Advanced Interface Bus (AIB) Usage Note

Version 1.2, 10/2019

David Kehlet
Research Scientist

Tim Hoang
Principal Engineer

Intel® Programmable Solutions Group

1 Introduction

This document describes requirements that a chiplet must meet to interoperate with Stratix 10, or to interoperate with chiplets built for Stratix 10. To interoperate with Stratix 10 a chiplet must implement a profile of AIB capabilities, conform to specific signal assignments and mechanical requirements, and meet additional requirements beyond the AIB specification. Similarly, to interoperate with a chiplet that was built for Stratix 10, a chiplet must implement a Stratix 10-like profile of AIB requirements. This document identifies the requirements for both cases:

1) A chiplet with an AIB interface intended to interoperate with Stratix 10; that interface is referred to here as a Stratix 10 AIB Master interface or a Master.
2) A chiplet with an AIB interface intended to interoperate with a Master interface, functioning similarly to the AIB of the Stratix 10 itself. That interface is referred to here as a Stratix 10 AIB Slave interface or a Slave.

Note that a chiplet could have both Master and Slave interfaces.

The **Advanced Interface Bus (AIB) Specification** [1] document, also known as the AIB Spec, is the source of requirement references.

1.1 Stratix 10 AIB Overview

Stratix 10 devices have 1 to 6 AIB interfaces, depending on the specific family member. You can find the number of AIBs for a member by referring to the Stratix 10 Product Tables, finding the "Total full duplex transceiver count" and dividing by 24. A Stratix 10 TX 2800, for example, has 144 total full duplex transceivers supported by 6 AIB interfaces.

Each Stratix 10 AIB interface has 24 data channels; each data channel has 20 Tx and 20 Rx data signals. Each data signal runs up to 2Gbps, producing an aggregate bandwidth per AIB interface of 960Gbps Tx and 960Gbps Rx (1.92Tbps total).

To source and sink data at 2Gbps per AIB data signal, the Stratix 10 FPGA user design must run at 500MHz. Your achievable system bandwidth depends on the FMax and routability of your user design inside Stratix 10, the data source and sink bandwidth of the attached Master interface chiplet, and the bandwidth capability of the AIB interface.

1.2 Stratix 10 EMIB

The Master and Slave interfaces in this document are implemented on the Stratix 10 EMIB. This establishes pinout requirements that are included in the descriptions following.

2 AIB Standard, Mode and Channels

The Master interface and Slave interface implement the AIB Plus requirements of the AIB Spec except as specifically noted in this document. Master and Slave interfaces use:

- Master or Slave specific AIB signal to bump assignments in the tables below.
- Master or Slave specific AIB bump locations in the spreadsheet of section 8.1.
- VCCIO voltage of 0.75V to 0.97V.

3 Master and Slave AIB Features

3.1 Master and Slave AIB IOs

Master and Slave AIB IOs support Asynchronous mode, SDR from 0.1Gbps to 1Gbps, and DDR from 0.2Gbps to 2Gbps. DLL and DCC per channel are strongly recommended for DDR above 800Mbps.

Redundancy is not required or recommended for Master or Slave AIB interfaces. Stratix 10 supports redundancy in a legacy (non-standard) mode.

3.2 AIB Interface Signals

A Master or Slave interface implements the AIB Spec signals in Table 1 in each AIB channel. Near side refers to the chiplet being described; far side refers to the other chiplet of the pair.

Table 1. AIB Signals in Each AIB Channel

Signal	I/O	Description
tx[19:0]	out	Synchronous data transmitted from the near side.
ns_fwd_clk/b	out	Near-side transfer clock, forwarded from the near side to the far side for capturing received data. Used by the far side to capture the near side's TX signals.
ns_fwd_div2_clk/b[1]	out	ns_fwd_clk divided by 2
ns_rcv_clk/b	out	Receive-domain clock forwarded from the near side to the far side for transmitting data from the far side. Far side uses this to produce fs_fwd_clk.
ns_rcv_div2_clk/b[1]	out	ns_rcv_clk divided by 2
rx[19:0]	in	Synchronous data received from the far side.
fs_fwd_clk/b	in	Far-side transfer clock, forwarded from the far side to the near side for capturing received data. Used by the near side to capture RX signals.
fs_rcv_clk/b[3]	in	Receive-domain clock forwarded from the far side to the near side for transmitting data to the far side. Near side uses this to produce ns_fwd_clk.
fs_rcv_div2_clk/b[1,4]	in	fs_rcv_clk divided by 2
ns_sr_data	out	Time-multiplexed sideband-control data from near side to far side.
ns_sr_clk/b	out	Forwarded serial shift register clock from near side to far side chiplet, driven by free running clock.
ns_sr_load	out	Sideband control load signal from near side to far side.
fs_sr_data	in	Time-multiplexed sideband-control data from far side to near side.
fs_sr_clk/b	in	Forwarded serial shift register clock from far side to near side chiplet, driven by free running clock.
fs_sr_load	in	Sideband control load control signal from far side to near side.
fs_mac_rdy	in	Ready signal from far side to near side.
ns_mac_rdy	out	Ready signal from near side to far side.
fs_adapter_rstn	in	Asynchronous adapter reset signal from far side to near side.
ns_adapter_rstn[2]	out	Asynchronous adapter reset signal from near side to far side.

1. These clocks are not in the AIB Spec and are not required, however an alternate scheme of providing half rate clocks may be required if these clocks are not used. See section 3.2.1. The DCD specifications of these clocks are the same as their source clocks.
2. See descriptions of ns_adapter_rstn in Table 3 and Table 5, particularly the handling of this output by Open Source versions.

3. The AIB spec allows for fs_rcv_clk to be sent by a Slave interface to the Master, however, this use is not recommended.
4. fs_rcv_div2_clk is used by Stratix 10's Slave AIB interface, from a Master's ns_rcv_div2_clk. Use of fs_rcv_div2_clk is not recommended.
5. See Table 5 for the configuration of all other S10 Chiplet IO bumps not listed in Table 1.

3.2.1 Half Rate (Divided by 2) Clocks

The Stratix 10 core often needs clocks to be 500MHz or less, and the AIB spec's transfer and receive clocks can easily exceed 500MHz. Stratix 10 has the capability to use a half rate clock from the same reference (0 PPM difference) as the related transfer or receive clock.

Stratix 10 can be used to generate the half rate clock from the common reference using a Stratix 10 internal PLL. That common reference may enter Stratix 10 through standard clock inputs. Instead, the method used by Stratix 10 is to have a S10 chiplet supply the half rate clocks through specific AIB bumps. Table 1 contains divided-by-2 clocks (see note 1) that come from the S10 chiplet to the Stratix 10 for use by the Stratix 10 core.

A Slave interface other than Stratix 10 should ignore div2 clock inputs and generate all needed div2 clocks itself.

3.2.2 Typical Clock and Data Configurations

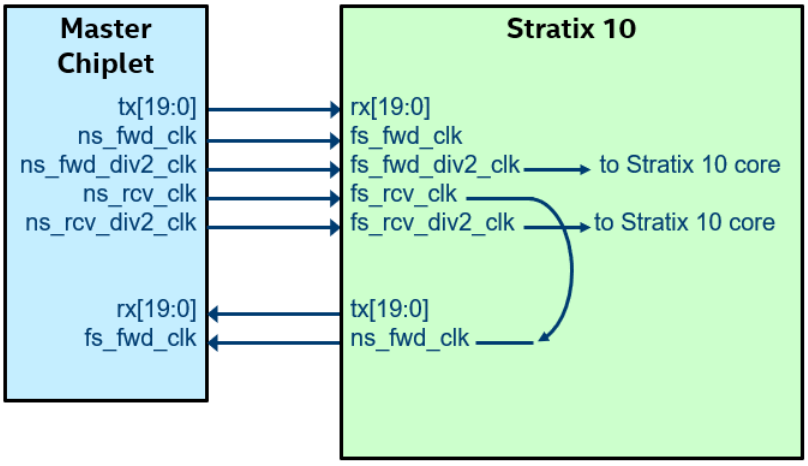


Figure 1. Typical Clock and Data Configuration with Stratix 10

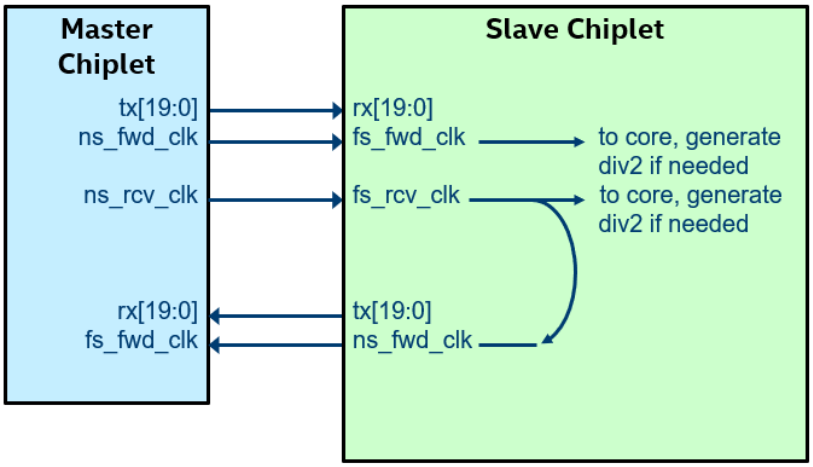


Figure 2. Typical Clock and Data Configuration, Master and Slave

See section 6 for AIB master and slave pair examples that include AIB MAC/PHY signals, in addition to the AIB signals between die.

