DECchip 21050 PCI-PCI Bridge

Hardware Implementation: An Application Note

Order Number: EC–Q9Y3A–TE

June 1994

This application note presents guidelines for hardware implementation of the DECchip 21050.

Revision/Update Information: This is a new application note.

Digital Equipment Corporation
Maynard, Massachusetts

June 1994

This document was prepared using DECwrite, Version 3.0.

Contents

List of Figures

List of Tables

Introduction

This document presents guidelines for hardware implementation of the DECchip 21050 in a system. This application note is limited to hardware implementation of 21050 only and does not cover any devices that might be behind the 21050, or initialization code needed to configure the DECchip 21050.

This application note includes implementation notes on layout, clocking, secondary bus IDSEL mapping, using data synchronization pins, interrupt routing, and secondary bus arbitration. Implementation on both a motherboard and an option card are covered. For most situations, hardware implementation issues are the same. An exception is the clocking discussion, which specifies separate guidelines for motherboard and option card implementations. In addition, guidelines for interrupt routing on option cards are provided in the interrupt discussion.

The following related documents may be useful.

Published by Digital Equipment Corporation:

- *DECchip 21050 PCI-PCI Bridge Data Sheet*, Revision 1.0
- *DECchip 21050 PCI-PCI Bridge Configuration: An Application Note*
- *DECchip 21050 PCI Evaluation Board User's Guide*

Published by the PCI Special Interest Group:

- *PCI Local Bus Specification*, Revision 2.0
- *PCI-PCI Bridge Architecture Specification*, Revision 1.0

Functional Overview

The DECchip 21050 PCI-PCI Bridge provides a connection between two independent PCI buses. The PCI buses operate concurrently, except when reads or non-posted writes are crossing the bridge. The two PCI buses are referred to as the *primary PCI bus*, which is the PCI bus closest to the CPU, and the *secondary PCI bus*, which is the PCI bus farther from the CPU.

The DECchip 21050 has two common applications. System designers can use the 21050 on a motherboard to add more devices or add-in card slots than a single PCI bus can support. Add-in card designers can use the 21050 to enable multiple device option cards. (PCI add-in cards are restricted to a single connection to the signals in the PCI connector.) Designers can also use the 21050 to isolate bus traffic on one PCI bus segment from other PCI bus segments.

The 21050 PCI-PCI Bridge has the following interfaces:

- **Primary and secondary PCI bus interfaces**
  Control and data to two PCI buses, including LOCK#, PERR#, and SERR#. To implement these interfaces, follow the guidelines in the *PCI Local Bus Specification,* Rev. 2.0. The 21050 primary interface implements Rev. 2.0 compliant 5-Volt drivers.
- **Secondary bus arbiter**
  Six request/grant pairs for secondary bus master control, plus one pin, **s_cfn_l**, to enable the arbiter.
- **Data synchronization pins**
  One control input (**s_dispst_l**) that marks posted writed data and conditionally

disables write posting, and one control output (**s_bufne_l**) that indicates when marked data has been delivered.

- **Clocks**
  One primary interface clock input (**p_clk**), one secondary interface clock input (**s_clk**), and seven secondary interface clock outputs (**s_clk_o<6:0>**).
- **Diagnostic pins**
  One test input (**goz_l**) that tri-states all outputs, and one test output (**nand_out**) that is the output of the diagnostic nand tree.

General Layout Guidelines

When using the 21050, consult the general layout guidelines provided in the *PCI Local Bus Specification* Rev. 2.0. Clock routing has some special requirements. Guidelines and requirements for clock routing are discussed in the Clocking section of this document. Guidelines for routing of secondary IDSEL signals are given in the Secondary IDSEL Mapping section. The following sections discuss pullups and expansion card routing.

Pullups

Pullups are required on the following shared PCI control signals: FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, PERR#, SERR#, and LOCK#. These signals must have pullups on both the primary and secondary PCI buses. Even if optional shared signals such as LOCK# are not used, they *must* be pulled up for proper 21050 operation. See the *PCI Local Bus Specification* for resistor value guidelines.

If the 21050 is implemented on a motherboard, you must place both primary and secondary bus pullups on the motherboard. If the 21050 is implemented on an option card with the primary bus interfacing to the card edge, place primary bus pullups on the motherboard and place secondary bus pullups on the option card.

