On-Chip Interconnect Protocols

Tarik Graba, Sumanta Chaudhuri
<tarik.graba@telecom-paristech.fr>
Plan

Why Protocols?

Background
- Handshake
- Burst Signaling
- Blocking & Non-Blocking Transfer

AXI
- AXI Channels
- AXI Highlights
- AXI3/AXI4 Differences
- AXI/OCP Differences

Protocol Compliance

References
An Example SoC from TI.

**OMAP** Used in various mobiles/tablets. e.g Amazon Kindle (OMAP 4430).
An Example SoC: TI OMAP 5432

- An Example SoC from TI.
  - Used in various mobiles/tablets. e.g Amazon Kindle (OMAP 4430).

- Multiple IPs from multiple vendors on the same chip.
- Each IP have different frequency, data width, addressing, Bandwidth/Latency Requirements etc. etc.
- How do they communicate?
An Example SoC from TI.

- Used in various mobiles/tablets. e.g. Amazon Kindle (OMAP 4430).
- Multiple IPs from multiple vendors on the same chip.
An Example SoC: TI OMAP 5432

- An Example SoC from TI.
  - Used in various mobiles/tablets. e.g. Amazon Kindle (OMAP 4430).
  - Multiple IPs from multiple vendors on the same chip.
  - Each IP have different frequency, data width, addressing, Bandwidth/Latency Requirements etc. etc.
An Example SoC from TI.

- Used in various mobiles/tablets. e.g. Amazon Kindle (OMAP 4430).
- Multiple IPs from multiple vendors on the same chip.
- Each IP have different frequency, data width, addressing, Bandwidth/Latency Requirements etc. etc.
- How do they communicate?
An Example SoC from TI.

- Used in various mobiles/tablets. e.g Amazon Kindle (OMAP 4430).
- Multiple IPs from multiple vendors on the same chip.
- Each IP have different frequency, data width, addressing, Bandwidth/Latency Requirements etc. etc.
- How do they communicate?
History : Shared Bus

- All masters are connected to the same set of wires.
- A bus controller will issue token to masters to utilize the bus.
- One master blocks the bus.
- Huge capacitative loading..
- Not suitable for high performance applications.
Modern Interconnect: Circuit Switched Network
The arbiter switches from one master to another using a pre-defined strategy.

- Round Robin.
- Priority Base.
- Time-out Based.
Transactions are transformed into network packets.

Packets will follow an available route to the slave.

Packet sizes and network topology are determined from application requirements.
On-Chip Protocols : Point to Point

- Dissociates network implementation from IP interface.
- Helps in plug’n play.
- Only the interface needs to be protocol compliant.
Design Goals for Modern Protocols

- Design Reuse, Plug’n play.
Design Goals for Modern Protocols

- Design Reuse, Plug’n play.
- Support of High Bandwidth/ Low Latency Traffic.
Design Goals for Modern Protocols

- Design Reuse, Plug’n play.
- Support of High Bandwidth/ Low Latency Traffic.
- Point-to-Point Protocols. (As opposed to shared bus)
Design Goals for Modern Protocols

- Design Reuse, Plug’n play.
- Support of High Bandwidth/ Low Latency Traffic.
- Point-to-Point Protocols. (As opposed to shared bus)
- Pipelined/Non-Blocking. Can have multiple outstanding requests.
Design Goals for Modern Protocols

- Design Reuse, Plug’n play.
- Support of High Bandwidth/ Low Latency Traffic.
- Point-to-Point Protocols. (As opposed to shared bus)
- Pipelined/Non-Blocking. Can have multiple outstanding requests.
- Be suitable for DRAM Traffic. (Out-of-order data, initial access latency)
Some On-Chip Interconnect Protocols

- **AMBA (Advanced Microcontroller Bus Architecture)**
  Open standard maintained by ARM.

- **OCP-IP (Open Core Protocol International Partnership)**
  Open standard developed by an industry consortium (TI, NXP, etc. etc.)

- Other standards include CoreConnect, VCI (Virtual Component Interface), STBus etc. etc.
Quiz

What are the differences between On-Chip Networks & Off-chip Networks?
Plan

- We will discuss some basic concepts: Handshake, Memory-Maps etc.
- We will discuss the AXI3 Standard first.
- We will see the differences between AXI3 & AXI4.
- We will see the differences between AXI OCP.
Plan

Why Protocols?

