PCI Tutorial
Agenda

PCI Fundamentals
- PCI Local Bus Architecture
- PCI Signals
- Basic Bus Operations
- PCI Addressing and Bus Commands
- PCI Configuration
- Electrical and Timing Specifications
- 64-bit Extension
- 66-MHz Overview
- PCI Variations

Xilinx PCI Solution
- The PCI Challenge
- Xilinx PCI with Design Examples
- Xilinx PCI Design Flow Overview
- Available Resources
PCI Fundamentals and Concepts
PCI Local Bus Architecture
Overview of the PCI Specification

- The PCI Local Bus Specification covers many different requirements for PCI compliance
Basic Bus Architecture

Diagram showing the components of a basic bus architecture:
- Processor System
- Host Bridge
- Expansion Bus Bridge
- PCI Local Bus
- SCSI Controller
- 100 Mbit Ethernet
- LAN
- 3D Graphics Card
- 56K Modem
- 3D Sound Card
- MPEG Video Capture Card
- Add-in Cards

Motherboard
Sample Transaction – PIO

Processor System

Host Bridge

Expansion Bus Bridge

SCSI Controller

100 Mbit Ethernet

Motherboard

ISA

LAN

PCI Local Bus

56K Modem

MPEG Video Capture Card

3D Graphics Card

Programmable I/O Type Interface

Target

3D Sound Card

Sample Transaction – PIO

Processor System

Host Bridge

Expansion Bus Bridge

SCSI Controller

100 Mbit Ethernet

Motherboard

ISA

LAN

PCI Local Bus

56K Modem

MPEG Video Capture Card

3D Graphics Card

Programmable I/O Type Interface

Target

3D Sound Card
Sample Transaction – DMA

- Processor System
- Host Bridge
- Expansion Bus Bridge
- SCSI Controller
- 100 Mbit Ethernet
- 3D Graphics Card
- 3D Sound Card
- 56K Modem
- MPEG Video Capture Card
- LAN

Use PIO to set up DMA engine
Use DMA for image transfer
Key Terms

- **Initiator**
  - Or Master
  - Owns the bus and initiates the data transfer
  - Every Initiator must also be a Target

- **Target**
  - Or Slave
  - Target of the data transfer (read or write)

- **Agent**
  - Any initiator/target or target on the PCI bus
PCI Signals
Clock and Reset

- **CLK**
  - PCI input clock
  - All signals sampled on rising edge
  - 33MHz is really 33.33333MHz (30ns clk. period)
  - The clock is allowed to vary from 0 to 33 MHz
    - The frequency can change “on the fly”
    - Because of this, no PLLs are allowed

- **RST#**
  - Asynchronous reset
  - PCI device must tri-state all I/Os during reset
Transaction Control
Initiator Signals

- **FRAME# – I/O**
  - Signals the start and end of a transaction

- **IRDY# – I/O**
  - “I-Ready”
  - Assertion by initiator indicates that it is ready to send or receive data
Transaction Control
Target Signals

- **DEVSEL# – I/O**
  - Device select
  - Part of PCI’s distributed address decoding
    - Each target is responsible for decoding the address associated with each transaction
    - When a target recognizes its address, it asserts DEVSEL# to claim the corresponding transaction
Transaction Control
Target Signals

- **TRDY# – I/O**
  - “T-Ready”
  - When the target asserts this signal, it tells the initiator that it is ready to send or receive data

- **STOP# – I/O**
  - Used by target to indicate that it needs to terminate the transaction
Transaction Control
Configuration Signals

- Uses the same signals as the target, plus . . .
- IDSEL – I
  - “ID-Sel”
  - Individual device select for configuration – one unique IDSEL line per agent
  - Solves the “chicken-and-egg” problem
    - Allows the system host to configure agents before these agents know the PCI addresses to which they must respond
Address and Data Signals

- **AD[31:0] – I/O**
  - 32-bit address/data bus
  - *PCI is little endian (lowest numeric index is LSB)*

- **C/BE#[3:0] – I/O**
  - 4-bit command/byte enable bus
  - * Defines the PCI command during address phase*
  - *Indicates byte enable during data phases*
    - Each bit corresponds to a “byte-lane” in AD[31:0] – for example, C/BE#[0] is the byte enable for AD[7:0]
Address and Data Signals

- **PAR – I/O**
  - *Parity bit*
  - *Used to verify correct transmittal of address/data and command/byte-enable*
  - *The XOR of AD[31:0], C/BE#[3:0], and PAR should return zero (even parity)*
    - In other words, the number of 1’s across these 37 signals should be even
Arbitration Signals