3.3 AIB Adapter

3.3.1 Word Marking and Word Alignment

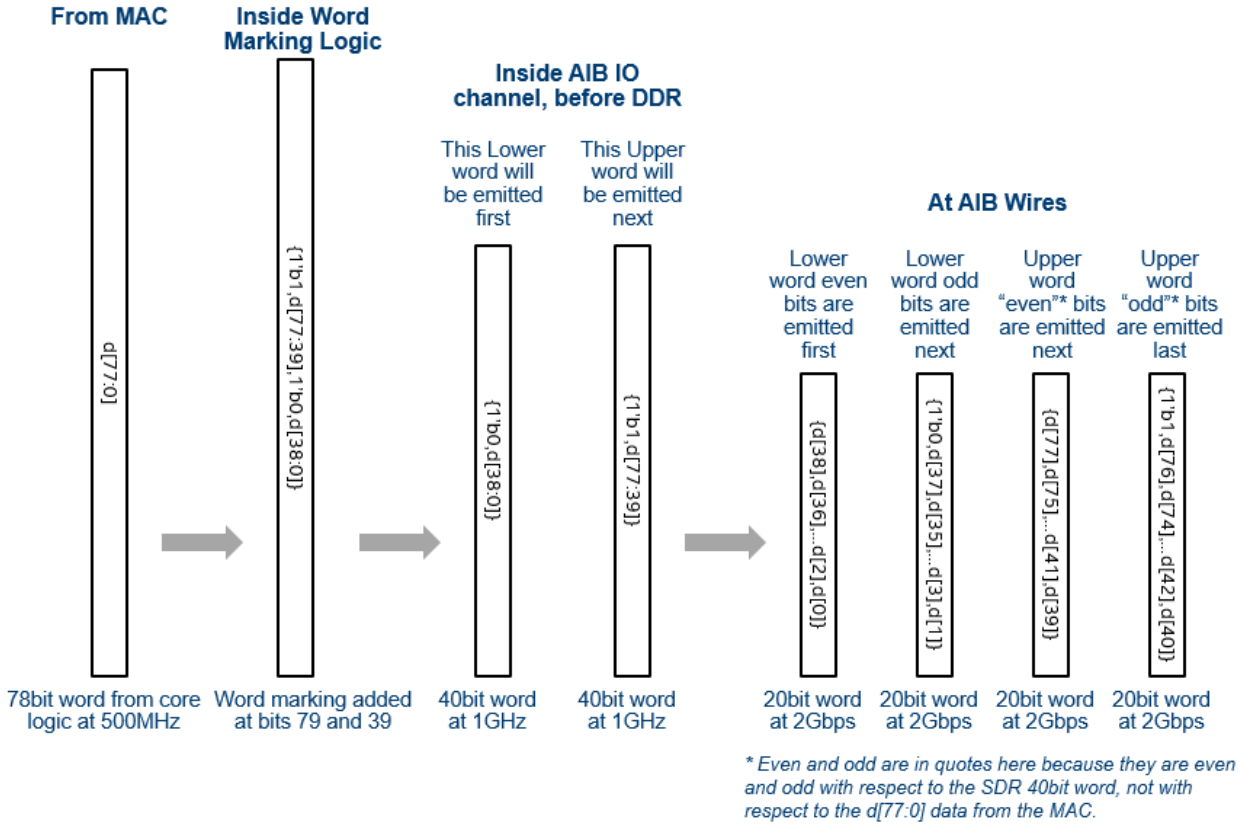


Figure 3. 2x Mode and Upper Word Marking

Stratix 10 requires word marking on Rx and outputs word marking on Tx to operate on 80bit quantities at half rate of the AIB clock (example: 2Gbps = double data rate, 1GHz = full rate, 500MHz = half rate). Word marking identifies the upper 40bits uniquely from the lower 40bits, facilitating demultiplexing into an 80bit half rate word. In an 80bit half rate data word, bits 79 and 39 are used for word marking. Figure 3 shows how bits 79 and 39 are marked and how the 78bit user data quantity from the MAC to AIB is sent over the AIB's 20bit wide tx[19:0] signals. Figure 3 and Figure 4 use the AIB Open Source's convention for bit numbering as d[77:0] at the MAC/PHY interface.

DDR to SDR conversion knows which 20bit word is even or odd by the edge of the clock (even is valid before the rising edge, odd valid before the falling edge). By examining the word marking bits, the receiving chiplet can reconstruct the 80bit word from the '0' marked lower 40bit word and the '1' marked upper 40bit word. This is shown in Figure 4.

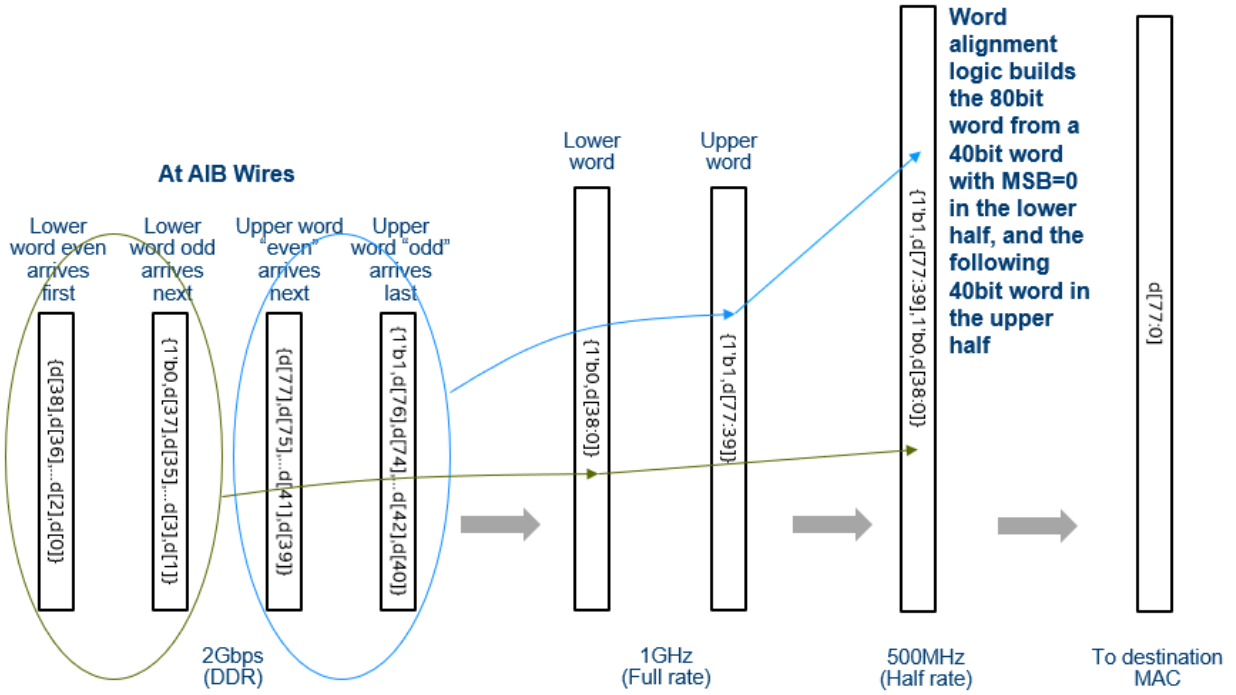


Figure 4. Reassembling Data Word in 2x Mode

3.3.2 Phase Comp FIFO, 2x Mode and Register Mode

A chiplet design may need a phase compensation FIFO to pass data from the AIB into the chiplet's core, and another phase comp FIFO from the chiplet's core to AIB IO for data going in the chiplet to Stratix 10 direction. The AIB Open Source at https://github.com/intel/aib-phy-hardware provides a combined phase comp and 2x mode FIFO, as shown in the example in the folder how2use/sim_phasecom. The AIB Open Source in this case performs word marking and word assembly for you.

A chiplet may implement only the simple adapter retiming register as defined in the AIB Spec. The example in the folder how2use/sim_aib_top is a full rate (1GHz) register mode case. The AIB Open Source passes 40b full rate words through both Rx and Tx. If you are building a S10 Chiplet to talk to Stratix 10, and you use register mode, you need to perform word marking prior to sending a word into the AIB Open Source for Tx to Stratix 10. If your 1GHz words are halves of a 78bit quantity then you need to set the lower word bit39 to 0, and upper word bit39 to 1. Otherwise, simply alternate bit39 between 0 and 1. On Rx to your S10 Chiplet, if your 1GHz words are halves of a 78bit quantity then you need to identify the lower word marked with 0 and the following word as the upper word marked with 1. Otherwise you may ignore bit 39.

3.3.3 Channel Alignment across Multiple AIB Channels

If a chiplet sends or receives a data word that is spread over more than one AIB channel, then the sending and receiving chiplets 10 need to engage in a channel alignment procedure. Once each channel's data is resynchronized to a common clock domain, the skew between channels may cause data to arrive on different cycles of the common clock.

A useful channel alignment scheme dedicates a bit out of an 80bit word for each channel to be aligned; this bit is called the strobe bit. At the transmitter the strobe bit is set to 1 in each 80bit word across all channels to be aligned, then for the next several 80bit words the strobe bit is set to 0. The cycle repeats every N words. Logic at the receiving side can determine alignment by watching the arrival of

the words with the strobe bit set to 1 bit set in the receiver's clock domain. Skew between channels is factored out by recording the relative cycle difference between channels and selecting the correct channel word to assemble into a larger bus. Figure 5 is an example of channel alignment using a strobe bit and using FIFOs on the receiving chiplet.

The value of N in the previous paragraph must be greater than the worst skew in 80bit cycles across all channels, and greater than the depth of the alignment FIFO. Typically, N is guard banded heavily for N=16 (repeat every 16 80bit cycles) or N=32 (repeat every 32 80bit cycles).

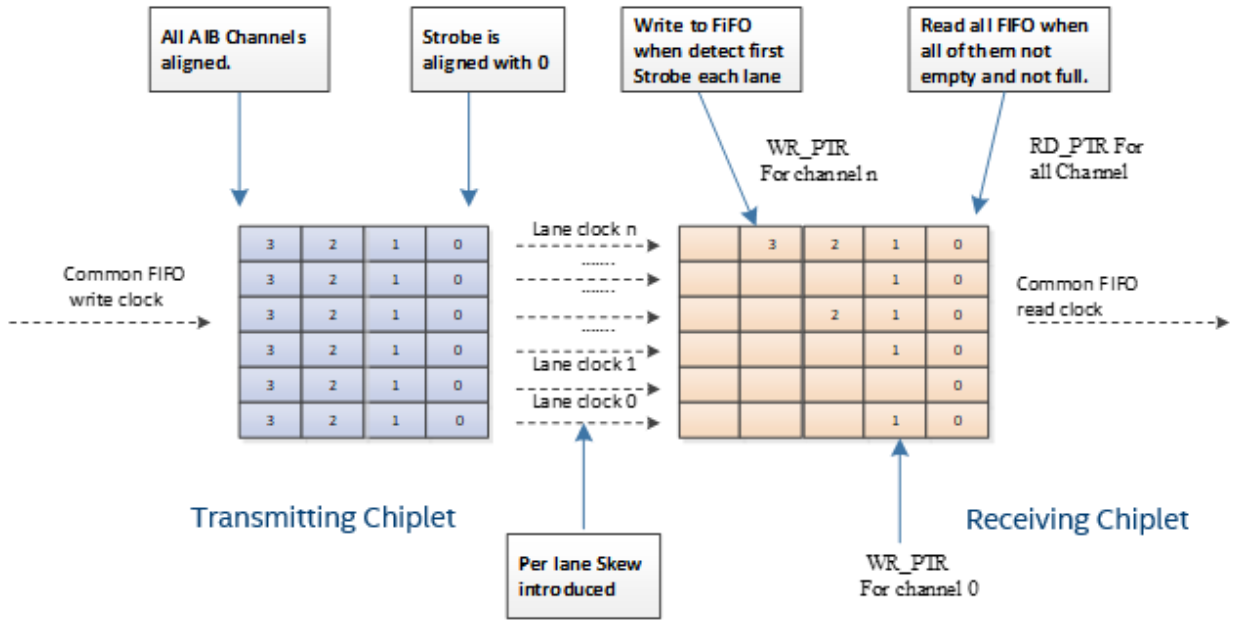


Figure 5. Channel Alignment

4 AIB Configuration and Control

You should implement the AIB configuration as a register file per channel to control input, output, DDR/SDR and other features. A fixed function may instead hardwire the feature selection, but register control provides more debugging capability.

