**Protocol** is a set of rules and the regulations which tells how the communication happens between master and slave (or) between two components.

There are two categories of protocols:

1. On-chip protocol – (protocols which defines the communication of components on the chip)
   1. APB, AHB, AXI, Open core protocol (OCP), peripheral component interconnect (PCI).
   2. There is no limitation on the port size because fabrication technology is so advanced that we can keep 1000 pins in nm space or even lesser that space that is not in the case of peripheral protocols so we can’t use too many ports in peripheral protocols.
   3. They are limited by frequency of operation.
   4. Theoretically there is no limitation on at what frequency AXI and AHB can run at.
      1. They can run at infinite frequency what is limiting them is timing closure of the chip.
      2. There will be lakhs of FF’s inside the chip, none of them should not give setup and hold time violations at the specified frequency.
      3. As the frequency increases, chances of setup violation is increases which results in meta stability condition, we don’t know the output of FF, then whole chip goes to meta stability state.
      4. So we are unable to close the timing at that frequency.
      5. Immediate solution is reducing the frequency of operation.
      6. That the limit the frequency of operation in AXI but not the AXI or AHB.
   5. Data communication happens terms of transactions.
   6. These transactions are happen in multiples phases like arbitration phase, request phase, data phase, response phase.
   7. Handshaking happen using dedicated signals.
2. Peripheral protocol – (protocols which defines the communication of components outside the chip)
   1. We can’t use too many ports.
   2. As number of ports increases packaging is more difficult. (maintain min gap between ports)
   3. Data communication happens in terms of packets/frames.
   4. No handshaking signals.

**Protocols are differentiation based on below features**:

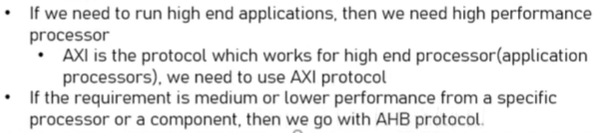
1. Performance (more)
2. Power consumption (less)
3. Cost (less)

In ideal protocol, we expect high performance, low power consumption and less cost.

For Intel chip, we get high performance and low power consumption but cost is high.

**How to decide which protocol**?

1. Understand the requirement.
   1. High or low performance. (\*\*\*\*\*\* must match\*\*\*\*\*\*)
   2. Power consumption
   3. Cost



ARM has three types of processors.

1. A (Application) series – AXI
2. M (Embedded) series - AHB
3. R (Real) series - AHB

**AHB protocol**

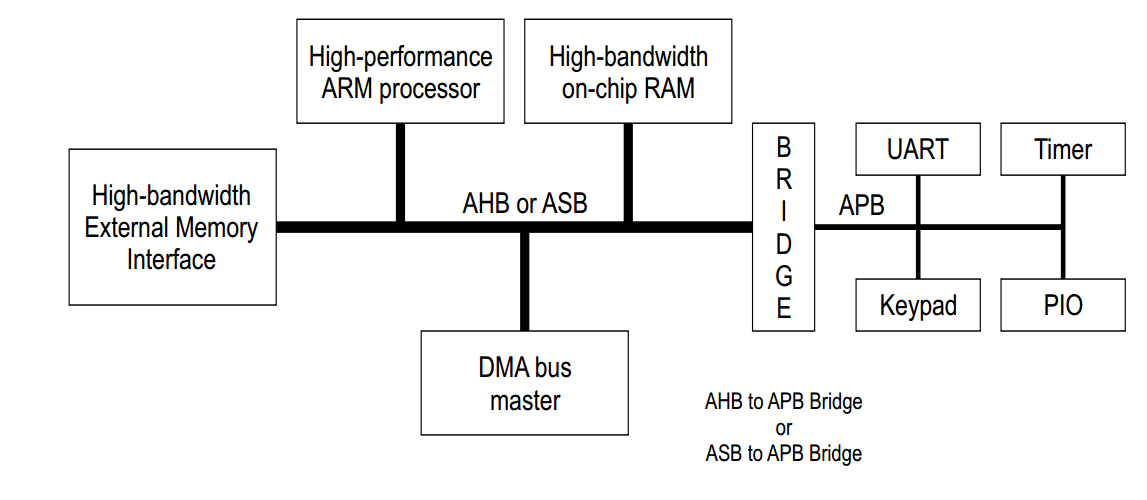
Advanced High-performance bus – All ARM processors will be either based on AXI or AHB.

