The AXI4 Bus

Overview of AXI4 bus operations

The “advanced extensible interface” (AXI) bus is a high-performance, point-to-point, master-slave parallel bus used to connect on-chip peripheral circuits (or IP blocks) to processor cores. The AXI standard is produced by ARM, and it is open and royalty free. Since its introduction in 2003, it has been adopted by many processor manufacturers, and it is now the recognized standard for on-chip busses. The complete AXI standard includes many high performance features, like variable address and data bus widths, burst operations, advanced caching functions, and out-of-order transactions. Many systems don’t require these high performance features, so the simpler, “lighter weight” AXI4-Lite protocol was introduced in 2010 to provide system designers with a smaller and less complex bus interface (AXI4-Lite does not include variable bus widths or cache support, and allows only one 32-bit data transfer per read/write transaction). The AXI4-Lite interface targets control and monitoring of IP blocks, and it is used in the ZYNQ device to connect the ARM and FPGA.

The AXI4 bus uses “channels” to separate read and write transactions into semi-independent operations that can each proceed at their own pace. The Read Address and Read Data channels transfer data from the slave to the master, and the Write Address, Write Data, and Write Response channels transfer data from the master to the slave. Each of these five channels includes two handshaking signals to control operations. The handshake signals use a Ready/Valid mechanism – the recipient asserts the Ready signal to indicate that it is ready to receive data, and the sender asserts the Valid signal to indicate data on the bus is ready to be sampled. Note that there is no implied sequence for these signals – the recipient can assert Ready whenever it is ready, and the sender can assert Valid whenever it drives the bus. If the receiver asserts Ready before the sender asserts Valid, then the transaction will occur within one clock cycle. If the sender asserts Valid before the receiver asserts Ready, then the sender must continue to drive the bus (and the Valid signal) until Ready is asserted. It is worth highlighting that the sender does not wait for Ready to be asserted before asserting Valid (if the sender did wait for Ready, the bus could “hang”).

The figure below shows a high-level view of the flow of AXI4-Lite bus transactions. Note that read and write transactions can occur simultaneously without interference. Also, the write address and write data can be presented at the same time, or the write data can be sent after the write address. And further, note that a second write address (A2) can be sent before the previous write cycle (A0) has been completed.

