DRC & LVS reports

2. Introduction

The TLRB AIB PHY is logic block designed according to the AIB architecture specification, and is configured for a TLRB specific application. The TLRB AIB PHY includes both a digital logic design portion and an analog design portion. The analog design portion of the TLRB AIB PHY transmits and receives digital signals directly from the AIB IO microbumps. The description of the analog blocks is in a separate section (Analog Blocks) within this specification. The first sections of this specification (prior to the Analog section) cover mostly the digital portion of the design.

The analog blocks do transport digital signals, yet are distinguished from the digital logic by providing support for operating at a possibly different voltage than the core digital logic. Any voltage level shifting circuitry necessary to adapt to the core digital logic levels is included within the analog blocks. Further, the analog blocks receive digital level inputs to control analog functions such as drive strength, pull-up, pull-down, tristate, and power down circuitry. The RX analog blocks also include circuitry to receive a differential clock from the microbumps. The AIB RX clock that captures the microbump data is required by the AIB architecture spec to be delayed via a programmable delay chain. This delay chain is also an analog block so that the delay can be made to be relatively insensitive to PVT.

The figure below (Figure 1 TLRB AIB PHY in Context) depicts a TLRB AIB PHY block within a chiplet. The main synchronous data path to and from the AIB PHY is via the TX/RX DATA interfaces. These interfaces may be adapted to a CPI bridge which allows use of IP blocks that provide standard interfaces (such as AMBA AXI). Prior to making use of the main synchronous data path, the AIB PHY first goes through power on reset (POR) sequence, and is repaired (if necessary) using redundancy control, and is configured for each AIB microbump IO type (and for the DLL and DCC if present). Finally, the reset signals are released (de-asserted) by the reset controller to enable use of the synchronous data path.

A JTAG data register (DR) is integrated into the PHY for scan and DFT testing the uBump IOs.

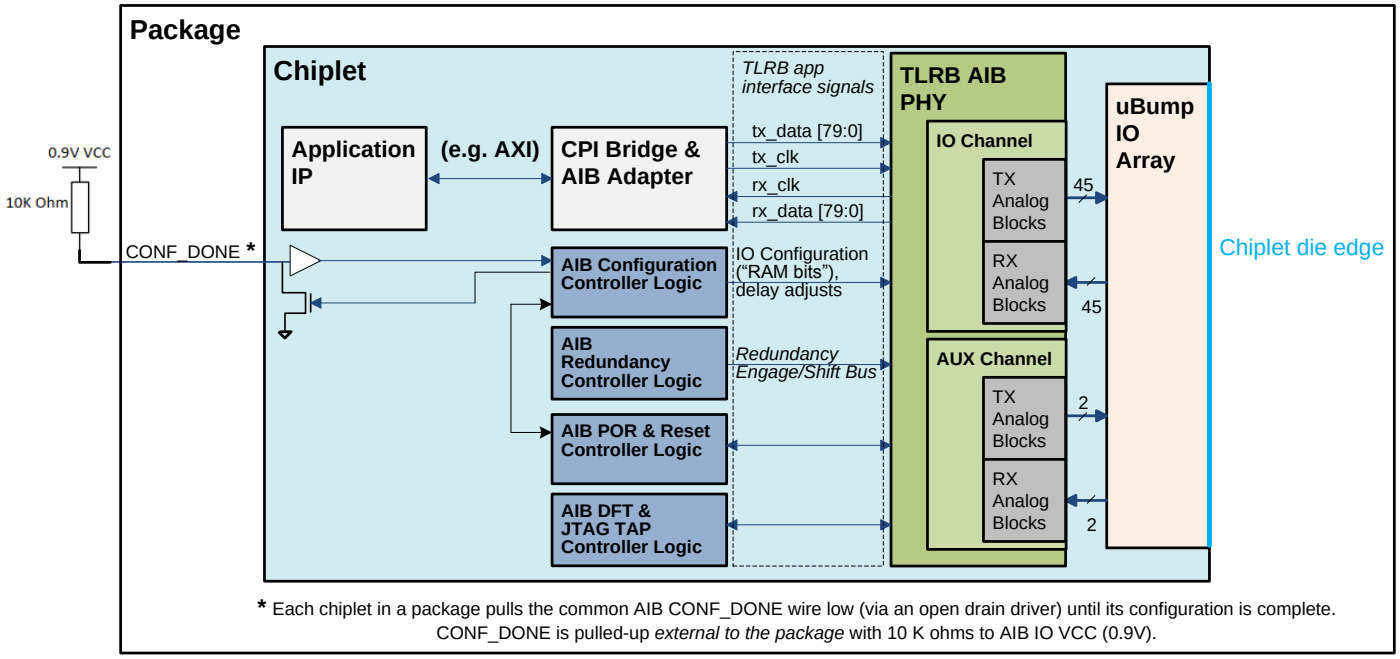


Figure 1 TLRB AIB PHY in Context

3. TLRB AIB PHY Requirements

3.1. AIB PHY Requirements Summary

The TLRB AIB PHY digital logic block implements/supports:

- an AIB *Base* configuration
  - Configurable for either DDR mode or SDR mode synchronous data
  - Redundancy scheme
  - JTAG boundary scan
  - *No* OSC clock
  - *No* serial shift register (SR)
  - *No* DLL or DCC
    - A placeholder manual mode DLL block is internally bypassed to implement a fixed ~500 ps delay for SDR mode
    - DDR mode (up to 400 MHz) requires a ~625 ps delay
  - *No* AIB Adapter logic
  - Superset option for Master/Slave
    - New to version 0.9 AIB spec. Channels can be either master or slave requiring *equal* TX and RX data micro bumps, and requires rotating of AIB IO assignments/partitioning. AIB Arch Spec: "separate master and slave design implementations are still recommended."
- 1 AIB *IO* channel
  - Total of 90 AIB IO cells (associated with 90 ubumps) partitioned (without Adapter logic) as follows:
    - 80 synchronous data (40 TX and 40 RX)
      - Designed to support fewer connected microbumps (e.g. 40 data with 20 TX and 20 RX)
    - 4 clock (2 TX and 2 RX for differential ended clocks)
    - 4 reset
    - 2 spare (active redundancy support *per channel*)
- 1 AIB *AUX* channel
  - 4 ubumps for POR and device detect (using *passive* redundancy)
    - Only 2 AIB IO cells due to a *passive* redundancy scheme using "double microbumps"
  - *No* JTAG controller (assuming a TAP controller implementing AIB JTAG instructions exists elsewhere in the chiplet).
  - *No* OSC clock oscillator
  - *No* non-volatile memory for storing Redundancy repair information is implemented within the TLRB AIB PHY AUX channel.
- Interfaces

    - An 80-bit synchronous TX/RX data interface suitable for simple adaptation to CPI (Chiplet Protocol Interface). According to the AIB Architecture specification, the *minimum* synchronous data width supported by any AIB channel is 40 ubumps. The synchronous data interface is 80 bits wide in each direction (TX/RX) when the AIB is in DDR mode. The synchronous interface in SDR mode uses only the low 40-bits of the interface.
    - AIB IO cell configuration interface
        - *No* configuration programming controller logic nor any configuration RAM is implemented within the AIB PHY
    - Active Redundancy engage interface (input signal per *pair* of AIB IO cells)
    - Reset and config_done interface
        - Observe config_done (package level C4 signal) to hold AIB block in standby
    - JTAG DFT interface to a TAP controller implemented *outside* of the AIB PHY
- Target a maximum of 1 GHz AIB synchronous data clocking (SDR), and 400 MHz DDR.
    - These targets are the maximums allowed for an AIB Base configuration without a DLL
    - Timing closure may limit the specified maximums actually realized
- Expected AIB synchronous data clocking is about ~250 MHz. This is one of the frequency points used for application bandwidth and latency performance calculations.
- Expected to route digital block IO to ubumps arrayed with a standard microbump pitch**.** The total number of ubumps is 94.
- Expected channel lengths of 3-5 mm (LR). **TBD.**
- Power of < 1 pJ per bit (amortized)
- 16 nm technology TSMC 16FFC
    - TSMC16FFC Choices
        - Voltage threshold (Vt) for standard cell libraries (HVT, SVT, LVT)
        - PODE vs. CPODE (compact) standard cells
        - Core voltage (e.g. 0.8 V +/- 10%)
        - IO voltage (e.g. 0.9 V +/- 10%)
        - Metal stack configuration & layers used by AIB PHY
- Voltage operating conditions range
- Temperature operating conditions range
    - Low Temp: **TBD**
    - High Temp: Designed for 110C
    - Provide life expectancy for 125C (**TBD**)

3.2. AIB PHY Configuration

The table below (Table 2 AIB PHY Implementation Configuration Parameters) lists some parameters as they are configured for the AIB TLRB PHY (TLRB column). The parameters include AIB Architecture Specification compliance points, technology node parameters, and application specific parameters such as bandwidth.

The table includes a second column that illustrates differences between the TLRB configuration and an example, prospective "AIB Base *Plus"* Legacy configuration.

Table 2 AIB PHY Implementation Configuration Parameters

Parameter	TLRB AIB PHY	Prospective AIB Plus Legacy	Comments
Process	TSMC 16FFC	GF 14LPP	
Metal stack			
Std cell library			
PVT corners			
min			
nom			
max			
AIB profiles supported	Base	Legacy, Gen1-LR Base plus?	
Functional waveforms	Yes	Yes	
AC Parameters	Yes	Yes	
JTAG boundary scan	Yes	Yes	
Redundancy scheme	Yes	Yes	
Pin order/type	Yes	Yes	
Bidirectional	Yes	Yes	
AUX (por/device_detect)	Yes	Yes	
DDR (a)	Yes (to 0.8Gbps)	Yes (to 2Gbps)	
SDR	Yes (to 1Gbps)	Yes (to 1Gbps)	
Free running clock (OSC)	No	Yes	Rqd if we add DLL/DCC
Serial shift register & reset controller	No	Yes	Rqd if we add DLL/DCC
DLL	No	Yes	
DCC	No	Yes	
Adapter register	No	Yes	Rqd if we add DLL/DCC
Adapter phase comp FIFO	No	?	Rqd if we add DLL/DCC
Adapter elastic FIFO	No	?	
Adapter FIFO 1x2x support	No	?	
Superset option for Master/Slave	No	No	
Configuration handshake - Conf_done	Yes	Yes	
DFT/DFX for KGD	Yes	Yes	
Min real data bandwidth	2Gbps+2Gbps		Real AXI app bandwidth (TX + RX)
Channel width	90+6	96	
RDL/pad included	No	?	
AUX width	6		
RDL/pad included	No	?	
AUX functions	Redundancy and config storage outside. JTAG TAP outside. Reset controller outside.		

3.3. Intel AIB Architecture Specification versus TLRB AIB PHY

The tables below (Table 3, Table 4, Table 5, and Table 6) provide guidance information relative to the applicability of every numbered section of the Intel AIB Architecture specification versus the AIB TLRB PHY implementation.

Table 3 AIB Arch Spec Sections 1-2

Section #	Section Heading	TLRB AIB PHY	Comment
1	**Introduction**	N/A	heading only
1.1	Objective of Specification	yes	
1.2	Terms and Acronyms	yes	
1.3	Document Organization	N/A	
1.4	Participants	N/A	
2	**Architecture (Topology)**	yes	
2.1	Compatibility Interface Points	yes	
2.1.1	Medium Dependent Interfaces (MDI)	yes	
2.1.2	Attachment Unit Interface (AUI)	no	TLRB AIB PHY implementation is PMA only
2.1.3	Media Independent Interface (MII)	no	No protocol layer is implemented.
2.2	Layer Interfaces	partial	Only an AIB PHY layer is implemented. No MAC, PLS, AUI.
2.3	Repeater	no	No repeater is implemented.
2.4	Scope of Standardization	N/A	
2.5	Interface Definition	partial	AIB Base configuration exceptions. No free running OSC clock.
2.6	Interface Type	yes	
2.7	Interface Generation	yes	GEN1
2.8	Interface Signal Class and Grouping	partial	AIB Base configuration exceptions. No SR or OSC clock. No DLL or DCC.

Table 4 AIB Arch Spec Section 3

Section #	Section Heading	TLRB AIB PHY	Comment
3	**Physical Layer Specification**	N/A	heading only
3.1	Interface Physical Arrangement	yes	A microbump arrangement and pitch (compliant with this section)
3.2	**AIB Channel**	N/A	heading only
3.2.1	Channel Definition	yes	
3.2.2	**AIB IO**	N/A	heading only
3.2.2.1	AIB I/O Structure	yes	AIB Base configuration exceptions. No DLL or DCC operating modes.
3.2.2.1.1	Output Buffer Operation	yes	
3.2.2.1.2	Input Buffer Operation	yes	AIB Base configuration exceptions. DDR limited to 400 MHz.
3.2.2.2	DLL	no	AIB Base configuration exceptions. No DLL.
3.2.2.3	DCC	no	AIB Base configuration exceptions. No DCC.
3.2.2.4	Clock Tree	yes	
3.2.2.4.1	DDR Clock Tree	yes	AIB Base configuration exceptions. No DLL or DCC.
3.2.2.4.2	SDR Clock Tree	yes	AIB Base configuration exceptions. No DLL or DCC.
3.2.2.4.3	SDR Shift Register Clock Tree	no	AIB Base configuration exceptions. No SR or OSC clock.
3.2.2.5	Redundancy	yes	AIB Base configuration exceptions. No SR or OSC clock.
3.2.2.6	I/O and Microbump Placement for Packaging	yes	
3.2.2.7	Power Domain and Level Shifter	yes	
3.2.2.8	Electrical Specification	yes	
3.2.3	**AIB Adapter**	N/A	heading only
3.2.3.1	Overview	partial	AIB Base configuration exceptions. No Adapter is implemented other than clock and data grouping
3.2.3.2	Use Models	yes	AIB Base configuration only
3.2.3.3	Signal Class	N/A	heading only
3.2.3.3.1	Synchronous Data Path Transfer	yes	RX and TX signals are grouped
3.2.3.3.2	FIFO	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.3	FIFO Phase Compensation	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.4	Basic Elastic FIFO	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.5	FIFO Register Mode	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.6	FIFO Bonding	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.7	FIFO Double Read/Write (2x) Mode	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.8	Word Marking and Word Alignment	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.9	Serial Shift Chain Transfers	no	AIB Base configuration exceptions. No Adapter logic.
3.2.3.3.10	Direct Asynchronous Signals	yes	POR and reset signals are implemented for an AIB Base configuration.

Table 5 AIB Arch Spec Sections 3.3 - 7

Section #	Section Heading	TLRB AIB PHY	Comment
3.3	**AIBAUX**	N/A	heading only
3.3.1	Overview	yes	AIB Base configuration exceptions. No OSC. External JTAP tap controller. No redundancy repair info non-volatile mem implemented.
3.3.2	Independent (Free Running) Clock Source	no	AIB Base configuration exception.
3.3.3	Redundancy Microbumps	yes	
3.3.4	JTAG	no	No JTAG TAP controller or JTAG bumps are implemented (optional)
3.3.5	POR Sync Up and Test	yes	
3.3.6	Superset Option for Master and Slave	yes	AIB PHY implementation can be either a master or slave w.r.t. POR, Device Detect, and resets signals.
4	**Reset Sequence**	N/A	heading only
4.1	Reset Scheme	no	AIB Base configuration. No SR or reset controller.
4.2	Reset Sequence Timing Requirement	no	AIB Base configuration. No SR or reset controller.
4.3	Synchronous Reset	no	AIB Base configuration. No SR or reset controller.
4.4	Asynchronous Reset	no	AIB Base configuration. No SR or reset controller.
4.5	Reset State Machine	no	AIB Base configuration. No SR or reset controller.
4.5.1	Oscillator Reset State Machine	no	AIB Base configuration. No SR or reset controller.
4.5.2	Slave_TX/Master_RX Reset State Machine	no	AIB Base configuration. No SR or reset controller.
4.5.3	Master_TX to Slave_RX Reset State Machine	no	AIB Base configuration. No SR or reset controller.
5	**JTAG Boundary Scan**	N/A	heading only
5.1	Overview	yes	
5.2	Reset Mode	yes	
5.3	Reset Override Mode	yes	
5.4	Shift Mode	yes	
5.5	Transmit Mode	yes	
5.6	Intest Mode	yes	
5.7	Jtag_clksel Mode	yes	
5.8	Jtag_clkdr Gated Clocking	yes	
5.9	Fast Clock Override Mode	no	AIB Base configuration exception. No Adapter logic is implemented.
6	**Configuration**	yes	
7	**AIB Feature Summary**	yes	"Required for Base Config." compliance table column applies

Table 6 AIB Arch Spec Section 8

Section #	Section Heading	TLRB AIB PHY	Comment
8	**Appendix: Optional Design Guidelines and Recommendations**	yes	Optional features (not required for compliance)
8.1	Driver Strength Selection	yes	**TBD**
8.2	DLL High Level Description	no	AIB Base configuration exception. No DLL (except "manual" mode)
8.3	DCC High Level Description	no	AIB Base configuration exception. No DCC
8.4	Receive and Transmit Clock Tree	no	AIB Base configuration exception. No DLL or DCC.
8.5	Adapter FIFO Usage Models	no	AIB Base configuration exception. No Adapter logic is implemented.
8.6	Analog Bit Capture	no	AIB Base configuration exception. No SR.
8.7	DFT/DFX	partial	AIB Base configuration exceptions. No OSC, SR, DLL, DCC, Adapter.
8.7.1	AIB I/O and AIB AUX	yes	
8.7.1.1	AIB IO Leakage Test	yes	
8.7.1.2	AIB IO and Adapter Test Path	no	AIB Base configuration exception. No Adapter logic is implemented.
8.7.1.3	Stuck at Test	yes	
8.7.1.4	DCC and DLL Test Path	no	AIB Base configuration exceptions. No DLL or DCC.
8.7.1.5	Reset State Machine Override	no	AIB Base configuration exception. No SR.
8.7.1.6	Digital Test Bus	yes	Debug signals are **TBD**.
8.7.1.7	POR and Device_Detect Override	yes	
8.7.1.8	OSC Override	no	AIB Base configuration exception. No OSC, SR.
8.7.2	AIB Adapter	no	AIB Base configuration exception. No Adapter logic is implemented.
8.7.2.1	Loopback	no	AIB Base configuration exception. No Adapter logic is implemented.
8.7.2.2	Scan	no	AIB Base configuration exception. No Adapter logic is implemented.
8.7.2.3	Shift Register Test	no	AIB Base configuration exception. No Adapter logic is implemented.
8.8	Power Bump Array	yes	
8.9	Chiplet Datasheet	yes	

3.4. AIB Base Microbump Mapping

The table below (Table 7 *Reference* AIB Base System Microbump Mapping for a 40 Data IO Channel) is directly derived from the AIB architecture specification by shifting down the uBump IDs from the corresponding Base *Plus* table in the AIB spec ("Table 3-2: Channel Microbump Ordering and Mapping for 40 (up to 120) IO AIB Plus System").

According to AIB architecture specification ("AIB Design Feature Summary" Table), an AIB Base configuration does *not* require: "Serial shift register reset controller and forwarded clock (ms_clk/b,

sl_clk/b)". Accordingly, the following AIB microbumps are absent from the *reference* 40 Data IO AIB PHY microbump mapping table:

- sr_sl_load, sr_sl_data, sr_sl_clk, sr_sl_clkb (*excluded*)
- sr_ms_load, sr_ms_data, sr_ms_clk, sr_ms_clkb (*excluded*)
- ms_clk, ms_clkb, sl_clk, sl_clkb (*excluded*)

For the TLRB AIB PHY, the reference table from the AIB spec for 40 Data bumps is **extended to 80 Data ubumps**.

The AIB TLRB PHY microbumps are mapped as in the tables below: Table 8 TLRB PHY AIB Base System Microbump Mapping (*TX* Data/Clock, Resets, Spares; 80 Data *IO* Chan), and Table 9 TLRB PHY AIB Base System Microbump Mapping (*RX* Data/Clocks/Resets; *80* Data IO Channel). The extended ubumps are shaded in these tables and are designated by uBump IDs of AIB50-AIB89. The low numbered uBumps IDs are retained to match those in the AIB Spec (shifted down for a Base configuration), and the added (extended) uBumps are shaded and assigned higher numbered IDs.

The "RX [ ]" named ubumps indicate chiplet bumps which receive data into the chiplet as strobed by RXCLK/RXCLKB (pseudo differential ended input clock). The "TX [ ]" named ubumps indicate chiplet bumps which transmit data from the chiplet as clocked out by TXCLK/TXCLKB (pseudo differential ended output clock).

The MRSTN and MARSTN are reset signals transmitted out of the chiplet to the uBumps when the AIB PHY is a master.

The SRSTN and SARSTN are reset signals received into the chiplet from the uBumps when the AIB PHY is a master.

There is a pair of "spare" bumps for redundancy that can be configured to be either TX or RX.

Continuing to reference the 80 uBump mapping tables, the right hand column labeled "Bits of redun_engage[ ] to set to 1 to Repair uBump pair" refers the top-level redundancy configuration input port vector (redun_engage [ ]). See section 5 for a more detailed description of this port. The table entries in the column indicate which bit(s) to set to a 1 in the redun_engage[ ] port vector in order to repair the uBump pair in the corresponding row. The other bits must be cleared to a 0. All bits of redun_engage[ ] are cleared to 0 if no repair is necessary.

Table 7 *Reference* AIB Base System Microbump Mapping for a 40 Data IO Channel

Bump ID	uBump Name	Pin name	Bump ID	uBump Name	Pin name
AIB48	sl_data[19]	RX[19]	AIB49	sl_data[18]	RX[18]
AIB46	sl_data[17]	RX[17]	AIB47	sl_data[16]	RX[16]
AIB44	sl_data[15]	RX[15]	AIB45	sl_data[14]	RX[14]
AIB42	sl_data[13]	RX[13]	AIB43	sl_data[12]	RX[12]
AIB40	sl_data[11]	RX[11]	AIB41	sl_data[10]	RX[10]
AIB38	sl_tran_clkb	RXCLKB	AIB39	sl_tran_clk	RXCLK
AIB36	sl_data[9]	RX[9]	AIB37	sl_data[8]	RX[8]
AIB34	sl_data[7]	RX[7]	AIB35	sl_data[6]	RX[6]
AIB32	sl_data[5]	RX[5]	AIB33	sl_data[4]	RX[4]
AIB30	sl_data[3]	RX[3]	AIB31	sl_data[2]	RX[2]
AIB28	sl_data[1]	RX[1]	AIB29	sl_data[0]	RX[0]
AIB26	sl_adapter_rstn	SARSTN	AIB27	sl_rstn	SRSTN
AIB24	spare[0]	spare[0]	AIB25	spare[1]	spare[1]
AIB22	ms_rstn	MRSTN	AIB23	ms_adapter_rstn	MARSTN
AIB20	ms_data[0]	TX[0]	AIB21	ms_data[1]	TX[1]
AIB18	ms_data[2]	TX[2]	AIB19	ms_data[3]	TX[3]
AIB16	ms_data[4]	TX[4]	AIB17	ms_data[5]	TX[5]
AIB14	ms_data[6]	TX[6]	AIB15	ms_data[7]	TX[7]
AIB12	ms_data[8]	TX[8]	AIB13	ms_data[9]	TX[9]
AIB10	ms_tran_clk	TXCLK	AIB11	ms_tran_clkb	TXCLKB
AIB8	ms_data[10]	TX[10]	AIB9	ms_data[11]	TX[11]
AIB6	ms_data[12]	TX[12]	AIB7	ms_data[13]	TX[13]
AIB4	ms_data[14]	TX[14]	AIB5	ms_data[15]	TX[15]
AIB2	ms_data[16]	TX[16]	AIB3	ms_data[17]	TX[17]
AIB0	ms_data[18]	TX[18]	AIB1	ms_data[19]	TX[19]

Table 8 TLRB PHY AIB Base System Microbump Mapping (*TX* Data/Clock, Resets, Spares; 80 Data *IO* Chan)

Bump ID	uBump Name	Pin name	Bump ID	uBump Name	Pin name	Bits of redun_engage [ ] to set to 1 to Repair uBump pair	JTAG chain (0 = AIB50 = jtag_ scan_in)
AIB24	spare[0]	spare[0]	AIB25	spare[1]	spare[1]	NONE (spare ubumps are never repaired)	44 -> 45
AIB22	ms_rstn	MRSTN	AIB23	ms_adapter_rstn	MARSTN	[22:21]	42 -> 43
AIB20	ms_data[0]	TX[0]	AIB21	ms_data[1]	TX[1]	[22:20]	40 -> 41
AIB18	ms_data[2]	TX[2]	AIB19	ms_data[3]	TX[3]	[22:19]	38 -> 39
AIB16	ms_data[4]	TX[4]	AIB17	ms_data[5]	TX[5]	[22:18]	36 -> 37
AIB14	ms_data[6]	TX[6]	AIB15	ms_data[7]	TX[7]	[22:17]	34 -> 35
AIB12	ms_data[8]	TX[8]	AIB13	ms_data[9]	TX[9]	[22:16]	32 -> 33
AIB10	ms_tran_clk	TXCLK	AIB11	ms_tran_clkb	TXCLKB	[22:15]	30 -> 31
AIB8	ms_data[10]	TX[10]	AIB9	ms_data[11]	TX[11]	[22:14]	28 -> 29
AIB6	ms_data[12]	TX[12]	AIB7	ms_data[13]	TX[13]	[22:13]	26 -> 27
AIB4	ms_data[14]	TX[14]	AIB5	ms_data[15]	TX[15]	[22:12]	24 -> 25
AIB2	ms_data[16]	TX[16]	AIB3	ms_data[17]	TX[17]	[22:11]	22 -> 23
AIB0	ms_data[18]	TX[18]	AIB1	ms_data[19]	TX[19]	[22:10]	20 -> 21
AIB68	ms_data[20]	TX[20]	AIB69	ms_data[21]	TX[21]	[22:9]	18 -> 19
AIB66	ms_data[22]	TX[22]	AIB67	ms_data[23]	TX[23]	[22:8]	16 -> 17
AIB64	ms_data[24]	TX[24]	AIB65	ms_data[25]	TX[25]	[22:7]	14 -> 15
AIB62	ms_data[26]	TX[26]	AIB63	ms_data[27]	TX[27]	[22:6]	12 -> 13
AIB60	ms_data[28]	TX[28]	AIB61	ms_data[29]	TX[29]	[22:5]	10 -> 11
AIB58	ms_data[30]	TX[30]	AIB59	ms_data[31]	TX[31]	[22:4]	8 -> 9
AIB56	ms_data[32]	TX[32]	AIB57	ms_data[33]	TX[33]	[22:3]	6 -> 7
AIB54	ms_data[34]	TX[34]	AIB55	ms_data[35]	TX[35]	[22:2]	4 -> 5
AIB52	ms_data[36]	TX[36]	AIB53	ms_data[37]	TX[37]	[22:1]	2 -> 3
AIB50	ms_data[38]	TX[38]	AIB51	ms_data[39]	TX[39]	[22:0]	0 -> 1

