A²B Transceiver

Silicon Anomaly List — AD2437

ABOUT AD2437 SILICON ANOMALIES

These anomalies represent the currently known differences between revisions of the A²B® AD2437 product(s) and the functionality specified in the AD2437 data sheet(s) and the Technical Reference manual.

SILICON REVISIONS

A silicon revision number with the form "-x.x" is branded on all parts. The silicon revision can be electronically determined by reading the **A2B_VERSION** and **A2B_PRODUCT** registers.

Silicon Revision	A2B_VERSION[7:0]	A2B_PRODUCT[7:0]
2.1	0x21	0x37
1.3	0x13	0x37

ANOMALY LIST REVISION HISTORY

The following revision history lists the anomaly list revisions and major changes for each anomaly list revision.

Date	Anomaly List Revision	Data Sheet Revision	Additions and Changes
02/20/2023	B	PrB	Added Silicon Revision 2.1
07/05/2022	A	PrB	Initial Version

NOTE

1. Analog Devices is in the process of updating documentation to provide terminology and language that is culturally appropriate. This is a process with a wide scope and will be phased in as quickly as possible. Thank you for your patience.

One Analog Way, Wilmington, MA 01887 U.S.A.
Tel: 781.935.5565
Technical Support
www.analog.com

SUMMARY OF SILICON ANOMALIES

The following table provides a summary of AD2437 anomalies and the applicable silicon revision(s) for each anomaly.

No.	ID	Description	Rev 1.3	Rev 2.1
1	18000046	Pseudo-Random Binary Sequence Test Mode May Report Erroneous Bit Errors	x	x
2	18000047	ADR1/CLKOUT1 Or ADR2/CLKOUT2 Is Not Functional For Certain GPIO Modes When SPI Is Enabled	x	x
3	18000050	Upslot Data Corruption During Partial Rediscovery	x	x
4	18000060	A^2B Subordinate Node May Drive Free-running I^2C Signals When Exposed To Invalid I^2C Protocol	x	x
5	18000064	Mailbox Empty Interrupt Is Not Generated When The Local Processor Uses SPI To Read The Receive Mailbox Registers	x	.
6	18000065	Simultaneous Bus And Remote Peripheral Access May Lead To Access Corruption	x	.
7	18000070	An A^2B Node Operating In LVI Mode May Become Unresponsive	x	.

Key: x = anomaly exists in revision
. = Not applicable

DETAILED LIST OF SILICON ANOMALIES

The following list details all known silicon anomalies for the AD2437 including a description, workaround, and identification of applicable silicon revisions.

1. 18000046 - Pseudo-Random Binary Sequence Test Mode May Report Erroneous Bit Errors:

DESCRIPTION:

The Pseudo-Random Binary Sequence (PRBS) test mode may erroneously report bit errors under following conditions:

1. When using 32-bit slot size in upstream/downstream or both directions and if any **A2B_TXXBAR[0-27]** channel is mapped to TX Frame Buffer locations between 28-31.
2. When SPI Data Tunnel is enabled and if the data tunnel slots are present in Slot-0 and Slot-1. Additionally, the PRBS mode stops the SPI data tunnel access.

WORKAROUND:

1. Don't use TX Frame Buffer locations greater than 27 in **A2B_TXXBAR[0-27]**, when using PRBS with downstream/upstream or both data slot with 32-bit slot size.
2. If SPI data tunnel and PRBS mode are used in conjunction, the SPI Data Tunnel Downstream Offset **(A2B_DTDNOFFS.DTDNOFFS)** should be >=2.

APPLIES TO REVISION(S):

1.3, 2.1

2. 18000047 - ADR1/CLKOUT1 Or ADR2/CLKOUT2 Is Not Functional For Certain GPIO Modes When SPI Is Enabled:

DESCRIPTION:

ADR1/CLKOUT1 and ADR2/CLKOUT2 can be used to source the clock to the peripherals. If SPI is enabled, then any one of the ADR1/CLKOUT1 or ADR2/CLKOUT2 is not functional based on the GPIO mode set in **A2B_PINCFG.GPIOSEL**.

1. ADR1/CLKOUT1 is not functional when SPI is enabled with Non-Default GPIO mode (**A2B_PINCFG.GPIOSEL** = 2/3).
2. ADR2/CLKOUT2 is not functional when SPI is enabled with Default GPIO mode (**A2B_PINCFG.GPIOSEL** = 0).

