Optimizing the PCI Interface Configuration for the RC32365

Application Note AN-435

Notes

By Rakesh Bhatia

Revision History

June 10, 2003: Initial publication.

Background

The RC32365 device is a member of the IDT™ Interprise™ family of integrated communications processors. It provides a single chip solution for virtual private network (VPN) platforms with bandwidth requirements ranging from full T1/E1 through fractional T3. Featuring a MIPS CPU core, the device also includes a memory controller supporting SDRAM memory, a PCI interface featuring an on-chip arbiter to simplify the design of embedded systems, and a security module that accelerates IPSEC performance.

This document discusses the various programmable configuration settings that affect data transfer rates between the processor and any other device linked on the PCI bus.

PCI Throughput

PCI throughput is defined as the amount of data transferred per unit time. Although this is the standard definition, most system designers using the PCI bus realize there is a certain amount of overhead involved in transferring data which must be taken into account when determining actual throughput. For example, in communications applications such as ATM or Ethernet, the overhead consists of cell headers or packet headers. Depending on the system, there will be other overheads such as stack pointers, encryption code, etc. This is one of the factors that makes the real-world performance highly system-dependent.

In situations where both the PCI interface and the CPU wish to access the IPBus, they must take turns. As the external CPU activity increases because of cache misses or the need to manipulate data in memory, the PCI is increasingly blocked from the IP bus, resulting in decreased PCI throughput. This is another reason the real-world PCI throughput is highly dependent on the operating system.

Optimizing the System PCI Throughput

In order to optimize the PCI throughput at a system level, general principles for increasing performance should be followed. These principles include but are not limited to:

- *Using advanced commands like MRL, MRM, and MWI.*
- *The burst size should be adjusted according to the maximum the system can handle. Note that the RC32365 has PCI target input and output FIFOs that are 64 words in size, which can help when using larger burst sizes.*
- *Using the proper arbitration between all masters.*

Optimizing RC32365's PCI Throughput

Although the actual throughput will vary from system to system, some configuration settings are provided in the processor's registers to ensure optimal performance of the PCI interface.

PCI Master Transactions

For PCI master transactions, throughput cannot be increased by changing programmable settings. However, some general rules, such as using bursts and using advanced operations, still apply. The behavior of PCI master transactions is explained in various sections of the RC32365 User Reference Manual.

Notes

Also, the RT field of the Local Base Address Control register provides some control in response to IPBus master reads that map through PCILBAx register to the PCI bus. When the MSI bit configures PCILBAx to use memory transactions and when the MSI bit is set, IPBus read operations use PCI I/O read transactions regardless of the state of this bit. However, this bit can be configured to either issue memory read transaction(s) on the PCI bus and pass data to IPBus as it becomes available or issue memory read line transaction(s) on the PCI bus and prefetch entire cache lines in anticipation of future IPBus reads.

Memory read line transactions resulting from IPBus master read transactions will cause the PCI bus interface to issue a memory read line burst transaction that transfers either an entire cache line or eight words, whichever is smaller. The PCI bus interface will attempt to perform burst PCI memory write transactions whenever possible. As long as data exists in the CPU master output FIFO, whose address is equal to that of the current data quantity being transferred plus four, and as long as the Master Latency Timer has not expired, then the PCI interface will add a data phase to the memory write transaction in progress.

PCI Target Transactions

Some programmable settings are provided to control the throughput of PCI target transactions. The Target Control Register is provided to help the user configure various settings as per system requirements. Although the Target Control register provided in the RC32365 is not as sophisticated as the Target control register provided in some of the other Interprise™ processors, it does offer some flexibility to the user.

PCI Target Control Register

The PCI target control register provides various means by which PCI performance can be enhanced. These register fields and the affect of using them are described below.

RTIMER

The Rtimer field of the RC32365's Target control register specifies the number of PCI clock cycles the PCI interface will wait for the first data of an access before issuing a retry. PCI Specification 2.2 sets the maximum limit of this timer at 16 PCI clock cycles, but in some systems it may be necessary to extend this limit. The minimum retry timer value is eight. Values less than eight are aliased to eight. In essence, increasing the value of this field ensures that the PCI device has enough time to respond. Instead of issuing a retry after the default 16 clock cycles, the limit can be increased up to 255 clock cycles and a repetition of the entire request/grant process can be avoided.

DTIMER

The disconnect timer field assists in delaying the issuance of a "disconnect" in a PCI transaction. This can be useful if slower PCI devices are being used. If the disconnect timer is programmed to a higher number, the interface waits for additional clock cycles before issuing a "disconnect", thereby allowing the slower device to catch up. It also prevents the entire arbitration process from being re-run. If a disconnect is issued, the prefetched data in the target read FIFO is flushed and will need to be fetched again. Since the value of this field will depend on the types of devices used, it is important that the optimum value be calculated by trial and error.

RDR and DDT

These fields provide additional flexibility that may be needed in some systems. The use of the "disable discard timer" field and the "retry when delayed read" field will vary from system to system and may or may not affect the throughput. These fields are provided as a feature rather than as controls to increase the throughput. Note that the PCI bus interface supports only one pending delayed read, and if a read is attempted while a delayed read is pending, the transaction is retried.

