Optimizing the PCI Interface on the RC3233x

Application Note AN-366

Notes

By Rakesh Bhatia

Revision History

April 30, 2002: Initial publication.

September 12, 2002: Removed pointer for the BAR register in the PCI Target Transactions section.

October 18, 2002: Changed device designation to RC3233x which includes 3 devices: RC32334, RC32333, and RC32332.

June 1, 2004: Revised PCI Arbitration Register section.

Background

The RC3233x are integrated processors that combine a 32-bit MIPS instruction set architecture (ISA) CPU core with a number of on-chip peripherals to enable direct connection to boot memory, main memory, IO, and PCI. The RC3233x also includes system logic for DMA, reset, interrupts, timers, and UARTs. The RC3233x components integrate all of the peripherals commonly associated with an embedded system to reduce board real estate, design time, and system cost.

PCI Interface

PCI throughput is defined as the amount of data transferred per unit of time. Although this is the standard definition, most PCI users realize that there is a certain amount of overhead involved in transferring data, and this overhead is accounted for in actual throughput. For example, in communications applications such as ATM or Ethernet, the overhead is cell headers or packet headers. Depending upon the system, there will be other overheads, such as stack pointers, encryption code, etc. Overhead is one of the factors that makes the real-world performance highly system-dependent.

Any transaction that is targeted to or from the SDRAM must be transacted via the IPBus of the RC3233x. In situations where both the PCI and the CPU wish to access the SDRAM, they must take turns via the IP Bus. As the external CPU activity increases because of cache misses or the need to manipulate data in memory, the PCI is increasingly blocked off the IP bus and PCI throughput begins to drop. This is another reason the real-world PCI throughput is highly operating system-dependent.

Optimizing the System PCI Throughput

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

- *Using advanced commands like MRL, MRM, and MWI*
- *Using the proper arbitration between all masters*
- *Adjusting burst size according to the maximum the system can handle. Note that the Y revision of RC32334/RC32332 has larger size FIFOs compared to the Z revision, which can help when using larger burst sizes (revision Z does not apply to the RC32333).*

Optimizing RC3233x PCI Throughput

Although the actual throughput will vary from system to system, some configuration settings are provided in the RC3233x to ensure optimal performance of the microprocessor's PCI interface.

Notes

PCI Master Transactions

For PCI master transactions, there are no programmable settings that can be changed to increase throughput. However, some internal hardware changes were made in the Y revision microprocessor which result in better handling of data internally when compared to the older Z revision. For a complete description of these changes, refer to IDT Application Note AN-350, RC32334/RC32332 Differences Between Z and Y Revisions.

PCI Target Transactions

Several programmable settings are provided to control the throughput of PCI target transactions. Also, changes were made to the internal hardware in the Y revision to ensure better performance over the older Z revision. These changes are also discussed in IDT AN-350. A new register, Target Control Register, is provided to help users configure various settings according to system requirements. Additional BARs (base address registers) are provided and existing BAR registers are modified.

The following example shows that the optimum setting for the PCI target control register in a typical application.

Example: A 1404 byte Ethernet packet is to be sent from one Ethernet card to another via the RC3233x device. The first card will perform target writes where the RC3233x is the target, and the second card will read from the RC3233x by performing target read operations. The CPU clock is 66Mhz and PCI clock is 33Mhz.

In this example, since it is feasible to perform bursts with this packet size, by setting the MW/MWI bit to 1, the PCI target control register at address b80020A4 is set to perform bursts on the local bus. The eager prefetch bit is set to expedite the read transactions. Also, the MRML bit is set to get better performance on read commands. The Rtimer is set to 24 and the Dtimer is set to 16 clocks for optimum performance. The derived value of the PCI Target Control register is 0x0D101018. The effect of not setting the prefetch bit in the BAR register vs. setting it and the effect of using the optimum configuration for the PCI target control register are shown in Table 1 below. A detailed explanation of these bits is provided in this section.