Point-to-point and shared 32-bit signals do not require pullups. This includes the secondary arbiter request/grant pairs and data synchronization signals. (If **s_dispst_l** is not used, however, tie it either high or low through a resistor.)

Expansion Card Routing

Follow the guidelines and requirements for routing on expansion cards that are provided in the *PCI Local Bus Specification*. The most important requirements are highlighted in this section.

PCI signals coming from the motherboard on the expansion card must be limited to only one load. This includes the primary CLK. These are the signals on the primary interface of the 21050. These signals also have trace length limitations, which are 1.5 inches for PCI signals and 2.5 inches for the primary CLK.

PCI signals on the secondary side of the 21050, however, do not need to adhere to these restrictive loading and trace length requirements. The secondary PCI bus can support the full 10 loads (including the 21050) and essentially can be treated like a motherboard PCI bus.

Clocking

The following sections provide an overview of 21050 clocking requirements, and explain how to implement clocks using the 21050 on a motherboard and an option card.

21050 Clocking

The 21050 has two clocking domains, one for the primary PCI interface and one for the secondary PCI interface. Both interfaces must operate at the same frequency and must be synchronous to each other.

The primary clocking domain is controlled by the **p_clk** input.

The secondary bus arbiter pins and the data synchronization pins operate in the secondary interface clocking domain. The secondary clocking domain is controlled by the **s_clk** input.

The relationship between the **p_clk** and **s_clk** inputs has the following restrictions:

- Both interfaces must operate at the same frequency.
- Maximum clock frequency is 33 MHz.
- Maximum delay between **p_clk** and **s_clk** is 7 ns for both rising and falling edges.
- Minimum delay between **p_clk** and **s_clk** is 0 ns, that is, **s_clk** cannot precede **p_clk** for both rising and falling edges.
- Input duty cycle of **p_clk** must be between 45–55%, measured at 1.5V.

21050 Output Clocks

The 21050 generates seven secondary bus clock outputs, **s_clk_o<6:0>**. These clocks are derived from **p_clk**. Use one of the secondary clock outputs as the secondary clock input to the 21050, and use the six other outputs for devices on the secondary bus.

These **s_clk_o<6:0>** clock outputs are generated directly from **p_clk,** and their relationship to **p_clk** has the following restrictions:

- All clock outputs must operate at the same frequency.
- Maximum delay between **p_clk** and **s_clk_o** is 5 ns for both rising and falling edges.
- Minimum delay between **p_clk** and **s_clk_o** is 0 ns for both rising and falling edges.
- 21050 skews the duty cycle by adding a maximum of 1.5 ns to clock high (and, therefore, removing that same amount from clock low).

It is recommended that the 21050 secondary clock outputs not be used as primary clock inputs to another 21050, because the 21050 can skew the duty cycle of the secondary clock outputs that it generates. As a general rule, do not use the secondary clock outputs when the 21050 is implemented on a motherboard, because a card containing another 21050 might be plugged into one of the option card slots. However, you must use the 21050 secondary clock outputs when the 21050 is implemented on an option card.

Clock Implementation on the Motherboard

If the 21050 is implemented on a motherboard and the secondary bus is routed to one or more PCI expansion card slots or to another 21050, you should not use the 21050 secondary clock outputs **s_clk_o<6:0>**, unless the designer can guarantee that the output clocks will meet PCI duty cycle specifications. This may be possible by controlling the skew of **p_clk**. Because an expansion card could implement a 21050, never route secondary clock outputs to an expansion card slot.

If you cannot use secondary clock outputs due to the restrictions listed in this section, you can substitute a low skew clock buffer that preserves duty cycle to generate secondary clocks. For example:

- Texas Instruments — CDC328A
- National Semiconductor — CGS74B2525
- IDC — 1DT74FCT805CT

One of the secondary clock signals must be connected to the 21050 secondary clock input **s_clk**. Plan the layout of clocks to minimize skew between the 21050 and other PCI devices.

If the 21050 secondary bus does not connect to any PCI option card slots or to the primary interface of another 21050, then you can use the secondary clock outputs according to the guidelines given in the following section.