The table below explains ADR1/CLKOUT1 and ADR2/CLKOUT2 availability based on SPI configuration and GPIO modes.

{ADR1, ADR2} Pin Functioning as CLKOUT[1,2]

SPI Configuration		GPIO Selection			
SPI Mode	Chip Select	GPIO Mode 0 (on SPI Pins)		GPIO Mode 2/3 (on I²C and Rx/Tx Pins)	
		CLKOUT1	CLKOUT2	CLKOUT1	CLKOUT2
SPI Slave	ADR1/$\overline{\text{SPISS}}$	N/A	x	N/A	ADR2
	SIO2/$\overline{\text{ASPISS}}$	x	x	x	ADR2
	ADR2/$\overline{\text{ASPISS}}$	x	N/A	x	N/A
SPI Master	ADR1/$\overline{\text{SPISSEL0}}$	N/A	x	N/A	ADR2
	SIO2/$\overline{\text{SPISSEL1}}$	ADR1	x	x	ADR2
	ADR2/$\overline{\text{SPISSEL2}}$	ADR1	N/A	x	N/A

[1] N/A denotes not applicable.

[2] x denotes not functional.

WORKAROUND:

Use any one of the workarounds listed below:

1. Use Default GPIO mode (**A2B_PINCFG.GPIOSEL** = 0) to source the clock to peripheral with ADR1/CLKOUT1 when SPI is enabled.
2. Use Non-Default GPIO mode (**A2B_PINCFG.GPIOSEL** = 2/3) to source the clock to peripheral with ADR2/CLKOUT2 when SPI is enabled.
3. Disable SPI (**A2B_SPICFG.SPIMODE** = 2) to use both ADR1/CLKOUT1 and ADR2/CLKOUT2.

APPLIES TO REVISION(S):

1.3, 2.1

3. 18000050 - Upslot Data Corruption During Partial Rediscovery:

DESCRIPTION:

When a subordinate node disconnects from the A²B bus during run-time, the host processor can attempt partial rediscovery of disconnected nodes without affecting data exchange between active upstream nodes. In the case where host processor initiates partial rediscovery of disconnected node and it results in a successful discovery, there would be data corruption in upslots contributed by other upstream nodes in the A²B network. This issue only occurs when disconnected nodes were contributing upslots to active nodes. The data corruption starts after successful rediscovery of the node and continues until the node is initialized.

For example, consider an A²B network *Main_node - Sub_node0 - Sub_node1*, with each subordinate node sending some upslots to *Main_node*. When *Sub_node1* disconnects from the network during run-time, the upslots of *Sub_node1* are replaced with previous known good samples and the communication with *Sub_node0* continues as-is without any data corruption. Now if the host processor attempts a partial rediscovery of *Sub_node1*, then upon successful rediscovery, there will be data corruption in the upslots contributed by *Sub_node0*, until the *Sub_node1* is completely initialized with its upslots settings.

WORKAROUND:

Use one of the workarounds listed below:

1. Once the subordinate node rediscovery is successful, the host processor must configure the **`A2B_LUPSLOTS, A2B_UPSLOTS, A2B_SLOTFMT, A2B_DATCTL, A2B_CONTROL.NEWSTRUCT`** registers of discovered node before configuring the remaining registers. This reduces the duration of upslots data corruption.
2. Before initiating the partial rediscovery, program the **`A2B_UPSLOT`** register of the last active upstream node to zero and remap the TDM channels of receiver nodes accordingly. In the example considered, the workaround involves clearing the **`A2B_UPSLOT`** register of *Sub_node0* and remapping the TDM channels of *Main_node*.
   Once there is a successful rediscovery and initialization of the dropped nodes, reconfigure the registers to their original value and revert back the TDM map of receiver node.

APPLIES TO REVISION(S):

1.3, 2.1

4. 18000060 - A²B Subordinate Node May Drive Free-running I²C Signals When Exposed To Invalid I²C Protocol:

DESCRIPTION:

The A²B subordinate node (when acting as a I²C controller accessing a remote peripheral) may toggle the I²C lines indefinitely when exposed to an invalid I²C protocol.
If the SCL line of an A²B subordinate node is spuriously pulled low, either by an external device or due to some event before the start of a remote I²C transaction from the subordinate node:
When a subordinate node receives a remote peripheral access request over the A²B bus, it checks the status of the local I²C bus. If the I²C bus is free, the A²B subordinate node starts the access; and if the I²C bus is busy, the subordinate node aborts the access. If the I²C bus is detected as busy (for example, due to glitch on the SCL line), but there is no valid I²C access on-going on the bus, then the A²B subordinate node will start driving the I²C signals continuously when the SCL line is released.

