Parallel Communications
Parallel Communications

• AMBA – AXI
  • Advanced eXtensible Interface
  • Burst based protocol
  • 5 independent transaction channels
    • Read address
    • Read data
    • Write address
    • Write data
    • Write response
Parallel Communications

- AMBA – AXI
Parallel Communications

- AMBA – AXI interface

  - Interfaces defined between
    - Master and interconnect
    - Slave and Interconnect
    - Master and Slave

- The Interconnect is implementation specific
Parallel Communications

• AMBA – AXI system topologies

• Three common interconnect topologies
  • Shared address and data buses
  • Shared address buses and multiple data buses
  • Multilayer - multiple address and data buses.

• In most systems
  • Address channel bandwidth requirement is significantly less than data channel bandwidth requirement
  • Use a shared address bus with multiple data buses
    • Enable parallel data transfers
    • Balance between system performance and interconnect complexity
Parallel Communications

• AMBA – AXI system topologies
  • All channels are unidirectional
  • All channels are independent
    • No fixed timing requirement between channels
  • Allows for Register Slice insertion
    • Add registers into the channels to increase clock speeds
    • Cost is increased latencies
    • Provides a good solution for a mix of short and long paths
Parallel Communications

• AMBA – AXI Signaling
  • All 5 channels use a handshake process to ensure proper communication
  • Either source or destination can control rate of transfer
    • CLK
    • VALID – generated by the source
    • READY – Generated by the destination
  • Information can be address, data, control

• Transactions occurs on rising CLK edge with both VALID and READY high
Parallel Communications

- AMBA – AXI Signaling
  - Source – Destination ≠ Master – Slave

- Master = processor
- Slave = memory or peripheral
- Source = origin of information
- Destination = recipient of information

- Read
  - Master = Processor, Slave = Memory
  - Control and address: Source = Processor, Destination = Memory
  - Data: Source = Memory, Destination = Processor

- Write
  - Master = Processor, Slave = Memory
  - Control and address: Source = Processor, Destination = Memory
  - Data: Source = Processor, Destination = Memory
Parallel Communications

• AMBA – AXI Signaling

• Transactions are burst oriented

  • Beat - increment of data transfer – (# bytes in a word)

  • Burst – series of transfers in a fixed configuration

  • Master drives Control and Address of first byte location to Slave
  • Slave is responsible for calculating subsequent addresses for the transfer

** Bursts are not allowed to cross 4KB boundaries
This is also the minimum address space for a slave
Parallel Communications

• Read - 4 Beat burst
Parallel Communications

• Read - 3 beat burst
Parallel Communications

- Write – 4 beat burst
Parallel Communications

• AMBA – AXI Signaling - Address

• Address – physical addressing
  
  • Signals to indicate desired R/W address
  • Supports 32, 40, 44 and 48 bit addressing
  
  • AxADDR[n:0]
    • n = 32,40,44 or 48

ARADDR[ ] - Address channel Read Address
AWADDR[ ] - Address channel Write Address
AxADDR[ ] - Address channel Address – applies to both R/W
Parallel Communications

- **AMBA – AXI Signaling - Control**

- **Beat Size (Burst Size)** - increment of data transfer
  - # of bytes transferred in each transaction
  - Sizes from 1-128 bytes supported
    - 64 bit processor would use 8 byte beat size
    - Could hardwire bits if this never changed
  - Specified by signals on Address Read and Address Write (AR/AW)

- **AxSize[2:0]**

** ARM calls this the Burst Size

<table>
<thead>
<tr>
<th>AxSIZE[2:0]</th>
<th>Bytes in transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b000</td>
<td>1</td>
</tr>
<tr>
<td>0b001</td>
<td>2</td>
</tr>
<tr>
<td>0b010</td>
<td>4</td>
</tr>
<tr>
<td>0b011</td>
<td>8</td>
</tr>
<tr>
<td>0b100</td>
<td>16</td>
</tr>
<tr>
<td>0b101</td>
<td>32</td>
</tr>
<tr>
<td>0b110</td>
<td>64</td>
</tr>
<tr>
<td>0b111</td>
<td>128</td>
</tr>
</tbody>
</table>
Parallel Communications

• AMBA – AXI Signaling - Control

• Burst Length – number of Beats in a Burst
  • # of Beats transferred in each Burst
  • Sizes from 1 - 256 Beats supported
  • Specified by signals on Address Read and Address Write Channels