Clock Implementation on an Option Card

If the 21050 is implemented on an option card, then you must use the secondary clock outputs for devices connected to the secondary bus and for the 21050 secondary clock input **s_clk**, because CLK signals coming from the card edge can be connected to only one load (in this case, the 21050 primary clock input **p_clk**). You *cannot* also use the CLK signal from the card edge to generate secondary clocks externally.

Follow these guidelines when implementing secondary clock outputs:

- Connect one **s_clk_o<6:0>** output to the 21050 **s_clk** input.
- Route **s_clk_o<6:0>** to minimize skew, that is, use the same amount of etch for the **s_clk_o<6:0>** outputs that are used as **s_clk** inputs as you use for the other **s_clk_o<6:0>** signals. It is recommended that the amount of delay due to etch be limited to less than 2 ns (10 inches if etch has a delay of 200 ns/inch).
- It is recommended that you use a series termination resistor of 20 Ohms to terminate the **s_clk_o<6:0>** outputs to limit reflections on the clock lines. Locate these resistors as close as possible to the 21050 **s_clk_o<6:0>** pins.

Secondary IDSEL Mapping

The *PCI Local Bus Specification* defines two formats for configuration transactions in hierarchical systems:

- Type 0 configuration transactions are used to configure devices on the same PCI bus.
- Type 1 configuration transactions are used to configure devices that reside on a downstream PCI bus.

When the 21050 detects a Type 1 configuration transaction for a device connected to the secondary interface of the bridge, the 21050 translates the Type 1 transaction into a Type 0 transaction on the downstream interface. Instead of using IDSEL to identify the target of a configuration transaction, the Type 1 configuration format uses a 5-bit field at bits <15:11> in the address as a device number. A device number in the Type 1 format is translated by the 21050 into an IDSEL line for Type 0 transactions on the target interface.

As required by the *PCI-to-PCI Bridge Architecture Specification,* the 21050 uses **s_ad<31:16>** as secondary IDSEL lines. The 21050 uses the mapping shown in Table 1 for translation of a device number to an **s_ad** pin. The 21050 supports mapping for device numbers 0 through F (hex). Configuration transactions with device numbers outside of this range are still forwarded, but no IDSEL mapping is performed (**s_ad<31:16>** are 00h).

Table 1 Device Number to Secondary IDSEL Mapping

Device Number p_ad<15:11>	**Secondary IDSEL s_ad<31:16>**	**Device Number p_ad<15:11>**	**Secondary IDSEL s_ad<31:16>**
00000	0000 0000 0000 0001	01000	0000 0001 0000 0000
00001	0000 0000 0000 0010	01001	0000 0010 0000 0000
00010	0000 0000 0000 0100	01010	0000 0100 0000 0000
00011	0000 0000 0000 1000	01011	0000 1000 0000 0000
00100	0000 0000 0001 0000	01100	0001 0000 0000 0000
00101	0000 0000 0010 0000	01101	0010 0000 0000 0000
00110	0000 0000 0100 0000	01110	0100 0000 0000 0000
00111	0000 0000 1000 0000	01111	1000 0000 0000 0000
		10000 - 11110	0000 0000 0000 0000
		11111	Generate Spec. Cyc. if **p_ad<7:2>** = 00h 0000 0000 0000 0000 if **p_ad<7:2>** > 00h

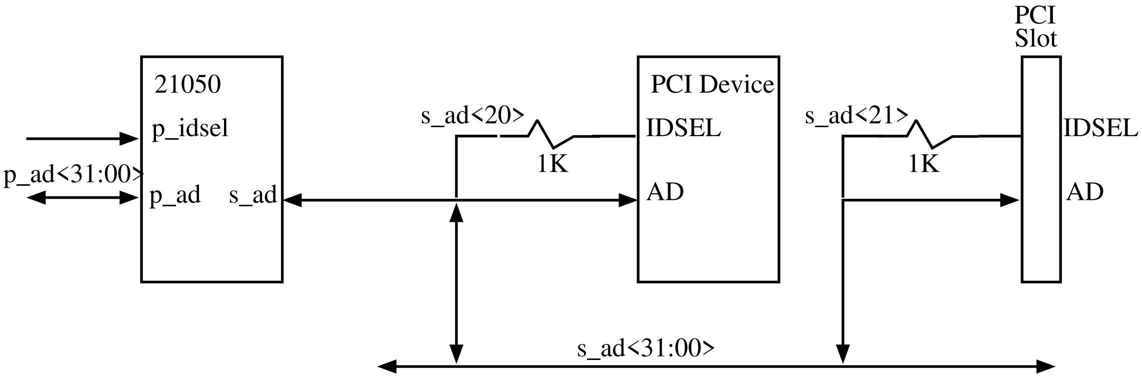
When implementing the 21050 in a system, each PCI device and PCI slot connected to the secondary PCI bus must have its IDSEL line connected to one of the **s_ad<31:16>** lines. To reduce the capacitive load on the **s_ad** line, this connection can be done through a 1K resistor, as shown in Figure 1. The 21050 supports address stepping on the address cycle of Type 0 transactions to allow additional time for the IDSEL signal to become valid when resistively coupled. This is the only time that the 21050 uses stepping, because the value of the IDSEL for data cycles and other transaction types is not important. If you think that the **s_ad** bus can support the additional capacitive load of the IDSEL pin, then you do not have to use the resistor.