Background
   Handshake
   Burst Signaling
   Blocking & Non-Blocking Transfer

AXI
   AXI Channels
   AXI Highlights
   AXI3/AXI4 Differences
   AXI/OCP Differences

Protocol Compliance

References
Basics: Clocked Interface

ACLK

INFORMATION

Transfer

Transfer

Transfer

Transfer

Transfer
Basics : VALID/READY Handshake

- A clocked interface can’t be stalled.
- Handshake mechanism to include stalling behavior.
Basics: VALID/READY Handshake

ACLK

INFORMATION

VALID

READY

Transfer

Master is ready, Slave not ready.
(Downstream Stall)
Basics: VALID/READY Handshake

ACLK

INFORMATION

VALID

READY

Master is not ready, Slave is ready. (Upstream Stall)

Transfer
Quiz

- What are the pros/cons for a valid/ready interface compared to clock interface?
- When would use handshake and when would you use clocked interface?
Bursts

- Successive writes/reads to a predefined address pattern.
- e.g. incremental (one dimensional), 2D Bursts.
- MRMD : Multiple Request Multiple Data.
- SRMD : Single Request Multiple Data.
Incremental Burst : MRMD

A. The master starts a read request by driving RD on MCmd, a valid address on MAddr, four on MBurstLength, INCR on MBurstSeq, and asserts MBurstPrecise. MBurstLength, MBurstSeq and MBurstPrecise must be kept constant during the burst. MReqLast must be deasserted until the last request in the burst. The slave is ready to accept any request, so it asserts SCmdAccept.

B. The master issues the next read in the burst. MAddr is set to the next word-aligned address (incremented by 4 in this case). The slave captures the address of the first request and keeps SCmdAccept asserted.

C. The master issues the next read in the burst. MAddr is set to the next word-aligned address (incremented by 4 in this case). The slave captures the address of the second request and keeps SCmdAccept asserted. The slave responds to the first read by driving DVA on SResp and the read data on SData.
Figure 32 illustrates a burst of four 32-bit words, wrapping burst read, with optional burst framing information (MReqLast/SRespLast). MReqLast flags the last request of the burst and SRespLast flags the last response of the burst. As a wrapping burst is precise, the MBurstLength signal is constant during the whole burst, and must be power of two. The wrapping burst address must be aligned to boundary MBurstLength times the OCP word size in bytes.

### Sequence

<table>
<thead>
<tr>
<th>Request Phase</th>
<th>Response Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MCmd</strong></td>
<td><strong>SCmdAccept</strong></td>
</tr>
<tr>
<td>IDLE</td>
<td>NULL</td>
</tr>
<tr>
<td>RD₁</td>
<td>DVA₁</td>
</tr>
<tr>
<td>RD₂</td>
<td>DVA₂</td>
</tr>
<tr>
<td>RD₃</td>
<td>DVA₃</td>
</tr>
<tr>
<td>RD₄</td>
<td>DVA₄</td>
</tr>
<tr>
<td>IDLE</td>
<td>NULL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>MAddr</strong></th>
<th><strong>SResp</strong></th>
<th><strong>SData</strong></th>
<th><strong>SRespLast</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8₁</td>
<td>NULL</td>
<td>D₁</td>
<td>A</td>
</tr>
<tr>
<td>0xC₂</td>
<td>DVA₁</td>
<td>D₂</td>
<td>B</td>
</tr>
<tr>
<td>0x0₃</td>
<td>DVA₂</td>
<td>D₃</td>
<td>C</td>
</tr>
<tr>
<td>0x4₄</td>
<td>DVA₃</td>
<td>D₄</td>
<td>D</td>
</tr>
</tbody>
</table>

- **MReqLast**
- **MBurstLength**
- **MBurstSeq**
- **MBurstPrecise**

Clk  

<table>
<thead>
<tr>
<th>0x8₁</th>
<th>0x03</th>
<th>0xC2</th>
<th>0x04</th>
<th>0x05</th>
<th>0x06</th>
<th>0x07</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD₁</td>
<td>RD₂</td>
<td>RD₃</td>
<td>RD₄</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>0x02</td>
<td>0x03</td>
<td>0x04</td>
<td>0x05</td>
<td>0x06</td>
<td>0x07</td>
</tr>
</tbody>
</table>

- DVA₁, DVA₂, DVA₃, DVA₄
- NULL

A, B, C, D, E, F, G
C. The master captures the first response data. The slave issues the second response.

D. The master captures the second response data. The slave issues the third response.

E. The master captures the third response data. The slave issues the fourth response, and asserts SRespLast to indicate the last response of the burst.

F. The master captures the last response data.
Quiz

- A processor missed a data in the cache and wants to a cache-line fill. Which type of burst shall it use?
- What is cache line size for common processors ARM/MIPS?
- If the data width of the interface is 128 bits, how many cycles (minimum) are required for a cache line fill?
- Which one is better? SRMD or MRMD?
DRAM is the main performance bottleneck in an embedded system.