Table 9 TLRB PHY AIB Base System Microbump Mapping (*RX* Data/Clocks/Resets; *80* Data IO Channel)

Bump ID	uBump Name	Pin name	Bump ID	uBump Name	Pin name	Bit(s) of redun_engage [ ] to set to 1 to Repair uBump pair	JTAG chain (89 = AIB89 = jtag_ scan_out)
AIB88	sl_data[39]	RX[39]	AIB89	sl_data[38]	RX[38]	[44:23]	88 -> 89
AIB86	sl_data[37]	RX[37]	AIB87	sl_data[36]	RX[36]	[43:23]	86 -> 87
AIB84	sl_data[35]	RX[35]	AIB85	sl_data[34]	RX[34]	[42:23]	84 -> 85
AIB82	sl_data[33]	RX[33]	AIB83	sl_data[32]	RX[32]	[41:23]	82 -> 83
AIB80	sl_data[31]	RX[31]	AIB81	sl_data[30]	RX[30]	[40:23]	80 -> 81
AIB78	sl_data[29]	RX[29]	AIB79	sl_data[28]	RX[28]	[39:23]	78 -> 79
AIB76	sl_data[27]	RX[27]	AIB77	sl_data[26]	RX[26]	[38:23]	76 -> 77
AIB74	sl_data[25]	RX[25]	AIB75	sl_data[24]	RX[24]	[37:23]	74 -> 75
AIB72	sl_data[23]	RX[23]	AIB73	sl_data[22]	RX[22]	[36:23]	72 -> 73
AIB70	sl_data[21]	RX[21]	AIB71	sl_data[20]	RX[20]	[35:23]	70 -> 71
AIB48	sl_data[19]	RX[19]	AIB49	sl_data[18]	RX[18]	[34:23]	68 -> 69
AIB46	sl_data[17]	RX[17]	AIB47	sl_data[16]	RX[16]	[33:23]	66 -> 67
AIB44	sl_data[15]	RX[15]	AIB45	sl_data[14]	RX[14]	[32:23]	64 -> 65
AIB42	sl_data[13]	RX[13]	AIB43	sl_data[12]	RX[12]	[31:23]	62 -> 63
AIB40	sl_data[11]	RX[11]	AIB41	sl_data[10]	RX[10]	[30:23]	60 -> 61
AIB38	sl_tran_clkb	RXCLKB	AIB39	sl_tran_clk	RXCLK	[29:23]	58 -> 59
AIB36	sl_data[9]	RX[9]	AIB37	sl_data[8]	RX[8]	[28:23]	56 -> 57
AIB34	sl_data[7]	RX[7]	AIB35	sl_data[6]	RX[6]	[27:23]	54 -> 55
AIB32	sl_data[5]	RX[5]	AIB33	sl_data[4]	RX[4]	[26:23]	52 -> 53
AIB30	sl_data[3]	RX[3]	AIB31	sl_data[2]	RX[2]	[25:23]	50 -> 51
AIB28	sl_data[1]	RX[1]	AIB29	sl_data[0]	RX[0]	[24:23]	48 -> 49
AIB26	sl_adapter_rstn	SARSTN	AIB27	sl_rstn	SRSTN	[23]	46 -> 47

3.5. AIB Microbump Connections PHY to PHY

The diagram below illustrates how the microbumps of a Master TLRB AIB PHY on one chiplet may be mapped and connected (through a silicon interposer) to the microbumps of a Slave TLRB AIB PHY on another chiplet. A rotated layout (180 degrees) of the Master Chiplet is depicted for the Slave Chiplet such that TX signals may connect to corresponding RX signals through the interposer with minimal length routes.

The Device Detect and POR signals may be connected with longer interposer routes since they are asynchronous.

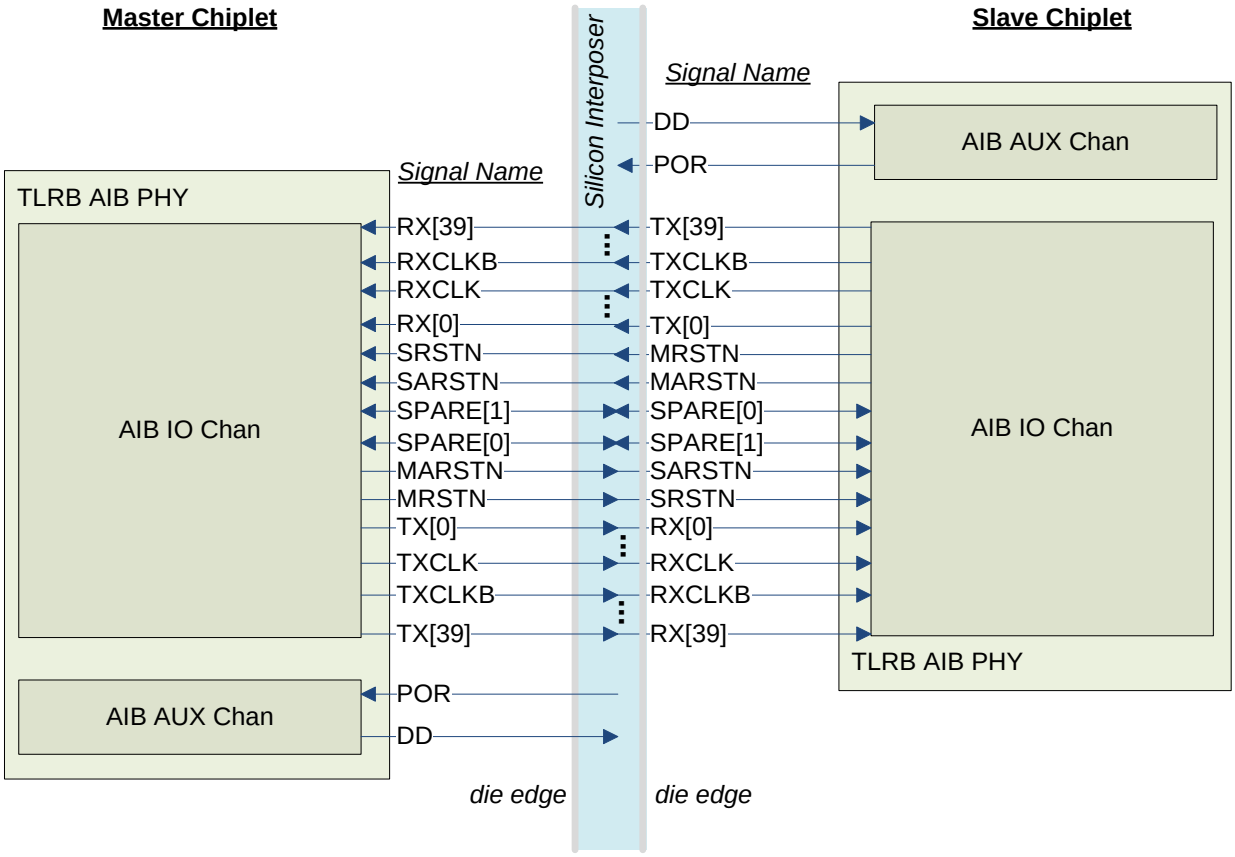


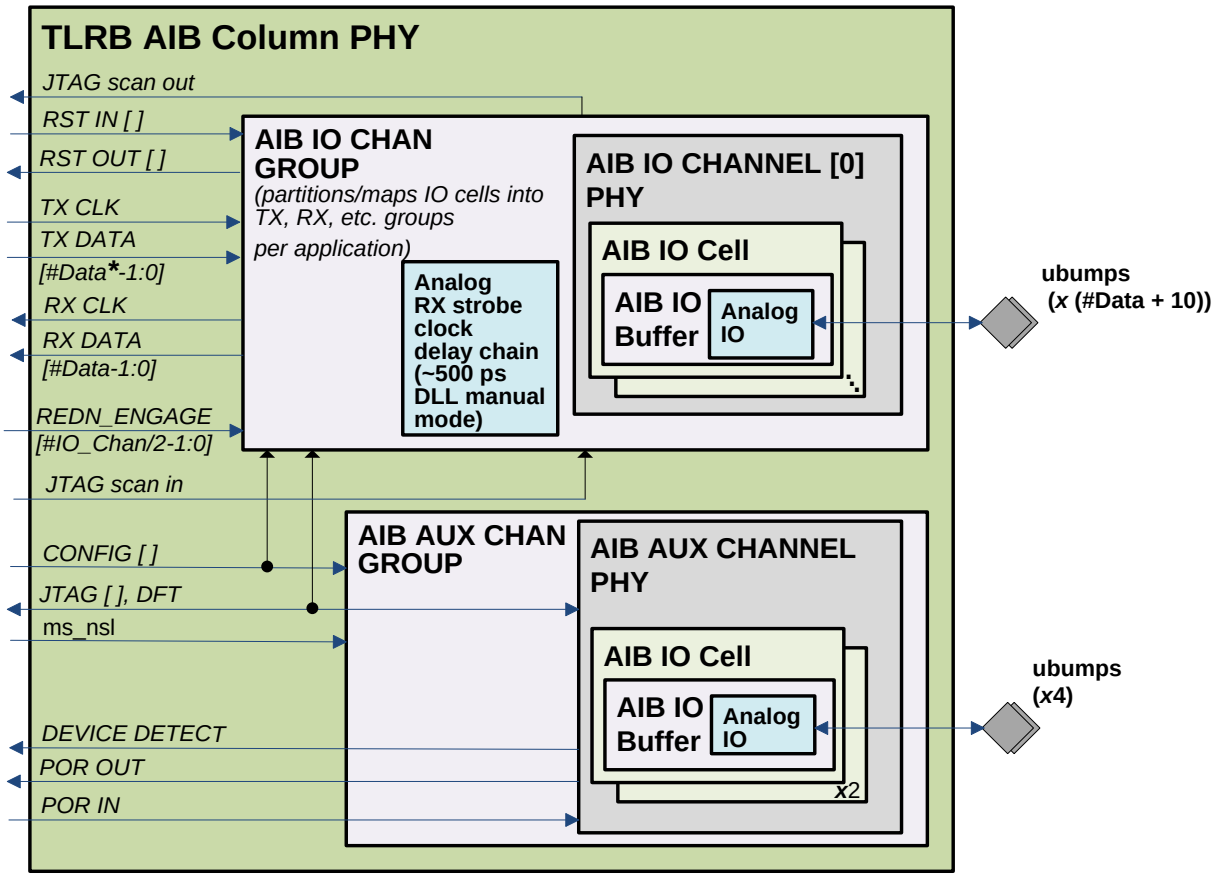
Figure 2 TLRB Microbump PHY to PHY connections.

4. AIB Digital Block Diagrams

4.1. TLRB AIB PHY Top

The figure below (Figure 3 TLRB AIB PHY Top Level Block Diagram) depicts the top level blocks, interfaces, and external ports of the TLRB AIB PHY. Also depicted are the hierarchical sub-blocks. Diagrams of the sub-blocks are described and illustrated in more detail in the immediately succeeding sections.

The AIB IO Channel Group wrapper block partitions, maps and combines the AIB IO Channel sub-block cell ports into RX, TX and Configuration groups that match the intended AIB application configuration, considering whether the PHY is master or slave (ms_nsl). The *lightweight* logic of the AIB IO Group block consists primarily of configurable wiring/mapping assignments, and in some cases buffering to drive multiple inputs to the AIB channel blocks with a common signal (e.g. use of a common drive strength for all TX AIB IO cells). Miscellaneous logic such as clock buffers and DFT overrides are also included in the top level PHY logic. There is one AIB IO Channel Group block per AIB IO channel. The AIB AUX Channel Group block performs a similar function for the AIB AUX channel.



* **#Data** is the number of synchronous Data microbumps per IO channel
#IO_Chan is the number of AIB IO Channel microbumps (**#Data** + 10)

Figure 3 TLRB AIB PHY Top Level Block Diagram

4.2. AIB IO Buffer

The figure below (Figure 4 AIB IO Buffer Block Diagram) depicts the lowest level of the digital logic.

The digital logic is expected to be a mix of synthesizable RTL and instantiated gates. At a minimum, gates are instantiated with standard logic cells (from a digital IP technology library) for any multiplexers (MUXs) which propagate clocks using “balanced” cell versions (e.g. CK prefix) in order to preserve the duty cycle integrity of propagated clock signals.

The MUX depicted in the diagram (in the RX DIGITAL block) is implemented internal to the AIB IO Buffer) so that its delay may be matched to the clock path redundancy MUX (implemented at the channel level) for RX synchronous data paths.

The AIB IO Buffer block includes a set of analog blocks that directly drive (TX) and receive (RX) signals from the microbumps (with AIB microbump voltage levels). The digital portion of the AIB IO Buffer is illustrated according to the AIB architecture specification (i.e. “Figure 3-13: Full AIB I/O Buffer”), and comprehends other details described throughout that specification.

The AIB IO Buffer block includes a sub-block called ”TX CLK DIG” that includes all of the digital logic in the TX clock path *from* the top-level TLRB AIB PHY TX clock input port (which plays the role of **RefClk** in the NE/FE in the eye mask test setups), *to* the AIB IO TX analog driver. The purpose of grouping all this logic into a single block is to allow the analog TX paths to be designed to meet the required TX output eye and skew timing independently of all the other digital logic. Also, this sub-block is the buffered source of the clock (ilaunch_clk) for the digital logic TX DFFs and latch (for which the timing is less critical than the path to the output TX driver).

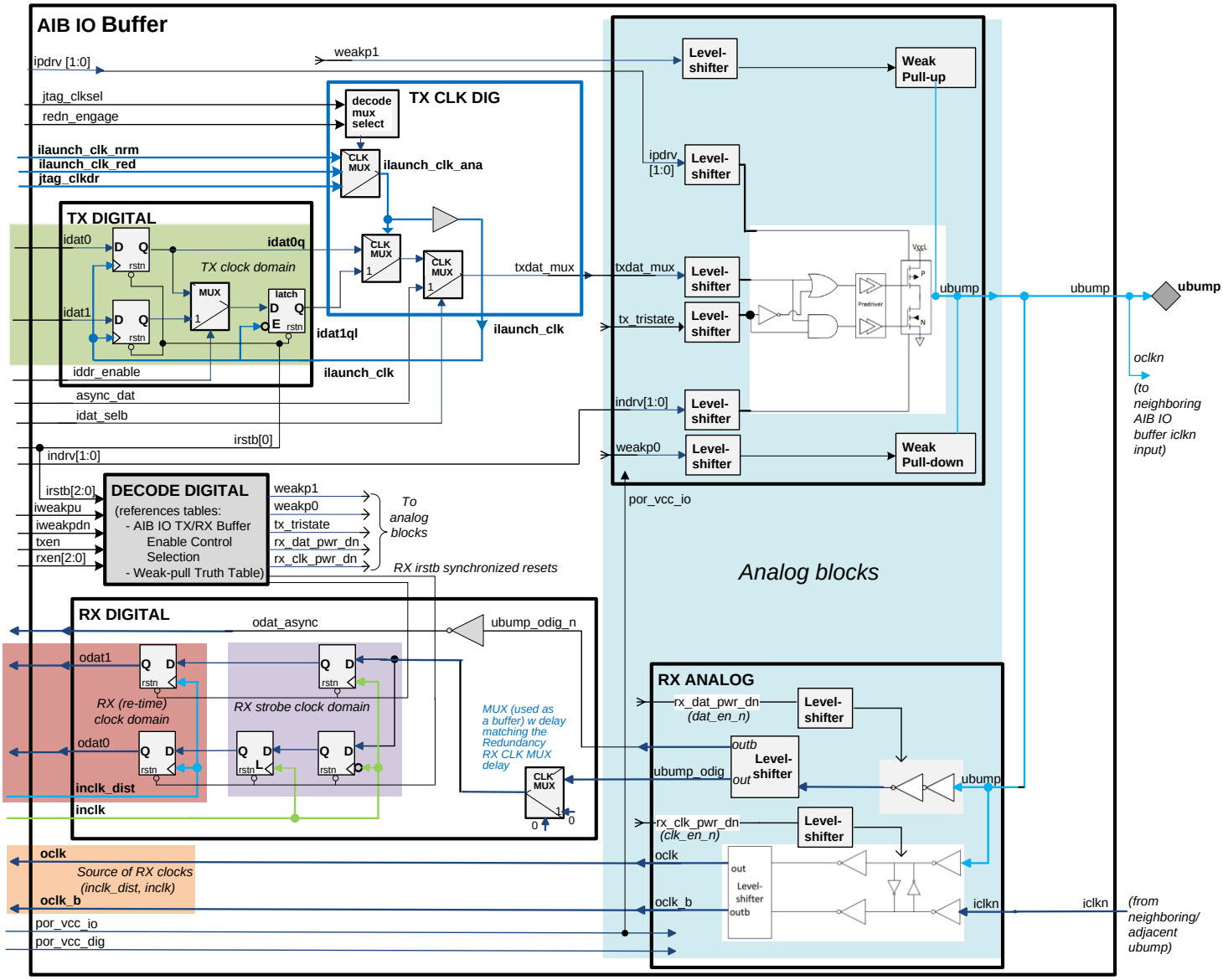


Figure 4 AIB IO Buffer Block Diagram

4.2.1. Decode Digital Block

The Decode Digital block depicted in the diagram above generates output digital control signals destined for the Analog blocks, and resets for the TX/RX digital DFFs and latches. The outputs of the block are generated (decoded) according to the tables ("Buffer Enable Control Selection for Full AIB IO Cell" and "Weak-pull True Table") in the AIB Architecture Spec.

The irstb [2:0] input port to the AIB IO Buffer consists of 3 flavors of the irstb (asserts active low and negates active high) reset signal:

- irstb[2] is irstb (analog reset)
- irstb[1] is irstb synchronized such that it asserts asynchronously and de-asserts (negates) synchronously relative to RX inclk
- irstb[0] is irstb synchronized such that it asserts asynchronously and de-asserts (negates) synchronously relative to TX tx_clk

4.2.1.1. TX Decode Logic

The logic for the TX Decode outputs is described by the following assignments (Verilog notation):

```
// TX Driver Enable
//
tx_en = txen & irstb[2]; // Enable driver if configured txen and reset is negated.


// TX Driver Tristate
//
tx_tristate = ~tx_en; // Opposite polarity of TX Driver Enable (tx_en)


// Pulldown Enable
//
weakp0 = iweakpdn    |                          // DFT pulldown enable OR-term
         (~irstb[2]) |                          // Pulldown if irstb is asserted OR-term
         (tx_tristate & (rxen[2:0] == 3'b010)); // Tristated TX and Disabled RX (rxen=010)


// Pullup Enable. Enable pullup if configured iweakpu and irstb is negated (DFT only).
//
weakp1 = iweakpu & irstb[2];
```

The resets for the TX DFFs and latch (which are clocked by tx_clk) are driven directly by the irstb[0] signal.

4.2.1.2. RX Decode Logic

The logic for the RX Decode outputs is described by the following assignments (Verilog notation):

```
// RX Data Receiver Enable
//
rx_dat_en = irstb[2] & ((rxen[2:0] == 3'b000) |   // 000 = Async data input.
                        (rxen[2:0] == 3'b100) |   // 100 = Sync SDR input.
                        (rxen[2:0] == 3'b001)  ); // 001 = Sync DDR input.


// RX Data Receiver Power Down (disable)
//
rx_dat_pwr_dn = ~rx_dat_en; // Opposite polarity of RX Data Receiver Enable


// RX Clock Receiver Enable
//
rx_clk_en = irstb[2] & (rxen[2:0] == 3'b011); // 011 = Input Clock buffer enabled.


// RX Clock Receiver Power Down (disable)
//
rx_clk_pwr_dn = ~rx_clk_en; // Opposite polarity of RX Clock Receiver Enable
```

The resets for the RX DFFs and latch (which are clocked by inclk) are based on the irstb[1] signal. However, to save power, each data bit path (odat0 and odat1) uses a separate, gated reset signal based on irstb[1] as follows:

```
// Resets to Bit 0 DFFs and Latch can only be negated high (released) if irstb is released
// AND the rxen configuration input decodes to either SDR or DDR.
//
<Bit 0 DFFs/Latch Reset> = irstb[1] & ((rxen[2:0] == 3'b100) |   // 100 = Sync SDR input.
                                       (rxen[2:0] == 3'b001)  ); // 001 = Sync DDR input.


// Resets to Bit 1 DFFs can only be negated high (released) if irstb is released
// AND the rxen configuration input decodes to DDR.
//
 <Bit 1 DFFs Reset> = irstb[1] & (rxen[2:0] == 3'b001); // 001 = Sync DDR input.
```

The configuration bits affecting the resets signals above (rxen[2:0]) are static when the corresponding irstb signal is negated (released). The configuration bits may only be changed while reset is asserted low.

4.3. AIB IO Cell

The figure below (Figure 5 AIB IO Cell Block Diagram) depicts the AIB IO Cell sub-block, which wraps the lower (lowest) level AIB IO buffer block together with JTAG DFT logic and redundancy multiplexers. AIB channels implement an array of AIB IO Cell blocks each of which is *identical*. A common identical netlist is generated for the AIB IO Cell, and this common netlist is instantiated (physically placed) multiple times to form an array of identical AIB IO Cells within each AIB channel. Every placed AIB IO Cell retains the capability to function as *any* of the defined AIB IO types:

- synchronous TX data (SDR/DDR mode sub-types)
- synchronous RX data
- TX clock (uses the TX data type configured in DDR mode)
- RX clock (uses only the differential receiver)
- asynchronous TX
- asynchronous RX

The irstb reset input ports (including both asynchronous and synchronous versions) are multiplexed with the JTAG DFT reset at the AIB IO channel level, which is a hierarchical level external to and above the AIB IO cell.

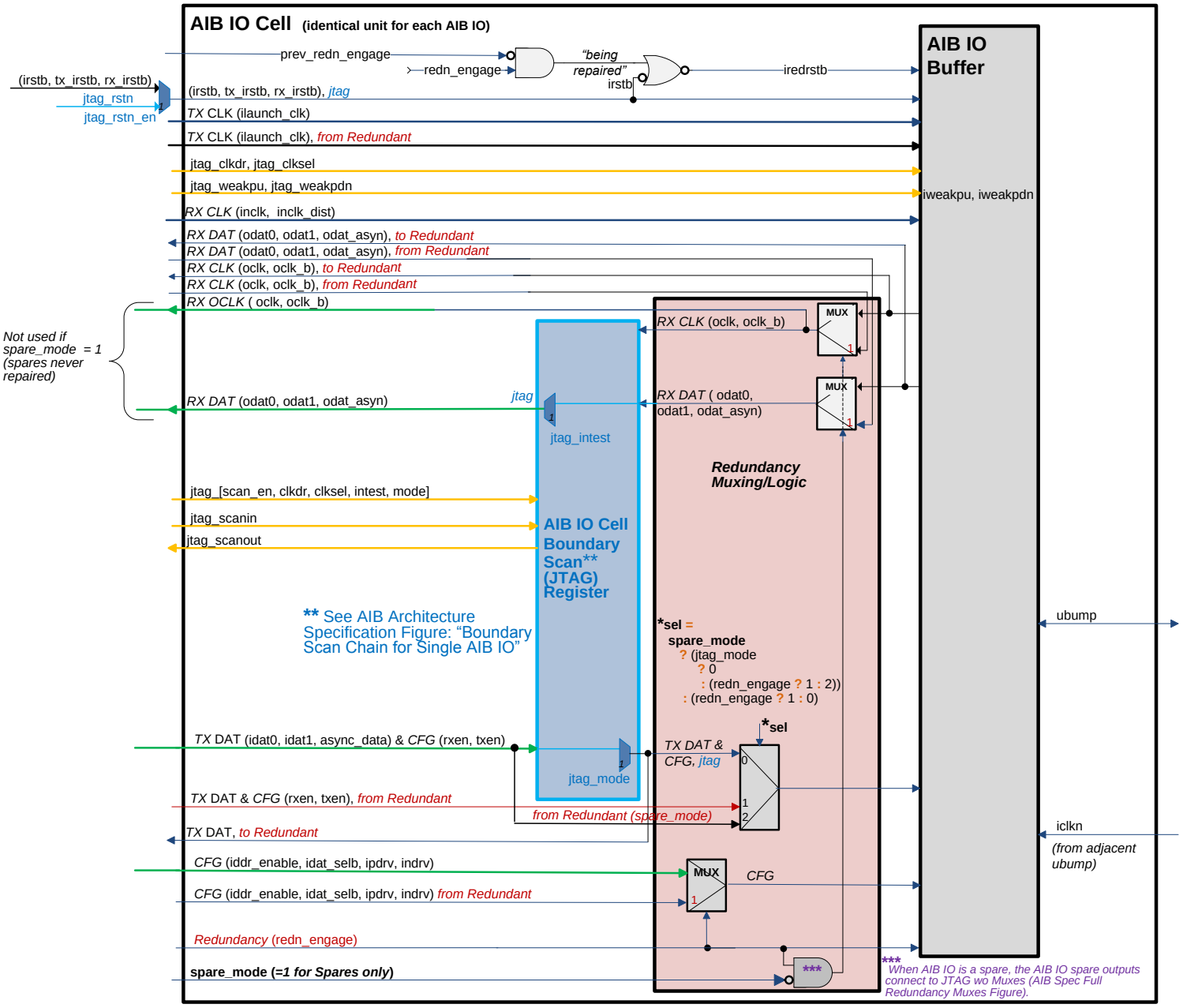
As indicated in the diagram, the Boundary Scan sub-block requires that *most* (but not all) signals to and from the AIB IO Buffer pass through an additional level of muxing (separate from the redundancy muxing) to allow for JTAG scan DFT stimulus and probing.

