On-Chip Bus Protocols

National Chiao Tung University
Chun-Jen Tsai
05/3/2011
Popular On-Chip Bus Architecture

- On-chip bus architecture is one of the crucial components for platform-based SoC design
- Popular SoC bus architecture:
  - AMBA – ARM’s bus architecture
  - CoreConnect – IBM’s bus architecture (for PowerPC)
  - Wishbone – Common architecture for open source IP design
Advanced MCU Bus Architecture

- ARM’s open standard on-chip bus specification:

**AMBA AHB**
- High performance
- Pipelined operation
- Multiple bus masters
- Burst transfers
- Split transactions

**AMBA ASB**
- High performance
- Pipelined operation
- Multiple bus masters

**AMBA APB**
- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals
AHB Features

- AMBA High-performance Bus (AHB) implements the features required for high-performance, high clock frequency systems including:
  - Burst transfers
  - Split transactions
  - Single-cycle bus master handover
  - Single-clock edge operation
  - Wider data bus configurations (64/128 bits)
AHB Components (1/2)

- AHB Components
  - **Bus master**: able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

  Typical masters: MCU, DMA, DSP

  - **Bus slave**: responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

  Typical slaves: slow-speed device, register files
AHB Components (2/2)

- Arbiter and decoder:
  - **Arbiter**: ensures that only one bus master at a time is allowed to initiate data transfers. Any arbitration algorithm, such as *highest priority* or *fair* access can be implemented.

  - **Decoder**: is used to decode the address of each transfer and provides a select signal for the slave that is involved in the transfer.

A single centralized decoder is required in all AHB implementations.
AHB is a multiplexed bus with a centralized arbiter.
Overview of Bus Transactions

- Master asserts a bus request signal to the Arbiter
- Arbiter grants the bus to the Master
- Master starts transfer by driving the address and control signals
- Slave responds by sending the status signal
- Uses write-data bus for data transfer from Master to Slave
- Uses read-data bus for data transfer from Slave to Master
AMBA Signal Naming Convention

- First letter of the name indicates the bus type, e.g.
  - H: AHB signals
  - P: APB signals
- A lower case n indicates that the signal is active low
- Timing diagram convention:
### AHB Common Signals (1/2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td>Clock source</td>
<td>This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.</td>
</tr>
<tr>
<td>HRESETn</td>
<td>Reset controller</td>
<td>The bus reset signal is active LOW and is used to reset the system and the bus. This is the only active LOW signal.</td>
</tr>
<tr>
<td>HADDR[31:0]</td>
<td>Master</td>
<td>The 32-bit system address bus.</td>
</tr>
<tr>
<td>HTRANS[1:0]</td>
<td>Master</td>
<td>Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.</td>
</tr>
<tr>
<td>HWRITE</td>
<td>Master</td>
<td>When HIGH this signal indicates a write transfer and when LOW a read transfer.</td>
</tr>
<tr>
<td>HSIZE[2:0]</td>
<td>Master</td>
<td>Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit), ..., up to 1024 bits.</td>
</tr>
<tr>
<td>HBURST[2:0]</td>
<td>Master</td>
<td>Indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.</td>
</tr>
</tbody>
</table>
### AHB Common Signals (2/2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HWDATA[31:0]</strong></td>
<td>Master</td>
<td>The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits (can be extended) is recommended.</td>
</tr>
<tr>
<td><strong>HSELx</strong></td>
<td>Decoder</td>
<td>Each AHB slave has its own slave select signal, which indicates that the current transfer is for the selected slave. This signal is a combinatorial decode of the address bus.</td>
</tr>
<tr>
<td><strong>HRDATA[31:0]</strong></td>
<td>Slave</td>
<td>The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits (can be extended) is recommended.</td>
</tr>
<tr>
<td><strong>HREADY</strong></td>
<td>Slave</td>
<td>A HIGH <strong>HREADY</strong> signal indicates that a transfer has finished. Note: Slaves require <strong>HREADY</strong> as both input and output signal.</td>
</tr>
<tr>
<td><strong>HRESP[1:0]</strong></td>
<td>Slave</td>
<td>The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.</td>
</tr>
</tbody>
</table>
AHB Simple Transfer

