Introductions & Sources

- We will consider a number of issues related to bus architectures in digital systems.
- Useful references:
  - AMBA™ Specification (Rev 2.0), ARM Ltd., 1999
  - Draft Chapter, “System-on-Chip”, Flynn & Luk

Basic concepts:

- Buses operate in units of cycles, Messages and transactions.
  - Cycles: A message requires a number of clock cycles to be sent from sender to receiver over the bus.
  - Message: These are logical unit of information. For example, a write message contains an address, control signals and the write data.
  - Transaction: A transaction consists of a sequence of messages which together form a transaction. For example, a memory read requires a memory read message and a reply with the requested data.
Synchronous vs Asynchronous

**Basic concepts:**

- **Typical Source Synchronous Data Transfer**
  - **Clock**
  - **Ctrl**
  - **Strobe 0**
  - **Data [15:0]**
  - **Strobe 1**
  - **Data [31:16]**

1. Master puts address on bus and asserts **READ** when address is stable
2. Memory puts data on bus and asserts **ACK** when data is stable
3. Master deasserts **READ** when data is ready
4. Memory deasserts **ACK**

Basic concepts:

**Bus arbitration**

- Only one bus master can control the bus.
- Need some way of deciding who is master – may use a bus arbiter:

Basic concepts:

**Bus pipelining**

- A transaction may take multiple cycles
- Overlap multiple transactions through pipelining:

<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>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Read</td>
<td>AR</td>
<td>AR</td>
<td>AG</td>
<td>RQ</td>
<td>P</td>
<td>RPLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Write</td>
<td>AR</td>
<td>AR</td>
<td>AG</td>
<td>Stall</td>
<td>Stall</td>
<td>RQ</td>
<td>ACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Write</td>
<td>AR</td>
<td>AR</td>
<td>AG</td>
<td>Stall</td>
<td>Stall</td>
<td>AG</td>
<td>Stall</td>
<td>RQ</td>
<td>ACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Read</td>
<td></td>
<td>AR</td>
<td>Stall</td>
<td>Stall</td>
<td>ARB</td>
<td>Stall</td>
<td>AG</td>
<td>Stall</td>
<td>RQ</td>
<td>P</td>
<td>RPLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Read</td>
<td>AR</td>
<td>Stall</td>
<td>ARB</td>
<td>Stall</td>
<td>AG</td>
<td>RQ</td>
<td>P</td>
<td>RPLY</td>
<td>RQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Read</td>
<td></td>
<td>AR</td>
<td>Stall</td>
<td>ARB</td>
<td>AG</td>
<td>Stall</td>
<td>Stall</td>
<td>Stall</td>
<td>RQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus busy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basic concepts: Split-transaction bus

- A bus transaction can be divided into two or more phases, e.g.
  - “Request” phase
  - “Reply” phase
- These can be split into two separate sub-transactions, which may or may not happen consecutively. If split, these must compete for the bus by arbitration.

Basic concepts: Pipelined only bus vs split-transaction bus

<table>
<thead>
<tr>
<th>Pipelined Bus</th>
<th>Split-Transaction Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Trans</td>
<td>RQ A</td>
</tr>
<tr>
<td>2. Trans</td>
<td>RQ B</td>
</tr>
<tr>
<td>3. Trans</td>
<td>RQ C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Request</th>
<th>Burst Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARB</td>
<td>Cmd</td>
</tr>
<tr>
<td>ARB</td>
<td>Adr</td>
</tr>
<tr>
<td>ARB</td>
<td>Data</td>
</tr>
<tr>
<td>ARB</td>
<td>Data</td>
</tr>
<tr>
<td>ARB</td>
<td>Data</td>
</tr>
<tr>
<td>ARB</td>
<td>Data</td>
</tr>
</tbody>
</table>

Burst transfer mode

<table>
<thead>
<tr>
<th>Arbitration</th>
<th>Message A</th>
<th>Message B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cmd</td>
<td>Adr</td>
<td>Data</td>
</tr>
<tr>
<td>Cmd</td>
<td>Adr</td>
<td>Data</td>
</tr>
<tr>
<td>Cmd</td>
<td>Adr</td>
<td>Data</td>
</tr>
<tr>
<td>Cmd</td>
<td>Adr</td>
<td>Data</td>
</tr>
<tr>
<td>Cmd</td>
<td>Adr</td>
<td>Data</td>
</tr>
</tbody>
</table>