When an AIB IO Cell is used as a spare, the spare_mode input port is hardwired to a logic1. To support JTAG DFT together with redundancy repair of 2 separate sources, each spare AIB IO cell requires 3 sources (as depicted by the 3 to 1 MUX in the diagram) of TX data and configuration to the AIB IO Buffer, whereas non-spare AIB IO Cells only require 2 sources.

In addition to the redun_engage input port, the AIB IO cell also includes a prev_redun_engage port that connects to the redun_engage of the previous AIB IO cell in the redundancy chain. The two ports are logically combined to determine if the cell is the first (broken) cell in the redundancy chain. The broken ("being repaired") cell is tri-stated with weak pull-down enabled in order to avoid leakage and contention.

Also, the AIB Architecture Spec states: "if redundancy repair is not required, spare microbumps should be set to default power down mode." Hence, the pair of AIB IO cells that are used as spares are also tri-stated with weak pull-down enabled when redundancy repair is not engaged (i.e. when the redun_engage input port is all 0s).

Figure 5 AIB IO Cell Block Diagram



4.4. AIB IO Channel

The figure below (Figure 6 AIB IO Channel Block Diagram) depicts the AIB IO Channel, which instantiates an array of AIB IO Cell blocks, together with the associated RX data path redundancy MUX for each IO. In general, a DCC and DLL are also instantiated, but for the case of the TLRB AIB PHY they are excluded (grayed out text/border) since the TLRB AIB PHY implements an AIB Base configuration. Also excluded are the OSC clock pass-through buffer, and the replica clock tree for the DLL feedback.

A selected pair of the RX clock (oclk/oclk_b) outputs from the array of AIB IO Cells is connected to the inputs of the redundancy CLK MUX. The selected clock pair used to capture RX synchronous Data ubumps is specified by an RTL build parameter indicated by *RXD* and *RXDredn* (redundant RX data clock) indices in the diagram.

The clock inputs to the IO Cells are conditionally routed (as specified by an RTL build parameter) to each IO cell depending on RX/TX group partitioning. Conditional routing (RTL build parameters) of input clocks to the AIB IO cells is indicated by a small box superimposed on the respective clock signal. The total number of AIB IO Cells implemented per channel (**N_IOC** in the diagram) may also be specified via RTL build parameters.

Every IO cell provides an alternate (redundant) source of inputs such that it can take over for (repair) a defective link in another IO cell (in an adjacent pair). For instance, a defective AIB IO Cell [0] link is dedicated to be repaired by AIB IO Cell [2]. Since AIB IO Cells [0] and [1] themselves are located at the boundary of the AIB IO channel, they do not repair other IO Cells, and the alternate (redundancy) inputs are tied to 0s.

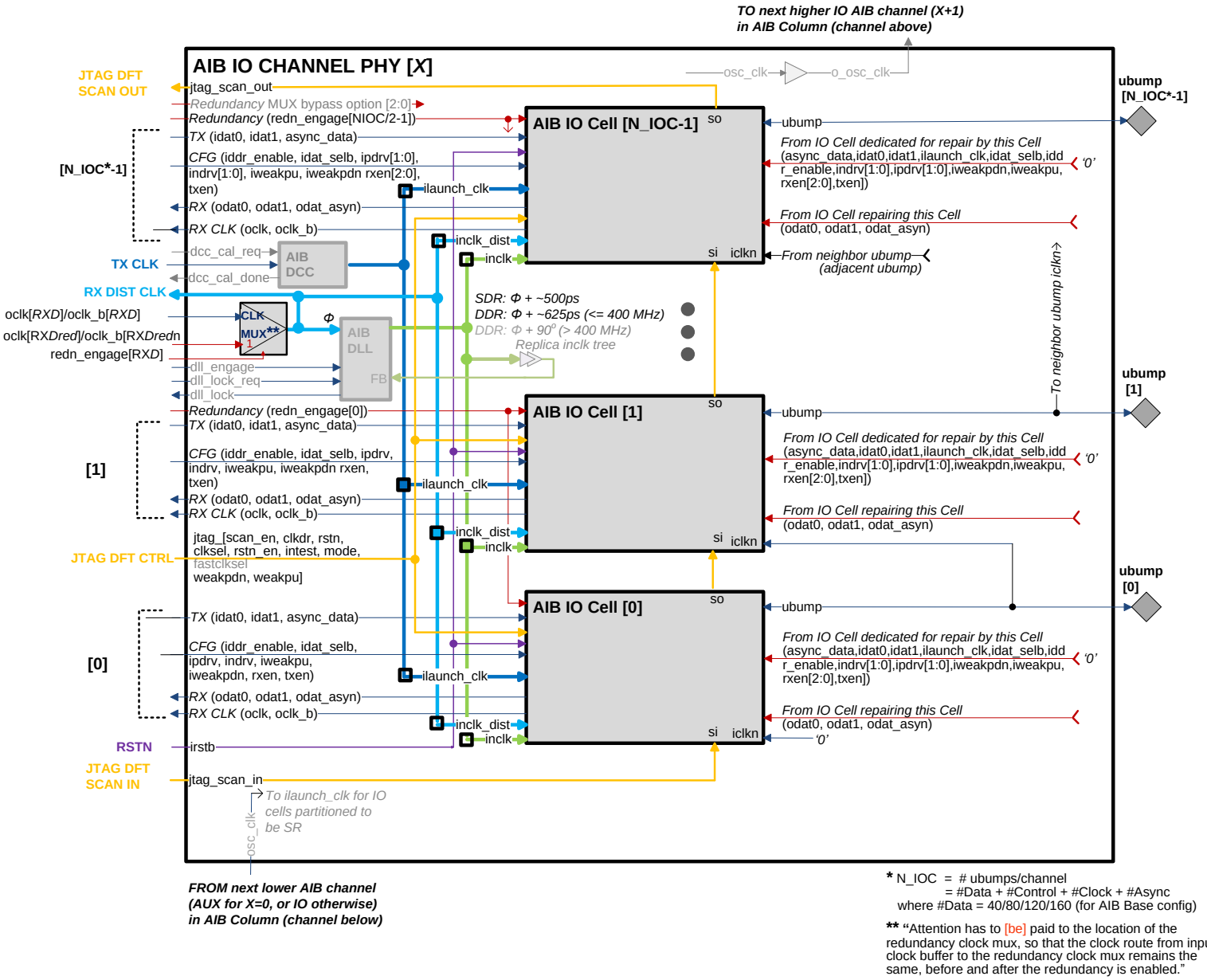


Figure 6 AIB IO Channel Block Diagram

4.5. AIB AUX Channel

The figure below (Figure 7 AIB AUX Channel Block Diagram) depicts the AIB AUX Channel, which instantiates just two AIB IO Cells for POR and Device Detect, respectively. The blocks with shaded text and borders including a JTAG TAP controller, a free running oscillator (OSC), and a non-volatile memory (in which is stored redundancy repair information) are *not* instantiated within the TLRB AIB PHY since it implements an AIB Base configuration. In general, however, the AIB architecture does require these blocks for some AIB implementation configurations (e.g. an AIB Base *Plus* configuration).

The POR and Device detect signals use “double microbumps” according to the AIB passive redundancy scheme.

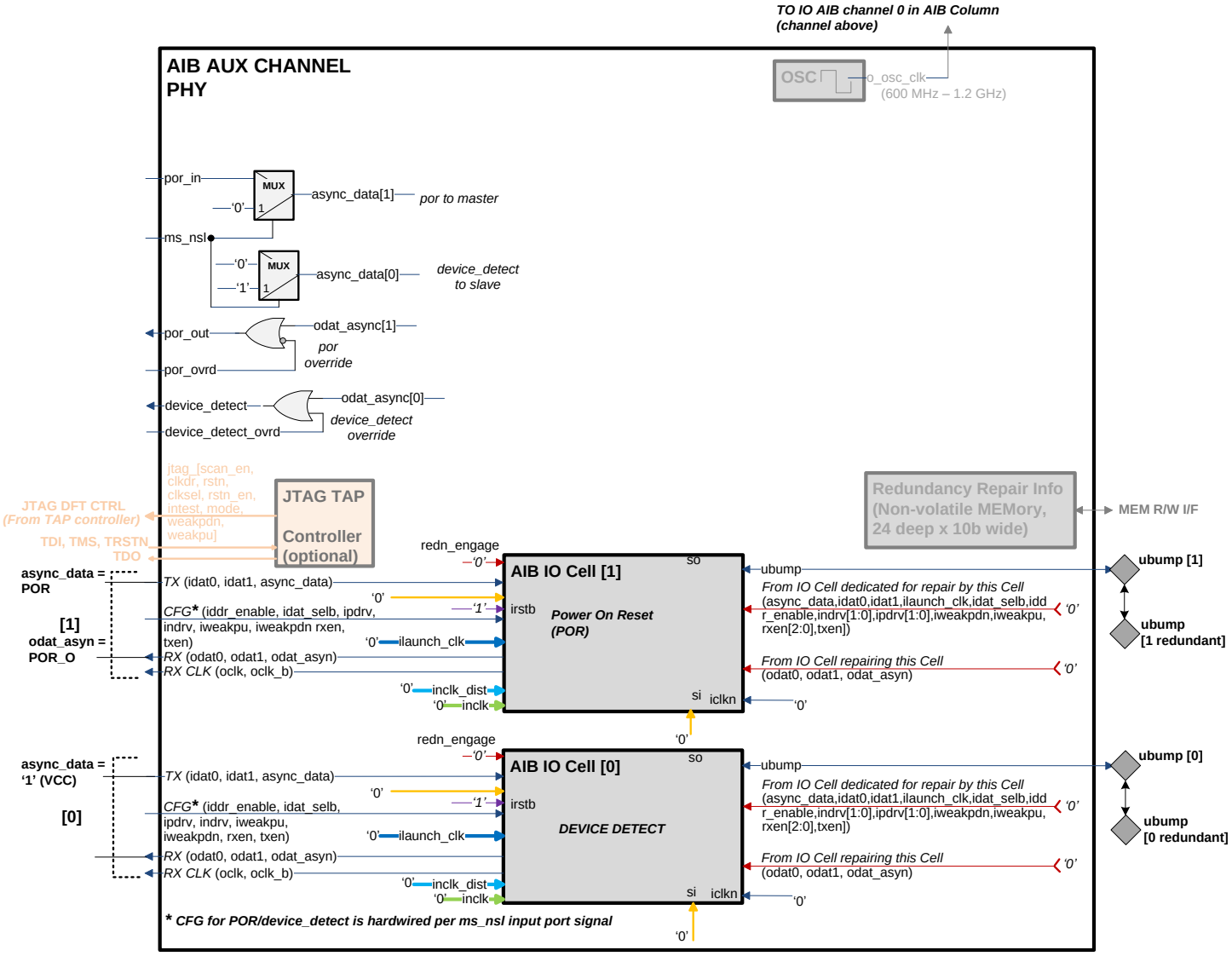


Figure 7 AIB AUX Channel Block Diagram

5. Digital I/O Ports

The digital I/O ports of the AIB TLRB PHY are enumerated and described in the table below (Table 11 Digital I/O Ports).

Table 11 Digital I/O Ports

Signal name	Direction	Description
		Synchronous Data (ND = # of synchronous Data uBumps; i.e. 80)
tx_data[*ND*-1:0]	Input	Synchronous (to tx_clk) data output transmitted to uBump This is the same as the "TX DATA" bus from the AIB Adapter to the AIB IOs as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
rx_data[*ND*-1:0]	Output	Synchronous (to rx_clk) data input received from uBump This is the same as the "RX DATA" bus from the AIB IOs to the AIB Adapter as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
tx_clk	Input	Transmit clock for synchronous tx_data
rx_clk	Output	Receive clock for synchronous rx_data
		Resets and Reset Control (asynchronous)
por_in *	Input	Slave: Input Power on reset (POR) (Driven by the slave from an external POR controller to the AIB AUX Channel.) Master: unused (tie low) 1=power on reset asserted
por_out *	Output	Slave: unused (=0) Master: Output POR (Monitored by the master in an external POR controller.) 1=power on reset asserted Driven by the AIB AUX Channel.
por_vcc_io *	Input	Power on reset VCC of IO power domain (to Analog IO Buffer, =0 for normal operation) 1=power on VCC reset asserted. Reset forces tristate and weak pull down in the AIB IO Buffers. Broadcasted to all AIB IOs during reset (POR stage) to avoid contention between chiplets. Drives the internal por_vcc_io input ports of all of the AIB IO Cells when the AIB is a slave. Unused when AIB is a master. When the AIB is a master, the por_vcc_io input ports of all the AIB IO Cells are otherwise driven by the **por_out** output port.
por_vcc_dig	Input	Power on reset VCC of digital logic power domain (to Analog IO Buffer, =0 for normal ops) 1=power on VCC reset asserted
device_detect *	Output	Device detect (Monitored by slave. Unused by master and driven to 0.) 1= Device detected (master present) Driven by the AIB AUX Channel.
adap_irstb	Input	AIB PHY reset IRSTB input. Resets *this* local AIB PHY by driving the internal irstb signal; active low
rstn_in	Input	AIB PHY reset input. (Source of ms_rstn signal driven to the uBump; active low)
rstn_out	Output	AIB PHY reset output (Derived from sl_rstn signal received from the uBump; active low)
adap_rstn_in	Input	Adapter reset input (Source of ms_adapter_rstn signal driven to the uBump; active low)
adap_rstn_out	Output	Adapter reset output (Derived from sl_adapter_rstn signal received from the uBump; active low)
		IO Cell Configuration (asynchronous/static). ***NB = # of ubumps per AIB IO Channel excluding the spare pair (i.e. 88)***
ms_nsl *	Input	Master Not Slave signal (hardwires the AUX channel POR/device_detect AIB IO cell configuration for master vs. slave mode) 0=Slave, 1=Master
iddr_enable	Input	DDR mode select (1=enable DDR mode)
idat_selb[*NB*-1:0]	Input	Asynchronous TX mode select for each AIB IO cell (1= select async_data **)
ipdrv[1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Data ubumps.
indrv[1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Data ubumps.
ipdrv_clk[1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Clock ubumps.
indrv_clk[1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Clock ubumps.
ipdrv_rst[1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Reset ubumps.

indrv_rst[1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Reset ubumps.
rxen[*NB*-1:0][2:0]	Input	Receive enable selection for each AIB IO cell (encoded per AIB spec tables)
txen[*NB*-1:0]	Input	Transmit enable for each AIB IO cell (1= TX enabled)
sdr_dly_adjust[ ]	Input	Adjustment setting for programmable RX inclk delay (DLL manual mode) for SDR mode
ddr_dly_adjust[ ]	Input	Adjustment setting for programmable RX inclk delay (DLL manual mode) for DDR mode
		Redundancy (asynchronous/static) ***TNB = Total # of ubumps per AIB IO Channel (i.e. 90)***
redun_engage [(*TNB/2)*-1: 0]	Input	Redundancy engage enable for Channel 0 for each *pair* of AIB IO Channel IO Buffers Selects source (1=redundant/0=normal) of clock, data, and configuration inputs to AIB IO Cell. Also selects the source of odat*outputs from the AIB IO Cell pair.
		JTAG/DFT
jtag_scan_en	Input	JTAG scan enable (1= enable scan shifting of the BSR chain). External to the TLRB, jtag_scan_en defaults to 0 upon power up.
jtag_clkdr	Input	JTAG data register clock. External to the TLRB, jtag_clkdr defaults to 0 upon power up.
jtag_rstn	Input	JTAG reset signal (active low) . External to the TLRB, jtag_rstn defaults to *1* upon power up.
jtag_rstn_en	Input	JTAG reset enable (1=enable/select the JTAG reset signal as the source of irstb). External to the TLRB, jtag_rstn_en defaults to 0 upon power up. Refer to the AIB Arch Spec JTAG "Reset Override Mode".
jtag_clksel	Input	JTAG clock select (1=select the JTAG data register clock). External to the TLRB, jtag_clksel defaults to 0 upon power up.
jtag_intest	Input	JTAG INTEST (internal test, e.g. probing). External to the TLRB, jtag_intest defaults to 0 upon power up.
jtag_mode	Input	JTAG mode (1 = select Boundary Scan Register source for outputs to AIB IO). External to the TLRB, jtag_mode defaults to 0 upon power up. Equivalent to the "EXTEST" function in a normal boundary scan.
jtag_weakpd	Input	JTAG weak pull down (1=enable weak pull-down)
jtag_weakpu	Input	JTAG weak pull up (1 = enable weak pull-up)
jtag_scan_in	Input	JTAG chain scan input data. External to the TLRB, jtag_scan_in defaults to 0 upon power up.
jtag_scan_out	Output	JTAG chain scan output data
device_detect_ovrd *	Input	DFT: Device Detect override (forces device_detect output high for wafer test) (1 = override; C4 at package level connected to logic low)
por_ovrd *	Input	DFT: POR override for wafer test (0 = override; C4 at package level connected to logic high)
dig_test_sel	Input	DFT: Selects source of debug signals driven to the dig_test_bus [] output from either the AUX channel (=0) or the IO channel (=1).
dig_test_bus [7:0]	Output	Debug signals from AIB IO and AUX Channels (designer defined signals , **TBD**)

* These AUX Channel signals as indicated above are VCC IO (vcc_io) level (nominally 0.9 V) signals. All other signals are VCC digital (vcc_dig) level (nominally 0.8 V) signals.

** Note that for a *Legacy* AIB configuration idat_selb = 0 selects async_data. The RTL code implementing the TLRB AIB PHY however sets a build parameter to configure the polarity of idat_selb such that idat_selb = 1 selects async_data, which is the polarity expected for an AIB *Base* configuration as specified by the AIB Architecture spec.

The only other logic ports of the TLRB AIB PHY not listed in the table above are the microbumps ports. The TLRB AIB PHY microbump I/O ports are routed directly to the placed microbumps (external to the AIB PHY block), and are directly driven and received by analog IO blocks within the TLRB AIB PHY. These ports are listed in Table 8 TLRB PHY AIB Base System Microbump Mapping (*TX* Data/Clock, Resets, Spares; 80 Data *IO* Chan) and in Table 9 TLRB PHY AIB Base System Microbump Mapping (*RX* Data/Clocks/Resets; *80* Data IO Channel).

5.1. Synch Data I/F Map to AIB DDR/SDR

The synchronous data adapter interface data busses (tx_data [], and rx_data []) as listed above (Digital I/O Ports) are mapped to the SDR and DDR ports of the AIB IO Buffers in a manner prescribed by the AIB Spec (newly specified in version 0.99).

The table below (Table 11 Mapping of Adapter Sync Data Bits to AIB IO DDR/SDR) depicts how each data bit of tx_data [] maps to a TX []/RX [] AIB uBump link pair, and then to a corresponding rx_data [] bit.

For the DDR case, where an even/odd numbered pair of bits maps to the same AIB pin name, the *even* numbered bit is transmitted first to the AIB link uBump (using the AIB IO Buffer idat0/odat0 ports). For example, tx_data[0] goes first on the TX[0]/RX[0] AIB link, followed by tx_data[1].

Note that ***for SDR mode, only the even numbered 40 bits of the tx_data [] and rx_data [] interface busses are used***. The odd numbers bits are to be driven to 0s for transmit and ignored for receive.

Table 11 Mapping of Adapter Sync Data Bits to AIB IO DDR/SDR

TX adapter data bits	TX AIB pin name	RX AIB pin name	RX adapter data bits
DDR (80 adapter interface bits)			
	AIB link		
tx_data[1: 0]	-> TX[0]	-> RX[0]	-> rx_data[1: 0]
tx_data[3: 2]	-> TX[1]	-> RX[1]	-> rx_data[3: 2]
tx_data[5: 4]	-> TX[2]	-> RX[2]	-> rx_data[5: 4]
…			
tx_data[79:78]	-> TX[39]	-> RX[39]	-> rx_data[79:78]
SDR (40 adapter interface bits)			
tx_data[0]	-> TX[0]	-> RX[0]	-> rx_data[0]
tx_data[2]	-> TX[1]	-> RX[1]	-> rx_data[2]
tx_data[4]	-> TX[2]	-> RX[2]	-> rx_data[4]
…			
tx_data[78]	-> TX[39]	-> RX[39]	-> rx_data[78]

6. Redundancy

The AIB architecture requires a pair of redundant (spare) AIB IOs per AIB IO channel to implement the active redundancy scheme. Active redundancy can repair just 1 faulty bit per AIB channel. The AIB architecture specifies a single spare pair of AIB IOs that are physically located in the middle of each channel for the purpose of active redundancy repair. The active redundancy scheme, when engaged, replaces a faulty pair of AIB IOs within the channel with its dedicated (physically adjacent) redundant pair. Each pair of AIB IOs within a channel is linked together in a redundancy chain. The chain begins with the AIB IO containing the faulty AIB IO, and ends with the redundant, spare AIB IO pair.

Associated with the active redundancy scheme is a "redundancy shift direction". The configuration inputs for the AIB IO pair containing *the* faulty bit within an AIB channel are shifted to a physically adjacent pair (dedicated redundant pair) in a direction towards the spare AIB IO pair (physically located in the middle of the channel layout). All intervening AIB IO pairs (pairs located between the faulty pair and the spare pair) likewise shift their configuration (and clock and data) inputs towards the spare pair when redundancy is engaged. The outputs from the AIB IO cells shift in the opposite direction (away from the redundancy engaged spare pair) towards the faulty pair such that the AIB IO cell outputs are always steered to the same higher level logical outputs whether redundancy is engaged or not.

In the AIBAUX section of the AIB architecture spec, the location of the spare (redundant) AIB IOs is now described to be exclusively within each AIB IO channel: (highlight added): "two spare microbumps reserved for redundancy repair are placed in each channel instead of AIBAUX."

The AIB architecture also requires redundancy repair information to be stored in non-volatile memory within the AUX channel. The memory is organized such that there are up to 24 addressable locations (corresponding to up to 24 channels implemented in an AIB column) where the repair info stored in each location is up to 10 bits wide (corresponding to up to 640 AIB IOs implemented per channel). However, the TLRB AIB PHY does not implement this memory within the AUX channel. Since the TLRB AIB PHY digital logic requires no interaction with a repair info memory that is located internal to the PHY, and since a repair info memory that is located external to the PHY facilitates possible migration to other technologies, this PHY implementation expects that the repair info memory is implemented elsewhere in the chiplet.

The figure below (Figure 8 Connections to Redundancy (spare) IO Cells) depicts the connections to the redundancy (spare) AIB IO cells.

AIB IO Cell Spare 0 can take over for *either* the SARSTN cell above in the RX group, *or* the MRST cell below in the TX group.

AIB IO Cell Spare 1 can take over for *either* the SRSTN cell above in the RX group, *or* the MARST cell below in the TX group.

At the boundary of the of the RX group the SRSTN and SARSTN cells are clocked by the RX clocks (the TX clock is tied-off to 0). The SRSTN and SARSTN cells may take over for other RX data cells in the RX group.

At the boundary of the of the TX group the MRSTN and MARSTN cells are clocked by the TX clock (the RX clocks are tied-off to 0s). The MRSTN and MARSTN cells may take over for other TX data cells in the TX group.

The spares themselves are never repaired. Spare cells receive no outputs (odat* signals) from other cells to be driven to the top-level output ports, and hence the redundant odat* inputs to the spare cells are tied-off to 0s. Also, the spare cells repair only reset cells, and hence the clock inputs to the spare cells are tied-off to 0s.