- Each transfer consists of an address/control cycle and one or more data cycles:
AHB Transfer with Wait State

Address phase

Data phase

HCLK

HADDR[31:0]

Control

HWDATA[31:0]

HREADY

HRDATA[31:0]

Data (A)
AHB Multiple Transfer
AHB Transfer Types

- HTRANS[1:0] indicates the type of transfer:
  - IDLE: masters do not need data to be transferred
  - BUSY: allows bus masters to insert IDLE cycles in the middle of bursts of transfers
  - NONSEQ: The address and control signals are unrelated to the previous transfer
  - SEQ: the address is related to the previous transfer
# Transfer Type Example

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTRANS[1:0]</td>
<td>NONSEQ</td>
<td>BUSY</td>
<td>SEQ</td>
<td>SEQ</td>
<td>SEQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HADDR[31:0]</td>
<td>0x20</td>
<td>0x24</td>
<td>0x24</td>
<td>0x28</td>
<td>0x2C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HBURST[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HWDATA[31:0]</td>
<td></td>
<td>Data (0x20)</td>
<td>Data (0x24)</td>
<td>Data (0x28)</td>
<td>Data (0x2C)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HREADY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HRDATA[31:0]</td>
<td></td>
<td>Data (0x20)</td>
<td>Data (0x24)</td>
<td>Data (0x28)</td>
<td>Data (0x2C)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Burst Operations

- Burst transfer type is specified by the HBURST signal
- Bursts must not cross a 1kB address boundary
- There are two types of burst operations:
  - Incrementing bursts: sequential address accesses
  - Wrapping bursts: modulo-address sequential accesses

<table>
<thead>
<tr>
<th>HBURST[2:0]</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>SINGLE</td>
<td>Single transfer</td>
</tr>
<tr>
<td>001</td>
<td>INCR</td>
<td>Incrementing burst of unspecified length</td>
</tr>
<tr>
<td>010</td>
<td>WRAP4</td>
<td>4-beat wrapping burst</td>
</tr>
<tr>
<td>011</td>
<td>INCR4</td>
<td>4-beat incrementing burst</td>
</tr>
<tr>
<td>100</td>
<td>WRAP8</td>
<td>8-beat wrapping burst</td>
</tr>
<tr>
<td>101</td>
<td>INCR8</td>
<td>8-beat incrementing burst</td>
</tr>
<tr>
<td>110</td>
<td>WRAP16</td>
<td>16-beat wrapping burst</td>
</tr>
<tr>
<td>111</td>
<td>INCR16</td>
<td>16-beat incrementing burst</td>
</tr>
</tbody>
</table>
HBURST Examples (1/2)

- **T1**
  - HCLK
  - HTRANS[1:0]: NONSEQ
  - HADDR[31:0]: 0x38
  - HBURST[2:0]: WRAP
  - HWRITE
  - HSIZE[2:0]: CONTROL FOR BURST SIZE = WORD
  - HPROT[3:0]:
  - HWDATA[31:0]: DATA (0x38)
  - HREADY
  - HRDATA[31:0]: DATA (0x38)

- **T2**
  - HCLK
  - HTRANS[1:0]: SEQ
  - HADDR[31:0]: 0xC
  - HBURST[2:0]:
  - HWRITE
  - HSIZE[2:0]:
  - HPROT[3:0]:
  - HWDATA[31:0]: DATA (0x3C)
  - HREADY
  - HRDATA[31:0]: DATA (0x3C)

- **T3**
  - HCLK
  - HTRANS[1:0]: SEQ
  - HADDR[31:0]: 0xD
  - HBURST[2:0]:
  - HWRITE
  - HSIZE[2:0]:
  - HPROT[3:0]:
  - HWDATA[31:0]: DATA (0x30)
  - HREADY
  - HRDATA[31:0]: DATA (0x30)

- **T4**
  - HCLK
  - HTRANS[1:0]: SEQ
  - HADDR[31:0]: 0x10
  - HBURST[2:0]:
  - HWRITE
  - HSIZE[2:0]:
  - HPROT[3:0]:
  - HWDATA[31:0]: DATA (0x34)
  - HREADY
  - HRDATA[31:0]: DATA (0x34)
## HBURST Examples (2/2)

