

AXI4 Protocol over AIB Latency Reference Guide Revision 1.0

24 FEBRUARY 2023

Contents

1. Background	2
2. Simulation Parameters.....	2
3. Channel Aligner Modes	3
4. AXI4-Stream Latency and FIFO Buffer (Channel Aligner in Synchronous Mode, SYNC_FIFO=1)	3
5. AXI4 Latency and FIFO Buffer (Channel Aligner in Synchronous Mode, SYNC_FIFO=1)	4
6. AXI4-Stream Latency and FIFO Buffer (Channel Aligner in Asynchronous Mode, SYNC_FIFO=0)	5
7. AXI4 Latency and FIFO Buffer (Channel Aligner in Asynchronous Mode, SYNC_FIFO=0)	6
8. How to Specify RX FIFO Buffer Size	6
8.1. AXI4-Stream	6
8.2. AXI4-MM	7
9. Latency Optimization	9
10. Addendum: Example waveform capture for Parameters A and B (Config 2 for AXI4-ST)	10
10.1. Latency A.....	10
10.2. Latency B.....	11

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.

Intel, the Intel logo and Stratix are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

© Intel Corporation.

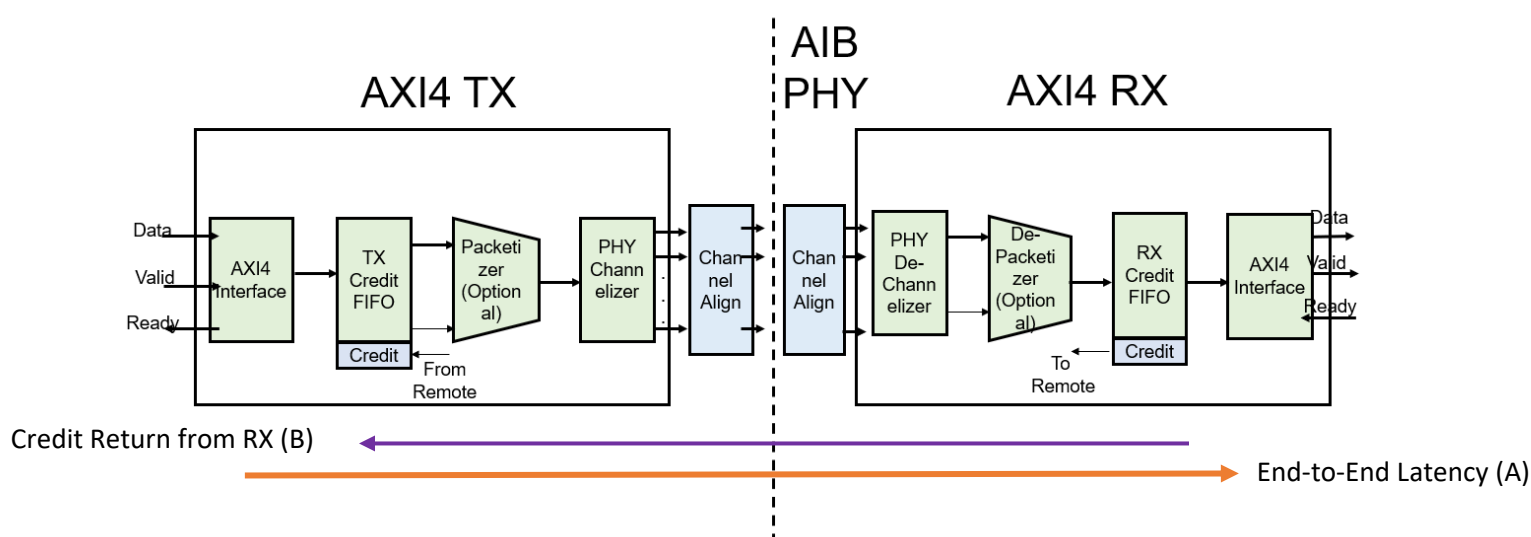
1. Background

There are 2 critical latency paths in AXI4 protocol:

- A) End-to-End latency – from TX to RX
- B) Credit return latency – from RX to TX

For AXI4 without flow control (AXI4-Stream without ready) B is not relevant for other AXI4 protocols with flow control both A and B play an important role as discussed below.

AXI4 protocol over AIB uses credit mechanism to manage flow control between follower and leader chiplet. This credit mechanism also allows to manage the multiple clock cycle handshake between chiplets.



To avoid stalling in the traffic (bubbles) the recommendation is to size the RX credit FIFO to match $A+B+k$ where k is an additional buffer. RX FIFO size is parameter for IP generation.