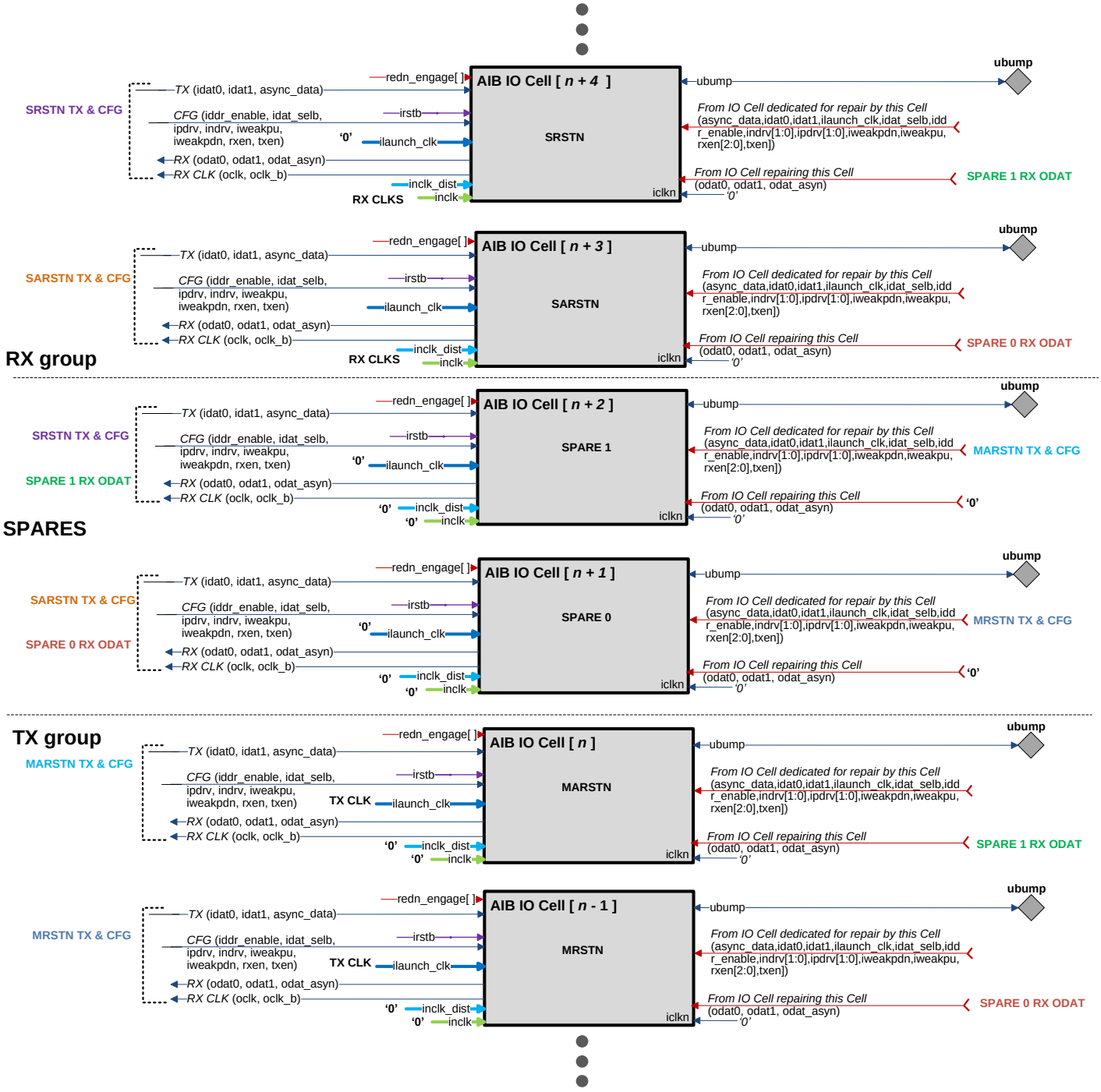


Figure 8 Connections to Redundancy (spare) IO Cells

7. Latency

Regarding latency, the AIB architecture specification states the following:

> "The latency for DDR/SDR Rx/Tx data signals is 1.5 Rx/Tx clock cycles each way plus the analog delay over the channel."

The TLRB AIB PHY meets this latency AIB specification such that the *total* delay of the path from the tx_data [ ] AIB PHY top-level port of the source chiplet to the rx_data [ ] AIB PHY top-level port of the destination chiplet is 3 cycles plus the channel delay.

The table below (Table 12 *Example* AIB PHY Synchronous Data Path Latency) breaks out the latency components for an example delay path from source chiplet to destination chiplet.

The table shows an example illustrating how the total path delay between chiplets always exceeds 3 cycles, and even could (architecturally) exceed 4 cycles. The analog delay over the channel includes the delay through the analog TX driver and through the analog RX receiver, and all the components added together could exceed a full clock cycle (depending on the UI length). Also, it is notable that the analog driver/receiver components of the latency vary over PVT, while the digital logic components do not.

Apart from the example latency analog delay values, the actual propagation delay values (**TBD**) of the analog driver/receiver blocks in the TLRB AIB PHY are required to compute the actual latency of an AIB link that uses the TLRB AIB PHY.

Table 12 *Example* AIB PHY Synchronous Data Path Latency

Latency component	Latency value (slowest PVT, and routing)	Latency value (fastest PVT, and routing)	Note	Category	
TX idat0/idat1 DFFs	1 cycle	1 cycle	TX staging DFFs	TX Digital logic	
TX ilaunch_clk MUX	½ cycle	½ cycle	DDR/SDR clock MUX		
TX Analog ubump driver	700 ps	350 ps	Example estimated values of the delay of the TLRB AIB PHY TX analog driver	TX Analog Driver	**analog delay over the channel**
ubump to ubump interposer channel routing	50 ps	30 ps	Estimate for 3-5 mm (LR) at 10 ps/mm (AIB Arch Spec typical interposer routing delay)	AIB Interposer routing	
Analog RX ubump receiver	300 ps	200 ps	Example estimated values of the delay of the TLRB AIB PHY RX analog receiver	RX Analog Receiver	
RX inclk DFFs	½ cycle	½ cycle	RX strobe DFFs	RX Digital logic	
RX odat0/odat1 DFFs	1 cycle	1 cycle	RX retime DFFs		
Total	**4.05 cycles**	**3.57 cycles**	Example values (estimated) **at 1 GHz**		

8. Bandwidth

The *raw* (e.g. without CPI or AXI protocol overhead) TX/RX synchronous bandwidth provided by a TLRB AIB PHY (in G bits per second) equals the clock frequency multiplied by *half* the number of synchronous Data ubumps (multiplied by 2 if DDR mode).

Maximum AIB Base configuration raw Bandwidth at 1 GHz (single direction):

SDR: 1 GHz ***x*** 80/2 uBumps = 40 Gbps (5 GBps)
DDR: 400 MHz ***x*** 80/2 uBumps ***x*** 2 = 32 Gbps (4 GBps)

For the maximum bandwidth case, the DDR bandwidth is less than the SDR bandwidth because an AIB Base configuration limits/restricts the maximum DDR clock frequency to 400 MHz for DDR (vs. 1 GHz for SDR).

TLRB AIB PHY raw Bandwidth (single direction) at 250 MHz:

SDR: 250 MHz ***x*** 80/2 uBumps = 10 Gbps (1.25 GBps)
DDR: 250 MHz ***x*** 80/2 uBumps ***x*** 2 = 20 Gbps (2.50 GBps)

The table below (Table 13 Bandwidth per Use Case Comparison) justifies an 80 microbump implementation (versus a minimum 40 ubump implementation allowed by the architecture spec) by suggesting the potential to use the TLRB AIB PHY in higher bandwidth applications (e.g. PCIe) than required by the TLRB application.

Table 13 Bandwidth per Use Case Comparison

Use case	AIB data bps	Data ubumps (TX+RX)	Total BW (single-direction) [G Bytes/sec]	Required TLRB BW [G Bytes/sec]	Comment
TLRB AIB	500 Mbps @ 250 MHz DDR	40	1.25 GBps	0.5 to 1 GBps	Tight. Depends on AXI/CPI/AIB efficiency
TLRB AIB	500 Mbps @ 250 MHz DDR	80	2.50 GBps	0.5 to 1 GBps	Good.
Max AIB Base	1Gbps @ 1 GHz SDR	40	2.50 GBps	4 GBps	Will restrict PCIe chiplet usage
Max AIB Base	1Gbps @ 1 GHz SDR	80	5 GBps	4 GBps	Probably OK.
Max AIB Base+	2Gbps @ 1 GHz DDR	40	5 GBps	4 GBps	Not POR for TLRB AIB PHY.
Max AIB Base+	2Gbps @ 1 GHz DDR	80	10 GBps	4 GBps	Good. Not POR for TLRB AIB PHY.

9. Timing

In addition to specifying digital logic behavior, the AIB architecture specification also specifies a number of timing constraints required for the logic delay paths, which need to be considered for an AIB PHY implementation.

The AIB clock forwarding architecture specifies a **source synchronous** interface to transfer (register to register) synchronous data type signals between chiplets. The source synchronous interface forwards a clock with the data from the output of one chiplet to the input of another chiplet.

Relative to source synchronous interface type variations, the AIB architecture specifies that:

- the forwarded clock is **edge-aligned** to the data
- the **same edge** of the clock both launches and captures the data
    - The same forwarded clock edge that launches the data also captures the data.
    - Timing paths across the AIB link are therefore always from:
        - *falling* clock edge *to falling* clock edge (SDR and DDR), or
        - *rising* clock edge *to rising* clock edge (DDR)

9.1. SDR mode AIB Data Microbump Timing

The diagram below (Figure 9 Near-End TX SDR Data Timing) illustrates the Near-End timing at the TX output of a chiplet for 1 GHz SDR mode. The shaded data labeled "e0" is launched on the *falling* edge of the TXCLK Near-End microbump to be captured by same the *falling* edge.

The launched data at the Near-End microbumps has **-**20 ps of **setup time** with respect to the output clock. The setup time provided at the output ubumps of the interface is negative since the leading edge of valid/stable data may change up to (at most) 20 ps *after* the capture clock.

The launched data at the Near-End microbumps has 980 ps (less the jitter to the next falling clock edge) of **hold time** with respect to the output clock. The data remains stable until at least 980 ps (less the jitter) after the capture edge when viewed at the microbump TX clock and data outputs (with Near-End eye mask loading conditions).

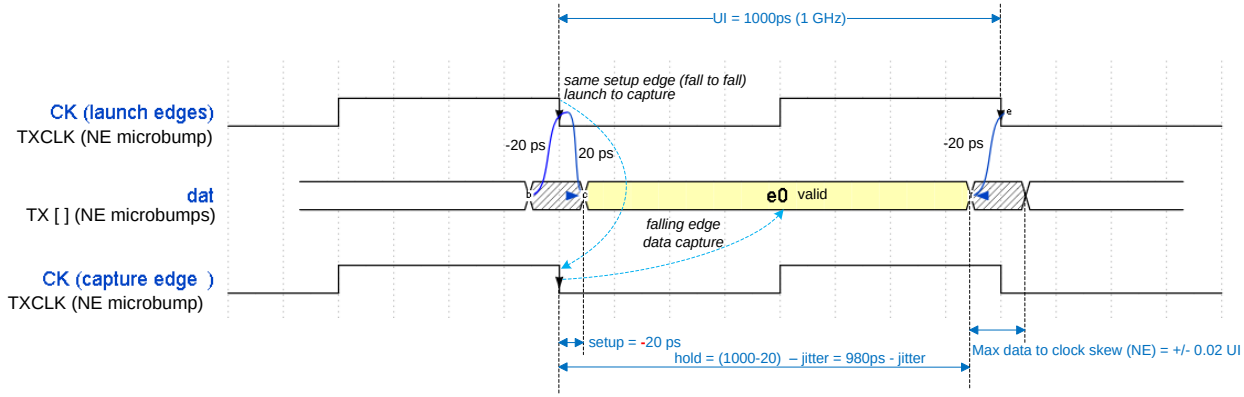


Figure 9 Near-End TX SDR Data Timing

The AIB spec requirement of 0.02 UI data to clock skew appears to be very aggressive considering technology limitations. For example, the spec requires a TLRB AIB PHY at 1 GHz (SDR mode) implementation to not vary the delay through **40** separate analog ubump TX drivers relative to the TX clock drivers by more than 20 ps.

The aggressive data to clock skew architectural requirement drives the implementation of a special TX ilaunch_clk distribution carefully routed to all of the analog TX ubump drivers to limit the skew. The TX data path clock includes the AIB IO Buffer data/clock path digital MUX logic (as well as the JTAG and redundancy MUX TX clock digital logic). The digital logic in the TX clock path considered for timing to the ubumps includes only that logic (multiplexers) that is in the critical TX clock path.

The ilaunch_clk is still required to be distributed to the DFFs within the AIB IO Cell blocks, but with less critical timing since the TX *next* output is architecturally staged and stable in the DFFs by half a cycle prior to being selected by the output TX MUX in the critical path. Timing must also consider that the *previous* data input to the MUX is changed coincident with the output TX MUX selection change.

9.2. DDR mode AIB Data Microbump Timing

The diagram below (Figure 10 Near-End TX DDR Data Timing) illustrates the Near-End timing at the TX output of a chiplet for 400 MHz DDR mode. The shaded data labeled “e0” is launched on the *falling* edge of the TXCLK Near-End microbump to be captured by same the *falling* edge. The shaded data labeled “o0” is launched on the *rising* edge of the TXCLK Near-End microbump to be captured by same the *rising* edge.

The “e0”,”o0” labeling is borrowed from the “DDR Transmit Timing Diagram” found in the AIB architecture specification.

The launched data at the Near-End microbumps has -25 ps of **setup time** with respect to the output clock. The setup time provided at the output ubumps of the interface is negative since the leading edge of valid/stable data may change up to (at most) 25 ps *after* the capture clock.

The launched data at the Near-End microbumps has 1225 ps (less the jitter *and* the duty cycle distortion to the next clock edge) of **hold time** with respect to the output clock. The data remains stable until at least 1225 ps (less the jitter & DCD) after the capture edge when viewed at the microbump TX clock and data outputs (with Near-End eye mask loading conditions).

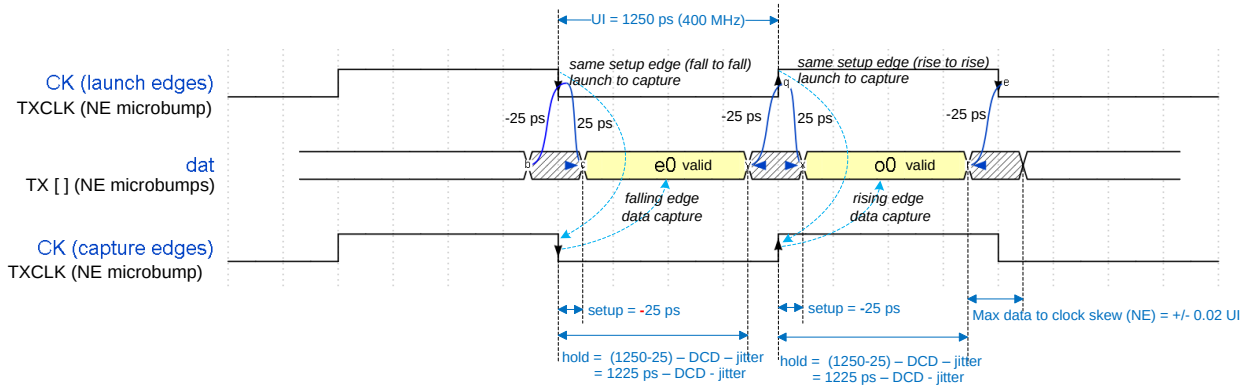


Figure 10 Near-End TX DDR Data Timing

9.3. DLL (manual programmable delay)

Since the TLRB AIB PHY implements an AIB Base configuration, the function of the DLL block is limited (restricted) to a manual programmable delay. Synchronous clock and data are edge-aligned when received at the microbumps. The received clock edge which launched the corresponding data is then sufficiently delayed within the TLRB AIB PHY such that the same clock edge which launched that data may reliably (with margin) be used to capture/strobe the synchronous ubump data into the first stage AIB IO Buffer RX DFFs.

The AIB specification recommends implementation of a programmable delay element (of about 500 ps delay in SDR mode). While the delay variation of a standard cell based delay line across PVT is significant, the variation of any given piece of hardware over voltage and temperature is much lower. The performance of any chip can be monitored with a process monitor (not included in the AIB PHY). A look-up table can be generated that provides the appropriate delay programming code for the process. The variation of this programmed delay is anticipated to be acceptable to operating the RX at the maximum specified clock rate.

The figure below (Figure 11 Strobe inclk clock delay) depicts how the delay element receives the RX clock (phase φ) from the microbumps (oclk) and drives a delayed version of the RX clock to the inclk balanced clock tree. The clock tree (inclk) strobes synchronous data from the microbumps into the first stage capture DFFs within the AIB IO Buffers. The delay adjustment configuration input ports (sdr_dly_adjust [] and ddr_dly_adjust []) provide the ability to tweak the delay programmable delay. The adjustment value used depends on the iddr_enable configuration input port.

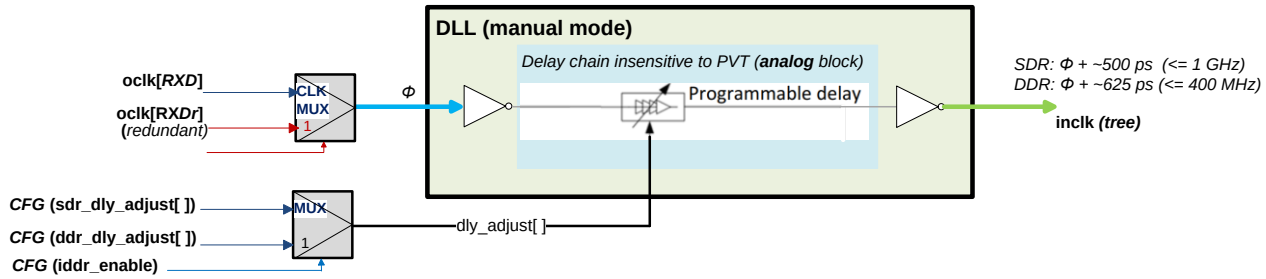


Figure 11 Strobe inclk clock delay

9.3.1. DLL SDR Mode

SDR mode supports clock rates up to 1 GHz (1 Gbps/ubump).

A constant value, manual programmable delay is targeted to be approximately 500 ps for all clock rates up to 1GHz. The value of about 500 ps corresponds to half a cycle of delay in order to center the rising edge of the clock in the middle of the data eye.

9.3.2. DLL DDR Mode

DDR mode supports clock rates up to 400 MHz (800 Mbps/ubump).

A constant value, manual programmable delay is targeted to be approximately 625 ps for all clock rates up to 400 MHz. The value of about 625 ps corresponds to a quarter cycle of delay in order to center both the rising and falling edges of the clock in the middle of the data eye.

Regarding the programmable delay setting, the AIB architecture spec states the following in the section headed "DLL High Level Description" (highlight added):

> "At lower clock rates, the DLL is not enabled so that the programmable delay only needs to be targeted for 400 MHz or higher clock frequencies. The appropriate default setting for the programmable delay has to be selected for all operations with a clock rate less than 400 MHz for DDR mode."

9.4. Delay Matching constraints

All data paths for synchronous data between the AIB IO Cells and the microbumps are required to be matched and minimized. The RX synchronous data paths include the redundancy multiplexers. These constraints require the TLRB AIB PHY implementation to include the matched routed lengths between the microbumps and the TLRB AIB PHY analog block IO microbump ports (length values **TBD**).

- Constrain the matched, minimized delays (for both min and max conditions) from the TX clock port (tx_clk) to the all of the TX data microbumps (including redundant) to meet the near end (NE) compliance eye mask requirements
  - The delay values specified take into consideration the input jitter and DCD on the application tx_clk, and the clock tree skew, jitter, and DCD contributed by the AIB PHY.
- Constrain the matched, minimized delays (for both min and max conditions) from all of the RX data microbumps (including redundant) to meet the setup and hold time of the first stage (strobe) DFFs in the AIB IO buffers.
  - The delay values specified take into consideration the characteristics of the input RX clock and data as specified by the AIB architecture specification far end (FE) compliance eye mask.
- Constrain the matched, minimized delays (for both min and max conditions) from the two differential RX clock microbumps (and the redundant set of RX clock microbumps) to the root of the RX clock tree (rx_clk port). The root of the RX clock tree is driven by the redundancy RX clocks MUX gate as in depicted a diagram in the clocks section (Figure 14 RX clocks).

The delay values of the constraints described above are technology, layout, and performance requirement dependent, and are **TBD**.

9.5. AIB IO Buffer inclk to inclk_dist Timing

The AIB architecture specification requires **2** RX clock inputs to each AIB IO Buffer block (inclk and inclk_dist). As such, there exist timing paths internal to each of the identical AIB IO Buffers between the two clocks (from inclk to inclk_dist).

The RX strobe clock (inclk) is specified to be delayed by ~500 ps in SDR mode while the RX distribution clock (inclk_dist) is routed without any added delay.

Referencing the timing diagram below, the setup time requirement for the retimed DFFs (inclk_dist) that drive odat[1:0] is given by:

Tsu = **T**period – (**T**delay + **T**co) **>** **T***setup* of the standard cell DFF

(where **T**delay and **T**co are late values)

Likewise, the hold time requirement is given by:

Thold = **T**delay + **T**co **>** **T***hold* of the standard cell DFF

(where **T**delay and **T**co are early values)

Tdelay is the fixed offset of inclk relative to inclk_dist. **T**delay equals the sum of the manually programmed ~500 ps delay and the delay difference between inclk clock tree distribution delay and the inclk_dist clock tree distribution delay (i.e. inclk tree delay minus inclk_dist tree delay).

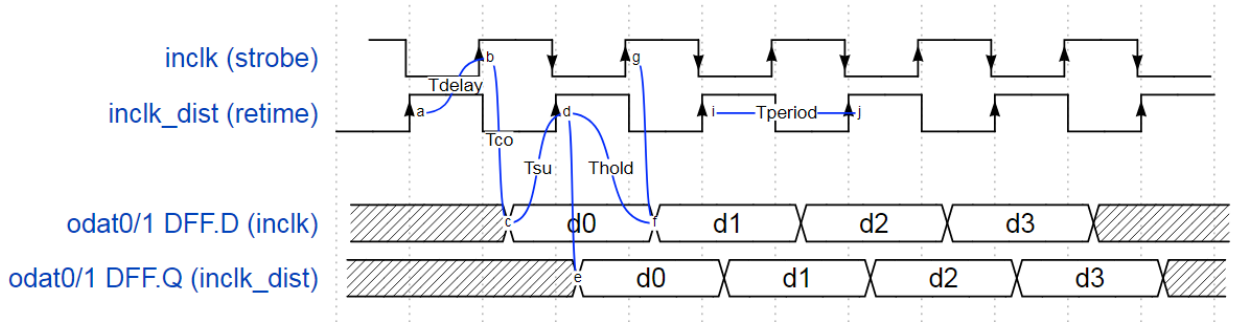


Figure 12 Timing Diagram (inclk to inclk_dist path internal to AIB IO cells)

10. Clocks and Resets

10.1.1. TX Clock

The TX clock launches the synchronous TX data from the AIB IO cells to the microbumps. The TX clock domain also includes the synchronous TX data application interface of the TLRB AIB PHY. Referencing the Figure 13 TX **clock** below, the tx_clk input port of the TLRB AIB PHY is driven from a source external to the AIB PHY within the chiplet. The total duty cycle distortion (DCD) allowed is +/- 3% (at the ubumps). The target for timing closure of the tx_clk is 1 GHz; however the TLRB application is expected to operate at about 250 MHz.

The duty cycle correction (DCC) block (grayed-out in the figure) is not required for an AIB Base configuration, and in the case of the TLRB AIB PHY the tx_clk passes through without modification.

The TX clock is distributed via a balanced, low skew clock tree to a selected TX group of AIB IO cells. The selected TX group of AIB IO cells is configured via an RTL build parameter. The TX group excludes the pair of spare AIB IOs to be used by the active redundancy scheme since a pair of reset TX AIB IOs at the group boundary serve as redundant cells for the TX clocked data group. Those cells not in the TX group tie-off their ilaunch_clk inputs to save power and limit clock skew. The TLRB AIB PHY does not implement a TX SR clock domain since it uses an AIB Base configuration, but in the case of a Base Plus configuration the ilaunch_clk of those AIB IO cells assigned to a separate TX SR group could likewise be connected to a clock derived from OSC_CLK via a build parameter.

The TX clock is also used to synchronously transmit data (tx_data [ ]) from the application to the PHY. The tx_data is captured by the PHY on the rising edge of every tx_clk where the PHY specifies the required setup and hold time required (values **TBD**).

The near end (NE) compliance eye mask specifies the timing requirements for the synchronous clock and data paths to the TX ubumps. Factors external to (outside of) the TLRB AIB PHY block contribute to TX eye mask compliance. These factors include the routing between the PHY and the ubumps, the jitter and duty cycle of the input tx_clk, and power supply noise. These factors (values **TBD**) need to be specified for the TLRB AIB PHY implementation to verify TX mask compliance.

The table below lists the required specifications for the tx_clk input signal (**TBD,** e.g. Intel AIB DLL specs).