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTRANS[1:0]</td>
<td>NONSEQ</td>
<td></td>
<td>SEQ</td>
<td>SEQ</td>
<td>SEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HADDR[31:0]</td>
<td>0x38</td>
<td></td>
<td>0xC</td>
<td>0x40</td>
<td>0x44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HBURST[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INCR4</td>
<td></td>
</tr>
<tr>
<td>HWRITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Control for burst</td>
</tr>
<tr>
<td>HSIZE[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIZE = Word</td>
</tr>
<tr>
<td>HPROT[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HWDATA[31:0]</td>
<td></td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>HREADY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HRDATA[31:0]</td>
<td></td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
</tbody>
</table>
AHB Slave Selection Mechanism

- HSEL: slave select signals
Slave Transfer Response

- HRESP[1:0] and HREADY signals the following conditions:
  - OKAY: complete the transfer immediately
  - WAIT: insert one or more wait states to allow time to complete the transfer
  - ERROR: signal an error to indicate that the transfer has failed
  - RETRY: perform the transfer again
  - SPLIT and RETRY: delay the completion of the transfer, but allow the master and slave to back off the bus, leaving it available for other transfers
AHB Arbitration (Master) Signals

- **HBUSREQx**: $(x = 0 \sim 15)$
  - Used by a bus master to request access to the bus

- **HLOCKx**:
  - Telling the arbiter that the master is performing a number of indivisible transfers

- **HGRANTx**:
  - Generated by the arbiter; a master gains ownership of the address bus when $\text{HGRANTx}$ is HIGH and $\text{HREADY}$ is HIGH at the rising edge of $\text{HCLK}$

- **HMASTER[3:0]**:
  - Indicating which master is currently granted the bus using the $\text{HMASTER[3:0]}$ signals

- **HMASTLOCK**:
  - Indicating that the current transfer is part of a locked sequence

- **HSPLIT[15:0]**
  - The 16-bit *Split Complete* bus is used by a SPLIT-capable slave to indicate which bus master can complete a SPLIT transaction. This information is needed by the arbiter so that it can grant the master access to the bus to complete the transfer.
Arbitration Example

- HCLK
- HBUSREQ_M1
- HBUSREQ_M2
- HGRANT_M1
- HGRANT_M2
- HMASTER[3:0]
- HTRAN[1:0]
- HADDR[31:0]
- HBURST[2:0]
- HWRITE
- HSIZE[2:0]
- HPROT[3:0]
- HWDATA[31:0]
- HREADY
Centralized Arbiter Advantage

- Each master can start driving the ADDR bus before the GRANT signal is active
  - No tricky timing bugs that are common in a tri-state bus
Split Transactions

- **In AHB:**
  - Arbiter masks the request from the requesting master until the slave is ready and then sends a grant signal to the master.
  - Each master can have a single outstanding split transaction.
  - A slave can have multiple outstanding split transactions.
  - No need to record address and control signals.

- **Slave’s Responsibility:**
  - Signal a “SPLIT” response to the arbiter so that the BUS can be given to other masters.
  - Tracking outstanding requests and matching responses.
  - In-order or out-of-order completion of outstanding requests.
Narrow AHB Slave on Wide Bus

- A slave only accept transfers as wide as its natural interface (or an ERROR response will be issued)
- For input, multiple cycles is required to read HWDATA bus (with help from external logic)
- For output, external glue logic may
  - Replicate the data onto both halves of the wide bus
  - Use additional logic to drive only half of the bus (reduce power consumption)
LEON AHB Slave Interface

-- AHB slave inputs

```vhdl
type ahb_slv_in_type is record
  hsel : std_logic_vector(0 to NAHBSLV-1); -- slave select
  haddr : std_logic_vector(31 downto 0); -- address bus (byte)
  hwrite : std_ulogic; -- read/write
  htrans : std_logic_vector(1 downto 0); -- transfer type
  hsize : std_logic_vector(2 downto 0); -- transfer size
  hburst : std_logic_vector(2 downto 0); -- burst type
  hwdata : std_logic_vector(31 downto 0); -- write data bus
  hprot : std_logic_vector(3 downto 0); -- protection control
  hready : std_ulogic; -- transfer done
  hmaster : std_logic_vector(3 downto 0); -- current master
  hmastlock : std_logic; -- locked access
  hbsel : std_logic_vector(0 to NAHBCFG-1); -- bank select
  hcache : std_logic; -- cacheable
  hirq : std_logic_vector(NAHBIRQ-1 downto 0); -- interrupt
end record;
```