2. Simulation Parameters

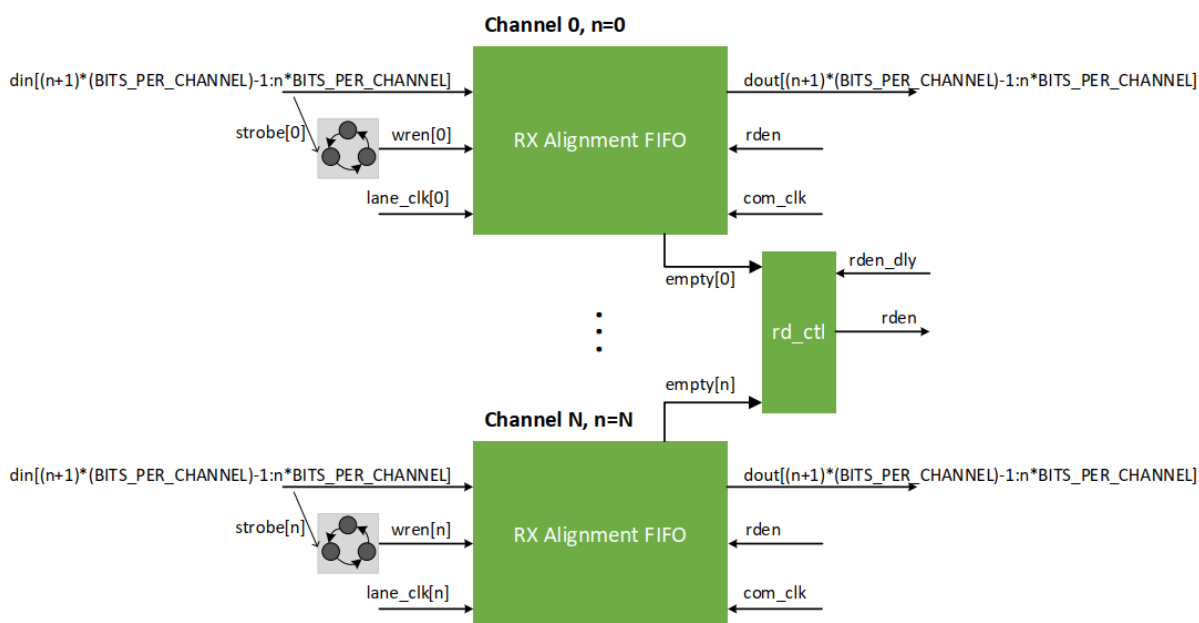
Below is a table of measured latency in simulation and RX FIFO buffer size recommendation for AXI4-Stream and AXI4 based on the following parameters.

AXI4 Mode	Freq (Ghz)
Half (Gen1)	0.5
Full (Gen1)	1
Quarter (Gen2)	0.5
Half (Gen2)	1
Full (Gen2)	2

3. Channel Aligner Modes

Supports 2 modes

- 1) SYNC_FIFO=1 channel align FIFO operates in synchronous mode, lane_clk and com_clk from same source which is usually m_rd_clk/m_wr_clk
- 2) SYNC_FIFO=0 channel align FIFO operates in asynchronous mode, lane_clk and com_clk from different source, FIFO write clock is lane_clk from the channel and the com_clk is m_rd_clk/m_wr_clk



Based on which mode you choose latency could be different, async mode results in slightly longer latency and therefore FIFO depth could be slightly more for async mode.

4. AXI4-Stream Latency and FIFO Buffer (Channel Aligner in Synchronous Mode, SYNC_FIFO=1)

Config No	Leader AIB Mode	Followe r AIB Mode	FIFO Mode (TX→RX)	A (normalized to AIB Fwd clk)	B (normalized to AIB Fwd clk)	k	Freq Factor between n AXI and AIB (F)	RX FIFO Size = Ceiling $((A+B+k)/F)$
1	AIB2.0 (Gen1)	AIB 1.0	Half→Half	44	40	2	2	43
2	AIB2.0 (Gen1)	AIB1.0	Full→Half	40	26	2	1	68

3	AIB2.0 (Gen1)	AIB1.0	Half→Full	34	40	2	1	76
4	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Half	50	38	2	2	45
5	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Full	31	28	2	1	61
6	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Quarter	88	66	2	4	39
7	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Half	48	30	2	1	80
8	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Quarter	84	31	2	1	117
9	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Full	34	38	2	1	74
10	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Quarter	88	44	2	1	134
11	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Full	39	65	2	1	106
12	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Half	58	66	2	1	126