- For initiators only!
- **REQ# – O**
  - Asserted by initiator to request bus ownership
  - Point-to-point connection to arbiter – each initiator has its own REQ# line
- **GNT# – I**
  - Asserted by system arbiter to grant bus ownership to the initiator
  - Point-to-point connection from arbiter – each initiator has its own GNT# line
Error Signals

- **PERR# – I/O**
  - Indicates *that a data parity error has occurred*
  - An agent that can report parity errors can have its PERR# turned off during PCI configuration

- **SERR# – I/O**
  - Indicates a *serious system error has occurred*
    - Example: Address parity error
  - May invoke NMI (non-maskable interrupt, i.e., a restart) in some systems
Basic Bus Operations
Terms

- **Doubleword**
  - 32 bits, most often known as a “DWORD”

- **Quadword**
  - 64 bits, sometimes known as a “QWORD”

- **Burst transaction**
  - Any transaction consisting of more than one data phase

- **Idle state (no bus activity)**
  - Indicated by FRAME# and IRDY# deasserted
Example #1 – Basic Write

- A four-DWORD burst from an initiator to a target
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
Write Example – Things to Note

- The initiator has a phase profile of 3-1-1-1
  - *First data can be transferred in three clock cycles (idle + address + data = “3”)*
  - *The 2\textsuperscript{nd}, 3\textsuperscript{rd}, and last data are transferred one cycle each (“1-1-1”)*)
Write Example – Things to Note

- The target profile is 5-1-1-1
  - Medium decode – DEVSEL# asserted on 2\textsuperscript{nd} clock after FRAME#
  - One clock period of latency (or wait state) in the beginning of the transfer
    - DEVSEL# asserted on clock 3, but TRDY# not asserted until clock 4
    - Ideal target write is 3-1-1-1

- Total of 4 data phases, but required 8 clocks
  - Only 50% efficiency
Target Address Decoding

- PCI uses distributed address decoding
  - A transaction begins over the PCI bus
  - Each potential target on the bus decodes the transaction’s PCI address to determine whether it belongs to that target’s assigned address space
    - One target may be assigned a larger address space than another, and would thus respond to more addresses
  - The target that owns the PCI address then claims the transaction by asserting DEVSEL#
Distributed Address Decoding

Programmable decoders
Each agent decodes address
DEVSEL# used to claim address

Initiator

Agent
Decoder

Agent
Decoder

Agent
Decoder

Agent
Decoder

AD[31:0]

DEVSEL#
Address decoders come in different speeds
- If a transaction goes unclaimed (nobody asserts DEVSEL#), “Master Abort” occurs
Example #2 – Target Read

- A four-DWORD burst read from a target by an initiator
More Terms

- **Turnaround cycle**
  - “Dead” bus cycle to prevent bus contention

- **Wait state**
  - A bus cycle where it is possible to transfer data, but no data transfer occurs
  - Target deasserts TRDY# to signal it is not ready
  - Initiator deasserts IRDY# to signal it is not ready

- **Target termination**
  - Target asserts STOP# to indicate that it needs to terminate the current transaction
Target Read Transaction

Initiator asserts FRAME# to start the transaction

Target inserts a wait state

Initiator acknowledges termination request

Target requests termination

PCI Basics - Slide 31

© 2000 Xilinx, Inc.
All Rights Reserved
Target Read – Things to Note

- Wait states may be inserted dynamically by the initiator or target by deasserting IRDY# or TRDY#

- Either agent may signal the end of a transaction
  - The target signals termination by asserting STOP#
  - The initiator signals completion by deasserting FRAME#
Zero and One Wait State

- These terms are used in popular PCI parlance* to describe how an agent signals its xRDY# signal during each data phase
  *Although the PCI spec uses the term “wait state,” it does not use terms such as “zero-wait-state agent” and “one-wait-state device.”

- A one-wait-state agent inserts a wait state at the beginning of each data phase
  - This is done if an agent – built in older, slower silicon – needs to pipeline critical paths internally
  - Reduces bandwidth by 50%
Zero and One Wait State

- The need to insert a wait state is typically an issue only when the agent is sourcing data (initiator write or target read)
  
  This is because such an agent would have to sample its counterpart’s xRDY# signal to see if that agent accepted data, then fan out to 36 or more clock enables (for AD[31:0] and possibly C/BE#[3:0]) to drive the next piece of data onto the PCI bus . . . all within 11 ns!
  
  - And even that 11 ns would be eaten up by a chip’s internal clock-distribution delay
Types of Target Termination

- **Target Retry**
  “I’m not ready, try again later.”

