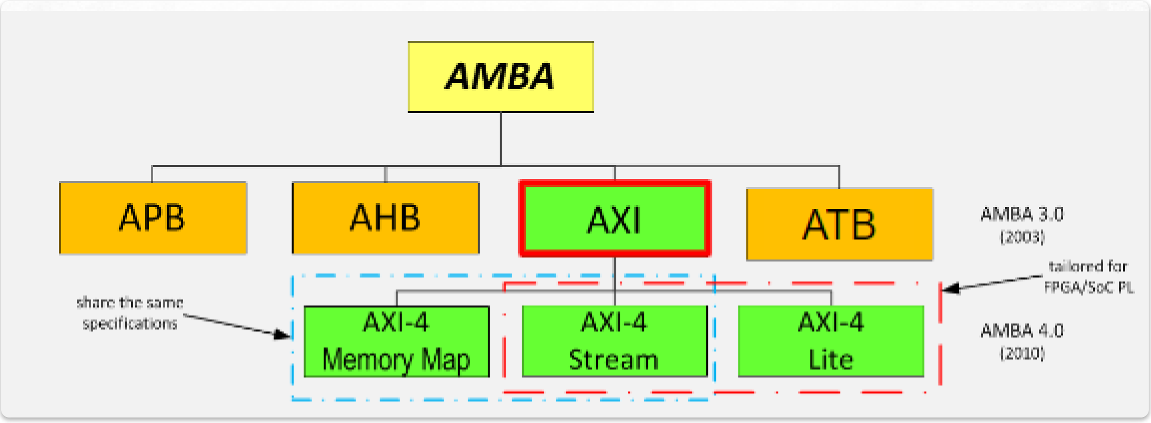
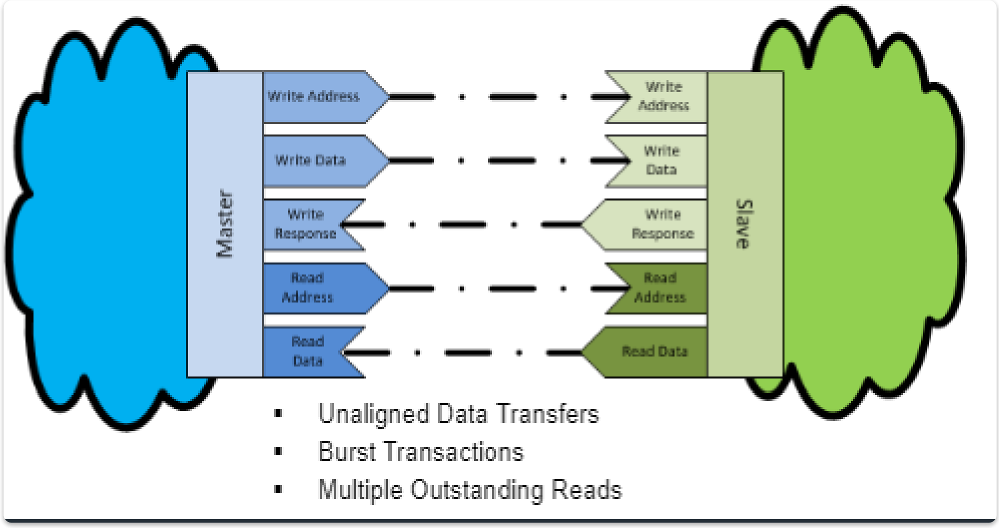
AXI:Introduction

Advance Extensible Interface (AXI) is part of AMBA (Advanced Microcontroller Bus Architecture), se figure below.
The APB, AHB, ATB bus architectures are used for internal ARM core topologies. AXI is targeted towards processor peripheral and other IP. It is the primary mechanism for moving data in out and through the PL.



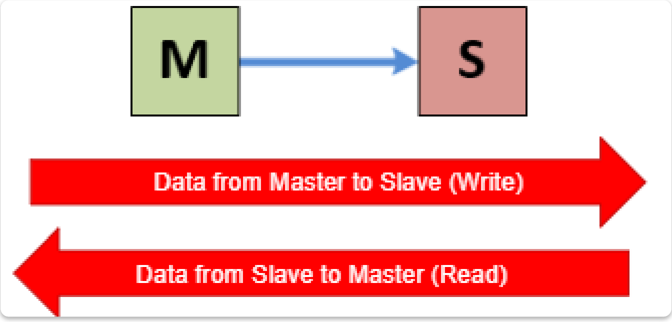
AXI Definition: It is a point-to-point connection. As it is point to point, there is NEVER any bus arbitration. The interface is made of separate address, control and data signals. This interface support, unaligned data transfer, burst transaction and multiple reads. The main goal of AXI technology is to achieve a unified interface protocol for all Xilinx IP, even those that are not related to embedded systems.

From the figure it can be seen that there are two channels of data, both are unidirectional regarding data transfer, one moves data from master to slave and one from slave to master. The protocol possess 5 channels (each of several bits), grouped into read and write transactions, which can be done simultaneously. Read and write channels are non-posted, it means that there is always a response as confirmation of the transaction. In the write operation there is an acknowledgment bus, in the read operation, the read data is used as response.



