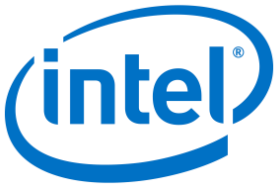


Advanced Interface Bus (AIB) Specification

2019.4.18

Revision 1.1

Revision History

Date	Version	Summary Of Changes
10/26/18	1.0	Initial version
04/18/19	1.1	Typographical fixes Minor wording changes Clarification of functionality Additional bump array patterns for medium and high-density bump pitches

Table of Contents

Table of Figures

Table of Tables

Glossary and Acronyms

This section defines phrases and acronyms that are not defined within the specification.

assert	To make a signal TRUE or active. For an active-high signal, this means asserting it HI; for an active-low signal, it means asserting it LO.
boustrophedon	Describes a back-and-forth motion. Literally means, “As the ox plows.”
bump	A die/packaging interconnect scheme that reflows solder bumps to make a connection from the die to the substrate to which it’s being mounted. There may be a variety of sizes and means of implementing the bump. In this specification, bumps are implemented as microbumps. The terms “bump” and “microbump” are used interchangeably.
CDM	Charged-device model for ESD
chiplet	A passivated bare die intended to be mounted on a substrate like an interposer.
compliance mask	Used to specify requirements on an eye diagram.
DCC	Duty-cycle correction. Used to ensure that the duty cycle of a signal remains within specification.
DCD	Duty-cycle distortion. Refers to effects that may result in a signals duty cycle being out of specification.
DDR	Double data rate. Data is captured on both edges of the clock.
de-assert	To make a signal FALSE or inactive. For an active-high signal, this means de-asserting it LO; for an active-low signal, it means de-asserting it HI.
DLL	Delay-locked loop. A circuit that can synthesize multiple frequencies and phase relationships from a single periodic signal. A digital version of a phase-locked loop (PLL).
quasi-differential signal	A signal implemented over two wires having opposing polarities. Similar to analog differential signals, but created using a pair of digital signals.
ESD	Electrostatic discharge

eye diagram	Used to illustrate high-speed signal switching in a manner that allows specification and compliance testing of the signal characteristics.
handshake	A protocol that allows two sides of an interface to be connected, configured, and synchronized.
HI	A logic *high* level, equivalent to data value b'1. In the context of this specification, it is implemented with a high voltage value.
I/O	Input/output. In the context of this specification, refers to inputs and outputs collectively.
JTAG	Joint Test Action Group. Refers to a standard for the testing of chip-to-chip interconnections as well as select internal chip testing. The control and observability afforded by the standard may be used for more than just testing.
latency	The time required for a signal to traverse a given path. Equivalent to "delay."
LO	A logic *low* level, equivalent to data value b'0. In the context of this specification, it is implemented with a low voltage value.
LSB	Least-significant bit. In the context of a shift register, the bit with the lowest number.
MAC	Media Access Controller. The lowest-level of the Link Layer, which is the layer above the PHY layer in the OSI reference model.
microbump	A bump interconnect scheme with very small bumps. Suitable for mounting a die onto a substrate like an interposer, where fine lines can be created, but not for a printed circuit board. In this specification, the terms "bump" and "microbump" are used interchangeably.
MSB	Most-significant bit. In the context of a shift register, the bit with the highest number.
OSI	Open Systems Interconnection. A project undertaken at the International Organization for Standardization (ISO), one outcome of which is the OSI reference model for networking.
overshoot	For the purposes of this specification, refers to a rising signal that exceeds the target static voltage before correcting itself

	and ringing down to the target level.
PHY	Physical layer. The bottom-most layer of the OSI reference model. It specifies electrical and basic signaling requirements.
redundancy	A technique for providing multiple instances of a resource for use in case of the failure of one of them. In the context of this specification, refers to techniques that can be implemented at or before testing and that are activated on power-up.
retiming	Refers to insertion or relocation of one or more flip-flops within a data path for the purpose of optimizing timing, power, and/or area.
RX	Receive. Applies to the inputs of a communications interface.
SDR	Single data rate. Data is captured on one edge of the clock.
sideband	Refers to signals that are not carried over the primary communication channels, but rather through auxiliary channels created specifically for those signals.
to forward	In the context of a communications link, refers to a control signal – often the clock – sent from one side of the link to the other.
TX	Transmit. Applies to the outputs of a communications interface.
UI	Unit interval. The time, relative to the clock period, required to transmit a symbol. Allows timing specifications that are independent of a specific clock frequency.
undershoot	For the purposes of this specification, refers to a falling signal that goes below the target static voltage before correcting itself and ringing down to the target level.
weak pull-up/down	An active or passive component connecting a signal to V_{DD}/ground to ensure that the signal, when not driven, will not float. The signal will be pulled to a HI/LO value. The term “weak” indicates that the pull-up/down component will pass limited current.

Note on Language

In order to provide clarity on normative language as distinct from informative language, the following indicates the use of modal verbs:

- “Shall” indicates a requirement. Failure to meet the requirement will result in non-compliance.
- “May” indicates an option. Implementation of an option will result in compliance.
- “Should” indicates a strong suggestion for something outside the scope of the specification. Failure to implement the suggestion will not result in non-compliance.
- The lack of one of the above modal verbs indicates informative language.

1 Introduction

1.1 Objective

This specification describes the Advanced Interconnect Bus (AIB) architecture, interconnect attributes, signal management, and the configuration interface required to design and build systems and peripherals that are compliant with the AIB specification. The AIB is intended for interconnecting chiplets mounted within a package with signal distances of 10 millimeters or less (often referred to as Ultra-Short Reach). While no maximum interconnect distance is specified, AIB signal electrical requirements detailed in Section 4 shall be met. AIB is not intended for signals interconnecting or connecting to external package pins.

The specification is intended to provide interoperation between compliant chiplets. Choices made by the chiplet designer with respect to such elements as number and type of data signals, maximum supported clock speed, and any functionality that exceeds the minimum requirements established in the AIB specification should be documented in the chiplet datasheet.

AIB is a physical interconnect. Link protocol and application layers are implemented on top of the AIB interface.

1.2 Compliance Summary

Table 1 summarizes the compliance points that shall be met in order to meet the AIB requirements. Each of the compliance points is discussed in the specification.

Compliance Point	Required		Optional
	AIB Base	AIB Plus	
AC Parameters (Section 4)	x	x	
JTAG boundary scan (Section 5)	x	x	
Redundancy (Section 3.2.4)	x	x	
Bump assignment (Section 6.3)	x	x	
AUX	x	x	

Compliance Point	Required		Optional
	AIB Base	AIB Plus	
(Section 1.3.4.4)			
SDR (Section 0)	x	x	
DDR (Section 2.1.3.2)		x	
Latency specification (Section 2.1.7)	x	x	
Free-running clock (Section 2.2.2.2)		x	
AIB Adapter (Section 2.2)		x	
Data-transfer ready (Section 3.1)	x	x	
Adapter reset (Section 3.2.3.1.1)		x	
DLL (Section 2.1.6.7)			x
DCC (Section 2.1.6.6)			x
Adapter retiming register (Section 2.2.1)		x	
Conf_done (Section 3.2.2.3.5)	x	x	

Table 1. Design Feature Summary

1.3 Architecture

The AIB implements a physical-layer, or PHY, interconnect scheme, occupying Layer 1 on the OSI Reference Model.

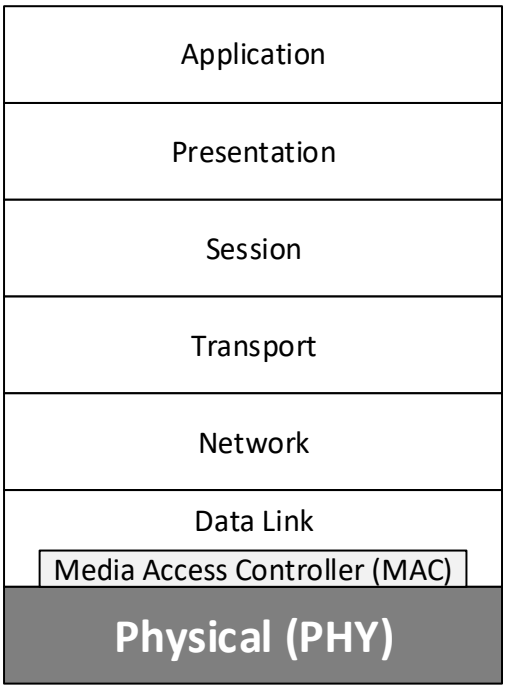


Figure 1. AIB in the OSI Reference Model

1.3.1 AIB Configurations

There are two AIB configurations: AIB Base and AIB Plus. AIB Base is a simpler interface; AIB Plus is intended for maximal performance.

AIB Base and AIB Plus configurations are not intended to be interconnected as two sides of a link due to signals required by AIB Plus configurations that are not present in AIB Base configurations.

There is no maximum data transmission frequency specified for AIB. However, in the interest of interoperability, a minimum data transmission frequency is specified (Sections 1.3.1.1 and 1.3.1.2) in order to provide a range of frequencies likely to be supported by multiple chiplets..

1.3.1.1 AIB Base

An AIB Base implementation shall support a minimum clock rate no greater than 1 Gigahertz (GHz) and a minimum data rate no greater than 1 Gigabit per second (Gbps). Data shall be transferred using a single data rate (SDR) format (Section 0). The full range of the operating clock and data rate should be documented in the chiplet data sheet.

1.3.1.2 AIB Plus

An AIB Plus implementation shall support a minimum clock rate no greater than 1 GHz and a minimum data rate no greater than 2 Gbps. Data shall be transferred using a double data rate (DDR) format (Section 2.1.3.2) or an SDR format. The full range of the operating clock and data rate should be documented in the chiplet data sheet.

AIB Plus implementations shall include an AIB Adapter block, specified in Section 2.2.

AIB Base and AIB Plus are compared in Table 2

Configuration	Clock/data relationship	Minimum Operating Clock Rate	Minimum Operating Data Rate
AIB Base	SDR	≤ 1 GHz	≤ 1 Gbps
AIB Plus	SDR	≤ 1 GHz	≤ 1 Gbps
	DDR	≤ 1 GHz	≤ 2 Gbps

Table 2. Clock and Data Rates

1.3.2 Near-Side and Far-Side Interfaces

AIB specifies an interface for one chiplet that can be connected to a compatible interface on a different chiplet. An AIB interface is one side of the connection. A chiplet designer may be creating only one of those AIB interfaces; the AIB interface to which it will connect is assumed to have been created by a different designer, putting it beyond the control of the designer.

In order to provide a clear distinction between behavior that the designer shall implement and behavior that the designer shall expect from the AIB interface to which it will connect, the AIB interface being created will be referred to as the Near-Side interface. The AIB interface to which the near-side interface will connect will be called the Far-Side interface. Signal names that reflect the signal origin will be prefixed with *ns_* or *fs_*, respectively.

Figure 2 illustrates Chiplet A being designed according to the AIB specification. It will eventually connect to Chiplet B. In this case, Chiplet A is the near side, and Chiplet B is the far side.

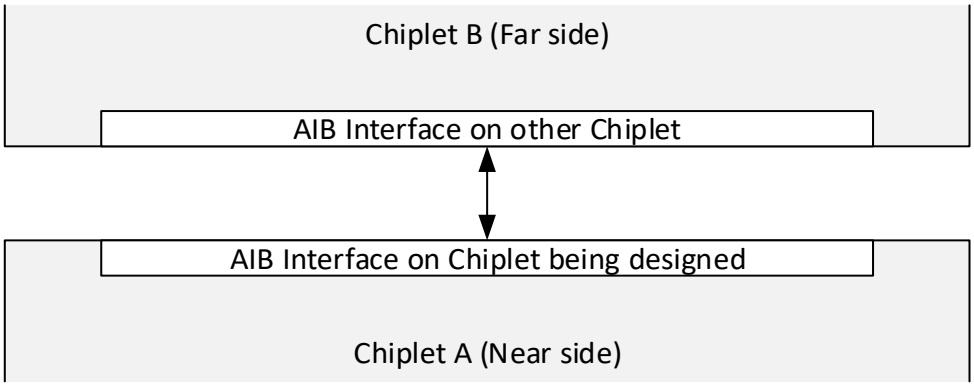


Figure 2. Near Side and Far Side

1.3.3 Master, Slave, and Dual-Mode Interfaces

An AIB interface pair has a master side and a slave side. Masters and slaves shall have specific roles only during initialization. Specifically:

- The master shall be responsible for providing a free-running clock signal (Section 2.2.2.2).
- The master shall provide a *device_detect* signal (Section 3.2.1).
- The slave shall provide a *power_on_reset* signal (Section 3.2.1)
- The master and slave shall have differently sized Sideband Control Signal shift registers (Section 2.2.2).
- The AIB Adapter (AIB Plus only) shall behave differently for masters and slaves during initialization (Section 2.2).

An AIB interface configured as a master shall be designed to connect to a slave; an AIB interface configured as a slave shall be designed to connect to a master.

The master/slave property of an interface is independent of the near-side/far-side property. A near-side interface can be either a master or slave, as can a far-side interface.

An AIB interface shall be configured as master or slave by one of the following two means:

- By designing the interface as a master or slave, referred to as a fixed interface.
- By implementing a dual-mode interface that can be configured as master or slave on power-up.

1.3.3.1 Fixed Interfaces

For fixed interfaces, the configuration of an interface as master or slave shall be implemented by the chiplet designer and should be documented in the chiplet data sheet.

A chiplet may have one or more masters, one or more slaves, or a mixture of masters and slaves.

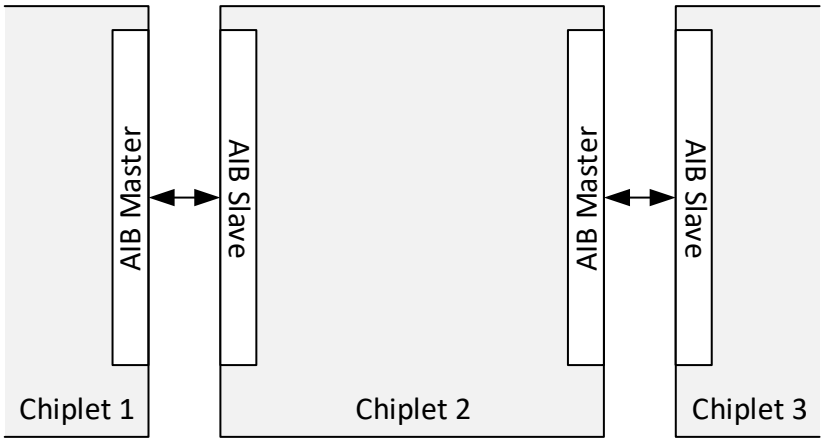


Figure 3. Fixed Master and Slave Interfaces

1.3.3.2 Dual-Mode Interfaces

A chiplet may implement both master and slave capability on an AIB interface, with master or slave configured prior to operation. This promotes greater interoperability, since the chiplet may connect to chiplets that have either master or slave AIB interfaces. Such interfaces are referred to as *dual-mode* interfaces. Any actions required to configure the dual-mode interface should be documented in the chiplet data sheet.

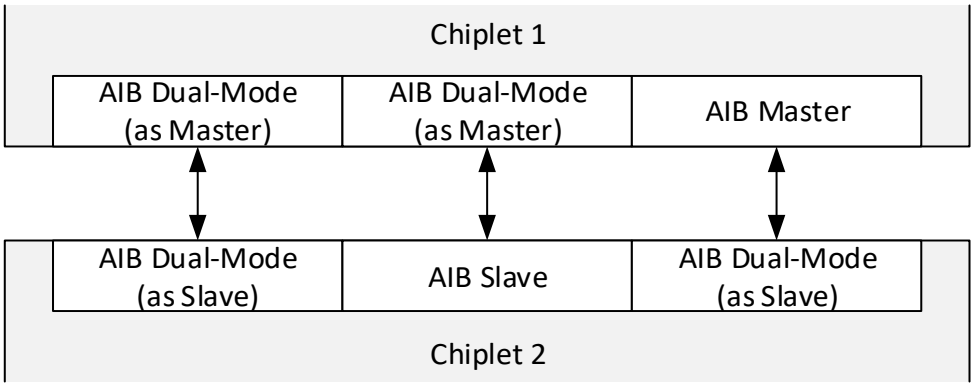


Figure 4. Dual-Mode Interfaces

Dual-mode interfaces shall implement a *dual_mode_select* signal that configures the interface either as a master or as a slave. The interface shall be configured as a master when the *dual_mode_select* signal is HI and as a slave when the *dual_mode_select* signal is LO. The selected mode shall be configured and stable before the end of the power-on reset phase (Section 3.2.1) and not during the configuration phase (Section 3.2.2).

1.3.4 AIB Interface

The AIB interface defines four signal types:

- Data signals
  - Inputs (*RX*): data input signals received by the interface

    - Outputs (*TX*): data output signals transmitted from the interface
- Clocks
    - Data clock out (*ns_fwd_clk, ns_rcv_clk*), sent to the receiving chiplet
    - Data clock in (*fs_fwd_clk, fs_rcv_clk*): received from the receiving chiplet
    - Free-running clock (AIB Plus only) (*ms_sr_clk, sl_sr_clk*): used by the sideband control signals
- Sideband control (AIB Plus only): used to implement calibration handshake.
- Asynchronous
    - *Power-on reset*: indicates whether a chiplet has completed power-on reset
    - *Device_detect*: used to verify presence of master
    - *ns_mac_rdy, fs_mac_rdy*: used to communicate that the MAC is ready for data transmission
    - *ns_adapter_rstn, fs_adapter_rstn* (AIB Plus only): used to reset the AIB Adapter registers, the AIB I/O register, and the calibration circuits.

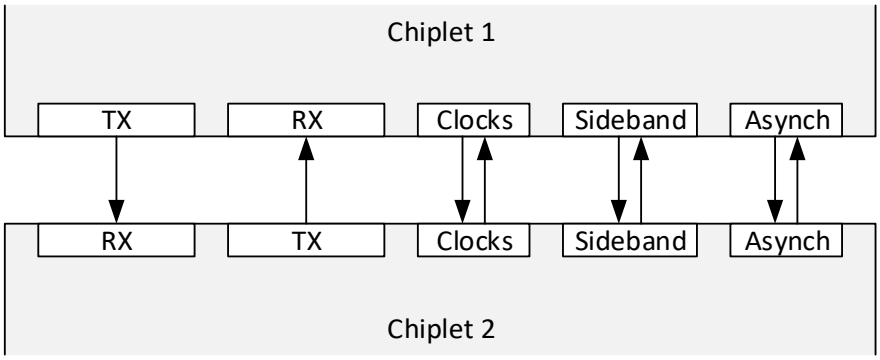


Figure 5. AIB Signal Types

Two interconnected chiplets will have the same set of signals. Signals from the near and far sides will be connected, such as *ns_mac_rdy* and *fs_mac_rdy*. Near-side signals should not be interconnected (e.g., *ns_mac_rdy* from one chiplet should not be connected to *ns_mac_rdy* from the other chiplet); far-side signals should not be interconnected (e.g., *fs_mac_rdy* from one chiplet should not be connected to *fs_mac_rdy* from the other chiplet. This is indicated in Figure 6, but the crossed wires are for illustration only. Microbump locations (Section 6.3) ensure that connections will be straight lines.

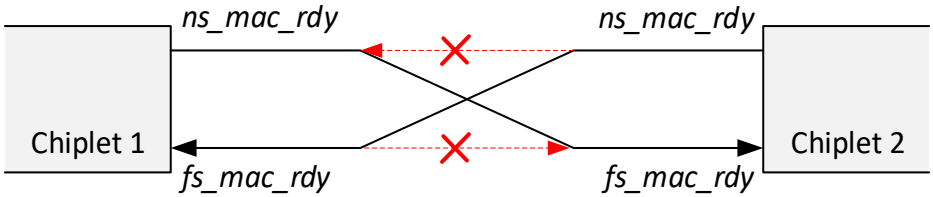


Figure 6. Interconnecting Near-Side and Far-Side Signals

Signal	Description	Present in AIB Base	Present in AIB Plus
TX	Synchronous data transmitted from the near side. (Section 2.1.1)	X	X
RX	Synchronous data received from the far side. (Section 2.1.2)	X	X
ns_fwd_clk/ns_fwd_clkb	Near-side transfer clock, forwarded from the near side to the far side for capturing received data. (Section 2.1.6.3)	X	X
fs_fwd_clk/fs_fwd_clkb	Far-side transfer clock, forwarded from the far side to the near side for capturing received data. (Section 2.1.6.3)	X	X
ns_rcv_clk/ns_rcv_clkb	Receive-domain clock forwarded from the far side to the near side for transmitting data from the near side. (Section 2.1.6.4)		X
fs_rcv_clk/fs_rcv_clkb	Receive-domain clock forwarded from the near side to the far side for transmitting data from the far side. (Section 2.1.6.4)		X
ns_sr_clk/ns_sr_clkb	Forwarded serial shift register clock from near side to far side chiplet, driven by free running clock. (Section 2.2.2.2)		X
fs_sr_clk/fs_sr_clkb	Forwarded serial shift register clock from far side to near side chiplet, driven by free running clock. (Section 2.2.2.2)		X
ns_sr_data	Time-multiplexed sideband-control data from near side to far side. (Section 2.2.2.5)		X
fs_sr_data	Time-multiplexed sideband-control data from far side to near side. (Section 2.2.2.5)		X
ns_sr load	Sideband control load signal from near side to far side. (Section 2.2.2.3)		X
fs_sr_load	Sideband control oad control signal from far side to near side. (Section 2.2.2.3)		X
ns_mac_rdy	Data-transfer-ready signal from near side to far side. (Section 3.1)	X	X
fs_mac_rdy	Data-transfer-ready signal from far side to near side. (Section 3.1)	X	X

Signal	Description	Present in AIB Base	Present in AIB Plus
ns_adapter_rstn	Asynchronous adapter reset signal from near side to far side (Section 3.2.3.3.1)		X
fs_adapter_rstn	Asynchronous adapter reset signal from far side to near side. (Section 3.2.3.3.1)		X

Table 3. AIB Interface Signals in AIB Channel

1.3.4.1 AIB Channel

AIB signals shall be grouped into AIB Channels. An AIB channel shall have a number of data signals not to exceed a limit determined by the microbump pitch of the chiplet.

The data signals may consist of half TX and half RX signals (“balanced” interface), all TX signals (“all-TX” interface) or all RX signals (“all-RX” interface).

Microbump pitch (µm)	Range of data signals per channel (Increments)					
	TX			RX		
	Balanced	All-TX	All-RX	Balanced	All-TX	All-RX
≤55	20-80 (20)	20-160 (20)	0	20-80 (20)	0	20-160 (20)
10	20-320 (20)	20-640 (20)	0	20-320 (20)	0	20-640 (20)

Table 4. Number of Data Signals per Channel

1.3.4.2 AIB Column

AIB channels shall be grouped into an AIB column. All AIB channels within a column shall have the same configuration (Base or Plus) and the same number of data signals.

A column shall additionally include an AUX block adjacent to the first channel.

Block type	Number allowed per column
Channels	1, 2, 4, 8, 12, 16, 24
AUX blocks	1

Table 5. Number of Channels per Column

.

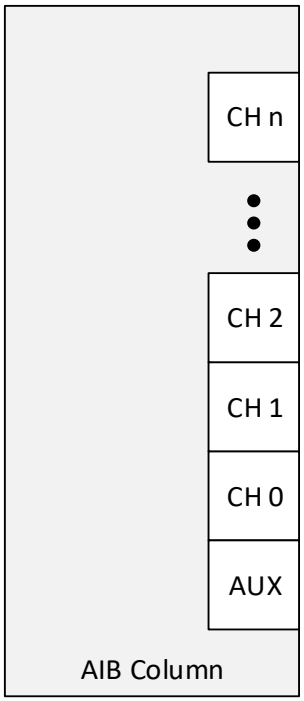


Figure 7. An AIB Column

1.3.4.3 AIB data paths

Each data path within a channel shall comprise an I/O Block (which shall be configurable up to four different ways (Section 2.1) and, for AIB Plus implementations, an AIB Adapter (Section 2.2).

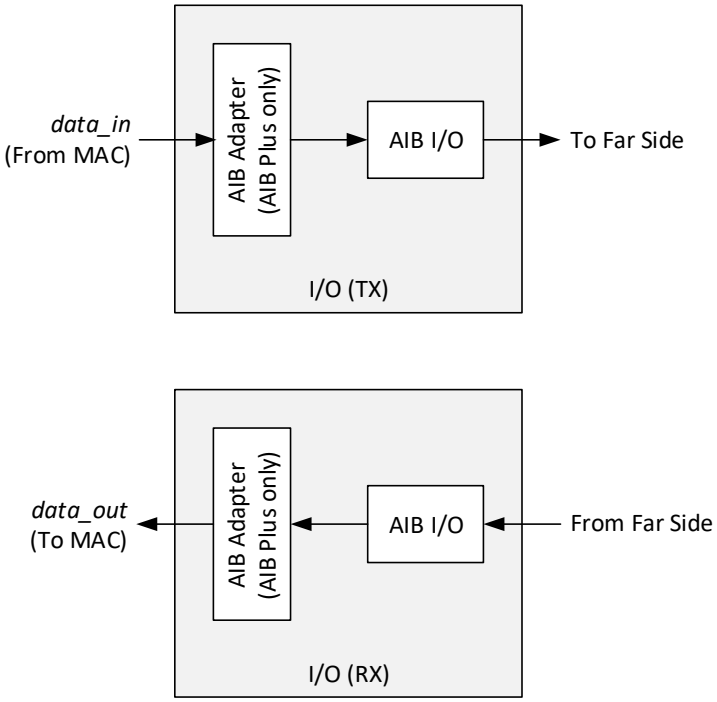


Figure 8. AIB Data Signals

1.3.4.4 AUX Block

The AUX block shall consist of two signals: *power_on_reset* and *device_detect*. These signals shall be used during power-up initialization (Section 3.2.1). There shall be one AUX block per column.

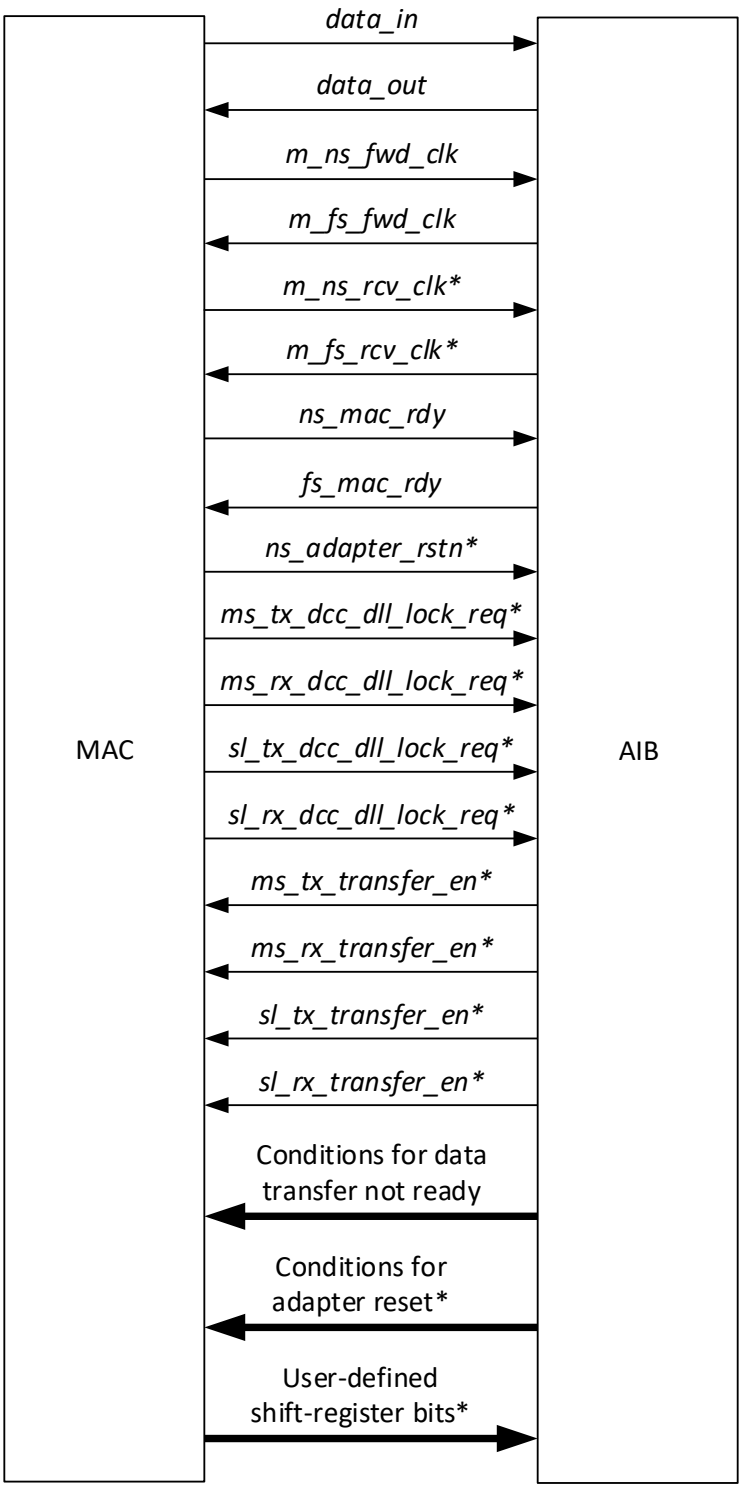
1.3.5 AIB-to-MAC Interface

The following signals shall constitute the interface to the MAC.

Signals	In (from MAC) Out (to MAC)	Description
data_in	In	For transmitting across the AIB link (Section 2.1.1)
data_out	Out	Received through the AIB link (Section 2.1.2)
m_ns_fwd_clk	In	For transmitting data from the near side to the far side (Section 2.1.6)
m_fs_fwd_clk	Out	Received from the far side and converted from quasi-differential to single-ended (Section 2.1.6.3)
m_ns_rcv_clk (AIB Plus only)	In	For capturing received data from the far side (Section 2.1.6)
m_fs_rcv_clk (AIB Plus only)	Out	Received from the far side and converted from quasi-differential to single-ended (Section 2.1.6.4).
ns_mac_rdy	In	For resetting near-side data transfers and communicating MAC readiness for calibration to the far side (Section 3.1)
fs_mac_rdy	Out	Indicates that the far-side MAC is ready to transmit data (Section 3.1)
ns_adapter_rstn (AIB Plus only)	In	Resets the AIB Adapter (section 3.1)
ms_rx_dcc_dll_lock_req *ms_tx_dcc_dll_lock_req* (AIB Plus only)	In	Initiates calibration of transmit and receive paths for a master interface (Section 3.2.3.3)
sl_rx_dcc_dll_lock_req *sl_tx_dcc_dll_lock_req* (AIB Plus only)	In	Initiates calibration of transmit and receive paths for a slave interface (Section 3.2.3.3)
ms_tx_transfer_en *ms_rx_transfer_en* (AIB Plus only)	Out	Indicate that calibration on the master is complete for transmit and receive paths (Section 3.2.3.3)

sl_tx_transfer_en *sl_rx_transfer_en* (AIB Plus only)	Out	Indicate that calibration on the slave is complete for transmit and receive paths (Section 3.2.3.3)
Signals indicating any conditions that may cause de-assertion of data-transfer ready	Out	Sent to MAC for possible data-transfer ready de-assertion by MAC (Section 3.1)
Signals indicating any conditions that may cause AIB Adapter reset and recalibration (AIB Plus only)	Out	Sent to MAC for possible adapter reset by MAC (Section 3.2.3)
User-defined shift-register bits (AIB Plus only)	In	For transmitting application-defined bits to the far side (Section 2.2.2.5.2)

Table 6. MAC Interface



*AIB Plus only

Figure 9. MAC Interface

2 Functional Specification

2.1 I/O Blocks

I/O blocks are divided into one of two types: *RX* or *TX*.