- **Target Disconnect with Data**
  “I couldn’t eat another bite . . . OK, just one more.”

- **Target Disconnect Without Data**
  “I couldn’t eat another bite . . . and I’m not kidding!”

- **Target Abort**
  “Major snafu alert!”
PCI Addressing
and Bus Commands

Xilinx PCI
PCI Address Space

- A PCI target can implement up to three different types of address spaces
  - **Configuration space**
    - Stores basic information about the device
    - Allows the central resource or O/S to program a device with operational settings
  - **I/O space**
    - Used mainly with PC peripherals and not much else
  - **Memory space**
    - Used for just about everything else
Types of PCI Address Space

- **Configuration space**
  - Contains *basic device information*, e.g., *vendor or class of device*
  - Also permits *Plug-N-Play*
    - Base address registers allow an agent to be mapped dynamically into memory or I/O space
    - A programmable interrupt-line setting allows a software driver to program a PC card with an IRQ upon power-up (without jumpers!)
Configuration space (cont’d)

- Contains 256 bytes
  - The first 64 bytes (00h – 3Fh) make up the standard configuration header, predefined by the PCI spec
  - The remaining 192 bytes (40h – FFh) represent user-definable configuration space

- This region may store, for example, information specific to a PC card for use by its accompanying software driver
Types of PCI Address Space

- I/O space
  - *This space is where basic PC peripherals (keyboard, serial port, etc.) are mapped*
  - *The PCI spec allows an agent to request 4 bytes to 2GB of I/O space*
    - For x86 systems, the maximum is 256 bytes because of legacy ISA issues
Types of PCI Address Space

- Memory space
  - *This space is used by most everything else – it’s the general-purpose address space*
    - The PCI spec recommends that a device use memory space, even if it is a peripheral
  - *An agent can request between 16 bytes and 2GB of memory space*
    - The PCI spec recommends that an agent use at least 4kB of memory space, to reduce the width of the agent’s address decoder
PCI Commands

- PCI allows the use of up to 16 different 4-bit commands
  - Configuration commands
  - Memory commands
  - I/O commands
  - Special-purpose commands

- A command is presented on the C/BE# bus by the initiator during an address phase (a transaction’s first assertion of FRAME#)
## PCI Commands

<table>
<thead>
<tr>
<th>C/BE#</th>
<th>Command</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt Acknowledge</td>
<td>Memory</td>
</tr>
<tr>
<td>0001</td>
<td>Special Cycle</td>
<td>I/O</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Read</td>
<td>Configuration</td>
</tr>
<tr>
<td>0011</td>
<td>I/O Write</td>
<td>Special-Purpose</td>
</tr>
<tr>
<td>0100</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0101</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read</td>
<td>Memory</td>
</tr>
<tr>
<td>0111</td>
<td>Memory Write</td>
<td>I/O</td>
</tr>
<tr>
<td>1000</td>
<td>Reserved</td>
<td>Configuration</td>
</tr>
<tr>
<td>1001</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>1010</td>
<td>Configuration Read</td>
<td>Memory</td>
</tr>
<tr>
<td>1011</td>
<td>Configuration Write</td>
<td>I/O</td>
</tr>
<tr>
<td>1100</td>
<td>Memory Read Multiple</td>
<td>Special-Purpose</td>
</tr>
<tr>
<td>1101</td>
<td>Dual Address Cycle</td>
<td>Memory</td>
</tr>
<tr>
<td>1110</td>
<td>Memory Read Line</td>
<td>I/O</td>
</tr>
<tr>
<td>1111</td>
<td>Memory Write and Invalidate</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
PCI Configuration
The Plug-and-Play Concept

Plug-and-Play (PNP)

- Allows add-in cards to be plugged into any slot without changing jumpers or switches
  - Address mapping, IRQs, COM ports, etc., are assigned dynamically at system start-up
- For PNP to work, add-in cards must contain basic information for the BIOS and/or O/S, e.g.:
  - Type of card and device
  - Memory-space requirements
  - Interrupt requirements
The Plug-and-Play Concept

- To make PNP possible in PCI, each PCI device maintains a 256-byte configuration space
  - The first 64 bytes (shown here) are predefined in the PCI spec and contain standard information
  - The upper 192 bytes may be used to store device-specific information
Configuration Transactions

- Are generated by a host or PCI-to-PCI bridge
- Use a set of IDSEL signals as chip selects
  - Dedicated address decoding
  - Each agent is given a unique IDSEL signal
- Are typically single data phase
  - Bursting is allowed, but is very rarely used
- Two types (specified via AD[1:0] in addr. phase)
  - Type 0: Configures agents on same bus segment
  - Type 1: Configures across PCI-to-PCI bridges
### Configuration Example