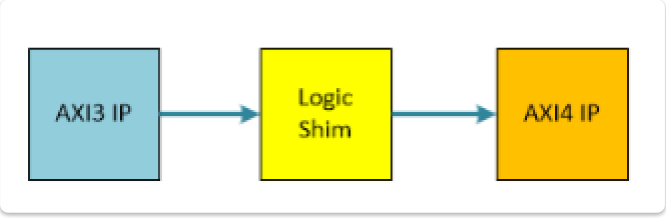
AXI Implementation: AXI define the signals that have to be connected between two points, and a protocol which is in an upper layer to those signals. However, it does not define any physical definition such as voltages.

Master/Slave Relationship: A master is any device that can initiate an AXI transaction (read or write). Slave, can never start a transaction. Arrows symbols of connection between master and slave indicates master-slave relation and NO the data direction.

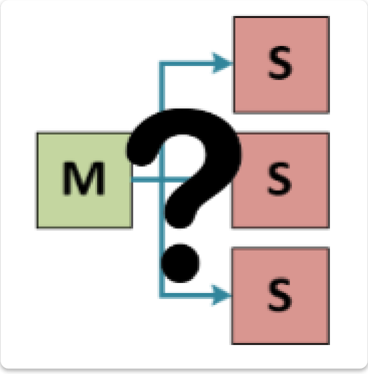

M
S
Data from Master to Slave (Write)
Data from Slave to Master (Read)

AXI: Shims and Interconnect IP:

A shim is basically a piece of logic that translate the logic between different versions of AXI. The Zynq 7000 PS was built with AXI3 IP standard, which xilinx IP is AXI 4 compliant. To connect to these two devices, based on different AXI IP a shim is needed in between. The shim is transparent for the user, as the tool automatically introduce it. The MPSoC and RFSoC devices are built based on AXI4 standard and no shim is needed for connection of peripheral



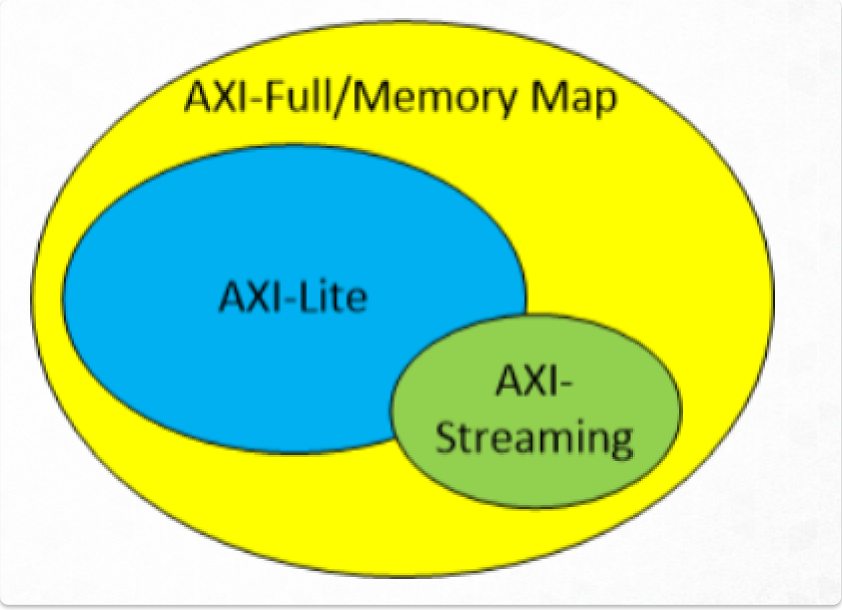
As AXI is point-to-point , How a master talk to several slaves?.



For that, the AXI Interconnect IP is used. It is a piece of IP resides in PL, that basically takes its input from a single AXI slave and maps it to one or more AXI masters, creating a AXI switch.