It is an on-chip protocol which is used to transfer data between master and slave.

**AHB Architecture:**

AHB Architecture consists of

1. Master.
2. Slave.
3. Interconnect.



Interconnect is used for connecting multiple masters to multiple slaves.

Inter-connect routes the master request to the target slave, routes the slave response to the correct master.

Inter-connect is made up of decoder and multiplexer logic.

1. **It decodes incoming request address, figures out in which direction request should be routed.**
2. It automatically knows in which direction response should route.
3. At any time 1master and 1slave connection get established, so interconnect knows which the master active now.

**AHB** gives **High performance** through pipelined operation, burst transfers, multiple bus masters and split transactions.

AHB supports max 16 masters and 16 slaves because load (capacitance) gets increase as we connect more components to interconnect. At one point, the signal integrity gets impacted.

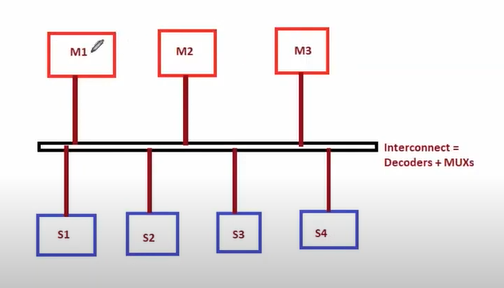
What if I need to connect more slaves then 16 – use one more interconnects, connect this to first interconnect using a bridge (AHB2AHB Bridge).

AHB transfers happen in 3 phases.

1. Arbitration phase.

2. Request phase.

3. Data phase.



**Why do we need arbitration/arbiter?**

Multiple processes/components are trying to access same resource, we need arbitrator.

What is resource – interconnect access to the bus.

Components that want access – Masters (M1, M2, M3)

Masters are trying to access the same resource – we need arbiter.

Role of arbiter – Allocate the resource to the requesting components based on guidelines.

############interconnect = arbiter + decoder + multiplexer. #################

1. Arbitration phase.

Master asserts a request to the arbiter. Arbiter will grant access to requesting master. Arbiter is present inside interconnect.

HBusReq – master is requesting to interconnect that I want to access the bus.

Hlock,

Hmasterx – which mater is active now?

Hgrantx - Hsplitx.

2. Request phase.

Happens multiple times.

Granted master drives address and control information to slave initiating a new transfer.

Haddr,Hwrite,Htrans,Hbrust,Hsize,Hprot.

3. Data phase.

Slave accepts Hwdata from master in case of write transfer.

Issues Hrdata from slave in case of read transfer.

Data phase is complete only when Hready=1

Hready,Hwdata,Hrdata,Hresp

**AHB VS AXI**:

1. Both protocol support maximum of 16masters and 16 slaves.
2. AXI allows all masters to communicate with the entire slave at same time.
   1. AXI has concept called ID , AWID/WID/BID/ARID/RID associated with each phase of transfer which gives unique ID to each transaction/each phase, allowing multiple transactions to happens concurrently (in any order) at same time.
3. AHB does not have IDS, so one master have to do transaction at any point of time.
4. Both can work at any frequency, but timing closure is limiting it frequency of operation.

**Round robin arbitration**:

1. First grant given to master1.
2. Once master1 complete transaction, grant given to master2.
3. So on

**Handshaking in AHB protocol**:

Handshaking is a mechanism through which master and slave talk to each other.

Handshaking signals: Htrans and Hready.

Htrans – issued by master to convey its state to slave.

**States of master**:

1. IDLE – Master is not doing any transaction, also there is no transfer in progress.

But it can complete data phase of previous address phase not initiate the new transaction.

1. BUSY – Master is in the middle of an ongoing transaction. Temporarily it is busy, hence it can’t do address phase.

It is addressing some other instruction, hence it got busy.

But it can still complete the data phase for previous address phase.

1. NONSEQ - Used by master to initiate a new transaction. It is the 1st transfer of the burst or only transfer in single transfer.
2. SEQ - 

Slave will convey its readiness with Hready=1.