• AxLEN[7:0]
  • Length = AxLEN[7:0] + 1

** Lengths > 16 only allowed in INCR mode
Parallel Communications

• AMBA – AXI Signaling - Control

  • Burst Type – Type of Burst

    • Fixed – same address used for each Beat of the Burst
      • Typical for R/W to a FIFO

    • Incremental – address is incremented (by Slave) by Beat size for each Beat in the Burst
      • Typical for accessing sequential memory

    • Wrapping – Incremental burst with boundary wrapping
      • 4 byte beat, 8 beat burst, starting address 0x08
      • Burst would access addresses 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C, 0x00, 0x04
      • Starting address must be Beat aligned
      • Burst lengths of 2, 4, 8 and 16 allowed

  • AxBURST[1:0]
Parallel Communications

- AMBA – AXI Signaling - Control
  - Additional Controls
    - AxCACHE[3:0]
      - Bufferable – Data (W) may be buffered prior to reaching final destination
      - Modifiable –
        - Data may be allocated to a location other than the final destination
        - Transactions can be re-ordered
        - Transaction controls can be modified
      - Allocation – Indicates Data was allocated previously
    - AxPROT[2:0]
      - Indicates access protection to memory locations

<table>
<thead>
<tr>
<th>AxCACHE</th>
<th>Value</th>
<th>Transaction attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>0</td>
<td>Non-bufferable</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Bufferable</td>
</tr>
<tr>
<td>[1]</td>
<td>0</td>
<td>Non-cacheable</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Cacheable modificable</td>
</tr>
<tr>
<td>[2]</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Read-allocate</td>
</tr>
<tr>
<td>[3]</td>
<td>0</td>
<td>No Write-allocate</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Write-allocate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AxPROT</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>0</td>
<td>Unprivileged access</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Privileged access</td>
</tr>
<tr>
<td>[1]</td>
<td>0</td>
<td>Secure access</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Non-secure access</td>
</tr>
<tr>
<td>[2]</td>
<td>0</td>
<td>Data access</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Instruction access</td>
</tr>
</tbody>
</table>
Parallel Communications

- AMBA – AXI Signaling - Data
  - Data – implementation specific
    - Read or Write date
    - Typically 32, 64, 128 bits wide
    - Relationship between Data width and Beat Size
      - $x_{DATA}[n:0]$
        - $n = 32, 64, 128, ...$

  RDATA[ ] - Read Data
  WDATA[ ] - Write Data
Parallel Communications

• AMBA – AXI Signaling - Data

  • Last – indicates the last transfer in a burst
    • RLAST – Read channel Last signal
      • generated by slave
    • WLAST – Write channel last signal
      • generated by the master
Parallel Communications

• AMBA – AXI Signaling - Data

  • Relationship between Data width and Beat Size
    • 1 Byte Beat
    • 5 Beat Burst
    • 32 bit data bus

Byte lane shifts each transfer
Parallel Communications

• AMBA – AXI Signaling - Data

  • Relationship between Data width and Beat Size
    • 4 Byte Beat
    • 3 Beat Burst
    • 64 bit data bus
    • 1st address = 0x04

  • Byte lanes shift each transfer

![Diagram of Data Transfer](image-url)
Parallel Communications

• AMBA – AXI Signaling - Data

  • Relationship between Data width and Beat Size

    • 4 Byte Beat
    • 4 Beat Burst
    • INCR mode
    • 64 bit data bus
    • 1st address = 0x04

![Diagram showing data transfers and address bits.](image-url)
Parallel Communications

• AMBA – AXI Signaling - Data

  • Relationship between Data width and Beat Size
  
  • 4 Byte Beat
  • 4 Beat Burst
  • INCR mode
  • 64 bit data bus
  • 1st address = 0x07

Un-aligned transfer
Parallel Communications

- AMBA – AXI Signaling - Data

  - Relationship between Data width and Beat Size
    - 4 Byte Beat
    - 4 Beat Burst
    - WRAP mode
    - 64 bit data bus
    - 1\textsuperscript{st} address = 0x04
Parallel Communications

• AMBA – AXI Signaling - Data

• Relationship between Data width and Beat Size
  • 4 Byte Beat
  • 4 Beat Burst
  • Fixed mode
  • 64 bit data bus
  • 1st address = 0x04

![Diagram showing data transfers and address space]
Parallel Communications

• AMBA – AXI Signaling - Data

• Write Strobe
  • Indicates which data lane contains valid data
  • One strobe for each lane

• WSTRB[n], where n is the number of bytes in the data bus
Parallel Communications

• AMBA – AXI Signaling - Response

  • Read and Write transactions include a response

  • Read responses are part of the Read Data Channel – RRESP[1:0]