Protocols are designed for efficient utilization of DRAM.

DRAM response can come out of order, has high initial latency.
Blocking & Non-Blocking

- **Blocking:**
  - Master doesn’t emit a new request until the previous is finished.
  - Max. Outstanding Reads=1.
  - e.g AMBA (ARM Advanced Microcontroller Bus Architecture) (APB, AHB ...), Wishbone

- **Non-Blocking:**
  - Master emits N requests before waiting for the responses.
  - Max. Outstanding Reads=N.
  - e.g AXI, OCP, VCI.
Figure 27 shows three read transfers to a slave that cannot pipeline responses after requests. This is the typical behavior of legacy computer bus protocols with a single WAIT or ACK signal. In each transfer, SCmdAccept is asserted in the same cycle that SResp is DVA. Therefore, the request-to-response latency is always 0, but the request accept latency varies from 0 to 2.

**Figure 27 Non-Pipelined Read**

A. The master starts the first read request, driving RD on MCmd and a valid address on MAddr. The slave asserts SCmdAccept, for a request accept latency of 0. When the slave sees the read command, it responds with DVA on SResp and valid data on SData. (This requires a combinational path in the slave from MCmd, and possibly other request phase fields, to SResp, and possibly other response phase fields.)

B. The master launches another read request. It also sees that SResp is DVA and captures the read data from SData. The slave is not ready to respond to the new request, so it deasserts SCmdAccept.

C. The master sees that SCmdAccept is low and extends the request phase. The slave is now ready to respond in the next cycle, so it simultaneously asserts SCmdAccept and drives DVA on SResp and the selected data on SData. The request accept latency is 1.

D. Since SCmdAccept is asserted, the request phase ends. The master sees that SResp is now DVA and captures the data.

E. The master launches a third read request. The slave deasserts SCmdAccept.
Non-Blocking Read

Figure 28 shows three read transfers using pipelined request and response semantics. In each case, the request is accepted immediately, while the response is returned in the same or a later cycle.

Sequence

A. The master starts the first read request, driving RD on MCmd and a valid address on MAddr. The slave asserts SCmdAccept, for a request accept latency of 0.

B. Since SCmdAccept is asserted, the request phase ends. The slave responds to the first request with DVA on SResp and valid data on SData.

C. The master launches a read request and the slave asserts SCmdAccept. The master sees that SResp is DVA and captures the read data from SData. The slave drives NULL on SResp, completing the first response phase.

Clk

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCmd</td>
<td>IDLE</td>
<td>RD₁</td>
<td>IDLE</td>
<td>RD₂</td>
<td>RD₃</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>MAddr</td>
<td>A₁</td>
<td>A₂</td>
<td>A₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MData</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCmdAccept</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SResp</td>
<td>NULL</td>
<td>DVA₁</td>
<td>NULL</td>
<td>DVA₂</td>
<td>NULL</td>
<td>DVA₃</td>
<td>NULL</td>
</tr>
<tr>
<td>SData</td>
<td></td>
<td>D₁</td>
<td></td>
<td>D₂</td>
<td></td>
<td>D₃</td>
<td></td>
</tr>
</tbody>
</table>

Request Phase

Response Phase

Tarik Graba, Sumanta Chaudhuri
Out-of-Order Execution

CLK

BURST LENGTH

3 0 1

ADDR

RD1 RD2 RD3

VALID

READY

RDATA

RD2 RD3 RD3 RD1 RD1 RD1 RD1 RVALID
**Basics : Memory-Mapped Slaves**

<table>
<thead>
<tr>
<th>TARGET</th>
<th>START ADDRESS</th>
<th>END ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLAVE0</td>
<td>0x00000000</td>
<td>0x1FFFFFFF</td>
</tr>
<tr>
<td>SLAVE1</td>
<td>0x20000000</td>
<td>0x2FFFFFFF</td>
</tr>
<tr>
<td>SLAVE2</td>
<td>0x30000000</td>
<td>0x3FFFFFFF</td>
</tr>
<tr>
<td>SLAVE3</td>
<td>0x40000000</td>
<td>0x4FFFFFFF</td>
</tr>
</tbody>
</table>
Basics: REQUEST & RESPONSE PATH

MASTERO

MASTER1

MASTER2

MASTER3

SLAVE0

SLAVE1

SLAVE2

SLAVE3

ARBITER

ADDR
DECODER

ARBITER

ID
DECODER

MASTER0 [ID 0-3]

MASTER0 [ID 4-7]

MASTER0 [ID 8-11]

MASTER0 [ID 12-15]
An IP has a databus width of 128 bits. The average read latency from IP to DRAM is 32 cycles. In the return path the IP needs an average bandwidth of 16 bytes/cycle. How many outstanding read requests should it issue?

