PCI
Older PC Implementations
Newer PC Implementations
Server Platforms
PCI Overview
PCI Features

- 32-Bit or 64-Bit address and data
- 66 or 33 down to 0 MHz synchronous operation
- Single or multiple bus masters
- Reflected bus signaling
- Stepped signaling
- Bus parity error reporting
- 5 or 3.3 volt operation
- Cache support
- JTAG testing
PCI Bus

- Bus Signals
- Bus Commands
- Bus Transactions
- Arbitration
# PCI Bus Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Lines</th>
<th>Master</th>
<th>Slave</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>1</td>
<td></td>
<td></td>
<td>Clock (33 MHz or 66 MHz)</td>
</tr>
<tr>
<td>AD</td>
<td>32</td>
<td>×</td>
<td>×</td>
<td>Multiplexed address and data lines</td>
</tr>
<tr>
<td>PAR</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Address or data parity bit</td>
</tr>
<tr>
<td>C/BE</td>
<td>4</td>
<td></td>
<td>×</td>
<td>Bus command/bit map for bytes enabled</td>
</tr>
<tr>
<td>FRAME#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Indicates that AD and C/BE are asserted</td>
</tr>
<tr>
<td>IRDY#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Read: master will accept; write: data present</td>
</tr>
<tr>
<td>IDSEL</td>
<td>1</td>
<td></td>
<td>×</td>
<td>Select configuration space instead of memory</td>
</tr>
<tr>
<td>DEVSEL#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Slave has decoded its address and is listening</td>
</tr>
<tr>
<td>TRDY#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Read: data present; write: slave will accept</td>
</tr>
<tr>
<td>STOP#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Slave wants to stop transaction immediately</td>
</tr>
<tr>
<td>PERR#</td>
<td>1</td>
<td></td>
<td></td>
<td>Data parity error detected by receiver</td>
</tr>
<tr>
<td>SERR#</td>
<td>1</td>
<td></td>
<td></td>
<td>Address parity error or system error detected</td>
</tr>
<tr>
<td>REQ#</td>
<td>1</td>
<td></td>
<td></td>
<td>Bus arbitration: request for bus ownership</td>
</tr>
<tr>
<td>GNT#</td>
<td>1</td>
<td></td>
<td></td>
<td>Bus arbitration: grant of bus ownership</td>
</tr>
<tr>
<td>RST#</td>
<td>1</td>
<td></td>
<td></td>
<td>Reset the system and all devices</td>
</tr>
</tbody>
</table>
## PCI Bus Signals (cont’d)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Lines</th>
<th>Master</th>
<th>Slave</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ64#</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Request to run a 64-bit transaction</td>
</tr>
<tr>
<td>ACK64#</td>
<td>1</td>
<td></td>
<td>×</td>
<td>Permission is granted for a 64-bit transaction</td>
</tr>
<tr>
<td>AD</td>
<td>32</td>
<td>×</td>
<td></td>
<td>Additional 32 bits of address or data</td>
</tr>
<tr>
<td>PAR64</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Parity for the extra 32 address/data bits</td>
</tr>
<tr>
<td>C/BE#</td>
<td>4</td>
<td>×</td>
<td></td>
<td>Additional 4 bits for byte enables</td>
</tr>
<tr>
<td>LOCK</td>
<td>1</td>
<td>×</td>
<td></td>
<td>Lock the bus to allow multiple transactions</td>
</tr>
<tr>
<td>SBO#</td>
<td>1</td>
<td></td>
<td></td>
<td>Hit on a remote cache (for a multiprocessor)</td>
</tr>
<tr>
<td>SDONE</td>
<td>1</td>
<td></td>
<td></td>
<td>Snooping done (for a multiprocessor)</td>
</tr>
<tr>
<td>INTx</td>
<td>4</td>
<td></td>
<td></td>
<td>Request an interrupt</td>
</tr>
<tr>
<td>JTAG</td>
<td>5</td>
<td></td>
<td></td>
<td>IEEE 1149.1 JTAG test signals</td>
</tr>
<tr>
<td>M66EN</td>
<td>1</td>
<td></td>
<td></td>
<td>Wired to power or ground (66 MHz or 33 MHz)</td>
</tr>
</tbody>
</table>
PCI bus access

- PCI is a Multimaster Bus
- All transactions *initiated* by a master
- All transactions to/from a *target*
PCI Bus Control Signals

- **FRAME#**
  - driven by master to indicate transfer start and end

- **IRDY#**
  - driven by master to indicate it is ready to transfer data

- **TRDY#**
  - driven by target to indicate it is ready to transfer data
Bus transaction start
## PCI Command Definition

<table>
<thead>
<tr>
<th>C/BE[3:0]#</th>
<th>Command Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt acknowledge</td>
</tr>
<tr>
<td>0001</td>
<td>Special Cycle</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Read</td>
</tr>
<tr>
<td>0011</td>
<td>I/O Write</td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read</td>
</tr>
<tr>
<td>0111</td>
<td>Memory Write</td>
</tr>
<tr>
<td>1010</td>
<td>Configuration Read</td>
</tr>
<tr>
<td>1011</td>
<td>Configuration Write</td>
</tr>
<tr>
<td>1100</td>
<td>Memory Read Multiple</td>
</tr>
<tr>
<td>1101</td>
<td>Dual Address Cycle</td>
</tr>
<tr>
<td>1110</td>
<td>Memory Read Line</td>
</tr>
<tr>
<td>1111</td>
<td>Memory Write and Invalidate</td>
</tr>
</tbody>
</table>
PCI Bus Read
Target Read Transaction