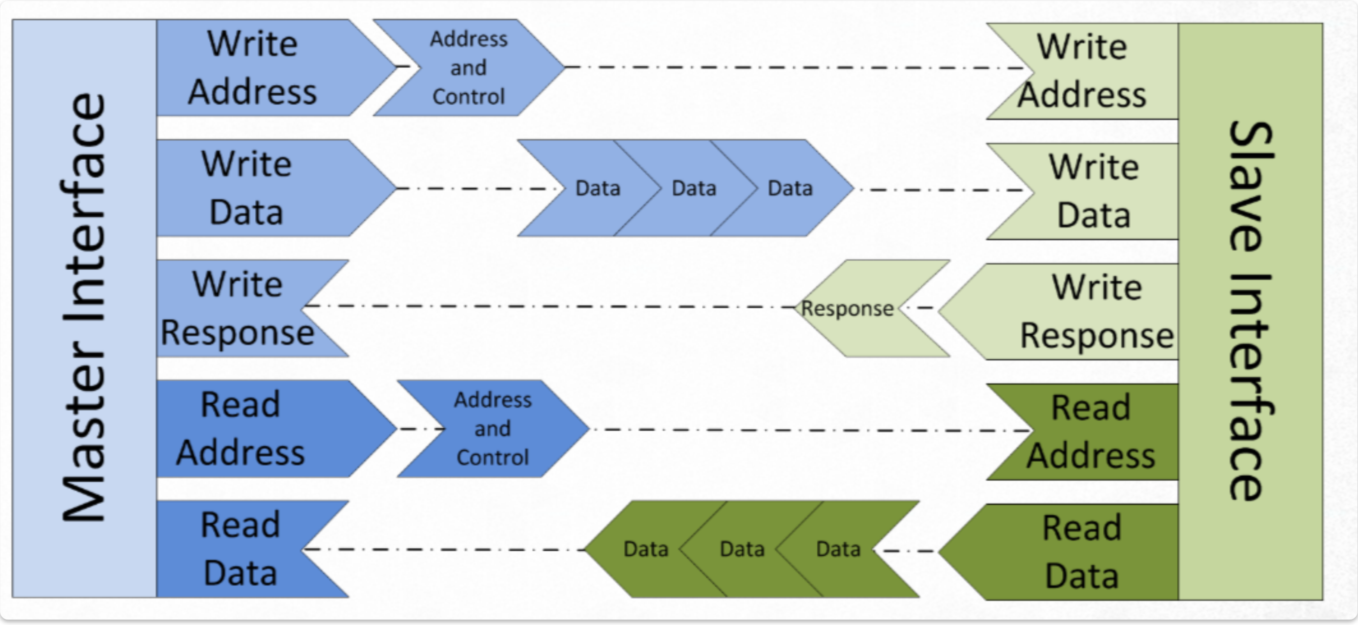
AXI Configuration

There are three possible AXI operation modes.

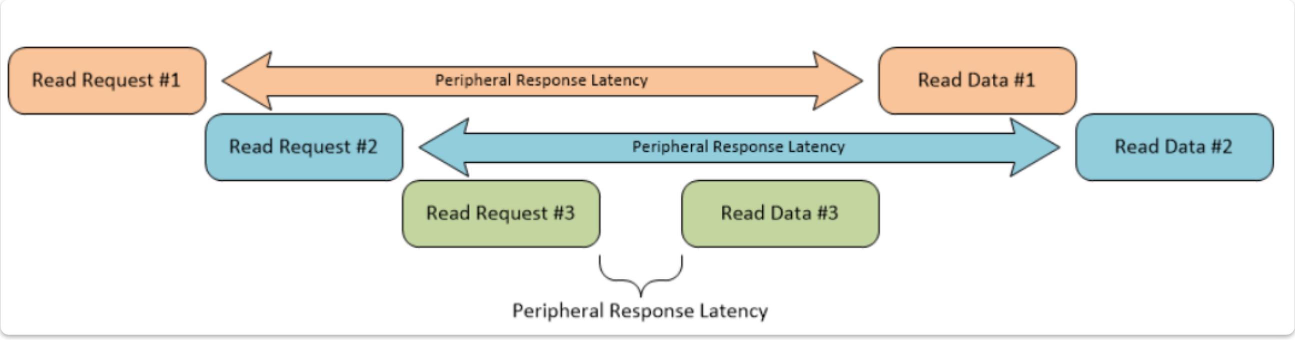


Please read sections: AXI protocol overview and Channel transfers and transactions of this documentation.

AXI-4 Full or AXI-4 Memory Map: It uses the full set of signals and bus widths and point-to-point address based data path. It is the base definition for AXI. It can be considered as the superset of all configurations. as the other are same but with restrictions in respect to AXI-4 full. AXI4-Full consist of five independent channels. Two are for read transactions and three for write transaction. Read and write operations contain independent address channels and data channels.