<table>
<thead>
<tr>
<th>C/BE#</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Interrupt Ack.</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>0100</td>
<td>Reserved</td>
</tr>
<tr>
<td>0101</td>
<td>Reserved</td>
</tr>
<tr>
<td>0110</td>
<td>Memory Read</td>
</tr>
<tr>
<td>0111</td>
<td>Memory Write</td>
</tr>
<tr>
<td>1000</td>
<td>Reserved</td>
</tr>
<tr>
<td>1001</td>
<td>Reserved</td>
</tr>
<tr>
<td>1010</td>
<td>Config. Read</td>
</tr>
<tr>
<td>1011</td>
<td>Config. Write</td>
</tr>
<tr>
<td>1100</td>
<td>Memory Read Mult.</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 &amp; Inv.</td>
</tr>
</tbody>
</table>

#### Timing Diagram

**Address Phase**
- **AD[31:0]**
- **C/BE#[3:0]**
- **IDSEL**
- **FRAME#**
- **IRDY#**
- **TRDY#**
- **DEVSEL#**

**Data Transferred**
- **Addr**
- **Cfg. Data**
- **CFG CMD**
- **Byte Enable**

**Note**
- IDSEL is asserted (active High) during the address phase.
- The host can do anything it wants to IDSEL outside of a configuration address phase.
- This time, the target asserts DEVSEL# based on IDSEL and not based on the address.
### Standard PCI Configuration Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device ID</td>
<td>31</td>
<td>00h</td>
</tr>
<tr>
<td>Vendor ID</td>
<td>16</td>
<td>04h</td>
</tr>
<tr>
<td>Status</td>
<td>15</td>
<td>08h</td>
</tr>
<tr>
<td>Command</td>
<td>0</td>
<td>0Ch</td>
</tr>
<tr>
<td>Class Code</td>
<td>31</td>
<td>10h</td>
</tr>
<tr>
<td>Revision ID</td>
<td>16</td>
<td>14h</td>
</tr>
<tr>
<td>Latency Timer</td>
<td>15</td>
<td>18h</td>
</tr>
<tr>
<td>Cache Line Size</td>
<td>0</td>
<td>1Ch</td>
</tr>
<tr>
<td>Base Address Register #0</td>
<td>31</td>
<td>20h</td>
</tr>
<tr>
<td>Base Address Register #1</td>
<td>16</td>
<td>24h</td>
</tr>
<tr>
<td>Base Address Register #2</td>
<td>15</td>
<td>28h</td>
</tr>
<tr>
<td>Base Address Register #3</td>
<td>0</td>
<td>2Ch</td>
</tr>
<tr>
<td>Base Address Register #4</td>
<td>31</td>
<td>30h</td>
</tr>
<tr>
<td>Base Address Register #5</td>
<td>16</td>
<td>34h</td>
</tr>
<tr>
<td>CardBus CIS Pointer</td>
<td>15</td>
<td>38h</td>
</tr>
<tr>
<td>Subsystem ID</td>
<td>0</td>
<td>3Ch</td>
</tr>
<tr>
<td>Subsystem Vendor ID</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>Expansion ROM Base Address</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Cap. List Ptr.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Max_Lat</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>Min_Gnt</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Interrupt Pin</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Interrupt Line</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

© 2000 Xilinx, Inc.  
All Rights Reserved
### Required by PCI Spec 2.2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Required by PCI Spec 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Device ID</td>
<td>Required</td>
</tr>
<tr>
<td>16</td>
<td>Vendor ID</td>
<td>Required</td>
</tr>
<tr>
<td>04</td>
<td>Command</td>
<td>Not Required</td>
</tr>
<tr>
<td>00</td>
<td>Revision ID</td>
<td>Required</td>
</tr>
<tr>
<td>08</td>
<td>Cache Line Size</td>
<td>Required</td>
</tr>
<tr>
<td>10</td>
<td>Latency Timer</td>
<td>Required</td>
</tr>
<tr>
<td>14</td>
<td>Base Address Register #0</td>
<td>Required</td>
</tr>
<tr>
<td>18</td>
<td>Base Address Register #1</td>
<td>Required</td>
</tr>
<tr>
<td>1Ch</td>
<td>Base Address Register #2</td>
<td>Required</td>
</tr>
<tr>
<td>20</td>
<td>Base Address Register #3</td>
<td>Required</td>
</tr>
<tr>
<td>24</td>
<td>Base Address Register #4</td>
<td>Required</td>
</tr>
<tr>
<td>28</td>
<td>Base Address Register #5</td>
<td>Required</td>
</tr>
<tr>
<td>2Ch</td>
<td>CardBus CIS Pointer</td>
<td>Required</td>
</tr>
<tr>
<td>30</td>
<td>Expansion ROM Base Address</td>
<td>Required</td>
</tr>
<tr>
<td>34</td>
<td>Cap. List Ptr</td>
<td>Required</td>
</tr>
<tr>
<td>38</td>
<td>Reserved</td>
<td>Required</td>
</tr>
<tr>
<td>3Ch</td>
<td>Interrupt Ptn</td>
<td>Required</td>
</tr>
<tr>
<td>3Ch</td>
<td>Interrupt Line</td>
<td>Required</td>
</tr>
</tbody>
</table>
Electrical & Timing Specifications
Signaling Environments