2.1.1 TX Block

A TX block shall register data before transmission. A DDR output (AIB Plus only; Section 2.1.3.2) shall combine two single-rate data streams into one double-rate stream.

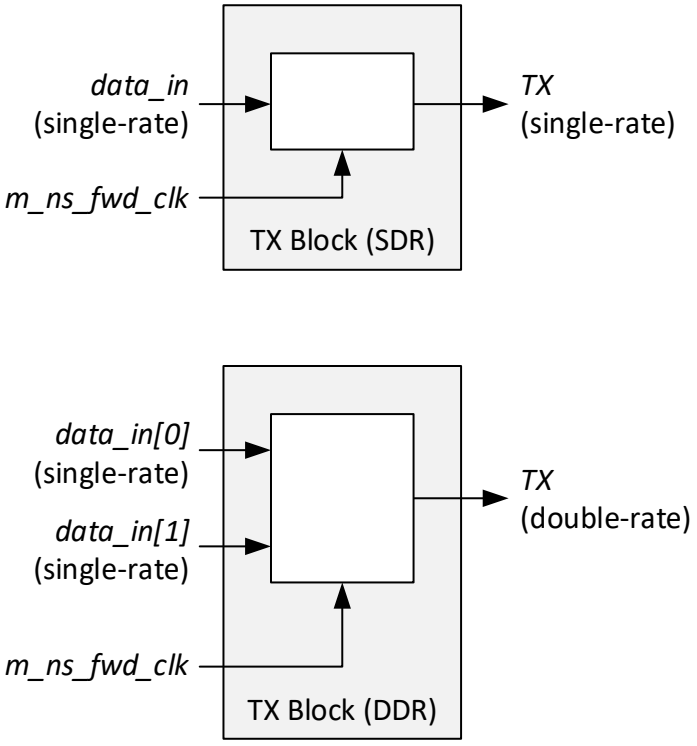


Figure 10. SDR and DDR TX Blocks

Inputs to the I/O block shall be mapped according to Figure 11 and Figure 12. For DDR signaling, two AIB I/O inputs are multiplexed onto one output (for example, inputs 0 and 1 are mapped to output 0). For a DDR-capable I/O (AIB Plus only), all DDR internal signals (double the number of *TX* signals) shall be implemented. If the I/O block is configured as SDR, then half of those internal signals shall not be used.

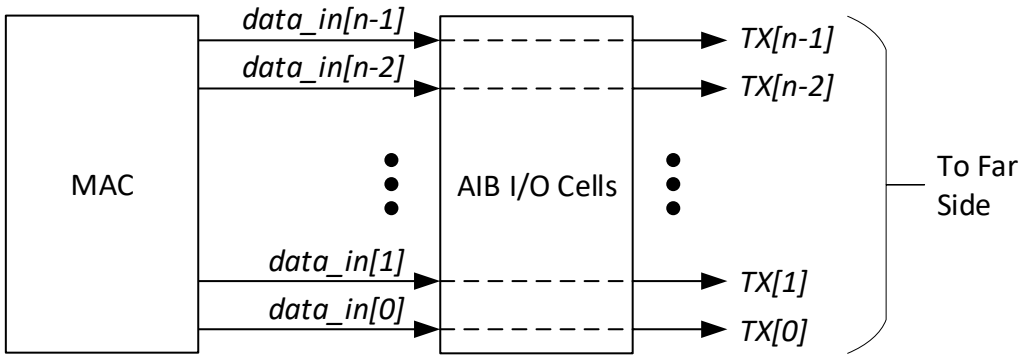


Figure 11. I/O Mapping: Transmit (AIB Base)

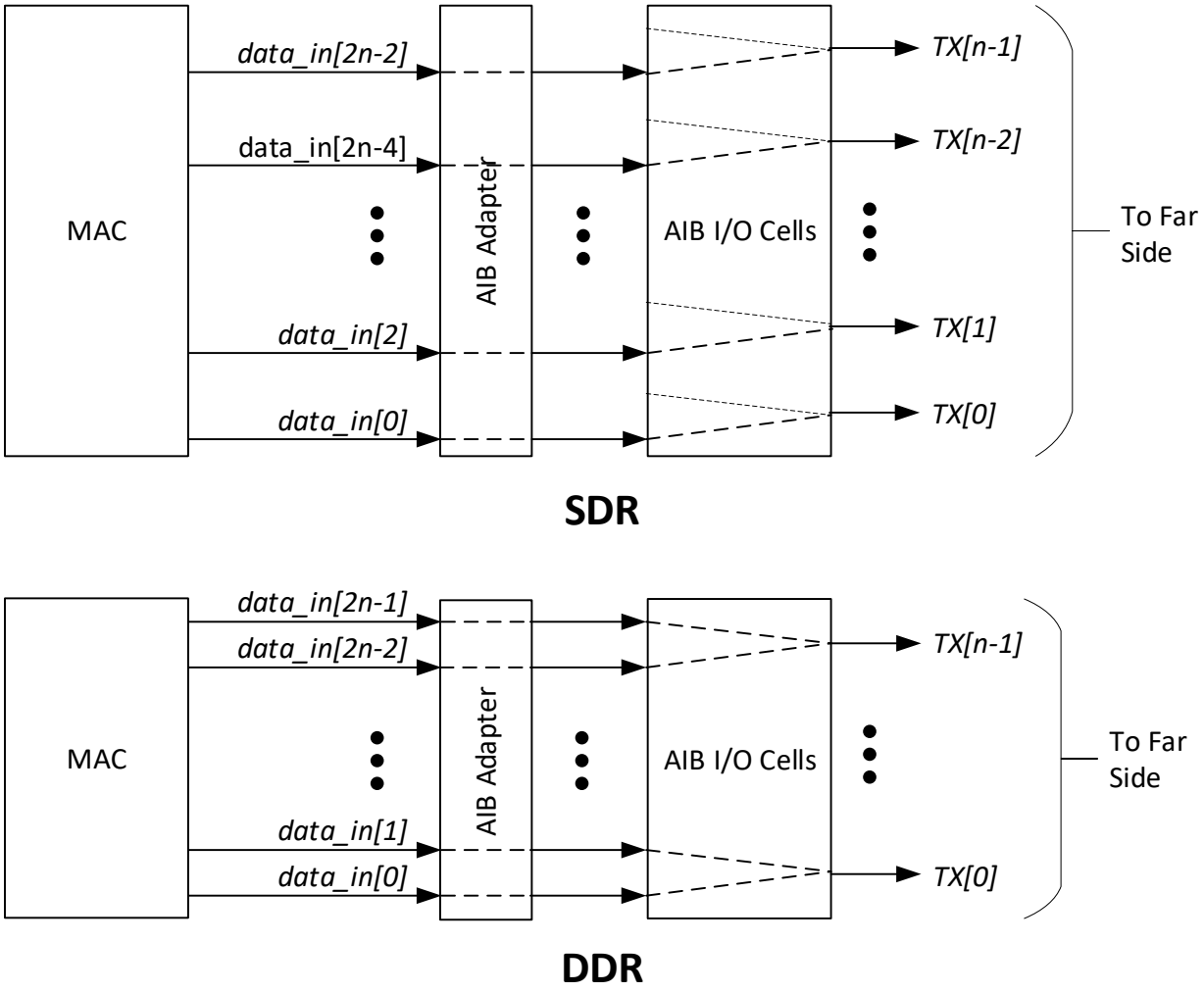


Figure 12. I/O Mapping: Transmit (AIB Plus)

2.1.2 RX Block

An RX block shall register incoming data using the forwarded clock. For a double-data-rate stream (AIB Plus only; Section 2.1.3.2), data shall be clocked into one of two registers, one for each edge of the clock.

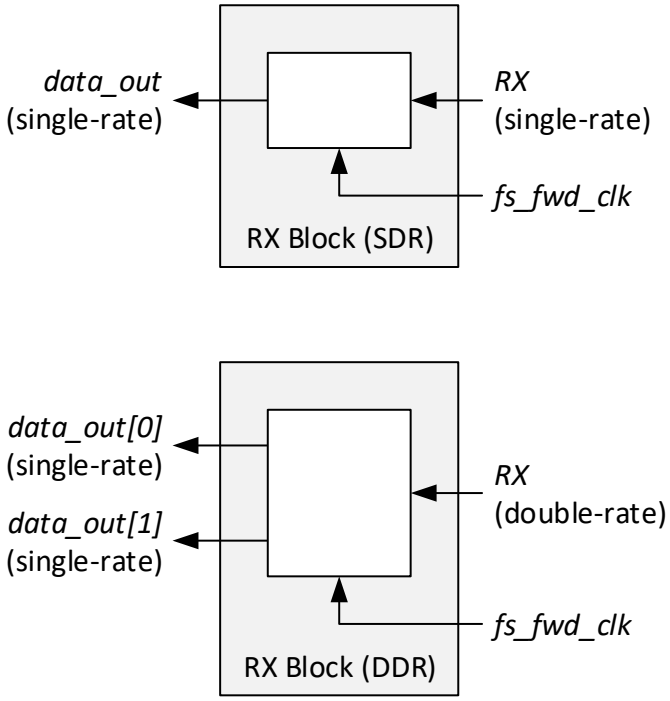


Figure 13. SDR and DDR RX Blocks

Outputs from the I/O block shall be mapped according to Figure 14 and Figure 15. For DDR signaling, one AIB I/O input is split into two AIB Adapter signals (for example, input 0 is mapped to AIB Adapter inputs 0 and 1). For a DDR-capable I/O (AIB Plus only), all DDR internal signals (double the number of *RX* signals) shall be implemented. If the I/O block is configured as SDR, then half of those internal signals shall not be used.

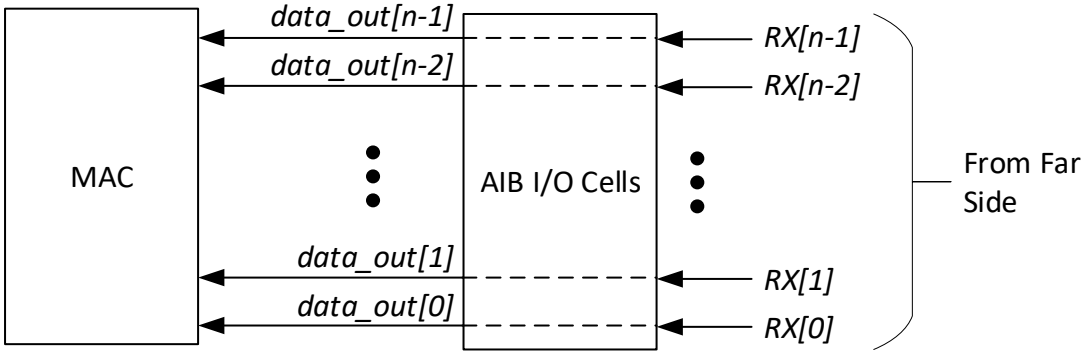


Figure 14. I/O Mapping: Receive (AIB Base)

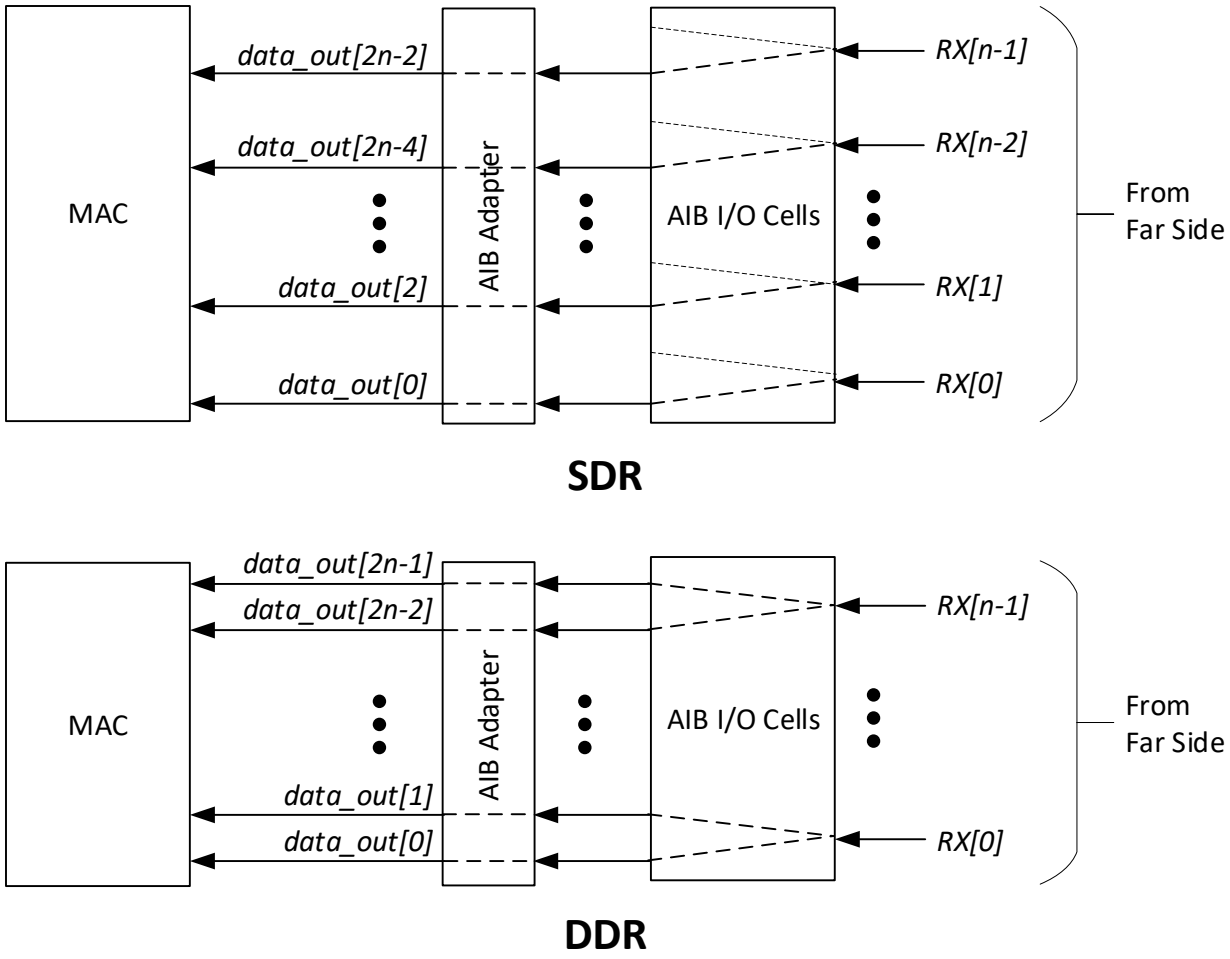


Figure 15. I/O Mapping: Receive (AIB Plus)

2.1.3 Data Exchange

AIB data signals shall be exchanged either as single-data-rate (SDR; AIB Base or Plus) or double-data-rate (DDR; AIB Plus only).

All data signals within an AIB channel shall use the same data exchange format.

2.1.3.1 SDR Data Exchange

AIB Base interfaces shall exchange data using a single-data-rate (SDR) relationship between the clock and data. Data shall be transmitted on the falling edge of the clock.

AIB Plus implementations shall also implement SDR data signals.

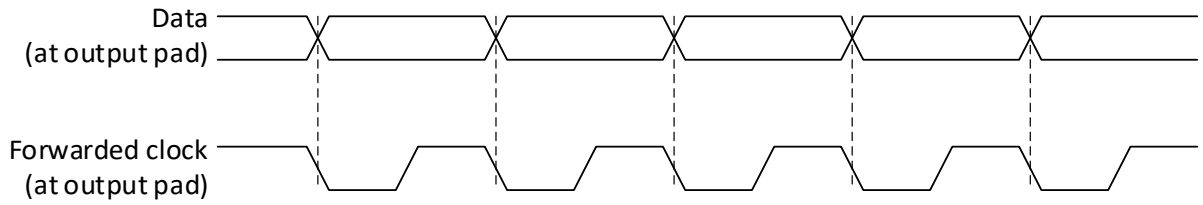


Figure 16. SDR Data/Clock Timing

2.1.3.2 DDR Data Exchange (AIB Plus only)

AIB Plus interfaces shall be capable of exchanging data using a double-data-rate (DDR) relationship between the clock and data. When using DDR exchange, data shall be transmitted on both rising and falling edges of the clock.

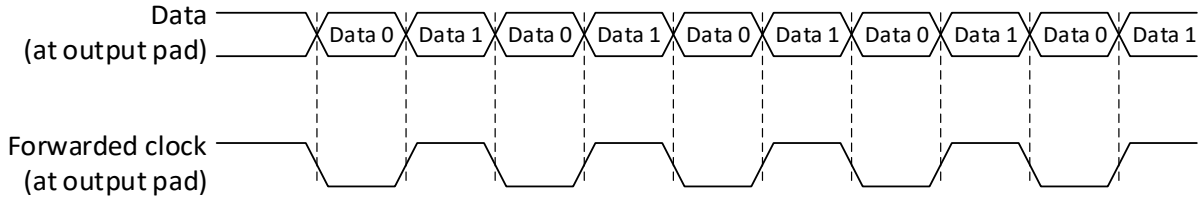


Figure 17. DDR Data/Clock Timing

An AIB I/O shall be capable of both SDR and DDR data exchange regardless of which option is selected for a specific application.

2.1.3.3 Signal Skew

Skew between data edges within a channel and between clock and data edges within a channel shall meet the following specification, where a unit interval (UI) is illustrated in Figure 18.

Symbol	Parameter	Measured at near-side output	Measured at far-side input
t_{DS}	Maximum data-to-data skew	0.04 UI	0.06 UI
t_{DCS}	Maximum data-to-clock skew	0.02 UI	0.03 UI

Table 7. Skew Specifications

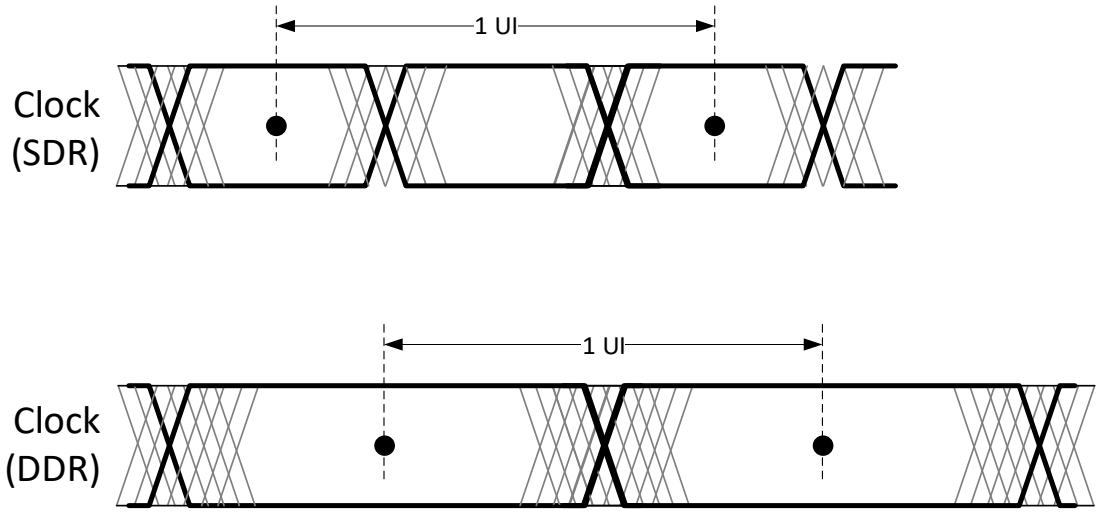


Figure 18. Unit Intervals for SDR, DDR Clocks

Skew relationships are illustrated in Figure 19. The relationships and specifications shall be met by *TX* signals, *ns_fwd_clk*, and *ns_fwd_cklb*, and by *ns_sr_clk*, *ns_sr_clkb*, *ns_sr_data*, and *ns_sr_load*.

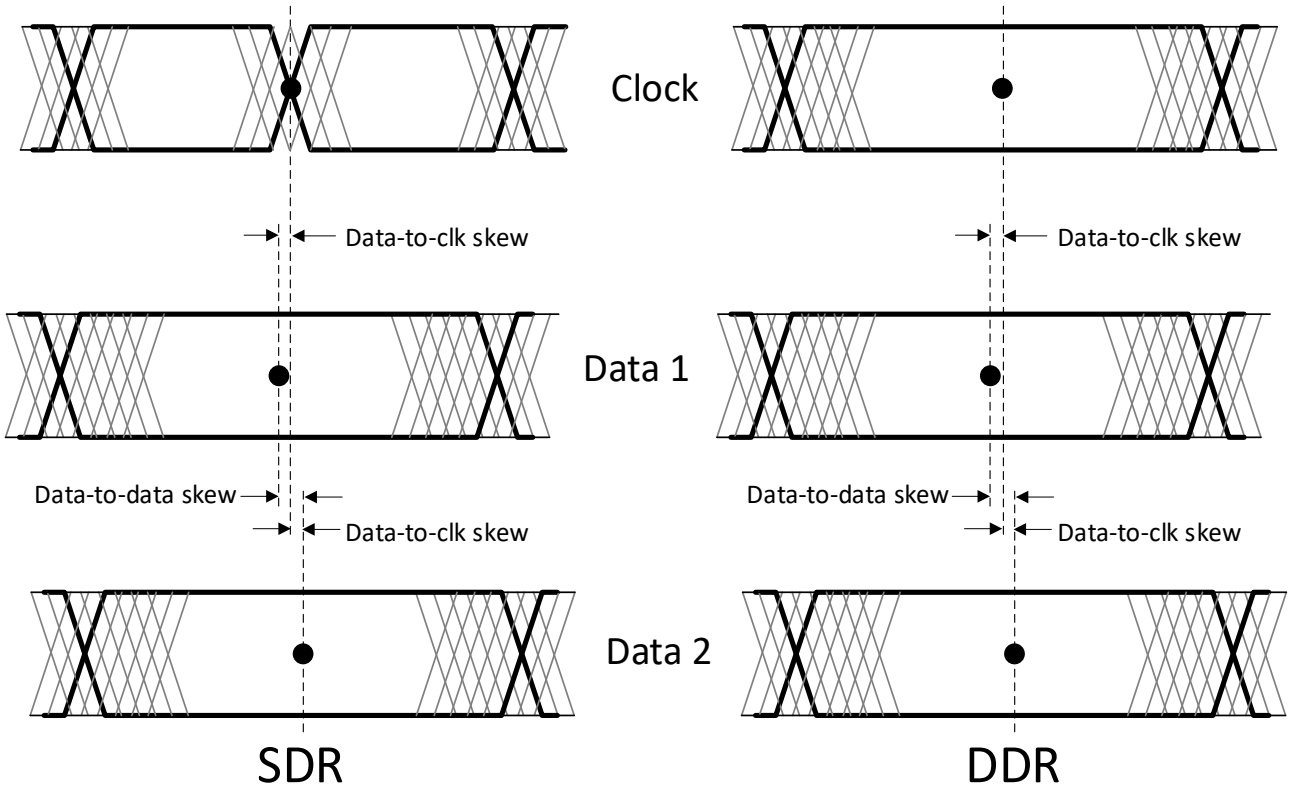


Figure 19. Skew Relationships

2.1.4 Tristate

All output signals shall be capable of being put into tristate.

2.1.5 Weak Pull-Up and Pull-Down

All inputs and outputs shall have an associated weak pull-up and weak pull-down with the equivalent driving strength of 10 – 20 kΩ. The inputs and outputs shall be configurable either with the pull-up, with the pull-down, or with neither the pull-up nor the pull-down.

2.1.6 Data-clock operation

2.1.6.1 Transmit Clock

The AIB layer shall receive a transmit clock, *m_ns_fwd_clk*, from the MAC. That clock shall be used to transmit data and it shall be forwarded to the far side (Section 2.1.6.3). AIB Plus implementations may receive a receive-domain clock from the far side for use as a transmit clock source (Section 2.1.6.4).

2.1.6.2 Receive Clock (AIB Plus only)

In AIB Plus implementations, the AIB layer shall receive a receive clock, *m_ns_rcv_clk*, from the MAC. That clock may optionally be sent to the far side for its use in transmitting data to the near side (Section 2.1.6.4).

2.1.6.3 Forwarded Transmit Clock

An AIB Base or AIB Plus channel shall forward its transmit clock to the far side. The clock signal shall be quasi-differential when moving from one chiplet to the other.

The near side shall receive the forwarded clock from the far side. Once converted from quasi-differential to single-ended, the clock shall be made available to the MAC.

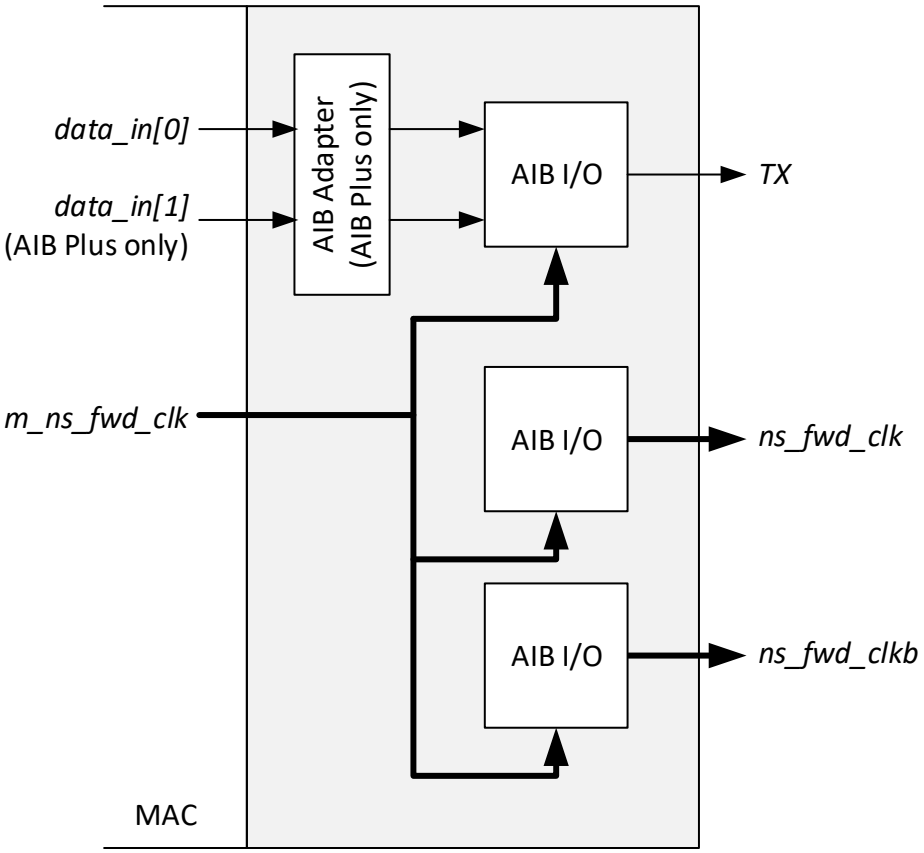


Figure 20. Forwarded Clock – Transmit

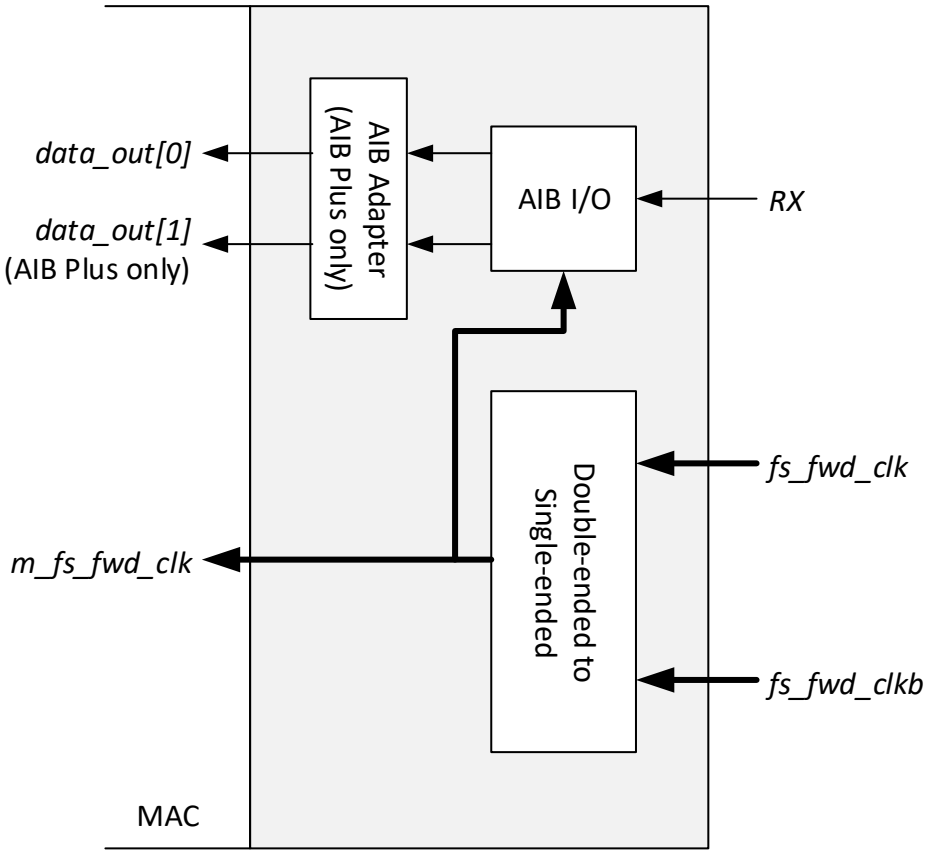


Figure 21. Forwarded Clock – Receive

2.1.6.4 Receive-Domain Transmit Clock (AIB Plus only)

An AIB Plus channel may use a receive-domain clock from the far side as its transmit clock instead of the clock from the near-side MAC if it is desired to transmit data using the far-side receive-clock domain rather than the near-side clock domain. The receive-domain clock shall be quasi-differential when moving from one chiplet to the other.

The receive-domain clock, when received from the far side, shall be sent to the near-side MAC. The MAC may select the receive-domain clock as the transmit clock to be sent to the AIB layer. This MAC selection is suggested in Figure 23 in light gray for clarity only. The specific means of implementing the clock selection within the MAC is not specified in this document.