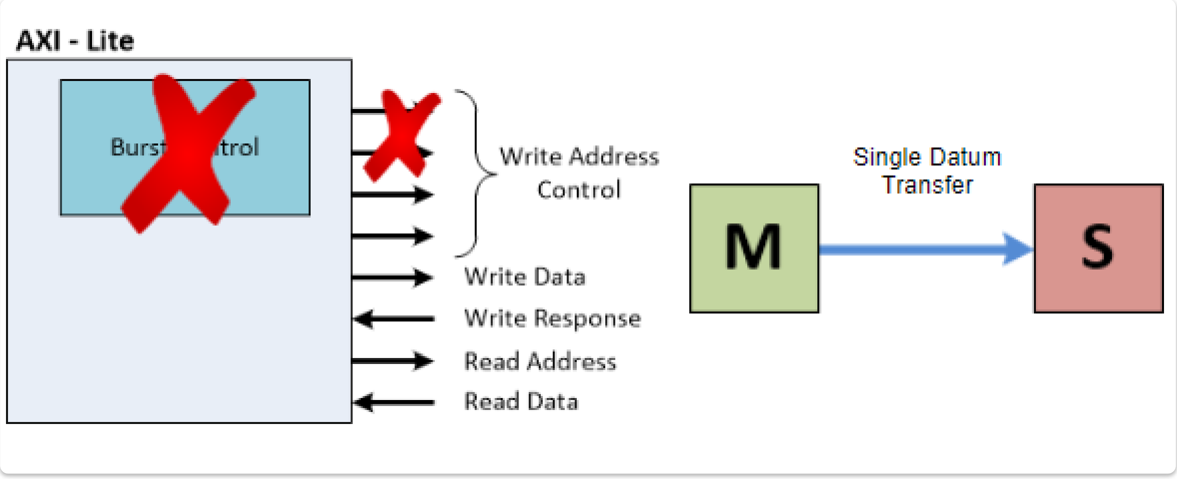


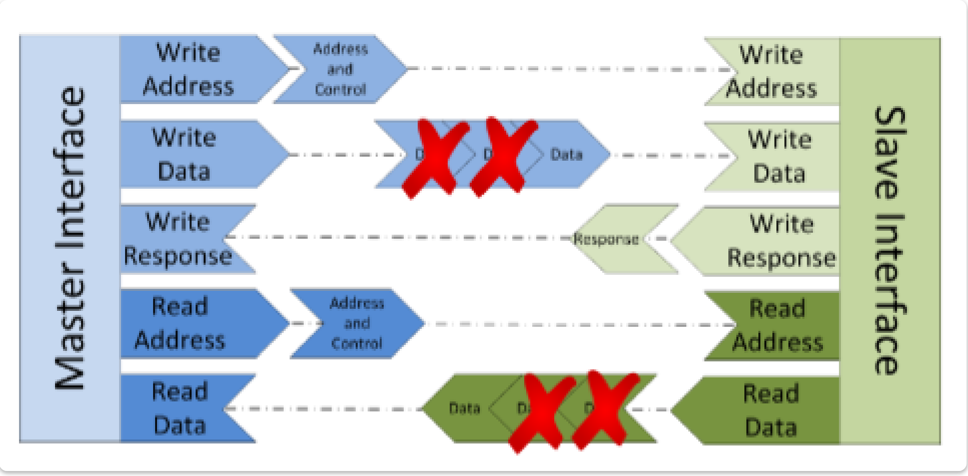
- Data burst are supported when a set of data is read or write, incrementing automatically the address., reducing overhead and improving communication efficiency.
- Data can be transmitted as individual address/data pairs or as
- Data can be transmitted as Single-address/multiple-data. Supporting up to 256 data beats (or data burst) per address. Reducing the overhead time (time used for all other protocol task besides sending data) to the same time as sending one data.

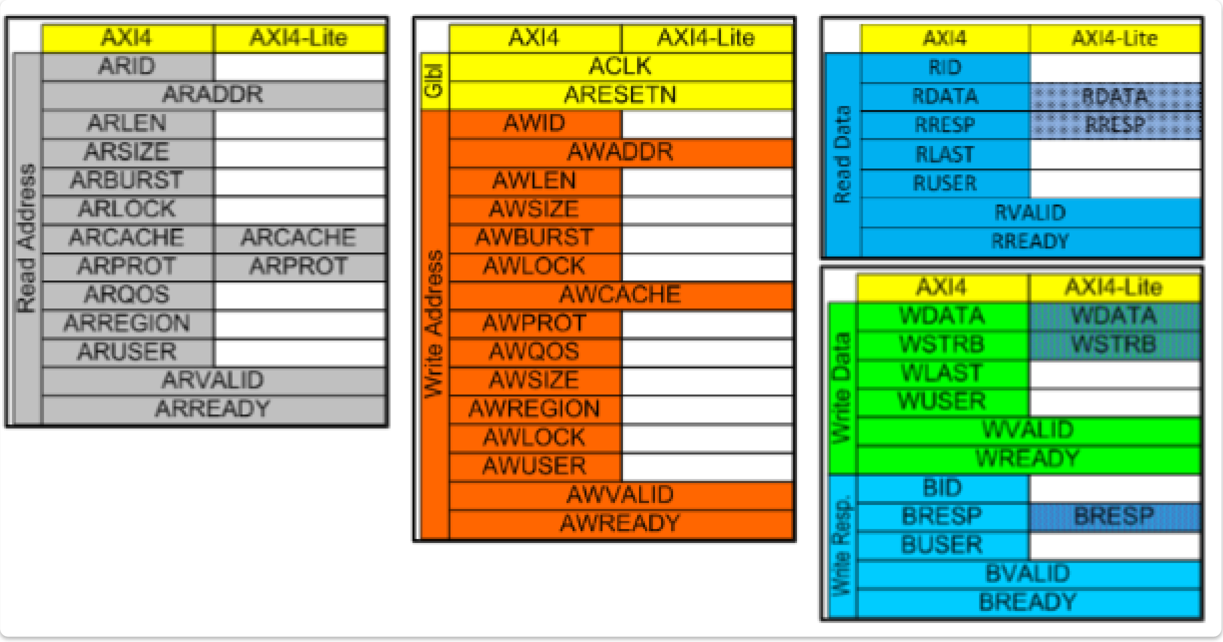
- Data width is parametrizable from 32 to 1024 in power-of-2 increment) However, in real implementation it is limited to 256 bits, as moving more than 256 reduces timing solutions, being better to move more data with smaller size than large pieces of data.
- Overlapped transactions are allowed. Thereby, we can receive data in different order to the requested data depending on the peripherals latency, using time more efficiently.
- 
Read Request #1
Peripheral Response Latency
Read Data #1
Read Request #2
Peripheral Response Latency
Read Data #2
Read Request #3
Read Data #3
Peripheral Response Latency

AXI-4 Lite: Very similar to memory map, but simplified to reduce PL requirements. The configuration eliminates the burst capability, sending only one data at a time. This is typically used when a peripheral has to be configured.

- Its main purpose is to enable the processors to *communicate control signals and messages* with peripheral. It is not useful for moving data, it is for instructions and configuration.
- Implementation widths are 23-bits or 64-bits.
- All access are non-modifable, non-bufferable and non-exclusive.
- Below is a detailed comparison of differences in signals between AXi-Full and -Lite.