5 AIB Open Source Signal Names

5.1 Original AIB Signal Names

The AIB Open Source at https://github.com/intel/aib-phy-hardware now contains Verilog files that translate the original Open Source signal names to the AIB Spec names. See files aib_lib/c3aibadap_wrap/rtl/aib_top_master.sv and how2use/sim_phasecom/c3aib_master.sv which have ports that match the AIB Spec. Table 2 is a mapping of the AIB spec interface names to the original AIB Open Source names for signals at the AIB die-to-die interface.

Table 2. AIB Spec Interface to AIB Open Source Signal Names

AIB Spec Interface Signal Name	Original AIB Open Source Signal Name
tx[19:0]	u_rx_data_out[19:0]
ns_fwd_clk/ns_fwd_clkb	u_rx_transfer_clk/u_rx_transfer_clk_n
ns_fwd_div2_clk/ns_fwd_div2_clkb	u_pld_pcs_rx_clk_out/u_pld_pcs_rx_clk_out_n
ns_rcv_clk/ns_rcv_clkb	u_pma_aib_tx_clk/u_pma_aib_tx_clk_n
ns_rcv_div2_clk/ns_rcv_div2_clkb	u_pld_pcs_tx_clk_out/u_pld_pcs_tx_clk_out_n

AIB Spec Interface Signal Name	Original AIB Open Source Signal Name
rx[19:0]	u_tx_data_in[19:0]
fs_fwd_clk/fs_fwd_clkb	u_tx_transfer_clk/u_tx_transfer_clk_n
ns_sr_data	u_ssr_data_out
ns_sr_clk/ns_sr_clkb	u_sr_clk_out/u_sr_clk_n_out
ns_sr_load	u_ssr_load_out
fs_sr_data	u_ssr_data_in
fs_sr_clk/fs_sr_clkb	u_sr_clk_in/u_sr_clk_n_in
fs_sr_load	u_ssr_load_in
fs_mac_rdy	u_pld_pma_rxpma_rstb
ns_mac_rdy	u_pld_pma_clkdiv_rx_user
fs_adapter_rstn	u_adapter_rx_pld_rst_n

5.2 AIB v1 and v2 Signal Names

Some signals have been added, deleted or changed with a new open source version "v2." In Table 3 any such signals are designated as v1 or v2 signals.

5.3 AIB Open Source Per Channel MAC/PHY Signal Names

Table 3 lists the AIB MAC/PHY signals that exist for each channel according to the AIB Spec and the Open Source code.

Table 3. AIB Open Source Per Channel MAC/PHY Signals

AIB Spec MAC/PHY Signals as implemented in Open Source .sv files	In (from MAC) Out (to MAC)	Signal Reference	Description
i_osc_clk	In	n/a	Used with Master only. Free running clock (see AIB Spec). In Open Source single channel c3_aib_master.sv i_osc_clk comes from the MAC (instantiating module).
data_in [39:0] (v1) [77:0] (v2)	in	m_ns_fwd_clk (1), m_wr_clk (2)	Data to the other chiplet. For Open Source v1, this is data for register mode only. For Open Source v2, this is data for register, FIFO 1x and FIFO 2x modes. (1) Register mode reference clock for both v1 and v2 (2) Open Source v2 FIFO 1x or FIFO 2x mode reference clock
m_ns_fwd_clk	in	n/a	Transfer clock to the other chiplet
m_ns_fwd_div2_clk (additional to the AIB Spec)	in	n/a	Used with Master only. Transfer clock to the other chiplet div2
m_wr_clk	in	n/a	Open Source v2 only. data_in reference clock for FIFO 1x or FIFO 2x modes. Must be 0 PPM to m_ns_fwd_clk.
data_out [39:0] (v1) [77:0] or [31:0] (v2)	out	m_fs_fwd_clk (1), m_rd_clk (2)	Data from the other chiplet. For Open Source v1, this is data for register mode only. For Open Source v2, this is data for register, FIFO 1x and FIFO 2x modes. (1) Register mode reference clock for both v1 and v2 (2) Open Source v2 FIFO 1x or FIFO 2x mode reference clock
m_fs_fwd_clk	out	n/a	Transfer clock from the other chiplet
m_fs_fwd_div2_clk (additional to the AIB Spec)	out	n/a	Transfer clock from the other chiplet div2. Derived inside the near side AIB.
m_rd_clk	in	n/a	Open Source v2 only. data_out reference clock for FIFO 1x or FIFO 2x modes. Must be 0 PPM to m_fs_fwd_clk.
m_ns_rcv_clk	in	n/a	Used with Master only. Receive domain clock sent to the other chiplet for it to send back as fs_fwd_clk. Note there is no

AIB Spec MAC/PHY Signals as implemented in Open Source .sv files	In (from MAC) Out (to MAC)	Signal Reference	Description
			m_ns_rcv_div2_clk input. ns_rcv_div2_clk is derived inside the near side AIB.
m_fs_rcv_clk	out	n/a	Used with Slave only. Receive domain clock received from the other chiplet to be set back as m_ns_fwd_clk.
i_rx_elane_data[77:0]* (alternate data_in)	in	i_rx_elane_clk	Open Source v1 only. Data to the other chiplet, 2:1 phase comp FIFO mode. data_in is ignored. You must still provide m_ns_fwd_clk and m_ns_fwd_div2_clk.
i_rx_elane_clk* (alternate data_in clock)	in	n/a	Open Source v1 only. Usually the same input as m_ns_fwd_div2_clk; must be 0 PPM from m_ns_fwd_div2_clk.
o_tx_elane_data[77:0]* (alternate data_out)	out	i_tx_elane_clk	Open Source v1 only. Data from the other chiplet, 2:1 phase comp FIFO mode. data_out should be ignored. You must still provide m_ns_rcv_clk.
i_tx_elane_clk* (alternate data_out clock)	in	n/a	Open Source v1 only. Must be 0 PPM from m_fs_fwd_div2_clk
ns_mac_rdy	in	Async	Indicates near side MAC readiness to the far side.
fs_mac_rdy	out	Async	Indicates far side MAC readiness.
ns_adapter_rstn	in	Async	Open Source v2 only. Resets near side adapter, also sent to far side.
fs_adapter_rstn	n/a	Async	Resets near side adapter, controlled by far side. fs_adapter_rstn is terminated in the adapter and is not brought to the MAC.
ms_rx_dcc_dll_lock_req ms_tx_dcc_dll_lock_req	in	Async	Used with Master only. Chiplets attached to Stratix 10 should not request calibration. If these signals are implemented, set to "1."
sl_rx_dcc_dll_lock_req sl_tx_dcc_dll_lock_req	in	Async	Used with Slave only. The Slave sets these to 1 to initiate near side and far side datapath calibration. Leave at 1 until a new calibration sequence is needed. These signals are transported to the far side through the slave to master sideband shift register. Inside the Master adapter the sl_*x_dcc_dll_lock_req signals are read from the shift register, used and terminated. No action is required of the Master MAC.
ms_tx_transfer_en ms_rx_transfer_en	out	Async	Indicates that the calibration of the Master has completed, same as the sideband shift register bit sent by the Master.
sl_tx_transfer_en sl_rx_transfer_en	out	Async	Indicates that the calibration of the Slave completed, same as the sideband shift register bit sent by the Slave.
ms_sideband[80:0]	out	Async	Master sideband shift register bits. See Table 50 in the AIB Spec. Reading user defined bits is supported in the open source code.
sl_sideband[72:0]	out	Async	Slave sideband shift register bits. See Table 51 in the AIB Spec. Reading user defined bits is supported in the open source code.
m_rxfifo_align_done	out	Async	Open Source v2 only. Used in 2:1 phase comp FIFO mode. HI means receive FIFO is aligned to incoming word marking bits.

* These signals are in the folder how2use/sim_phasecom/c3aibadapt_wrap, not in aib_top/rtl.

5.4 AIB Open Source Application/PHY Signal Names

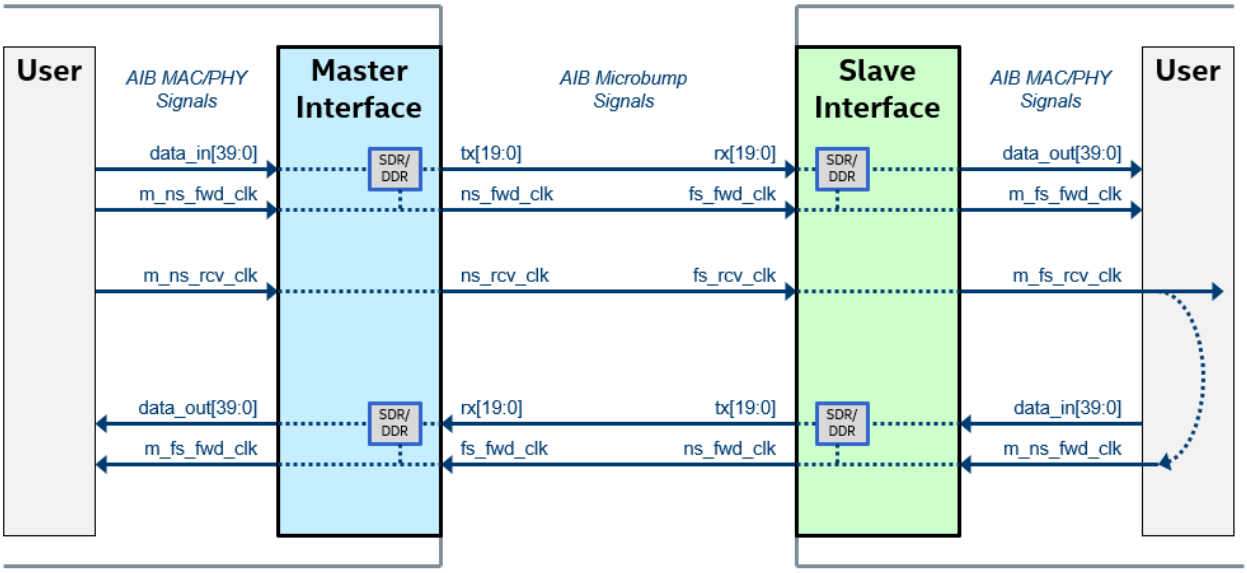
Table 4 lists the AIB application/PHY signals at the AIB interface level, in contrast to the per-channel signals just described. All signals are asynchronous except the free running clock input.