5. AXI4 Latency and FIFO Buffer (Channel Aligner in Synchronous Mode, SYNC_FIFO=1)

Config No	Leader AIB Mode	Followe r AIB Mode	FIFO Mode (TX→RX)	A (normalized to AIB Fwd clk)	B (normalized to AIB Fwd clk)	k	Freq Factor between n AXI and AIB (F)	RX FIFO Size = Ceiling $((A+B+k)/F)$
1	AIB2.0 (Gen 1)	AIB 1.0	Half→Half	44	40	2	2	42
2	AIB2.0 (Gen 2)	AIB2.0 (Gen2)	Half→Half	52	42	2	2	48
3	AIB2.0 (Gen 2)	AIB2.0 (Gen2)	Full→Full	33	26	2	1	61
4	AIB2.0	AIB2.0 (Gen2)	Quarter→Quarter	92	72	2	4	42

	(Gen 2)							
--	---------	--	--	--	--	--	--	--

6. AXI4-Stream Latency and FIFO Buffer (Channel Aligner in Asynchronous Mode, SYNC_FIFO=0)

Config No	Leader AIB Mode	Followe r AIB Mode	FIFO Mode (TX→RX)	A (normalized to AIB Fwd clk)	B (normalized to AIB Fwd clk)	k	Freq Factor between n AXI and AIB (F)	RX FIFO Size = Ceiling ((A+B+k)/F)
1	AIB2.0 (Gen1)	AIB 1.0	Half→Half	50	46	2	2	49
2	AIB2.0 (Gen1)	AIB1.0	Full→Half	46	29	2	1	77
3	AIB2.0 (Gen1)	AIB1.0	Half→Full	37	45	2	1	84
4	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Half	50	38	2	2	45
5	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Full	34	31	2	1	67
6	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Quarter	88	64	2	4	39
7	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Half	54	33	2	1	89
8	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Quarter	96	34	2	1	132
9	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Full	37	45	2	1	84
10	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Quarter	100	50	2	1	152
11	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Full	44	76	2	1	122
12	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Half	64	80	2	1	146

7. AXI4 Latency and FIFO Buffer (Channel Aligner in Asynchronous Mode, SYNC_FIFO=0)

Config No	Leader AIB Mode	Followe r AIB Mode	FIFO Mode (TX→RX)	A (normalized to AIB Fwd clk)	B (normalized to AIB Fwd clk)	k	Freq Factor between n AXI and AIB (F)	RX FIFO Size = Ceiling $((A+B+k)/F)$
1	AIB2.0 (Gen1)	AIB 1.0	Half→Half	50	46	2	2	49
2	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Half→Half	58	48	2	2	54
3	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Full→Full	36	29	2	1	67
4	AIB2.0 (Gen2)	AIB2.0 (Gen2)	Quarter→Quarter	104	84	2	4	48

8. How to Specify RX FIFO Buffer Size

Can be specified as a configuration parameter in IP configuration file. Here are examples for AXI4-Stream and AXI4-MM

8.1. AXI4-Stream

```

MODULE axi_st
// PHY and AIB Configuration
NUM_CHAN                7
CHAN_TYPE                Gen1Only //Gen1Only, Gen2Only, Gen2, AIBO
// Channel Alignment Strobe Configuration
TX_ENABLE_STROBE        True      // If False, all strobe functionality is removed.
RX_ENABLE_STROBE        True      // If False, all strobe functionality is removed.

```

TX_PERSISTENT_STROBE reused for data	True	// If True strobes are persistent (always there). If false, they are recoverable and can be
RX_PERSISTENT_STROBE reused for data	True	// If True strobes are persistent (always there). If false, they are recoverable and can be
TX_USER_STROBE	True	// If True, then we input user generated signal
RX_USER_STROBE	True	// If True, then we input user generated signal
TX_STROBE_GEN1_LOC	8	// Location of Strobe when in Gen1 Mode
RX_STROBE_GEN1_LOC	8	// Location of Strobe when in Gen1 Mode
// Word Marker Configuration		
TX_ENABLE_MARKER	True	// If False, all Marker functionality is removed.
RX_ENABLE_MARKER	True	// If False, all Marker functionality is removed.
TX_PERSISTENT_MARKER reused for data	True	// If True Markers are persistent (always there). If false, they are recoverable and can be
RX_PERSISTENT_MARKER reused for data//	True	// If True Markers are persistent (always there). If false, they are recoverable and can be
TX_USER_MARKER	True	
RX_USER_MARKER	True	
TX_MARKER_GEN1_LOC	39	// Location of Marker when in Gen1 Mode
RX_MARKER_GEN1_LOC	39	// Location of Marker when in Gen1 Mode
TX_REG_PHY	False	
RX_REG_PHY	False	
SUPPORT_ASYMMETRIC llink ST	True	// Support Asymmetric Gearboxing (e.g. Full to Half)
{		
TX_FIFO_DEPTH	1	
RX_FIFO_DEPTH	68 # for Full→Half	
output user_tdata	256	
input user_tready	ready	
output user_tvalid	valid	
}		