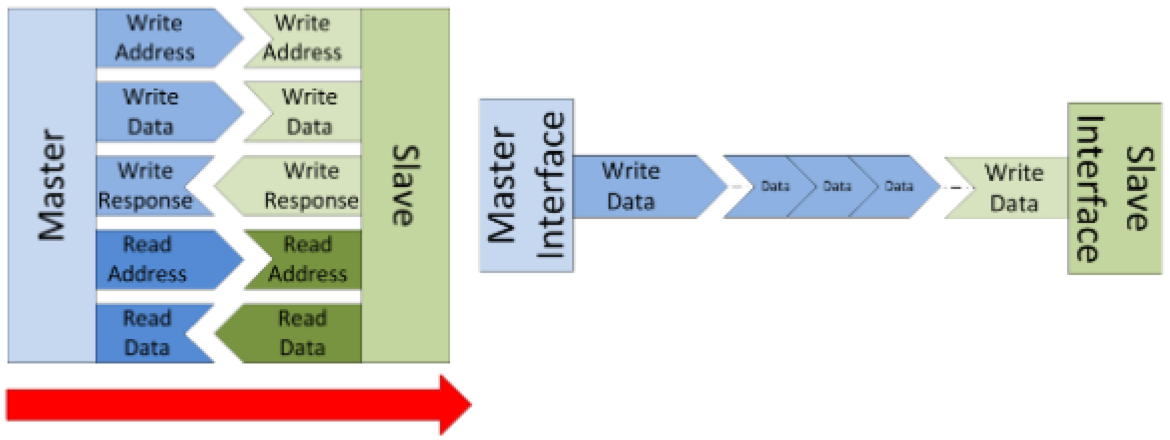
	AXI4	AXI4-Lite
Read Address	ARID	
	ARADDR	ARADDR
	ARLEN	
	ARSIZE	
	ARBURST	
	ARLOCK	
	ARCACHE	ARCACHE
	ARPROT	ARPROT
	ARQOS	
	ARREGION	
	ARUSER	
	ARVALID	ARVALID
	ARREADY	ARREADY

	AXI4	AXI4-Lite
Glbl	ACLK	ACLK
	ARESETN	ARESETN
Write Address	AWID	
	AWADDR	AWADDR
	AWLEN	
	AWSIZE	
	AWBURST	
	AWLOCK	
	AWCACHE	AWCACHE
	AWPROT	
	AWQOS	
	AWSIZE	
	AWREGION	
	AWLOCK	
	AWUSER	
	AWVALID	AWVALID
	AWREADY	AWREADY

	AXI4	AXI4-Lite
Read Data	RID	
	RDATA	RDATA
	RRESP	RRESP
	RLAST	
	RUSER	
	RVALID	RVALID
	RREADY	RREADY

	AXI4	AXI4-Lite
Write Data	WDATA	WDATA
	WSTRB	WSTRB
	WLAST	
	WUSER	
	WVALID	WVALID
	WREADY	WREADY
Write Resp.	BID	
	BRESP	BRESP
	BUSER	
	BVALID	BVALID
	BREADY	BREADY

AXI-4 Stream: Unlike AXI4 Memory Map, AXI-4 Stream send the data burst eliminating the address phase to reduce the timing overhead. It is used typically for video or audio. The protocol is the same as AXI-4. Sideband signals are used to indicate start and end of transaction.