- The PCI spec describes two different electrical environments
  - 5V signaling
  - 3.3V signaling

- Technically, these names have *nothing* to do with the actual supply voltage
  - Rather, they are tied to logic-level thresholds
Signaling Environments

- 5V signaling is the most common
- 66MHz PCI buses can only use 3.3V signaling
  - Note that 33MHz PCI can still use either
- Some plug-in cards can support both signaling environments – these are known as “universal” cards
Note that logic levels are not absolute. Rather, they are functions of \( V_{DD} \)!

The TTL-based logic levels allow 3.3V devices to operate with 5V PCI signaling.

**5V Signaling**
- \( V_{DD} = 5.0V \)
- \( V_{OH} = 2.4V \)
- \( V_{IH} = 2.0V \)
- \( V_{IL} = 0.8V \)
- \( V_{OL} = 0.55V \)
- \( V_{SS} = 0V \)

**3.3V Signaling**
- \( V_{DD} = 3.3V \)
- \( V_{OH} = 0.9 \times V_{DD} = 2.97V \)
- \( V_{IH} = 0.5 \times V_{DD} = 1.65V \)
- \( V_{IL} = 0.3 \times V_{DD} = 0.99V \)
- \( V_{OL} = 0.1 \times V_{DD} = 0.33V \)
- \( V_{SS} = 0V \)
Reflective-Wave Switching

- PCI uses “reflective waves”
  - Each wire on the PCI bus is a non-terminated transmission line, which causes signals to reflect over the length of the trace.
  - Valid voltage levels are obtained after one reflection; this reduces the cost of PCI by not requiring high-powered output drivers.

The signal travels to the non-terminated end
The signal is then reflected back to complete the signal propagation

NON-TERMINATED TRANSMISSION LINE

Output driver

Reflection

5V
0V
Reflective-Wave Switching

- This display shows how a reflection looks

- Signal is driven to 0.5\(V_{CC}\)

- Wave reflection boosts the signal up near full \(V_{CC}\)

- HSPICE Simulation of Intel PCI Speedway

- 50pF Load

- Far End

- Driver End
 PCI – AC Specifications
Input Requirements – Clamp Diodes

- Clamp diodes protect inputs from momentary short-circuit current caused by tri-stating delays

- **5V signaling environment**
  - Inputs must be clamped to ground
  - Upper clamp to 5V rail is optional

- **3.3V signaling environment**
  - Inputs must be clamped to both ground and $V_{CC}$
  - For universal cards (or any other 3.3V device that interfaces to 5V PCI), inputs must be clamped to the 5V supply, not the 3.3V supply

$V_{CC} = 3.3V, 5V$
Bad Diode Clamping
Why Must Universal Cards Clamp to 5V?

A universal card clamped to 3.3V, that is plugged into a 5V signaling bus, will create a short-circuit connection between the motherboard’s 5V and 3.3V supplies!
33MHz PCI Timing Specification

30ns Bus Cycle Time

\[
\begin{align*}
\text{t_{val}} & \quad 11\text{ns} & \quad \text{t_{prop}} & \quad 10\text{ns} & \quad \text{t_{su}} & \quad 7\text{ns} & \quad \text{t_{skew}} & \quad 2\text{ns} \\
\text{Max clock-to-valid} & & \text{Wave propagation} & & \text{Input setup} & & \text{Clock skew} & \quad = \quad 30\text{ns}
\end{align*}
\]

Other Requirements:
Hold time : 0ns
Min clock-to-out : 2ns
Output off time : 28ns

All timing parameters are measured at the package pin
Why Is PCI Timing So Tough?

