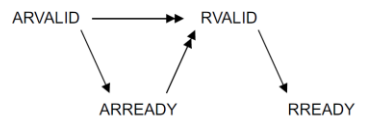
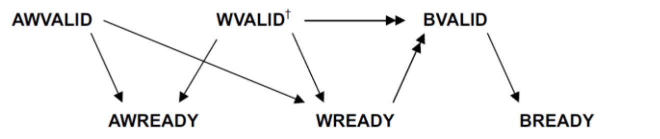
AXI Signal Cheatsheet

Channels (Fundamental Signals)

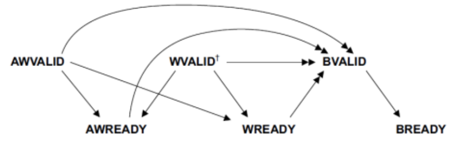
➔ Address Read	**ARID**, **ARADDR –** Address information **ARVALID**, **ARREADY** – Control handshake **ARBURST, ARLEN, ARSIZE –** Burst Type, Number(LEN) of transfer in burst, Size of each transfer **ARLOCK, ARCACHE, ARPROT** – Priority, Modifiable, Security properties of access
➔ Address Write	**AWID**, **AWADDR –** Address information **AWVALID**, **AWREADY -** Control handshake **AWBURST, AWLEN, AWSIZE –** Burst Type, Number(LEN) of transfer in burst, Size of each transfer **AWLOCK, AWCACHE, AWPROT** – Priority, Modifiable, Security properties of access
➔ Read Data	**RVALID**, **RREADY**, **RLAST** handshake **RID, RDATA –** Transaction **RRESP** – Completion of *txⁿ*
➔ Write Data	**WVALID**, **WREADY**, **WLAST** handshake **WSTRB** – Per 8bit(Byte) to indicate which bytes are valid Write Data Channel information treated as buffered – Master can write without ack for prev. *txⁿ* **WID**, **WDATA –** Transaction
🡐 Write Response	**BID, BVALID, BRESP, BREADY –** Write response after last data transfer of transaction



Read transaction handshake dependencies



† Dependencies on the assertion of **WVALID** also require the assertion of **WLAST**



† Dependencies on the assertion of **WVALID** also require the assertion of **WLAST**

Burst Type	Address for each beat in burst	Usecase
FIXED	Same address	FIFO Access
INCR	Increments by size of transfer	Sequential Memory
WRAP	Similar to INCR with wrap around	Cache Line Access

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

AxCACHE	Value	Transaction attribute
[0]	0	Non-bufferable
	1	Bufferable
[1]	0	Non-cacheable
	1	Cacheable
[2]	0	No Read-allocate
	1	Read-allocate
[3]	0	No Write-allocate
	1	Write-allocate

AxPROT	Value	Function
[0]	0	Unprivileged access
	1	Privileged access
[1]	0	Secure access
	1	Non-secure access
[2]	0	Data access
	1	Instruction access

AxLOCK[1:0]	Access type
0b00	Normal access
0b01	Exclusive access
0b10	Locked access
0b11	Reserved

AXI3/4 Signals quick reference

Write Signal	Master (Default Value)	Slave (Default Value)	Read Signal
🟦 AWID	🟧 All zeros	🟩 -	🟦 ARID
🟦 AWADDR	🟩 -	🟩 -	🟦 ARADDR
🟦 AWREGION*	🟧 All zeros	🟧 -	🟦 ARREGION*
🟦 AWLEN	🟧 All zeros (Single transfer)	🟩 -	🟦 ARLEN
🟦 AWSIZE	🟧 Data bus width	🟩 -	🟦 ARSIZE
🟦 AWBURST	🟧 b01, INCR	🟩 -	🟦 ARBURST
🟦 AWLOCK	🟧 All zeros, Normal Access	🟧 -	🟦 ARLOCK
🟦 AWCACHE	🟧 All zeros	🟧 -	🟦 ARCACHE
🟦 AWPROT	🟩 -	🟧 -	🟦 ARPROT
🟦 AWQoS*	🟧 All zeros	🟧 -	🟦 ARQoS*
🟦 AWVALID	🟩 -	🟩 -	🟦 ARVALID
🟨 AWREADY	🟩 -	🟩 -	🟨 ARREADY
🟦 WID**	🟧 -	🟧 -	
🟦 WDATA	🟩 -	🟩 -	🟨 RDATA
🟦 WSTRB	🟧 All ones	🟩 -	
🟦 WLAST	🟩 -	🟧 -	🟨 RLAST***
🟦 WVALID	🟩 -	🟩 -	🟨 RVALID
🟨 WREADY	🟩 -	🟩 -	🟦 RREADY
🟨 BID	🟧 -	🟩 -	🟨 RID
🟨 BRESP	🟧 -	🟧 b00, OKAY	🟨 RRESP
🟨 BVALID	🟩 -	🟩 -	
🟦 BREADY	🟩 -	🟩 -	

* indicates AXI 4 Only ** indicates AXI3 only

*** Swap requirement for read since slave initiates

🟩 Required

🟧 Optional

🟦 Source Master

🟨 Source Slave

ACLK, ARESETn, Low Power Signals are not shown in the above table

AXI4 specific	ALEN* , ALOCK* changes with AXI4
	USER* signals on each channel are not specified since AXI specification does not define the function of these signals

AXI Conceptual Quick Reference

Register Slices

Each AXI channel transfers information in **only one direction**, and the architecture **does not** require any fixed relationship between the channels except for:

- a write response must always follow the last write transfer in the *tx^n*.
- read data must always follow the address to which the data relates.