Base Address Registers

A "perform prefetch" field is provided in the 4 Base address control registers. This field must be used if any of the following conditions exists:

- *More than one word needs to be read*
- *Advanced commands such as MRL and MRM are to be used*
- *To fetch more than 1 word on the local bus.*

Notes

If the "perform prefetch" is enabled and the MRL /MRM commands are used, 8 to 16 words can be prefetched. Therefore, when the next word is to be read by the PCI device, the data would already exist in the target read FIFO. When this bit is selected, the user has the flexibility to fine tune the prefetching by using different PCI commands, provided the PCI master allows the use of such commands.

Memory Read (MR)

The MR bits control the behavior of PCI memory read transactions and can be tweaked to either perform no prefetching or to treat a MR transaction as either MRL or MRM transaction.

Memory Read Line (MRL) Prefetching

This bit controls the behavior of PCI memory read line transactions. It allows for memory read line transactions to be treated as MRM transactions.

Memory Read Multiple (MRM) Prefetching

The MRM bit controls the behavior of PCI memory read multiple transactions on the local bus and provides for a prefetched 16 word to burst from local address space whenever there are less than 8 words in the PCI target output FIFO. It can also be used in the Aggressive Prefetching mode which keeps prefetching 16 word bursts from local address space as long as room exists for them in the PCI target output FIFO.

The affect of using these controls would be system-specific. For example, if a large block of data were to be transferred and the software issues single mem_read commands, it will actually help to convert these commands to memory_read_multiples. However, if a single_read was truly intended, using these bits wastes the PCI bandwidth. These bits can be used in conjunction with the "perform prefetch" bit in the BAR register discussed above.

Command Registers

Memory Write Invalidate Bit

The MWI bit [bit 4 of the Command register] can be used to burst up to 8 words on the local bus. Note that in some system-dependent applications, reducing the target write burst size on the local bus can help to balance the target read vs. target write throughput.

FBB Fast Back to Back Enable

When this bit is set, the PCI bus interface is allowed to generate fast back-to-back transactions to different agents as described in PCI Specification 2.2, section 3.4.2. When this bit is cleared, fast back-to-back transactions are only performed to the same agent. Essentially, this avoids contention on the bus and thereby improves performance. Note: the RC32365 never generates fast back-to-back transactions.

Other Features

The RC32365 provides other features that the system designer should be familiar with when optimizing the overall system throughput. These features are discussed below.

Target Ready Time-out Register

The TT field in this register indicates how many PCI clock cycles the PCI bus interface will wait as a master for the assertion of TRDYN. Setting this field to zero results in an infinite time-out period (i.e., no time-out).

Retry Limit Register

The RL field of this register indicates how many times the PCI bus interface will retry a transaction. Setting this field to zero results in an infinite retry limit (i.e., no limit).

PCI Arbitration register

Features that control idle bus behavior and arbiter parking have also been added in the RC32365.

Notes

EAP Bit in PCIC Register

When the EAP bit is set and the PCI bus interface is configured to operate in PCI host mode with an internal arbiter, the PCI bus parking feature is enabled. The enable arbiter parking function allows parking of the PCI bus on the last master that was granted the bus in the event that no other master requests the bus after the last master is done with it. This helps to save some time because the bus is immediately available to the bus master if it should require the use of the bus again. The master will not need to assert its REQ# and can initiate the transaction by sampling the bus idle and having its GNT# asserted.

Idle Grant Mode

The Idle Grant Mode bit controls the operation of the internal arbiter when the PCI interface is configured to operate in PCI host mode with internal arbiter. When the internal arbiter is used and this bit is cleared, then the arbiter operates in a static idle grant mode. Once a grant is asserted while the PCI bus is idle, the grant will remain asserted until the requested transaction completes. When the internal arbiter is used and this bit is set, then the arbiter operates in a dynamic idle grant mode. This means that while the PCI bus is idle, the arbiter may take away a grant from one master and pass it to another.

Idle grant mode can be used in systems that have multiple devices where one is a high priority master and the other device is a slower device. The arbiter may take away the grant from one master and issue it to another master. If the arbiter idle grant mode bit is not used, the higher priority master will have to wait until a timeout occurs for the slower, lower priority device.

Both of the arbiter features mentioned above have been added to further facilitate fairness between PCI devices and should help the performance of most systems. Therefore, it is recommended that both these features be used.

Summary

- Use advanced commands such as MRM, MRL, and MWI.
- Use longer burst whenever possible.
- The arbitration method between different masters should ensure that it does not adversely affect system performance.
- The "perform prefetch" bit in the BAR registers must be used when advanced commands are to be used.
- Because the MW/MWI bit can adversely affect a system, they should be used after carefully considering the system requirements. The same applies for the MRML bits.
- In most cases, the Rtimer bits will have a positive affect on the system performance and should, therefore, be used.
- For slower systems, delayed read and write bits should be considered.
- The Dtimer is used in conjunction with the prefetching mode as mentioned above.
- The arbiter parking and bus idle modes could have an affect on systems where a certain master is constantly requesting the bus or if the system has a master that is slower than other devices on the PCI interface. In such cases, these features can be used to optimize overall system performance.

Conclusion

Although a significant amount of PCI efficiency in a system depends on the PCI cards used, configuring the RC32365 as described in this document should improve overall system performance.

Acknowledgements

The following people contributed to this document with relevant technical information:

Paul Snell, IDT Inc.

Mitrajit Chatterjee, IDT Inc.

Notes

References

79RC32365 User Reference Manual

PCI Local Bus Specification. Rev. 2.2 - PCI special interest group

PCI System Architecture. 4th edition - Mindshare, Inc., Tom Shanley and Don Anderson

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.