-- AHB slave outputs

```vhdl
type ahb_slv_out_type is record
  hready : std_ulogic; -- transfer done
  hresp : std_logic_vector(1 downto 0); -- response type
  hrdata : std_logic_vector(31 downto 0); -- read data bus
  hsplit : std_logic_vector(15 downto 0); -- split completion
  hcache : std_ulogic; -- cacheable
  hirq : std_logic_vector(NAHBIRQ-1 downto 0); -- interrupt
  hconfig : ahb_config_type; -- memory access reg.
  hindex : integer range 0 to NAHBSLV-1; -- diagnostic use only
end record;
```
LEON AHB Master Interface

-- AHB master inputs
type ahb_mst_in_type is record
    hgrant : std_logic_vector(0 to NAHBMST-1); -- bus grant
    hready : std_ulogic; -- transfer done
    hresp : std_logic_vector(1 downto 0); -- response type
    hrd ata : std_logic_vector(31 downto 0); -- read data bus
    hcache : std_ulogic; -- cacheable
    hirq : std_logic_vector(NAHBIQR-1 downto 0); -- interrupt
end record;

-- AHB master outputs
type ahb_mst_out_type is record
    hbusreq : std_ulogic; -- bus request
    hlock : std_ulogic; -- lock request
    htrans : std_logic_vector(1 downto 0); -- transfer type
    haddr : std_logic_vector(31 downto 0); -- address bus (byte)
    hwrite : std_ulogic; -- read/write
    hsize : std_logic_vector(2 downto 0); -- transfer size
    hburst : std_logic_vector(2 downto 0); -- burst type
    hprot : std_logic_vector(3 downto 0); -- protection control
    hwdata : std_logic_vector(31 downto 0); -- write data bus
    hirq : std_logic_vector(NAHBIQR-1 downto 0); -- interrupt
    hconfig : ahb_config_type; -- memory access reg.
    hindex : integer range 0 to NAHBMST-1; -- diagnostic use only
end record;
AHB Arbiter and Decoder Interfaces
APB Features

- AMBA Peripheral Bus (APB) is optimized for minimal power consumption and reduced interface complexity
- All devices on APB are slaves:
  - Unpipelined access
  - Zero-power interface during non-peripheral bus activity
  - Timing can be provided by decoding with strobe timing
  - Write data valid for the whole access
APB Bridge

- The APB bridge appears as a slave module which handles the bus handshake and control signal retiming on behalf of the local peripheral bus.
- The bridge provides latching of all address, data and control signals, as well as providing a second level of decoding to generate slave select signals for the APB peripherals.
# APB Signals

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK (bus clock)</td>
<td>The rising edge of PCLK is used to time all transfers on the APB</td>
</tr>
<tr>
<td>PRESETn</td>
<td>APB reset signal, normally connects to the system bus reset signal</td>
</tr>
<tr>
<td>PADDR[31:0]</td>
<td>APB address bus, up to 32-bits, driven by the peripheral bus bridge unit</td>
</tr>
<tr>
<td>PSELx</td>
<td>A signal from the secondary decoder, within the peripheral bus bridge unit, to select each peripheral bus slave x</td>
</tr>
<tr>
<td>PENABLE (APB strobe)</td>
<td>Timing signal for all accesses on APB. The rising edge of PENABLE signals the middle of an APB transfer (i.e. the 2nd cycle of transfer)</td>
</tr>
<tr>
<td>PWRITE</td>
<td>When HIGH this signal indicates an APB write access and when LOW a read access</td>
</tr>
<tr>
<td>PRDATA</td>
<td>APB read data bus, up to 32-bits, driven by the selected slave during read cycles (when PWRITE is LOW)</td>
</tr>
<tr>
<td>PWDATA</td>
<td>APB write data bus, up to 32-bit, driven by the peripheral bus bridge unit during write cycles (when PWRITE is HIGH)</td>
</tr>
</tbody>
</table>
APB State Diagram