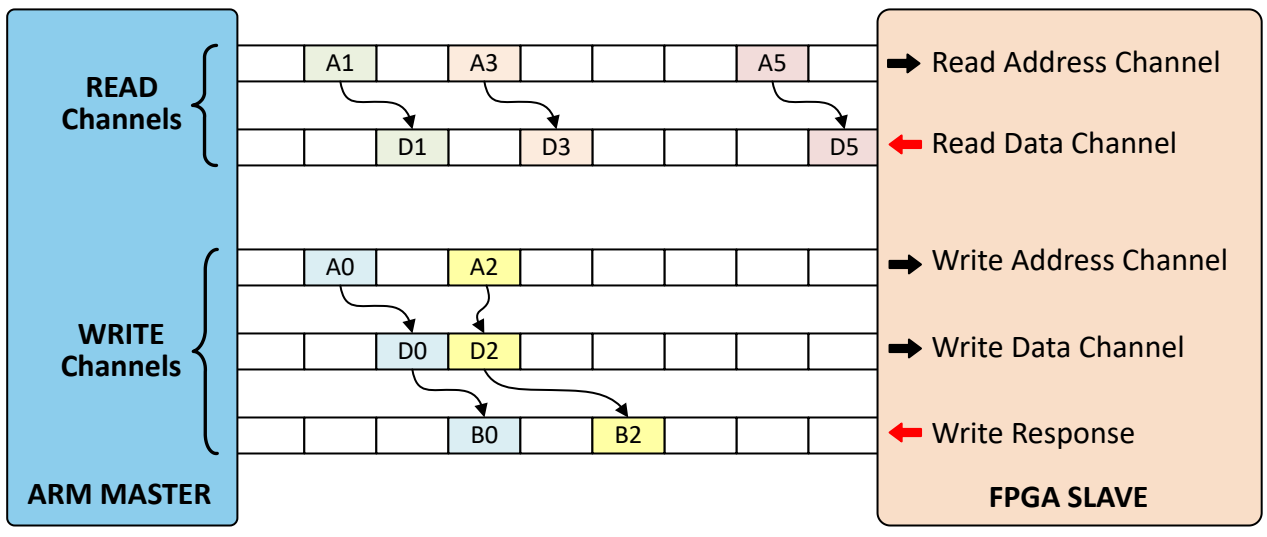

Figure 1. High-level AXI4-Lite data flow showing the five channels

Each channel includes a collection of signals that communicate all needed data and status information, and each channel has its own unique READY and VALID strobes (so, there are five pairs of strobes). Each channel runs independently of the other channels, and the handshake signals let each channel operate at its own best speed. This means, for example, if the slave device on the Write Address channel isn’t ready to accept a new address, it can hold off the master without effecting traffic on any other channel. If the Read Data channel was simultaneously transferring data back to the master, that data transfer could continue unabated, even as the Write Address channel was stalled.

The table below shows all AXI signals. The AXI4-Lite protocol uses a subset of those signals, and they are shown in the shaded boxes (we are concerned with the AXI4-Lite signals).

AWREADY	WDATA[N:0]	BREADY	ARREADY	RDREADY
AWADDR[31:0]	WDSTRB[N/8:0]	BRESP[1:0]	ARADDR[31:0]	RDATA[N:0]
AWLEN[7:0]	WDLAST		ARLEN[7:0]	RDRESP[1:0]
AWSIZE[2:0]			ARSIZE[2:0]	RDLAST
AWPROT[2:0]			ARPROT[2:0]	
AWBURST[1:0]			ARBURST[1:0]	
AWLOCK			ARLOCK	
AWCACHE[3:0]			ARCACHE[3:0]	
AWREGION[3:0]			ARREGION[3:0]	
AWQOS[3:0]			ARQOS[3:0]	

Figure 2. Complete list of AXI4 signals

To complete a write transfer:

1. The master drives the Write Address (0xF8000000) and Write Data (0xA5A5F0F0) on the write channels, and drives AWVALID and WDVALID strobes to tell the slave the address and data are valid (Note: the master also drives 0b1111 on the WDSTRB signal to indicate all four bytes are valid);
2. The slave asserts AWREADY and WDREADY signals to indicate it can sample the data;
3. When the AWVALID/WREADY and WDVALID/WDREADY strobes are asserted at the same time, the handshake is complete, and all strobes are deasserted;
4. The slave drives “00” on BRESP and asserts the BVALID strobe to indicate the transfer was successful, and the next rising edge completes the transaction.