An AIB Plus interface shall generate either an active receive-domain clock or an inactive receive-domain clock. An active near-side receive-domain clock shall present the near-side receive-domain clock at the near-side receive-domain output clock bump. An inactive near-side receive-domain clock shall have a static value on the near-side receive-domain clock bump. If active, the data sheet should document the receive-domain clock frequency. If inactive, the data sheet should document that fact.

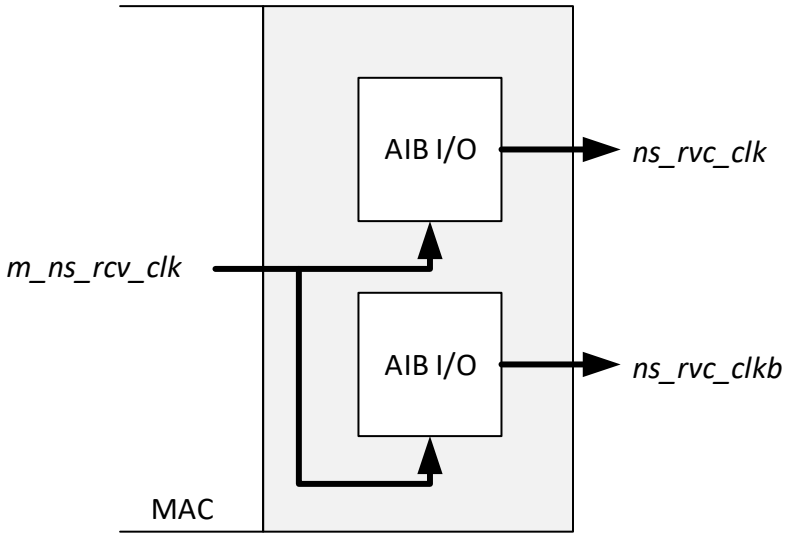


Figure 22. Receive-Domain Clock – Transmit (AIB Plus only)

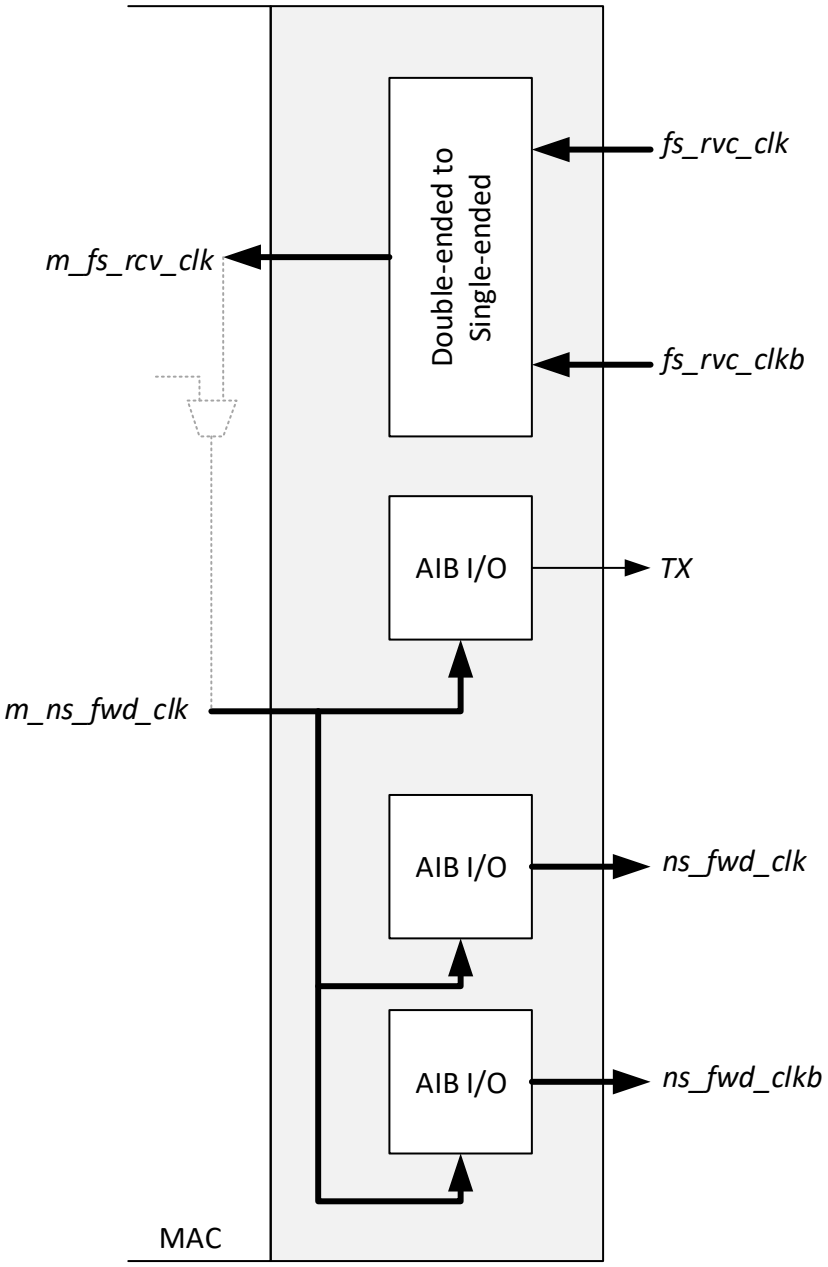


Figure 23. Receive-Domain Clock – Receive (AIB Plus only)

2.1.6.5 Clock Duty Cycle Requirements

The forwarded clock signal shall meet the following duty-cycle specification.

Symbol	Parameter		Near end
t_{FDCD}	Maximum forwarded-clock DCD	SDR	±10%
		DDR	±3%
t_{RDCD}	Maximum receive-domain clock DCD		±10%

Table 8. Clock Duty-Cycle Requirements

2.1.6.6 Optional Forwarded-Clock Duty-Cycle Correction (AIB Plus only)

The transmit clock may pass through a duty-cycle correction (DCC) block prior to clocking the transmit register and prior to being forwarded if necessary for meeting the clock duty-cycle specification under all conditions of voltage and temperature (Section 2.1.6.5).

If the DCC is not present, calibration shall proceed as if it were (Section 3.2.3).

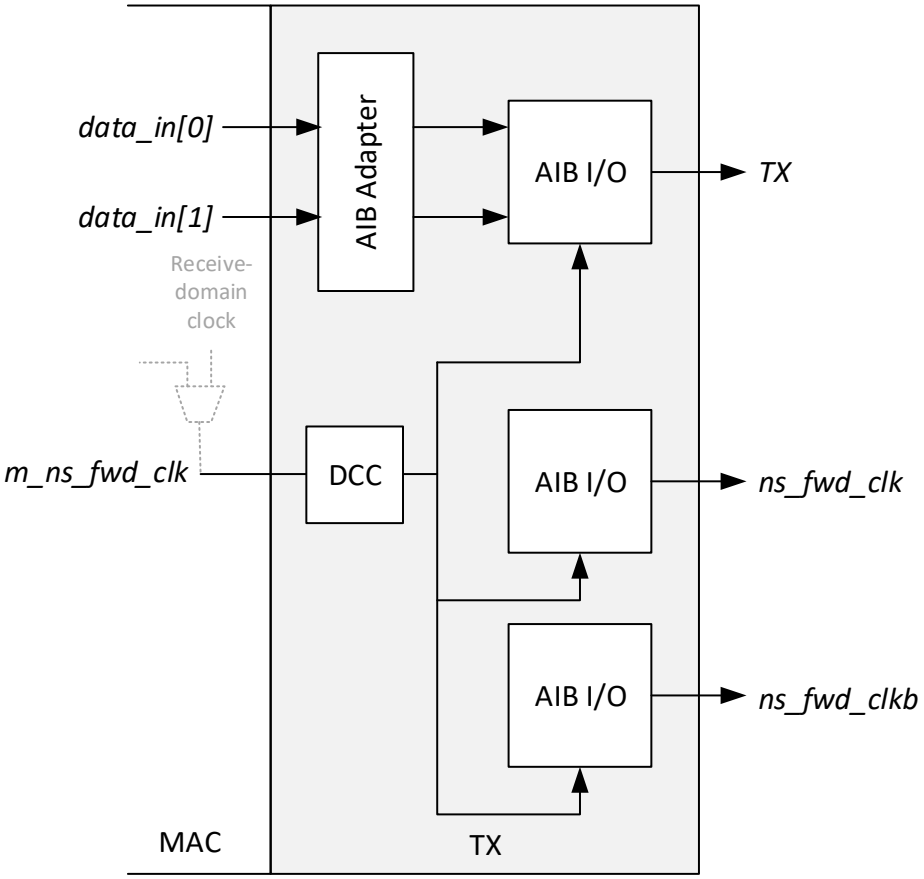


Figure 24. Duty-Cycle Correction (AIB Plus only)

2.1.6.7 Optional Forwarded Clock Phase Selection (AIB Plus only)

The received forwarded clock may pass through a delay-locked loop (DLL) for phase

correction before clocking the capture register if necessary to ensure correct data sampling under all conditions of voltage and temperature.

If the DLL is not present, calibration shall proceed as if it were (Section 3.2.3).

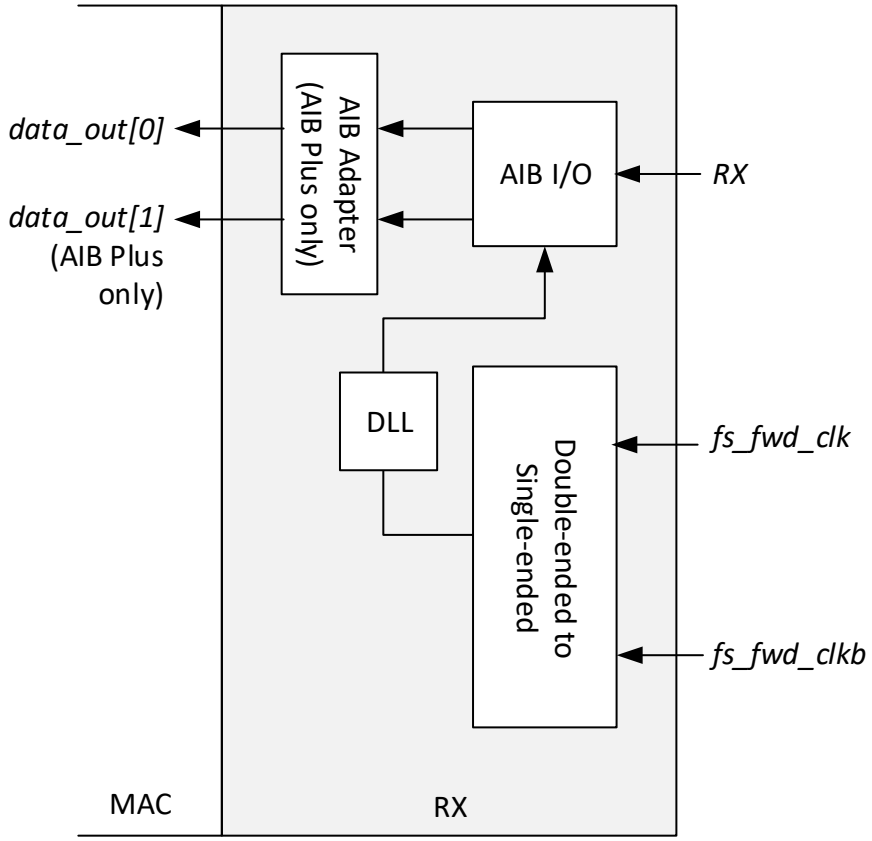


Figure 25. Delay-Locked Loop

2.1.7 Latency

The latency for a signal from the input of a transmitting I/O block to the transmitted output, or from a received input to the output of the receiving I/O block, shall comply with Table 9. Latency Specification and Figure 26. The specified latency includes only the sequential delay and does not include analog delays through the transmitting output buffer, the receiving input buffer, or the interposer traces.

Latency path	SDR	DDR
Transmitting	1 CLK cycle (1 UI)	1.5 CLK cycle (3 UI)
Receiving	1 CLK cycle (1 UI)	1.5 CLK cycle (3 UI)

Table 9. Latency Specification

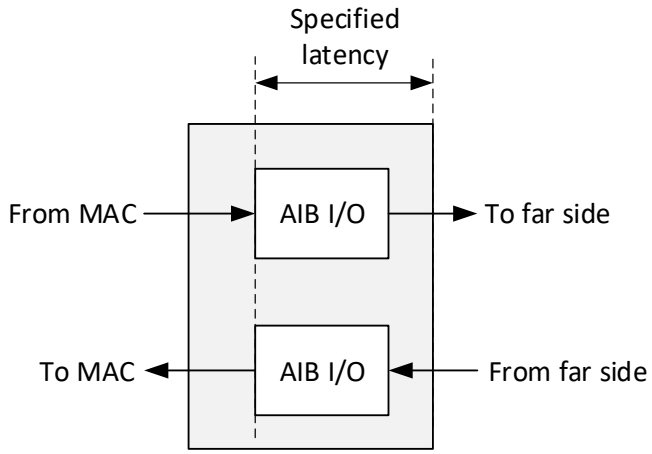


Figure 26. Latency Measurement

2.1.8 Asynchronous mode

I/O blocks that pass *xx_mac_rdy* and *xx_adapter_reset* signals shall operate in an asynchronous mode.

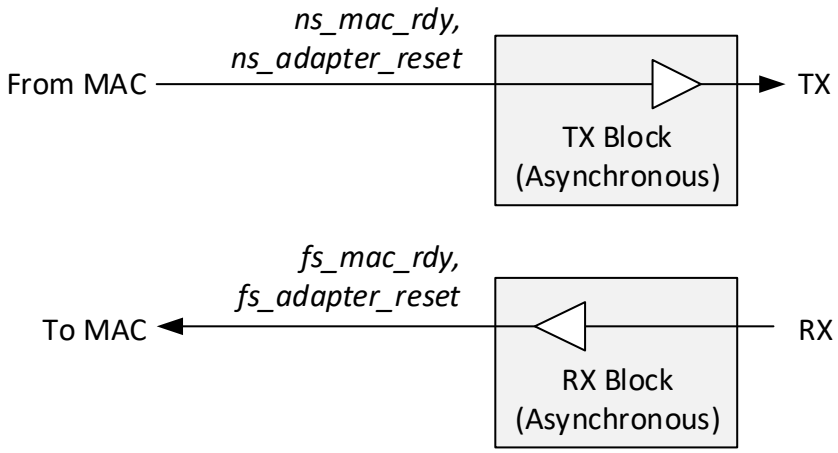


Figure 27. Asynchronous I/O Mode.

2.1.9 Mismatched Interfaces

The near side shall interoperate with a far side having a different number of I/Os than the near side provided:

- The two interfaces are of the same type (AIB Base or AIB Plus)
- The interfaces are aligned on the *spare* bumps.

If a near-side interface is connected to a far-side interface having fewer I/Os than the near-side interface, then the unused I/Os in the near-side interface shall be placed into standby mode (Section 3.1.1).

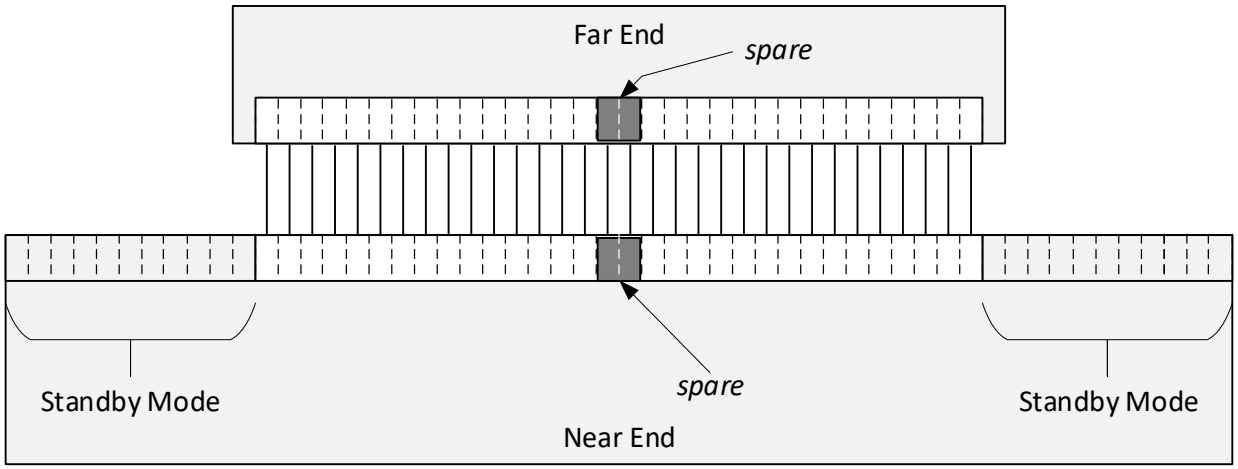


Figure 28. Mismatched Interfaces

2.1.10 Unused Channels

Any unused channels shall have the same number of data signals as the used channels. The data signals in the unused channels shall be put into standby mode.

2.2 AIB Adapter (AIB Plus only)

The AIB Adapter shall provide data retiming, sideband control signals, and calibration control for DDR mode. There shall be one AIB Adapter per channel.

2.2.1 Data-Retiming Register

Within an AIB Plus interface, the AIB Adapter shall implement at least one retiming register in the data path on the transmit and receive sides. All data signals within a channel shall have the same retiming configuration. The data sheet should indicate the latency through the AIB Adapter.

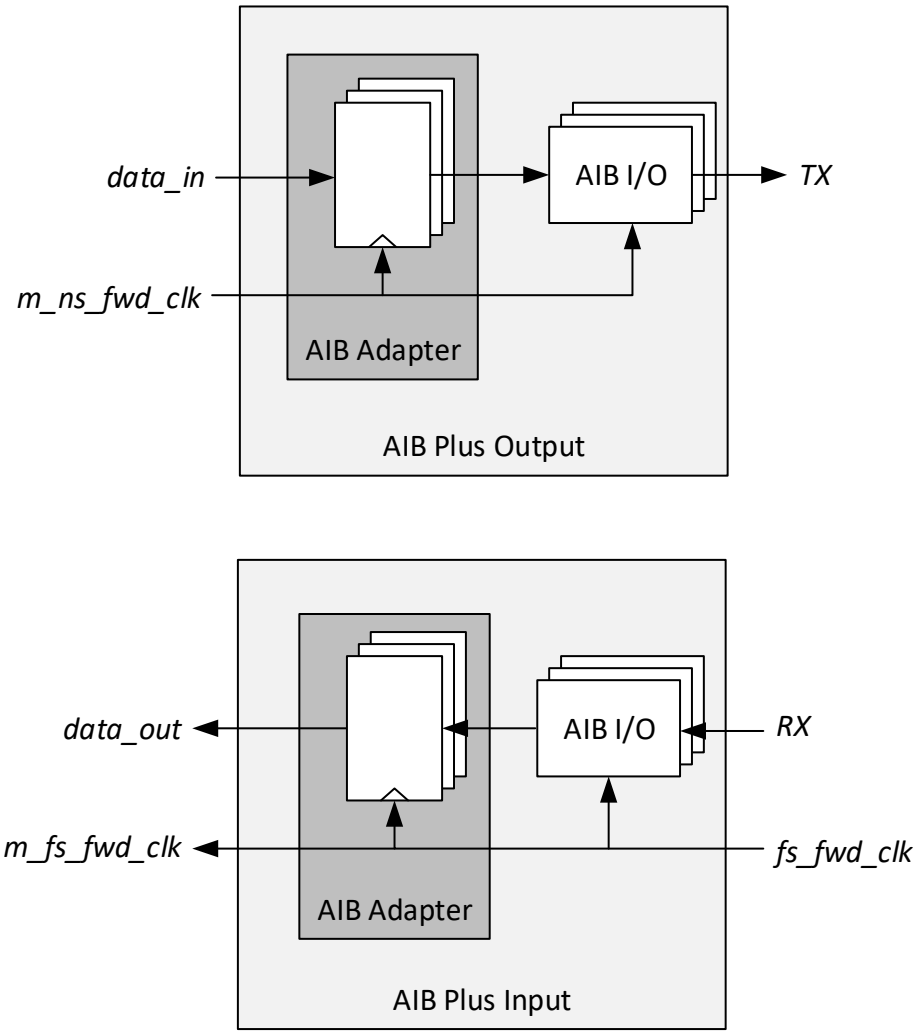


Figure 29. Retiming Register

2.2.2 Sideband Control Signals

An AIB interface shall provide control signals for calibration and communication of control and status information between the near-side and far-side chiplets. These signals shall be time-domain multiplexed onto a single data line. Internal signals shall be first loaded into a parallel register before being shifted out serially, starting with the most-significant bit (MSB). Received serial signals shall be loaded into a parallel register upon assertion of the received *load* signal. A *load* signal (Section 2.2.2.3) shall coordinate the timing of the loading from and to parallel registers. The *load* signal shall be generated on the sending side and shall be forwarded from the sending chiplet to the receiving chiplet.

The sideband control shift register shall be clocked by a free-running clock (Section 2.2.2.2) such that the shift registers shall constantly send data. It shall not be possible to stop or restart the sending of sideband control signals.

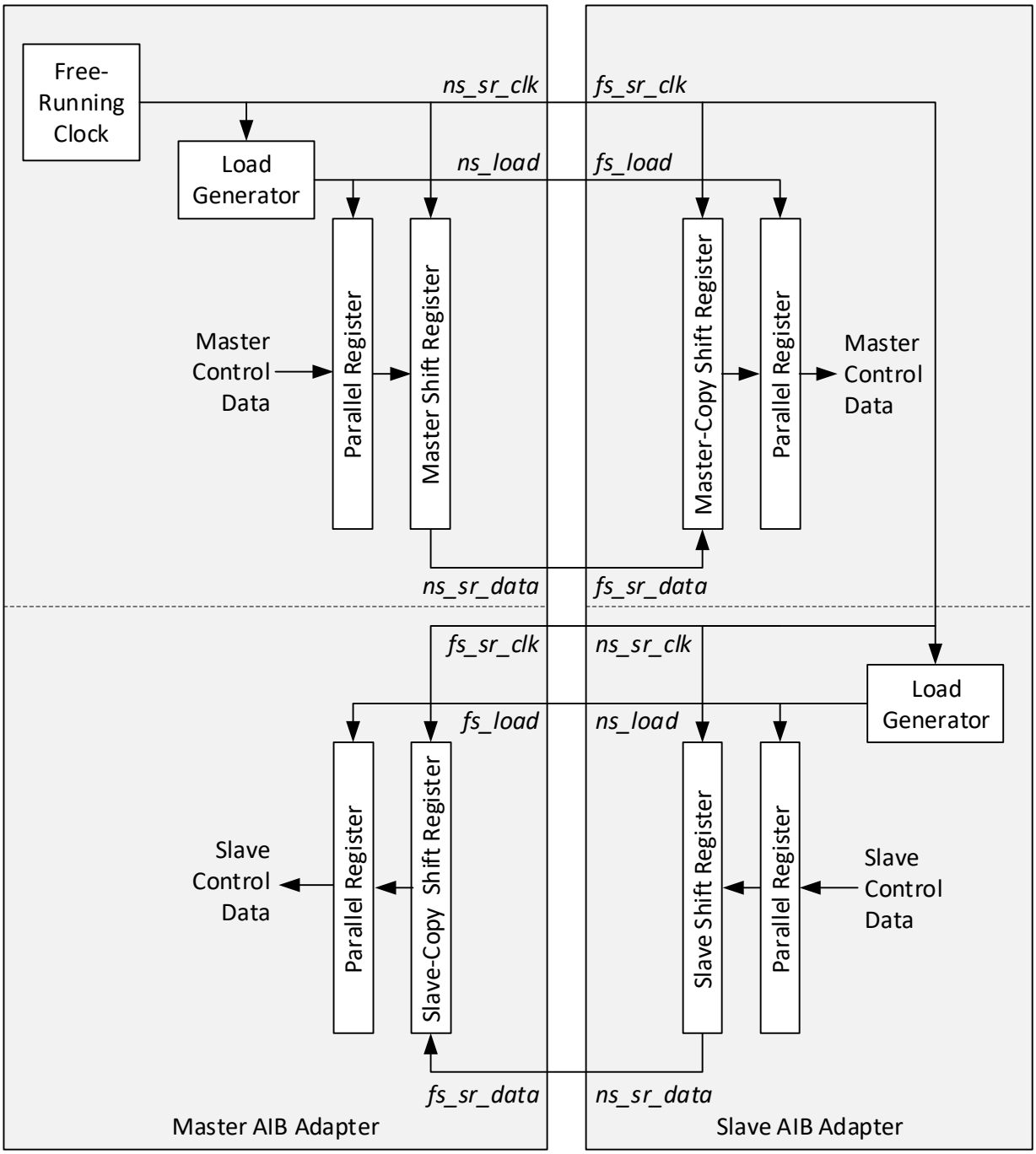


Figure 30. Sideband Control Shift Registers

2.2.2.1 Shift-Register Configurations

The definition of the shift register and the position of signals shall depend upon whether the interface is a master, slave, or dual-mode.

2.2.2.1.1 Master Interface

A master interface shall include two shift registers: one for transmitting master sideband control signals to the slave (the master shift register), and one for receiving sideband control signals from the slave (the slave-copy shift register). The master shift register shall contain 81 bits. The slave-copy shift register shall contain 73 bits. All bits shall be implemented regardless of whether optional signals are implemented. Any signals not implemented shall permanently maintain their default values as defined in Table 50.

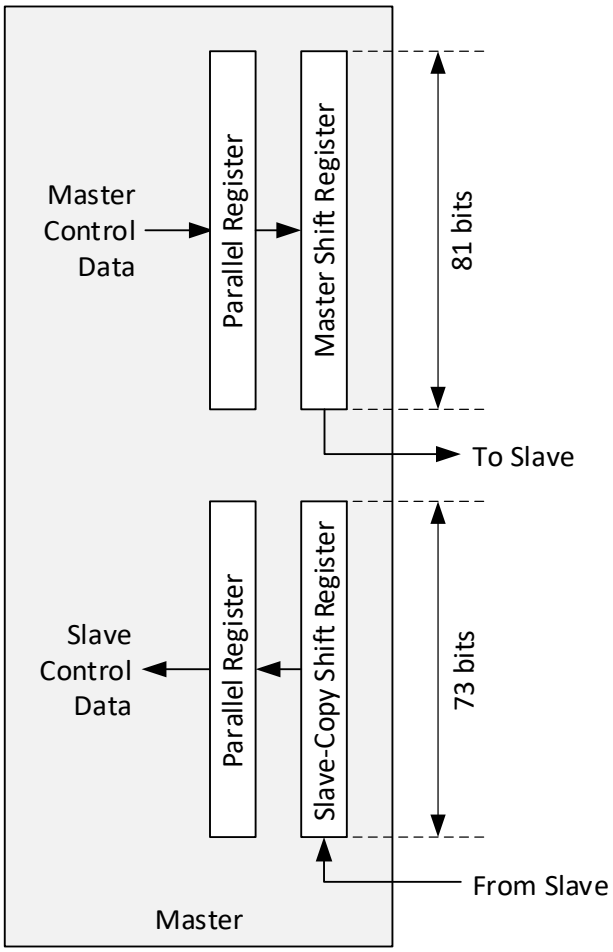


Figure 31. Master Sideband Control Shift Register

2.2.2.1.2 Slave Interface

A slave interface shall include two shift registers: one for transmitting slave sideband control signals to the master (the slave register), and one for receiving sideband control signals from the master (the master-copy shift register). The slave shift register shall contain 73 bits. The master-copy shift register shall contain 81 bits. All bits shall be implemented regardless of whether optional signals are implemented. Any signals not implemented shall permanently maintain their default values as defined in Table 51

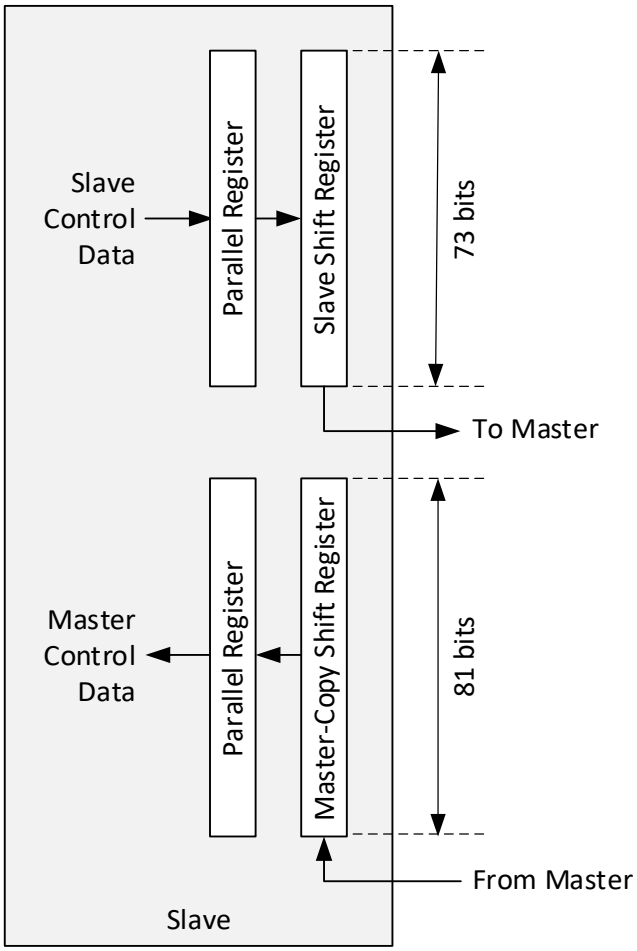


Figure 32. Slave Sideband Control Shift Register

2.2.2.1.3 Dual-Mode Interface

When configured as a master, a dual-mode interface shall implement an output shift register configured as a master and an input shift register configured as a slave. When configured as a slave, a dual-mode interface shall implement an output shift register configured as a slave and an input shift register configured as a master.

The *dual_mode_select* signal (Section 1.3.3.2) shall select the shift-register configuration appropriate to the selected mode.

2.2.2.2 Free-Running Clock

The shift registers shall be clocked by a free-running clock. The free-running clock shall be generated on the master side and forwarded to the slave side.

The free-running clock shall be independent of the data clocks. It shall run continuously during operation. It shall have a frequency within the range specified in Table 10.

Parameter	Min	Max
Free-Running Clock Frequency (default)	600 MHz	1 GHz
Free-Running Clock Frequency if user-defined signals (Section 2.2.2.5.2) are not used	50 Mhz	1 Ghz

Table 10. Free-Running Clock Frequency

2.2.2.2.1 Free-Running Clock Generation

The master interface chiplet shall be responsible for generating the free-running clock and distributing it to the slave interface via the *sr_clk* signal (*ns_sr_clk* when sent from the near side; *fs_sr_clk* when received from the far side). The slave interface (especially dual-mode) chiplet can either use forwarded free running clock from master interface chiplet or use its existing free running clock to support output operation of its sideband control signals. If an external oscillator is used to generate the free-running clock, then output of that oscillator shall be run through the master interface before it is distributed to the slave interface via *sr_clk* in order to ensure the correct phase relationships between the free-running clock and the *load* signal (Section 2.2.2.3).