- PCI handshaking performed every clock cycle
  - These signals can’t be pipelined
- 7ns setup + clock_delay → 100+ MHz!  [33 MHz PCI]
- 3ns setup + clock_delay → 220+ MHz!  [66 MHz PCI]
Add-In Card Design

- **Trace length**
  - All 32-bit PCI signals must be no more than 1.5″
  - All 64-bit ext. signals must be no more than 2.0″

- **Clock trace must be exactly 2.5″ (± 0.1″)**
  - Routed to only one load
  - Needed for clock-skew control

- **PCI device requirements**
  - One pin per signal!
  - Max input capacitance is 10pF (unless the device is on the motherboard, where 16pF is OK)
System Issues – Bus Loading

- No PCI spec requirement as to the loading on the bus; however:
  - A driver must meet the 10ns propagation spec

- The rule of thumb is 10 loads max for 33MHz
  - Motherboard devices count as one load
  - Each add-in card slot counts as two load
  - Since most PC motherboards must have >2 PCI devices, they usually have no more than 4 slots

- More slots are available using PCI-to-PCI bridges or peer-to-peer PCI systems
64-Bit and 66MHz PCI
The 64-Bit PCI Extension

- Doubles the available PCI bandwidth (keeping the clock frequency at 33MHz) to 264 MB/sec
- 64-bit PCI can use both 5V and 3.3V signaling
- Only used for memory transactions
The 64-Bit PCI Extension

- Mixing and matching is allowed
  - A 32-bit card can be plugged into a 64-bit slot
  - A 64-bit card can be plugged into a 32-bit slot
  - Use of the 32-bit vs. 64-bit datapath is negotiated between the initiator and target at the start of each transaction
Additional 64-Bit Pins

- **AD[63:32]**

- **C/BE#[7:4]**
  - *Used only for byte enables (not for PCI commands) except in the special case of Dual Address Cycle, discussed later*

- **PAR64**
  - *The XOR of AD[63:32], C/BE#[7:4], and PAR64 must equal zero (even parity)*
Additional 64-Bit Pins

- **REQ64#**
  - Mirrors FRAME# – indicates that the initiator requests a 64-bit transaction

- **ACK64#**
  - Mirrors DEVSEL# – indicates that the target is able to fulfill the transaction request using the 64-bit datapath
  - By not asserting ACK64#, the target tells the initiator that it is a 32-bit agent
64-Bit Initiator to 64-Bit Target

- Works the same as a standard 32-bit transaction, with the following additions:
  - The initiator asserts REQ64# to mirror FRAME#
  - The target, in response, asserts ACK64# to mirror DEVSEL#

- Data is transferred on AD[31:0] and AD[63:32]
  - C/BE#[7:4] and PAR64 are also used

- The starting address must be QUADWORD aligned (i.e., divisible by 8: AD[2] = 0)
64-Bit → 64-Bit Example

- Upper datapath used during data phases (don’t-care in address phase)

- REQ64# mirrors FRAME#

- ACK64# mirrors DEVSEL#
66MHz PCI Overview

- Pushes PCI bandwidth as high as 528MB/sec
- Most often used with the 64-bit extension, although it is legal to have 32-bit 66MHz PCI
- The signaling protocol is the same as with 33MHz PCI
66MHz PCI Overview

- 66MHz PCI can only use 3.3V signaling
- A device that can operate at 66MHz has the 66MHz Capable bit set in the Status Register
- The loading allowance is cut in half (5 loads), so only one or two add-in slots are possible
- For open systems, 66MHz add-in cards must also work on a 33MHz PCI bus
PCI Variations
Overview of PCI Variations

- As a well-defined standard, PCI and its various flavors have been widely adopted by many industries that require high-bandwidth data systems
  - Industrial computing
  - Datacom and telecom
  - Portable systems
  - Desktop systems

- Up-and-coming variations on PCI look to push its maximum bandwidth even higher
PCI Variations
Same Protocol (Different Form Factor)

- **PMC**
  - PCI in a mezzanine form factor

- **CompactPCI**
  - PCI in a Eurocard (VME-style) form factor
  - Used as a passive backplane
  - Used in physically rugged environments such as industrial and telecom systems

- **Mini PCI**
  - New proposed PCI standard for portable systems
PCI Variations
Modified Protocol

- **CardBus**
  - *PCI in a PCMCIA form factor (portable systems) *
  - *Point-to-point, only slightly different protocol from standard PCI *

- **AGP (Advanced Graphics Port)**
  - *PCI-like point-to-point protocol *
  - *Primarily used for PC graphics cards *

- **PCI-X**
  - *New proposal to push bandwidth over 1GB/sec *
  - *Backward compatible with standard PCI *
CompactPCI