When assigning secondary AD signals to secondary IDSEL signals on an expansion card, make sure that the device numbers are consistent with the required interrupt binding given in Table 2.

Note

Some early PCI host bridges automatically claim all configuration commands directed to device 0. Therefore, it is recommended that you avoid using device 0, if this situation might apply.

Figure 1 Secondary IDSEL Implementation Example



Data Synchronization and Interrupts

The *PCI Local Bus Specification* requires that either the interrupt handler (service routine) or the device that initiates the interrupt guarantees that all buffers are flushed between the device and the final destination. To accomplish this, the interrupt service routine of the device driver can perform a read of the device, or the device itself can perform a read of the last location written by the device. In either case, the read forces buffers between the device and the final destination to be flushed.

Interrupts originating from secondary bus devices are not routed through the 21050. However, the 21050 does offer two implementation-specific pins that can be used to implement hardware data synchronization: **s_dispst_l** and **s_bufne_l**. The following sections describe using these data synchronization pins, and implementing interrupt binding on option cards.

Using the 21050 Data Synchronization Pins

Although Section 6.3.4 of the *PCI Local Bus Specification* states that device drivers are ultimately responsible for guaranteeing consistency of data and interrupts, the 21050 does provide a hardware feature that you can use to synchronize data.

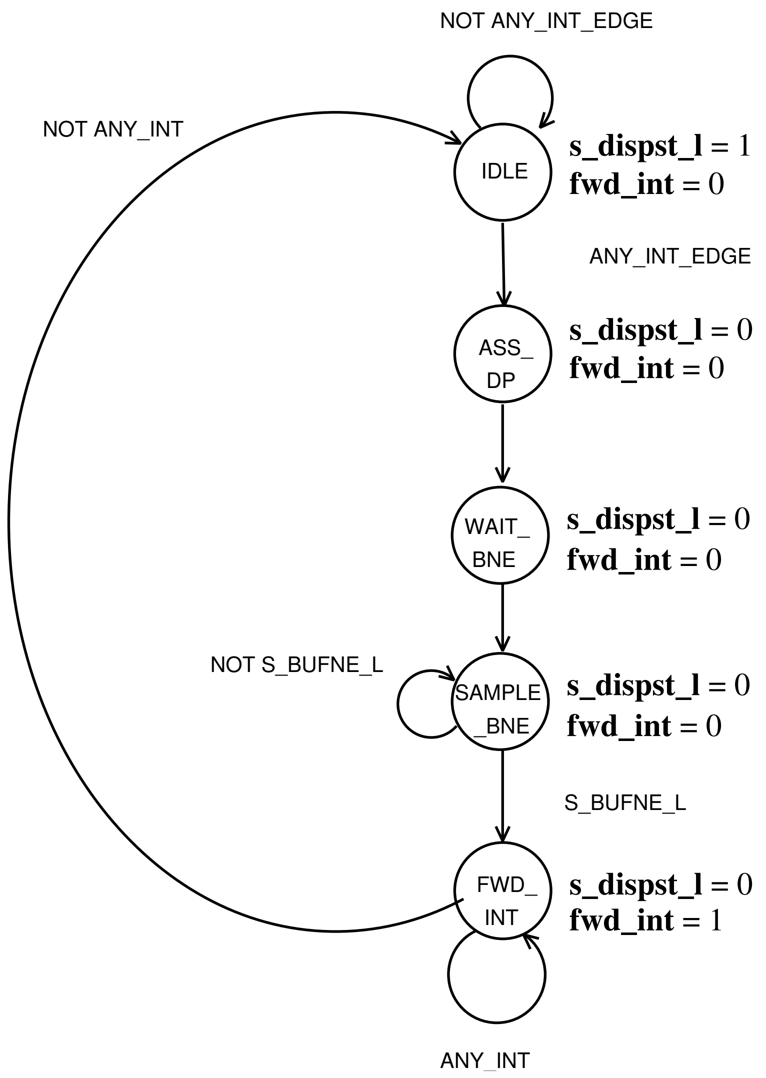
The 21050 implements the following pins:

- **s_dispst_l** (input)
  An asserting (falling) edge marks write data currently in both the upstream and downstream data buffers. If the Disable Posting bit in the 21050 configuration space is set, write posting in both directions is disabled as long as **s_dispst_l** remains asserted.
- **s_bufne_l** (output)
  This pin is asserted (low) when write data marked by the asserting edge of **s_dispst_l** remains in write buffers. **s_bufne_l** will assert two cycles after **s_dispst_l** assertion. **s_bufne_l** deasserts when all marked posted write data is flushed. This output is always enabled (except during diagnostics) and does not need a pullup.

Use these pins to block forwarding of interrupts until all posted write data is flushed. You can do this by enabling and disabling write posting dynamically by setting the Disable Posting bit in the 21050 configuration space and controlling the **s_dispst_l** pin.

Although use of the data synchronization pins is not specified, Figure 2 is an example of a possible implementation. This example assumes that an interrupt edge from a device causes **s_dispst_l** to assert to the 21050, marking any posted write data and disabling write posting. As soon as the marked write data is flushed, indicated by the deassertion of **s_bufne_l**, the interrupt is forwarded and **s_dispst_l** deasserts, enabling write posting again. Note that **s_bufne_l** is not valid until two cycles after the **s_dispst_l** assertion. Interrupt deassertion is always forwarded immediately. You can implement this logic in an external PAL. In this application the Disable Posting bit must be set. During initialization, set the Disable Posting bit in the 21050 implementation-specific configuration space.

Figure 2 Data Synchronization Implementation Example



Controlling Write Posting

You can also configure the 21050 to always enable or always disable write posting. For example:

- If **s_bufne_l** status is not used and write posting is always enabled:

- Tie **s_dispst_l** high through a 1K resistor.
- Leave **s_bufne_l** floating.

- If **s_bufne_l** status is not used and write posting is always disabled:
  - Tie **s_dispst_l** low through a 1K resistor.
  - Leave **s_bufne_l** floating.
  - Set the Disable Posting bit in the 21050 configuration space [Cfg Addr 40 (hex), bit 1].

Interrupt Binding on Option Cards

When PCI-PCI Bridges are used in option cards, a binding is required by the *PCI-to-PCI Bridge Architecture Specification* between the device number (as given in the Type 1 configuration address and, therefore, the IDSEL line) and the INT***x***# line it uses when requesting an interrupt. The PCI connector has only four interrupt lines assigned to it: INTA#, INTB#, INTC#, and INTD#. Multiple devices might have to share these lines.

Since only the BIOS knows how the PCI INT***x***# lines are routed to the system interrupt controller, a mechanism is required to inform the device driver of which IRQ its device will request an interrupt on. The interrupt line register stores this information. The BIOS code assumes the binding as listed in Table 2 behind a PCI-PCI Bridge and writes the IRQ number in each device. The interrupt binding defined in Table 2 is mandatory for option cards utilizing PCI-PCI Bridges.

Table 2 Interrupt Binding on Option Cards

Device Number on Secondary Bus	Interrupt Pin on Device	Interrupt Pin on Connector
	INTA#	INTA#
0, 4, 8, 12,	INTB#	INTB#
16, 20, 24, 28	INTC#	INTC#
	INTD#	INTD#
	INTA#	INTB#
1, 5, 9, 13,	INTB#	INTC#
17, 21, 25, 29	INTC#	INTD#
	INTD#	INTA#
	INTA#	INTC#
2, 6, 10, 14,	INTB#	INTD#
18, 22, 26, 30	INTC#	INTA#
	INTD#	INTB#
	INTA#	INTD#
3, 7, 11, 15,	INTB#	INTA#
19, 23, 27, 31	INTC#	INTB#
	INTD#	INTC#

Arbitration

The following sections describe primary and secondary bus arbitration.