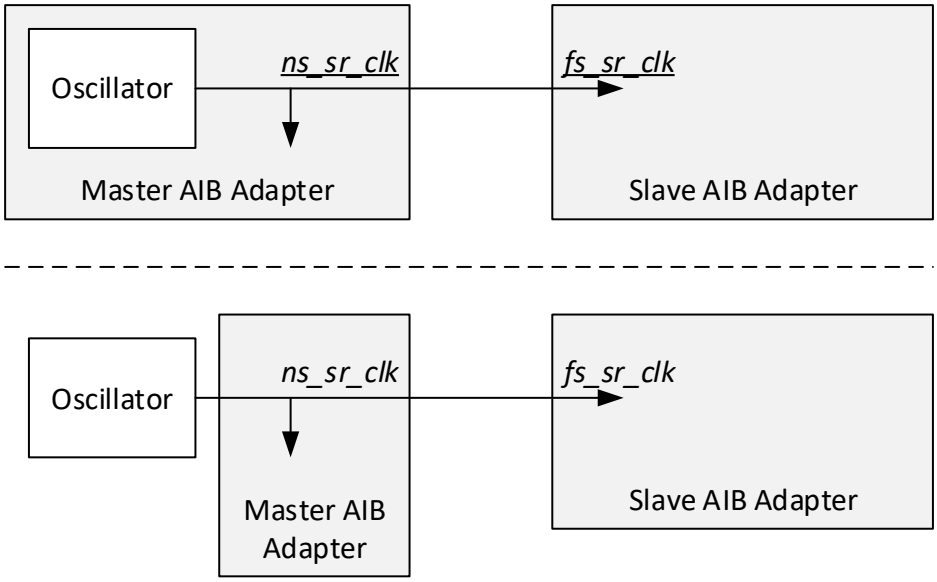


Figure 33. Free-running Clock Generation Options

2.2.2.3 Load Signal

An *sr_load* signal (*ns_sr_load* when sent from the near side; *fs_sr_load* when received from the far side) shall provide synchronous data transfer between parallel and serial shift registers (Figure 32). It shall be created from the free-running clock by a sideband-control state machine and transmitted as an SDR signal (Figure 16). The period of the *load* signal shall be a number of free-running clock cycles that is one more than the number of bits in the shift register (i.e., 1/74 or 1/82).

The *sr_load* signal shall remain LO until asserted HI; it shall remain asserted HI for one clock cycle before being de-asserted LO.

When transmitting sideband control signals, the control signal data shall be clocked into the parallel register when the *load* signal is asserted. When receiving sideband control signals, the received serial data shall be transferred into the parallel register when the *load* signal is asserted.

There shall be no valid control-bit data when the *load* signal is asserted. The MSB shall be shifted out on the falling edge of the *load* signal.

2.2.2.4 Control Signal Timing

When loading a new sideband control signal value into the parallel register for shifting out, the value presented to the parallel load register shall remain valid for at least the maximum time between *load* assertions.

The first bit of the shift register to be shifted out when transmitting shall be the MSB (bit 72 or 80); the last bit to be shifted in when receiving shall be the LSB (bit 0).

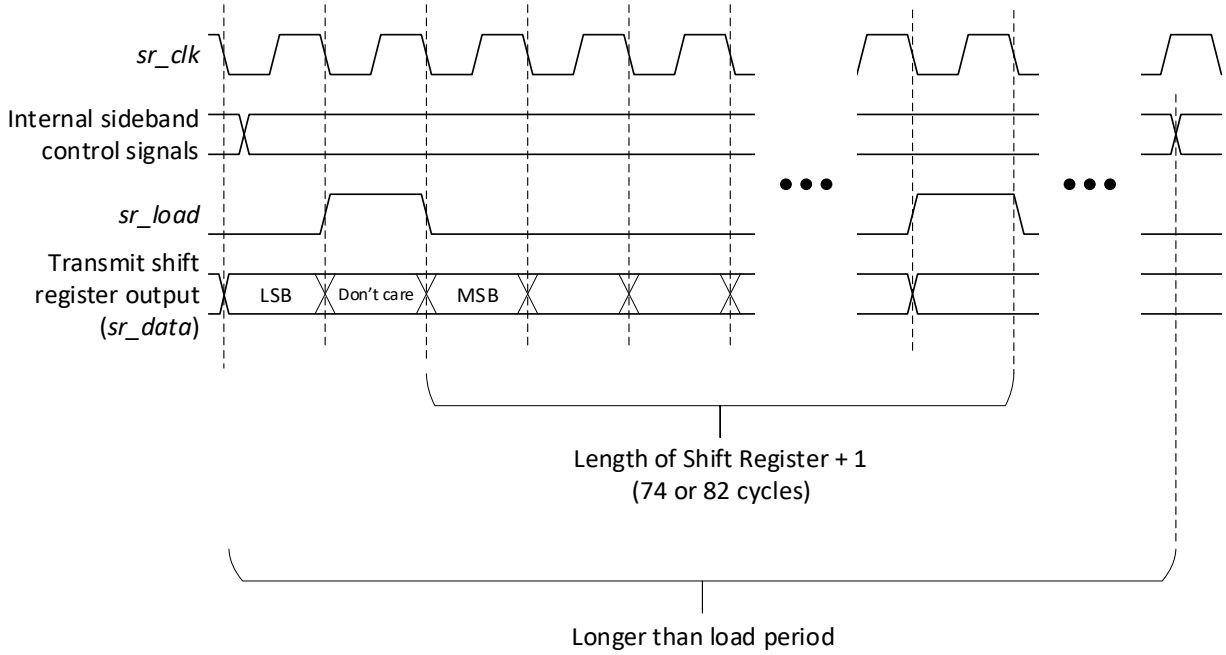


Figure 34. Sideband-Control Shift-Register Timing

2.2.2.5 Control Signals

Sideband control signals fall into one of the following categories:

- Calibration handshake
- Selective reset
- User-defined
- Reserved

The bits for any unused signals shall be maintained with default values for correct shift-

register length.

The sideband control signals are summarized in Table 11 and are detailed in Table 51.

2.2.2.5.1 Calibration Bits

Calibration bits shall be generated by internal state machines as described in Section 3.2.3.

2.2.2.5.2 User-Defined Bits

User-defined bits are available for application use. Since both sides need to understand the function of user-defined bits, using these bits may limit chiplet interoperability. If implemented in an application, user-defined bits should be described in the chiplet data sheet.

2.2.2.5.3 Shift-Register Signals

Table 11 defines the sideband control signals for master and slave chiplets. The table is organized by signal type; in-order signal tables with default values are provided in Section 7.2. *ms* prefixes refer to signals originating on the master side; *sl* prefixes refer to signals originating on the slave side.

Signal name	Signal function	Bits	Signal origin (far-side or MAC)	Bit number	
				Master	Slave
Calibration (Section 3.2.3)					
ms_osc_transfer_en *sl_osc_transfer_en*	Oscillator calibration complete	1	FS	80	72
ms_tx_dcc_cal_done *sl_tx_dcc_cal_done*	TX DCC calibration complete	1	FS	68	31
ms_rx_transfer_en *sl_rx_transfer_en*	RX calibration complete	1	FS	75	70
ms_rx_dll_dcc_lock_req *sl_rx_dll_dcc_lock_req*	Start RX calibration	1	MAC	NA	69
ms_rx_dll_lock *sl_rx_dll_lock*	RX DLL locked	1	FS	74	68
ms_tx_transfer_en *sl_tx_transfer_en*	TX calibration complete	1	FS	78	64
ms_tx_dll_dcc_lock_req *sl_tx_dll_dcc_lock_req*	Start TX calibration	1	MAC	NA	63
User-defined					
external_cntl[]	Defined by protocol and/or application	sl: 30 ms: 63	MAC or FS	0-4 8-65	32-57 28-30 0-26
Other					
Reserved			NA	79 76-77 69-73 66-67 5-7	71 65-67 58-62 27

Table 11. Sideband Control Signals

3 Reset and Initialization

3.1 Data-Transfer Ready

A data-transfer ready signal shall be made available for control by the MAC layer. The data-transfer ready signal may be de-asserted due to application-driven changes, including but not limited to:

- An intentional change in clock frequency
- Receipt of bad data

De-asserting the data-transfer ready signal may also be necessary due to conditions within the AIB interface, which may include but are not limited to:

- Completion of configuration during power-up
- Initiation of reset by the far side
- Loss of DLL lock

Internal AIB conditions indicating the need for de-assertion of the data-transfer ready signal shall be sent to the MAC so that the MAC can de-assert the data-transfer ready signal.

For AIB Plus interfaces, once data-transfer ready has been re-asserted after having been de-asserted, the AIB Adapter shall be re-calibrated (Section 3.2.3). The reverse is not true: calibration may be initiated without de-asserting data-transfer ready first.

3.1.1 Standby Mode

A signal shall be placed into standby mode by one of the following means:

- Driving the signal LO
- Putting the signal into tristate and enabling the weak pull-down.

During initialization, data outputs shall be placed into standby mode.

3.1.2 Data-Transfer Ready Signals

Each channel shall have an *ns_mac_rdy* signal that is controlled by the near-side MAC. When the *ns_mac*_rdy signal is asserted HI by the MAC, it shall indicate that the near side is ready for calibration and data transfer. De-assertion of *ns_mac_rdy* shall affect only its own channel; other channels may continue transmitting data.

The *ns_mac_rdy* signal shall be forwarded to the far side, where it shall be received on the *fs_mac_rdy* input in order to inform the far-side MAC that the near-side MAC is or is not ready for calibration.

3.1.3 The Effects of De-asserting Data-Transfer Ready

While the *ns_mac_rdy* signal is de-asserted:

- Data transmission shall halt

- Data outputs shall be placed into standby mode (Section 3.1.1)
- The clock output *ns_fwd_*clk shall go into standby mode.
- The reset signal *ns_mac_*rdy shall be sent to the far side of the interface in order to communicate that data transmission has halted and to allow for the far side to be reset.

The contents of retiming registers (Section 2.2.1) shall be undefined following de-assertion of data-transfer ready.

De-assertion of the data-transfer ready signal shall not affect the free-running clock signals or the sideband-control signals.

3.2 Initialization

Initialization will consist of two or three steps in sequence:

- Power-on reset synchronization
- Configuration
- Calibration (AIB Plus only)

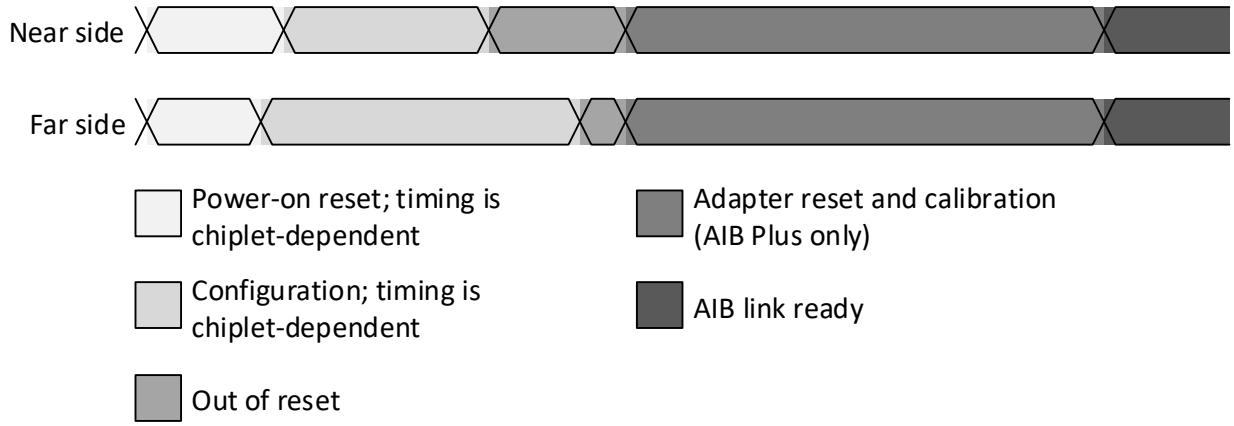


Figure 35. AIB Initialization

If there are multiple AIB Plus interfaces on a single chiplet, they shall all come out of configuration at the same time, but they may complete adapter reset and calibration at different times depending on implementation.

3.2.1 Power-on Reset Synchronization

Power-on reset, being the first step in initialization, shall not require any features enabled by configuration (in the manner that the SDR/DDR option is configured, for example), since configuration will not occur until after power-on reset.

3.2.1.1 Power-on Reset Signals

Two signals shall participate in power-on reset: *power_on_reset* and *device_detect*. Their function shall depend on whether the interface is a master, a slave, or dual-mode (Section 1.3.3).

- For master interfaces:
  - *power_on_reset* shall be implemented as an input.
  - *device_detect* shall be implemented as an output.
- For slave interfaces:
  - *power_on_reset* shall be implemented as an output.
  - *device_detect* shall be implemented as an input.
- For dual-mode interfaces, the *dual_mode_select* (Section 1.3.3.2) signal shall select the function of the *power_on_reset* and *device_detect* signals as input/output (master) or output/input (slave).

3.2.1.2 Power-on Reset Sequence

During power-on reset, all input and output signals shall be placed into standby mode (Section 3.1.1). The power-on reset sequence shall proceed as follows:

1. The master interface shall assert its *device_detect* signal HI to indicate its presence to slave interfaces on different chiplets. If no *device_detect* signal is detected by the slave, then the slave may act both to ensure that it and its chiplet are in a safe state and to alert the MAC.
2. Each chiplet shall implement its own power-on reset routine. At the beginning of the routine, slave interfaces shall assert their *power_on_reset* signals HI.
3. When a chiplet completes its power-on reset sequence:
   a. Master interfaces shall begin the configuration stage.
   b. Slave interfaces shall de-assert their *power_on_reset* signals LO and begin the configuration stage.

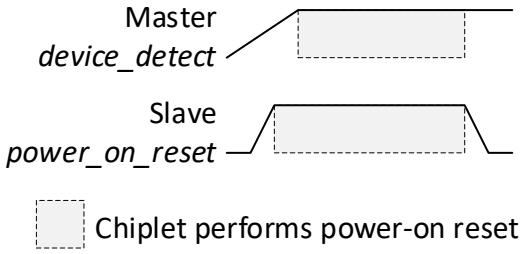


Figure 36. Power-on reset synchronization

3.2.1.3 Unused Interfaces

In order to ensure correct operation for chiplets with unused master interfaces, the *power_on_reset* inputs for those unused interfaces shall have weak pull-ups.

In order to ensure correct operation for chiplets with unused slave interfaces, the *device_detect* inputs for those unused interfaces shall have weak pull-downs.

3.2.1.4 Test Provision

In order to test the power-on reset sequence at the wafer level, two signals shall be provided for use by automated test equipment to override the *power_on_reset* and *device_detect* signals when there is no master/slave pair available. *por_ovrd* overrides

the *power_on_reset signal*, and *device_detect_ovrd* overrides the *device_detect* signal.

3.2.2 Configuration

Configuration may include:

- Host chiplet configuration (in the case of an FPGA or similar chiplet)
- AIB interface configuration
- AIB redundancy activation

The *ns_mac_rdy* signal shall be de-asserted LO during configuration and shall be asserted HI when configuration completes and the chiplet is ready for calibration and data transfer. The clock input from the MAC shall be stable prior to assertion of *ns_mac_rdy*.

3.2.2.1 Output State During Configuration

All outputs, including data outputs, the free-running clock output, sideband-control output, reset outputs, and adapter reset outputs shall be in standby mode (Section 3.1.1) during configuration. The free-running clock output, sideband-control output, reset outputs, and adapter reset outputs must come out of standby mode upon completion of configuration.

3.2.2.2 Chiplet Configuration

Configuration of any non-AIB aspects of the chiplet is outside the scope of this specification.

3.2.2.3 AIB Interface Configuration

3.2.2.3.1 Power-Up Configuration

All intended AIB features shall be configured at power-up.

3.2.2.3.2 JTAG Configuration

The chiplet data sheet should document the configuration requirements that allow for successfully implementation of JTAG EXTEST and INTEST operations.

3.2.2.3.3 Free-Running Clock Oscillator Stability (AIB Plus only)

Within a master interface, the oscillator used for the free-running clock shall be stable before configuration is complete.

3.2.2.3.4 Sideband Control Shift Register Readiness (AIB Plus only)\

The sideband control shift register shall be operational once configuration is complete.

3.2.2.3.5 Configuration Completion Signals

Each chiplet shall have a *conf_done* signal. *conf_done* shall be an open-drain output. It shall be asserted LO when configuring, and it shall be released when configuration of all interfaces on the chiplet is complete, the analog circuits are stable, and the free-running

clock is stable. *conf_done* shall indicate only that AIB configuration is complete. No other configuration completion (MAC, FPGA, etc.) shall be included in the generation of the *conf_done* signal.

All *conf_done* signals from all chiplets of a module should be connected in a wired-AND configuration to generate a module-level *CONF_DONE* signal that shall be HI when all chiplets on the module have completed AIB configuration. The pull-up resistor used to implement the wired-AND function may reside on the module containing the chiplets with AIB interfaces, or it may reside off the module. The *CONF_DONE* signal should be provided as an output of the module regardless of the resistor placement.

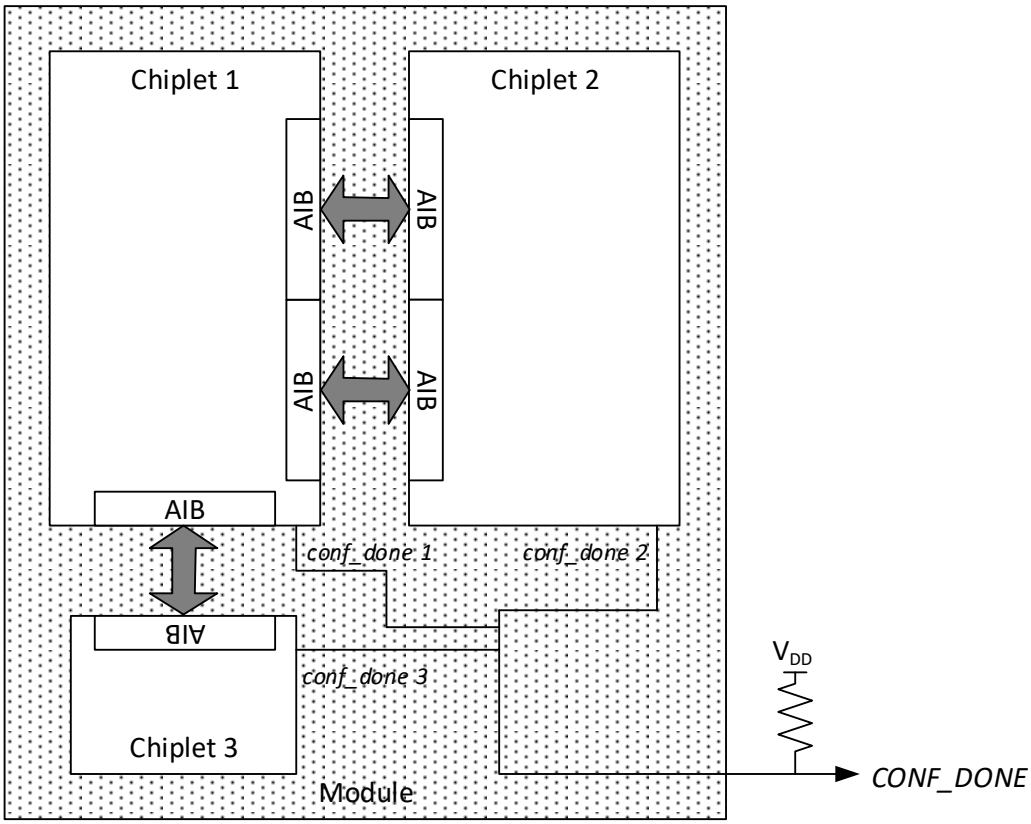


Figure 37. Configuration completion signals.

Data outputs shall remain in standby mode (Section 3.1.1) until *CONF_DONE* is asserted, including in the case where CONF_DONE is pulled low some time after being asserted high.

The resistance and VDD values should comply with

Parameter	Value
Pull-up resistance	1 kΩ
Pull-up V_{DD}	0.9 V

Table 12. Wired-AND Pull-Up Guidelines

3.2.3 Calibration (AIB Plus only)

The calibration sequence shall proceed as follows:

- Adapter reset
- Free-running-clock synchronization
- Data path calibration

The adapter reset phase and free-running-clock synchronization phase may run concurrently.

3.2.3.1.1 Adapter Reset Signal

An AIB interface shall have an *ns_adapter_rstn* signal that is asserted by the MAC. It shall be forwarded to the far side of the interface, driving the far side's *fs_adapter_rstn* input. Likewise, the interface shall have an *fs_adapter_rstn* input that accepts the *ns_adapter_rstn* signal from the far side.

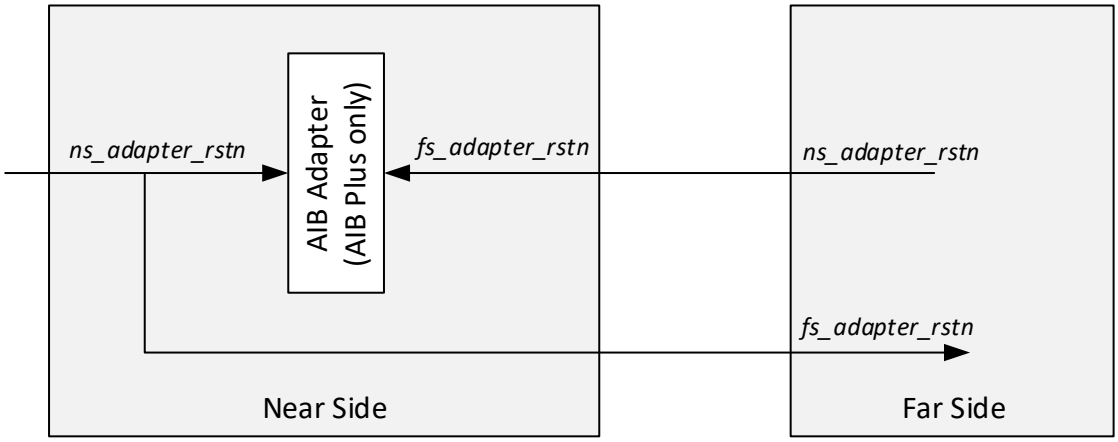


Figure 38. Adapter Reset

When either the near-side or far-side adapter reset signal is asserted LO, the adapter shall reset the calibration state machines. If adapter reset follows de-assertion of data-transfer ready, *ns_mac_rdy* must be asserted HI before *ns_adapter_rstn* is asserted HI.

3.2.3.2 Free-Running-Clock Synchronization

The free-running clocks are synchronized across the interface according to Figure 39. The numbers in black indicate the sequence of steps.

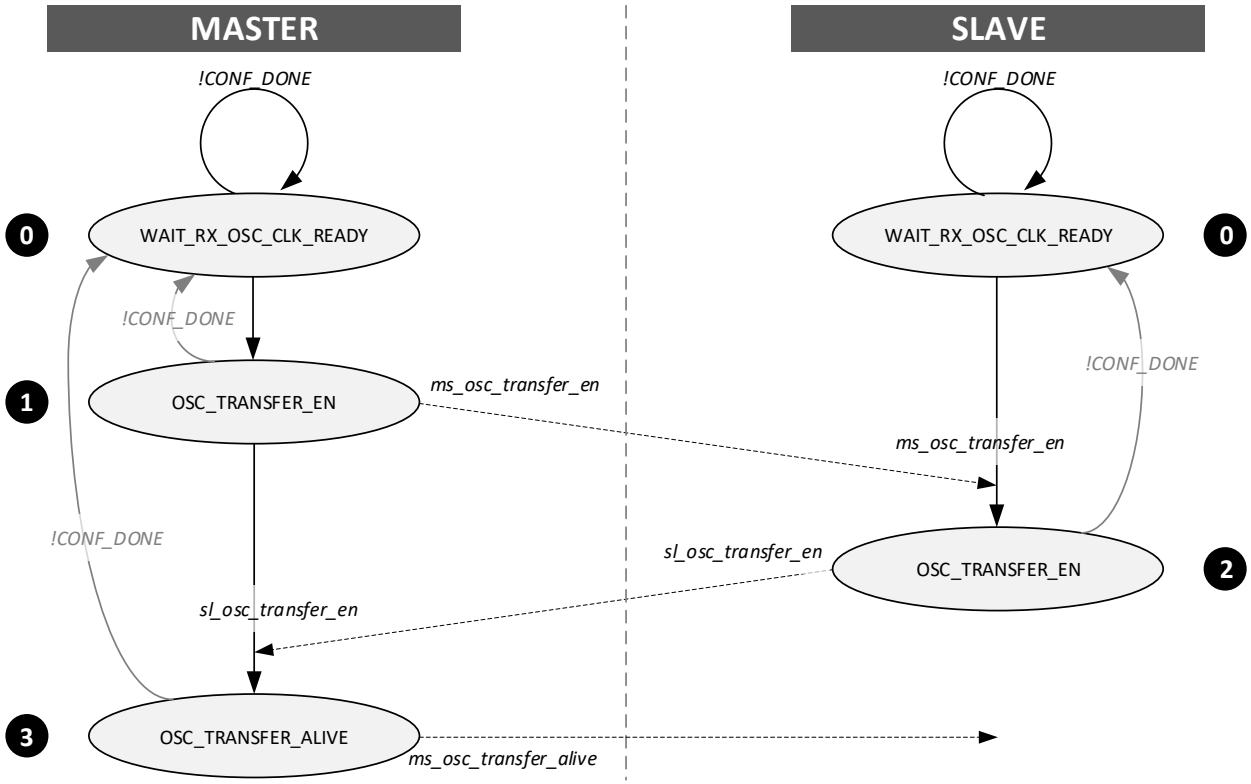


Figure 39. Free-running Clock Calibration State Machine

3.2.3.3 Data-Path Calibration

Data-path calibration shall be implemented via state machines on the master and slave sides of the interface that intercommunicate via the sideband control signals.

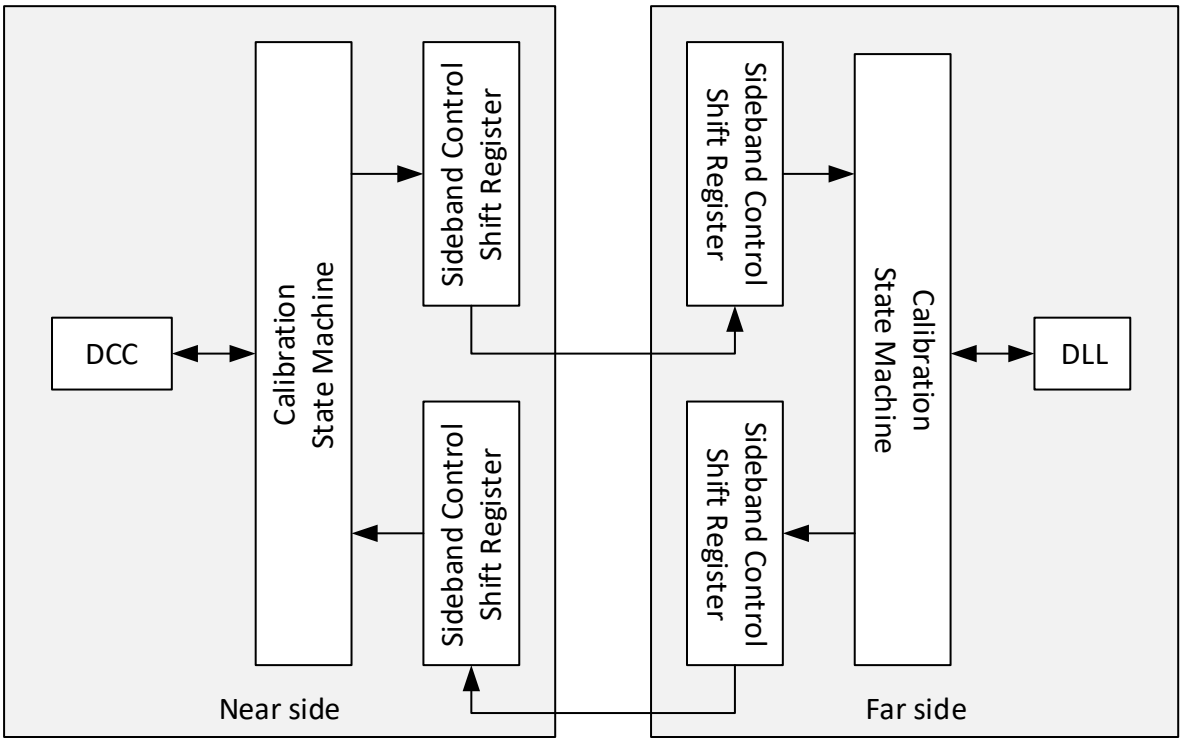


Figure 40. Data-Path Calibration Architecture

Following the de-assertion of the adapter-reset signal(s), a calibration request shall be made by asserting a calibration request signal. Separate calibration sequences shall be used for a master transmitting to a slave (Section 3.2.3.3.6) and a slave transmitting to a master (Section 3.2.3.3.7). The two sequences may occur in any order or concurrently.

3.2.3.3.1 Adapter Reset

Data-path calibration shall be initiated when the MAC layer asserts the *ns_adapter_rstn* signal LO or the far side asserts the *fs_adapter_rstn* signal LO. If the data-transfer ready signal was de-asserted prior to the start of calibration, then the *ns_mac_rdy* signal must be asserted HI prior to asserting the adapter-reset signals HI

The MAC must de-assert the adapter-reset signal prior to requesting calibration start.

3.2.3.3.2 Calibration Request

Calibration can be requested by either the master or the slave using the *ms_xx_dcc_dll_lock_req* signal or the *sl_xx_dcc_dll_lock_req* signal, respectively, where “xx” is *tx* or *rx* according to the direction of dataflow.

Calibration initiator	Dataflow direction	Initiation signal
Master	Master to slave	*ms_tx_dcc_dll_lock_req*
Slave	Master to slave	*sl_rx_dcc_dll_lock_req*
Master	Slave to master	*ms_rx_dcc_dll_lock_req*
Slave	Slave to master	*sl_tx_dcc_dll_lock_req*

Table 13. Calibration Initiation Signals

Calibration for a dataflow direction shall commence when both master and slave side have asserted their calibration request signals for that dataflow direction. Calibration request signals shall remain asserted until a new calibration is requested.

3.2.3.3.3 DCC Calibration

Upon receipt of an *xx_dcc_dll_lock_req* signal, the DCC shall be calibrated. The means of calibration is not specified and is left to the designer. If the optional DCC is not present, then the state machines in Section 3.2.3.3 shall remain the same, with the DCC calibration state serving only to provide a signal indicating DCC calibration completion.