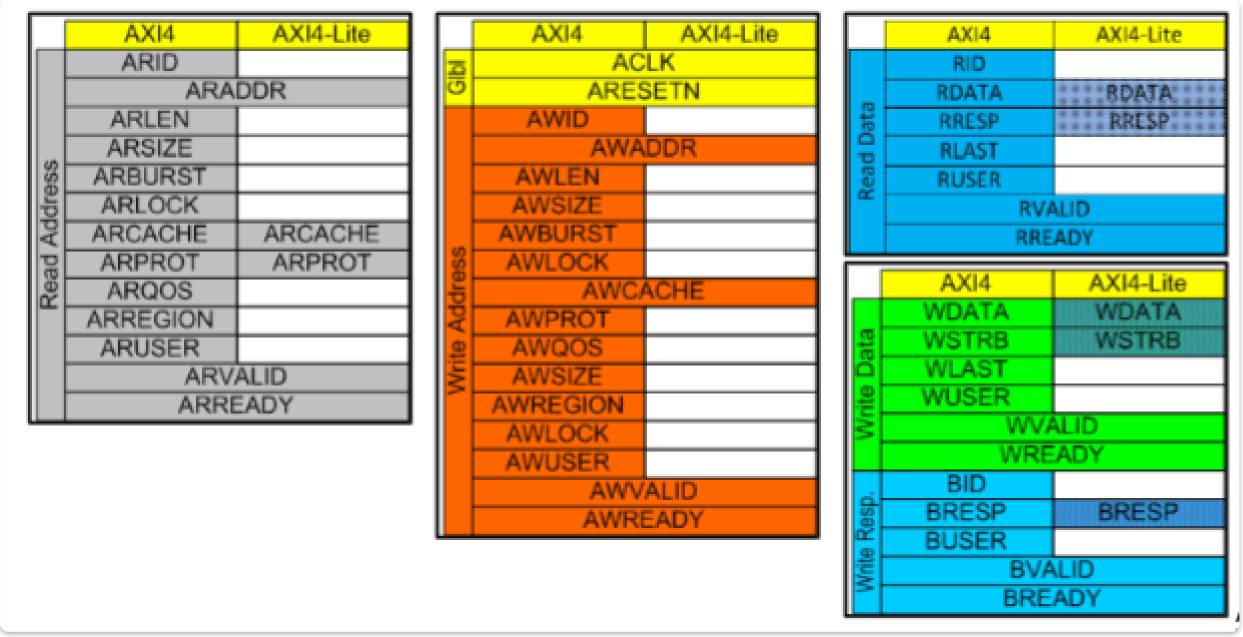
Data Only Moves from Master to Slave

The following terms are specific to streaming:

- *Transfer*: A transfer represents a single piece of data moving across an AXI-Stream interface. It is defined by a *single TVALID-TREADE handshake.*
- *Packet*: It is a group of bytes moving together across AXI-Stream. It can be considered as an AXI "burst" of AXI-Full implementation. A packet consist of a single transfer or multiple transfers.
- *Frame*: It is the highest level of byte grouping. It contains a integer number of packets. A frame can be a large number of bythes, such as an entire video frame buffer.
- *Steam*: It refers to the transport of data from one source to one destination.

Some other features of AXI stream, are:

- Only single direction data is supported. Master to Slave.
- If two devices needs to steam to each other (bidirectional transactions of data), two AXI-streaming channels are required, this is dual simple.
- No address channel exist. As overhead is reduced to the minimum. All data is sent to a unspecified address. Therefore, in this implementation the master and slave have to be carefully configured (the protocol is very application dependent) to understand with each other and know what to do with the data.
- Optional control signals are available, such as start-of-frame and end-of.frame. However, those markers can be embedded into the stream, removing the need for sideband control signals.
- A signal for marking if data is valid and when the slave is ready to receive the date are also available.
- Below there is a summary of the set of signals:



	AXI4	AXI4-Lite
Read Address	ARID	
	ARADDR	
	ARLEN	
	ARSIZE	
	ARBURST	
	ARLOCK	
	ARCACHE	ARCACHE
	ARPROT	ARPROT
	ARQOS	
	ARREGION	
	ARUSER	
	ARVALID	
	ARREADY	

	AXI4	AXI4-Lite
Glbl	ACLK	
	ARESETN	
Write Address	AWID	
	AWADDR	
	AWLEN	
	AWSIZE	
	AWBURST	
	AWLOCK	
	AWCACHE	
	AWPROT	
	AWQOS	
	AWSIZE	
	AWREGION	
	AWLOCK	
	AWUSER	
	AWVALID	
	AWREADY	

	AXI4	AXI4-Lite
Read Data	RID	
	RDATA	RDATA
	RRESP	RRESP
	RLAST	
	RUSER	
	RVALID	
	RREADY	

	AXI4	AXI4-Lite
Write Data	WDATA	WDATA
	WSTRB	WSTRB
	WLAST	
	WUSER	
	WVALID	
	WREADY	
Write Resp.	BID	
	BRESP	BRESP
	BUSER	
	BVALID	
	BREADY	

Summary:

1. AXI is part of ARM´s AMBA specification. It is used by Xilinx to connect dedicated silicon resources in the PS to IP in the PL.
    - AXI master does not have to be a processor, but any IP that can initiate a transaction.
2. AXI is a point-to-point interconnect mechanism
    - Xilinx offers IP Interconnects, which allow one master to connect with several slaves.
    - Three varieties of AXI are available: AXI-Full, AXI-Stream, AXI-Lite. These are differentiated mainly by the amount of data and time required in a transaction.