- **IDLE**
  - PSELx = 0
  - PENABLE = 0

- **SETUP**
  - PSELx = 1
  - PENABLE = 0

- **ENABLE**
  - PSELx = 1
  - PENABLE = 1

Transfer paths:
- From IDLE to SETUP
- From SETUP to ENABLE
- From ENABLE to IDLE

No transfer paths:
- From IDLE to IDLE
- From SETUP to SETUP
- From ENABLE to ENABLE
APB Bridge Interface

System bus slave interface
Read data
Reset
Clock

APB bridge

PSEL1
PSEL2
... 
PSELn
PENABLE
PADDR
PWRITE
PWDATA

Selects
Strobe
Address and control
Write data
APB Data to AHB via Bridge
AHB Data to APB via Bridge
APB Slave Interface

- Select: PSELx
- Strobe: PENABLE
- Address and control: PADDR, PWRITE
- Reset: PRESETn
- Clock: PCLK
- Write data: PWDATA
- PRDATA
IBM CoreConnect On-Chip Bus

- Composed of three types of buses: Processor Local Bus (PLB), On-Chip Peripheral Bus (OPB), and Device Control Register (DCR) Bus
- No-fee, no-royalty license for industry usage of the specifications, PLB/OPB arbiters, Bridges, macro design files, bus toolkits, and test suites

<table>
<thead>
<tr>
<th>Performance Features</th>
<th>CoreConnect 32</th>
<th>CoreConnect 64</th>
<th>CoreConnect 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLB Width</td>
<td>32-Bit</td>
<td>64-Bit</td>
<td>128-Bit</td>
</tr>
<tr>
<td>Max Frequency</td>
<td>66 MHz</td>
<td>133 MHz</td>
<td>183 MHz *</td>
</tr>
<tr>
<td>Max Bandwidth</td>
<td>264 MB/s</td>
<td>800 MB/s</td>
<td>2.9 GB/s *</td>
</tr>
</tbody>
</table>
CoreConnect Architecture
Processor Local Bus

- Fully synchronous, supports up to 16 masters (configurable)
- 32-, 64-, and 128-bit architecture; extensible to 256-bit
- Separate read/write data buses
- Overlapped transfers
- High bandwidth capabilities:
  - Burst transfers
  - Pipelining
  - Split transactions
  - Overlapped arbitration
  - 4-level of bus request priorities for each masters
On-Chip Peripheral and DCR Buses

- **OPB**
  - Fully synchronous
  - 32-bit bus (both address and data)
  - Support single-cycle data transfers
  - Support multiple masters

- **DCR**
  - Daisy-chained connection of the CPU and devices
  - Fully synchronous movement of GPR data between CPU and slave logic
Arbiters & Bridges

- **PLB arbiter**
  - Arbitration for up to 8 PLB master devices
  - Includes watchdog timer and separate address, read data, and write data paths
  - Supports address pipelining

- **OPB arbiter**
  - Arbitration for up to 4 OPB master devices

- **PLB to OPB bridge**
  - PLB slave and OPB master device
  - Support dynamic bus sizing for OPB connections
  - Support burst reads and writes

- **OPB to PLB bridge**
  - PLB master and OPB slave device
  - 64-bit PLB master interface
Example: PowerPC 440-based SoC
PLB Macro (Interface Logic)

- Custom logics interconnected via PLB macro logics:
PLB Address & Transfer Qualifiers

- SYS_plbClk
- SYS_plbReset
- PLB_MnAddrAck
- PLB_MnRearbitrate
- PLB_MnBusy
- PLB_MnErr
- PLB_pendReq
- PLB_pendPri(0:1)
- PLB_reqPri(0:1)

Master Interface

Request Qualifiers

- Mn_request
- Mn_priority(0:1)
- Mn_busLock
- Mn_RNW
- Mn_BE(0:3)
- Mn_size(0:3)
- Mn_type(0:2)
- Mn_compress
- Mn.guarded
- Mn_ordered
- Mn_lockErr
- Mn_abort
- Mn_ABus(0:31)
PLB Transfer Protocol (1/2)