Is there a maximum limit on no. of outstanding reads?
Plan

Why Protocols?

Background
  Handshake
  Burst Signaling
  Blocking & Non-Blocking Transfer

AXI
  AXI Channels
  AXI Highlights
  AXI3/AXI4 Differences
  AXI/OCP Differences

Protocol Compliance

References
AMBA History

**AMBA 1**

- ASB™

**AMBA 2**

- APB™
- AHB™

**AMBA 3**

- APB2
- AHB-Lite
- ATB™
- AXI3™

**AMBA 4**

- AXI4
- AXI4-Lite
- AXI4-Stream
- ACE™
- ACE-Lite

*Figure 1 - Evolution of AMBA Standards*
AXI3 WRITE ADDRESS CHANNEL

- AWID[3:0]
- AWADDR[31:0]
- AWLEN[3:0]
- AWSIZE[2:0]
- AWBURST[1:0]
- AWLOCK[1:0]
- AWCACHE[1:0]
- AW PROT[1:0]
- AWVALID
- AWREADY
AXI3 WRITE ADDRESS CHANNEL

- **AWID[3:0]**
- **AWADDR[31:0]**
- **AWLEN[3:0]**
- **AWSIZE[2:0]**
- **AWBURST[1:0]**
- **AWLOCK[1:0]**
- **AWCACHE[1:0]**
- **AWPROT[1:0]**

**MASTER**

**SLAVE**

- **max no. of IDs supported 16**
- **Byte Address**
- **max Burst Length is 16**
- **Size of each beat in the burst, can be 1,2,4...128 Bytes**
- **Burst type supported: incr, wrap, fixed**

Special Transactions
AXI3 WRITE DATA CHANNEL

- WID[3:0]
- WDATA[31:0]
- WSTRB[3:0]
- WLAST

- ARVALID
- ARREADY
AXI3 WRITE DATA CHANNEL

INFORMATION

[3:0] WID

[31:0] WDATA

[3:0] WSTRB

[2:0] WLAST

HANDSHAKE

[1:0] ARVALID

ARREADY

MASTER

SLAVE

Data width can be 1, 2, 4, 8, ..., 128 Bytes

Byte Enable within Data word

$\log_2(\text{data width})$

Tarik Graba, Sumanta Chaudhuri
AXI3 WRITE RESPONSE CHANNEL

Diagram showing the AXI3 write response channel with signals:
- Information
- Handshake
- BID[3:0]
- BRESP[3:0]
- BREADY
- BVALID

Master and Slave sides are connected with these signals.
AXI3 WRITE RESPONSE CHANNEL

Possible values: okay, exokay, s1verr, decerr

BID[3:0] must match the request AWID

BVALID

BREADY
AXI3 WRITE BURST
AXI3 READ ADDRESS CHANNEL

- ARID[3:0]
- ARADDR[31:0]
- ARLEN[3:0]
- ARSIZE[2:0]
- ARBURST[1:0]
- ARLOCK[1:0]
- ARCACHE[1:0]
- AR PROT[1:0]
- ARVALID
- ARREADY
AXI3 READ RESPONSE CHANNEL

- MASTER
  - HANDSHAKE: RREADY
  - INFORMATION: RID[3:0], RDATA[31:0], RRESP[3:0], RLAST

- SLAVE
  - HANDSHAKE: RVALID
  - INFORMATION: RID[3:0], RDATA[31:0], RRESP[3:0], RLAST
AXI3 READ RESPONSE CHANNEL

Data width can be 1, 2, 4, 8, ..., 128 Bytes
Possible values: okay, exokay, slvrr, decrr
Denotes the last byte of a burst

must match the ARID of the request
AXI3 READ BURST
AXI3 READ BURST

Diagram showing the timing of AXI3 read burst signals, including ACLK, ARADDR, ARVALID, ARREADY, RDATA, RLAST, RVALID, and RREADY.
There must be no combinatorial paths between input and output signals on both master and slave interfaces.

combinatorial paths (Not Allowed)
The AXI protocol includes a single active LOW reset signal, ARESETn. The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of ACLK.