  • Write responses are on a separate Write Response Channel – BRESP[1:0]

• Four responses
  • OKAY
  • EXOKAY
  • SLVERR
  • DECERR

<table>
<thead>
<tr>
<th>RRESP[1:0]</th>
<th>BRESP[1:0]</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b00</td>
<td></td>
<td>OKAY</td>
</tr>
<tr>
<td>0b01</td>
<td></td>
<td>EXOKAY</td>
</tr>
<tr>
<td>0b10</td>
<td></td>
<td>SLVERR</td>
</tr>
<tr>
<td>0b11</td>
<td></td>
<td>DECERR</td>
</tr>
</tbody>
</table>
Parallel Communications

• AMBA – AXI Signaling - Response

  • OKAY
    • Success of a normal access
    • Failure of an exclusive access
    • Exclusive access to a slave that does not support exclusive access

  • EXOKAY
    • Success of an exclusive access

  • SLVERR
    • Unsuccessful transaction – indicated by slave
      • FIFO over/under run
      • Write to Read only location
      • ...

  • DECERR
    • Access to a non existent location
      • Typically handled by a dummy slave
Parallel Communications

• AMBA – AXI Signaling – Transaction Identifiers

  • There is no strict relationship between the 5 channels in AXI
    • A read does not need to wait for data before doing something else
    • A write can do something else while waiting for the slave to respond
      with a READY signal
    • Write data must remain in order with respect to write addresses

  • This allows for multiple ongoing transactions – but how do you
    keep them straight?

    • Transaction identifier

    • Typically support 4-8 bits of identifier
      • 16 – 256 ongoing transactions
Parallel Communications

- AMBA – AXI Signaling – Transaction Identifiers

- Rules
  - Transactions from different Masters have no order restriction
  - Transactions R/W with the same ID must remain in order
  - Transactions R – W – Resp with the same ID have no restrictions
  - Transactions with different IDs have no restrictions
  - Read and response transactions have no restrictions
  - Write Data must follow Write addresses

- Transactions from 1 Master to multiple slaves with the same transaction ID must remain in order
Parallel Communications

- **AMBA AXI Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARID</td>
<td>Master</td>
</tr>
<tr>
<td>ARADDR</td>
<td>Master</td>
</tr>
<tr>
<td>ARLEN</td>
<td>Master</td>
</tr>
<tr>
<td>ARSIZE</td>
<td>Master</td>
</tr>
<tr>
<td>ARBURST</td>
<td>Master</td>
</tr>
<tr>
<td>AROCLK</td>
<td>Master</td>
</tr>
<tr>
<td>ARCACHE</td>
<td>Master</td>
</tr>
<tr>
<td>ARPROM</td>
<td>Master</td>
</tr>
<tr>
<td>AROQOS</td>
<td>Master</td>
</tr>
<tr>
<td>ARREGION</td>
<td>Master</td>
</tr>
<tr>
<td>ARUSER</td>
<td>Master</td>
</tr>
<tr>
<td>ARVALID</td>
<td>Master</td>
</tr>
<tr>
<td>ARREADY</td>
<td>Slave</td>
</tr>
</tbody>
</table>

**Read Address**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWID</td>
<td>Master</td>
</tr>
<tr>
<td>AWADDR</td>
<td>Master</td>
</tr>
<tr>
<td>AWLEN</td>
<td>Master</td>
</tr>
<tr>
<td>AROSIZE</td>
<td>Master</td>
</tr>
<tr>
<td>AROBUST</td>
<td>Master</td>
</tr>
<tr>
<td>AROCLK</td>
<td>Master</td>
</tr>
<tr>
<td>ARCATCHE</td>
<td>Master</td>
</tr>
<tr>
<td>ARPROM</td>
<td>Master</td>
</tr>
<tr>
<td>ARQOS</td>
<td>Master</td>
</tr>
<tr>
<td>ARREGION</td>
<td>Master</td>
</tr>
<tr>
<td>ARUSER</td>
<td>Master</td>
</tr>
<tr>
<td>ARVALID</td>
<td>Master</td>
</tr>
<tr>
<td>ARREADY</td>
<td>Slave</td>
</tr>
</tbody>
</table>

**Write Address**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>WID</td>
<td>Master</td>
</tr>
<tr>
<td>WDATA</td>
<td>Master</td>
</tr>
<tr>
<td>WSTRB</td>
<td>Master</td>
</tr>
<tr>
<td>WLAST</td>
<td>Master</td>
</tr>
<tr>
<td>WUSER</td>
<td>Master</td>
</tr>
<tr>
<td>WVALID</td>
<td>Master</td>
</tr>
<tr>
<td>WREADY</td>
<td>Slave</td>
</tr>
</tbody>
</table>