- A PLB bus transaction is grouped under an address cycle and a data cycle

- **Address cycle:**
  - Master drives address and transfer qualifier signals (Request)
  - Arbiter grants the bus ownership, presents the signals to the slave (Transfer)
  - Slave latches the address and transfer qualifiers (Address-Acknowledge)
PLB Transfer Protocol (2/2)

- Data cycle: for each data beat,
  - Master drives the write data bus for a write transfer or sample the read data bus for a read transfer (Transfer)
  - Data acknowledge signals are required after the beat
PLB Transfer Protocol Example
Example: High Bandwidth Systems

- CoreConnect provides many macro logics for various architecture design:

```
Master A -> PLB Arbiter 1 -> PLB Cross-Bar Switch Macro
  \        \                               /  \     \     \
  \        \                  Slave Bus 1 /     \ Slave Bus 2
Master B -->                  Slave Bus 1       Slave Bus 2
  \        \                               /  \     \
  \        \                  Slave Bus 1 /     \ Slave Bus 2
Master C --> PLB Arbiter 2    Slave A      Slave B
  \        \                               /  \     \
  \        \                  Slave Bus 1 /     \ Slave Bus 2
Master D -->                  Slave C      Slave D
```

```
OPB Implementation

- OPB uses distributed multiplexer bus implementation:
DCR Bus

- DCR bus only requires slow data throughput, a ring-type data bus is typically used.
Wishbone Bus

- **Wishbone**
  - An open bus architecture for many open-source IP projects

- **Features:**
  - One bus architecture for all applications (on-chip/off-chip)
  - Supports point-to-point, shared bus, crossbar switch, and switched fabric interconnections
  - 64-bit address space, 8~64-bit data bus (expandable)
  - Handshaking protocol allows each IP core to throttle its data transfer speed
  - Support single read and write cycles, and RMW cycles
  - User defined TAGs for identifying data transfer types
  - Support synchronous design for off-chip bus
Wishbone Logical Architecture

- Wishbone provides a simple, compact architecture
Master/Slave Direct Interfacing

- Wishbone permits point-to-point direct master/slave interconnection:
- SYSCON module on a Wishbone bus generates clock and reset signals
TAG Signals & Strobe Signals

- There are three types of Tag signals: Data Tag, Address Tag, and Cycle Tag
  - Timing of the tag signals are defined in the Wishbone spec.; the semantics are defined in user IP datasheet
- The strobe output [STB_O] from the master logic indicates a valid data transfer cycle; It is used to qualify various other signals on the interface such as [SEL_O()]

Example: Data Tag Signals

- Input data tag type [TGD_I()] is used on MASTER and SLAVE interfaces. It contains information (e.g. parity, error correction, time stamp) that is associated with the data input array [DAT_I()], and is qualified by signal [STB_I]
- Output data tag works similarly
- The name and operation of a data tag must be defined in the user Wishbone datasheet
Crossbar/Data Flow Connections

Crossbar interconnect

Data-flow interconnect

DIRECTION OF DATA FLOW
Handshaking Protocols

- All bus cycles use a handshaking protocol between the MASTER and SLAVE interfaces.
- MASTER asserts [STB_O] when it is ready to transfer data; [STB_O] remains asserted until the SLAVE asserts one of the cycle terminating signals [ACK_I], [ERR_I] or [RTY_I].
- At every rising edge of [CLK_I] the terminating signal is sampled to see if [STB_O] should be negated.

![Diagram of handshaking protocol]
Single R/W

- Left: Read
- Right: Write
Block R/W

- the [CYC_O] signal is asserted for the duration of a BLOCK transfer cycle. To hold the access until the end of the cycle the [LOCK_O] signal must be asserted. During each of the data transfer phases (within the block transfer), the handshaking protocol between [STB_O] and [ACK_I] is maintained.
The RMW (read-modify-write) cycle is used for indivisible semaphore operations. During the first half of the cycle a single read data transfer is performed. During the second half of the cycle a write data transfer is performed.
“Regular” Shared Bus Request

- For a shared bus interconnect, bus request/grant protocols must be used:
  - MASTERS request the bus by asserting their [CYC_O] signals
  - One clock edge after the assertion of a [CYC_O] signal the arbiter grants the bus to one of the MASTERS that requested it by asserting grant lines GNT(N..0)