Basic concepts: Burst transfer mode
### Bus Bandwidth

<table>
<thead>
<tr>
<th>Bus</th>
<th>Width (bits)</th>
<th>Bus Speed (MHz)</th>
<th>Bus Bandwidth (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit ISA</td>
<td>8</td>
<td>8.3</td>
<td>7.9</td>
</tr>
<tr>
<td>16-bit ISA</td>
<td>16</td>
<td>8.3</td>
<td>15.9</td>
</tr>
<tr>
<td>EISA</td>
<td>32</td>
<td>8.3</td>
<td>21.8</td>
</tr>
<tr>
<td>VL6</td>
<td>32</td>
<td>33</td>
<td>127.2</td>
</tr>
<tr>
<td>PCI</td>
<td>32</td>
<td>33</td>
<td>127.2</td>
</tr>
<tr>
<td>PCI 2.1</td>
<td>64</td>
<td>66</td>
<td>508.6</td>
</tr>
<tr>
<td>AGP</td>
<td>32</td>
<td>66</td>
<td>254.3</td>
</tr>
<tr>
<td>AGP (x2 mode)</td>
<td>32</td>
<td>66x2</td>
<td>508.6</td>
</tr>
<tr>
<td>AGP (x4 mode)</td>
<td>32</td>
<td>66x4</td>
<td>1,017.3</td>
</tr>
</tbody>
</table>

### Bus Hierarchy

- Based around ARM processor
  - AHB – Advanced High-Performance Bus
    - Pipelining of Address / Data
    - Split Transactions
    - Multiple Masters
  - APB – Advanced Peripheral Bus
    - Low Power / Bandwidth Peripheral Bus

### AMBA Bus

- Encourages modular design and design reuse
- Well defined interface protocol, clocking and reset
- Low-power support (helped by two-level partitioning)
- On-chip test access – built-in structure for testing modules connected on the bus

### AMBA Bus Design Goals

- Transactions on AHB
  - Bus master obtain access to the bus
  - Bus master initiates transfer
  - Bus slave provides response

- Pipelining of Address / Data
- Split Transactions
- Multiple Masters

- Low Power / Bandwidth Peripheral Bus
AMBA bus arbitration

Simple AHB Transfer

AHB Transfer with wait states

Multiple transfers with Pipelining
Burst mode transfer (undefined length)

Slave Transfer Responses

- 1. Complete transfer immediately (single cycle transfer)
- 2. Insert one or more wait states to allow completion
- 3. Signal error to indicate transfer failed
- 4. Back off from the bus, try later (RE-TRY or SPLIT responses)

Slave Transfer Responses

- 1. Complete transfer immediately (single cycle transfer)
- 2. Insert one or more wait states to allow completion
- 3. Signal error to indicate transfer failed
- 4. Back off from the bus, try later (RE-TRY or SPLIT responses)

Retry Responses on the AHB bus

Advanced Peripheral Bus (APB)
### IBM CoreConnect Bus

- SRAM/ROM Peripheral Controller
- External Bus Master Controller
- PC
- UART
- USB
- GPIO
- OPB Arbiter
- OPB Bridge
- DMA Controller
- MAL
- Device Control Register Bus
- Processor Local Bus (PLB) 128-bit
- PPC440 CPU
- Inst. Data
- PLB Arbiter
- PC133/DDR133 SDRAM Controller
- PCLX Bridge
- SRAM Controller
- Custom Logic
- Reset Clock Control Power Mgmt
- 10/100 Ethernet

### CoreConnect vs AMBA

<table>
<thead>
<tr>
<th>IBM CoreConnect</th>
<th>ARM AMBA 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Local Bus</td>
<td>AMBA High-performance Bus</td>
</tr>
<tr>
<td>Bus Architecture</td>
<td>32-, 64-, and 128-bits</td>
</tr>
<tr>
<td>Data Buses</td>
<td>Separate Read and Write</td>
</tr>
<tr>
<td>Key Capabilities</td>
<td>Separate Read and Write</td>
</tr>
<tr>
<td>Masters Supported</td>
<td>Multiple Bus Masters</td>
</tr>
<tr>
<td>Bridge Function</td>
<td>Single Master: The APB Bridge</td>
</tr>
<tr>
<td>Data Buses</td>
<td>Multiple Bus Masters</td>
</tr>
</tbody>
</table>

#### Crossbar Switch Approach
- Uses asynchronous channels
- Different modules can run at different clock frequencies
- Globally Asynchronous, Locally Synchronous (GALS) system

#### Network-on-chip approach
- Array of tiles
- Each tile contains client logic and router logic
- 2-D mesh topology
- Uses data packets, not wires, for communication
- Predictable delay, and noise