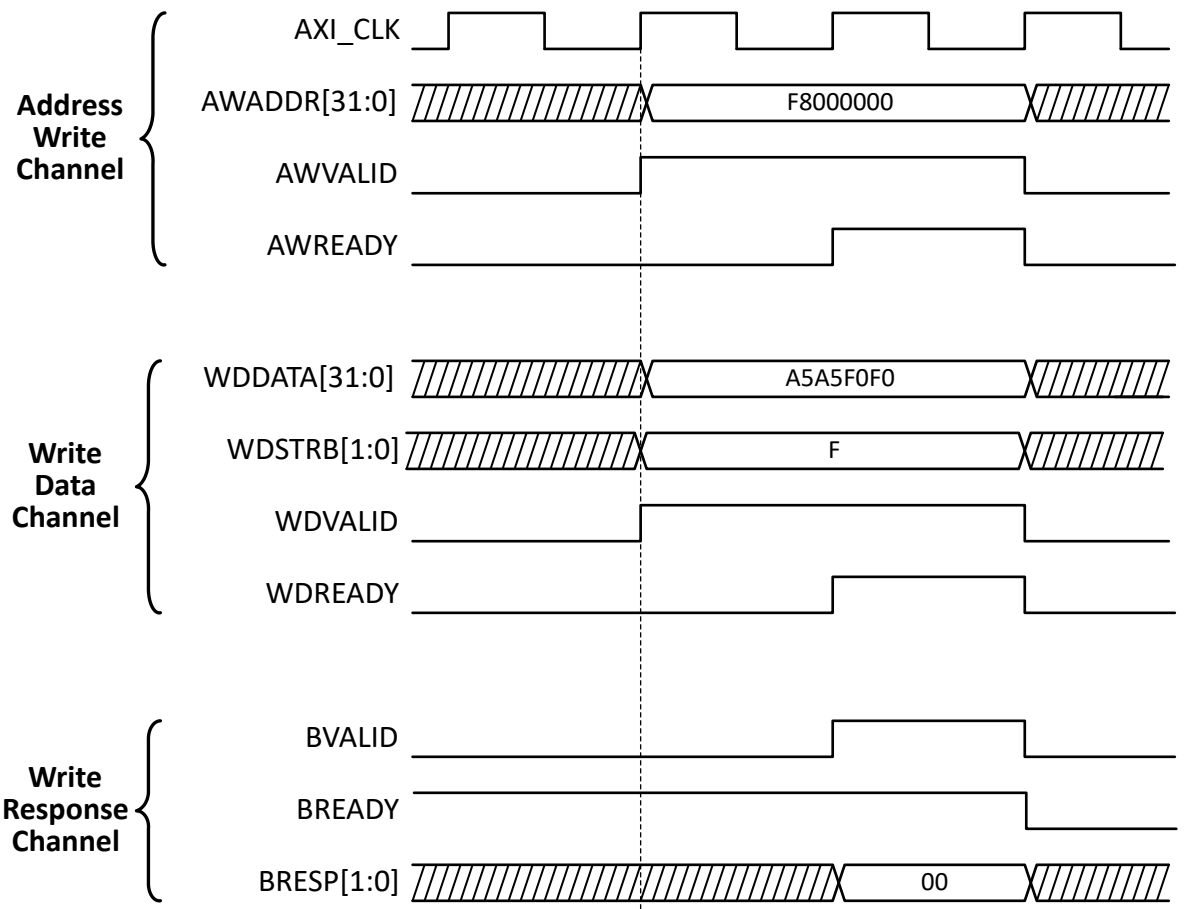

Figure 4. AXI4-Lite Write Transaction timing

To complete a read transfer (moving data from the slave to the master):

1. The master drives the Read Address (0xF0000000) channel and asserts ARVALID to tell the slave the address is valid;
2. The slave asserts ARREADY to indicate it can sample the address. When ARVALID and ARREADY are both asserted, the slave has the address, the handshake is complete, and the master and slave deassert ARVALID and ARREADY, respectively;
3. The slave drives the Read Data (0xFFFF0000) channel and asserts RVALID to tell the master the data is ready, and the master drives RREADY to indicate it is ready to sample the data;
4. When RREADY and RVALID are both asserted, the transaction is complete and the strobes are deasserted;
5. The slave drives “00” on the RRESP signals in the Read Data channel to indicate data was received successfully (or other codes as indicated).