Table 4. AIB Application/PHY Signal Names

AIB Application/PHY Signals as implemented in Open Source .sv files	In (from application Out (to application)	Description
m_power_on_reset*	out	Used with Master only. A copy of the por signal from the Slave, qualified by override by i_aibaux_por_vccl_ovrd or m_por_ovrd.
m_power_on_reset_i	in	Open Source v2 only. Used with Slave only. Controls the por signal sent to the Master.
i_aibaux_por_vccl_ovrd (1) m_por_ovrd* (2)	in	Used with Master only. The Master chiplet should receive this signal from a Master C4 bump and input it to the AIB PHY. (1) Open Source v1. Note polarity is reversed from spec sense. Note Open Source v1 is Master only. If 0, the Master is held in por reset. If 1, the Master uses the AUX por input. (2) Open Source v2. If 0, the Master uses the AUX por input. If 1, the Master is held in por reset.
m_device_detect*	out	Open Source v2 only. Used with Slave only. A copy of the device_detect signal from the Master, qualified by m_device_detect_ovrd.
m_device_detect_ovrd*	in	Open Source v2 only. Used with Slave only. The Slave chiplet should receive this signal from a Slave C4 bump and input it to the AIB PHY. A Slave uses this input for test to control the Slave AIB device_detect when not connected to a Master. If 0, the Slave uses the device_detect from the AUX. If 1, the Slave outputs m_device_detect=1.
dual_mode_select	in	Open Source v2 only. If 1, configure the AIB PHY as Master. If 0, configure the AIB PHY as Slave.
i_adpt_hard_rst_n (1) i_conf_done (2)	in	Single control to reset all AIB adapters in the interface. LO=reset, HI=out of reset. This same signal from the application should be used to control the chiplet's CONF_DONE pin (LO=CONF_DONE is pulled down, HI=CONF_DONE is not pulled down). (1) Open Source v1. (2) Open Source v2. Note that the application must read the value of the CONF_DONE pin to detect if other chiplets have completed configuration, and that read capability is not defined here.
i_iocsr_rdy_aibaux	in	Open Source v1 only. Reset of AUX channel. Connect to same source as connected to i_adpt_hard_rst_n.
i_osc_clk	in	A free running clock from the application. You must configure the AIB AUX channel to select this clock instead of the AUX internally generated clock.

* See Figure 11, Figure 12 and Figure 13 for additional descriptions.

6 AIB Master and Slave Pair Examples with MAC/PHY Signals



Example Master / Slave DDR Register Mode AIB with Master as Clock Source

Figure 6. Example Master / Slave DDR Register Mode AIB with Master as Clock Source

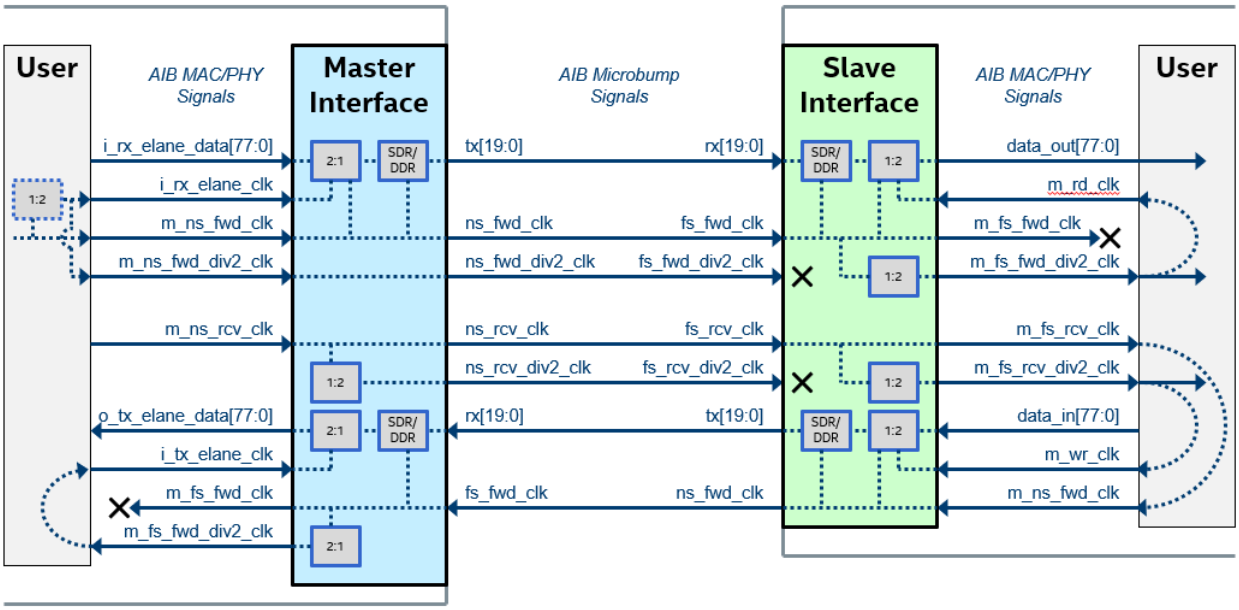


Figure 7. Example 2x Mode Master / 2x Mode Slave DDR Phase Comp FIFO Mode AIB with Master as clock source. In this example the Master uses Open Source v1 signal names and the Slave generates MAC/PHY div2 clocks locally. Contrast with a Stratix 10 slave that requires the Master to send div2 clocks (see section 3.2.1).

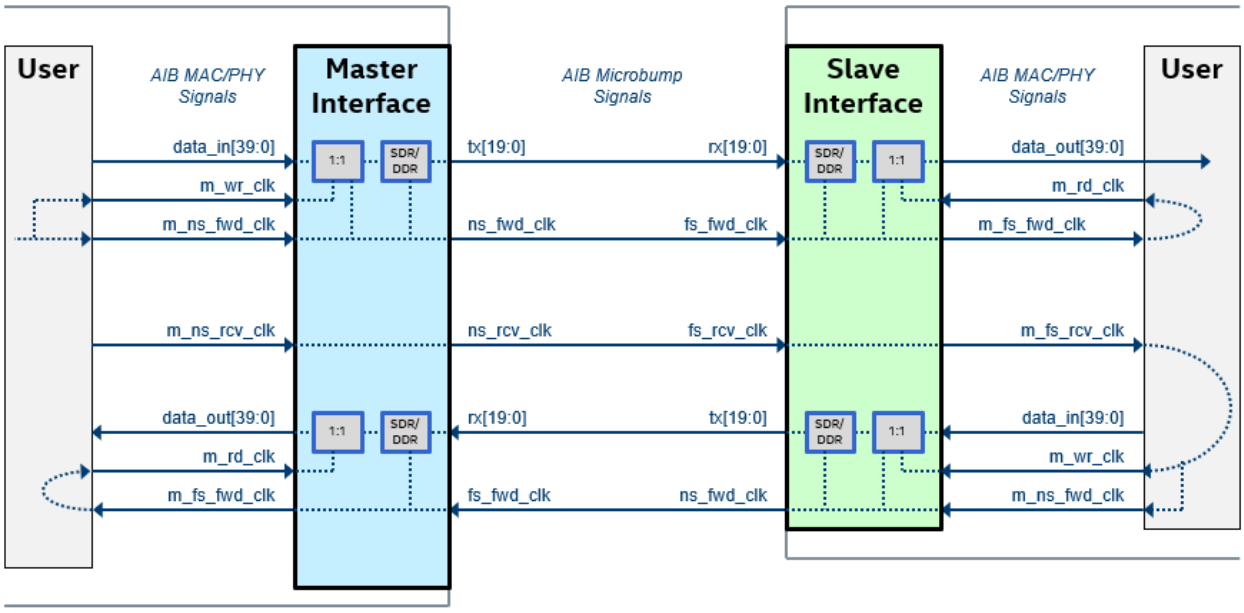


Figure 8. Example 1x Mode Master / 1x Mode Slave DDR Phase Comp FIFO Mode AIB with Master as clock source

7 AIB Physical Design

7.1 Stratix 10 EMIB

A S10 chiplet uses the Stratix 10 EMIB that connects to the West side of Stratix 10. This defines a S10 chiplet as having AIB on its East side as shown in Figure 9.Figure 1

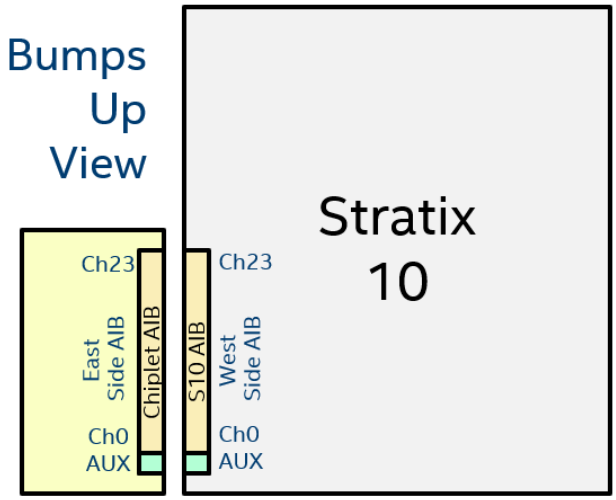


Figure 9. S10 Chiplet with East Side AIB

7.2 AIB Bump Assignments

7.2.1 Data Channel AIB Bump Assignments and Input/Output Configuration with Stratix 10

You should use the existing Master or Slave IO configurations of the AIB open source in your design. Since unused pins in an interface may be configured as input or output by Stratix 10, the interface needs to make sure it does not cause driver conflict or allow an input to float.

Table 5. AIB Bump Table with IO Configuration as Used by AIB Open Source

Bump ID	Slave Bump Name	Slave IO[1]	Master Bump Name	Master IO[1]
AIB61	unused_AIB61	GND	unused_AIB61	in
AIB50	unused_AIB50	in	unused_AIB50	GND
AIB72	unused_AIB72	GND	unused_AIB72	in
AIB73	unused_AIB73	GND	unused_AIB73	in
AIB75	unused_AIB75	in	unused_AIB75	GND
AIB74	unused_AIB74	GND	unused_AIB74	in
AIB91	unused_AIB91	in	unused_AIB91	GND
AIB90	unused_AIB90	in	unused_AIB90	GND
AIB95	fs_sr_data	in	ns_sr_data	out
AIB94	fs_sr_load	in	ns_sr_load	out
AIB85	fs_sr_clk	in	ns_sr_clk	out
AIB84	fs_sr_clkb	in	ns_sr_clkb	out
AIB76	unused_AIB76	in	unused_AIB76	GND
AIB77	unused_AIB77	in	unused_AIB77	GND
AIB58	unused_AIB58	GND	unused_AIB58	in
AIB63	unused_AIB63	GND	unused_AIB63	in
AIB48	fs_rcv_div2_clk	in	ns_rcv_div2_clk	out
AIB55	fs_rcv_div2_clkb	in	ns_rcv_div2_clkb	out
AIB62	unused_AIB62	in	unused_AIB62	GND
AIB60	unused_AIB60	in	unused_AIB60	GND
AIB53	fs_fwd_div2_clk	in	ns_fwd_div2_clk	out
AIB54	fs_fwd_div2_clkb	in	ns_fwd_div2_clkb	out
AIB49	fs_mac_rdy	in	ns_mac_rdy	out
AIB56	fs_adapt_rstn	in	ns_adapt_rstn[2]	out
AIB51	unused_AIB51	in	unused_AIB51	GND
AIB52	unused_AIB52	in	unused_AIB52	GND
AIB57	ns_rcv_clk	out	fs_rcv_clk	in
AIB59	ns_rcv_clkb	out	fs_rcv_clkb	in
AIB64	unused_AIB64	GND	unused_AIB64	in
AIB65	ns_adapt_rstn	out	fs_adapt_rstn	in
AIB80	unused_AIB80	GND	unused_AIB80	in
AIB81	unused_AIB81	GND	unused_AIB81	in
AIB78	unused_AIB78	GND	unused_AIB78	in
AIB79	unused_AIB79	GND	unused_AIB79	in
AIB87	fs_rcv_clk	in	ns_rcv_clk	out
AIB86	fs_rcv_clkb	in	ns_rcv_clkb	out
AIB83	ns_sr_clk	out	fs_sr_clk	in
AIB82	ns_sr_clkb	out	fs_sr_clkb	in
AIB89	unused_AIB89	GND	unused_AIB89	in
AIB88	unused_AIB88	GND	unused_AIB88	in
AIB93	ns_sr_data	out	fs_sr_data	in
AIB92	ns_sr_load	out	fs_sr_load	in
AIB71	unused_AIB71	in	unused_AIB71	GND
AIB70	unused_AIB70	in	unused_AIB70	GND
AIB68	unused_AIB68	in	unused_AIB68	GND
AIB69	unused_AIB69	in	unused_AIB69	GND
AIB66	unused_AIB66	in	unused_AIB66	GND
AIB67	unused_AIB67	GND	unused_AIB67	in
AIB20	tx[0]	out	rx[0]	in
AIB21	tx[1]	out	rx[1]	in
AIB22	tx[2]	out	rx[2]	in