CPU clock: 66 MHz. PCI clock: 33MHz

Prefetch bit in BAR Register	PCI Target Control Register Value	Target Write Throughput MB/sec	Target Read Throughput MB/sec	Overhead/ Processing Time µsec	Total Throughput MB/sec
Enabled	0x0D101018	13.12	51.75	268.05	6.98
Enabled	0x01080810	13.12	21.33	270.01	6.34
Disabled	0x01080810	13.12	18.20	265.99	6.24
Disabled	0x0D101018	13.12	18.20	265.80	6.24

Table 1 PCI Throughput with Various Configuration Settings

Various values of the PCI Target Control register were tried by setting each feature one at a time. The above optimum settings were derived for this application. Although each application will be different, the basic theory behind the optimization stays the same: use bursts whenever possible.

In another application, an external master performs burst memory reads from the RC3233x device as target, and reads are 128 byte bursts. Figures 1 and 2 below show the actual arbitration and how it affects the PCI throughput when optimum settings are used. Specifically, Figure 1 shows the optimum setting to be 0x7ef82828 and the RC3233x latching onto the PCI bus and holding the transaction for its entire completion. There are no retries. By comparison, Figure 2 shows a scenario where the default setting of 0x01080810 is used for the PCI Target Control register, resulting in several retries.

Notes

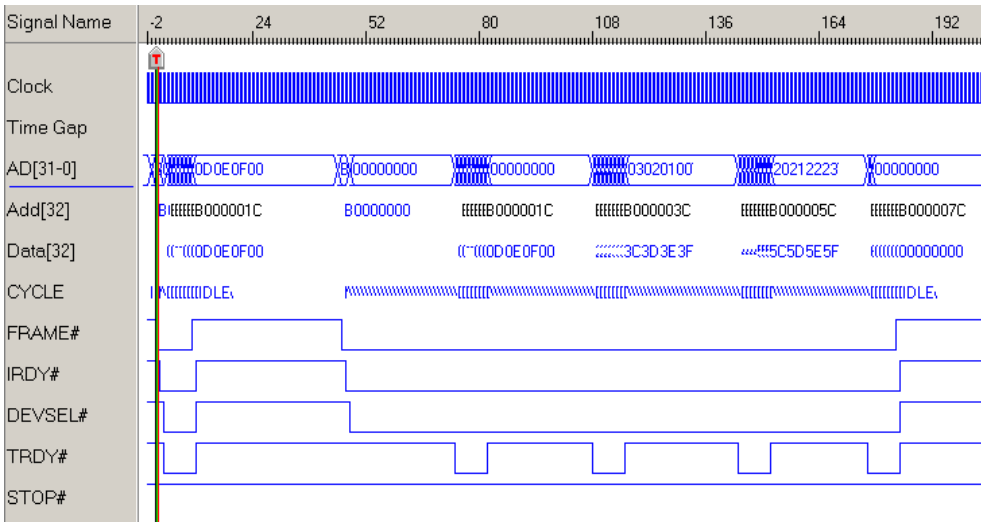

Figure 1 RC3233x Target 128-byte Burst Reads, Target Control reg=0x7ef82828

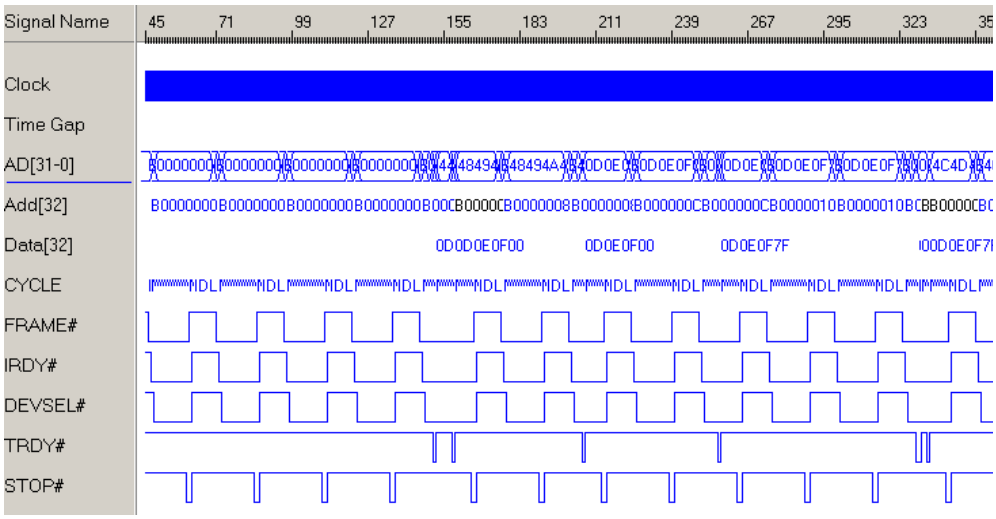

Figure 2 RC3233x Target Reads, with Default Settings Showing Extensive Retries

The optimal settings for the PCI Target Control register were derived from the above two examples and are shown in Table 2 below. It is recommended that these settings be followed initially and that the system be verified with other settings. For example, having a threshold of 8 may not be suitable for all applications.