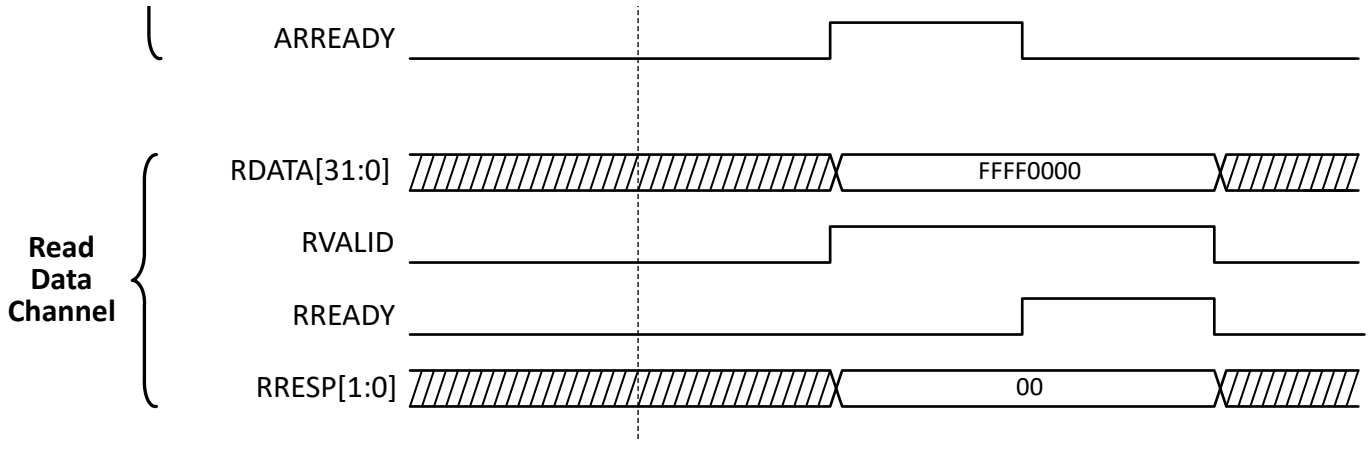

Figure 4. AXI4-Lite Read Transaction timing

Using the READY and VALID strobes lets both the master and the slave control the rate at which information moves between the master and slave: if the master sends an address and asserts VALID to signal the slave the bus contains valid data, the slave can hold READY de-asserted until it is ready to accept the data (and this prevents the master from starting a new transaction). This channel/handshake architecture lets the channels operate somewhat independently, and lets read and write data move simultaneously.

The table below lists/defines all 160 AXI4-Lite bus signals. 128 signals transport address and data, and 12 more accompany the addresses to define memory protections and caching options (in a typical use, protections and caching are turned off, so these signals default to 0). Ten signals are used for handshaking (two per channel), and two for global clock and reset. Four of the remaining eight signals define which bytes to write in a write cycle (typically all of them), and four are response codes that indicate whether a transfer encountered errors (errors are exceedingly rare, so these codes are typically 0). Not so complicated!

AXI4-Lite Channel Signals		
Global Signals		
ACLK	M ➡ S	Clock Source
ARESETN	M ➡ S	Global Reset
Write Address		
AWVALID	M ➡ S	Write address valid (write address/control signals are valid)
AWREADY	M ⬅ S	Write address ready (slave ready to accept address)
AWADDR[31:0]	M ➡ S	32-bit write address
AWWPROT[2:0]	M ➡ S	Protection type (for security – usually ignored)
AWCACHE[3:0]	M ➡ S	Memory type (cacheable or not – usually ignored)
Write Data		
WDVALID	M ➡ S	Write data valid (write data from master is valid)
WDREADY	M ⬅ S	Write data ready (slave ready to accept write data)
WDDATA[31:0]	M ➡ S	32-bit write data
WDSTRB[3:0]	M ➡ S	Write strobes (indicate which bytes written – usually all)
Write Response		
BVALID	M ⬅ S	Write response valid (slave generates when response valid)
BREADY	M ➡ S	Write response ready (master ready to accept response)
BRESP[1:0]	M ⬅ S	Write response (indicates status of write transaction)
Read Address		
ARVALID	M ➡ S	Read address valid (read address/control signals are valid)
ARREADY	M ⬅ S	Read address ready (slave ready to accept address)
ARADDR[31:0]	M ➡ S	32-bit read address
ARWPROT[2:0]	M ➡ S	Protection type (for security – usually ignored)
ARCACHE[3:0]	M ➡ S	Memory type (cacheable or not – usually ignored)
Read Data		
RVALID	M ⬅ S	Read data valid (read data from slave is valid)
RREADY	M ➡ S	Read ready (master ready to accept read data)
RDATA[31:0]	M ⬅ S	32-bit read data
RRESP[1:0]	M ⬅ S	Read response (indicates transfer status)

Figure 4. List of AXI4-Lite signals

AXI interconnect Core

Whenever the ARM (the AXI master) is connected to an FPGA IP block (the AXI slave), the AXI interconnect bus-management IP core is automatically inserted into the design. This interconnect core supports the full AXI protocols (AXI4, AXI4-Lite, and AXI-Stream), and it is used by default because many AXI variations are possible. The core ensures communications between the master and slave run smoothly, regardless of the differences in configuration.

In our case, we will use the AXI4-Lite protocol with a single master and a single slave, and so no discrepancies are possible. The interconnect core is still shown in the block diagram, but it will devolve to simple pass-through wires during synthesis.

Contact Us

contact@realdigital.org
(509) 336 9656
Mon - Fri: 9am - 5pm

Our Location

655 SW James Pl
Pullman WA 99163
USA