Bump ID	Slave Bump Name	Slave IO[1]	Master Bump Name	Master IO[1]
AIB23	tx[3]	out	rx[3]	in
AIB24	tx[4]	out	rx[4]	in
AIB25	tx[5]	out	rx[5]	in
AIB26	tx[6]	out	rx[6]	in
AIB27	tx[7]	out	rx[7]	in
AIB28	tx[8]	out	rx[8]	in
AIB29	tx[9]	out	rx[9]	in
AIB43	ns_fwd_clk	out	fs_fwd_clk	in
AIB42	ns_fwd_clkb	out	fs_fwd_clkb	in
AIB30	tx[10]	out	rx[10]	in
AIB31	tx[11]	out	rx[11]	in
AIB32	tx[12]	out	rx[12]	in
AIB33	tx[13]	out	rx[13]	in
AIB34	tx[14]	out	rx[14]	in
AIB35	tx[15]	out	rx[15]	in
AIB36	tx[16]	out	rx[16]	in
AIB37	tx[17]	out	rx[17]	in
AIB38	tx[18]	out	rx[18]	in
AIB39	tx[19]	out	rx[19]	in
AIB44	ns_mac_rdy	out	fs_mac_rdy	in
AIB45	unused_AIB45	GND	unused_AIB45	in
AIB18	rx[18]	in	tx[18]	out
AIB19	rx[19]	in	tx[19]	out
AIB16	rx[16]	in	tx[16]	out
AIB17	rx[17]	in	tx[17]	out
AIB14	rx[14]	in	tx[14]	out
AIB15	rx[15]	in	tx[15]	out
AIB12	rx[12]	in	tx[12]	out
AIB13	rx[13]	in	tx[13]	out
AIB10	rx[10]	in	tx[10]	out
AIB11	rx[11]	in	tx[11]	out
AIB41	fs_fwd_clk	in	ns_fwd_clk	out
AIB40	fs_fwd_clkb	in	ns_fwd_clkb	out
AIB8	rx[8]	in	tx[8]	out
AIB9	rx[9]	in	tx[9]	out
AIB6	rx[6]	in	tx[6]	out
AIB7	rx[7]	in	tx[7]	out
AIB4	rx[4]	in	tx[4]	out
AIB5	rx[5]	in	tx[5]	out
AIB2	rx[2]	in	tx[2]	out
AIB3	rx[3]	in	tx[3]	out
AIB0	rx[0]	in	tx[0]	out
AIB1	rx[1]	in	tx[1]	out
AIB46	unused_AIB46	in	unused_AIB46	GND
AIB47	unused_AIB47	in	unused_AIB47	GND

1. GND means set that AIB output to LO (GND).
2. In Open Source v1 the Master signal ns_adapter_rstn does not exist. The Master should set this output HI.

7.2.2 AUX Channel Bump Assignments

A chiplet AIB interface AUX should only electrically connect to the *por* and *device_detect* signals. Any unused AUX microbumps should not be electrically connected on-die, even if a microbump is present on the chiplet for mechanical attachment to the package.

Master interface chiplets connecting to Stratix 10 should set Stratix 10's unused Slave AUX inputs LO. This can be accomplished by connecting the Master side package vias to the Stratix 10 EMIB to a "VSSP" ground which is connected to a package ball and to board VSS/GND though a 1K ohm resistor. If you know the Slave is a chiplet that has "no connect" at those locations instead of inputs, you may use "no connect" on the Master side instead of VSSP.

For Masters, the two *device_detect* bumps should be connected by Master on-die wiring after the Master output buffer. For Masters, the two *por* bumps should be connected by Master on-die wiring before the Master input buffer.

For Slaves, the two *device_detect* bumps should be connected by Slave on-die wiring before the Slave input buffer. For Slaves, the two *por* bumps should be connected by Slave on-die wiring after the Slave output buffer.

Table 6. AUX Channel Bump Table with IO Configuration as Used by AIB Open Source

Bump ID	Slave Bump Name	Slave AUX IO	Slave-side connection to Master Chiplet	Master Bump Name	Master AUX IO	Master-side connection to Slave Chiplet
AIB0	unused_AIB0	n/a	no connect	unused_AIB0	n/a	VSSP[1]
AIB1	unused_AIB1	n/a	no connect	unused_AIB1	n/a	VSSP
AIB10	unused_AIB10	n/a	no connect	unused_AIB10	n/a	VSSP
AIB11	unused_AIB11	n/a	no connect	unused_AIB11	n/a	VSSP
AIB12	unused_AIB12	n/a	no connect	unused_AIB12	n/a	VSSP
AIB13	unused_AIB13	n/a	no connect	unused_AIB13	n/a	VSSP
AIB14	unused_AIB14	n/a	no connect	unused_AIB14	n/a	VSSP
AIB15	unused_AIB15	n/a	no connect	unused_AIB15	n/a	VSSP
AIB16	unused_AIB16	n/a	no connect	unused_AIB16	n/a	VSSP
AIB17	unused_AIB17	n/a	no connect	unused_AIB17	n/a	VSSP
AIB18	unused_AIB18	n/a	no connect	unused_AIB18	n/a	VSSP
AIB19	unused_AIB19	n/a	no connect	unused_AIB19	n/a	VSSP
AIB2	unused_AIB2	n/a	no connect	unused_AIB2	n/a	VSSP
AIB20	unused_AIB20	n/a	no connect	unused_AIB20	n/a	VSSP
AIB21	unused_AIB21	n/a	no connect	unused_AIB21	n/a	VSSP
AIB22	unused_AIB22	n/a	no connect	unused_AIB22	n/a	VSSP
AIB23	unused_AIB23	n/a	no connect	unused_AIB23	n/a	VSSP
AIB24	unused_AIB24	n/a	no connect	unused_AIB24	n/a	no connect
AIB25	unused_AIB25	n/a	no connect	unused_AIB25	n/a	no connect
AIB26	unused_AIB26	n/a	no connect	unused_AIB26	n/a	VSSP
AIB27	unused_AIB27	n/a	no connect	unused_AIB27	n/a	no connect
AIB28	unused_AIB28	n/a	no connect	unused_AIB28	n/a	no connect
AIB29	unused_AIB29	n/a	no connect	unused_AIB29	n/a	no connect
AIB3	unused_AIB3	n/a	no connect	unused_AIB3	n/a	VSSP
AIB30	unused_AIB30	n/a	no connect	unused_AIB30	n/a	no connect
AIB31	unused_AIB31	n/a	no connect	unused_AIB31	n/a	no connect
AIB32	unused_AIB32	n/a	no connect	unused_AIB32	n/a	no connect
AIB33	unused_AIB33	n/a	no connect	unused_AIB33	n/a	no connect
AIB34	unused_AIB34	n/a	no connect	unused_AIB34	n/a	no connect
AIB35	unused_AIB35	n/a	no connect	unused_AIB35	n/a	no connect

Bump ID	Slave Bump Name	Slave AUX IO	Slave-side connection to Master Chiplet	Master Bump Name	Master AUX IO	Master-side connection to Slave Chiplet
AIB36	unused_AIB36	n/a	no connect	unused_AIB36	n/a	no connect
AIB37	unused_AIB37	n/a	no connect	unused_AIB37	n/a	no connect
AIB38	unused_AIB38	n/a	no connect	unused_AIB38	n/a	no connect
AIB39	unused_AIB39	n/a	no connect	unused_AIB39	n/a	no connect
AIB4	unused_AIB4	n/a	no connect	unused_AIB4	n/a	VSSP
AIB40	unused_AIB40	n/a	no connect	unused_AIB40	n/a	no connect
AIB41	unused_AIB41	n/a	no connect	unused_AIB41	n/a	VSSP
AIB42	unused_AIB42	n/a	no connect	unused_AIB42	n/a	VSSP
AIB43	unused_AIB43	n/a	no connect	unused_AIB43	n/a	VSSP
AIB44	unused_AIB44	n/a	no connect	unused_AIB44	n/a	VSSP
AIB45	unused_AIB45	n/a	no connect	unused_AIB45	n/a	no connect
AIB46	unused_AIB46	n/a	no connect	unused_AIB46	n/a	no connect
AIB47	unused_AIB47	n/a	no connect	unused_AIB47	n/a	no connect
AIB48	unused_AIB48	n/a	no connect	unused_AIB48	n/a	no connect
AIB49	unused_AIB49	n/a	no connect	unused_AIB49	n/a	no connect
AIB5	unused_AIB5	n/a	no connect	unused_AIB5	n/a	VSSP
AIB50	unused_AIB50	n/a	no connect	unused_AIB50	n/a	no connect
AIB51	unused_AIB51	n/a	no connect	unused_AIB51	n/a	no connect
AIB52	unused_AIB52	n/a	no connect	unused_AIB52	n/a	no connect
AIB53	unused_AIB53	n/a	no connect	unused_AIB53	n/a	no connect
AIB54	unused_AIB54	n/a	no connect	unused_AIB54	n/a	no connect
AIB55	unused_AIB55	n/a	no connect	unused_AIB55	n/a	no connect
AIB56	unused_AIB56	n/a	no connect	unused_AIB56	n/a	VSSP
AIB57	unused_AIB57	n/a	no connect	unused_AIB57	n/a	no connect
AIB58	unused_AIB58	n/a	no connect	unused_AIB58	n/a	VSSP
AIB59	unused_AIB59	n/a	no connect	unused_AIB59	n/a	no connect
AIB6	unused_AIB6	n/a	no connect	unused_AIB6	n/a	VSSP
AIB60	unused_AIB60	n/a	no connect	unused_AIB60	n/a	VSSP
AIB61	unused_AIB61	n/a	no connect	unused_AIB61	n/a	VSSP
AIB62	unused_AIB62	n/a	no connect	unused_AIB62	n/a	VSSP
AIB63	unused_AIB63	n/a	no connect	unused_AIB63	n/a	VSSP
AIB64	unused_AIB64	n/a	no connect	unused_AIB64	n/a	VSSP
AIB65	unused_AIB65	n/a	no connect	unused_AIB65	n/a	VSSP
AIB66	unused_AIB66	n/a	no connect	unused_AIB66	n/a	VSSP
AIB67	unused_AIB67	n/a	no connect	unused_AIB67	n/a	VSSP
AIB68	unused_AIB68	n/a	no connect	unused_AIB68	n/a	VSSP
AIB69	unused_AIB69	n/a	no connect	unused_AIB69	n/a	VSSP
AIB7	unused_AIB7	n/a	no connect	unused_AIB7	n/a	VSSP
AIB70	unused_AIB70	n/a	no connect	unused_AIB70	n/a	VSSP
AIB71	unused_AIB71	n/a	no connect	unused_AIB71	n/a	VSSP
AIB72	unused_AIB72	n/a	no connect	unused_AIB72	n/a	VSSP
AIB73	unused_AIB73	n/a	no connect	unused_AIB73	n/a	VSSP
AIB74	device_detect	in	Slave chiplet input through microbump (AUX case) or C4 bump (No AUX)	device_detect	out	Master chiplet output through microbump (AUX case) or C4 bump (No AUX)
AIB75	device_detect	n/a	Slave chiplet input through microbump (AUX case) or package connection to AIB74 (No AUX)	device_detect	n/a	Master chiplet output through microbump (AUX case) or package connection to AIB74 (No AUX)
AIB76	unused_AIB76	n/a	no connect	unused_AIB76	n/a	VSSP
AIB77	unused_AIB77	n/a	no connect	unused_AIB77	n/a	VSSP
AIB78	unused_AIB78	n/a	no connect	unused_AIB78	n/a	VSSP
AIB79	unused_AIB79	n/a	no connect	unused_AIB79	n/a	VSSP
AIB8	unused_AIB8	n/a	no connect	unused_AIB8	n/a	VSSP