3.2.3.3.4 DLL Calibration

Following DCC calibration, the receiving DLL shall be calibrated. The means of calibrating the DLL is not specified and is left to the designer.

If the optional DLL is not present, then the state machine in Section 3.2.3.3 shall remain the same, with the DLL lock state serving only to provide a signal indicating DLL lock completion.

3.2.3.3.5 Calibration Completion

Calibration completion shall be indicated by the following signals. Full completion shall be indicated when all four signals are asserted HI. All four signals, once asserted, shall remain asserted until a new calibration sequence is requested.

Calibration Completion Signal	Meaning
ms_tx_transfer_en	Master transmit block has completed calibration.
ms_rx_transfer_en	Master receive block has completed calibration
sl_tx_transfer_en	Slave transmit block has

	completed calibration
sl_rx_transfer_en	Slave receive block has completed calibration

Table 14. Calibration Completion Signals

3.2.3.3.6 Calibration of Master-to-Slave Datapath

Master-to-slave calibration shall comply with Figure 41. The numbers in black indicate the sequence of steps. The *ms_osc_transfer_alive* signal shall come from the free-running clock synchronization state machine (Section 3.2.3.2).

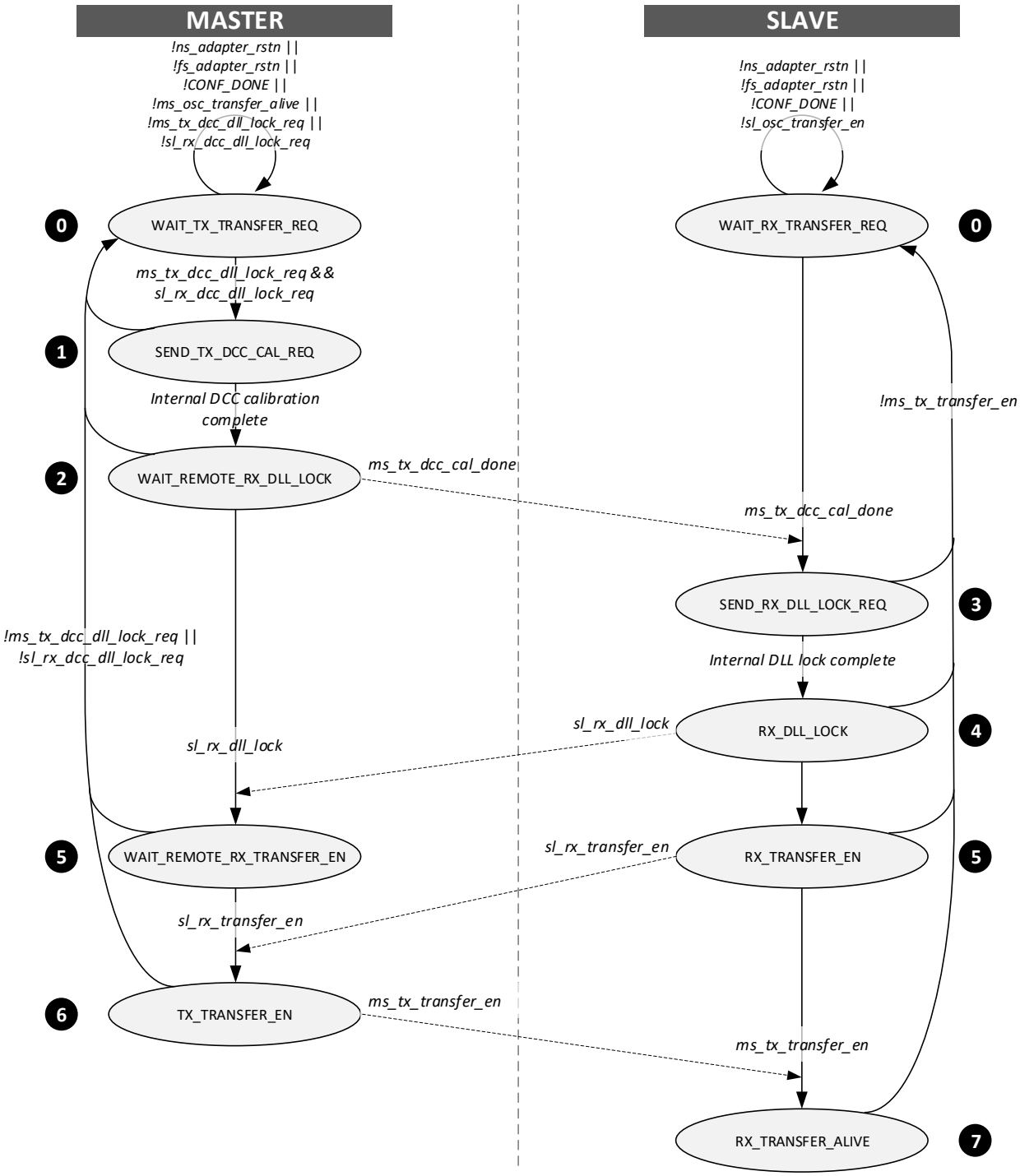


Figure 41. Master-to-Slave Datapath Calibration State Machine

Signals used in the master-to-slave calibration sequence are listed in Table 15.

Signals	Description
sl_rx_dcc_dll_lock_req	Request from slave to start calibration. Once asserted, shall remain asserted until a new calibration is requested.
ms_tx_dcc_dll_lock_req	Request from master to start calibration. Once asserted, shall remain asserted until a new calibration is requested.
ms_tx_dcc_cal_done	Indicates that master has completed its DCC calibration. Once asserted, shall remain asserted until a new calibration is requested.
sl_rx_dll_lock	Indicates that slave has completed its DLL lock procedure. Once asserted, shall remain asserted until a new calibration is requested.
sl_rx_transfer_en	Indicates that slave has completed its RX path calibration and is ready to receive data. Once asserted, shall remain asserted until calibration is complete.
ms_tx_transfer_en	Indicates that master has completed its TX path calibration and is ready to receive data.

Table 15. Master-to-Slave Calibration Signals

3.2.3.3.7 Calibration of Slave-to-Master Datapath

Slave-to-master calibration shall comply with Figure 42. The numbers in black indicate the sequence of steps. The *ms_osc_transfer_alive* signal shall come from the free-running clock synchronization state machine (Section 3.2.3.2).

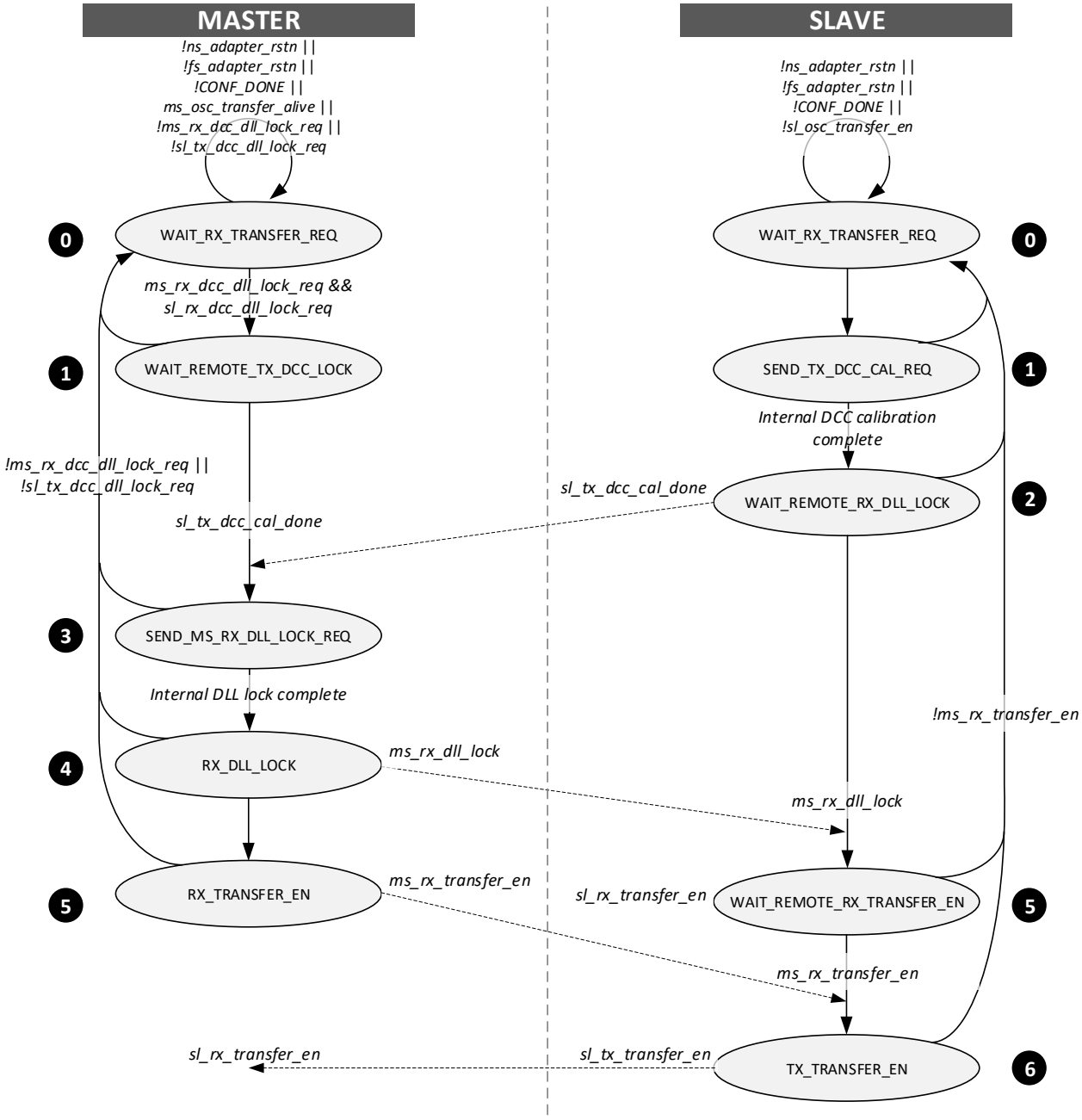


Figure 42. Slave-to-Master Datapath Calibration State Machine

Signals	Description
sl_tx_dcc_dll_lock_req	Request from slave to start calibration. Once asserted, shall remain asserted until a new calibration is requested.

Signals	Description
ms_rx_dcc_dll_lock_req	Request from master to start calibration. Once asserted, shall remain asserted until a new calibration is requested.
sl_tx_dcc_cal_done	Indicates that slave has completed its DCC calibration. Once asserted, shall remain asserted until a new calibration is requested.
ms_rx_dll_lock	Indicates that master has completed its DLL lock procedure. Once asserted, shall remain asserted until a new calibration is requested.
sl_tx_transfer_en	Indicates that slave has completed its TX path calibration and is ready to receive data. Once asserted, shall remain asserted until calibration is complete.
ms_rx_transfer_en	Indicates that master has completed its RX path calibration and is ready to receive data. Once asserted, shall remain asserted until calibration is complete.

Table 16. Slave-to-Master Calibration Signals

3.2.4 AIB Link Ready

When both *sl_tx_transfer_en* and *ms_tx_transfer_en* are true, then the link shall be ready to transmit data.

3.3 Redundancy

In order to improve interposer assembly yields, AIB interfaces shall implement a redundancy scheme. Data signals, clocks, and sideband control signals (AIB Plus only) shall implement an active redundancy scheme; *power_on_reset* and *device_detect* signals shall implement a passive redundancy scheme.

3.3.1 Active Redundancy

Within each channel, two *spare* bumps shall be provided for implementation of active redundancy.

If a connection from one chiplet to another is either open or shorted, that signal shall be rerouted two signals away. All signals in the direction of the signal shift shall also have their signals rerouted by two signals. The shift shall terminate at the *spare* bumps. Signal shifting shall occur only between the faulty signal and the *spare* signal.

This signal shift shall be implemented by both interconnected chiplets.

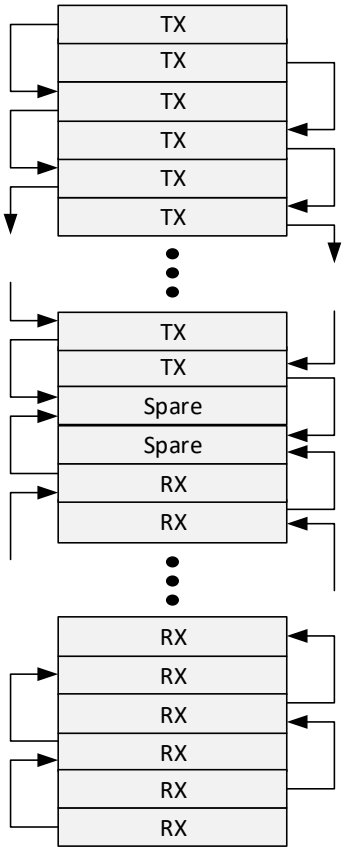


Figure 43. Redundancy Routing

When a faulty connection is detected and repaired, the output cell connected to the faulty connection shall be placed into standby mode.

For channels with unused data signals (Section 2.1.9), only used data signals shall be rerouted.

If two or more connections within one channel are faulty, then it shall not be possible to repair the connection.

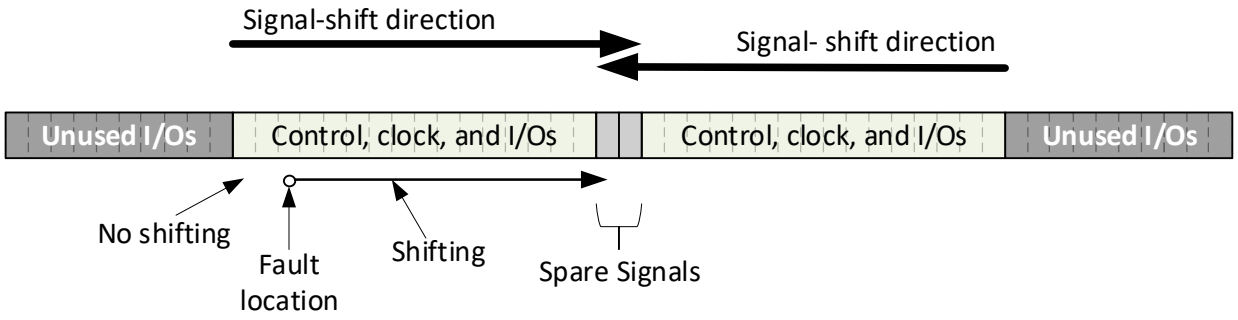


Figure 44. Direction of Redundancy Signal Shift

3.3.1.1 Neighboring-Cell Requirements

Each I/O cell, in addition to implementing its primary function, shall be capable of

implementing the function of a neighboring rerouted signal. Specifically:

- The *spare* cells shall be configurable as output or input.
- Synchronous cells (outputs, inputs, clocks) that may carry rerouted asynchronous signals (control signals) shall be capable of asynchronous mode (Section 2.1.8).
- Asynchronous cells (control signals) that may carry rerouted synchronous signals (outputs, inputs, clocks) shall be capable of SDR synchronous mode (Section 2.1.3).

3.3.1.2 Redundancy Storage

During module test, if it is determined that a signal connection within a channel between two chiplets is faulty, the necessary redundancy activation shall be determined and stored in each chiplet.

3.3.1.3 Redundancy Activation

With each power-up event, the stored redundancy information shall be retrieved and applied.

3.3.1.4 Redundancy Documentation

Each chiplet should document both how to store redundancy and how to retrieve and activate redundancy in the data sheet.

3.3.2 Passive Redundancy

Two signals shall be active prior to configuration: *power_on_reset* and *device_detect*. In order to improve yields with respect to these signals, two bumps shall be provided for each signal, and both bumps shall be soldered down to connect to two separate identical traces on the interposer. This passive redundancy scheme shall repair opens, but not shorts.

4 Electrical Specification

4.1 Eye Diagram

Compliance of data and clock signals shall be verified using a compliance mask on an eye diagram that specifies the minimum voltage swing (HI – LO), the minimum duration during which the output voltage will be stable, and the maximum allowed over- and undershoot.

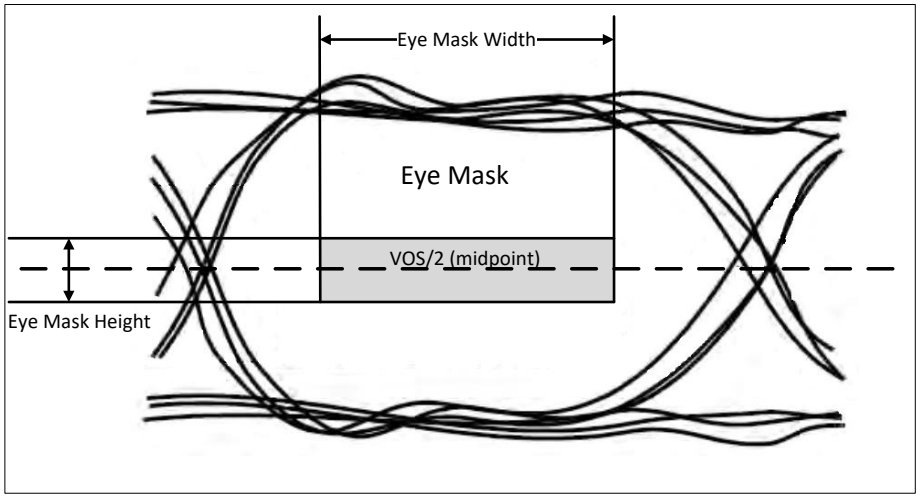


Figure 45. Compliance Eye Mask

Compliance is measured both at output bumps and at input bumps. It is applicable to both DDR/SDR modes for data, clocks or active control signals (such as **_sr_data*, and **_sr_load)*. The connection between the near end and the far end is referred to as the *trace*.

For the receive-domain clock (Section 2.1.6.4), the near end is on the receiving side, since that is where that clock originates; the far end becomes the transmitting side. Timing specifications for the receive-domain clock are more stringent than those of the forwarded clock since the receive-domain clock traverses a trace from near end to far end before being forwarded across a second trace.

Signals (data, clock, and active control signals) shall meet the voltage and timing specifications provided in Table 17.

- Delays are specified in terms of *unit interval*, or *UI*. This refers to the minimum interval between data changes, and it is therefore a function of the clock frequency.
- Near-end timing specifications reflect microbumps with 55-µm pitch and a 0.5-fF capacitive load.
- Skew parameters are measured from the center of the eye.
- Jitter margin shall include all possible jitter contributors within the chiplet, including but not limited to reference clock jitter; intrinsic jitter contributed by any phase-locked loop, clock-data recovery, delay-lock loop, or the clock network;

jitter caused by power-supply noise; and jitter caused by switching noise.

Symbol	Parameter		Near end	Trace	Far end
V_{EH}	Minimum difference between LO and HI (eye diagram height)		±90 mV		±90 mV
V_{HI}	Minimum output voltage for HI state		0.7 V		0.7 V
V_{LO}	Maximum output voltage for LO state		0 V		0 V
V_{OSmin}	Minimum output swing		0.7 V		0.7 V
V_{OSmax}	Maximum output swing		0.9 V		0.9 V
t_{EW}	Minimum duration of valid output (eye diagram width)	Data	0.56 UI	0.1 UI	0.4 UI
		Forwarded Clock	0.56 UI	0.1 UI	0.4 UI
t_{BEW}	Minimum duration of valid output (eye diagram width) for receive-domain clock		0.66 UI	0.1 UI	0.56 UI

Table 17. Electrical signal specifications

4.2 Overshoot and Undershoot

Signal overshoot and undershoot shall meet the specification in Table 18. Overshoot and Undershoot Specifications, where the symbols are illustrated in Figure 46. Overshoot and Undershoot.

Symbol	Parameter	Level	Units
V_{OA}	Maximum peak overshoot amplitude	0.25*Vos	V
V_{UA}	Maximum peak undershoot amplitude	-0.25*Vos	V
T_{OD}	Maximum overshoot duration	0.4	ns
T_{UD}	Maximum undershoot duration	0.4	ns

Table 18. Overshoot and Undershoot Specifications

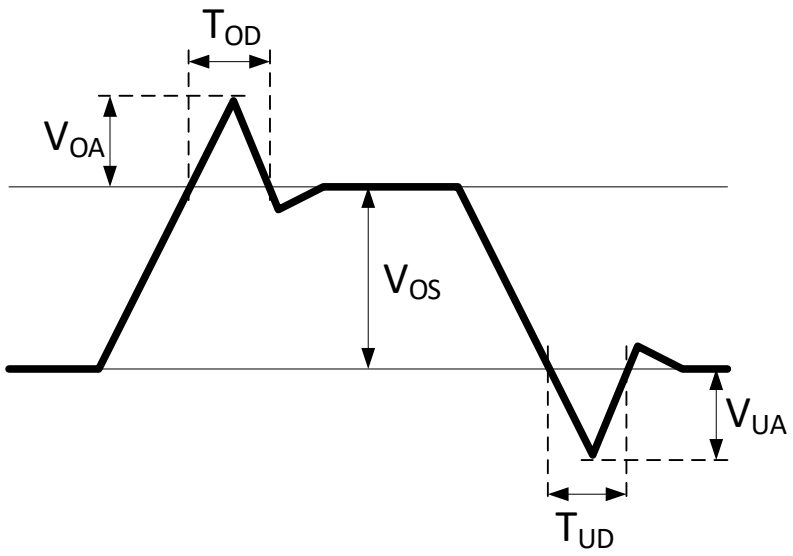


Figure 46. Overshoot and Undershoot

4.3 Electrostatic Discharge (ESD) Protection

AIB microbumps shall meet the ESD specifications in Table 19.

Parameter	Microbump Pitch	Minimum
Discharge voltage (CDM)	55 µm	70 V
	10 µm	TBD
Discharge current	55 µm	0.5 A
	10 µm	TBD

Table 19. ESD Specifications

5 JTAG

5.1 JTAG I/Os

All JTAG I/Os shall be fully compliant with the JTAG specification, IEEE 1149.1.

5.2 JTAG cells

JTAG cells are simplified to eliminate the launch register. This means that:

- Data being shifted into the scan chain will be immediately applied during the shifting process without need for transferring the data from the chain into the launch registers. As data will be presented temporarily while being shifted, the data shall cause no undesired behavior while being shifted.
- It shall be possible to connect AIB JTAG cells into a single scan chain.
- It shall not be possible to create a single scan chain that mixes both standard JTAG cells and AIB JTAG cells.

The JTAG scan chain shall include all control and data I/Os, but not the *power_on_reset* or the *device_detect* signals.

5.3 AIB Private JTAG instructions

Manipulation of an AIB interface shall use the following private JTAG instructions. The operation codes should be documented in the chiplet data sheet.

Instruction Name	Description
AIB_SHIFT_EN	Enables boundary-scan register contents to be shifted out to TDO while test data is shifted into the boundary-scan registers via TDI.
AIB_SHIFT_DIS	Disables shifting of boundary-scan data out to TDO or in from TDI.
AIB_TRANSMIT_EN	Enables the forcing of a value onto output pins.
AIB_TRANSMIT_DIS	Disables the forcing of a value onto output pins
AIB_RESET_EN	Reset the registers in the input and output cells. Not effective until AIB_RESET_OVRD_EN is asserted to override the operational reset signal.
AIB_RESET_DIS	De-assert the reset signal from the registers in the input and output cells. Not effective until AIB_RESET_OVRD_EN is asserted to override the operational reset signal.
AIB_RESET_OVRD_EN	Applies the reset state set by AIB_RESET_EN or AIB_RESET_DIS, overriding the operational reset signal.
AIB_RESET_OVRD_DIS	Applies the reset state set by the operational reset signal.
AIB_WEAKPU_EN	Enable weak pull-up on all AIB IO blocks within a column.
AIB_WEAKPU_DIS	Disable weak pull-up on all AIB IO blocks within a column.
AIB_WEAKPDN_EN	Enable weak pull-down on all AIB IO blocks within a column.
AIB_WEAKPDN_DIS	Disable weak pull-down on all AIB IO cells within a column.
AIB_INTEST_EN	Enable testing of data path between the MAC layer (AIB Base) or the AIB Adapter (AIB Plus) and the I/O block.
AIB_INTEST_DIS	Disable testing of data path between the MAC layer (AIB Base) or the AIB Adapter (AIB Plus) and the I/O block.
AIB_JTAG_CLKSEL	Select the JTAG clock or the operational clock for the register in the I/O block. Takes an argument: if HI, then the JTAG clock is selected; if LO, then the operational clock is selected.

Table 20. Private JTAG instructions

6 Physical Signal Arrangement

6.1 Interface Orientation

Die orientation shall be with respect to the die facing up with the alignment mark (or the [0,0] origin) at the lower left. Sides shall be referred to as East/West/North/South with respect to this orientation. Interface orientation should be documented in the chiplet data sheet.

- The west interface shall have Channel 0 at the bottom. This shall be a slave or dual-mode interface.
- The north interface shall have Channel 0 at the left. This shall be a slave or dual-mode interface.
- The east interface shall have Channel 0 at the bottom. This shall be a master or dual-mode interface.
- The south interface shall have Channel 0 at the left. This shall be a master or dual-mode interface.
- The AUX block shall be placed at the end of a column, next to Channel 0.

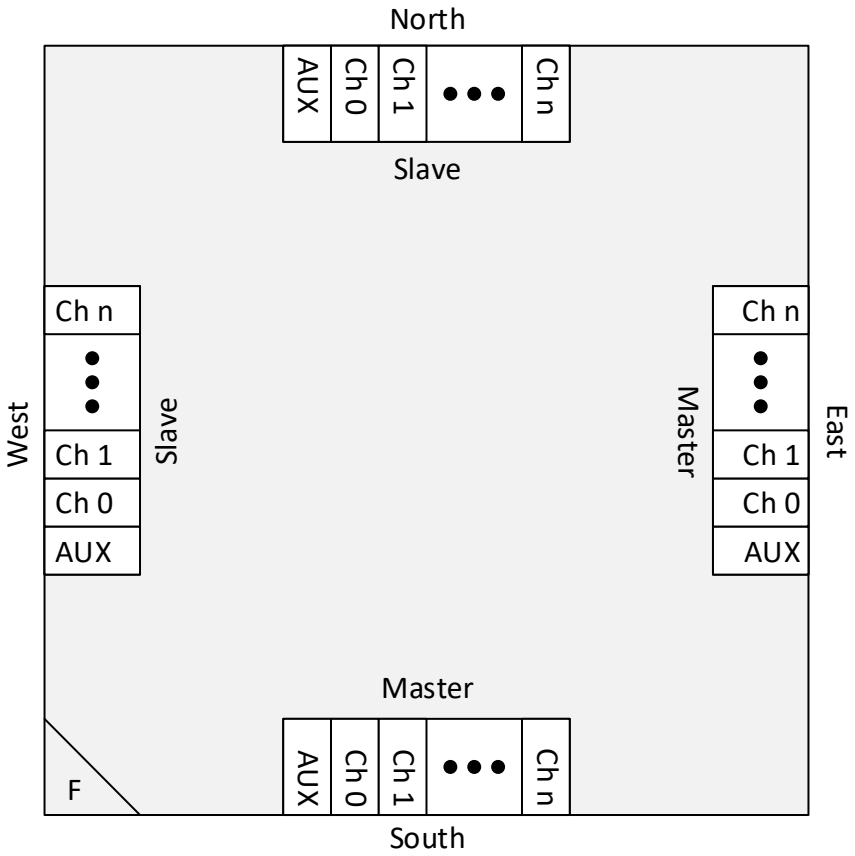


Figure 47. Interface Orientation

This convention facilitates interconnection of interfaces between chiplets in a way that ensures correct master/slave and channel/AUX interconnect (Figure 48).

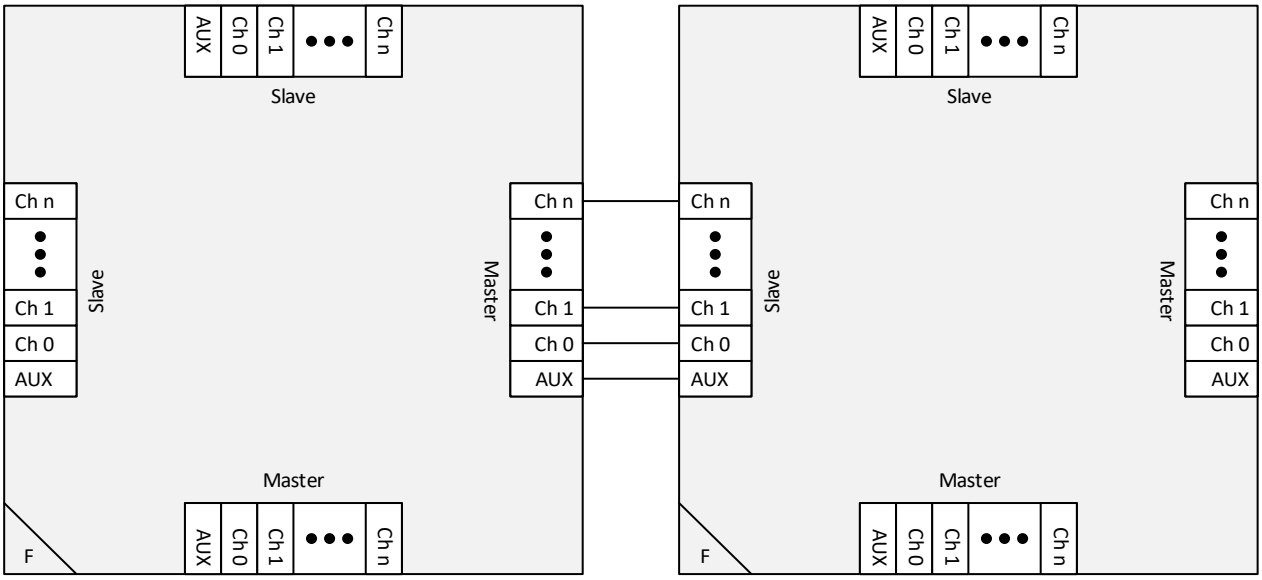


Figure 48. Standard Chiplet-to-Chiplet Interconnection

In the event of a chiplet being rotated from the standard orientation prior to being connected, the interface on the rotated chiplet must be dual-mode and configured to ensure a master/slave relationship (example shown in Figure 49). This is intended to facilitate straight, balanced signal connections. The long AUX (Section 1.3.4.4) connections are acceptable, since these are static signals.