Table 14 tx_clk Specifications

tx_clk Parameter	Specification (TBD)
frequency (min)	125 MHz
frequency (max, DDR)	400 MHz
frequency (max, SDR)	1 GHz
DCD	+/- 2%
jitter	18 ps

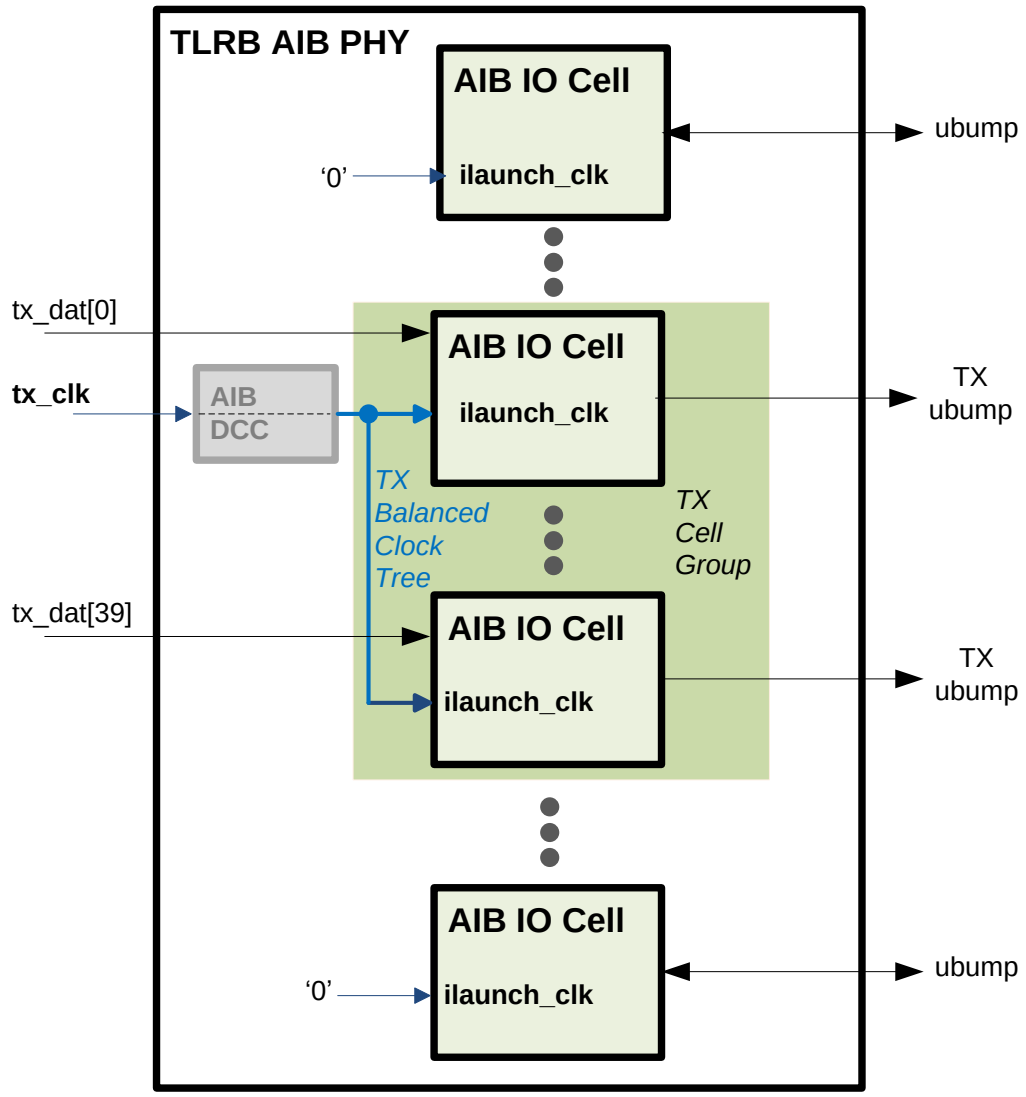


Figure 13 TX clock

DFT Note:

Those AIB IO Cells that are not in the TX Cell group do not receive an ilaunch_clk input signal. The ilaunch_clk input is tied off to logic 0 for these cells. However, the cells outside of the TX Cell are still capable of clocking TX output data to the uBumps via the JTAG clock in DFT mode. For DFT purposes, the iddr_enable (IDDREN) inputs of the AIB Cells not in the TX Cell group are all tied to logic 1. This hardwired IDDREN setting (=1) allows synchronous RX data cells to be wafer tested in DFT mode using synchronous DDR (and SDR) JTAG loopback settings.

10.1.2. RX Clocks

The source of the RX clocks is from a single selected pair of microbumps. There is a corresponding selected pair of adjacent AIB IO Cell blocks (where the iclkp cell of the pair is configured to be a pseudo differential ended receiver while the iclkn cell of the pair is unused) in the RX group of AIB IO cells. The pair of AIB IO cells used for the source of RX clocks is determined by a build parameter in the RTL. The same build parameter also determines the adjacent pair to be used as the redundant source of RX clocks.

Referencing the figure below (Figure 14 RX clocks), there are two RX clocks generated:

- **inclk** (also called strobe clock or capture clock)
- **rx_clk** (TLRB AIB PHY RX interface clock top-level output port; also called inclk_dist, or distribution clock)

The inclk RX clock captures (strobes) the synchronous RX data from the microbumps into DFFs within the AIB IO cells. The RX data captured in the inclk DFFs is transferred (internal to the AIB IO Buffer block) to the application re-timed DFFs (clocked by rx_clk).

Both RX clocks are distributed via a balanced, low skew clock trees to the selected RX data group of AIB IO cells. The selected RX group of AIB IO cells is configured via an RTL build parameter. The RX group excludes the pair of spare AIB IOs to be used by the active redundancy scheme since a pair of reset RX AIB IOs at the group boundary serve as redundant cells for the RX clocked data group. The inclk and inclk_dist ports of all of the AIB IO cells not in the RX group are tied-off to minimize the skew and power of the RX clock trees.

The root of the RX inclk clock is delayed (offset) relative to the root of the rx_clk clock tree by about 500 ps (SDR mode) using a feature limited DLL block implementing only a manual delay mode. The actual delay falls in a range around 500ps. The extent of the range over which timing can be closed depends upon factors such as the input Far End (FE) clock to data skew, the internal clock skew, the routing from the RX clock/data ubumps, the standard cell technology DFF setup/hold timing specs, the required operating conditions, etc.

For integration with the RX application logic, the TLRB AIB PHY specifies the minimum and maximum clock to output time (**T**co) for the rx_data [] port with respect to the rising edge of rx_clk input port (values **TBD**).

As also represented in the figure below, the redundant oclk and oclk_b inputs to each AIB IO Cell (labeled with “redn”) are tied low (logic 0) *except for* the AIB IO Cell that drives oclk and oclk_b from the RX CLK and RX CLKB uBumps. The connections to the redundant oclk and oclk_b inputs are limited in order to minimize the skew and power of the rx_clk tree. Consequently, the JTAG BSR registers (AIB Spec Figure 5-4) that capture OCLK_AIB into RX_REG[1] and OCLKB_AIB into RX_REG[2] will read back as 0s for all AIB IO cells for which redundancy is engaged except for the actual RX CLK cell (i.e. AIB ID 39).

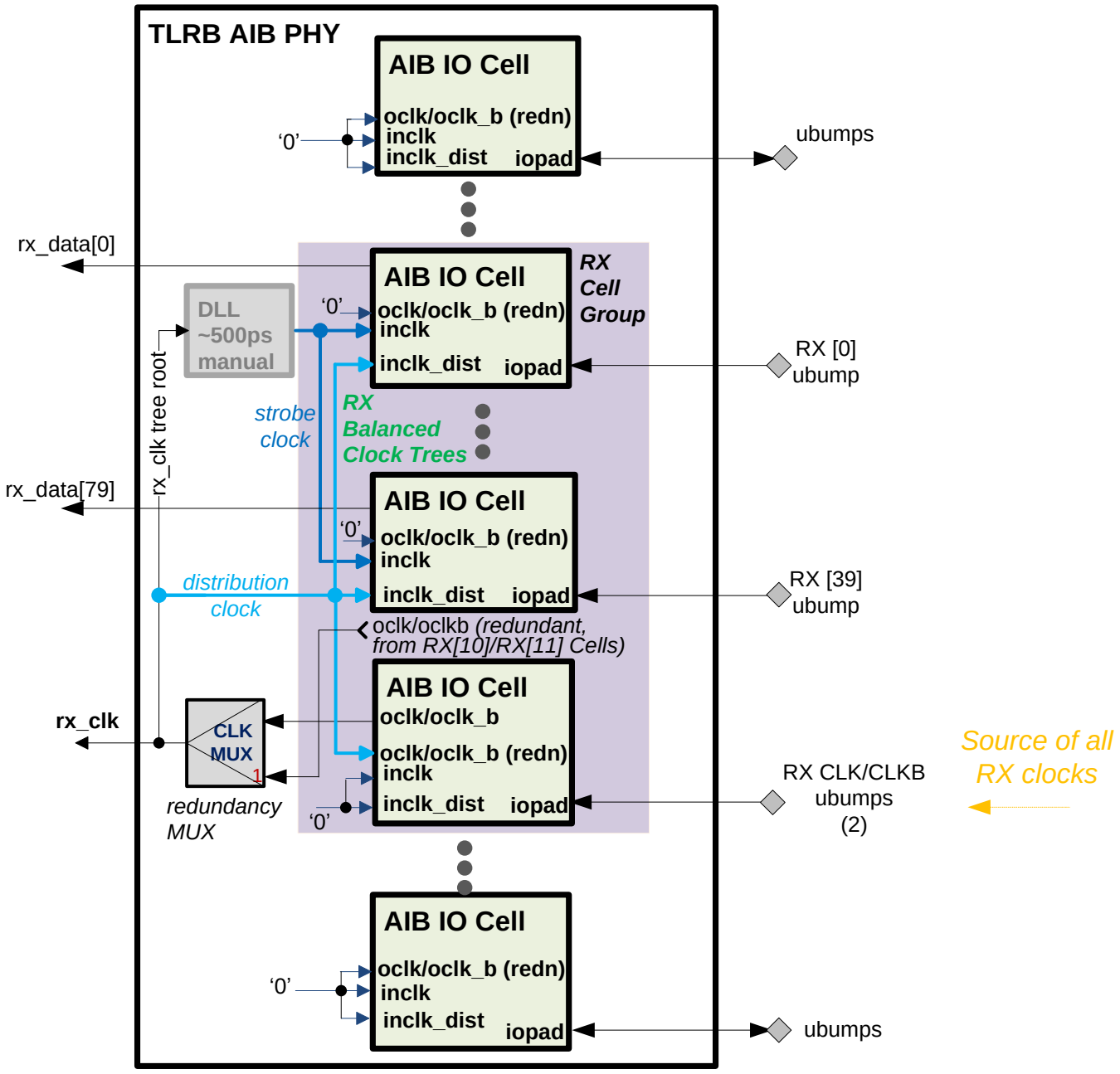


Figure 14 RX clocks

10.1.3. Resets

The AIB IO cells are held in reset by the internal irstb signal (active low). The assertion of the irstb signal (logic 0) asynchronously tristates and weakly pulls-down the microbump signals. The negation of the irstb signal (logic 1) asynchronously enables the AIB IO TX drivers (if configured for TX), and disables the microbump pullups/pulldowns.

The figure below (Figure 15 Reset Logic Diagram) depicts TLRB AIB PHY reset logic. A reset controller, external to the TLRB AIB PHY, can reset the local chiplet AIB PHY (via rstn_in), or reset the remote chiplet AIB PHY (via adap_rstn_in). The remote chiplet AIB PHY is at other end of the AIB link. The reset controller can also monitor the reset signals from the remote AIB PHY. Depending upon whether the AIB PHY is a master or a slave (as indicated by the ms_nsl top-level input port configuration signal), the reset signals get mapped to different reset microbumps as listed in (Table 11 Digital I/O Ports), and depicted in the figure.

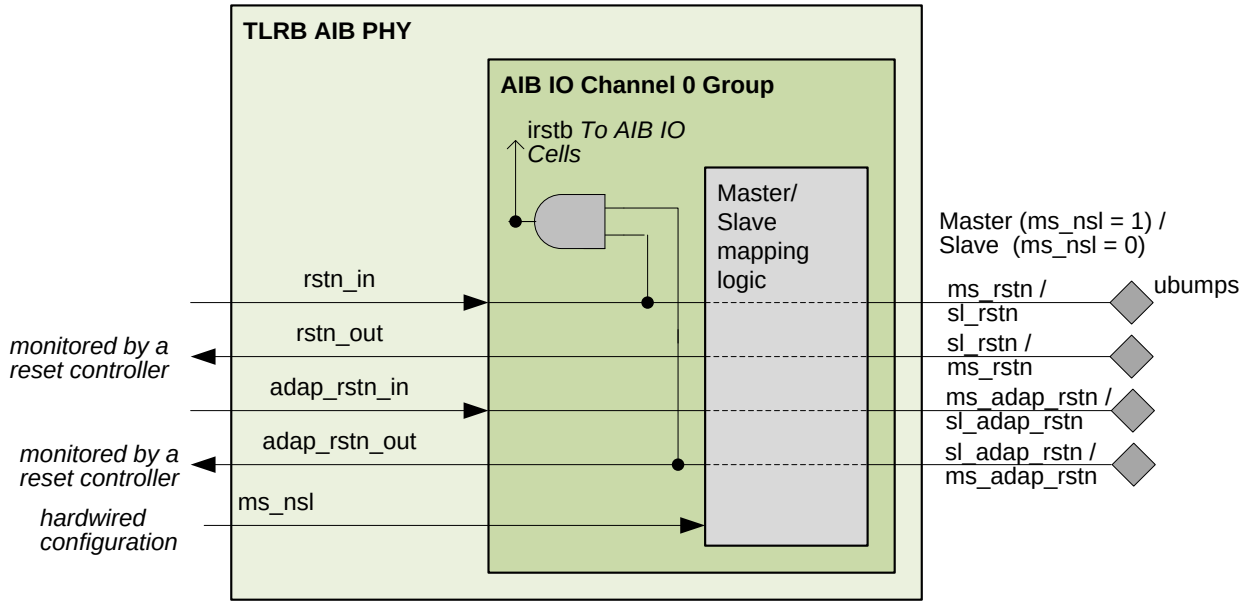


Figure 15 Reset Logic Diagram

Internal to the TLRB AIB PHY the resets are asserted asynchronously and negated (released) synchronously relative to the applicable clocks for DFFs (and latches) within the AIB Cell Buffer blocks.

At chiplet power up the resets are asserted, and all microbumps are weakly pulled low. Prior to a reset controller releasing resets (rstn_in/adap_rstn_in) to the AIB TLRB PHY a sequence of events including device_detect, POR, and configuration is first required.

10.1.4. Power On Reset (POR)

The TLRB AIB PHY behaves differently with respect to the POR and device_detect top-level I/O ports depending upon whether it is configured to be a master or a slave.

The ms_nsl input port (configuring the TLRB to be either a master or slave) is required to be set prior to power on reset of the TLRB AIB link to implement the POR protocol as it is described in the AIB Architecture specification. In this sense, ms_nsl is hardwired and static prior to power on reset.

TLRB AIB PHY configured to be a Master

The AIB master monitors (within a master POR controller external to TLRB AIB PHY) the slave-POR (por_out) port signal generated from the slave chiplet as conveyed over the AUX channel. The master POR controller monitors this remote slave-POR signal along with local master chiplet signals (PORs and config_done) to determine configuration status. The monitored remote slave-POR signal is used to generate the por_vcc_io and por_vcc_dig input port signals which force all the AIB IOs into tristate mode with enabled weak pull down to avoid contention.

Also, the AIB master generates (internal to the AUX channel) a device detect signal to send over the AIB link to the slave. The device detect signal (active high) indicates to the slave that the master is powered on and is present.

TLRB AIB PHY configured to be a Slave

The AIB slave monitors (within a POR controller external to TLRB AIB PHY) the device_detect output port signal. The device_detect signal is generated from the master chiplet, and is conveyed over the AUX channel to indicate that the master is present. The slave POR controller combines the device_detect signal from the TLRB AIB PHY with local slave chiplet signals (PORs and config_done). The combined control signal is used to generate the por_vcc_io and por_vcc_dig input port signals which force all the AIB IOs into tristate mode with enabled weak pull down to avoid contention.

Additionally, the AIB slave monitors the device_detect signal for the purposes of generating (within a POR controller) the por_in input port signal to the TLRB AIB PHY to be conveyed to the AIB slave over the AUX channel.

11. DFT

The TLRB AIB PHY supports the DFT features specified by the AIB architecture specification for an AIB Base configuration.

TLRB AIB PHY DFT features:

- AIB Boundary Scan
  - *simplified* (as specified in the AIB Architecture Spec) version of the JTAG (IEEE 1149.1) standard to support AIB IO testing

- AIB IO Leakage Test
  - weak (10-20 K ohm) pull-up and pull-down controllable via the boundary scan register (BSR)

- Stuck At Test
  - JTAG testing of AIB IOs using CLKDR.
    High level sequence:
    1. scan input vector into TX BSR
    2. propagate BSR to AIB IOs
    3. scan output vector from RX BSR

- Digital Test Bus

- Input ports override control for device detect and POR.

Test features related to unimplemented AIB components such as the DLL, DCC, Adapter, OSC, and external reset controllers are excluded from the DFT feature list above since the TLRB AIB PHY implements an AIB Base configuration.

11.1. Digital Test Bus

The AIB architecture specification requires a digital test bus for DFT (debug). The signals to be probed (probe-able at wafer test since routed to C4 bumps) using the test bus are *determined by the designers* (intended for in-house debug use only). The AIB architecture spec suggests some DLL, DCC, and OSC signals, but since the TLRB AIB PHY implements a base AIB configuration (which excludes DLL, DCC, and OSC) this suggested set of signals does not apply.

The figure below (Figure 16 TLRB AIB PHY Digital Test Bus (for DFT) illustrates how the TLRB AIB PHY uses the digital test bus.

AIB AUX channel digital test bus signal list and mapping: **TBD**

AIB IO channel digital test bus signal list and mapping: **TBD**

The test bus C4s do not need to be routed to package pins.

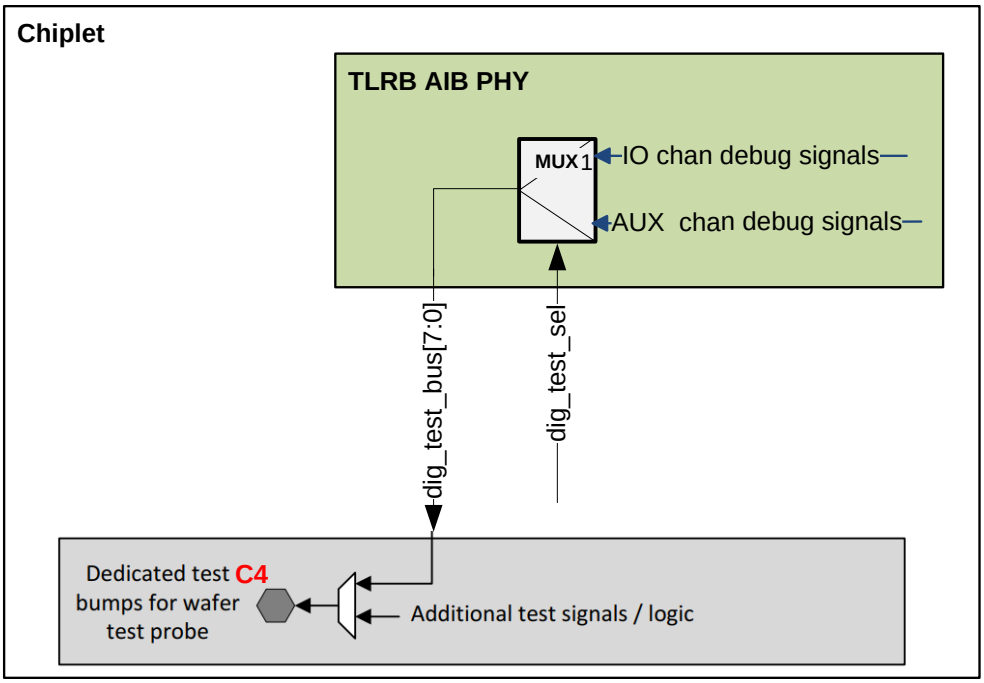


Figure 16 TLRB AIB PHY Digital Test Bus (for DFT)

12. Layout

The AIB architecture requires some constraints concerning the relative layouts of the AIB IO array vs. the microbump array (highlight added):

- "The microbump array pattern is not constrained by AIB specification."
- "Bump ordering and numbering should follow the AIB IO numbering as in Table 3-3 in order to minimize both on die routing from AIB IOs to AIB microbumps and package routings."
    - By this description, an "AIB IO" does not include its corresponding ubump.
    - Table 3-3 title description is "Channel Microbump Ordering and Mapping for 40 IO AIB Plus System"

The figure below (Figure 17 AIB IO Array versus uBump Array Layout) depicts the correspondence between AIB IO array layout and the microbump array layout, and the constraints on the routes between the AIB IO cells and microbumps.

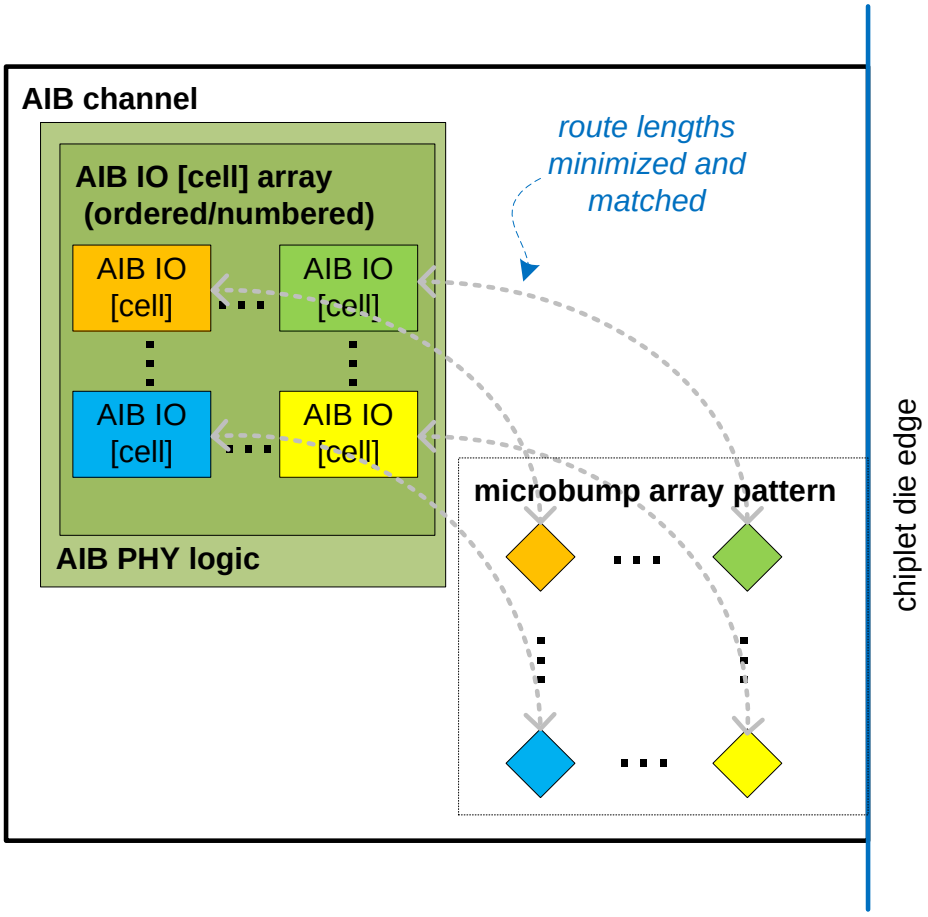


Figure 17 AIB IO Array versus uBump Array Layout

12.1. AIB I/O Cell Array Placement

The *example* AIB I/O block placement in the figure below is derived from the Intel AIB Spec ("Figure 3-33: Example of AIB IO Block Placement vs. Redundancy Shift Direction").

The figure is modified relative to the Intel AIB Spec figure as follows:

- Exclude uBumps signals that are not required for an AIB Base configuration
- Add 40 more synchronous data uBumps
- Add 2 columns (shaded) to each side of the layout to fit the extra data IOs (80 data IOs vs. 40 data IOs)

For a few corresponding connected TX to RX IO pairs between chiplets, the connecting arrows between chiplets illustrate the direction of signal flow, and equal lengths.

For Chiplet B, the redundancy chain connection order is depicted by the red lines (with arrows pointing towards the spares and towards which the function of a defective uBump link pair is shifted).

This example layout preserves the *odd number of IO columns* suggested by the Intel AIB Spec so that the same layout may be used for both and slave with optimal routing between chiplets.

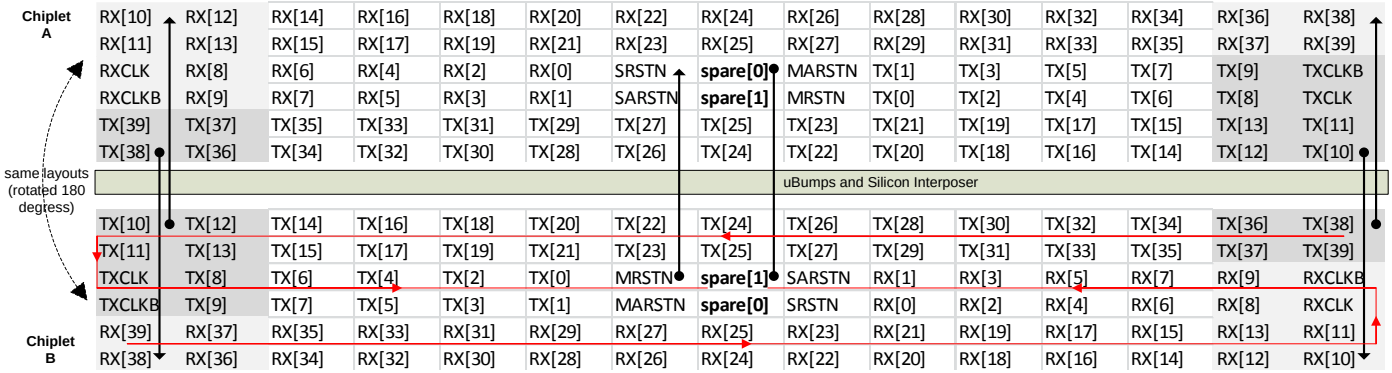


Figure 18 Example AIB I/O Block Placement

The figure above is a pre-layout EXAMPLE based on the Intel AIB Spec.