AXI Transaction

AXI4-Full Data Transaction

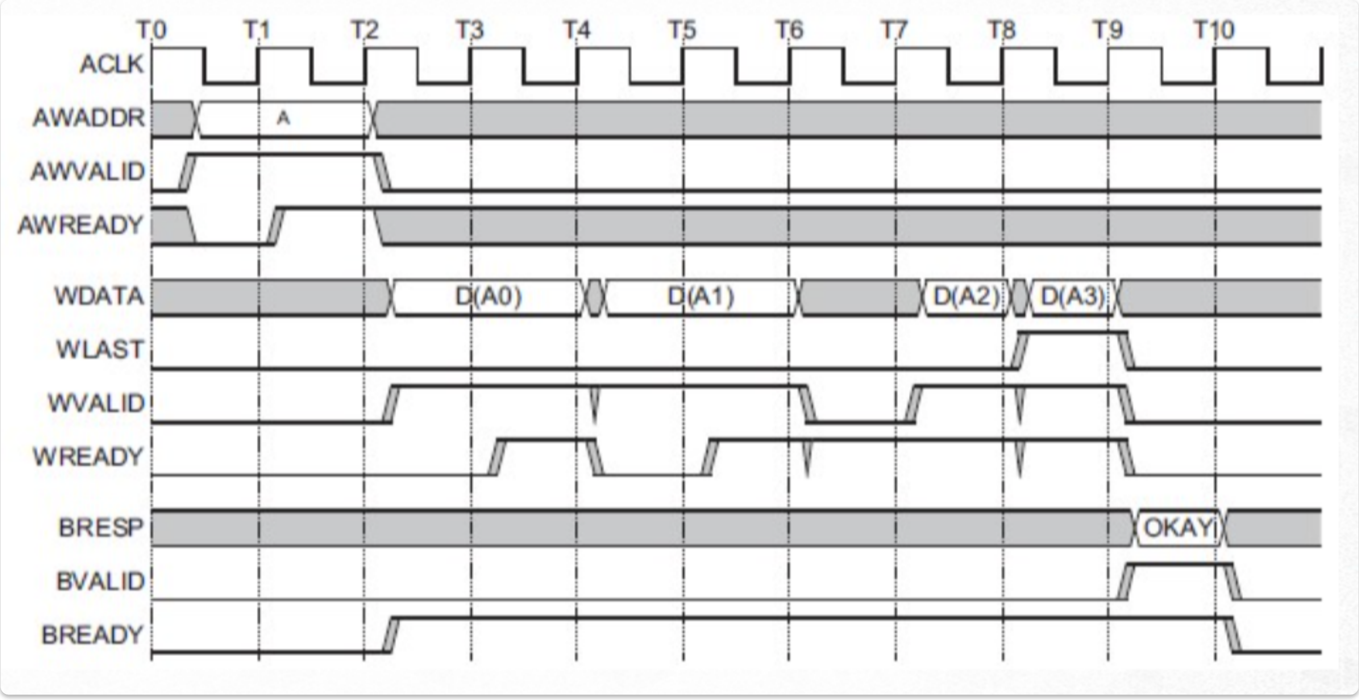
BURST WRITE OPERATION

The Figure below shows an example of *Burst data Write transaction* in AXI4Full. The signals represent the three channels or write operation.

Signals beginning with "AW" represents "Address Write" channel, "W" the "Write Data" channel and "B" for "Write Response" channel.

ACLK: Clock of the communication
AWADDR: Send the address where the data will be written. (note that this is the address of the byte, not the WORD!. One address every 8bits (or every 1 byte)).
AWVALID: The Master set it high to indicate at the Slave, that the Master is ready to begin.
AWREADY: Write address ready. Slave generates this signal when it can accept Write Address and control signals. After this signal is high, AWADDR is received and data transaction starts through *WDATA* channel.
Important note: This relationship between xVALID and xREADY is considered as handshake between Master and Slave (o source and destination channles). This handshake exist in every AXI protocol.(AXI-full, AXI-lite and AXI Stream)

WLAST: Indicates the last data in the burst to be sent.
WVALID: Similarly to the address, this signal is generated from the master to tell the slave that the data on the *WDATA* channel is valid. The slave responds through *WREADY*, indicating that is ready to receive that piece of data.
BRESP: Once all the data is sent, this signal is asserteded. Also, *BVALID* is asserted to indicate if the data was received correctly or not.
BREADY: Is high during all period that data is being transfer, once it finished is low again (together with *BRESP* and *BVALID*).

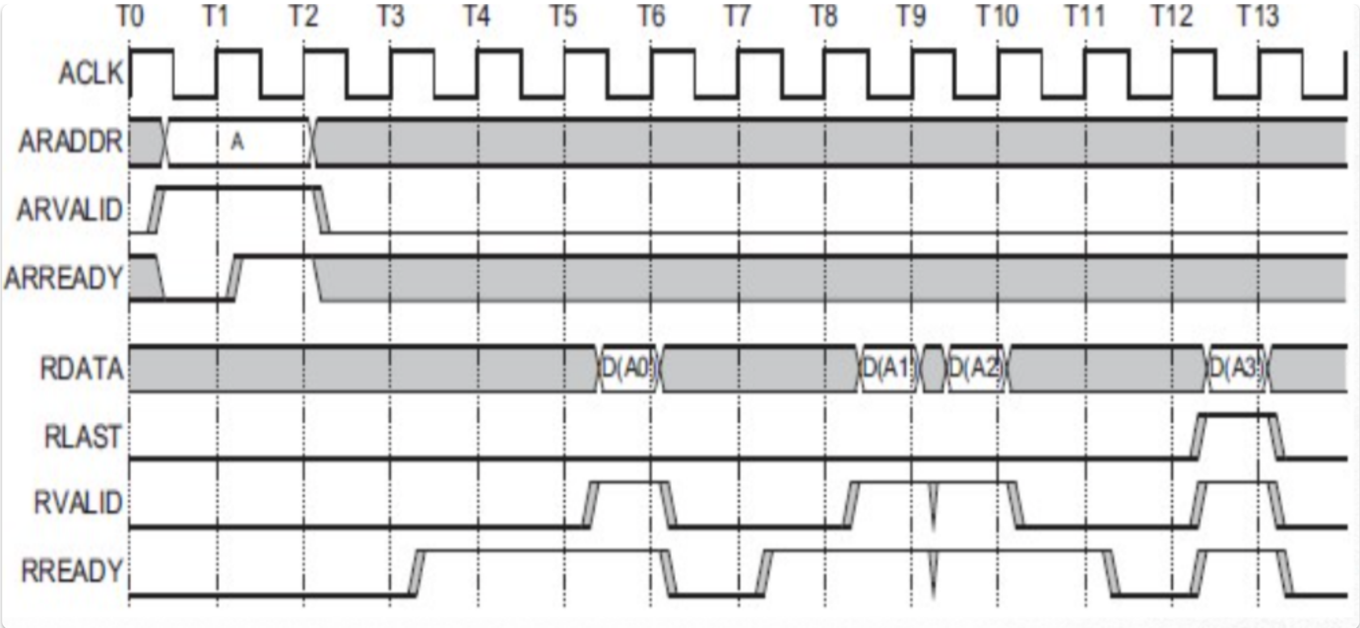


BURST READ OPERATION

Similarly to write operation, "AR" stands for "Read Address" channel and "R" for "Read Data" channel. For read operation, only two channels are involved, as the acknowledgment signal is the data read itself.