- Initiator asserts FRAME# to start the transaction.
- Target inserts a wait state.
- Initiator acknowledges termination request.
- Target requests termination.
PCI Bus Write
**Basic Write Transaction**

- **CLK**: Clock signal.
- **AD[31:0]**: Address bus.
- **C/BE#[3:0]**: Chip Enable/Byte Enable bus.
- **FRAME#**: Transaction Frame signal.
- **IRDY#**: Initiator Ready signal.
- **TRDY#**: Target Ready signal.
- **STOP#**: Stop signal.
- **DEVSEL#**: Device Selection signal.

**Key Points**:
- **Initiator asserts FRAME#** to start the transaction.
- **Data is transferred on any clock edge** where both IRDY# & TRDY# are asserted.
- **Initiator deasserts FRAME#** to signal the final data phase; the transaction completes when the last piece of data is transferred.
PCI Arbitration
Centralized Arbitration
PCI Bus Arbitration
Arbitration

- Arbitration is access based
  - Master must arbitrate for each bus access
- Central arbitration scheme
  - Each master has a unique request and grant signal
- Arbitration is hidden
  - Occurs during previous bus cycle
Bus Parking

- Parking permits the arbiter to select an agent, by asserting its GNT#, when no other agent is using or requesting the bus
- The arbiter determines how this selection is made
  - Fixed, Last Used, ..., or None
PCI Hardware
PCI Card Connectors

5V 32-Bit Connector

5V 64-Bit Connector

3.3V 32-Bit Connector

3.3V 64-Bit Connector

Keyway

External Connector (Bezel) edge
5 V To 3.3 V Migration Path

- 5 Volt Card
- Dual-Voltage Card
- 3 Volt Card

- 5 Volt System
- Key away from backpanel
- 3 Volt System
- Key near backpanel

External Connector (Bezel) edge
PCI Configuration
System Initialization

- Configuration allows software (BIOS) to initialize the system
- Each device has configuration registers
- At power up software scans bus(es)
- Software analyses system requirements
- Configuration registers are set to configure individual devices
Configuration Types

- Specific bus commands
  - configuration read \( (C/BE# = 1010) \)
  - configuration write \( (C/BE# = 1011) \)

- Type 0
  - local PCI bus
    - IDSEL line indicates device
    - address field indicates register

- Type 1
  - remote PCI bus (through bridge)
    - address field indicates bus, device and register
### Configuration Space Header

| Device ID | Status | Vendor ID | Command | Class Code | Rev | Cache Line Size | Header Type | Latency Timer | Base Address 0 | Base Address 1 | Base Address 2 | Base Address 3 | Base Address 4 | Base Address 5 | Cardbus CIS Pointer | Subsystem ID | Subsystem Vendor ID | Expansion ROM Base Address | Max_Lat | Min_Gnt | Interrupt Pin | Interrupt Line |
|-----------|--------|-----------|---------|------------|-----|----------------|-------------|---------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------------|-------------|-------------------|--------------------------|----------|--------|--------------|------------------|
|           |        |           |         |            |     |                |             |               | 10             | 14             | 18             | 1C              | 20              | 24              | 28                | 30                   | 2C         | 30                | 34                      | 38       | 3C               |              |                    |
Figure: Type 0 PCI Configuration Cycle

Figure: Type 1 PCI Configuration Cycle
PCI Configuration Ports on PC

- All x86 based PCs use the IO Bus to support an address and data port for PCI configuration
  - Address port is 4 bytes wide located at 0x0CF8
  - Data port is 4 bytes wide located at 0x0CFC
- Configuration cycles first write (IO IN) a canonical address to the address port
  - A read (IO OUT) from the data port will now return the 4 byte config register(s) addressed
  - A write (IO IN) to the data port will now set the addressed configuration register(s)
Base Address for PCI Memory Space

TYPE  00  locate anywhere in lower 4GB
      01  locate beyond 1MB
      10  locate anywhere beyond 4GB
      11  reserved

Base Address for PCI I/O Space
BAR Management

- Write all 1s to a BAR and then read
  - If return value is all 0s then not used
  - If not zero, then check least sig bit
    - If 1 then IO assignment
    - If 0 then memory assignment
    - Bit position of least sig bit is used for size determination (i.e. if bit 6 is on decoder requires $2^6$ or 64 bytes of space)
BAR Management (cont’d)

- If all 1s are written to a BAR and the return value is 0xFFFF0000
  - The BAR is a memory decoder
  - The BAR is not prefetchable
  - The BAR requires an address < 4GB
  - The BAR requires $2^{16}$ or 64 KB of space
- 0xFFFF0008 as above but prefetchable
- 0xFFFF000C prefetchable and 64 bit
BAR Management (cont’d)

- If all 1s are written to a BAR and the return value is 0xFFFFF001
  - The BAR is an IO decoder
  - The BAR requires an address < 64 KB
  - The BAR requires $2^{12}$ or 4 KB of space
- Minimum memory size is 16 bytes
- Minimum IO size is 4 bytes