The actual working layout (adjusted for the actual IO Cell sizes and uBump layout) is as below (work-in-progress):

RX35	RX34	RX37	RX36	RX39	RX38
RX33	RX32	RX31	RX30	RX29	RX28
RX23	RX22	RX25	RX24	RX27	RX26
RX21	RX20	RX19	RX18	RX17	RX16
RX11	RX10	RX13	RX12	RX15	RX14
FS_FWD_CLKB RXCLKB	FS_FWD_CLK RXCLK	RX9	RX8	RX7	RX6
RX1	RX0	RX3	RX2	RX5	RX4
(EMPTY) SARSTN	FS_MAC_RDY SRSTN	SPARE[0]	SPARE[1]	NS_MAC_RDY MRSTN	(EMPTY) MARSTN
TX4	TX5	TX2	TX3	TX0	TX1
TX6	TX7	TX8	TX9	NS_FWD_CLK TXCLK	NS_FWD_CLKB TXCLKB
TX14	TX15	TX12	TX13	TX10	TX11
TX16	TX17	TX18	TX19	TX20	TX21
TX26	TX27	TX24	TX25	TX22	TX23
TX28	TX29	TX30	TX31	TX32	TX33
TX38	TX39	TX36	TX37	TX34	TX35

CHIPLET DIE EDGE

←---------- Less than 312.48 um ----------→

Some of the ID numbers and names of the AIB IO uBumps have been updated (changed relative to the V1.0 DARPA CHIPS Intel specification referenced by this specification) in later, open versions of the Intel AIB specification (also V1.0), after the numbers and names used in this (TLRB AIB PHY) specification and implementation were determined.

The table below provides a translation from the AIB uBump ID numbers and names used in this specification, to the numbers and names used in the newer Intel specification. The uBump numbers and names which differ are highlighted.

Table 16 Translation of uBump Names & ID #s from this Spec to the Open Intel AIB Spec

Intrinsix TLRB Spec derived from: Intel *(CHIPS)* *AIB Architecture Overview V1.0 8 June 2018*				**Intel *Advanced Interface Bus (AIB) DRAFT Specification 1.0 – 11/29/2018* ("public" AIB Spec)**				
uBump ID	**uBump Name**	**uBump ID**	**uBump Name**	**uBump ID**	**uBump Name**	**uBump ID**	**uBump Name**	**Direction**
AIB88	RX[39]	AIB89	RX[38]	AIB88	RX[39]	AIB89	RX[38]	Input
AIB86	RX[37]	AIB87	RX[36]	AIB86	RX[37]	AIB87	RX[36]	Input
AIB84	RX[35]	AIB85	RX[34]	AIB84	RX[35]	AIB85	RX[34]	Input
AIB82	RX[33]	AIB83	RX[32]	AIB82	RX[33]	AIB83	RX[32]	Input
AIB80	RX[31]	AIB81	RX[30]	AIB80	RX[31]	AIB81	RX[30]	Input
AIB78	RX[29]	AIB79	RX[28]	AIB78	RX[29]	AIB79	RX[28]	Input
AIB76	RX[27]	AIB77	RX[26]	AIB76	RX[27]	AIB77	RX[26]	Input
AIB74	RX[25]	AIB75	RX[24]	AIB74	RX[25]	AIB75	RX[24]	Input
AIB72	RX[23]	AIB73	RX[22]	AIB72	RX[23]	AIB73	RX[22]	Input
AIB70	RX[21]	AIB71	RX[20]	AIB70	RX[21]	AIB71	RX[20]	Input
AIB48	RX[19]	AIB49	RX[18]	AIB68	RX[19]	AIB69	RX[18]	Input
AIB46	RX[17]	AIB47	RX[16]	AIB66	RX[17]	AIB67	RX[16]	Input
AIB44	RX[15]	AIB45	RX[14]	AIB64	RX[15]	AIB65	RX[14]	Input
AIB42	RX[13]	AIB43	RX[12]	AIB62	RX[13]	AIB63	RX[12]	Input
AIB40	RX[11]	AIB41	RX[10]	AIB60	RX[11]	AIB61	RX[10]	Input
AIB38	RXCLKB	AIB39	RXCLK	AIB58	fs_fwd_clkb	AIB59	fs_fwd_clk	Input
AIB36	RX[9]	AIB37	RX[8]	AIB56	RX[9]	AIB57	RX[8]	Input
AIB34	RX[7]	AIB35	RX[6]	AIB54	RX[7]	AIB55	RX[6]	Input
AIB32	RX[5]	AIB33	RX[4]	AIB52	RX[5]	AIB53	RX[4]	Input
AIB30	RX[3]	AIB31	RX[2]	AIB50	RX[3]	AIB51	RX[2]	Input
AIB28	RX[1]	AIB29	RX[0]	AIB48	RX[1]	AIB49	RX[0]	Input
AIB26	SARSTN	AIB27	SRSTN	AIB46	(empty)	AIB47	fs_mac_rdy	Input
AIB24	spare[0]	AIB25	spare[1]	AIB44	spare[0]	AIB45	spare[1]	Input/Output
AIB22	MRSTN	AIB23	MARSTN	AIB42	ns_mac_rdy	AIB43	(empty)	Output
AIB20	TX[0]	AIB21	TX[1]	AIB40	TX[0]	AIB41	TX[1]	Output
AIB18	TX[2]	AIB19	TX[3]	AIB38	TX[2]	AIB39	TX[3]	Output
AIB16	TX[4]	AIB17	TX[5]	AIB36	TX[4]	AIB37	TX[5]	Output
AIB14	TX[6]	AIB15	TX[7]	AIB34	TX[6]	AIB35	TX[7]	Output
AIB12	TX[8]	AIB13	TX[9]	AIB32	TX[8]	AIB33	TX[9]	Output
AIB10	TXCLK	AIB11	TXCLKB	AIB30	ns_fwd_clk	AIB31	ns_fwd_clkb	Output
AIB8	TX[10]	AIB9	TX[11]	AIB28	TX[10]	AIB29	TX[11]	Output
AIB6	TX[12]	AIB7	TX[13]	AIB26	TX[12]	AIB27	TX[13]	Output

AIB4	TX[14]	AIB5	TX[15]	AIB24	TX[14]	AIB25	TX[15]	Output
AIB2	TX[16]	AIB3	TX[17]	AIB22	TX[16]	AIB23	TX[17]	Output
AIB0	TX[18]	AIB1	TX[19]	AIB20	TX[18]	AIB21	TX[19]	Output
AIB68	TX[20]	AIB69	TX[21]	AIB18	TX[20]	AIB19	TX[21]	Output
AIB66	TX[22]	AIB67	TX[23]	AIB16	TX[22]	AIB17	TX[23]	Output
AIB64	TX[24]	AIB65	TX[25]	AIB14	TX[24]	AIB15	TX[25]	Output
AIB62	TX[26]	AIB63	TX[27]	AIB12	TX[26]	AIB13	TX[27]	Output
AIB60	TX[28]	AIB61	TX[29]	AIB10	TX[28]	AIB11	TX[29]	Output
AIB58	TX[30]	AIB59	TX[31]	AIB8	TX[30]	AIB9	TX[31]	Output
AIB56	TX[32]	AIB57	TX[33]	AIB6	TX[32]	AIB7	TX[33]	Output
AIB54	TX[34]	AIB55	TX[35]	AIB4	TX[34]	AIB5	TX[35]	Output
AIB52	TX[36]	AIB53	TX[37]	AIB2	TX[36]	AIB3	TX[37]	Output
AIB50	TX[38]	AIB51	TX[39]	AIB0	TX[38]	AIB1	TX[39]	Output

12.1.1. RTL Re-map of uBump AIB IDs using `defines

The AIB PHY supports an RTL feature to re-map the AIB IDs according to the highlighted uBump ID columns (public AIB spec) in the table above.

If the Verilog text macro ***`AIB_ID_REMAP*** is defined, then the following bussed ports connections to the AIB IO Channel Hard Macro (internal to PHY) are permuted as indicated below. The effect of the permutation is to make the top-level port IDs of the PHY RTL match the public AIB Specification as in the table above.

```
itrx_aib_phy_io_chan u_io_chan(  // AIB IO CHANNEL HARD MACRO INSTANCE SNIPPET

            .ubump       ({    ubump[89:70],     ubump[19:0],     ubump[69:20]}),
            .txen        ({     txen[87:68],      txen[19:0],      txen[67:20]}),
            .rxen        ({     rxen[87:68],      rxen[19:0],      rxen[67:20]}),
            .idat_selb   ({idat_selb[87:68], idat_selb[19:0], idat_selb[67:20]}),

                ...
```

If the **both** of the Verilog text macros (***`AIB_ID_REMAP*** and ***`AIB_ID_REMAP_IOCHAN***) are defined, then the AIB ID remapping is done **directly** internal to the AIB IO Channel Hard Macro without permutation. This is the preferred setting for new designs that re-implement the AIB IO Channel Hard Macro.

For new designs, **the recommended RTL top level module, and RTL configuration** are as indicated by the following example tool string:

```
"tlrb_aib_phy_ext_mc.sv +define+AIB_ID_REMAP +define+AIB_ID_REMAP_IOCHAN
+define+ITRX_AIB_JTAGSM –defparam NCH=<Number of AIB IO channels>"
```

12.2. AIB Column Edge Layout

The top-level AIB IO Channel block and the AIB AUX channel block of a TLRB AIB PHY are required to be laid out as an AIB column (implementing 1 AIB channel and 1 AIB IO channel). The architecture specification imposes a layout restriction on the layout of an AIB column (highlight added):

> "Identical AIB channels can be stacked on top of one another to create a full AIB column, with channel 0 always at the bottom of the column [for both master and slave chiplets] with respect to the vertical chiplet die edge."

The diagram below (Figure 19 AIB Column AUX/IO Channel Layout Ordering) depicts how the TLRB AIB PHY (implementing only the shaded IO Chan0 and AUX Chan blocks) are required to be oriented along the vertical edges of the chiplet die.

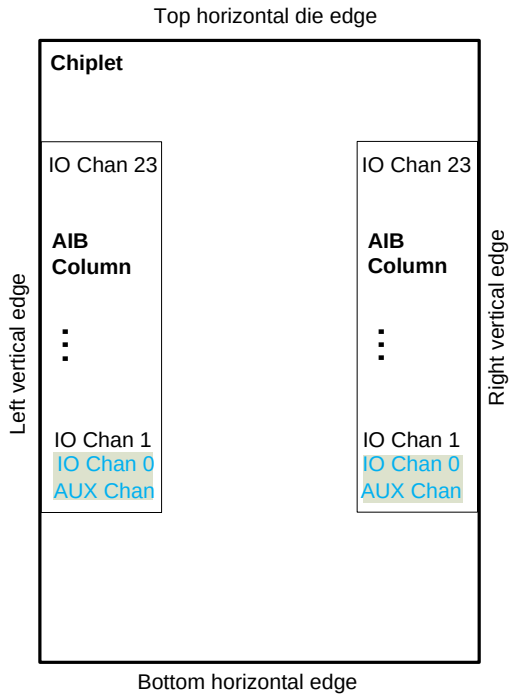


Figure 19 AIB Column AUX/IO Channel Layout Ordering

13. Analog Blocks

The remainder of this section is intentionally omitted for versions of this spec that are published publically.

13.1. Rx Clock Programmable Delay Line (PDL)

13.2. AIB IO Buffer Analog

13.2.1. AIB IO Design Hierarchy

13.2.2. AIB IO Power Domains and POR

13.2.3. AIB IO ESD

13.2.4. Electromigration

13.2.5. AIB IO Buffer TX Analog

13.2.5.1.TX Level Shifters

13.2.5.2.IO TX Electrical Specifications

13.2.6. AIB IO Buffer RX Analog

13.2.6.1.RX Level Shifters

13.2.6.2.IO RX Electrical Specifications

14. TLRB AIB PHY Controller & Register Map Extensions

The TRLB AIB PHY module as described in the preceding sections provides flexibility relative to integrating the module into a chiplet, by allowing direct control and observation of the low-level AIB configuration and JTAG interface signal ports.

These low-level signal ports are the same signals as are defined in the Intel AIB Architecture specification (e.g. idat_selb, jtag_intest, etc.), and are fully listed in section 5 (Digital I/O Ports) of this specification.

Although the TRLB AIB PHY module may be implemented without the extensions described in this section, this section introduces a *new* module called TLRB AIB PHY **Extended**. The purpose of this new module is to simplify and facilitate the integration of a TLRB AIB PHY module into a chiplet. This new module extends the base TLRB AIB PHY module of the preceding sections by including (wrapping) the TLRB AIB PHY module together with additional modules that offer memory mapped registers and controller functions.

The base-core TLRB AIB PHY module (subsequently referred to as the **base-core** TLRB AIB PHY) is wrapped together with an APB Registers module, and a JTAG TAP controller module. The resulting wrapped module is called TLRB AIB PHY Extended, and is depicted in the figure below (Figure 23 TLRB AIB PHY Extended).

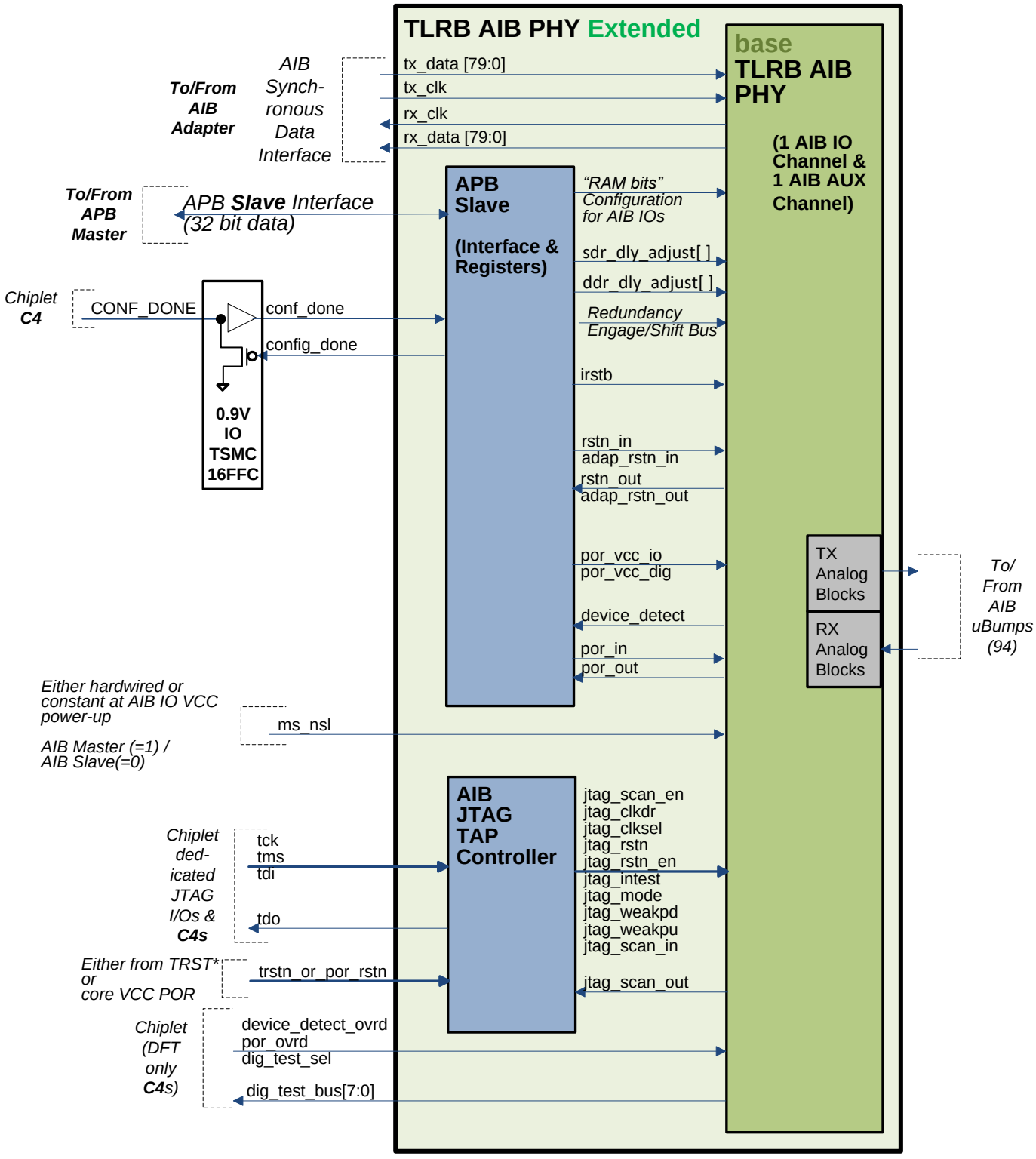
The TLRB AIB PHY Extended module adds 2 new **standard** top-level interfaces (ports):

1. ARM AMBA APB Interface (32-bit data) for memory mapped register access
2. IEEE JTAG 1149.1 Interface (4 pins)

Although 2 new interfaces are added, the new interfaces are now standard, and replace many other lower level interface ports/signals.

The APB interface further (in addition to the AIB configuration and JTAG related low level ports) replaces the AUX channel Device Detect status, the AUX channel PoR control and status, and the IO channel asynchronous resets.

Figure 23 TLRB AIB PHY Extended



14.1. Digital I/O Ports (TLRB AIB PHY Extended)

The top-level Digital I/O Ports of the TLRB AIB PHY Extended are simplified relative to the base-core TLRB AIB PHY.

Referring to the table of TLRB AIB PHY Extended Digital I/O Ports below, and comparing relative to the base-core TLRB AIB PHY Digital I/O ports table:

- The “Synchronous Data” group (including tx_clk, tx_data [79:0], rx_clk, and rx_data [79:0] signals/busses) interfaces are *the same* as for the base-core TLRB AIB PHY.
- The set of non-JTAG DFT signals (device_detect_ovrd, por_ovrd, dig_test_sel, and dig_test_bus [7:0]) are *the same* as for the base-coreTLRB AIB PHY. All of these signals need to connect to DFT C4s chiplet pins (not required to be package pins) as required by the Intel AIB Architecture Spec.
- The “JTAG” group of signals is *new* for the TLRB AIB PHY Extended module, and *replaces* the jag_* ports which have been removed from the top level.
- The “APB Interface” group of signals is *new* for the TLRB AIB PHY Extended module, and *replaces* the AIB IO cell configuration ports, and delay adjustment ports which have been removed from the top level.
    - Also, the AIB AUX channel Device Detect slave status, PoR master status, and PoR slave control bit top level ports have been replaced and remapped to an APB register.
    - Also, the AIB IO Channel Reset control and status top-level ports (e.g. rstn_in, adap_rstn_in, rstn_out, and adap_rstn_out) have been replaced and remapped to an APB register.
- The ms_nsl port signal is *the same* as for the base-core TLRB AIB PHY. It needs to be either hardwired for AIB master or slave operation, or otherwise set to a constant value prior to turning on power to the AIB IOs (VccL aka VCC_IO).
- The conf_done (configuration done) signal port is *new* for the TLRB AIB PHY Extended module. The Intel AIB Architecture spec describes this signal in Section 6. The Intel spec requires this signal to be connected to a C4 on each AIB chiplet within a package.
- The AIB IO channel por_vcc_* signals are no longer top level digital IO ports. How they are driven depends on whether the TLRB AIB PHY is a master or a slave as described in the section below (APB & POR Reset Control).

Table 30 Digital I/O Ports (TLRB AIB PHY Extended)

Signal name	Direction	Description
		Synchronous Data (ND = # of synchronous Data uBumps; i.e. 80)
tx_data[*ND*-1:0]	Input	Synchronous (to tx_clk) data output transmitted to uBump This is the same as the "TX DATA" bus from the AIB Adapter to the AIB IOs as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
rx_data[*ND*-1:0]	Output	Synchronous (to rx_clk) data input received from uBump This is the same as the "RX DATA" bus from the AIB IOs to the AIB Adapter as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
tx_clk	Input	Transmit clock for synchronous tx_data
rx_clk	Output	Receive clock for synchronous rx_data
		IO Cell Configuration (asynchronous/static)
ms_nsl *	Input	Master Not Slave signal (hardwires the AUX channel POR/device_detect AIB IO cell configuration for master vs. slave mode) 0=Slave, 1=Master
conf_done	Input	Configuration Done (Connects to a chiplet C4. See section 6 of the Intel AIB Architecture Spec for a detailed description.)
config_done	Output	Drives the "open drain" chiplet C4 IO buffer capable of driving the AIB CONF_DONE chiplet/package IO. The CONF_DONE package pin is specified by the Intel AIB spec to be a package pin, and implements the wired-logic (logical AND of AIB interface status across multiple chiplets) configuration done function. The Intel AIB spec signal "config_done" turns off the open drain driver when asserted high so that CONF_DONE may be pulled up.
		DFT
dig_test_bus[7:0]	Output	Debug signals from AIB IO and AUX Channels (designer defined signals , **TBD**)
device_detect_ovrd *	Input	DFT: Device Detect override (forces device_detect output high for wafer test) (1 = override; C4 at package level connected to logic low)
por_ovrd *	Input	DFT: POR override for wafer test (0 = override; C4 at package level connected to logic high)
dig_test_sel	Input	DFT: Selects source of debug signals driven to the dig_test_bus [] output from either the AUX channel (=0) or the IO channel (=1).
		JTAG (See section 5 of the Intel AIB Architecture Spec for detailed descriptions)
tck	Input	Test Clock
tdo	Output	Test Data Output
tdi	Input	Test Data Input
tms	Input	Test Mode Select
trstn_or_por_rstn	Input	Asynchronous JTAG controller logic reset (active low). The IEEE 1149.1 standard has a "reset at power-up" requirement. Inclusion of the optional TRST* (TRSTN) pin meets this requirement. Alternatively, a POR signal from the system may be used to meet this requirement. This input signal "trstn_or_por_rstn" can either be driven by TRST*, or by an on-chip POR reset signal from the system. The Intel AIB standard does not explicitly include or require the TRST* pin.
		APB Slave Interface ***(See the ARM AMBA 3 APB Protocol v1.0 specification for detailed descriptions)***
pclk	Input	Clock.
presetn	Input	Reset. The APB reset signal is active LOW. This signal is normally connected directly to the system bus reset signal.
paddr[11:0]	Input	Address.
pwrite	Input	Direction.
psel	Input	Select.
penable	Input	Enable.
pwdata[31:0]	Input	Write Data.
prdata [31:0]	Output	Read Data.
pready	Output	Ready.

Signal name	Direction	Description
		Configuration and status outputs for system use (synchronous to APB Clock)
r_apb_iddr_enable	output	APB register bit field (IDDREN_DAT). Control/configuration bit to local AIB PHY.
r_apb_ns_rstn	output	APB register bit field (RSTN). Control bit to near-side, local AIB PHY. Drives MRSTN uBump.
r_apb_ns_adap_rstn	output	APB register bit field (ADAPTER_RSTN). Control bit to near-side, local AIB PHY. Drives MARSTN uBump.
r_apb_fs_rstn	output	APB register bit field (RSTN_IN). Status bit from far-side, remote AIB PHY. Received SRSTN uBump synchronized to pclk.
r_apb_fs_adap_rstn	output	APB register bit field (ADAPTER_RSTN_IN). Status bit from far-side, remote AIB PHY. Received SARSTN uBump synchronized to pclk.

** These are **VCC_IO level** AUX Channel signals. All other ports are VCC_DIG levels.*

14.2. JTAG TAP Controller

The TLRB AIB PHY Extended module includes a JTAG TAP controller sub-module. The JTAG TAP controller sub-module drives (controls) all of the "jtag_*" input signal ports of the base-core TLRB AIB PHY. The JTAG TAP controller also receives the jtag_scan_out signal port from the base-core TLRB AIB PHY. The "jtag_" signals are defined in the Intel AIB spec.

The TLRB AIB PHY Extended module replaces the jtag_* ports of the TLRB AIB PHY with a set of 4 standard JTAG (IEEE 1149.1) signals:

tck	Test Clock
tdo	Test Data Output
tdi	Test Data Input
tms	Test Mode Select

The signals in the table above are ports of the TLRB AIB Extended module, and are intended to be connected (external to the TLRB AIB Extended module) to I/O drivers/receivers from the TSMC 16FFC IO IP library (with C4 pads) that are configured for JTAG standard voltage levels. The external JTAG pins are further described in section 5 (JTAG Boundary Scan) of the AIB Architecture Specification. JTAG clocking (TCK) supports frequencies up to 33 1/3 MHz.

The JTAG TAP Controller implements the BYPASS instruction (encoded as 7'b111_1111), and private AIB instructions to support AIB IO test.

AIB Boundary Scan Register Private Instructions (copied from Table 5-1 of the Intel AIB Spec)

Instruction Name	Instruction Binary	Description (for AIB IO boundary scan)
AIB_SHIFT_EN	7'b000_1100	Enable serial shift operation
AIB_SHIFT_DIS	7'b000_1101	Disable serial shift operation
AIB_TRANSMIT_EN	7'b000_1110	Enable TX transmit output
AIB_TRANSMIT_DIS	7'b000_1111	Disable TX transmit output
AIB_RESET_EN	7'b001_0000	Enable reset
AIB_RESET_DIS	7'b001_0001	Disable reset
AIB_WEAKPU_EN	7'b001_0010	Enable weak pull up. Global AIB control signal connects to all AIB IO cells.
AIB_WEAKPU_DIS	7'b001_0011	Disable weak pull up. Global AIB control signal connects to all AIB IO cell.
AIB_WEAKPDN_EN	7'b001_0100	Enable weak pull down. Global AIB control signal connects to all AIB IO cell
AIB_WEAKPDN_DIS	7'b001_0101	Disable weak pull down. Global AIB control signal connects to all AIB IO cell
AIB_INTEST_EN	7'b001_0110	Enable in test operation
AIB_INTEST_DIS	7'b001_0111	Disable in test operation
AIB_JTAG_CLKSEL	7'b001_1000	Select clock for JTAG operation for AIB_IO clock
AIB_RESET_OVRD_EN	7'b100_1000	Enable reset overriding for AIB IO's reset signals
AIB_RESET_OVRD_DIS	7'b100_1001	Disable reset overriding for AIB IO's reset signals

14.2.1. Selection of alternate RTL code for the JTAG TAP FSM using a `define macro

If the Verilog text macro `ITRX_AIB_JTAGSM is defined, then an alternative, Intrinsix Verilog module is selected for the JTAG TAP FSM. The Intrinsix version of the JTAG TAP FSM Verilog module is logically equivalent to the original Intel version (dbg_test_jtagsm.v). The Intrinsix version is only provided as a convenience by not requiring the use AIB RTL sources as distributed by Intel.