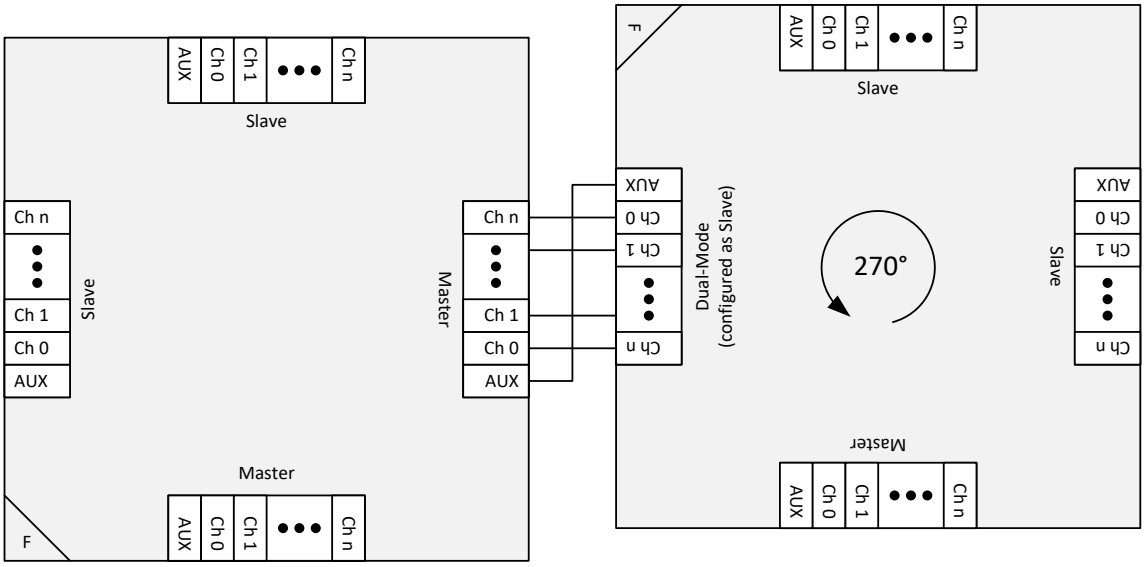


Figure 49. Rotated Chiplet-to-Chiplet Interconnection

6.2 Bump Configuration

Microbumps shall be configured in staggered arrays as shown in Figure 50.

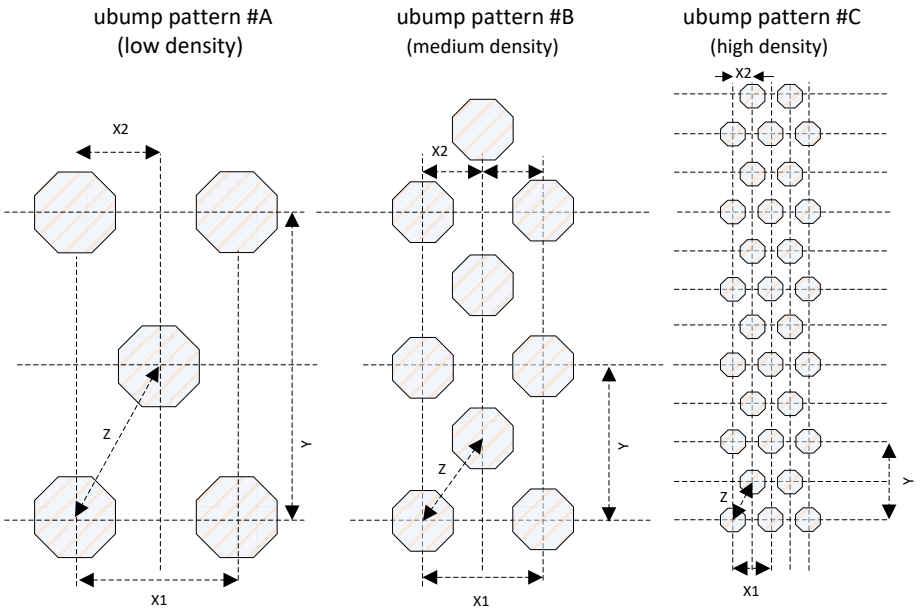


Figure 50. Bump Spacing for Microbump Array

There are three supported bump arrays for low-density, medium-density and high-density microbumps.

- Low density shall refer to any microbump pitch up to 55 µm.
- Medium density shall refer to any microbump pitch up to 52 µm.
- High density shall refer to any microbump pitch up to 20 µm.

The spacing of the microbump array shall meet the following specifications (Table 21) in µm. The X1 dimension may be reduced, but the Y dimension shall not be reduced. The channel height shall be fixed at 312.48 µm. For medium and high density arrays, depending on actual minimum bump pitch technology, either X1 or Z reaches minimum bump pitch first such that other dimension (Z or X1) will be set to non-minimum value.

Symbol	Description	Low density	Medium density	High density
X1	Aligned-row bump-to-bump pitch	55 or ≥ minimum bump pitch such that Z minimum is met	55 or ≥ minimum bump pitch such that Z minimum is met	55 or ≥ minimum bump pitch such that Z minimum is met
X2	Staggered-row bump-to-bump pitch	X1÷2	X1÷2	X1÷2
Z	Diagonal bump-to-bump pitch	≥ minimum bump pitch	≥ minimum bump pitch	≥ minimum bump pitch
Y	Aligned-column bump-to-bump pitch	104.16	52.08	26.04

Table 21. Bump Spacing Specifications for Bump Array

6.3 Bump Assignment Process

The low-density bump assignment process is based on the signal relationships for Balanced, All-TX, and All-RX configurations for both AIB Base and AIB Plus, as illustrated in Figure 51 and Figure 52. The goal of this placement process is to ensure the shortest, most direct connections between the near side and the far side. The connections between functional signals are described above. The *spare(0)* signal from the near side connects to the *spare(1)* signal on the far side; the *spare(1)* signal from the near side connects to the *spare(0)* signal on the far side.

The medium-density bump array is derived from the low-density bump array. The high-density bump array is derived from the medium-density bump array.

All bump array share the same bump table process.

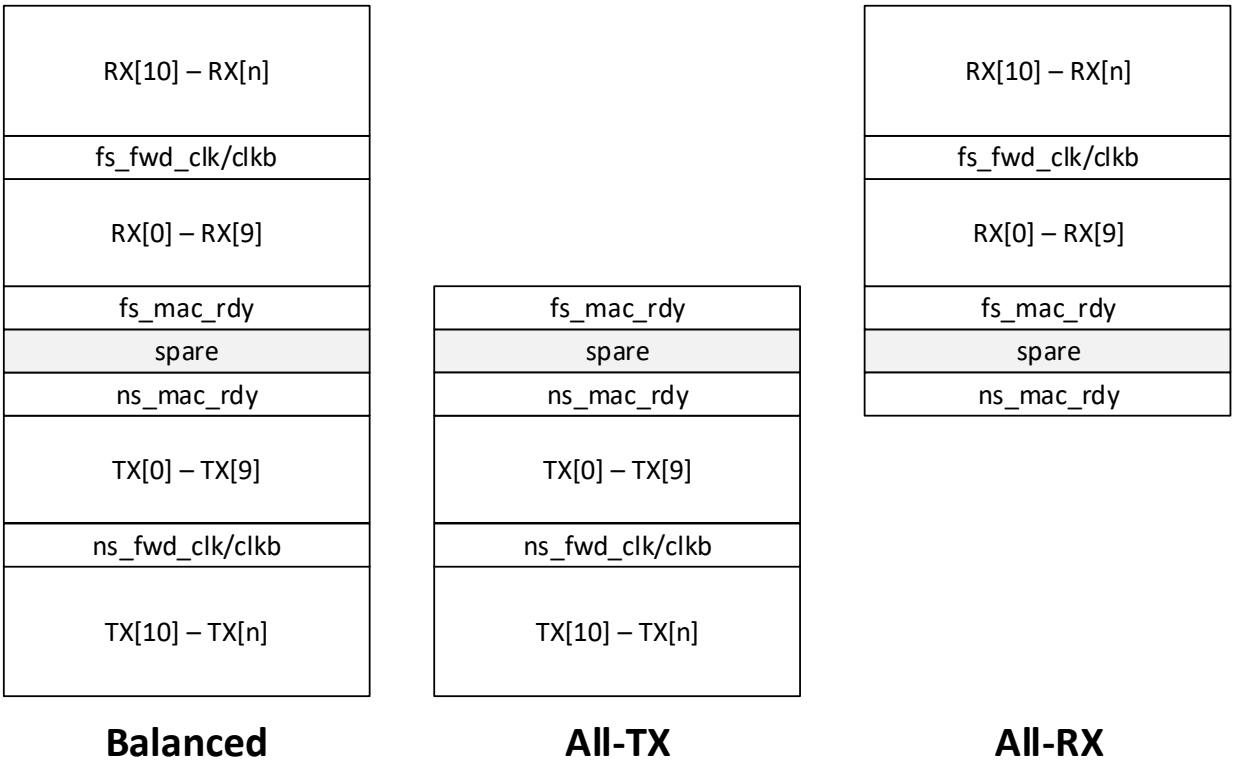


Figure 51. Signal Relationships for AIB Base Configurations

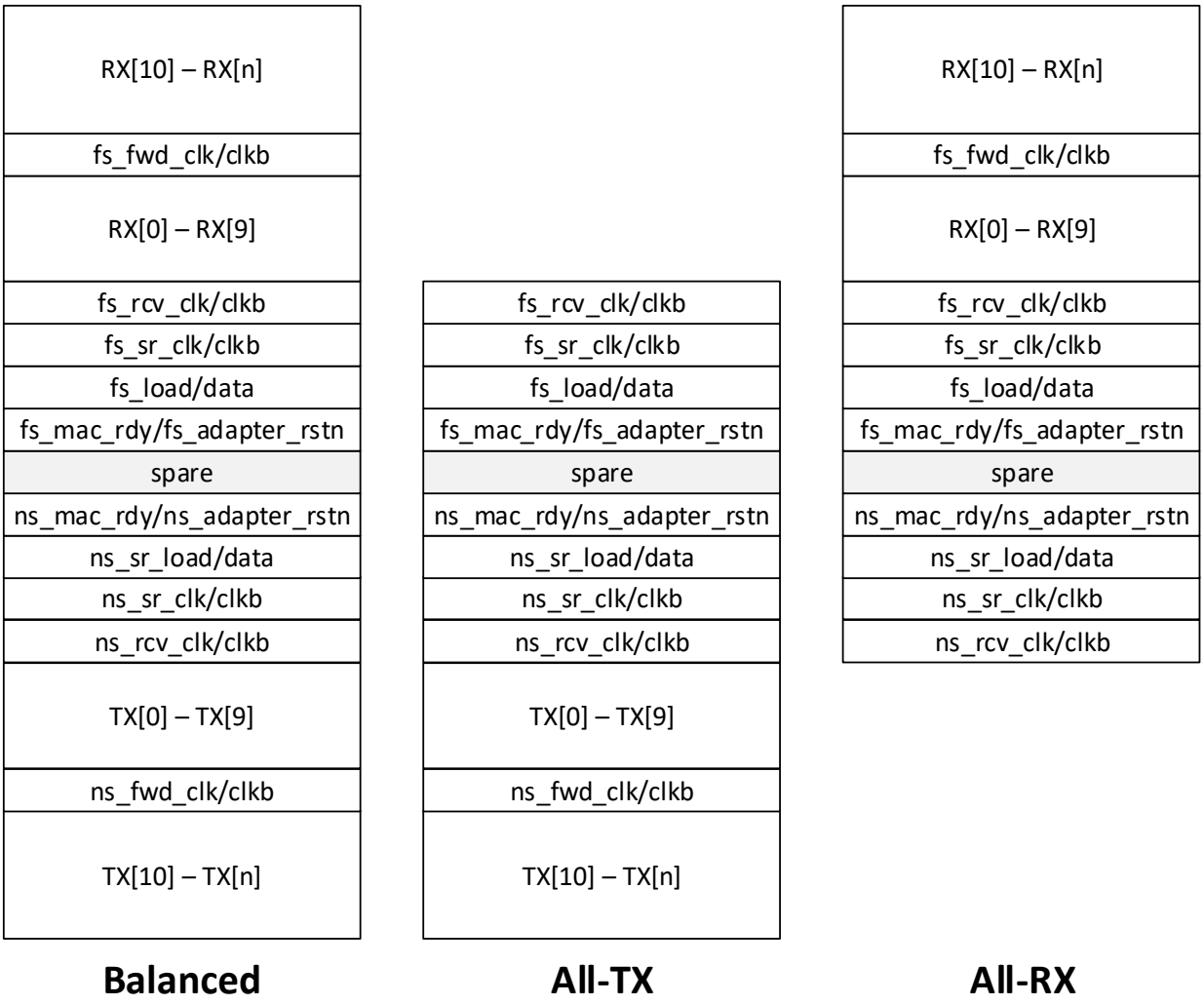


Figure 52. Signal Relationships for AIB Plus Configurations

6.3.1 Data-Signal Bump Assignment

Because a channel may have from 40 to 160 or 640 data signals in increments of 40, specific bump assignments for every possible configuration are not practical. Instead, an algorithm determines how bumps shall be assigned for any legal data signal count.

Bumps shall be assigned as follows. Any bumps labeled as “(empty)” are available for any connection that doesn’t involved an active AIB signal.

6.3.1.1 Bump Table

1. Create a *bump table* with pairs of signals. Odd-numbered signals are on the left, even-numbered signals are on the right.

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO

Table 22. Bump Table: Starting

2. The middle row of the table shall be for the *spare* signals (left and right).

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
	spare[0]	I/O		spare[1]	I/O

Table 23. Bump Table: Spare signals

6.3.1.1.1 AIB Base, Balanced

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:
   1. Empty on left (since no adapter reset); MAC ready on right
   2. Five pairs of data inputs
   3. The forwarded negative clock on left; the forwarded positive clock on right
   4. The remaining pairs of used inputs
   5. Any unused inputs

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
	RX[n-1]	in		RX[n-2]	in
	. . .			. . .	
	RX[11]	in		RX[10]	in
	fs_fwd_clkb	in		fs_fwd_clk	in
	RX[9]	in		RX[8]	in
	RX[7]	in		RX[6]	in
	RX[5]	in		RX[4]	in
	RX[3]	in		RX[2]	in
	RX[1]	in		RX[0]	in
	(empty)	in		fs_mac_rdy	in
	spare[0]	I/O		spare[1]	I/O

Table 24. Bump Table: Inputs (AIB Base, Balanced)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to

the far side, shall be placed:

1. MAC ready on left; empty on right (since no adapter reset)
2. Five pairs of data outputs
3. The forwarded positive clock on left; the forwarded negative clock on right
4. The remaining pairs of used outputs.
5. Any unused outputs

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
	RX[n-1]	in		RX[n-2]	in
	. . .			. . .	
	fs_fwd_clkb	in		fs_ fwd _clk	in
	RX[9]	in		RX[8]	in
	RX[7]	in		RX[6]	in
	RX[5]	in		RX[4]	in
	RX[3]	in		RX[2]	in
	RX[1]	in		RX[0]	in
	(empty)	in		fs_mac_rdy	in
	spare[0]	I/O		spare[1]	I/O
	ns_mac_rdy	out		(empty)	out
	TX[0]	out		TX[1]	out
	TX[2]	out		TX[3]	out
	TX[4]	out		TX[5]	out
	TX[6]	out		TX[7]	out
	TX[8]	out		TX[9]	out
	ns_ fwd _clk	out		ns_ fwd _clkb	out
	TX[10]	out		TX[11]	out
	. . .			. . .	
	TX[n-2]	out		TX[n-1]	out

Table 25. Bump Table: Complete (AIB Base, Balanced)

6.3.1.1.2 AIB Base, All-TX

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:
    1. Empty on left (since no adapter reset); MAC ready on right

Bump ID	Bump Name	IO
	(empty)	in
	spare[0]	I/O

Bump ID	Bump Name	IO
	fs_mac_rdy	in
	spare[1]	I/O

Table 26. Bump Table: Inputs (AIB Base, All-TX)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to the far side, shall be placed:
    1. MAC ready on left; empty on right (since no adapter reset)
    2. Five pairs of data outputs
    3. The forwarded positive clock on left; the forwarded negative clock on right
    4. The remaining pairs of used outputs.
    5. Any unused outputs

Bump ID	Bump Name	IO
	(empty)	in
	spare[0]	I/O
	ns_mac_rdy	out
	TX[0]	out
	TX[2]	out
	TX[4]	out
	TX[6]	out
	TX[8]	out
	ns_ fwd _clk	out
	TX[10]	out
	. . .	
	TX[n-2]	out

Bump ID	Bump Name	IO
	fs_mac_rdy	in
	spare[1]	I/O
	(empty)	out
	TX[1]	out
	TX[3]	out
	TX[5]	out
	TX[7]	out
	TX[9]	out
	ns_ fwd _clkb	out
	TX[11]	out
	. . .	
	TX[n-1]	out

Table 27. Bump Table: Complete (AIB Base, All-TX)

6.3.1.1.3 AIB Base, All-RX

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:
    1. Empty on left (since no adapter reset); MAC ready on right
    2. Five pairs of data inputs
    3. The forwarded negative clock on left; the forwarded positive clock on right
    4. The remaining pairs of used inputs
    5. Any unused inputs

Bump ID	Bump Name	IO
	RX[n-1]	in
	. . .	
	RX[11]	in
	fs_fwd_clkb	in
	RX[9]	in
	RX[7]	in
	RX[5]	in
	RX[3]	in
	RX[1]	in
	(empty)	in
	spare[0]	I/O

Bump ID	Bump Name	IO
	RX[n-2]	in
	. . .	
	RX[10]	in
	fs_fwd_clk	in
	RX[8]	in
	RX[6]	in
	RX[4]	in
	RX[2]	in
	RX[0]	in
	fs_mac_rdy	in
	spare[1]	I/O

Table 28. Bump Table: Inputs (AIB Base, All-RX)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to the far side, shall be placed:
    1. MAC ready on left; empty on right (since no adapter reset)

Bump ID	Bump Name	IO
	RX[n-1]	in
	. . .	
	fs_fwd_clkb	in
	RX[9]	in
	RX[7]	in
	RX[5]	in
	RX[3]	in
	RX[1]	in
	(empty)	in
	spare[0]	I/O
	ns_mac_rdy	out

Bump ID	Bump Name	IO
	RX[n-2]	in
	. . .	
	fs_fwd_clk	in
	RX[8]	in
	RX[6]	in
	RX[4]	in
	RX[2]	in
	RX[0]	in
	fs_mac_rdy	in
	spare[1]	I/O
	(empty)	out

Table 29. Bump Table: Complete (AIB Base, All-RX)

6.3.1.1.4 AIB Plus, Balanced

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:

1. Adapter reset on left; MAC ready on right
2. Shift-register load on left; shift-register data on right
3. Shift-register negative clock on left; shift-register positive clock on right
4. The receive-domain negative clock on left; the receive-domain positive clock on right
5. Five pairs of data inputs
6. The forwarded negative clock on left; the forwarded positive clock on right
7. The remaining pairs of used inputs
8. Any unused inputs

Bump ID	Bump Name	IO
	RX[n-1]	in
	. . .	
	RX[11]	in
	fs_fwd_clkb	in
	RX[9]	in
	RX[7]	in
	RX[5]	in
	RX[3]	in
	RX[1]	in
	fs_rcv_clkb	in
	fs_sr_clkb	in
	fs_sr_load	in
	fs_adapter_rstn	in
	spare[0]	I/O

Bump ID	Bump Name	IO
	RX[n-2]	in
	. . .	
	RX[10]	in
	fs_fwd_clk	in
	RX[8]	in
	RX[6]	in
	RX[4]	in
	RX[2]	in
	RX[0]	in
	fs_rcv_clk	in
	fs_sr_clk	in
	fs_sr_data	in
	fs_mac_rdy	in
	spare[1]	I/O

Table 30. Bump Table: Inputs (AIB Plus, Balanced)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to the far side, shall be placed:
    1. MAC ready on left; adapter reset on right
    2. Shift-register data on left; shift-register load on right
    3. Shift-register positive clock on left; shift-register negative clock on right
    4. The receive-domain positive clock on left; the receive-domain negative clock on right
    5. Five pairs of data outputs
    6. The forwarded positive clock on left; the forwarded negative clock on right
    7. The remaining pairs of used outputs.
    8. Any unused outputs

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
	RX[n-1]	in		RX[n-2]	in
	. . .			. . .	
	fs_fwd_clkb	in		fs_ fwd _clk	in
	RX[9]	in		RX[8]	in
	RX[7]	in		RX[6]	in
	RX[5]	in		RX[4]	in
	RX[3]	in		RX[2]	in
	RX[1]	in		RX[0]	in
	fs_ rcv _clkb	in		fs_ rcv _clk	
	fs_sr_clkb	in		fs_sr_clk	in
	fs_sr_load	in		fs_sr_data	in
	fs_adapter_rstn	in		fs_mac_rdy	in
	spare[0]	I/O		spare[1]	I/O
	ns_mac_rdy	out		ns_adapter_rstn	out
	ns_sr_data	out		ns_sr_load	out
	ns_sr_clk	out		ns_sr_clkb	out
	ns_ rcv _clk	out		ns_ rcv _clkb	out
	TX[0]	out		TX[1]	out
	TX[2]	out		TX[3]	out
	TX[4]	out		TX[5]	out
	TX[6]	out		TX[7]	out
	TX[8]	out		TX[9]	out
	ns_ fwd _clk	out		ns_ fwd _clkb	out
	TX[10]	out		TX[11]	out
	. . .			. . .	
	TX[n-2]	out	AIB1	TX[n-1]	out

Table 31. Bump Table: Complete (AIB Plus, Balanced)

6.3.1.1.5 AIB Plus, All-TX

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:
    1. Adapter reset on left; MAC ready on right
    2. Shift-register load on left; shift-register data on right
    3. Shift-register negative clock on left; shift-register positive clock on right
    4. The receive-domain negative clock on left; the receive-domain positive clock on right

Bump ID	Bump Name	IO
	fs_rcv_clkb	in
	fs_sr_clkb	in
	fs_sr_load	in
	fs_adapter_rstn	in
	spare[0]	I/O

Bump ID	Bump Name	IO
	fs_rcv_clk	in
	fs_sr_clk	in
	fs_sr_data	in
	fs_mac_rdy	in
	spare[1]	I/O

Table 32. Bump Table: Inputs (AIB Plus, All-TX)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to the far side, shall be placed:
    1. MAC ready on left; adapter reset on right
    2. Shift-register data on left; shift-register load on right
    3. Shift-register positive clock on left; shift-register negative clock on right
    4. Empty bumps since there is no receive-domain clock output
    5. Five pairs of data outputs
    6. The forwarded positive clock on left; the forwarded negative clock on right
    7. The remaining pairs of used outputs.
    8. Any unused outputs

Bump ID	Bump Name	IO
	fs_ rcv _clkb	in
	fs_sr_clkb	in
	fs_sr_load	in
	fs_adapter_rstn	in
	spare[0]	I/O
	ns_mac_rdy	out
	ns_sr_data	out
	ns_sr_clk	out
	(empty)	out
	TX[0]	out
	TX[2]	out
	TX[4]	out
	TX[6]	out
	TX[8]	out
	ns_ fwd _clk	out
	TX[10]	out
	. . .	
	TX[n-2]	out

Bump ID	Bump Name	IO
	fs_ rcv _clk	
	fs_sr_clk	in
	fs_sr_data	in
	fs_mac_rdy	in
	spare[1]	I/O
	ns_adapter_rstn	out
	ns_sr_load	out
	ns_sr_clkb	out
	(empty)	out
	TX[1]	out
	TX[3]	out
	TX[5]	out
	TX[7]	out
	TX[9]	out
	ns_ fwd _clkb	out
	TX[11]	out
	. . .	
	TX[n-1]	out

Table 33. Bump Table: Complete

(AIB Plus, All-TX)

6.3.1.1.6 AIB Plus, All-RX

3. Moving up from the middle, in order, the following signals, which are all inputs from the far side, shall be placed:
   1. Adapter reset on left; MAC ready on right
   2. Shift-register load on left; shift-register data on right
   3. Shift-register negative clock on left; shift-register positive clock on right
   4. Empty bumps since there is no receive-domain clock input
   5. Five pairs of data inputs
   6. The forwarded negative clock on left; the forwarded positive clock on right
   7. The remaining pairs of used inputs
   8. Any unused inputs

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
	RX[n-1]	in		RX[n-2]	in
	. . .			. . .	
	RX[11]	in		RX[10]	in
	fs_fwd_clkb	in		fs_fwd_clk	in
	RX[9]	in		RX[8]	in
	RX[7]	in		RX[6]	in
	RX[5]	in		RX[4]	in
	RX[3]	in		RX[2]	in
	RX[1]	in		RX[0]	in
	(empty)	in		(empty)	in
	fs_sr_clkb	in		fs_sr_clk	in
	fs_sr_load	in		fs_sr_data	in
	fs_adapter_rstn	in		fs_mac_rdy	in
	spare[0]	I/O		spare[1]	I/O

Table 34. Bump Table: Inputs (AIB Plus, All-RX)

4. The “edge of chiplet” shall be directly above the topmost row.
5. Moving down from the *spare* signal, the following signals, which are all outputs to the far side, shall be placed:
   1. MAC ready on left; adapter reset on right
   2. Shift-register data on left; shift-register load on right
   3. Shift-register positive clock on left; shift-register negative clock on right
   4. The forwarded positive clock on left; the forwarded negative clock on right
   5. Five pairs of data outputs
   6. The borrowed positive clock on left; the borrowed negative clock on right

7. The remaining pairs of used outputs.
8. Any unused outputs

Bump ID	Bump Name	IO
	RX[n-1]	in
	. . .	
	fs_fwd_clkb	in
	RX[9]	in
	RX[7]	in
	RX[5]	in
	RX[3]	in
	RX[1]	in
	(empty)	in
	fs_sr_clkb	in
	fs_sr_load	in
	fs_adapter_rstn	in
	spare[0]	I/O
	ns_mac_rdy	out
	ns_sr_data	out
	ns_sr_clk	out
	ns_ rcv _clk	out

Bump ID	Bump Name	IO
	RX[n-2]	in
	. . .	
	fs_ fwd _clk	in
	RX[8]	in
	RX[6]	in
	RX[4]	in
	RX[2]	in
	RX[0]	in
	(empty)	
	fs_sr_clk	in
	fs_sr_data	in
	fs_mac_rdy	in
	spare[1]	I/O
	ns_adapter_rstn	out
	ns_sr_load	out
	ns_sr_clkb	out
	ns_ rcv _clkb	out

Table 35. Bump Table: Complete (AIB Plus, All-RX)

6. Finally, assign bump ID starting at the bottom left with AIB0, bottom right with AIB1, and then moving up – even numbers on the left, odd numbers on the right.

The following tables illustrate the six versions of this algorithm for AIB Base and AIB Plus; Balanced, All-TX, and All-RX configurations. *n* represents the number of RX and/or TX signals.

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBy	RX[n-1]	in	AIBy+1	RX[n-2]	in
	. . .			. . .	
AIBx+28	fs_fwd_clkb	in	AIBx+29	fs_fwd_clk	in
AIBx+26	RX[9]	in	AIBx+27	RX[8]	in
AIBx+24	RX[7]	in	AIBx+25	RX[6]	in
AIBx+22	RX[5]	in	AIBx+23	RX[4]	in
AIBx+20	RX[3]	in	AIBx+21	RX[2]	in
AIBx+18	RX[1]	in	AIBx+19	RX[0]	in
AIBx+16	(empty)	in	AIBx+17	fs_mac_rdy	in
AIBx+14	spare[0]	I/O	AIBx+15	spare[1]	I/O
AIBx+12	ns_mac_rdy	out	AIBx+13	(empty)	out
AIBx+10	TX[0]	out	AIBx+11	TX[1]	out
AIBx+8	TX[2]	out	AIBx+9	TX[3]	out
AIBx+6	TX[4]	out	AIBx+7	TX[5]	out
AIBx+4	TX[6]	out	AIBx+5	TX[7]	out
AIBx+2	TX[8]	out	AIBx+3	TX[9]	out
AIBx	ns_fwd_clk	out	AIBx+1	ns_fwd_clkb	out
	. . .			. . .	
AIB0	TX[n-2]	out	AIB1	TX[n-1]	out

Table 36. Bump-Table Exemplar (AIB Base, Balanced)
x=n/2-10; y=n+8

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBx+16	(empty)	in	AIBx+17	fs_mac_rdy	in
AIBx+14	spare[0]	I/O	AIBx+15	spare[1]	I/O
AIBx+12	ns_mac_rdy	out	AIBx+13	(empty)	out
AIBx+10	TX[0]	out	AIBx+11	TX[1]	out
AIBx+8	TX[2]	out	AIBx+9	TX[3]	out
AIBx+6	TX[4]	out	AIBx+7	TX[5]	out
AIBx+4	TX[6]	out	AIBx+5	TX[7]	out
AIBx+2	TX[8]	out	AIBx+3	TX[9]	out
AIBx	ns_fwd_clk	out	AIBx+1	ns_fwd_clkb	out
	. . .			. . .	
AIB0	TX[n-2]	out	AIB1	TX[n-1]	out

Table 37. Bump-Table Exemplar (AIB Base, All-TX)
x=n-10

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBy	RX[n-1]	in	AIBy+1	RX[n-2]	in
	. . .			. . .	
AIB18	fs_fwd_clkb	in	AIB19	fs_fwd_clk	in
AIB16	RX[9]	in	AIB17	RX[8]	in
AIB14	RX[7]	in	AIB15	RX[6]	in
AIB12	RX[5]	in	AIB13	RX[4]	in
AIB10	RX[3]	in	AIB11	RX[2]	in
AIB8	RX[1]	in	AIB9	RX[0]	in
AIB6	(empty)	in	AIB7	fs_mac_rdy	in
AIB4	spare[0]	I/O	AIB5	spare[1]	I/O
AIB2	ns_mac_rdy	out	AIB3	(empty)	out

Table 38. Bump-Table Exemplar (AIB Base, All-RX)
y=n+8

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBy	RX[n-1]	in	AIBy+1	RX[n-2]	in
	. . .			. . .	
AIBx+44	RX[11]	in	AIBx+45	RX[10]	in
AIBx+42	fs_fwd_clkb	in	AIBx+43	fs_fwd_clk	in
AIBx+40	RX[9]	in	AIBx+41	RX[8]	in
AIBx+38	RX[7]	in	AIBx+39	RX[6]	in
AIBx+36	RX[5]	in	AIBx+37	RX[4]	in
AIBx+34	RX[3]	in	AIBx+35	RX[2]	in
AIBx+32	RX[1]	in	AIBx+33	RX[0]	in
AIBx+30	fs_rcv_clkb	in	AIBx+31	fs_rcv_clk	in
AIBx+28	fs_sr_clkb	in	AIBx+29	fs_sr_clk	in
AIBx+26	fs_sr_load	in	AIBx+27	fs_sr_data	in
AIBx+24	fs_adapter_rstn	in	AIBx+25	fs_mac_rdy	in
AIBx+22	spare[0]	I/O	AIBx+23	spare[1]	I/O
AIBx+20	ns_mac_rdy	out	AIBx+21	ns_adapter_rstn	out
AIBx+18	ns_sr_data	out	AIBx+19	ns_sr_load	out
AIBx+16	ns_sr_clk	out	AIBx+17	ns_sr_clkb	out
AIBx+14	ns_rcv_clk	out	AIBx+15	ns_rcv_clkb	out
AIBx+12	TX[0]	out	AIBx+13	TX[1]	out
AIBx+10	TX[2]	out	AIBx+11	TX[3]	out
AIBx+8	TX[4]	out	AIBx+9	TX[5]	out
AIBx+6	TX[6]	out	AIBx+7	TX[7]	out
AIBx+4	TX[8]	out	AIBx+5	TX[9]	out
AIBx+2	ns_fwd_clk	out	AIBx+3	ns_fwd_clkb	out
AIBx	TX[10]	out	AIBx+1	TX[11]	out
	. . .			. . .	
AIB0	TX[n-2]	out	AIB1	TX[n-1]	out