- Used in Telecom and Industrial Applications that Require a Rugged Form Factor (Eurocard)
- IC Devices Maintain Specs from Standard PCI
- CompactPCI Imposes Additional Requirements for Add-in Cards as well as for the Host System
- Supports up to Eight Plug-in Slots
- Spec Maintained by the PCI Industrial Computer Manufacturers Group (PICMG), of which Xilinx is a Member
PCI / CompactPCI Comparison

CompactPCI (Eurocard)

Standard PCI Slot
Hot-Swap CompactPCI

- Live Insertion and Removal of Cards
  - High-availability systems (e.g., telecom)

- Electrical Issues
  - Signal lines need precharge to ~1V
  - Leakage current must be less than 10µA
  - Limited early power (2A maximum)

- Adds Hot-Swap Register to configuration space

- Not to be confused with Hot-Plug
Hot-Swap CompactPCI
Levels of Compliance

🔹 “Capable” – The Minimum Level of Compliance
  — Must be at least PCI v2.1 compliant
  — Must be able to tolerate precharge voltage (~1V)
  — Must be able to tolerate asynchronous RST#

🔹 “Friendly” – The Intermediate Level of Compliance
  — Must at least be Hot-Swap “Capable”
  — Must also support the Capabilities List Pointer (PCI v2.2)

🔹 “Ready” – The Highest Level of Compliance
  — Must at least be Hot-Swap “Friendly”
  — Must also support internal Bias Voltage and Early power
The Xilinx PCI Solution
The PCI Challenge

1. Must Guarantee Timing
   — Setup, Max, and Min

2. Must Guarantee the Behavior
   — Now and in the future

3. Must Meet Performance Requirements
   — Sustained throughput at 66MHz

4. Must Meet Cost Target
   — Single-chip solution
The PCI Challenge

The Critical Path

Setup: 3ns
Hold time: 0ns
Clk-out max: 6ns
Clk-out min: 2ns

PCI Signal

20 loads
72 loads

Logic

FF

PCI CLK

PCI Signal

The critical path is equivalent to 220+ MHz!
Real Flexibility
- Uses standard FPGAs
- Back-end de-coupled from core

Real Performance
- Zero wait-state at 0-33MHz
- Full 32 or 64-bit data path

Real Compliance
- PCI Initiator and Target
- Guaranteed timing

Xilinx PCI Solution™
The Real-PCI™

PCI Bus

Reference Design
User Design
FIFO(s)
DMA(s)
Back-end Interface
PCI Interface
Zero wait-state

Real Compliance
- PCI Initiator and Target
- Guaranteed timing

PCI Basics - Slide 83
© 2000 Xilinx, Inc.
All Rights Reserved
Why The Real-PCI From Xilinx?

"The Xilinx PCI Core has the most flexibility and complete feature set of any vendor offering a programmable PCI core that we extensively evaluated. We were most impressed with the silicon features, as well as a smooth migration path across various cores and device densities"

"If our system performance requirements change, we have every confidence that we can import a new core to the existing design with minimal engineering effort".

Joel Van Doren, Senior design engineer
Xerox Engineering Systems
Cost-Reduction

- External PLD
  7K Gates
- External DLLs, memories, Controllers and translators
- PCI ASSP
  PCI Master I/F
- 15K Gates Logic
- PCI Master I/F

Standard Chip

The **Real-PCI™** Solution $<$6

- PCI Basics - Slide 85
- © 2000 Xilinx, Inc.
  All Rights Reserved
ASSP Replacement & Integration

System & Memory controllers, DLLs, Level translators ($20)

Glue Logic

External PLD PCI Master I/F ($5)

Supported Devices
XC2S30 XC2S50
XC2S100 XC2S150

The Real-PCI

Standard Chip PCI Master I/F ($15)

Memory ($9)
Supporting Reference Designs

- Asynchronous FIFOs & DMA Controller
- Power Management Module

User Design
Up to 135,000 System Gates

Power Management Module

Asynchronous FIFOs

Custom DMA Controller

Real-PCI Interface
PCI 64
PCI 32

PCI Bus
Extra Support from PCI Experts

- Provide Worldwide Access to Certified PCI Experts that allow:
  - Support for targeting additional devices
  - Complete turnkey integration