For new designs, **the recommended RTL top level module, and RTL configuration** are as indicated by the following example tool string:

```
"tlrb_aib_phy_ext_mc.sv +define+AIB_ID_REMAP +define+AIB_ID_REMAP_IOCHAN
+define+ITRX_AIB_JTAGSM -defparam NCH=<Number of AIB IO channels>"
```

14.3. AMBA (APB) Slave Module

As specified by the Intel AIB Architecture Spec, the AIB I/O Buffer is universal. Each universal AIB I/O buffer needs to be configured according its functional use (e.g. RX vs. TX, clock vs. data, synchronous vs. asynchronous) by setting the values of its configuration inputs. The configuration inputs for each AIB I/O buffer are listed below. These AIB IO Buffer configuration input ports are defined in the AIB Spec where they are also called "RAM bits".

- ipdrv[1:0]
- indrv[1:0]
- iddren
- idat_selb
- txen
- rxen[2:0]

The AIB Spec also defines a "Manual DLL setting" for configuring the delay of the programmable delay line, and specifies that the setting is required for an AIB Base Configuration. The corresponding inputs to the TLRB AIB PHY dedicated for this purpose are:

- sdr_dly_adjust[ ]
- ddr_dly_adjust[ ]

An ARM AMBA APB Slave module within the TLRB AIB PHY Extended module drives the RAM bits and manual DLL adjustment settings as described above. The APB module also drives and receives *other* AIB IO Buffer port signals as depicted in the block diagram (Figure 23 TLRB AIB PHY Extended).

The APB module supports the following top-level interface signal pins as specified by the APB standard (AMBA 3 APB Protocol v1.0). Register access completes with 0 wait states for both reads and writes.

APB Slave Interface Signal	Short Description	Direction
pclk	Clock.	Input
presetn	Reset.	Input
paddr[11:0]	Address.	Input
pwrite	Direction.	Input
psel	Select.	Input
penable	Enable.	Input
pwdata[31:0]	Write Data.	Input
prdata [31:0]	Read Data.	Output
pready	Ready.	Output

14.3.1. APB & POR Reset Control

The figure below (Figure 24 POR Control) illustrates how both the AIB master and AIB slave drive the por_vcc_dig and por_vcc_io input port signals of the base-core TLRB AIB PHY to the AIB IO channel sub-blocks. These signals perform the function of the POR broadcasted control signal referred to in the Intel AIB Spec. The Intel AIB Spec (POR Sync Up and Test section) refers to a signal that is broadcasted to all of the AIB IOs to "force IOs into tristate mode with enabled weak pull down to avoid contention".

Both the IO channel block and the AUX channel block in the figure use multiple instances of the same exact AIB IO cell buffer design including logic, circuitry, and layout. The AIB IO cell buffer (identically replicated for every AIB IO) is designed to support two different power rails VCC_DIG and VCC_IO. However, for the AUX channel, the VCC_DIG and VCC_IO rails of the AIB IO Cell buffer modules are tied together, and are driven from a common VCC (VCC_IO). Consequently, for the AUX channel, the POR signals can cross the AIB link and produce an effect that depends only on the link power (VCC_IO) being powered on.

As further depicted in the diagram, the AIB slave can control the por_vcc_dig and por_vcc_io signals of both itself and the master via APB register settings.

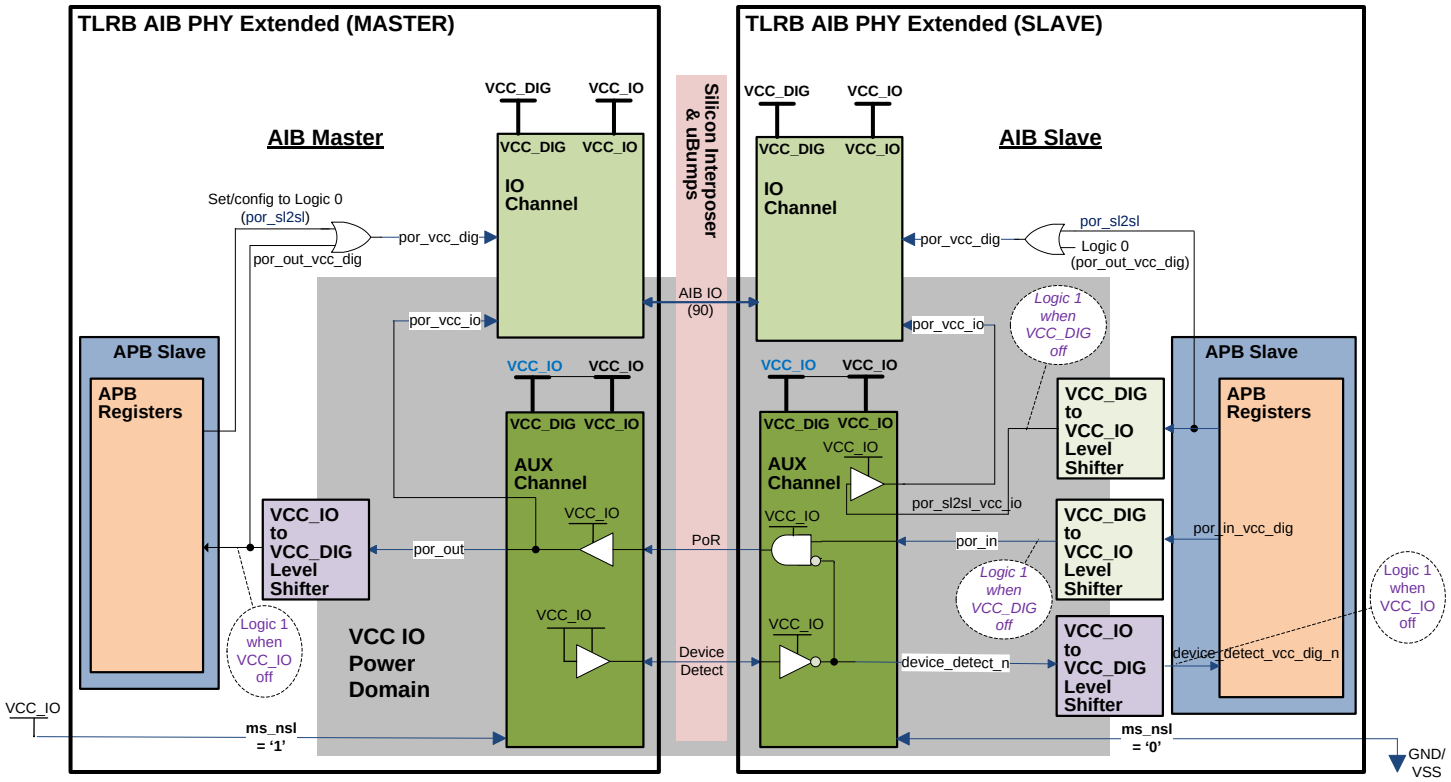


Figure 24 POR Control

14.4. AMBA (APB) Register Map

Register Name / Link to Full Description	Short Register Description	Offset Address	AIB Type
TX_DRV_STRENGTH	TX Drive Strength	0x000	AIB IO Channel Registers *
TX CONFIG	TX Configuration	0x004	
RX_ENABLE	RX Enable	0x008	
RX_DELAY_ADJUST	RX Clock Delay Adjustment	0x00C	
REPAIR_ADDR	Repair Information	0x010	
RESETS	Resets (Configuration and Status)	0x014	
CONFIG_DONE	Configuration Done signal and Status	0x800	AIB PHY Register
DD_POR	Device Detect and Power on Reset	0x804	AIB AUX Channel Register
REVISION	TLRB AIB implementation revision	0xFFC	AIB PHY Register

* The TLRB implementation supports a single AIB IO channel. The channel registers reserve APB address space so that they may be replicated up to 24 times for future implementations that specify multiple channels.

14.4.1. TX_DRV_STRENGTH

offset from base	0x0																				default	0x00000555
TX Drive Strength Register																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
11:10	IPDRV_DAT	rw	0x1	Drive strength for TX Data (P-driver)
9:8	INDRV_DAT	rw	0x1	Drive strength for TX Data (N-driver)
7:6	IPDRV_RST	rw	0x1	Drive strength for TX Resets (P-driver)
5:4	INDRV_RST	rw	0x1	Drive strength for TX Resets (N-driver)
3:2	IPDRV_CLK	rw	0x1	Drive strength for TX Clocks (P-driver)
1:0	INDRV_CLK	rw	0x1	Drive strength for TX Clocks (N-driver)

IPDRV[1:0] / INDRV[1:0]	Operation Modes / Description for both P-driver and N-driver
00	25% less driving strength
01	Normal driving strength (default)
10	25% more driving strength
11	50% more driving strength

14.4.2. TX CONFIG

offset from base	0x4																					default	0x000000D7							
TX Configuration Register																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
7	TXEN_DAT_GROUP1	rw	0x1	TXEN for TX Data (TX[39:20] uBumps)
6	IDDREN_DAT	rw	0x1	IDDREN for TX Data
5	IDAT_SELB_DAT	rw	0x0	IDAT_SELB for TX Data
4	IDAT_SELB_RST	rw	0x1	IDAT_SELB for TX Resets
3	IDAT_SELB_CLK	rw	0x0	IDAT_SELB for TX Clocks
2	TXEN_DAT_GROUP0	rw	0x1	TXEN for TX Data (TX[19: 0] uBumps)
1	TXEN_RST	rw	0x1	TXEN for TX Resets
0	TXEN_CLK	rw	0x1	TXEN for TX Clocks

The encodings of IDDREN, IDAT_SELB, and TXEN match the Intel AIB Spec descriptions, and the table in this specification (Table 11 Digital I/O Ports).

14.4.3. RX_ENABLE

offset from base	0x8																					default	0x00000492							
RX Enable Register																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
11:9	RXEN_DAT_GROUP1	rw	0x2	RXEN[2:0] for RX Data (RX[39:20] uBumps)
8:6	RXEN_DAT_GROUP0	rw	0x2	RXEN[2:0] for RX Data (RX[19: 0] uBumps)
5:3	RXEN_RST	rw	0x2	RXEN[2:0] for RX Resets
2:0	RXEN_CLK	rw	0x2	RXEN[2:0] for RX Clocks

RXEN[2:0]	Description
000	Asynchronous data input.
001	Synchronous DDR input.
010	Disabled RX. (default)
011	Input Clock buffer enabled.
100	Synchronous SDR input.
Other codes	Disabled RX.

The RXEN inputs of the TX AIB IOs are tied-off to “Disabled RX” (010). The RXEN of TX AIB IOs may be set and tested via the JTAG BSR.

14.4.4. RX_DELAY_ADJUST

offset from base	0xC																														default	0x00000000
RX Delay Adjustment Register																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

bits	field name	s/w access	reset value	field description
13:8	DDR_DLY_ADJUST	rw	0x00	RX DLL Manual Mode delay adjust for DDR
5:0	SDR_DLY_ADJUST	rw	0x00	RX DLL Manual Mode delay adjust for SDR

The RX strobe clock (inclk) delay line is composed of 64 total delay cells supporting programmable delays ranging from the delay of 1 cell to the delay of 64 cells.

The encoding of the adjust fields is as follows:

< >_DLY_ADJUST [5:0] field Value (< > = “SDR” or “DDR”)	Delay (Number of Delay Line cells)
0b000000	1 cell of 64 total cells
0b000001	2 cells of 64 total cells
• • •	• • •
0b111111	64 cells of 64 total cells

14.4.5. REPAIR_ADDR

offset from base	0x10																				default	0x00000000
Redundancy Repair Address Register																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
11	VALID	rw	0x0	Repair address is valid. If no repair is necessary set this bit to 0.
10	ADDR_DIR	rw	0x0	The repair address bit 10 indicates direction of redundancy repair (logic 1 for TX direction, logic 0 for RX direction)
9:0	ADDR_LOC	rw	0x0	The location of the faulty connection relative to the spares. Address [9:0] == 0 is for the first two bits which are adjacent to the spare bits.

The format of the REPAIR_ADDR register follows the format described in the Intel AIB Spec.

The LOC field indicates the “distance away” from the spare [1:0] uBump pair in the center of the redundancy chain.

For example, if ADDR_DIR = 1 (TX):

ADDR_LOC = 0 → MRSTN / MARSTN

ADDR_LOC = 1 → TX[0] / TX[1]

…

ADDR_LOC = 21 → TX[38] / TX[39]

Also, if ADDR_DIR = 0 (RX):

ADDR_LOC = 0 → SRSTN / SARSTN

ADDR_LOC = 1 →RX[0] / RX[1]

…

ADDR_LOC = 21→ RX[38] / RX[39]

14.4.6. RESETS

offset from base	0x14																													default	0x00000000
Redundancy Repair Address Register																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
4	ADAP_RSTN_IN	ro*	0x0	Value of adap_rstn_out (active low) port of the base-core TLRB AIB PHY (SARSTN uBump).
3	RSTN_IN	ro*	0x0	Value of rstn_out (active low) port of the base-core TLRB AIB PHY (SRSTN uBump).
2	ADAPTER_RSTN	rw	0x0	Drives adap_rstn_in (active low) port of the base-core TLRB AIB PHY to this value. The rstn_out port directly drives the MARSTN AIB uBump (when IRSTB is released).
1	RSTN	rw	0x0	Drives rstn_in (active low) port of the base-core TLRB AIB PHY to this value. The rstn_out port directly drives the MRSTN uBump (when IRSTB is released).
0	IRSTB	rw	0x0	Drives irstb (active low) port of the TLRB AIB PHY to this value.

***** The read-only status bits are synchronized to the APB clock. The synchronization registers are reset to 0x0 when APB reset is asserted; however the "reset value" that is read following APB reset negation (release) depends on the value of the status bit as driven by the external source of the status bit.

As described in the Intel AIB Spec, the AIB IO channel reset (IRSTB), resets *all* AIB IO DFFs and tristates all TX outputs and enables weak pull down at all microbumps at the same time to avoid contention during reset.

IRSTB needs to be released (set to 1) following configuration done.

14.4.7. CONFIG_DONE

offset from base	0x800																													default	0x00000000
Configuration Done Register																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
1	CONF_DONE	ro	0x0	"CONF_DONE" (AIB Arch Spec signal) Status from Chiplet C4 pin. Pulled up at the package level until all chiplets driver DONE via open drain.
0	DONE	rw	0x0	Configuration done (output to open drain driver) Drives Intel AIB Spec "config_done" signals

14.4.8. DD_POR

offset from base	0x804	default	0x0000000C

Device Detect and Power On Reset Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
3	POR_SL2SL	rw	0x1	Value of POR to drive from the AIB slave to its own AIB IO channel(s) por_vcc_dig and por_vcc_io inputs. When the AIB is a master (ms_nsl = 1) this bit must be set to 0.
2	POR_SL2MS	rw	0x1	Value of POR to drive from the AIB slave to the AIB master over the AIB AUX channel link. When the AIB is a master (ms_nsl = 1) this bit has no effect. The slave AUX channel must also be receiving Device Detect from the master in order for POR to be asserted (high) over the AIB link from slave to master.
1	POR	ro	0x0	AUX channel value of POR as monitored by the AIB master. Always reads 0x0 when AIB is a slave (ms_nsl = 0).
0	DD	ro	0x0	AUX channel value of Device Detect as monitored by the AIB slave. Always reads 0x0 when AIB is a master (ms_nsl = 1).

14.4.9. REVISION

offset from base	0xFFC	default	0x0A052200

Revision Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	field name	s/w access	reset value	field description
31:8	DEVICE_ID	ro	0x0A0522	The upper 24 bits is compressed ASCII format (5 bits per capital letter), and decodes to "TAIB", for TLRB AIB.
7:0	REVISION	ro	0x00	The upper 4 bits are the MAJOR revision. The lower 4 bits are the MINOR revision.

14.5. Bring Out of Reset Sequence

The basic sequence of steps (described in more detail in the Intel AIB Spec) by which the TLRB AIB PHY is brought out of reset to establish “normal operation” is as follows:

1. POR
2. Configuration
3. config_done/CONF_DONE
4. Reset signals released

Relative to programming of the TLRB AIB PHY Extended module APB Registers, the sequence of steps is elaborated below:

1. POR (DD_POR Register)

   Slave sequence:

   1. Loop while (DD_POR.DD != 1)
   2. Write DD_POR fields:
      DD_POR.POR_SL2SL ← 0
      DD_POR.POR_SL2MS ← 0

   Master sequence:

   1. Write DD_POR field:
      DD_POR.POR_SL2SL ← 0
   2. Loop while (DD_POR.POR != 1)

2. Configuration

Write *all* the following channel configuration registers/fields in any order.

Slave *or* Master:

- TX_DRV_STRENGTH
  Field values are application specific. (Default or any values work for Verilog simulation)

- TX_CONFIG

  TX_CONFIG.IDDREN_DAT ← '1' if DDR, '0' if SDR

  TX_CONFIG.TXEN_DAT_GROUP1 ← '1' if 80 TX Data ubumps, '0' if reduced to 40 TX Data ubumps

  TX_CONFIG.<*other fields*> ← "reset value" values

- RX_ENABLE

  RX_ENABLE.RX_DAT_GROUP1 ← 3'b001 if DDR, or 3'b100 if SDR if given 80 RX Data ubumps; Leave it as default 3'b010 if reduced to 40 RX Data ubumps

  RX_ENABLE.RX_DAT_GROUP0 ← 3'b001 if DDR, 3'b100 if SDR;

  RX_ENABLE.RXEN_RST ← 3'b000, Async input for reset;

  RX_ENABLE.RXEN_CLK ← 3'b011, input clock enable;

- RX_DELAY_ADJUST
  The field values to be written are **TBD** (based on PD results). Reset values of `b0 work for Verilog simulations.

2. Configuration (continued)

Slave *vs.* Master:

- REPAIR_ADDR

  Write the "reset value" field values of `b0 for both slave and master if no repair is required.

  If any redundancy repair engaged, set bit 11 to 1, and set the other 2 fields accordingly. Notice this register needs to be programed differently for the Master and the Slave. For example, if Master bump #66 is broken:

  - Master bump #66 is for TX[22], so field 'ADDR_DIR' is 1 for TX, and 'ADDR_LOC' is 13, so program 12'hc0d to Master REPAIR_ADDR register;
  - Accordingly, Slave RX[22] is to be repaired, so program 12'h80d to Slave REPAIR_ADDR register;

3. config_done/CONF_DONE (CONFIG_DONE Register)

Slave or Master sequence:

1. CONFIG_DONE.DONE ← '1'
2. Loop while (CONFIG_DONE.CONF_DONE != 1)

4. Reset signals released (RESETS Register)

Slave or Master sequence:

1. RESETS.RSTN ← '1'
   RESETS.ADAPTER_RSTN ←'1'
   RESETS.IRSTB ← '1'
2. Loop while ((RESETS.RSTN_IN != 1) || (RESETS.ADAPTER_RSTN_IN != 1))

14.6. DFT Scan

TBD

The logic added to the base-core TLRB AIB PHY (e.g. APB and JTAG modules) to comprise the total AIB TLRB PHY Extended module is synthesizable RTL. It is expected that scan chains and associated top-level scan ports are inserted during synthesis to allow testing of all of the added DFFs within AIB TLRB PHY Extended module (exclusive of the AIB IO Channel sub-block and AIB AUX channel sub-block within the base-core TLRB AIB PHY).

input	scan_mode
input	scan_shift
input	scan_rst_n
input	scan_clk
input	scan_in
output	scan_out

14.6.1. ATPG Override of the AIB IO Channel BSR Chain

The JTAG BSR chain infrastructure (as described in 4.3 AIB IO Cell) within the AIB IO Channel sub-block may be overridden with an ATPG scan chain. Logic added to the base-core TLRB AIB PHY provides some top level ports and multiplexers to support this override feature.

The Intel AIB Architecture specification V1.0 (Table 1 Reference Document Items number 1) describes the ATPG override of the BSR chain requirement in a section labeled: “8.7.1.2 AIB IO and Adapter Test Path”.

Referring to the figure below (Figure 25 AIB IO Channel Boundary Scan Chain Used in ATPG Test), the following top level ports provide access to a single separate AIB IO Channel ATPG scan chain:

input	atpg_bsr_ovrd_mode
input	atpg_bsr_scan_shift
input	atpg_bsr_scan_shift_clk
input	atpg_bsr_scan_in
output	atpg_bsr_scan_out

If the override feature is unused, then tie-off all the inputs above to Logic0, and leave the output unconnected.

The ATPG scan chain piggy-backs onto the boundary scan register (BSR) chain within the AIB IO Channel. The selection of ATPG vs. JTAG/BSR mode use is controlled by the atpg_bsr_ovrd_mode input port (‘0’ = JTAG/BSR, ‘1’ = ATPG). The JTAG vs. the ATPG use of the BSR chain is mutually exclusive.

Since the ATPG scan chain leverages the JTAG boundary scan infrastructure (for which boundary scan register cell flip-flops are not reset), the flip-flops constituting the chain are not reset-able. A constant bit vector of equal to the length of the chain has to be shifted in to the BSR chain to initialize the flip-flops.

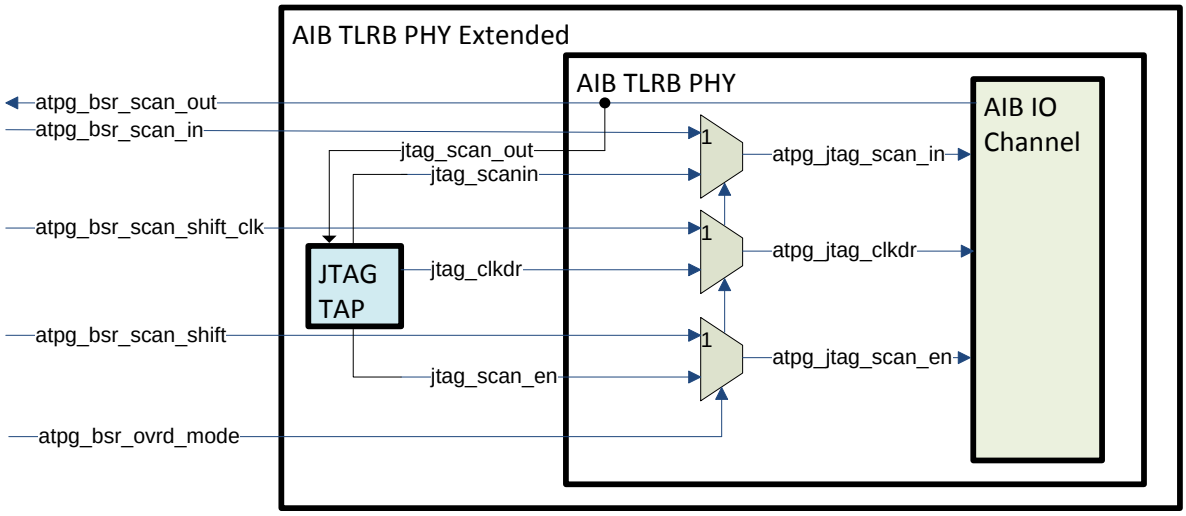


Figure 25 AIB IO Channel Boundary Scan Chain Used in ATPG Test

Note: The Intel AIB spec requires/expects the JTAG_INTEST instruction/signal to be loaded prior to testing the paths to/from the adapter (logic external to the TLRB AIB PHY) using the ATPG scan chain.

Relevant snippet from Intel AIB spec section "8.7.1.2 AIB IO and Adapter Test Path" (highlight added):

> "Apart from having the ATPG override, to be able to observe the functional path from the adapter and override the signal to the adapter, an additional INTEST multiplexer as shown in Figure 5-4 is added. The control signal of this multiplexer is activated using the INTEST JTAG instruction."

15. TLRB AIB PHY BSR Infrastructure Collated Summary

The tables below depict collated details about the standard Intel AIB JTAG BSR scan chain and infrastructure as it applies to the TLRB AIB PHY. Each line entry within the tables represents one AIB IO Cell. There are 90 AIB IO Cells altogether within the AIB IO Channel. Each AIB Cell contains 12 boundary scan single-bit registers (See Figure from the Intel AIB Spec below). The total number of boundary scan register bits therefore is 1080.

For identification purposes, the first register bit of the boundary scan register is labeled bit 0, and it receives data from jtag_scan_in. The last register bit of the boundary scan register is 1079, and it drives jtag_scan_out. Internally the registers are chained such that bit 0 drives bit 1, bit 1 drives bit 2, and so on. The entries in the tables are listed in the boundary scan register bit order, starting with bit 0, and ending with bit 1079.

The BSR infrastructure provides a JTAG INTEST function where signals to the external AIB Adapter from BSR output cell ports (i.e. odat_async, odat1, odat0) can be driven with the values from selected boundary scan register bits. Also, signal values from the AIB Adapter or Configuration Bits can be sampled and stored into BSR registers from the BSR input cell ports (i.e. rxen [2:0], txen, async_dat, idat1, idat0).

Using the JTAG BSR INTEST instruction, a pattern from the external AIB Adapter driving the tx_data [79:0] top-level input port of the PHY may be observed by scanning out the bits captured by the BSR chain and extracting and ordering those bits corresponding to tx_data[n] as indicated in the tables.

Likewise, as the tables illustrates, the JTAG INTEST outputs, driven to the external AIB Adapter from the PHY, comprise only the rx_data [79:0] top-level output ports from the PHY. A pattern scanned into the bits corresponding to rx_data[n] as indicated in the table below may be observed external to the PHY within the AIB Adapter.

ABP Configuration inputs (from DFFs internal to the TLRB AIB PHY) can also be captured by the BSR scan chain using the JTAG INTEST instruction. The APB labels in the tables below are the same as the APB configuration register bit field names in the register map except for the following:

TX_GROUP1 = TXEN_DAT_GROUP1 APB register map field.