Table 39. Bump-Table Exemplar (AIB Plus, Balanced)
x=n/2-12; y=n+20

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBx+30	fs_rcv_clkb	in	AIBx+31	fs_rcv_clk	in
AIBx+28	fs_sr_clkb	in	AIBx+29	fs_sr_clk	in
AIBx+26	fs_sr_load	in	AIBx+27	fs_sr_data	in
AIBx+24	fs_adapter_rstn	in	AIBx+25	fs_mac_rdy	in
AIBx+22	spare[0]	I/O	AIBx+23	spare[1]	I/O
AIBx+20	ns_mac_rdy	out	AIBx+21	ns_adapter_rstn	out
AIBx+18	ns_sr_data	out	AIBx+19	ns_sr_load	out
AIBx+16	ns_sr_clk	out	AIBx+17	ns_sr_clkb	out
AIBx+14	(empty)	out	AIBx+15	(empty)	out
AIBx+12	TX[0]	out	AIBx+13	TX[1]	out
AIBx+10	TX[2]	out	AIBx+11	TX[3]	out
AIBx+8	TX[4]	out	AIBx+9	TX[5]	out
AIBx+6	TX[6]	out	AIBx+7	TX[7]	out
AIBx+4	TX[8]	out	AIBx+5	TX[9]	out
AIBx+2	ns_fwd_clk	out	AIBx+3	ns_fwd_clkb	out
AIBx	TX[10]	out	AIBx+1	TX[11]	out
	. . .			. . .	
AIBxx	TX[n-2]	out	AIBxx	TX[n-1]	out

Table 40. Bump-Table Exemplar (AIB Plus, All-TX)
x=n-12

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIBy	RX[n-1]	in	AIBy+1	RX[n-2]	in
	. . .			. . .	
AIB32	RX[13]	in	AIB33	RX[12]	in
AIB30	RX[11]	in	AIB31	RX[10]	in
AIB28	fs_fwd_clkb	in	AIB29	fs_fwd_clk	in
AIB26	RX[9]	in	AIB27	RX[8]	in
AIB24	RX[7]	in	AIB25	RX[6]	in
AIB22	RX[5]	in	AIB23	RX[4]	in
AIB20	RX[3]	in	AIB21	RX[2]	in
AIB18	RX[1]	in	AIB19	RX[0]	in
AIB16	(empty)	in	AIB17	(empty)	in
AIB14	fs_sr_clkb	in	AIB15	fs_sr_clk	in
AIB12	fs_sr_load	in	AIB13	fs_sr_data	in
AIB10	fs_adapter_rstn	in	AIB11	fs_mac_rdy	in
AIB8	spare[0]	I/O	AIB9	spare[1]	I/O
AIB6	ns_mac_rdy	out	AIB7	ns_adapter_rstn	out
AIB4	ns_sr_data	out	AIB5	ns_sr_load	out
AIB2	ns_sr_clk	out	AIB3	ns_sr_clkb	out
AIB0	ns_rcv_clk	out	AIB1	ns_rcv_clkb	out

Table 41. Bump-Table Exemplar (AIB Plus, All-RX)
y=n+18

6.3.1.2 Bump Map

Create a *bump map* for low-density bump array as follows:

1. Allocate six columns and enough rows to accommodate all signals. “Row” and “Column” apply to a bump array with the side of the chiplet on the top.
2. Bumps are in a staggered array. Every other row starting with the first row is used for odd-numbered bumps. Every other row starting with the second row is used for even-numbered bumps.
3. If Balanced then:
   3.1. The middle two bumps (middle two rows, middle two columns) receive the *spare* signal bumps (odd and even).
4. If All-TX then
   4.1. If AIB Base then
      4.1.1. The two bumps in middle two columns, the row for the *spare* signals are calculated as follows, where *n* is the bump ID of the odd *spare* signal in the bump table:
         4.1.1.1. (n-1) mod 3 + 1 for the odd-spare row, middle columns
         4.1.1.2. One row down for the even-spare row, middle columns
   4.2. If AIB Plus then
         4.2.1.1. (n-1) mod 3 + 1 for the odd-spare row, middle columns
         4.2.1.2. One row down for the even-spare row, middle columns
5. If All-RX then:
   5.1. If AIB Base then:
      5.1.1. The middle columns, 3rd and 4th rows, receive the *spare* signal bumps (odd and even).
   5.2. If AIB Plus then:
      5.2.1. The middle columns, 5th and 6th rows, receive the *spare* signal bumps (odd and even).
6. For odd-numbered bumps, move first to the left from the *spare* bump and assign odd-numbered bumps in descending order until the left-most column is reached. From there, move up two rows and continue, this time to the right. When the rightmost column is reached, move up again two more rows and proceed back to the left. Continue in this winding fashion (boustrophedon) until all odd-numbered bumps have been assigned (figure). Any remaining bumps in the final row remain unassigned.
7. Moving back to the odd-numbered *spare* bump, move to the right and assign odd-numbered bumps in ascending order until the rightmost column is reached. From there, move down two rows and continue, this time to the left. Continue the boustrophedon movement until all remaining odd-numbered bumps have been assigned (figure), leaving any remaining bumps in the last row unassigned.
8. Repeat this process for even-numbered bumps starting from the even *spare* bump and using the even-bump rows (figure).
9. Repeat for all channels in the AIB interface.

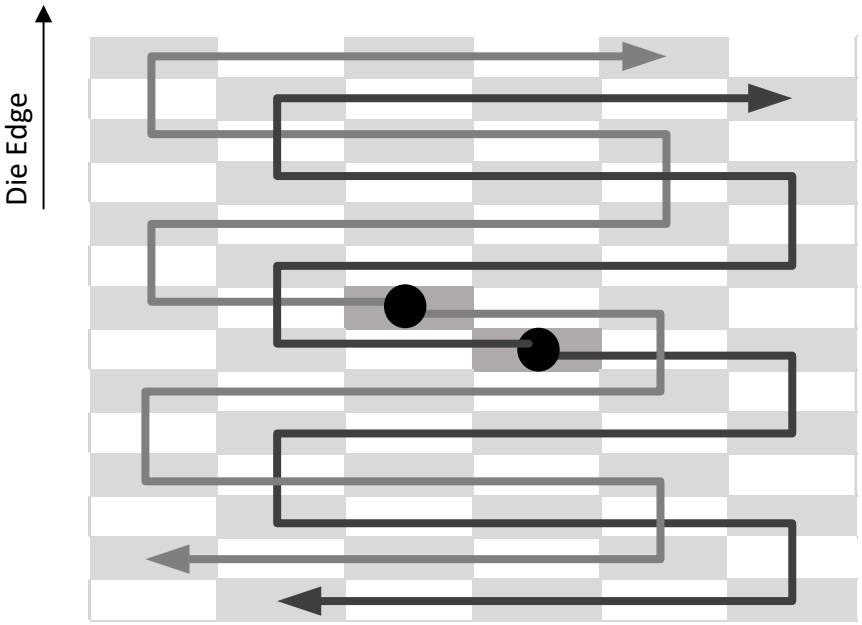


Figure 53. Bump Map Assignment Order

The bump map for medium-density pitch shall be created by combining two staggered low-density rows into a single row. The bump map for high-density pitch shall be created by combining two staggered medium-density rows into a single row as shown in Figure 54.

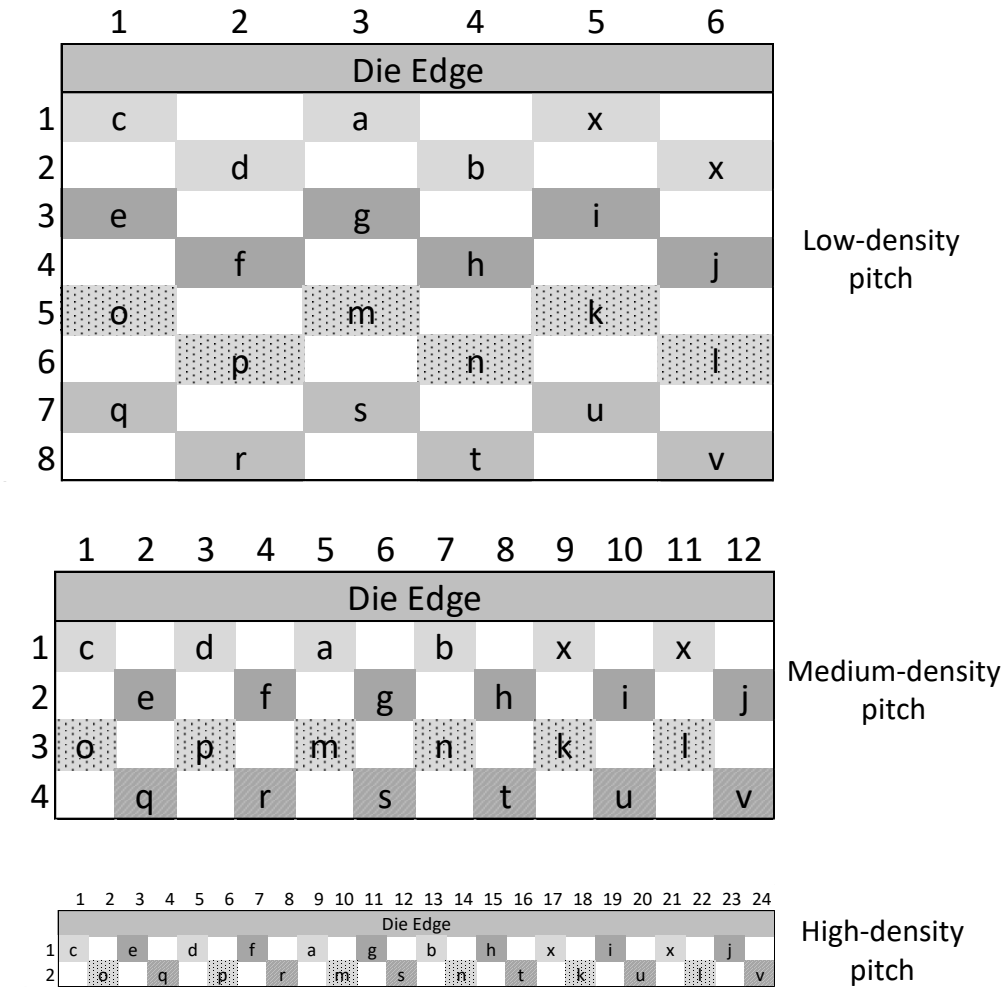


Figure 54 Bump Map Migration Between Different Bump Densities

The IO physical placement shall not necessary be the same as the bump placement as a result of optimization of the die area, IO performance and power network. IO physical placement is not part of the AIB specification.

6.3.2 AIB AUX Signal Assignment

The signals from the AIB AUX block (*power_on_reset*, *device_detect*) shall be placed below channel 0 in a column. Master chiplets shall have output bumps for *device_detect* signals and input bumps for *power_on_reset*. Slave chiplets shall have input bumps for *device_detect* signals and output bumps for *power_on_reset.* There shall be two bumps per signal for passive redundancy (Section 3.3.2). The AUX bump table and map are shown below.

Master chiplet				Slave chiplet		
Bump ID	Bump Name	IO		Bump ID	Bump Name	IO
AIBX3	device_detect	out		AIBX3	device_detect	in
AIBX2	device_detect	out		AIBX2	device_detect	in
AIBX1	por	in		AIBX1	por	out
AIBX0	por	in		AIBX0	por	out

Table 42. AIB AUX Signal Bump Table

6.3.3 Example Bump Tables

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB48	RX[19]	in	AIB49	RX[18]	in
AIB46	RX[17]	in	AIB47	RX[16]	in
AIB44	RX[15]	in	AIB45	RX[14]	in
AIB42	RX[13]	in	AIB43	RX[12]	in
AIB40	RX[11]	in	AIB41	RX[10]	in
AIB38	fs_fwd_clkb	in	AIB39	fs_fwd_clk	in
AIB36	RX[9]	in	AIB37	RX[8]	in
AIB34	RX[7]	in	AIB35	RX[6]	in
AIB32	RX[5]	in	AIB33	RX[4]	in
AIB30	RX[3]	in	AIB31	RX[2]	in
AIB28	RX[1]	in	AIB29	RX[0]	in
AIB26	(empty)	in	AIB27	fs_mac_rdy	in
AIB24	spare[0]	I/O	AIB25	spare[1]	I/O
AIB22	ns_mac_rdy	out	AIB23	(empty)	out
AIB20	TX[0]	out	AIB21	TX[1]	out
AIB18	TX[2]	out	AIB19	TX[3]	out
AIB16	TX[4]	out	AIB17	TX[5]	out
AIB14	TX[6]	out	AIB15	TX[7]	out
AIB12	TX[8]	out	AIB13	TX[9]	out
AIB10	ns_fwd_clk	out	AIB11	ns_fwd_clkb	out
AIB8	TX[10]	out	AIB9	TX[11]	out
AIB6	TX[12]	out	AIB7	TX[13]	out
AIB4	TX[14]	out	AIB5	TX[15]	out
AIB2	TX[16]	out	AIB3	TX[17]	out
AIB0	TX[18]	out	AIB1	TX[19]	out

Table 43. Example Bump Table (AIB Base, 40 I/Os, Balanced)

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB28	RX[19]	in	AIB29	RX[18]	in
AIB26	RX[17]	in	AIB27	RX[16]	in
AIB24	RX[15]	in	AIB25	RX[14]	in
AIB22	RX[13]	in	AIB23	RX[12]	in
AIB20	RX[11]	in	AIB21	RX[10]	in
AIB18	fs_fwd_clkb	in	AIB19	fs_fwd_clk	in
AIB16	RX[9]	in	AIB17	RX[8]	in
AIB14	RX[7]	in	AIB15	RX[6]	in
AIB12	RX[5]	in	AIB13	RX[4]	in
AIB10	RX[3]	in	AIB11	RX[2]	in
AIB8	RX[1]	in	AIB9	RX[0]	in
AIB6	(empty)	in	AIB7	fs_mac_rdy	in
AIB4	spare[0]	I/O	AIB5	spare[1]	I/O
AIB2	ns_mac_rdy	out	AIB3	(empty)	out
AIB0	ns_fwd_clk	out	AIB1	ns_fwd_clkb	out

Table 44. Example Bump Table (AIB Base, 20 RX)

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB26	(empty)	in	AIB27	fs_mac_rdy	in
AIB24	spare[0]	I/O	AIB25	spare[1]	I/O
AIB22	ns_mac_rdy	out	AIB23	(empty)	out
AIB20	TX[0]	out	AIB21	TX[1]	out
AIB18	TX[2]	out	AIB19	TX[3]	out
AIB16	TX[4]	out	AIB17	TX[5]	out
AIB14	TX[6]	out	AIB15	TX[7]	out
AIB12	TX[8]	out	AIB13	TX[9]	out
AIB10	ns_fwd_clk	out	AIB11	ns_fwd_clkb	out
AIB8	TX[10]	out	AIB9	TX[11]	out
AIB6	TX[12]	out	AIB7	TX[13]	out
AIB4	TX[14]	out	AIB5	TX[15]	out
AIB2	TX[16]	out	AIB3	TX[17]	out
AIB0	TX[18]	out	AIB1	TX[19]	out

Table 45. Example Bump Table (AIB Base, 20 TX)

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB60	RX[19]	in	AIB61	RX[18]	in
AIB58	RX[17]	in	AIB59	RX[16]	in
AIB56	RX[15]	in	AIB57	RX[14]	in
AIB54	RX[13]	in	AIB55	RX[12]	in
AIB52	RX[11]	in	AIB53	RX[10]	in
AIB50	fs_fwd_clkb	in	AIB51	fs_fwd_clk	in
AIB48	RX[9]	in	AIB49	RX[8]	in
AIB46	RX[7]	in	AIB47	RX[6]	in
AIB44	RX[5]	in	AIB45	RX[4]	in
AIB42	RX[3]	in	AIB43	RX[2]	in
AIB40	RX[1]	in	AIB41	RX[0]	in
AIB38	fs_rcv_clkb	in	AIB39	fs_rcv_clk	in
AIB36	fs_sr_clkb	in	AIB37	fs_sr_clk	in
AIB34	fs_sr_load	in	AIB35	fs_sr_data	in
AIB32	fs_adapter_rstn	in	AIB33	fs_mac_rdy	in
AIB30	spare[0]	I/O	AIB31	spare[1]	I/O
AIB28	ns_mac_rdy	out	AIB29	ns_adapter_rstn	out
AIB26	ns_sr_data	out	AIB27	ns_sr_load	out
AIB24	ns_sr_clk	out	AIB25	ns_sr_clkb	out
AIB22	ns_rcv_clk	out	AIB23	ns_rcv_clkb	out
AIB20	TX[0]	out	AIB21	TX[1]	out
AIB18	TX[2]	out	AIB19	TX[3]	out
AIB16	TX[4]	out	AIB17	TX[5]	out
AIB14	TX[6]	out	AIB15	TX[7]	out
AIB12	TX[8]	out	AIB13	TX[9]	out
AIB10	ns_fwd_clk	out	AIB11	ns_fwd_clkb	out
AIB8	TX[10]	out	AIB9	TX[11]	out
AIB6	TX[12]	out	AIB7	TX[13]	out
AIB4	TX[14]	out	AIB5	TX[15]	out
AIB2	TX[16]	out	AIB3	TX[17]	out
AIB0	TX[18]	out	AIB1	TX[19]	out

Table 46. Example Bump Table (AIB Plus, 40 I/Os, Balanced)

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB38	fs_rcv_clkb	in	AIB39	fs_rcv_clk	in
AIB36	fs_sr_clkb	in	AIB37	fs_sr_clk	in
AIB34	fs_sr_load	in	AIB35	fs_sr_data	in
AIB32	fs_adapter_rstn	in	AIB33	fs_mac_rdy	in
AIB30	spare[0]	I/O	AIB31	spare[1]	I/O
AIB28	ns_mac_rdy	out	AIB29	ns_adapter_rstn	out
AIB26	ns_sr_data	out	AIB27	ns_sr_load	out
AIB24	ns_sr_clk	out	AIB25	ns_sr_clkb	out
AIB22	(empty)	out	AIB23	(empty)	out
AIB20	TX[0]	out	AIB21	TX[1]	out
AIB18	TX[2]	out	AIB19	TX[3]	out
AIB16	TX[4]	out	AIB17	TX[5]	out
AIB14	TX[6]	out	AIB15	TX[7]	out
AIB12	TX[8]	out	AIB13	TX[9]	out
AIB10	ns_fwd_clk	out	AIB11	ns_fwd_clkb	out
AIB8	TX[10]	out	AIB9	TX[11]	out
AIB6	TX[12]	out	AIB7	TX[13]	out
AIB4	TX[14]	out	AIB5	TX[15]	out
AIB2	TX[16]	out	AIB3	TX[17]	out
AIB0	TX[18]	out	AIB1	TX[19]	out

Table 47. Example Bump Table (AIB Plus, 20 TX)

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB38	RX[19]	in	AIB39	RX[18]	in
AIB36	RX[17]	in	AIB37	RX[16]	in
AIB34	RX[15]	in	AIB35	RX[14]	in
AIB32	RX[13]	in	AIB33	RX[12]	in
AIB30	RX[11]	in	AIB31	RX[10]	in
AIB28	fs_fwd_clkb	in	AIB29	fs_fwd_clk	in
AIB26	RX[9]	in	AIB27	RX[8]	in
AIB24	RX[7]	in	AIB25	RX[6]	in
AIB22	RX[5]	in	AIB23	RX[4]	in
AIB20	RX[3]	in	AIB21	RX[2]	in
AIB18	RX[1]	in	AIB19	RX[0]	in
AIB16	(empty)	in	AIB17	(empty)	in
AIB14	fs_sr_clkb	in	AIB15	fs_sr_clk	in
AIB12	fs_sr_load	in	AIB13	fs_sr_data	in
AIB10	fs_adapter_rstn	in	AIB11	fs_mac_rdy	in
AIB8	spare[0]	I/O	AIB9	spare[1]	I/O
AIB6	ns_mac_rdy	out	AIB7	ns_adapter_rstn	out
AIB4	ns_sr_data	out	AIB5	ns_sr_load	out
AIB2	ns_sr_clk	out	AIB3	ns_sr_clkb	out
AIB0	ns_rcv_clk	out	AIB1	ns_rcv_clkb	out

Table 48. Example Bump Table
(AIB Plus, 20 RX)

6.3.4 Example Bump Maps

The following example bump maps represent the bump tables above as assigned to bumps. It includes the AUX block, which shall be placed next to Channel 0.

Power microbumps are subject to chiplet requirements as established for different silicon processes, data rates, and configurations. Chiplets may implement a power-distribution network using C4 (core) bumps or microbumps to meet the power requirements. Power microbumps may be inserted between rows of I/O microbumps, in increments of six microbumps, in order to maintain signal order.

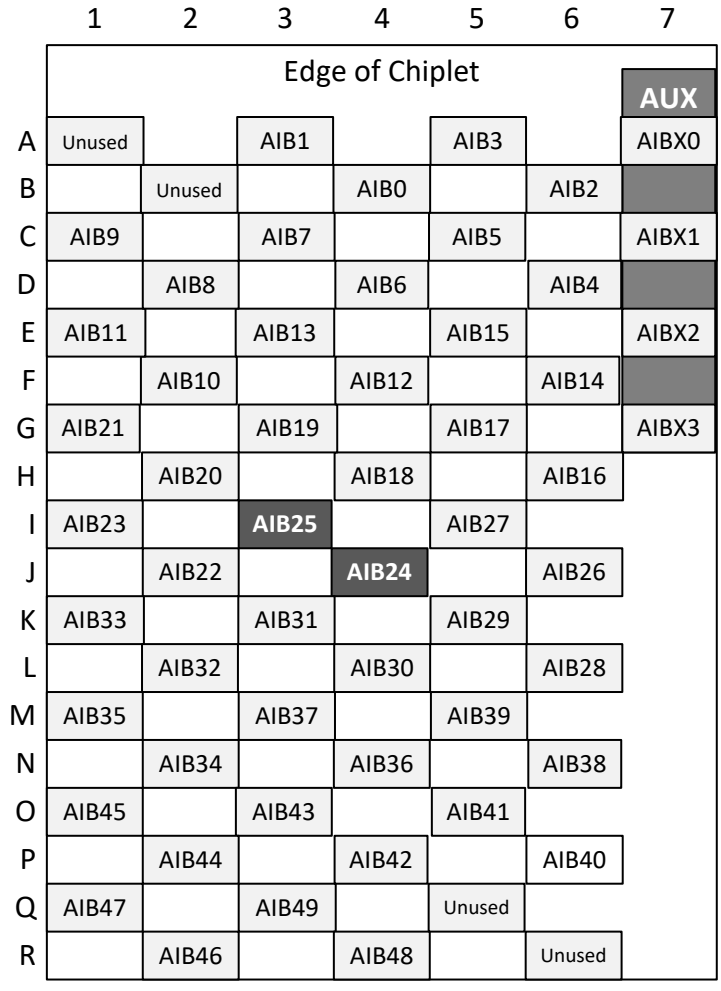


Figure 55. Low-Density Bump Map

(AIB Base, 40 I/Os, Balanced)

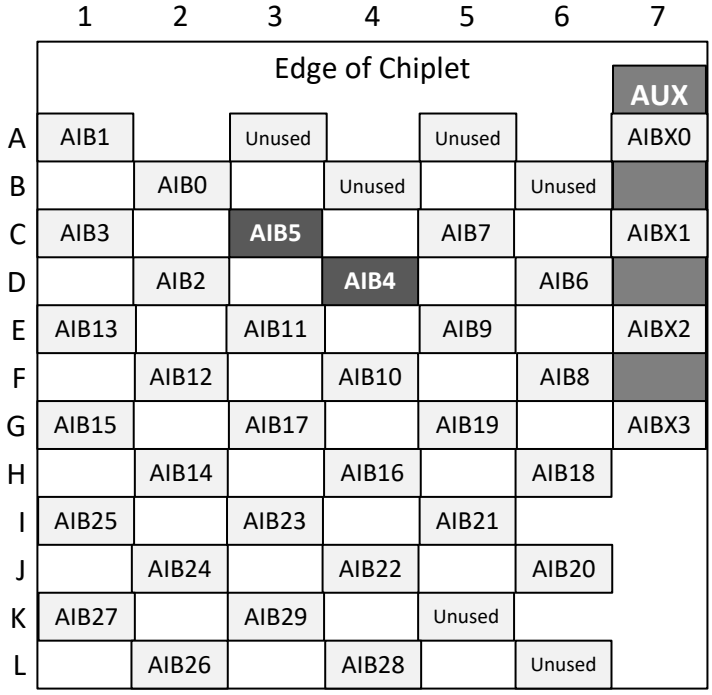


Figure 56. Low-Density Bump Map (AIB Base, 20 TX)

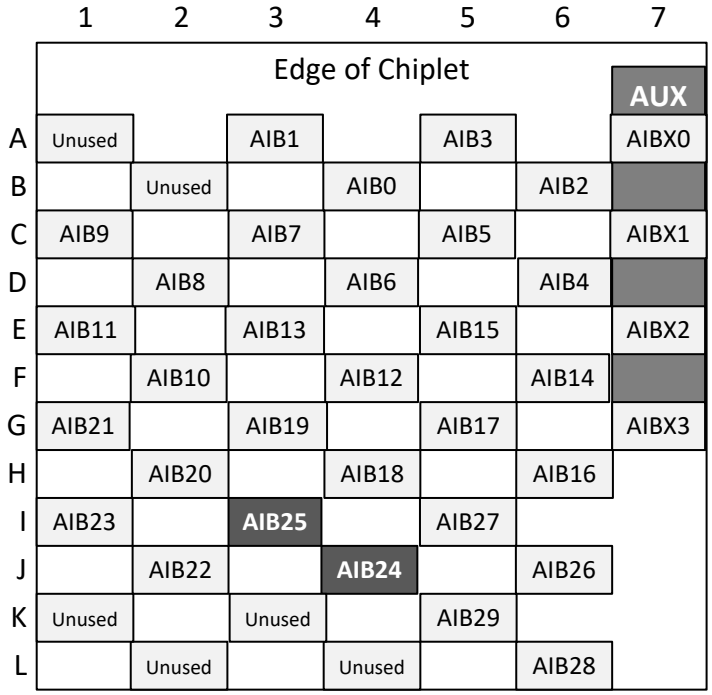


Figure 57. Low-Density Bump Map

(AIB Base, 20 RX)

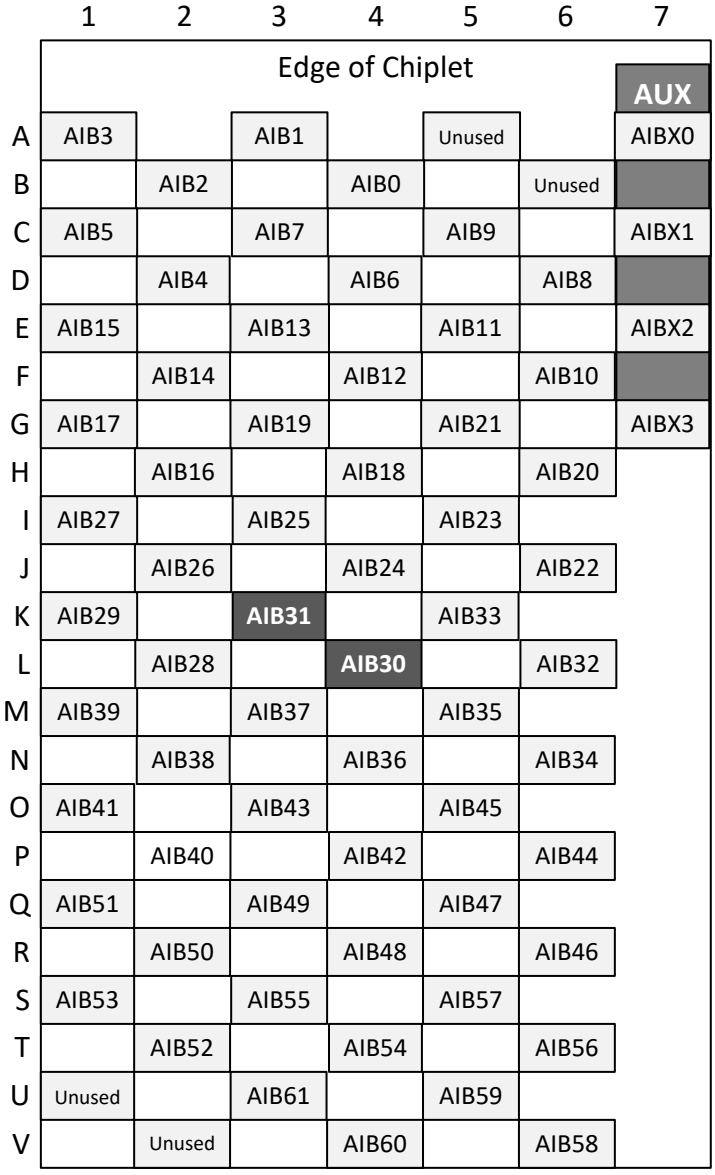


Figure 58. Low-Density Bump Map (AIB Plus, 40 I/Os, Balanced)

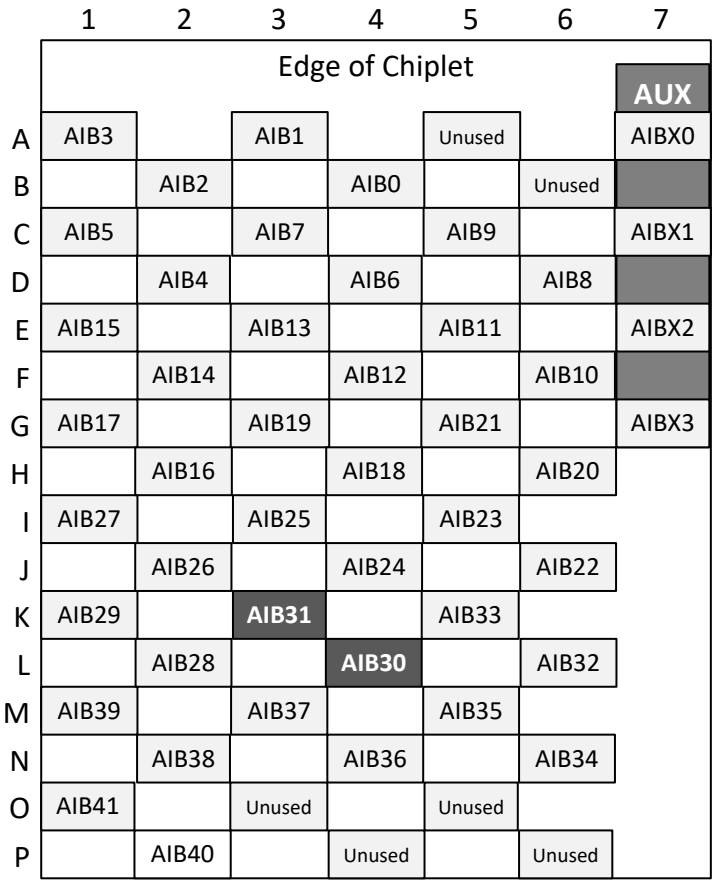


Figure 59. Low-Density Bump Map (AIB Plus, 20 TX)