8.2. AXI4-MM

MODULE axi_mm		
// PHY and AIB Configuration		
NUM_CHAN	2	
CHAN_TYPE	Gen1Only	//Gen1Only, Gen2Only, Gen2, AIBO
TX_RATE	Half	// Full, Half, Quarter
RX_RATE	Half	// Full, Half, Quarter
// Channel Alignment Strobe Configuration		
TX_ENABLE_STROBE	True	// If False, all strobe functionality is removed.
RX_ENABLE_STROBE	True	// If False, all strobe functionality is removed.
TX_PERSISTENT_STROBE be reused for data	True	// If True strobes are persistent (always there). If false, they are recoverable and can
RX_PERSISTENT_STROBE be reused for data	True	// If True strobes are persistent (always there). If false, they are recoverable and can
TX_USER_STROBE	False	// If True, then we input user generated signal
RX_USER_STROBE	False	// If True, then we input user generated signal
TX_STROBE_GEN1_LOC	7	// Location of Strobe when in Gen1 Mode
RX_STROBE_GEN1_LOC	7	// Location of Strobe when in Gen1 Mode
// Word Marker Configuration		
TX_ENABLE_MARKER	True	// If False, all Marker functionality is removed.
RX_ENABLE_MARKER	True	// If False, all Marker functionality is removed.
TX_PERSISTENT_MARKER be reused for data	True	// If True Markers are persistent (always there). If false, they are recoverable and can

RX_PERSISTENT_MARKER	True	// If True Markers are persistent (always there). If false, they are recoverable and can be reused for data//
TX_USER_MARKER	False	
RX_USER_MARKER	False	
TX_MARKER_GEN1_LOC	39	// Location of Marker when in Gen1 Mode
RX_MARKER_GEN1_LOC	39	// Location of Marker when in Gen1 Mode
TX_REG_PHY	False	
RX_REG_PHY	False	
// Packetization		
TX_ENABLE_PACKETIZATION	True	
RX_ENABLE_PACKETIZATION	True	
TX_PACKET_MAX_SIZE	0	// Number of bits to packetize to. 0 means all available data.
RX_PACKET_MAX_SIZE	0	// Number of bits to packetize to. 0 means all available data.
PACKETIZATION_PACKING_EN	False	// If True, enable packing which makes better use of the BW
llink AR		
{		
TX_FIFO_DEPTH	1	
RX_FIFO_DEPTH	42	
output user_arid 4		
output user_arsize 3		
output user_arlen 8		
output user_arburst 2		
output user_araddr 32		
output user_arvalid valid		
input user_arready ready		
}		
llink AW		
{		
TX_FIFO_DEPTH	1	
RX_FIFO_DEPTH	42	
output user_awid 4		
output user_awsized 3		
output user_awlen 8		
output user_awburst 2		
output user_awaddr 32		
output user_awvalid valid		
input user_awready ready		
}		
llink W		
{		
TX_FIFO_DEPTH	1	
RX_FIFO_DEPTH	42	
output user_wid 4		
output user_wdata 128		
output user_wstrb 16		
output user_wlast		
output user_wvalid valid		
input user_wready ready		
}		
llink R		
{		
TX_FIFO_DEPTH	1	
RX_FIFO_DEPTH	42	
input user_rid 4		
input user_rdata 128		
input user_rlast		
input user_rresp 2		
input user_rvalid valid		

```
output user_rready ready
}
llink B
{
  TX_FIFO_DEPTH    1
  RX_FIFO_DEPTH    42
  input user_bid    4
  input user_bresp  2
  input user_bvalid valid
  output user_bready ready
}
```

9. Latency Optimization

These latencies reported here are out of the box. Here are some tips to improve latency with system knowledge:

- a) Lane to lane to skew modeled in testbench as delay between AIB and CA. This delay is worst cased at 5 clock cycle this can be optimized for your interface and system once you have the system parameters extracted.
- b) Improve RX/TX phasecomp RD delay with the knowledge of AIB write vs read clock wander margin. These values can be programmed to AIB CSR 0x208/0x218.

10.1. Latency A

[illegible]

10.2. Latency B

Credit return from follower to leader