TX_GROUP0 = TXEN_DAT_GROUP0 APB register map field.

RX_GROUP1 = RXEN_DAT_GROUP1 APB register map field.

RX_GROUP0 = RXEN_DAT_GROUP0 APB register map field.

AIB IO Cell BSR Chain Bit Range High : Low		AIB IO Cell ID (AIB<*ID*>)	AIB IO uBump Name	AIB JTAG INTEST BSR OUTPUTS TO ADAPTER					AIB JTAG INTEST BSR INPUTS FROM ADAPTER OR CONFIGURATION							
				bit offset: 11	10	9	8	7	6	5	4	3	2	1	0	
				odat_async	oclk	oclkb	odat1	odat0	rxen0	rxen1	rxen2	txen	async_dat	idat1	idat0	
			TX[*n*]									APB Config		tx_data[*n*]	tx_data[*n*]	
11	0	50	38	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	77	76	
23	12	51	39	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	79	78	
35	24	52	36	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	73	72	
47	36	53	37	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	75	74	
59	48	54	34	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	69	68	
71	60	55	35	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	71	70	
83	72	56	32	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	65	64	
95	84	57	33	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	67	66	
107	96	58	30	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	61	60	
119	108	59	31	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	63	62	
131	120	60	28	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	57	56	
143	132	61	29	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	59	58	
155	144	62	26	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	53	52	
167	156	63	27	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	55	54	
179	168	64	24	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	49	48	
191	180	65	25	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	51	50	
203	192	66	22	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	45	44	
215	204	67	23	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	47	46	
227	216	68	20	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	41	40	
239	228	69	21	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP1*	'0'	43	42	
251	240	0	18	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	37	36	
263	252	1	19	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	39	38	
275	264	2	16	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	33	32	
287	276	3	17	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	35	34	
299	288	4	14	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	29	28	
311	300	5	15	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	31	30	
323	312	6	12	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	25	24	
335	324	7	13	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	27	26	
347	336	8	10	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	21	20	
359	348	9	11	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	23	22	
371	360	10	TXCLK	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TXEN_CLK*	'0'	'1'	'0'	
383	372	11	TXCLKB	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TXEN_CLK*	'0'	'0'	'1'	
395	384	12	8	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	17	16	
407	396	13	9	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	19	18	
419	408	14	6	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	13	12	
431	420	15	7	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	15	14	
443	432	16	4	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	9	8	
455	444	17	5	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	11	10	
467	456	18	2	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	5	4	
479	468	19	3	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	7	6	
491	480	20	0	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	1	0	
503	492	21	1	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TX GROUP0*	'0'	3	2	
515	504	22	MRSTN	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TXEN_RST*	*RSTN*	'0'	'0'	
527	516	23	MARSTN	N/C	N/A	N/A	N/C	N/C	'0'	'1'	'0'	*TXEN_RST*	*ADAPTER_RSTN*	'0'	'0'	
539	528	24	SPARE[0]	N/C	N/A	N/A	N/C	N/C	*RXEN_RST*	*RXEN_RST*	*RXEN_RST*	'0'	'0'	'0'	'0'	
551	540	25	SPARE[1]	N/C	N/A	N/A	N/C	N/C	*RXEN_RST*	*RXEN_RST*	*RXEN_RST*	'0'	'0'	'0'	'0'	
563	552	26	SARSTN	*ADAP_RSTN_IN*	N/A	N/A	N/C	N/C	*RXEN_RST*	*RXEN_RST*	*RXEN_RST*	'0'	'0'	'0'	'0'	
575	564	27	SRSTN	*RSTN_IN*	N/A	N/A	N/C	N/C	*RXEN_RST*	*RXEN_RST*	*RXEN_RST*	'0'	'0'	'0'	'0'	

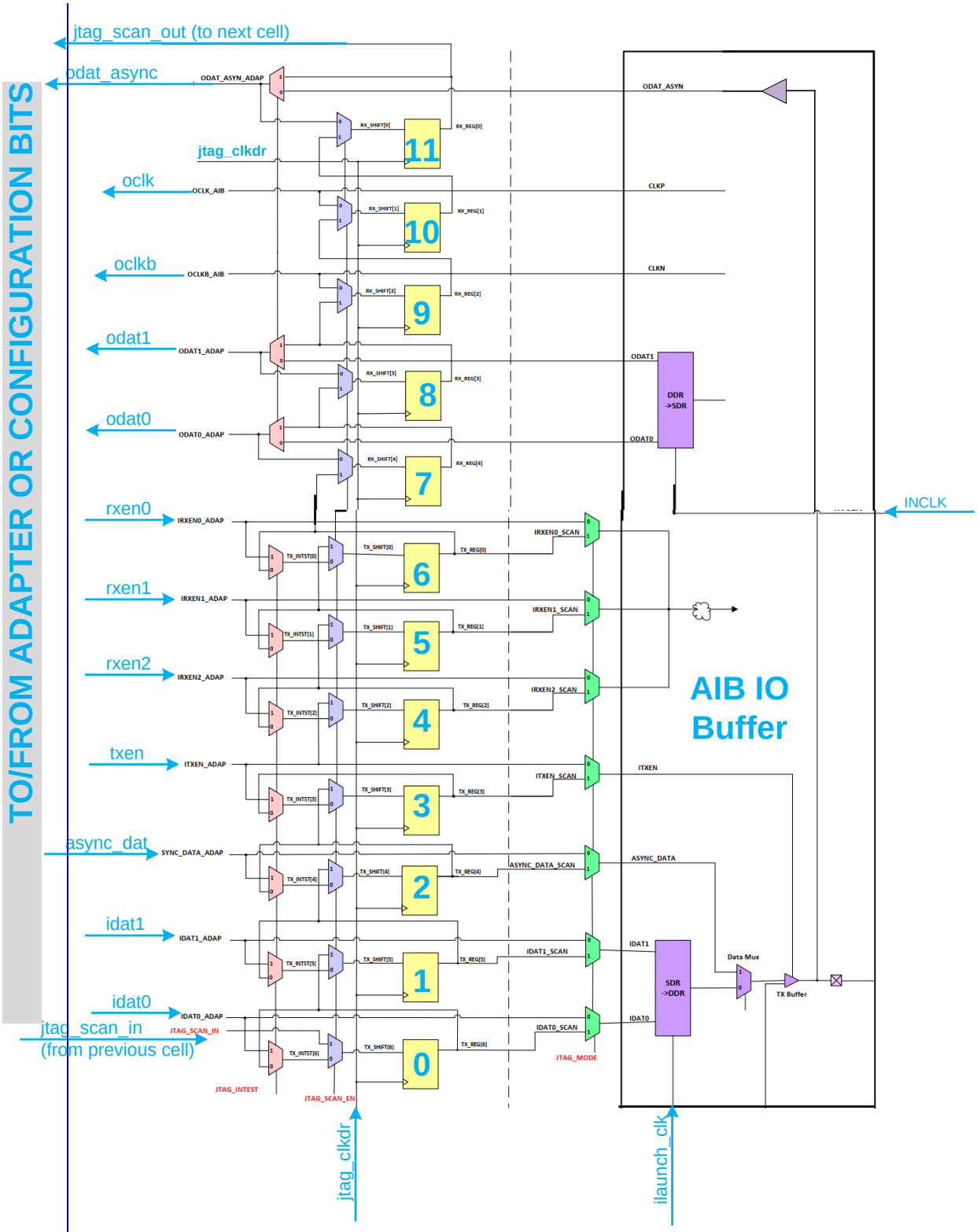
AIB IO Cell BSR Chain Bit Range High : Low		AIB IO Cell ID (AIB<*ID*>)	AIB IO uBump Name	AIB JTAG INTEST BSR OUTPUTS TO ADAPTER					AIB JTAG INTEST BSR INPUTS FROM ADAPTER OR CONFIGURATION						
				bit offset: 11	10	9	8	7	6	5	4	3	2	1	0
				odat_async	oclk	oclkb	odat1	odat0	rxen0	rxen1	rxen2	txen	async_dat	idat1	idat0
			RX[*n*]				rx_data[*n*]	rx_data[*n*]		APB Config					
587	576	**28**	1	N/C	N/A	N/A	3	2	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
599	588	**29**	0	N/C	N/A	N/A	1	0	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
611	600	**30**	3	N/C	N/A	N/A	7	6	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
623	612	**31**	2	N/C	N/A	N/A	5	4	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
635	624	**32**	5	N/C	N/A	N/A	11	10	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
647	636	**33**	4	N/C	N/A	N/A	9	8	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
659	648	**34**	7	N/C	N/A	N/A	15	14	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
671	660	**35**	6	N/C	N/A	N/A	13	12	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
683	672	**36**	9	N/C	N/A	N/A	19	18	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
695	684	**37**	8	N/C	N/A	N/A	17	16	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
707	696	**38**	RXCLKB	N/C	N/A	N/A	N/C	N/C	*RXEN_CLK*	*RXEN_CLK*	*RXEN_CLK*	'0'	'0'	'0'	'0'
719	708	**39**	RXCLK	N/C	N/A	N/A	N/C	N/C	*RXEN_CLK*	*RXEN_CLK*	*RXEN_CLK*	'0'	'0'	'0'	'0'
731	720	**40**	11	N/C	N/A	N/A	23	22	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
743	732	**41**	10	N/C	N/A	N/A	21	20	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
755	744	**42**	13	N/C	N/A	N/A	27	26	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
767	756	**43**	12	N/C	N/A	N/A	25	24	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
779	768	**44**	15	N/C	N/A	N/A	31	30	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
791	780	**45**	14	N/C	N/A	N/A	29	28	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
803	792	**46**	17	N/C	N/A	N/A	35	34	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
815	804	**47**	16	N/C	N/A	N/A	33	32	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
827	816	**48**	19	N/C	N/A	N/A	39	38	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
839	828	**49**	18	N/C	N/A	N/A	37	36	*RX GROUP0*	*RX GROUP0*	*RX GROUP0*	'0'	'0'	'0'	'0'
851	840	**70**	21	N/C	N/A	N/A	43	42	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
863	852	**71**	20	N/C	N/A	N/A	41	40	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
875	864	**72**	23	N/C	N/A	N/A	47	46	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
887	876	**73**	22	N/C	N/A	N/A	45	44	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
899	888	**74**	25	N/C	N/A	N/A	51	50	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
911	900	**75**	24	N/C	N/A	N/A	49	48	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
923	912	**76**	27	N/C	N/A	N/A	55	54	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
935	924	**77**	26	N/C	N/A	N/A	53	52	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
947	936	**78**	29	N/C	N/A	N/A	59	58	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
959	948	**79**	28	N/C	N/A	N/A	57	56	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
971	960	**80**	31	N/C	N/A	N/A	63	62	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
983	972	**81**	30	N/C	N/A	N/A	61	60	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
995	984	**82**	33	N/C	N/A	N/A	67	66	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1007	996	**83**	32	N/C	N/A	N/A	65	64	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1019	1008	**84**	35	N/C	N/A	N/A	71	70	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1031	1020	**85**	34	N/C	N/A	N/A	69	68	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1043	1032	**86**	37	N/C	N/A	N/A	75	74	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1055	1044	**87**	36	N/C	N/A	N/A	73	72	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1067	1056	**88**	39	N/C	N/A	N/A	79	78	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'
1079	1068	**89**	38	N/C	N/A	N/A	77	76	*RX GROUP1*	*RX GROUP1*	*RX GROUP1*	'0'	'0'	'0'	'0'

The figure below is a snippet *copied* directly from a figure in the Intel AIB Specification ("Figure 5-4: Boundary Scan Chain for Single Full AIB IO Block").

The AIB IO Block depicted in the figure corresponds to a single AIB IO Cell.

The figure has been annotated in light blue with labels corresponding to labels in the tables above so that the tables may be correlated with the figure.

Figure 26 BSR JTAG Bits per AIB IO



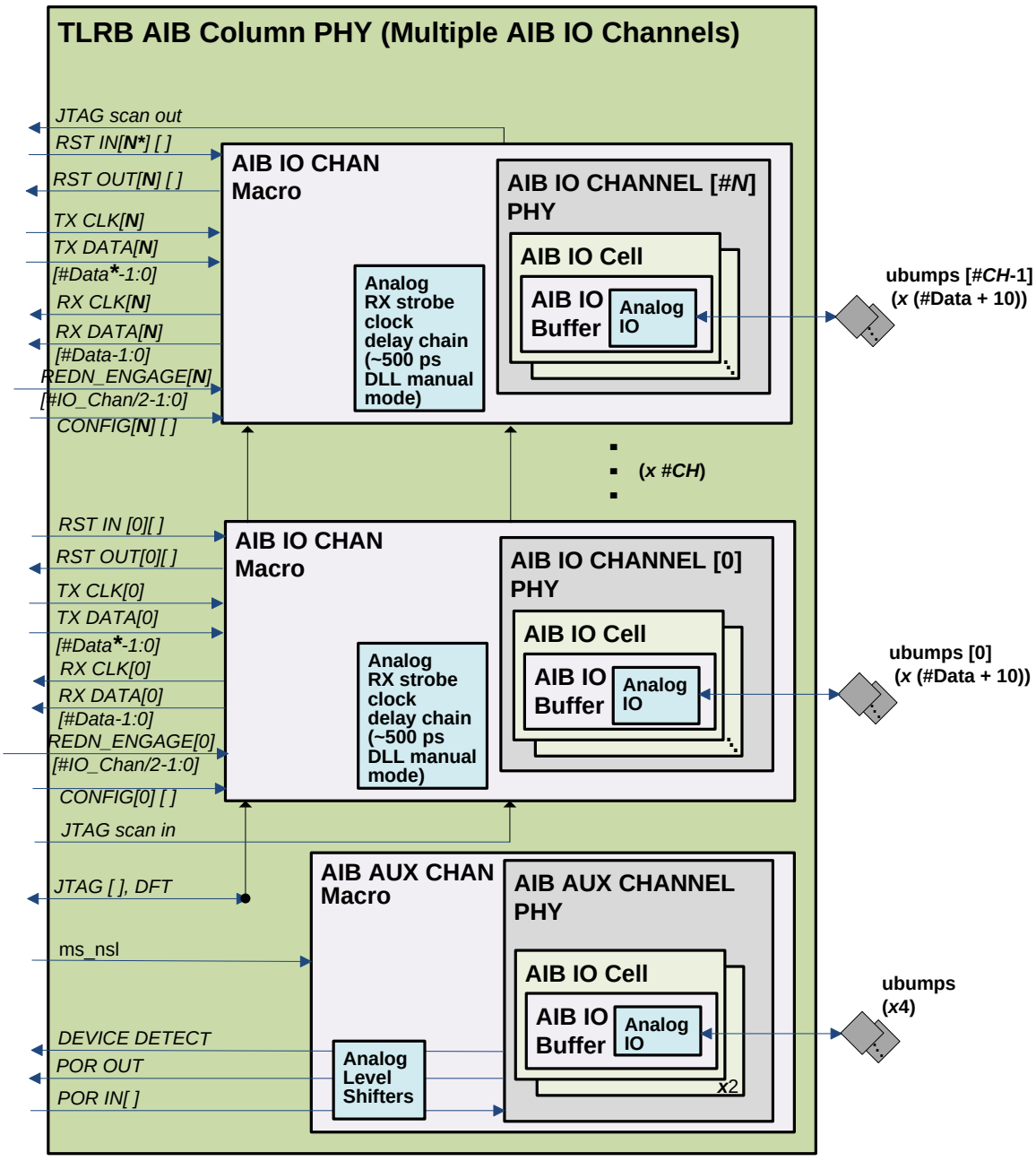
16. Multiple AIB IO Channels

The TLRB AIB PHY as it is described in the preceding sections implements a *single* (one) AIB IO channel and one AIB AUX channel. This section describes a further extension to the TLRB AIB PHY that implements multiple AIB IO channels, and one AUX channel.

The Intel AIB spec (snippet below) describes **6** possible (allowed) multi-channel implementations:

> "The total number of channels discretely varies from 1, 2, 4, 6, 12, 16, and 24, depending on total required bandwidth. The discrete number of channels is mainly to support common channel interfaces that enable plug and play capability. The redundancy register is only enabled to support up to 24 channels."

All of the AIB IO channels implemented within a multi-channel TLRB AIB PHY are identical instances (hard macros). Individually, they all have the same configuration and the same number of data signals.



* ***#Data*** is the number of synchronous Data microbumps per IO channel
#IO_Chan is the number of AIB IO Channel microbumps (**#Data** + 10)
#CH is the number of AIB IO Channels
N is the number of AIB IO Channels minus 1 (***#CH*** -1)

Signal name	Direction	Description (*NC* = # of AIB IO Channels)
		Synchronous Data (ND = # of synchronous Data uBumps; i.e. 80)
tx_data [*NC*-1:0] [*ND*-1:0]	Input	Synchronous (to tx_clk) data output transmitted to uBump This is the same as the "TX DATA" bus from the AIB Adapter to the AIB IOs as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
rx_data [*NC*-1:0] [*ND*-1:0]	Output	Synchronous (to rx_clk) data input received from uBump This is the same as the "RX DATA" bus from the AIB IOs to the AIB Adapter as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
tx_clk [*NC*-1:0]	Input	Transmit clock(s) for synchronous tx_data
rx_clk [*NC*-1:0]	Output	Receive clock(s) for synchronous rx_data
		IO Cell Configuration (asynchronous/static). ***NB = # of ubumps per AIB IO Channel excluding the spare pair (i.e. 88)***
iddr_enable [*NC*-1:0]	Input	DDR mode select (1=enable DDR mode)
idat_selb [*NC*-1:0] [*NB*-1:0]	Input	Asynchronous TX mode select for each AIB IO cell (1= select async_data)
ipdrv [*NC*-1:0] [1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Data ubumps.
indrv [*NC*-1:0] [1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Data ubumps.
ipdrv_clk [*NC*-1:0] [1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Clock ubumps.
indrv_clk [*NC*-1:0] [1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Clock ubumps.
ipdrv_rst [*NC*-1:0] [1:0]	Input	ubump TX drive strength high (P-driver) selection. Common to all TX IO channel Reset ubumps.
indrv_rst [*NC*-1:0] [1:0]	Input	ubump TX drive strength low (N-driver) selection. Common to all TX IO channel Reset ubumps.
rxen [*NC*-1:0] [*NB*-1:0][2:0]	Input	Receive enable selection for each AIB IO cell (encoded per AIB spec tables)
txen[*NC*-1:0] [*NB*-1:0]	Input	Transmit enable for each AIB IO cell (1= TX enabled)
sdr_dly_adjust [*NC*-1:0] [9:0]	Input	Adjustment setting for programmable RX inclk delay (DLL manual mode) for SDR mode
ddr_dly_adjust [*NC*-1:0] [9:0]	Input	Adjustment setting for programmable RX inclk delay (DLL manual mode) for DDR mode
		Reset and Ready Signaling
adap_irstb[*NC*-1:0]	Input	AIB PHY reset IRSTB input. Resets *this* local AIB PHY by driving the internal irstb signal; active low
rstn_in [*NC*-1:0]	Input	AIB PHY reset input. (Source of ms_rstn signal driven to the uBump; active low)
rstn_out [*NC*-1:0]	Output	AIB PHY reset output (Derived from sl_rstn signal received from the uBump; active low)
adap_rstn_in [*NC*-1:0]	Input	Adapter reset input (Source of ms_adapter_rstn signal driven to the uBump; active low)
adap_rstn_out [*NC*-1:0]	Output	Adapter reset output (Derived from sl_adapter_rstn signal received from the uBump; active low)
		Redundancy (asynchronous/static) ***TNB = Total # of ubumps per AIB IO Channel (i.e. 90)***
redun_engage [*NC*-1:0] [(*TNB/2*)-1: 0]	Input	Redundancy engage enable for Channel 0 for each *pair* of AIB IO Channel IO Buffers Selects source (1=redundant/0=normal) of clock, data, and configuration inputs to AIB IO Cell. Also selects the source of odat*outputs from the AIB IO Cell pair.

Relative to TLRB AIB PHY Extended implementation, there are some APB memory mapped registers that are replicated per AIB IO Channel as in the table below. Also, some of the top level ports of the TLRB AIB PHY Extended implementation are expanded in dimension as per AIB IO Channel as in the table below.

A single APB Slave block corresponding to a single APB Slave interface configures and controls all of the AIB IO Channels.

A single AIB JTAG TAP Controller block corresponding to a single JTAG Interface accesses all of the BSR infrastructure chains within all of the AIB IO Channels. Internal to the TLRB AIB PHY, the JTAG scan_out of the BSR chain of channel number 0 drives the JTAG scan_in of BSR chain of channel number 1. The output of channel 1 drives the input of channel 2, and so on. The last AIB IO channel (highest numbered) drives the JTAG scan out top level port.

Step 2 ("Configuration") from the "Bring Out of Reset Sequence" is required to be repeated (in any channel order) for each AIB IO Channel.

Likewise, Step 4 ("Reset signals released") is required to be repeated (in any channel order) for each AIB IO Channel.

Register Name / Link to Full Description	Short Register Description	Offset Address	AIB Type
TX_DRV_STRENGTH	TX Drive Strength	0x000 + (0x20 *x* Channel Number)	AIB IO Channel Registers *
TX CONFIG	TX Configuration	0x004 + (0x20 *x* Channel Number)	
RX_ENABLE	RX Enable	0x008 + (0x20 *x* Channel Number)	
RX_DELAY_ADJUST	RX Clock Delay Adjustment	0x00C + (0x20 *x* Channel Number)	
REPAIR_ADDR	Repair Information	0x010 + (0x20 *x* Channel Number)	
RESETS	Resets (Configuration and Status)	0x014 + (0x20 *x* Channel Number)	

* For example, the APB addresses for Channel Number 0 are: 0x000, 0x004, 0x008, 0x00C, 0x010, 0x014.
While the APB addresses for Channel Number 23 are: 0x2E0, 0x2E4, 0x2E8, 0x2EC, 0x2F0, 0x2F4.

Signal name	Direction	Description (*NC* = # of AIB IO Channels)
		Synchronous Data (ND = # of synchronous Data uBumps; i.e. 80)
tx_data[*NC*-1:0] [*ND*-1:0]	Input	Synchronous (to tx_clk) data output transmitted to uBump This is the same as the "TX DATA" bus from the AIB Adapter to the AIB IOs as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
rx_data[*NC*-1:0] [*ND*-1:0]	Output	Synchronous (to rx_clk) data input received from uBump This is the same as the "RX DATA" bus from the AIB IOs to the AIB Adapter as depicted in the Intel AIB Spec (Figure 3-48: AIB Adapter Block Diagram)
tx_clk[*NC*-1:0]	Input	Transmit clock for synchronous tx_data
rx_clk[*NC*-1:0]	Output	Receive clock for synchronous rx_data
		Configuration and status outputs for system use (synchronous to APB Clock)
r_apb_iddr_enable [*NC*-1:0]	output	APB register bit field (IDDREN_DAT). Control/configuration bit to local AIB PHY.
r_apb_ns_rstn [*NC*-1:0]	output	APB register bit field (RSTN). Control bit to near-side, local AIB PHY. Drives MRSTN uBump.
r_apb_ns_adap_rstn [*NC*-1:0]	output	APB register bit field (ADAPTER_RSTN). Control bit to near-side, local AIB PHY. Drives MARSTN uBump.
r_apb_fs_rstn [*NC*-1:0]	output	APB register bit field (RSTN_IN). Status bit from far-side, remote AIB PHY. Received SRSTN uBump synchronized to pclk.
r_apb_fs_adap_rstn [*NC*-1:0]	output	APB register bit field (ADAPTER_RSTN_IN). Status bit from far-side, remote AIB PHY. Received SARSTN uBump synchronized to pclk.
		ATPG/DFT
atpg_bsr_ovrd_mode [*NC*-1:0]	Input	Intel AIB Spec: "There will be 1 chain [ATPG-BSR] per channel."
atpg_bsr_scan_shift [*NC*-1:0]	Input	
atpg_bsr_scan_shift_clk [*NC*-1:0]	Input	
atpg_bsr_scan_in [*NC*-1:0]	Input	
atpg_bsr_scan_out [*NC*-1:0]	Output	
dig_test_sel [$clog2 (*NC*):0]	Input	The most significant bit (MSb) selects the AIB IO channels (=1) vs. the AUX channel (=0). The least significant bits, [$clog2 (NC)-1:0], select a particular AIB IO channel when the MSb is set to a 1.

16.1. Multiple AIB IO Channel RTL Sources

The TLRB AIB PHY that supports multiple AIB IO Channels *requires additional, different System Verilog source files/modules* relative to the single AIB IO Channel PHY. All of the added source files include a Verilog parameter port named "NCH" which indicates the number of AIB IO channels to implement. All of the added sources are named with "_mc" (Multi-Channel) postfixes. All of the added sources expand their relevant IO ports dimensions to NCH channels. The table below provides the complete list of added source files.

Source File Name	Summary Difference Relative to Single AIB IO Channel
itrx_aib_phy_apbs_mc.sv	Replicates the per-channel APB registers for each of the NCH channels.
itrx_aib_phy_ext_mc.sv	Instantiates itrx_aib_phy_mc.sv.
itrx_aib_phy_mc.sv	Replicates and connects the AIB IO channel hard macro for each of the NCH channels.
tlrb_aib_phy_ext_mc.sv	**Top-level** that instances tlrb_aib_phy_mc.sv.
tlrb_aib_phy_mc.sv	Instantiates itrx_aib_phy_ext_mc.sv.

For new designs, **the recommended RTL top level module, and RTL configuration** are as indicated by the following example tool string:

```
"tlrb_aib_phy_ext_mc.sv +define+AIB_ID_REMAP +define+AIB_ID_REMAP_IOCHAN
+define+ITRX_AIB_JTAGSM –defparam NCH=<Number of AIB IO channels>"
```