Primary Bus Arbitration

The 21050 provides request (**p_req_l**) and grant (**p_gnt_l**) signals for primary bus access arbitration in accordance with the *PCI Local Bus Specification*. These signals connect to external arbiter logic provided by the system.

Secondary Bus Arbitration

You can configure the 21050 to use either its internal arbiter or an external arbiter for secondary bus access arbitration. The 21050 internal secondary bus arbiter supports six bus masters plus the 21050. The 21050 has six request inputs (**s_req_l<5:0>**) and six grant outputs (**s_gnt_l<5:0>**). When the internal arbiter is used, 21050 request and grant lines are internal to the chip.

To use the 21050 secondary bus arbiter:

- Tie the central function pin, **s_cfn_l**, to ground (low) through a 1K resistor.
- Connect secondary bus masters to the request and grant line pairs. The 21050 has two arbitration modes, both rotating, so order is not important (initial highest priority is at **s_req_l<0>**, however). Tie unused **s_req_l** input pins high through a 1K resistor. You can leave unused **s_gnt_l** output pins unconnected. These signals are compliant with the REQ# and GNT# specifications in the *PCI Local Bus Specification* and should be treated accordingly.
- The default mode rotates highest priority among all six external bus masters, but gives the 21050 highest priority on alternate transactions. You can select a second mode, which uses rotating priority among all bus masters including the 21050. Set this alternate mode by writing a 1 to the Arbitration Mode bit in the 21050 configuration space (Cfg addr 40h, Bit 0).

To use an external secondary bus arbiter:

- Tie the central function pin, **s_cfn_l**, high through a 1K resistor
- The 21050 reconfigures the **s_req_l<0>** and **s_gnt_l<0>** pins to act as external grant and request pins. Because **s_gnt_l<0>** is an output pin, the 21050 uses this pin as an external secondary bus request line. Connect this pin to one of the request lines of the external arbiter. Because **s_req_l<0>** is an input pin, the 21050 uses this pin as an external secondary grant line. Connect this pin to one of the grant lines of the external arbiter. Tie all **s_req_l<5:1>** high through a 1K resistor. You can leave **s_gnt_l<5:1>** unconnected.
- The external arbiter must use some kind of fairness algorithm. If a fixed priority algorithm is used, a deadlock condition might occur.

Technical Support, Ordering, and Associated Literature

This section describes how to order DECchip information and technical support, as well as how to order DECchip products and associated literature.

Calling the DECchip Information Line for Information and Technical Support

Use the following numbers to call the DECchip Information Line:

United States and Canada	**1-800-332-2717**
TTY (United States only)	**1-800-332-2515**
Outside North America	**+1-508-568-6868**

Ordering DECchip Products

To order the DECchip 21050 PCI-to-PCI Bridge, contact your local Digital sales office. Your sales representative may be able to help you take advantage of discounts and volume pricing.

You can order the following DECchip products from Digital.

Product	Order Number
DECchip 21050 PCI-to-PCI Bridge	21050-AA

Ordering Associated DECchip Literature

The following table lists some of the DECchip literature that is available. For a complete list and for information about ordering, contact the DECchip Information Line.

Title	Order Number
DECchip 21050 PCI-to-PCI Bridge Data Sheet	EC–Q9XVA–TE
DECchip 21050 PCI-PCI Bridge Product Brief	EC–Q9Y0A–TE
DECchip 21050 PCI-PCI Bridge Configuration: An Application Note	EC–Q9Y4A–TE
DECchip 21050 PCI-PCI Bridge Hardware Implementation: An Application Note	EC–Q9Y3A–TE

Ordering Third-Party Literature

You can order the following third-party literature directly from the vendor.

Title	Vendor
PCI Local Bus Specification, Revision 2.0 *PCI-to-PCI Bridge Architecture Specification,* Revision 1.0	PCI Special Interest Group M/S HF3-15A 5200 N.E. Elam Young Pkwy Hillsboro, Oregon 97124-6497 (503) 696-2000