To avoid metastability associated with asynchronous de-assertion of Reset.
AXI HIGHLIGHTS : Ordering Rules

- Key to Out-of-Order Transactions Processing.
- At a master interface, read data with the same ARID value must arrive in the same order in which the master issued the addresses.
- In a sequence of read transactions with different ARID values, the slave can return the read data in a different order than that in which the transactions arrived.
AXI HIGHLIGHTS : Write Data Interleaving

- Write Data with different AWIDs can be interleaved.
- Not supported anymore in AXI4.
AXI HIGHLIGHTS : 4K Crossing

- Bursts must not cross 4KB boundaries.
- This is enforced so that a burst doesn’t cross over to another slave.
- The slave memory maps has to be aligned to 4KB boundaries.
AXI HIGHLIGHTS: Special Transactions

- Cache Related Transactions. Used to indicate cache allocate and write policy.

- **write policy**
  - write-back: Processor writes data to cache. Cache writes the data back to main memory, when the system bus is free.
  - write-through: The processor write is complete only when the data has reached both main memory and cache.

- **allocate policy**:
  - Cacheable
  - Bufferable.
  - Allocate/Don’t allocate a cache line.

- Total 16 different cache related special transactions.
  \[(A\times CACHE[3 :0]) \times=R D/W R\]
AXI HIGHLIGHTS : Special Transactions

- To indicate protected accesses. AxPROT[1 :0]

- Exclusive Access
  - implemented in the slave.
  - Signalled by AxLOCK[1 :0], BRESP[1 :0], RRESP[1 :0].
  - e.g an exclusive write is successful (denoted by EXOKAY) if no other master has written to the address space between previous read and the write to the same location.

- Locked Access
  - implemented in the network.
  - Signalled by AxLOCK[1 :0].
  - The network guarantees that only the master is allowed access to a region, until an unlocked access comes from the same master.
Features not supported in AXI4

- Locked transaction is no longer supported.
- WID signal disappears. Write Data interleaving is no longer supported.
Additional Features in AXI4

- Burstlength for incremental bursts can be 256 (16 for AXI3).
- Additional signal AxQOS. ($x=$RD/WR) 4 bit signal, higher value indicates higher priority
- Additional signal AxREGION. ($x=$RD/WR). Each memory mapped slave interface can be divided into regions (logical interfaces).
- Additional signal AxUSER. ($x=$RD/WR).
AXI/OCP Differences

- No Separate WR/RD channel like AXI.
- A common command channel which issue both write/read commands.
- OCP can have several of the command channels in parallel. For multi-threading.
- Supports posted writes. (i.e no response required)
## Quiz

**Is there a Bug?**

<table>
<thead>
<tr>
<th>ARID</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BURST LENGTH</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ADDR</td>
<td>0xFF8</td>
<td>0x400</td>
<td>0x800</td>
</tr>
<tr>
<td>VALID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDATA</td>
<td>0xFF8</td>
<td>0xFFC</td>
<td>0x1000</td>
</tr>
<tr>
<td>RVALID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Plan

Why Protocols?

Background
  Handshake
  Burst Signaling
  Blocking & Non-Blocking Transfer

AXI
  AXI Channels
  AXI Highlights
  AXI3/AXI4 Differences
  AXI/OCP Differences

Protocol Compliance

References
Executable Specifications.

- AXI specifications are available as a set of SystemVerilog Assertions.
- Can be instantiated in the netlist for protocol checking.
- Can be synthesized into FPGA/Emulators.
Constrained random verification.

- Use VIPs (Verification IPs).
- Generate random AXI Traffic from a BFM (Bus Functional Master) towards IP interface.
- Constrain the traffic according to IP specs.
- Functional Coverage.
  - Need to check all possible combination of valid transactions.
  - e.g. check burst length = (1, 2, 4, 8) in each page of 4K in a 1M memory space. Need to hit $4 \times 256$ different cases.
  - Need to check that no rule is violated for all combinations.
- No Tape-out without 100% coverage.
Constrained random verification.

Diagram:
- AXI VIP (Active BFM) -> DEVICE UNDER TEST
- Passive Protocol Checker
- AXI VIP
- Other Devices
Assertion based verification.

- Use ABVIPS (Assertion Based Verification IPs).
- Can be used in Formal Verification Tools. (e.g. Incisive Formal Verifier)
- In formal verification, functional coverage is 100% by definition.
Plan

Why Protocols?

Background
  Handshake
  Burst Signaling
  Blocking & Non-Blocking Transfer

AXI
  AXI Channels
  AXI Highlights
  AXI3/AXI4 Differences
  AXI/OCP Differences

Protocol Compliance

References
References

Amba specs.

Cadence amba vips.

Ocp-ip specs.