Bump ID	Slave Bump Name	Slave AUX IO	Slave-side connection to Master Chiplet	Master Bump Name	Master AUX IO	Master-side connection to Slave Chiplet
AIB80	unused_AIB80	n/a	no connect	unused_AIB80	n/a	no connect
AIB81	unused_AIB81	n/a	no connect	unused_AIB81	n/a	no connect
AIB82	unused_AIB82	n/a	no connect	unused_AIB82	n/a	no connect
AIB83	unused_AIB83	n/a	no connect	unused_AIB83	n/a	no connect
AIB84	unused_AIB84	n/a	no connect	unused_AIB84	n/a	no connect
AIB85	por	out	Slave chiplet output through microbump (AUX case) or C4 bump (No AUX)	por	in	Master chiplet input through microbump (AUX case) or C4 bump (No AUX)
AIB86	unused_AIB86	n/a	no connect	unused_AIB86	n/a	no connect
AIB87	por	n/a	Slave chiplet output through microbump (AUX case) or package connection to AIB85 (No AUX)	por	n/a	Master chiplet input through microbump (AUX case) or package connection to AIB85 (No AUX)
AIB88	unused_AIB88	n/a	no connect	unused_AIB88	n/a	no connect
AIB89	unused_AIB89	n/a	no connect	unused_AIB89	n/a	no connect
AIB9	unused_AIB9	n/a	no connect	unused_AIB9	n/a	VSSP
AIB90	unused_AIB90	n/a	no connect	unused_AIB90	n/a	no connect
AIB91	unused_AIB91	n/a	no connect	unused_AIB91	n/a	no connect
AIB92	unused_AIB92	n/a	no connect	unused_AIB92	n/a	VSSP
AIB93	unused_AIB93	n/a	no connect	unused_AIB93	n/a	VSSP
AIB94	unused_AIB94	n/a	no connect	unused_AIB94	n/a	VSSP
AIB95	unused_AIB95	n/a	no connect	unused_AIB95	n/a	VSSP

1. All VSSP may be connected together and pulled down by a single 1K resistor to VSS or GND. If the connected Slave device is known to have no connect at those locations, you may substitute no connect for VSSP.

7.2.3 Alternate No AUX Channel AIB

With only 4 bumps utilized in the AUX channel, it is anticipated that an AIB interface may be built without an AIB AUX channel. This may be useful in fitting a 24 channel AIB interface into an 8mm maximum die height.

A No AUX Master interface in this case may drive *device_detect* from a C4 bump and receive *por* with a C4 bump, each connected to respective Stratix 10 EMIB microbumps through package surface traces. See Figure 10 for an example No AUX Master. To avoid floating inputs to Stratix 10, at the EMIB you should tie those pins to VSSP (1K ohm pulldown to VSS or GND).

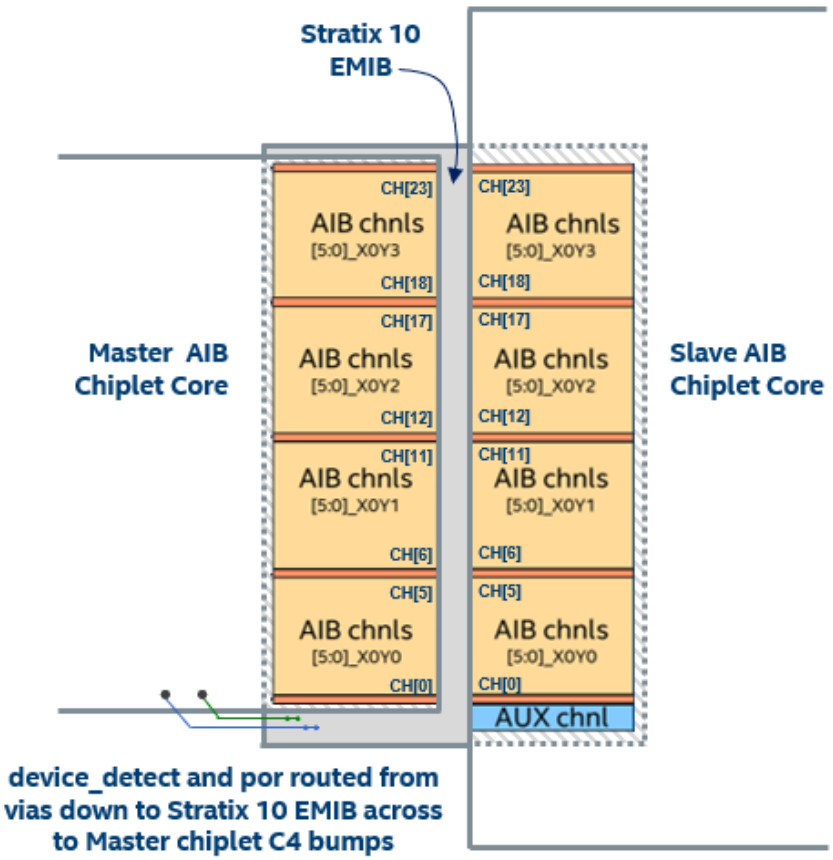


Figure 10. No AUX Master Chiplet

A No AUX Master drives *device_detect* and observes *por* from C4 bump IOs with the same behavior as described in the AIB Spec. A No AUX Slave may similarly be constructed with vias to EMIB routed to C4 bump IOs on the Slave chiplet. See examples in Figure 11, Figure 12 and Figure 13.

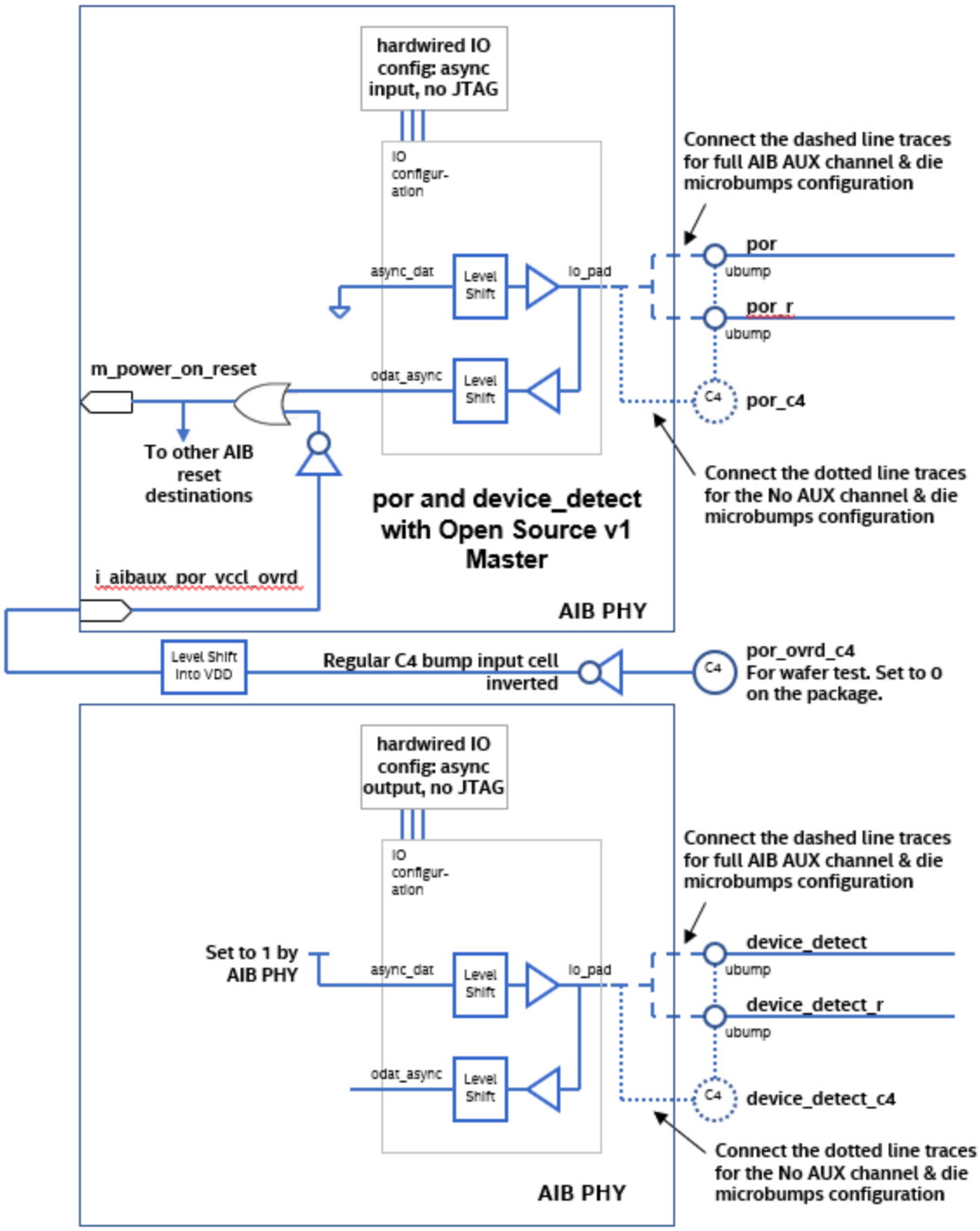


Figure 11. Example por and device_detect with Open Source v1 Master including No AUX connections

