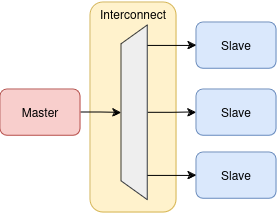
AXI

**AXI, or Advanced eXtensible Interface,** is a bus specification that offers an advanced and flexible approach to interconnecting components in hardware systems. Developed by ARM, AXI has become a common choice in hardware designs, especially in FPGA (Field-Programmable Gate Array) and SoC (System-on-Chip) environments.

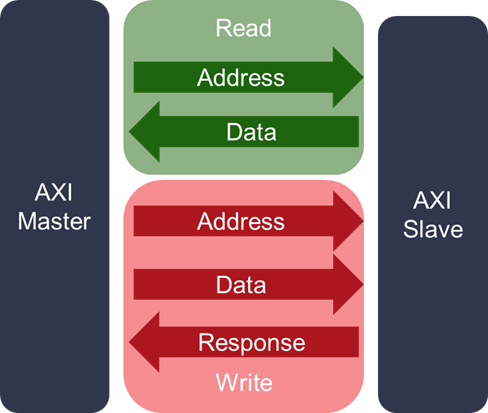
The AXI architecture defines three main types of interfaces: AXI4, AXI4-Lite and AXI4-Stream. Each of these interfaces is designed to meet different communications requirements, from complex read and write transactions to continuous streaming data transfers.  
  
 In this project, **AXI Lite** was chosen, representing a reduced and simplified variant of the AXI protocol. This protocol fulfills all project requirements while consuming fewer hardware resources compared to the full version. Additionally, it provides a more straightforward implementation, particularly suitable for peripherals that necessitate simplified communication. AXI Lite operates on a synchronous communication scheme between a master and a slave, facilitating data transfer and **signal transmission through** **dedicated channels**, including the address channel, data channel, and control channel.

However it is important to note that the protocol has some limitations, such as all transactions having a burst length of 1, all accesses being non-modifiable and non-bufferable, and the lack of support for exclusive accesses. Additionally, the fixed data bus width must be either 32 or 64 bits, with most components using a 32-bit interface.

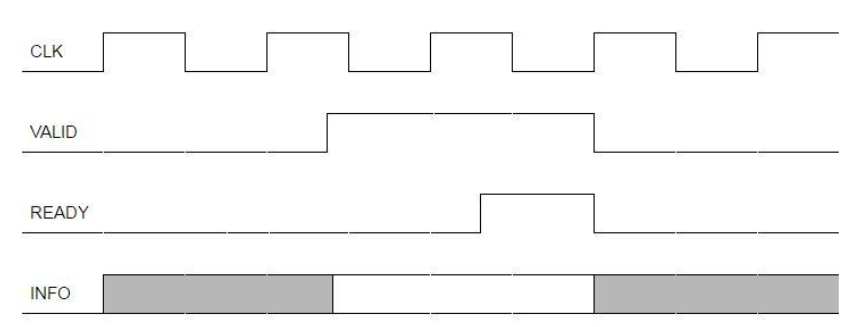
In order to be able to connect all the peripherals of this project to the SOC, there was a need to use an **AXI Interconnect**, as this facilitates communication between different AXI modules in a system. AXI Interconnect acts as a hub, connecting one or more masters and slaves in an AXI system, managing the transfer of data between these components, ensuring that they occur efficiently and without conflicts.  


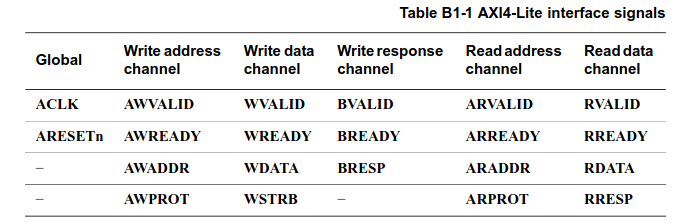
The **read** operation on the AXI Lite is initiated when the master sends a valid address signal (ARVALID), and an address indicating where it wants to read the data (ARADDR). The slave responds when it is ready to provide data (RVALID), signaling with a valid data signal and sending the data (RDATA) to the master. And finally the master confirms receipt of the data (RREADY).

The **writing** operation begins when the master sends signals valid address (AWVALID), valid data (WVALID), data to be sent (WDATA) and the address (AWADDR) where the data must be written. Finally, the slave confirms receipt of the write request (BVALID) and the master confirms the completion of the transaction (BREADY).



In both reading and writing operations, AXI Lite uses **handshaking**, ensuring efficient and controlled data transfer between the master and slave on the bus. The VALID signal indicates that the data is valid to be transferred and the READY indicates that the component is ready to accept data or commit the transaction.



Signals: 

https://docs.xilinx.com/v/u/en-US/ug1037-vivado-axi-reference-guide

https://support.xilinx.com/s/article/1053914?language=en\_US

<https://docs.xilinx.com/v/u/en-US/ug761_axi_reference_guide>

AMBA ® AXI ™ and ACE ™ Protocol Specification (PDF)