Invalid activity on the I²C bus causes this scenario. There is no issue, when the I²C bus is detected as busy due to proper I²C access ongoing on the bus.

There is no error generated or reported when this issue occurs, however one can observe that the remote peripheral accesses will always fail while the subordinate node register accesses will pass.

WORKAROUND:

Use one of the steps listed below to exit the error condition:

1. Generate a STOP bit on the I²C bus of the subordinate node.
2. Power cycle the A²B subordinate node.
3. Apply Soft Reset to subordinate node from Host processor.

APPLIES TO REVISION(S):

1.3, 2.1

5. 18000064 - Mailbox Empty Interrupt Is Not Generated When The Local Processor Uses SPI To Read The Receive Mailbox Registers:

DESCRIPTION:
If the local processor uses SPI to read the receive mailbox data registers, the mailbox full bit does not get cleared (**A2B_MBOX[n]STAT.MB[n]FULL** = 1). Therefore, the main node does not generate the mailbox empty interrupt (**A2B_MBOX[n]STAT.MB[n]EIRQ** = 0).

NOTE: This issue is applicable only for receive mailbox, for all mailbox lengths.

WORKAROUND:
Use any one of the workarounds listed below:
1. Use I²C to read the receive mailbox registers at subordinate node side.
2. To clear the mailbox full bit, switch the mailbox direction to transmit mailbox, clear the mailbox data and then switch back to receive mailbox mode. This procedure has been explained below with an example where **A2B_MBOX[n]** is used as receive mailbox with 4 bytes length. To receive more than 4 bytes via **A2B_MBOX[n]**:
   a. Enable GPIO over distance functionality between subordinate node and main node such that the local processor can signal host to send new payload to **A2B_MBOX[n]** after it has successfully read the last byte from the mailbox registers.
   b. To allow a new payload to be written to **A2B_MBOX[n]**, do the following writes from the host:
      i. Write 0x32 to subordinate node **A2B_MBOX[n]CTL** // Disable MBOX[n], turn off interrupts, set transmit mode
      ii. Write 0x33 to subordinate node **A2B_MBOX[n]CTL** // Re-enable in transmit mode
      iii. Read subordinate node **A2B_MBOX[n]B3** // Read last byte of MBOX[n] to make it empty
      iv. Write 0x3c to subordinate node **A2B_MBOX[n]CTL** // Disable MBOX[n], turn on interrupts, set receive mode
      v. Write 0x3d to subordinate node **A2B_MBOX[n]CTL** // Re-enable in receive mode
      vi. Write new payload to subordinate node **A2B_MBOX[n]**

APPLIES TO REVISION(S):
1.3

6. 18000065 - Simultaneous Bus And Remote Peripheral Access May Lead To Access Corruption:

DESCRIPTION:
Host processor can access the A²B subordinate node and its peripherals either via I²C or SPI. If there are concurrent bus accesses to a subordinate node registers and its I²C remote peripheral, it may lead to one of the following erroneous behaviors:
1. Bus access to Subordinate node registers via I²C may get corrupted when there is a simultaneous remote peripheral access via SPI or vice-versa.
2. **A2B_INTTYPE** may get corrupted when a subordinate node raises interrupt while I²C remote peripheral access via SPI is in progress.
3. For an AD242x subordinate node, remote peripheral access, Interrupt data **(A2B_INTTYPE)** or bus access may get corrupted.

WORKAROUND:
Do not use I²C and SPI concurrently for accessing subordinate node registers and remote peripherals. It is also essential to disable the subordinate node interrupt reporting at main node by clearing **A2B_INTMSK2.SLVIRQEN** bit when accessing I²C remote peripheral via SPI.

APPLIES TO REVISION(S):
1.3

7. 18000070 - An A²B Node Operating In LVI Mode May Become Unresponsive:

DESCRIPTION:
An A²B transceiver operating in LVI mode may become unresponsive at elevated VIN which is within the datasheet operating condition range.

WORKAROUND:
Use any of the following workarounds:
1. Use Non-LVI Mode.
2. In LVI Mode, use VIN = 3.17V to 3.298V.

APPLIES TO REVISION(S):
1.3

This page intentionally left blank.