**Read Data**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>RID</td>
<td>Slave</td>
</tr>
<tr>
<td>RDATA</td>
<td>Slave</td>
</tr>
<tr>
<td>RRESP</td>
<td>Slave</td>
</tr>
<tr>
<td>RUSER</td>
<td>Slave</td>
</tr>
<tr>
<td>RVALID</td>
<td>Slave</td>
</tr>
<tr>
<td>RREADY</td>
<td>Master</td>
</tr>
</tbody>
</table>

**Write Data**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWEI</td>
<td>Master</td>
</tr>
<tr>
<td>AWEADDR</td>
<td>Master</td>
</tr>
<tr>
<td>AWELE</td>
<td>Master</td>
</tr>
<tr>
<td>AWRSIZE</td>
<td>Master</td>
</tr>
<tr>
<td>AWRBURST</td>
<td>Master</td>
</tr>
<tr>
<td>AWRLOCK</td>
<td>Master</td>
</tr>
<tr>
<td>AWCACHE</td>
<td>Master</td>
</tr>
<tr>
<td>AWPROT</td>
<td>Master</td>
</tr>
<tr>
<td>AWQOS</td>
<td>Master</td>
</tr>
<tr>
<td>AWREGION</td>
<td>Master</td>
</tr>
<tr>
<td>AWUSER</td>
<td>Master</td>
</tr>
<tr>
<td>AWVALID</td>
<td>Master</td>
</tr>
<tr>
<td>AWRREADY</td>
<td>Slave</td>
</tr>
</tbody>
</table>
Parallel Communications

• AMBA – AXI Interconnect

• The interconnect ties Master to Slaves
  • 1::1
  • 1:: Many
  • Many :: 1
  • Many :: Many

• Addresses used to identify source/destination
• Transaction IDs appended with additional decode bits to differentiate Masters used to allow out of order operation
Parallel Communications

- AMBA – AXI Lite
  - All Burst Lengths are 1
  - All Data is full width 32b or 64b
  - All accesses are non-modifiable, non-bufferable, non-exclusive

<table>
<thead>
<tr>
<th>Global</th>
<th>Write address channel</th>
<th>Write data channel</th>
<th>Write response channel</th>
<th>Read address channel</th>
<th>Read data channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLK</td>
<td>AWVALID</td>
<td>WVALID</td>
<td>BVALID</td>
<td>ARVALID</td>
<td>RVALID</td>
</tr>
<tr>
<td>ARESETn</td>
<td>AWREADY</td>
<td>WREADY</td>
<td>BREADY</td>
<td>ARREADY</td>
<td>RREADY</td>
</tr>
<tr>
<td></td>
<td>AWADDR</td>
<td>WDATA</td>
<td>BRESP</td>
<td>ARADDR</td>
<td>RDATA</td>
</tr>
<tr>
<td></td>
<td>AWPROT</td>
<td>WSTRB</td>
<td>–</td>
<td>ARPROT</td>
<td>RRESP</td>
</tr>
</tbody>
</table>
Parallel Communications

• AMBA – ACE

• AXE Coherency Extensions
  • All Masters must see the same data
  • On a write to a location – all copies must be updated
  • Adds 3 channels + other signals + states
    • Snoop Address
    • Snoop Data
    • Snoop Response
  • Caches must support coherency extensions
Parallel Communications

• AMBA – ACE

• Cache Line states

  • Valid or in-valid
    • Data at this location is valid or not

  • Dirty or Clean
    • The data at this location has been modified since it was read in

  • Unique or shared
    • The data at this location is sharable or not
Parallel Communications

• AMBA – ACE

  • Read

    • Master issues a read to it’s cache at sharable address
    • Not in the local cache
    • Interconnect passes the request to other caches via the Snoop Address Channel

    • If any cache has the data (valid), it responds via the Snoop Response/Data Channel
    • Interconnect then provides the data to the Master (to be placed in cache locally)

    • If no copies are found Interconnect initiates a request to the next level of memory
Parallel Communications

• AMBA – ACE

• Write
  
  • Master issues a write to it’s cache at sharable address
  • Interconnect passes the request to other caches via the Snoop Address Channel
  
  • If any cache has the data (valid), it clears the line in cache and it responds via the Snoop Response Channel that the data has been removed (invalidated)
  • If one of the valid lines was dirty, a write back is initiated before clearing
  
  • If a write back is required, Interconnect performs the writeback prior to writing to the local cache