Chapter B1
AMBA AXI4-Lite

This chapter defines the AXI4-Lite interface and associated protocol. AXI4-Lite is suitable for simpler control register-style interfaces that do not require the full functionality of AXI4.

This chapter contains the following sections:

B1.1 Definition of AXI4-Lite

This section defines the functionality and signal requirements of AXI4-Lite components.

The key functionality of AXI4-Lite operation is:

- all transactions are of burst length 1
- all data accesses use the full width of the data bus
  - AXI4-Lite supports a data bus width of 32-bit or 64-bit.
- all accesses are Non-modifiable, Non-bufferable
- Exclusive accesses are not supported.

B1.1.1 Signal list

Table B1-1 shows the required signals on an AXI4-Lite interface.

Table B1-1 AXI4-Lite interface signals

Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

AXI4 signals modified in AXI4-Lite

The AXI4-Lite interface does not fully support the following signals:

RRESP, BRESP

The EXOKAY response is not supported on the read data and write response channels.

AXI4 signals not supported in AXI4-Lite

The AXI4-Lite interface does not support the following signals:

AWLEN, ARLEN The burst length is defined to be 1, equivalent to an **AxLEN** value of zero.

AWSIZE, ARSIZE All accesses are defined to be the width of the data bus.

Note

AXI4-Lite requires a fixed data bus width of either 32-bit or 64-bit.

AWBURST, ARBURST

The burst type has no meaning because the burst length is 1.

AWLOCK, ARLOCK

All accesses are defined as Normal accesses, equivalent to an **AxLOCK** value of zero.

AWCACHE, ARCACHE

All accesses are defined as Non-modifiable, Non-bufferable, equivalent to an **AxCACHE** value of `0b0000`.

WLAST, RLAST All bursts are defined to be of length 1, equivalent to a **WLAST** or **RLAST** value of 1.

B1.1.2 Bus width

AXI4-Lite has a fixed data bus width and all transactions are the same width as the data bus. The data bus width must be, either 32-bits or 64-bits.

ARM expects that:

- the majority of components use a 32-bit interface
- only components requiring 64-bit atomic accesses use a 64-bit interface.

A 64-bit component can be designed for access by 32-bit masters, but the implementation must ensure that the component sees all transactions as 64-bit transactions.

Note

This interoperability can be achieved by including, in the register map of the component, locations that are suitable for access by a 32-bit master. Typically, such locations would use only the lower 32-bits of the data bus.

B1.1.3 Write strobes

The AXI4-Lite protocol supports write strobes. This means multi-sized registers can be implemented and also supports memory structures that require support for 8-bit and 16-bit accesses.

All master interfaces and interconnect components must provide correct write strobes.

Any slave component can choose whether to use the write strobes. The options permitted are:

- to make full use of the write strobes
- to ignore the write strobes and treat all write accesses as being the full data bus width
- to detect write strobe combinations that are not supported and provide an error response.

A slave that provides memory access must fully support write strobes. Other slaves in the memory map might support a more limited write strobe option.

When converting from full AXI to AXI4-Lite, a write transaction can be generated on AXI4-Lite with all write strobes deasserted. Automatic suppression of such transactions is permitted but not required. See *Conversion, protection, and detection* on page B1-131.

B1.1.4 Optional signaling

AXI4-Lite supports multiple outstanding transactions, but a slave can restrict this by the appropriate use of the handshake signals.

AXI4-Lite does not support AXI IDs. This means all transactions must be in order, and all accesses use a single fixed ID value.

Note

Optionally, an AXI4-Lite slave can support AXI ID signals, so that it can be connected to a full AXI interface without modification. See *Interoperability* on page B1-128.

AXI4-Lite does not support data interleaving, the burst length is defined as 1.

B1.2 Interoperability

This section describes the interoperability of AXI and AXI4-Lite masters and slaves. Table B1-2 shows the possible combinations of interface, and indicates that the only case requiring special consideration is an AXI master connecting to an AXI4-Lite slave.

Table B1-2 Full AXI and AXI4-Lite interoperability

Master	Slave	Interoperability
AXI	AXI	Fully operational.
AXI	AXI4-Lite	AXI ID reflection is required. Conversion might be required.
AXI4-Lite	AXI	Fully operational.
AXI4-Lite	AXI4-Lite	Fully operational.

B1.2.1 Bridge requirements of AXI4-Lite slaves

As Table B1-2 shows, the only interoperability case that requires special consideration is the connection of an AXI4-Lite slave interface to a full AXI master interface.

This connection requires AXI ID reflection. The AXI4-Lite slave must return the AXI ID associated with the address of a transaction with the read data or write response for that transaction. This is required because the master requires the returning ID to correctly identify the transaction response.

If an implementation cannot ensure that the AXI master interface only generates transactions in the AXI4-Lite subset, then some form of adaptation is required. See *Conversion, protection, and detection* on page B1-131.

B1.2.2 Direct connection requirements of AXI4-Lite slaves

An AXI4-Lite slave can be designed to include ID reflection logic. This means the slave can be used directly on a full AXI connection, without a bridge function, in a system that guarantees that the slave is accessed only by transactions that comply with the AXI4-Lite subset.

Note

This specification recommends that the ID reflection logic uses **AWID**, instead of **WID**, to ensure compatibility with both AXI3 and AXI4.

B1.3 Defined conversion mechanism

This section defines the requirements to convert any legal AXI transaction for use on an AXI4-Lite component. *Conversion, protection, and detection* on page B1-131 discusses the advantages and disadvantages of the various approaches that can be used.