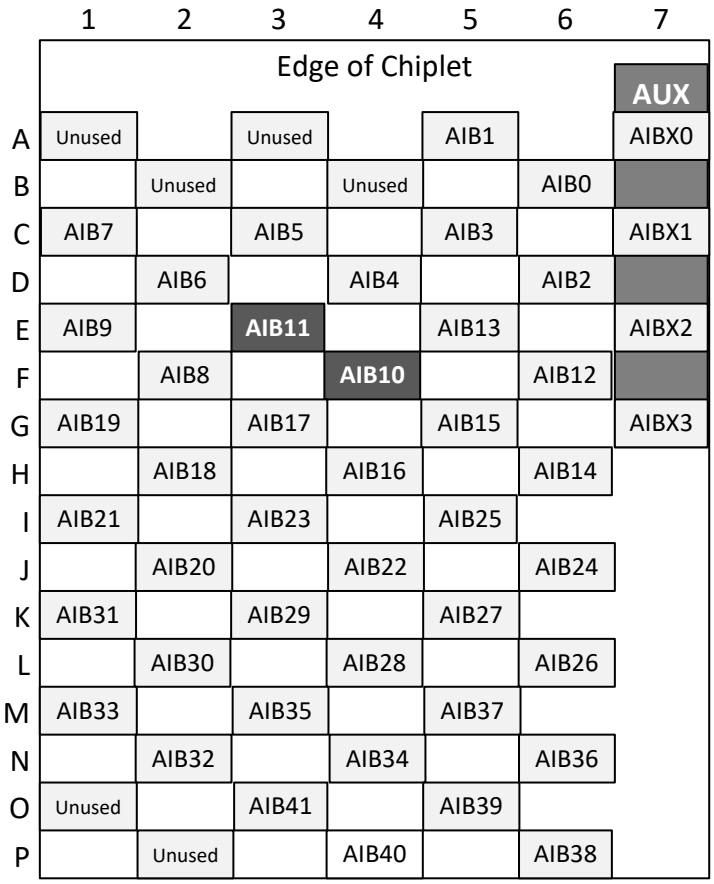


Figure 60. Low-Density Bump Map (AIB Plus, 20 RX)

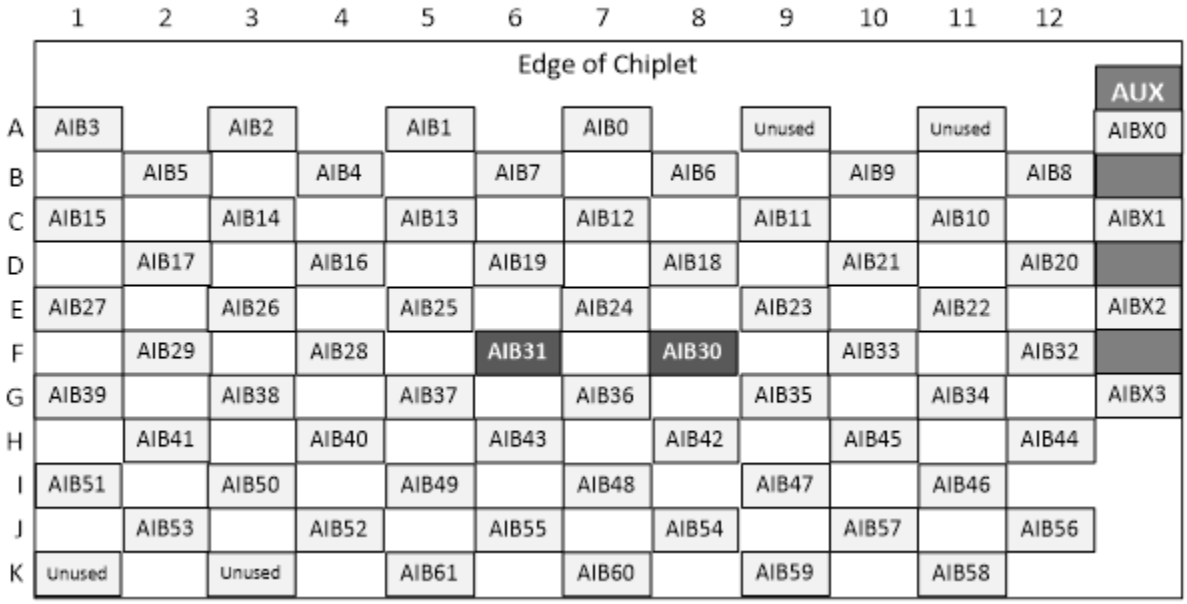


Figure 61 Medium-Density Bump Map (AIB Plus, 40 I/Os, Balanced)

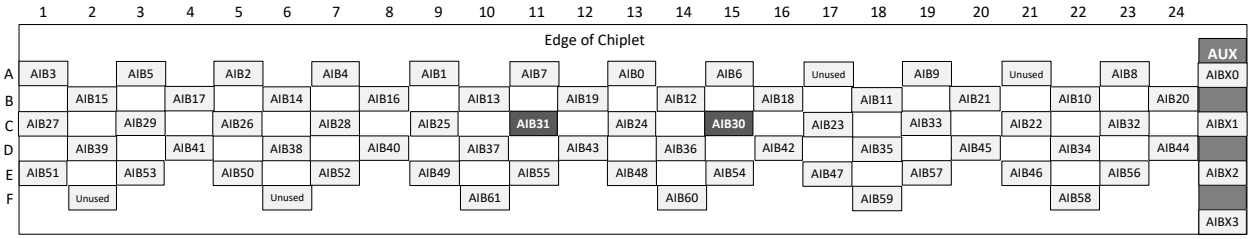


Figure 62 High-Density Bump Map (AIB Plus, 40 I/Os, Balanced)

6.4 Stacking Channels into a Column

Channel 0 shall abut the AUX block bumps. The orientation shall depend on whether the interface is placed on the East/West sides or the North/South sides of the chiplet.

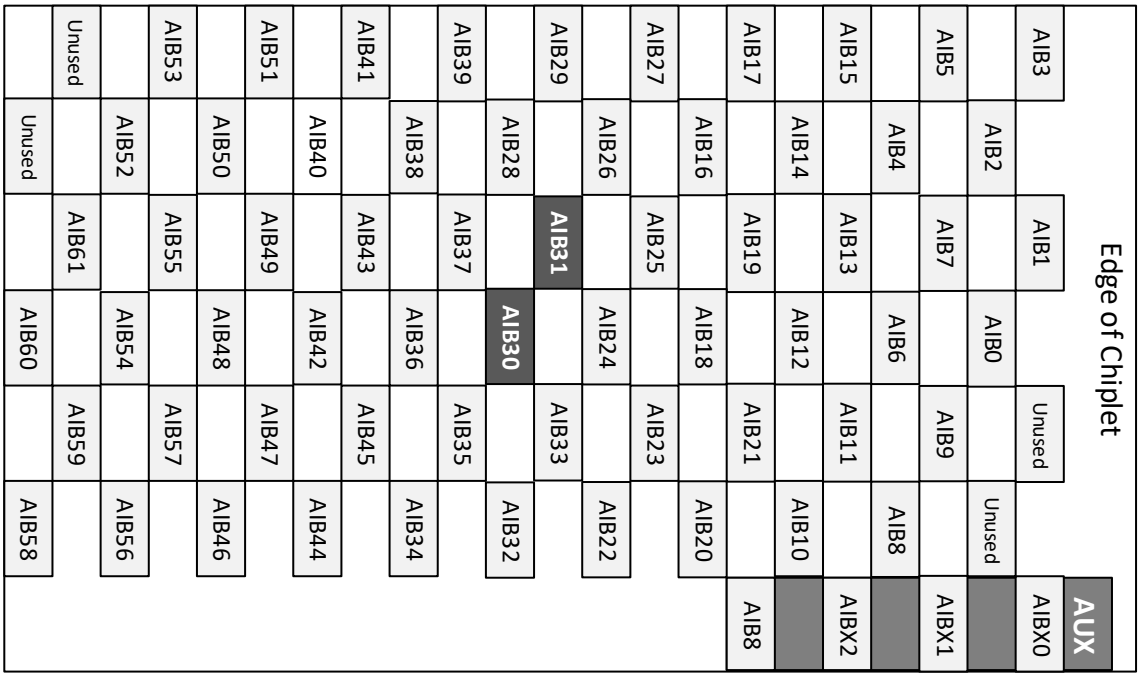


Figure 63. Low-Density Channel 0 and AUX: East Side

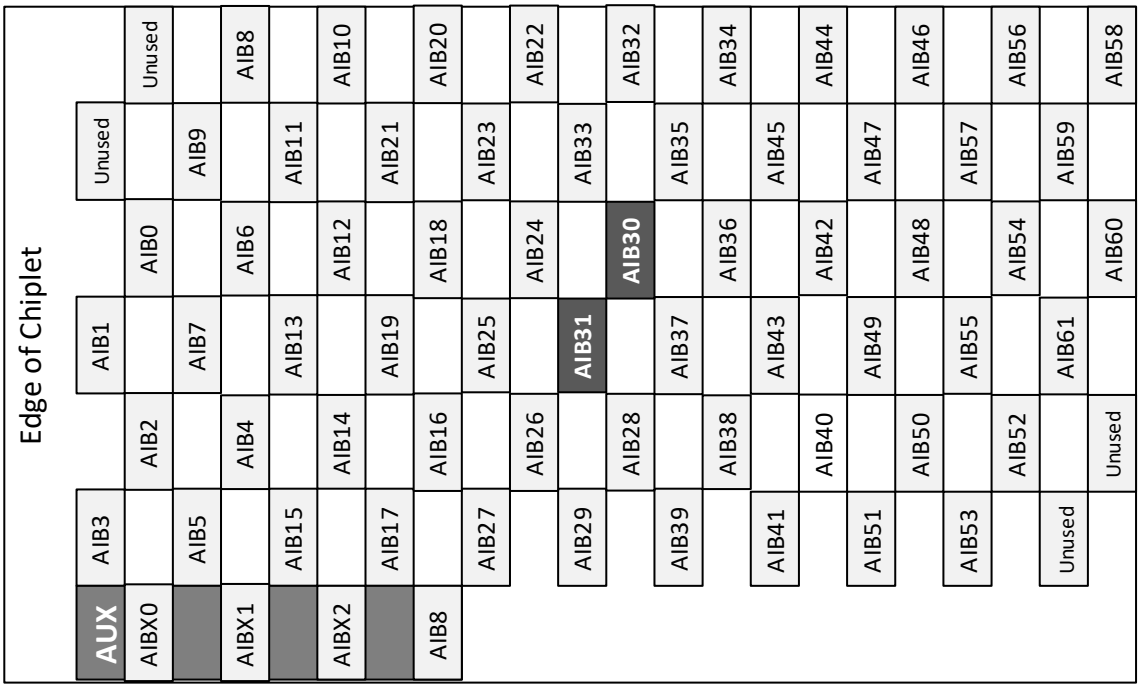


Figure 64. Low-Density Channel 0 and AUX: West Side

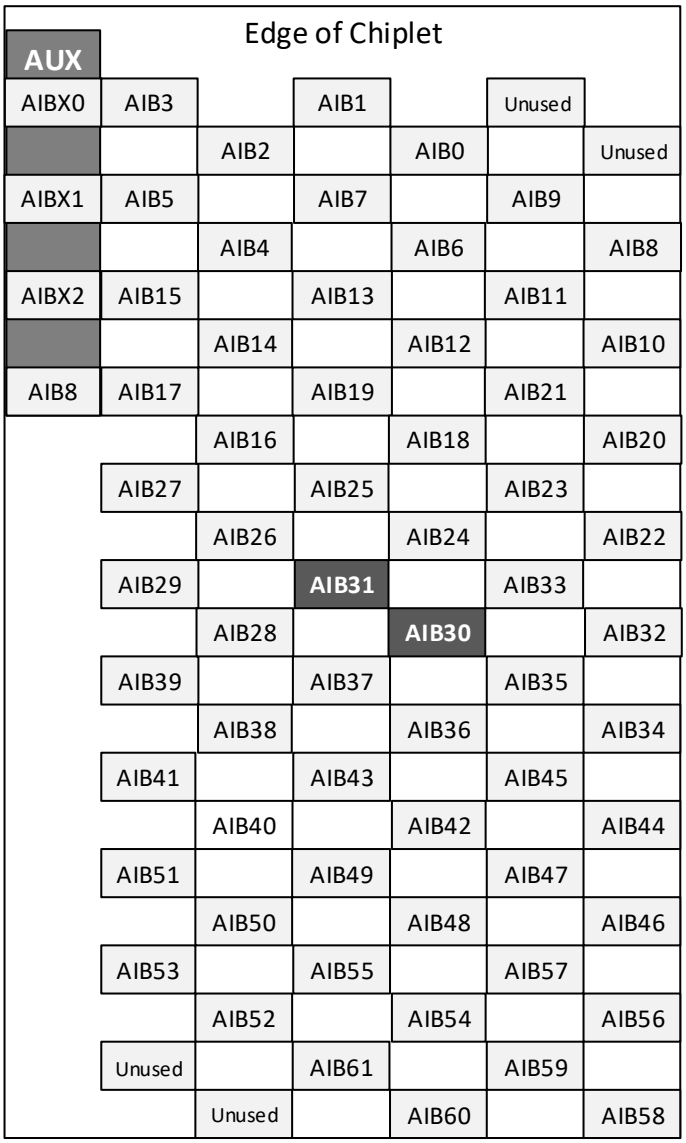


Figure 65. Low-Density Channel 0 and AUX: North Side

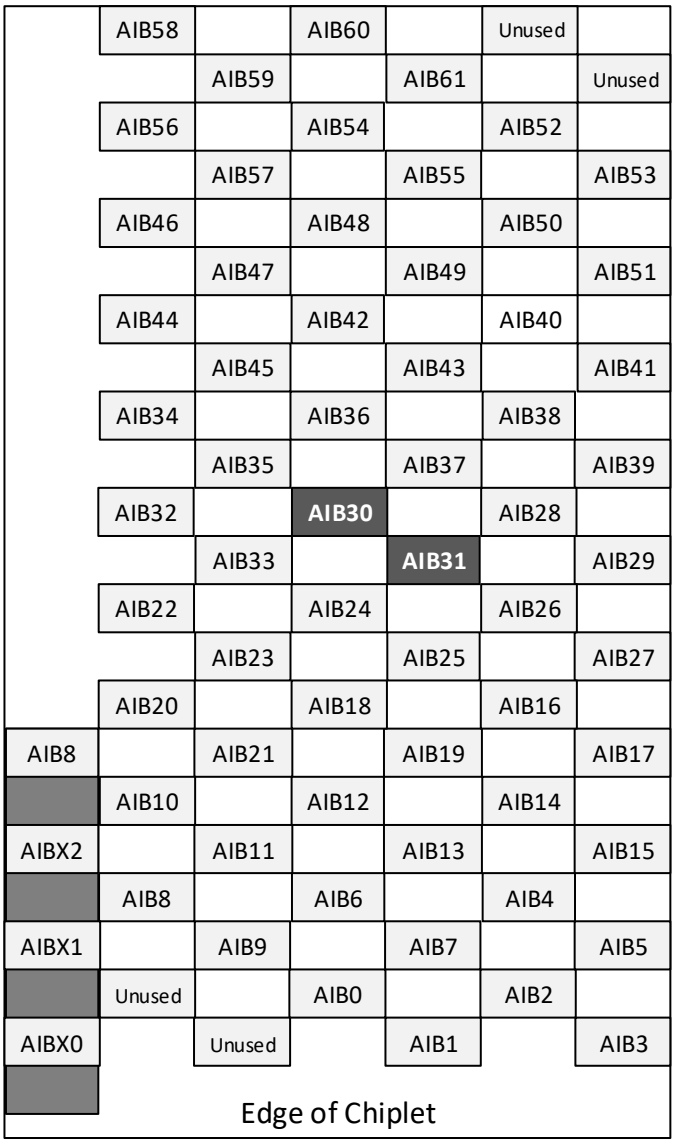


Figure 66. Low-Density Channel 0 and AUX: South Side

The remaining channels shall be stacked by abutting each channel against the prior channel. The order shall depend on whether the interface is placed on the East/West sides or the North/South sides of the chiplet.

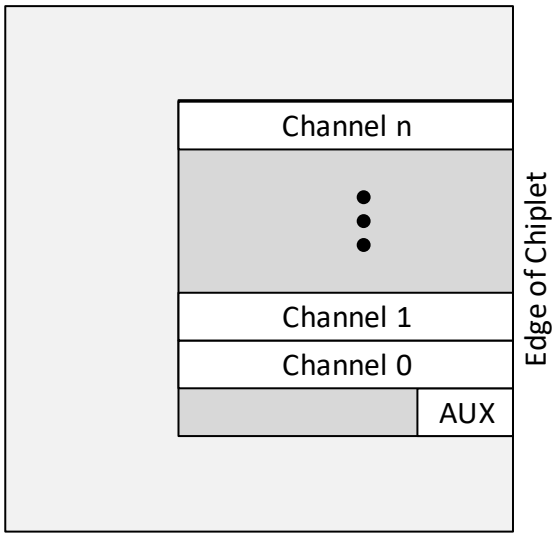


Figure 67. Channel Stacking: East Side

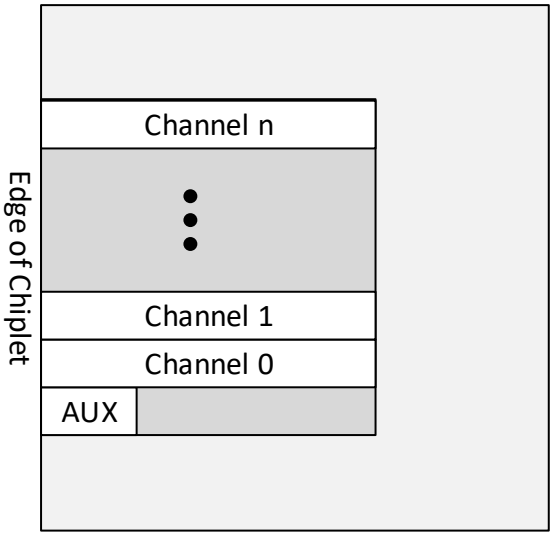


Figure 68. Channel Stacking: West Side

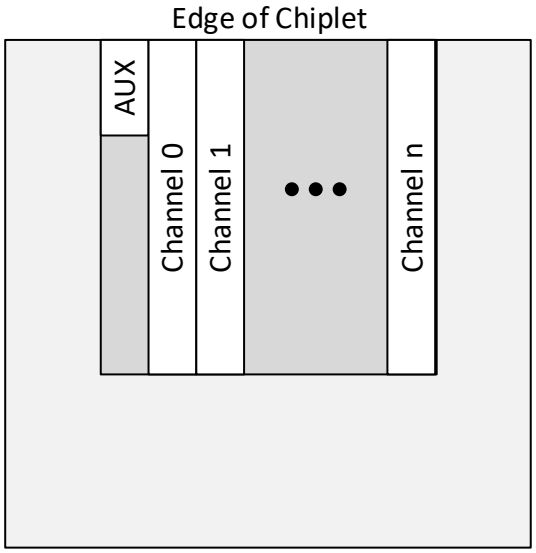


Figure 69. Channel Stacking: North Side

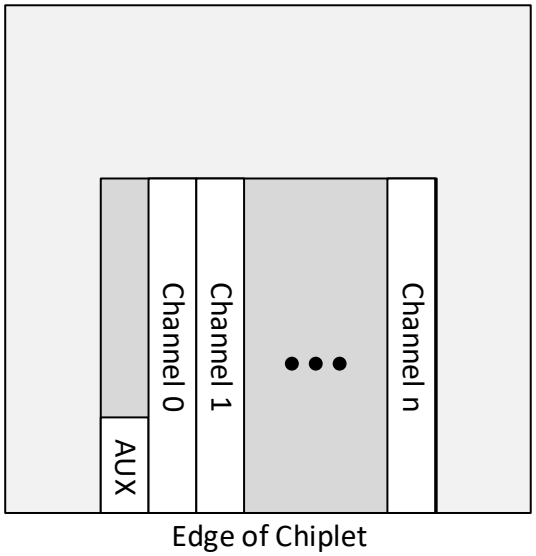


Figure 70. Channel Stacking: South Side

6.5 Channel-Number Semantics

The AIB Specification assumes no application-level semantics associated with channel numbers within a column. If an application implements channel-number semantics that impact how one interface connects to another, then it is the responsibility of the designer to resolve any channel-number conflicts.

6.6 Alternate (Master) Bump Map

Alternate 40-data-signal bump table could be used for devices intending to connect to an existing AIB interface devices. The table and associated bump map are shown in the appendix (Section 7.1).

If alternate implementation is to interface with an AIB Plus implementation, redundancy shall be disabled. In addition, one row of microbumps shall be inserted after each set of six channels when stacking channels into a column.

An AIB interface designed to connect to this bump map shall be a master.

7 Appendices

7.1 Alternate (Master) Bump Map

7.1.1 Alternate (Master) Bump Table

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB61	unused_AIB61	tri	AIB50	unused_AIB50	tri
AIB72	unused_AIB72	tri	AIB73	unused_AIB73	tri
AIB75	unused_AIB75	tri	AIB74	unused_AIB74	tri
AIB91	unused_AIB91	tri	AIB90	unused_AIB90	tri
AIB95	ns_sr_data	out	AIB94	ns_sr_load	out
AIB85	ns_sr_clk	out	AIB84	ns_sr_clkb	out
AIB76	unused_AIB76	tri	AIB77	unused_AIB77	tri
AIB58	unused_AIB58	tri	AIB63	unused_AIB63	tri
AIB48	unused_AIB48	tri	AIB55	unused_AIB55	tri
AIB62	unused_AIB62	tri	AIB60	unused_AIB60	tri
AIB53	unused_AIB53	tri	AIB54	unused_AIB54	tri
AIB49	ns_mac_rdy	out	AIB56	ns_adapt_rstn	out
AIB51	unused_AIB51	tri	AIB52	unused_AIB52	tri
AIB57	fs_rcv_clk	tri	AIB59	fs_rcv_clkb	tri
AIB64	unused_AIB64	tri	AIB65	fs_adapt_rstn	in
AIB80	unused_AIB80	tri	AIB81	unused_AIB81	tri
AIB78	unused_AIB78	tri	AIB79	unused_AIB79	tri
AIB87	ns_rcv_clk	out	AIB86	ns_rcv_clkb	out
AIB83	fs_sr_clk	in	AIB82	fs_sr_clkb	in
AIB89	unused_AIB89	tri	AIB88	unused_AIB88	tri
AIB93	fs_sr_data	in	AIB92	fs_sr_load	in
AIB71	unused_AIB71	tri	AIB70	unused_AIB70	tri
AIB68	unused_AIB68	tri	AIB69	unused_AIB69	tri
AIB66	unused_AIB66	tri	AIB67	unused_AIB67	tri
AIB20	RX[0]	in	AIB21	RX[1]	in
AIB22	RX[2]	in	AIB23	RX[3]	in
AIB24	RX[4]	in	AIB25	RX[5]	in
AIB26	RX[6]	in	AIB27	RX[7]	in
AIB28	RX[8]	in	AIB29	RX[9]	in
AIB43	fs_fwd_clk	in	AIB42	fs_fwd_clkb	in
AIB30	RX[10]	in	AIB31	RX[11]	in
AIB32	RX[12]	in	AIB33	RX[13]	in

Bump ID	Bump Name	IO	Bump ID	Bump Name	IO
AIB34	RX[14]	in	AIB35	RX[15]	in
AIB36	RX[16]	in	AIB37	RX[17]	in
AIB38	RX[18]	in	AIB39	RX[19]	in
AIB44	fs_mac_rdy	in	AIB45	unused_AIB45	tri
AIB18	TX[18]	out	AIB19	TX[19]	out
AIB16	TX[16]	out	AIB17	TX[17]	out
AIB14	TX[14]	out	AIB15	TX[15]	out
AIB12	TX[12]	out	AIB13	TX[13]	out
AIB10	TX[10]	out	AIB11	TX[11]	out
AIB41	ns_fwd_clk	out	AIB40	ns_fwd_clkb	out
AIB8	TX[8]	out	AIB9	TX[9]	out
AIB6	TX[6]	out	AIB7	TX[7]	out
AIB4	TX[4]	out	AIB5	TX[5]	out
AIB2	TX[2]	out	AIB3	TX[3]	out
AIB0	TX[0]	out	AIB1	TX[1]	out
AIB46	unused_AIB46	tri	AIB47	unused_AIB47	tri

Table 49. Alternate Channel Bump Table

7.1.2 Alternate (Master) Channel Bump Map

Figure 71 shows full alternate bump map with allocation for power and ground bump. VCCIO bumps are dedicated for last stage of AIB I/O buffers. VCCD bumps are dedicated for digital circuits including AIB adapter. If the AIB chiplet is connected to Intel FPGA using Intel EMIB silicon interposer, VCCIO will be powered by FPGA and varies between 0.75V to 0.97V. The maximum VCCIO current consumption should not exceed 40mA per AIB channel.

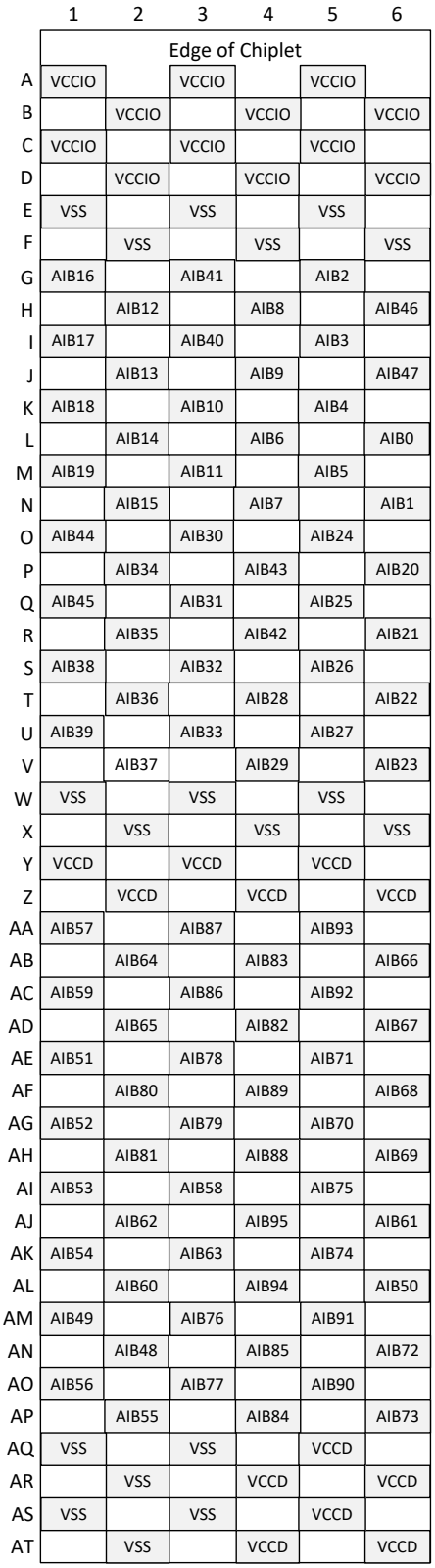


Figure 71. Alternate Bump Map

7.2 Sideband-Control-Signal Shift Register Mapping (AIB Plus only)

Bit Order	Side Band Control Signals from Master to Slave	Bit Width	Default Value	Descriptions
[80]	ms_osc_transfer_en	1	1	MS output to SL to indicate MS OSC transfer has been enabled.
[79]	Reserved	1	1	Reserved
[78]	ms_tx_transfer_en	1	1	TX output to RX to indicate that TX OSC transfer has been enabled.
[77]	Reserved	1	1	Reserved
[76]	Reserved	1	1	Reserved
[75]	ms_rx_transfer_en	1	1	RX output to TX to indicate that RX is ready for data transfer.
[74]	ms_rx_dll_lock	1	1	RX output to TX to indicate that RX DLL achieves lock.
[73:71]	Reserved	1	1	Reserved
[70:69]	Reserved	1	1	Reserved
[68]	ms_tx_dcc_cal_done	1	1	TX output to RX to indicate that DCC calibration is done
[67]	Reserved	1	0	Reserved
[66]	Reserved	1	1	Reserved
[65:12]	User defined	1	0	For application use
[11]	User defined	1	0	For application use
[10]	User defined	1	0	For application use
[9]	User defined	1	0	For application use
[8]	User defined	1	0	For application use
[7]	Reserved	1	1	Reserved
[6]	Reserved	1	0	Reserved
[5]	Reserved	1	1	Reserved
[4:0]	User defined	1	0	For application use

Table 50. Master Sideband-Control Signals

Bit Order	Side Band Control Signals from Slave to Master	Bit Width	Default Value	Descriptions
[72]	sl_osc_transfer_en	1	1	RX output to MS to indicates SL OSC transfer has been enabled.

Bit Order	Side Band Control Signals from Slave to Master	Bit Width	Default Value	Descriptions
[71]	Reserved	1	0	Reserved
[70]	sl_rx_transfer_en	1	1	RX output to MS to indicate SL RX is ready to receive data
[69]	sl_rx_dll_dcc_lock_req	1	1	DLL/DCC calibration request from RX to TX AIB to start full DLL/DCC calibration.
[68]	sl_rx_dll_lock	1	1	RX output to MS (adapter and PHY) to indicate SL DLL achieves lock.
[67:65]	Reserved	1	0	Reserved
[64]	sl_tx_transfer_en	1	1	TX sends to RX (adapter and PHY) that it is ready for TX data transfer.
[63]	sl_tx_dll_dcc_lock_req	1	1	PHY DLL/DCC calibration request from TX to RX AIB to start full DLL/DCC calibration.
[62]	Reserved	1	0	Reserved
[61]	Reserved	1	0	Reserved
[60]	Reserved	1	1	Reserved
[59]	Reserved	1	0	Reserved
[58]	Reserved	1	1	Reserved
[57:32]	User defined	1	0	For application use
[31]	sl_tx_dcc_cal_done	1	1	TX AIB to notify MS that DCC calibration is complete.
[30:28]	User defined	1	0	For application use
[27]	Reserved	1	0	Reserved
[26:17]	User defined	1	0	For application use
[16]	User defined	1	0	For application use
[15]	User defined	1	0	For application use
[14]	User defined	1	0	For application use
[13]	User defined	1	0	For application use
[12:0]	User defined	1	0	For application use

Table 51. Slave Sideband-Control Signal