MULTI VIDEO DESIGN
# Ordering Information

<table>
<thead>
<tr>
<th>Design File Access</th>
<th>PCI64 Design Kit DO-DI-PCI64-DK</th>
<th>PCI64 DO-DI-PCI64</th>
<th>PCI32 Design Kit DO-DI-PCI32-DK</th>
<th>PCI32 Spartan DO-DI-PCI32-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit 66 MHz</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Virtex</td>
<td>Virtex</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>64-bit 33 MHz</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Virtex</td>
<td>Virtex-E</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32-bit 33 MHz</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>Spartan-II</td>
</tr>
<tr>
<td></td>
<td>Virtex</td>
<td>Virtex-E</td>
<td>Virtex-E</td>
<td>Spartan-XL</td>
</tr>
<tr>
<td></td>
<td>Spartan-II</td>
<td>Spartan-XL</td>
<td>Spartan-XL</td>
<td>Spartan</td>
</tr>
<tr>
<td></td>
<td>Spartan-XL</td>
<td>Spartan</td>
<td>4000XLA</td>
<td>4000XLA</td>
</tr>
<tr>
<td>Development Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototyping board</td>
<td>Nallatech</td>
<td>-</td>
<td>VCC</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BallyLINX</td>
<td>-</td>
<td>HotPCI</td>
<td>-</td>
</tr>
<tr>
<td>SW Driver Development Tools</td>
<td>NuMega</td>
<td>-</td>
<td>NuMega</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SoftICE Driver Suite</td>
<td>-</td>
<td>SoftICE Driver Suite</td>
<td>-</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Designs</td>
<td>PCI Bridge Designs</td>
<td>PCI Bridge Designs</td>
<td>PCI Bridge Designs</td>
<td>PCI Bridge Designs</td>
</tr>
<tr>
<td>Design Examples</td>
<td>Reference Drivers</td>
<td>-</td>
<td>Reference Drivers</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Board Gerber files</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PCI System Architecture Text Book</td>
<td>PCI System Arch Text Book</td>
<td>Online</td>
<td>PCI System Arch Text Book</td>
<td>Online</td>
</tr>
<tr>
<td></td>
<td>Design Guide</td>
<td></td>
<td>Design Guide</td>
<td></td>
</tr>
</tbody>
</table>
The Xilinx PCI Design Flow

- The PCI Lounge
- Core Configuration and Download
- Simulation and Synthesis
- Implementation – Xilinx Alliance/Foundation
The PCI Lounge

- A Special Area for Xilinx PCI Customers on WebLINX
  - PCI core configuration and download
  - Latest design data – no CD required!

- To Gain Access, Customers Submit an Online Registration form
  - The customer selects a unique username and password
  - The customer also gives his or her product serial number as proof of purchase
PCI Lounge Registration

- Register your LogiCORE product at:
  

- During the registration process, you will be asked for:
  
  - Your LogiCORE product’s serial number
  - Xilinx User ID and password (the same as your SmartAgent ID)
    - You can create your own User ID during the registration process, if necessary
  - The type of LogiCORE PCI Interface you purchased
PCI Lounge Registration

- In less than 2 business days, you will be sent a confirmation that you have access
Downloading from the Internet

- Once you are registered for the PCI Lounge, you can download your LogiCORE Interface in **two easy steps**:
  1. Customize your LogiCORE PCI Interface on WebLINX
  2. Download your tailor-made core
Downloading from the Internet

- A Java-based GUI allows you to customize configuration options easily
  
  — See Chapter 3 of the LogiCORE PCI User’s Guide for more detailed information
Downloading from the Internet

- When it comes time to download, you can tailor the downloaded fileset for your design environment, based on:
  - Device and package
  - Design-entry tool
  - Archiving method
    - .zip for PC
    - .tar.gz for Unix
Xilinx PCI Design Flow Overview

CORE Configuration

Design Entry
- Symbol
- Sim.Model
- User Design
  - HDL or Schematic
  - Synthesis
    - User design only
    - Netlist

Design Verification
- Functional Simulation
- Timing Simulation
- Place & Route
  - Netlist
  - Constraints
- Netlist

Design Implementation
Simulation and Synthesis

- Functional and Timing simulation with -
  Model Technology
  A MENTOR GRAPHICS COMPANY

- Synthesis with -
  Synopsys®
  Synplicity®
  Exemplar®
Xilinx Implementation

- Supported by Standard Xilinx Alliance and Foundation Series, V2.1i software
Available PCI Resources

- LogiCORE PCI Lounges on WebLINX
  — [www.xilinx.com/pci](http://www.xilinx.com/pci)

- Xilinx Hotline
  — Expert PCI support engineers

- WWW based solutions and PCI Expert Journal
  — [support.xilinx.com](http://support.xilinx.com)

- PCI-SIG discussion group
  — Mailing list subscription information on [www.pcisig.com/forum.html](http://www.pcisig.com/forum.html)