B1.3.1 Conversion rules

Conversion requires that the AXI data width is equal to or greater than the AXI4-Lite data width. If this is not the case then the AXI data width must first be converted to the AXI4-Lite data width.

Note

AXI4-Lite does not support EXOKAY responses, so the conversion rules do not consider this response.

The rules for conversion from a full AXI interface are as follows:

- If a transaction has a burst length greater than 1 then the burst is broken into multiple transactions of burst length 1. The number of transactions that are created depends on the burst length of the original transaction.
- When generating the address for subsequent beats of a burst, the conversion of bursts with a length greater than 1 must take into consideration the burst type. An unaligned start address must be incremented and aligned for subsequent beats of an INCR or WRAP burst. For a FIXED burst the same address is used for all beats.
- Where a write burst with length greater than 1 is converted into multiple write transactions, the component responsible for the conversion must combine the responses for all of the generated transactions, to produce a single response for the original burst. Any error response is sticky. That is, an error response received for any of the generated transactions is retained, and the single combined response indicates an error. If both a SLVERR and a DECERR are received then the first response received is the one that is used for the combined response.
- A transaction that is wider than the destination AXI4-Lite interface is broken into multiple transactions of the same width as the AXI4-Lite interface. For transactions with an unaligned start address, the breaking up of the burst occurs on boundaries that are aligned to the width of the AXI4-Lite interface.
- Where a wide transaction is converted to multiple narrower transactions, the component responsible for the conversion must combine the responses to all of the narrower transactions, to produce a single response for the original transaction. Any error response is sticky. If both a SLVERR and a DECERR are received then the first response received is used for the combined response.
- Transactions that are narrower than the AXI4-Lite interface are passed directly and are not converted.
- Write strobes are passed directly, unmodified.
- Write transactions with no strobes are passed directly.

  Note

  The AXI4-Lite protocol does not require these transactions to be suppressed.

- The **AxLOCK** signals are discarded for all transactions. For a sequence of locked transactions any lock guarantee is lost. However, the locked nature of the transaction is lost only at any downstream arbitration. For an exclusive sequence, the AXI signaling requirements mean any exclusive write access must fail.
- The **AxCACHE** signals are discarded. All transactions are treated as Non-modifiable and Non-bufferable.

  Note

  This is acceptable because AXI permits Modifiable accesses to be treated as Non-modifiable, and Bufferable accesses to be treated as Non-bufferable.

- The **AxPROT** signals are passed directly, unmodified.

- The **WLAST** signal is discarded.
- The **RLAST** signal is not required, and is considered asserted for every transfer on the read data channel.

B1.4 Conversion, protection, and detection

Connection of an AXI4-Lite slave to an AXI4 master requires some form of adaptation if it can not be ensured that the master only issues transactions that meet the AXI4-Lite requirements.

This section describes techniques that can be adopted in a system design to aid with the interoperability of components and the debugging of system design problems. These techniques are:

Conversion This requires the conversion of all transactions to a format that is compatible with the AXI4-Lite requirements.

Protection This requires the detection of any non-compliant transaction. The non-compliant transaction is discarded, and an error response is returned to the master that generated the transaction.

Detection This requires observing any transaction that falls outside the AXI4-Lite requirements and:

- notifying the controlling software of the unexpected access
- permitting the access to proceed at the hardware interface level.

B1.4.1 Conversion and protection levels

Different levels of conversion and protection can be implemented:

Full conversion

This converts all AXI transactions, as described in *Defined conversion mechanism* on page B1-129.

Simple conversion with protection

This propagates transactions that only require a simple conversion, but suppresses and error reports transactions that require a more complex task.

Examples of transactions that are propagated are the discarding of one or more of **AxLOCK** and **AxCACHE**.

Examples of transactions that are discarded and generate an error report are burst length or data width conversions.

Full protection

Suppress and generate an error for every transaction that does not comply with the AXI4-Lite requirements.

B1.4.2 Implementation considerations

A protection mechanism that discards transactions must provide a protocol-compliant error response to prevent deadlock. For example, in the full AXI protocol, read burst transactions require an error for each beat of the burst and a correctly asserted **RLAST** signal.

Using a combination of detection and conversion permits hardware implementations that:

- do not prevent unexpected accesses from occurring
- provide a mechanism for notifying the controlling software of the unexpected access, so speeding up the debug process.

In complex designs, the advantage of combining conversion and detection is that unforeseen future usage can be supported. For example, at design time it might be considered that only the processor programs the control register of a peripheral, but in practice, the peripheral might need to be programmed by other devices, for example a DSP or a DMA controller, that cannot generate exactly the required AXI4-Lite access.

The advantages and disadvantages of the different approaches are:

- Protection requires a lower gate count.
- Conversion ensures the interface can operate with unforeseen accesses.
- Conversion increases the portability of software from one system to another.

- Conversion might provide more efficient use of the AXI infrastructure. For example, a burst of writes to a FIFO can be issued as a single burst, rather than needing to be issued as a set of single transactions.
- Conversion might provide more efficient use of narrow links, where the address and data payload signals are shared.
- Conversion might provide more flexibility in components that can be placed on AXI4-Lite interfaces. By converting bursts and permitting sparse strobes, memory can be placed on AXI4-Lite, with no burst conversion required in the memory device. This is, essentially, a sharing of the burst conversion logic.