This means, for example, that the write data can appear at an interface before the write address for the transaction. This can occur if the write address channel contains more register stages than the write data channel. Similarly, the write data might appear in the same cycle as the address. This allows register slice insertion at almost any point in any channel, at the cost of an additional cycle of latency. (Tradeoff b/w max. freq. & latency)

Transactions

When a master initiates a transfer to slave, the set of operations on the bus forms the **AXI Transaction** which contains the required payload data transferred as an **AXI Burst** which can comprise of multiple data transfers called **AXI Beats**.

The master begins each burst by driving control information and the address of the first byte in the transaction to the slave. As the burst progresses, the slave must calculate the addresses of subsequent transfers in the burst.

AXI supports unaligned (using strobe), burst-based outstanding and out-of-order transactions.

Operation

- Clock: On master and slave interfaces there must be no combinatorial paths between input and output signals.
- All **VALID** signals **should** be driven low on reset. Other signals can take any value.
- Transfer occurs only when both the **VALID** and **READY** signals are HIGH.
- Cache signals control:
    - how a transaction progresses through the system
    - how system-level caches handle the transaction.

IP Recommendations

- Does not recommend a default **AWREADY** state of LOW, because it forces the transfer to take at least two cycles, one to assert **AWVALID** and another to assert **AWREADY**.
- Any AXI3 component that requires a WID signal can generate this from the AWID value. (Legacy consideration)
- For example, a master must not wait for **AWREADY** to be asserted before driving **WVALID**. A deadlock condition can occur if the slave is waiting for **WVALID** before asserting **AWREADY**.
- When accessing a data-sensitive device like a FIFO, a master must use a burst length that exactly matches the size of the required data transfer.
- While it is acceptable to wait for VALID to be asserted before asserting READY, it is also acceptable to assert READY before detecting the corresponding VALID. This can result in a more efficient design

Bursts

- For wrapping bursts,
    - the burst length must be 2, 4, 8, or 16
    - start address must be aligned to size of each transfer.
- A burst must not cross a 4KB address boundary.
- Early termination of bursts is not supported
- The size of any transfer must not exceed the data bus width of either agent in the transaction
- When a master generates a transfer that is narrower than its data bus, the address and control information determine which byte lanes the transfer uses:
    - in incrementing or wrapping bursts, different byte lanes are used on each beat of the burst
    - in a fixed burst, the same byte lanes are used on each beat.
- Reads have response for every transfer in burst but write has a single response for entire burst.

Memory Attribute Signaling ACACHE

- Bufferable [0] – For writes, Assertion of this bit means that interconnect/other component can delay transaction reaching slave by any no. of cycles
- Cacheable [1] - The characteristics of a tx^n at destination does not have to match the original tx^n. For writes this means that several different writes can be merged together. For reads this means that the contents of a location can be prefetched, or the values from a single fetch can be used for multiple read tx^n
- Read Allocate (RA) [2], Write Allocate (WA) [3] - When this bit is asserted, r/w allocation of the transaction is recommended but is not mandatory. The R/WA bit must not be asserted if the C bit is deasserted.

Formulae

- Size of $tx^n = 2^{\text{ARSIZE}} \times (\text{ARLEN} + 1)$
  This is also the wrap boundary.

Interconnect

- **Memory Slave** should handle all types of tx^n. **Peripheral Slave** can restrict support to a few tx^n types. (Lower signal count)
- When the interconnect is required to determine the destination address space or slave space, it must realign the address and write data. This is required to assure that the write data is signaled as valid only to the slave for which it is destined
- In AXI4, all Device transactions using the same ID to the same slave must be ordered with respect to each other**.** ARM believes that most implemented AXI3 interconnects support the required AXI4 behavior.
- When a master is connected to an interconnect, the interconnect appends additional bits to the **ARID**, **AWID** and **WID** that are unique to that master port. Response from slave are directed to the master after removing these additional bits.
- A master must not start the write part of an exclusive access (AxLOCK)sequence until the read part is complete
- When a master uses the AxLOCK signals for a transaction to show that it is a locked transaction then the interconnect must ensure that only that master can access the targeted slave region, until an unlocked transaction from the same master completes. An arbiter within the interconnect must enforce this restriction.
- Before a master starts a locked sequence of either read or write transactions it must ensure that it has no other transactions waiting to complete.

AXI4 Specific changes (AXI4)

- AQoS, AREGION have been added
- ALEN*, ALOCK* definitions change
- AXI4 makes the following changes to the AXI3 memory attribute signaling:
    - the AxCACHE[1] bits are renamed as the Modifiable bits
    - ordering requirements are defined for Non-modifiable transactions
    - the meanings of Read-allocate and Write-allocate are updated.

  Write Response: The slave must wait for **AWVALID**, **AWREADY**, **WVALID**, and **WREADY** to be asserted before asserting BVALID the slave must also wait for WLAST to be asserted before asserting BVALID, because the write response, BRESP must be signaled only after the last data transfer of a write transaction.
- The additional dependency above means that an AXI3 slave that accepts all write data and provides a write response before accepting the address is not compliant with AXI4 (By issuing a write response, the slave takes responsibility for hazard checking the write transaction against all subsequent transactions).Converting an AXI3 legacy slave to AXI4 requires the addition of a wrapper that ensures a returning write response is not provided until the appropriate address has been accepted by the slave. Any AXI3 master complies with the AXI4 write response requirements.
- AXI4 removes support for write data interleaving. In AXI4, all the write data for a transaction must be provided in consecutive transfers on the write data channel.
- Introduces the concept of single-copy atomicity size.
- AXI4 removes support for locked transactions because of the complexity of interconnect & ability to make a QoS guarantee