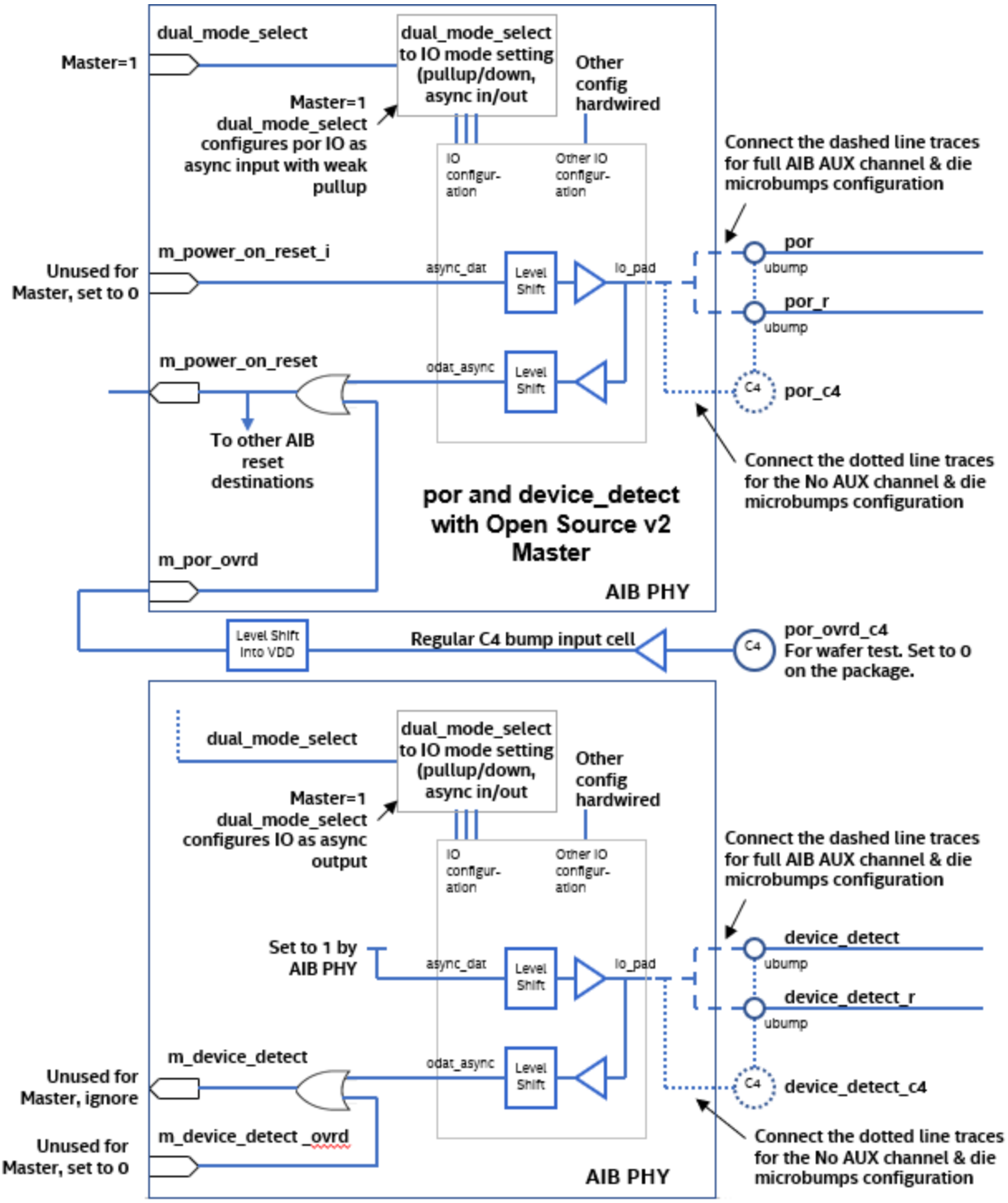


Figure 12. Example por and device_detect with Open Source v2 Master including No AUX connections

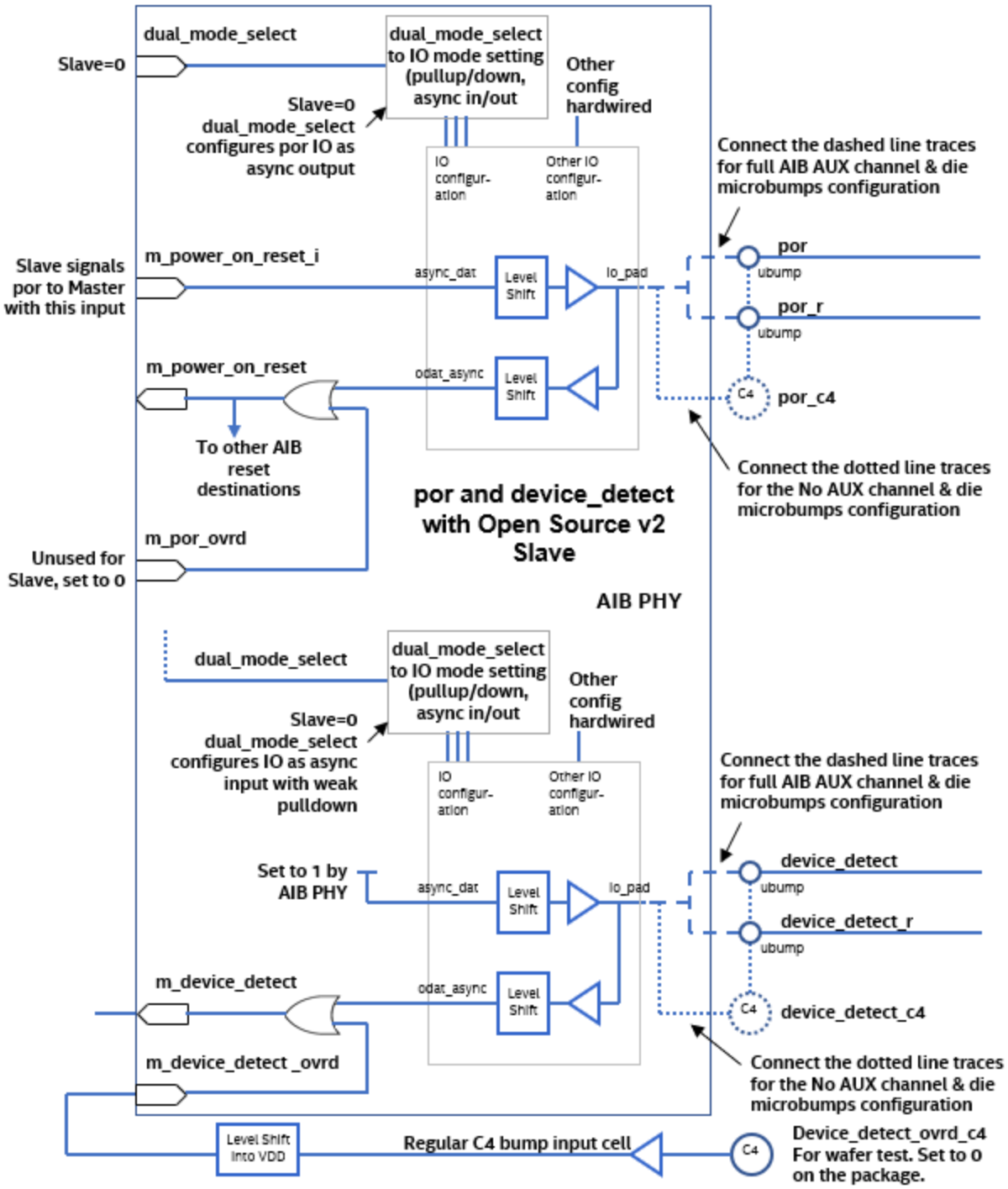


Figure 13. Example por and device_detect with Open Source v2 Slave including No AUX connections

7.3 Channel Stacking

A Master uses the East channel stacking as shown in the AIB Spec. A Slave uses the West channel stacking. For compatibility with Stratix 10 both Master and Slave add two rows of microbumps in gaps between AUX and channel0, channels 5 and 6, 11 and 12, 17 and 18, and above 23 as shown in Figure 14:

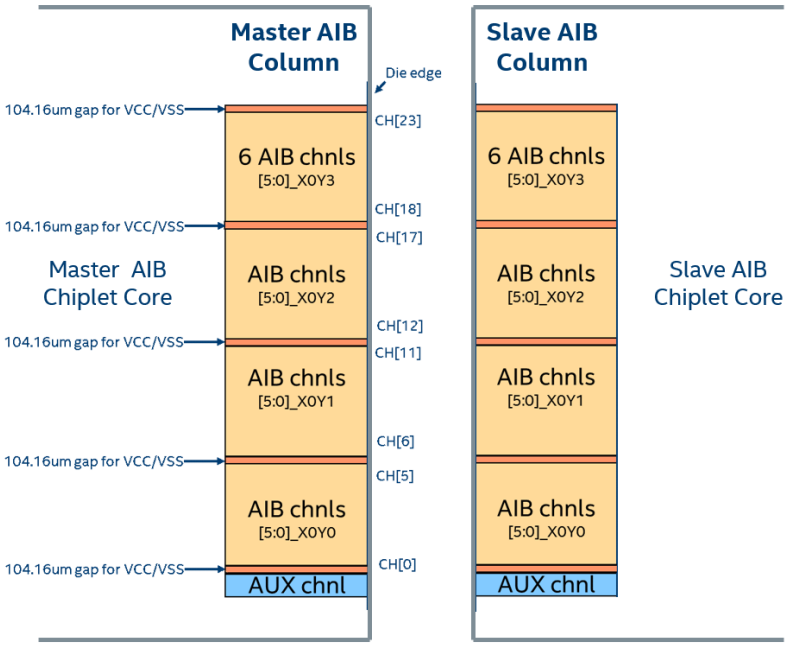


Figure 14. AIB Channel Stacking

7.4 AIB Power and Ground

The Alternate Bump Map in the AIB Spec identifies bumps as VCCIO and VDD.

7.4.1 VCCIO

VCCIO is supplied by Stratix 10. A Stratix 10 chiplet uses VCCIO to power the input and output stages of its AIBIO cells, and draws no more current than the maximum of Table 7, as stated in the AIB Spec.

Table 7. S10 Chiplet Maximum Current Draw from VCCIO

VCCIO Maximum Current
40mA per channel

For testability of the S10 chiplet, the S10 chiplet connects VCCIO to C4 bumps, typically 3 per 24 channel AIB interface. Microbumps are typically not probed, and the C4 bumps are a means for test equipment to power the S10 chiplet's VCCIO before assembly.

A Stratix 10 chiplet provides voltage translators between the input and output stages of the AIBIO cells and the rest of the S10 chiplet.

7.4.2 VDD

VDD is optionally supplied by a S10 chiplet into the VDD microbumps on rows AQ through AT, and drawn from the VDD microbumps on rows Y and Z. The purpose is to supply power to S10 chiplet circuitry in the area under the AIB bump array that is not powered by VCCIO. See Figure 15.