The master firstplaces the address that wants to read into the *ARADDR* channel and asserts the *ARVALID* signal. The salves respond through the *ARREADY*, indicating that it is ready to the master, who can see this signal in the next clock transitioning. This completes the address phase and now activity starts on read data channel.

Thereafter, the master asserts the *RREADY* signal to indicates that is ready to receive data. The slave then places data on the *RDATA* channel and asserts the *RVALID* signal. After data is received, Master take low *RREADY* and assert it again in the next clock cycle to receive a new data and the cycle is repeated. In this example, the 4 burst data was shown. It is possible to send up to 256 data bursts.
Notice that during the burst, the address channel has not changed. However, the address is automatically incremented by an offset after each read. That is part of the protocol when burst is used.



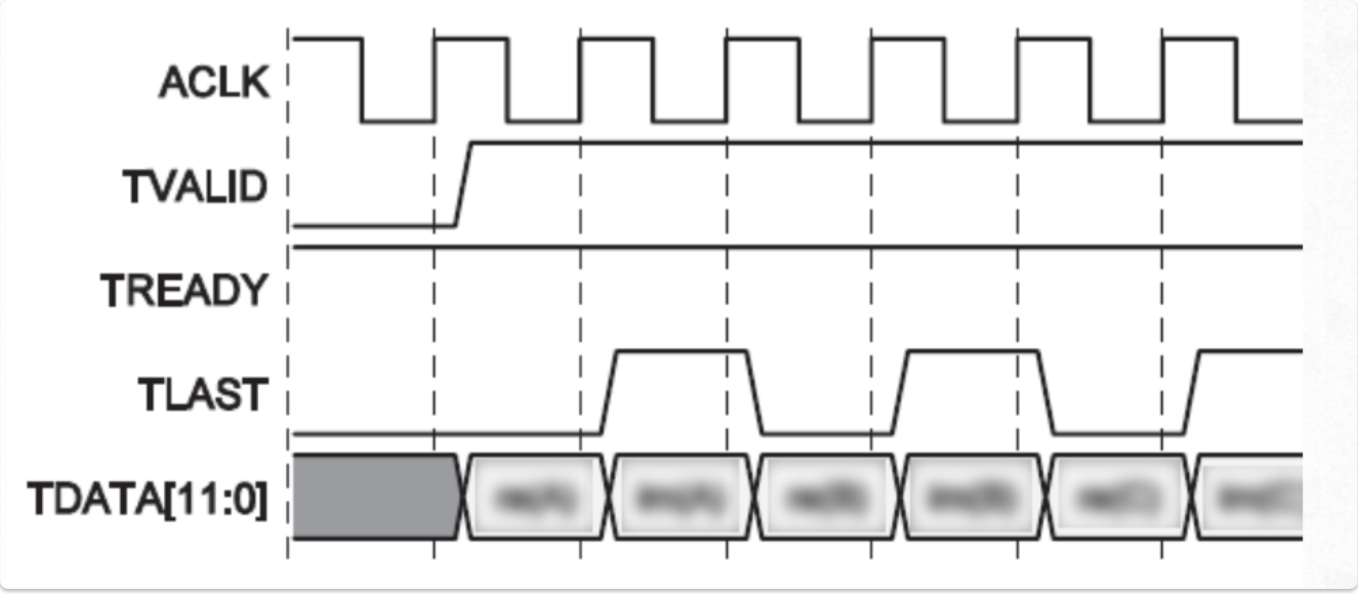
For AXI4-Lite, the same signals are used, with limitation of some control signals that disappear, also only single-datum burst is possible.

AXI4-Strean Eliminates the address write and read channel and response channel.

The followiong example shows the signals for one transaction. The example shows that the slave is always ready to receive data, as *TREADY* is always assert, also the master is capable of issuing new data at every clock, as the signal *TVALID* is permanently high. Both signals can be configured by the user to indicate beginning and end of data frames.

This example uses the *TLAST* signal to indicates that the packets are composed of two data burst.

In this form master and slave achieves maximum throughput (speed to transferring data).



This shows different possible valed transactions with different timings, including delays in between the data transfer or reading.

An interesting summary and more detailed explanation can be found here

Summary

AXI offers the following configurations:

- AXI-Full is the superset of all signals and capabilities. (e.g. memory access or moving data)
- AXI-Lite is optimized for PL implementation (e.g. Peripheral configuration)
- AXI-Streaming is optimized for one way data pumping (e.g. video streaming)

AXI-Full and AXI-Lite are both capable of reading and writing:

- Handshaking signals indicate readiness of the master and slave and data status.
- AXI-streaming is written only with a minimum of handshaking.