7.4.3 Power and Ground Bumps

The Stratix 10 EMIB connects all the VDD microbumps together, all the VSS microbumps together and all the VCCIO microbumps together. The multi-chip package of chiplets + Stratix 10 should connect VDD C4 bumps and microbumps together, also VSS C4 bumps and microbumps together using surface traces. VCCIO microbumps should be connected in the package using surface traces.

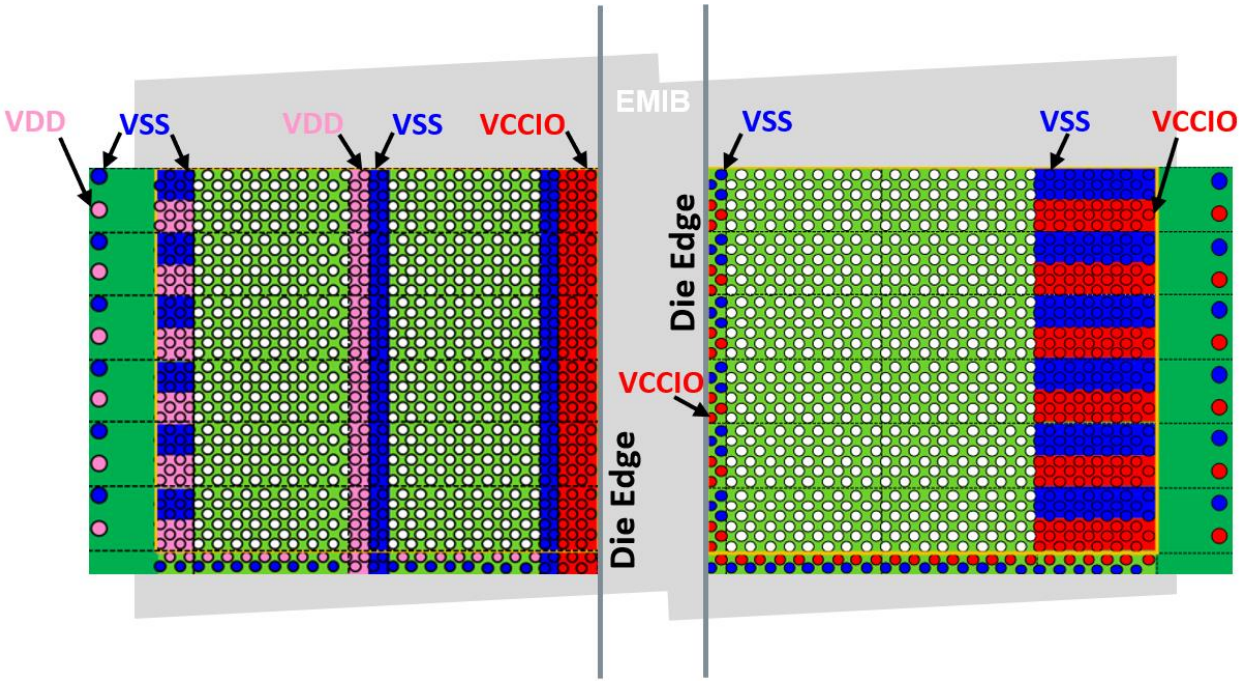


Figure 15. Master (left) and Slave (right) AIB Power Microbumps using Stratix 10 EMIB. Six AIB channels with one power/VSS gap at bottom.

8 AIB Bump Locations and Assignments

The locations of microbumps and surrounding C4 bumps are given in the spreadsheet referenced in section 8.1. Table 8 gives the spreadsheet's bump names and description, and the translation to the Verilog names of module aib_top.

Table 8. Master Chiplet AIB Bumps

Location Spreadsheet Bump/Pin name	Verilog aib_top signal name	Pin direction	Bump type	Pin Type/Function
AIB{0-95}_CH{0-5}_X0Y0	io_aib_ch{0-5}[95:0]	inout	µbump	AIB signals for channels 0-5
AIB{0-95}_CH{0-5}_X0Y1	io_aib_ch{6-11}[95:0]	inout	µbump	AIB signals for channels 6-11
AIB{0-95}_CH{0-5}_X0Y2	io_aib_ch{12-17}[95:0]	inout	µbump	AIB signals for channels 12-17
AIB{0-95}_CH{0-5}_X0Y3	io_aib_ch{18-23}[95:0]	inout	µbump	AIB signals for channels 18-23
AIB_AUX{0-95}_X0Y0	io_aib_aux[95:0]	inout	µbump	AIB signals for aux channel
VCC_HSSI (AIB Spec uses VDD)	n/a	power	C4/µbump	Digital supply for AIB and MAC circuits on chiplet core side of AIBIO (away from microbumps)
VCCL_HSSI (AIB Spec uses VCCIO)	n/a	power	C4/µbump	IO supply from Stratix 10 regulated to 0.75V-0.97V for S10 chiplet AIBIO circuits on microbump side of AIB IO cell
VSSGND (AIB Spec uses VSS)	n/a	ground	C4/µbump	Digital VSS

8.1 AIB Bump Locations Spreadsheet

Needs Updating for Slave AIB Bump Locations. See the companion file “Stratix 10 Chiplet AIB Profile_v1_0_aib_bump_locations.xlsx”.

8.2 Bump Mechanical

The figure and table below are for chiplets using the existing Stratix 10 EMIB.

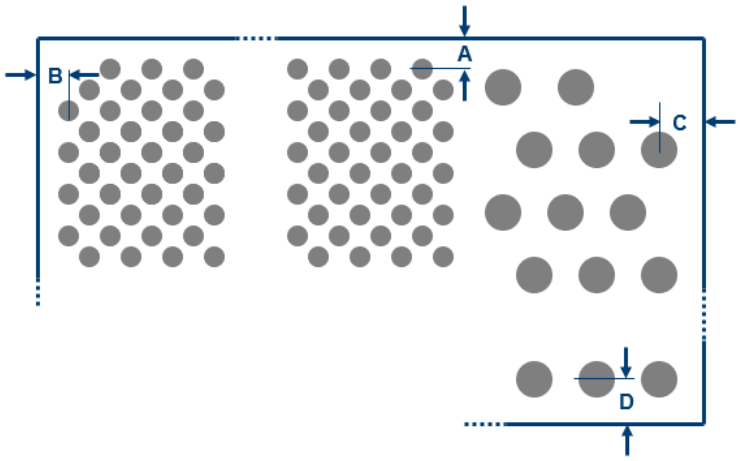


Figure 16. S10 Chiplet Bump Centers to Die Edges

Table 9. S10 Chiplet Feature Dimensions

Feature	Dimension Max	Dimension Min
A	127.6 um	101.12 um
B	163.48 um	101.12 um
C	241.96 um	123.96 um
D	241.96 um	123.96 um

The dimensions in Table 9 are to the scribe line center. Use the bump locations spreadsheet for the microbump to C4 bump spacing and positions.

9 AIB Initialization

The following steps are a combination of the AIB Spec, Stratix 10 requirements and application of the AIB Open Source.

Table 10. AIB Initialization

Master AIB Initialization	Slave AIB Initialization
• Exit reset. • Set AIB AUX *device_detect* high. • AIB AUX por signal from the Slave is used by the Master interface to keep its AIB outputs in standby mode.	• Exit reset. • Read the AIB AUX *device_detect* signal. • If *device_detect* is LO, there is no Master attached and the Slave interface should remain in standby state. If *device_detect* is HI, the Slave drives the AIB AUX por signal HI.
Common Master and Slave Initialization	
• The applications hold their AIB adapters in reset by setting *i_adpt_hard_rst_n* LO. *i_adpt_hard_rst_n* LO also keeps the AIB outputs in standby mode. • Pull down the open-drain CONF_DONE pin to LO.	

<table>
<tr><td colspan="2">• Each MAC sets ns_mac_rdy LO.
• Each MAC sets ns_adapter_rstn LO.</td></tr>
<tr><th>Master AIB Initialization</th><th>Slave AIB Initialization</th></tr>
<tr><td>• Wait until AIB AUX por is LO.</td><td></td></tr>
<tr><th colspan="2">Common Master and Slave Initialization</th></tr>
<tr><td colspan="2">• Configure.
• Release CONF_DONE.
• Monitor CONF DONE for other chiplets that may still be pulling CONF_DONE LO. Once CONF_DONE is HI, set i_adpt_hard_rst_n HI.
• Each MAC sets ns_mac_rdy HI.
• Each MAC sets ns_adapter_rstn HI.
• The channel's calibration state machines are held in reset while fs_adapter_rstn is LO. Each AIB channel starts calibration once fs_adapter_rstn is HI. If you are using the AIB Open Source adapter then this step is handled for you.</td></tr>
<tr><th>Master AIB Initialization</th><th>Slave AIB Initialization</th></tr>
<tr><td>• Monitor sl_tx_transfer_en from the Slave, and ms_tx_transfer_en from the Master's own AIB Adapter.</td><td>• Monitor ms_tx_transfer_en from the Master, and sl_tx_transfer_en from the Slave's own AIB Adapter.</td></tr>
<tr><th colspan="2">Common Master and Slave Initialization</th></tr>
<tr><td colspan="2">• When both sl_tx_transfer_en and ms_tx_transfer_en are true, then the link shall be ready to transmit data.
• If you are using 2:1 phase comp FIFO mode, m_rxfifo_align_done=HI means the receive FIFO is aligned to the incoming upper word marking.</td></tr>
<tr><th>Master AIB Initialization</th><th>Slave AIB Initialization</th></tr>
<tr><td>• At any point, if the AIB AUX microbump signal por is HI then go back to the start.</td><td></td></tr>
</table>

9.1 Stratix 10 *device_detect* workaround

Stratix 10 has an issue with *device_detect* in that it expects unsupported JTAG behavior from the S10 Chiplet if Stratix 10 reads *device_detect* as HI. To work around this issue, in the package disconnect AUX AIB74 and AUX AIB75 between the Master Chiplet and the Stratix 10 EMIB. Ground those AUX AIB74 and AUX AIB75 Stratix 10 EMIB wires on the package.

10 Additional Information

[1] "Advanced Interface Bus Specification," Revision 1.2, Intel Corporation, September 2019, https://github.com/intel/aib-phy-hardware/tree/master/docs

Revision History

6/2019: Updates to initialization, mechanical, added the *ns_mac_rdy* signal from the AIB Spec into the list supported by a S10 Chiplet. Changed the MAC data bit numbering in the Word Marking section to match the AIB Open Source. Clarified 2x FIFO mode MAC/PHY clocking.

7/2019: Added *por_ovrd* with description. Simplified AIB Open Source MAC/PHY Signals table.

8/2019: Added Stratix 10 AIB Overview. Added clock reference to signals in the MAC/PHY signals table.

9/2019: Updated the bump to edge min and max dimensions.

10/2019: Added AIB Slave interface requirements and descriptions. Changed the document title to encompass chiplets that have master and/or slave interfaces.

11/2019: Added m_rxfifo_align_done to MAC/PHY signal list. Added m_rxfifo_align_done to MAC/PHY signal list.