Bit	Value	Feature
31	0	Reserved
30:27	1111	Enable eager prefetch mode
26	1	Burst 8 words on local bus
25:24	10	Set threshold to 8
23:20	1111	Prefetch 8 words on mem_rd and mem_rd_line commands

Table 2 Optimal Settings for the PCI Target Control Register (Page 1 of 2)

Notes

Bit	Value	Feature
19	1	Default value for DDT
18	0	Default value for RDR
17:16	00	Reserved
15:8	0001 0000	Dtimer set to 16 clock cycles
7:0	0001 1000	Rtimer set to 24

Table 2 Optimal Settings for the PCI Target Control Register (Page 2 of 2)

Using Table 2 above, the optimal value for the PCI target control register is 0x07EF81018. The following sections describe the functions of these bits in detail and the affect of using these bits.

Base Address Registers

A "prefetch" bit is provided in the 4 BAR registers. This bit must be selected when:

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

If this bit is selected 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 use of such commands.

PCI Target Control Register

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

Eager Prefetch bits [30:27]

Bits [30:27] allow for eager prefetching of data on certain types of commands, such as mem_read_line and mem_read_multiple. It has no affect on single read commands, such as mem_read. If a target read transaction does not disconnect after the initial prefetch, the next block of data is prefetched, provided there is room in the Target Read FIFO and the eager prefetch facility is enabled by bits [30:27]. It is recommended that the system designer consider using this feature, since it will improve system performance as follows: the prefetch buffer will be loading from the memory on the destination bus, and the PCI bus master would be reading the data from the other end of the buffer. There is a trade-off. If the system is not moving blocks of data, but rather doing isolated reads from specific locations, enabling these features will degrade system performance rather than improve it. The local bus will be prefetching 8 words when it actually needs to fetch only one word, wasting bandwidth.

Memory Write / Memory Write Invalidate bit [26]

The MW/MWI bit [26] of the Target Control Register can be used to burst up to 8 words on the local bus. This is a new feature in the Y revision and allows for a PCI burst transaction of 8 words to be transferred as an 8 word across the local bus. The default value of this bit is 4, which provides for some flexibility as far as replacing the Z with the Y revision is concerned. 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.

Threshold for Target Write FIFO bits [25:24]

The intention of Threshold bits [25:24] is to wait until at least a certain number of data words are free in the FIFO before accepting any new write commands. For example, if a threshold setting of 8 is matched with a burst size of 8 and the master wants to burst 8 words to the RC3233x, there is always space in the FIFO for the master to do so.

However, if the threshold is set to 4, there is a chance that the burst will get broken into two 4-byte chunks, which would result in 4-byte bursts on the local bus, and that would probably have a negative impact on system performance.

Notes

Note that the FIFO cannot hold two complete 8-word bursts, so a threshold of 8 may slow PCI throughput since the FIFO may periodically go empty. Since the overall bus utilization is what most system designers would be concerned about, rather than absolute maximum PCI throughput, the eight word threshold would probably do more good if used (compared to not using it).

MRML bits [23:20]

The MRML bits[23:20] are used to prefetch 8 words on memory_ read and memory_ read_ line target reads. In other words, these commands would behave like memory_read_multiple command. The affect of using these bits would be system specific in the sense 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 prefetch bit in the BAR register discussed above.

Delayed Transactions bits [19:18]

The use of the “disable discard timer" and "retry when delayed read" bits will vary from system to system and may or may not affect the throughput. These bits 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. If a read is attempted while a delayed read is pending, the transaction is retried.

Dtimer bits [15:8]

The disconnect timer bits [15:8] assist 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 more clock cycles before issuing a "disconnect", thereby allowing the slower device to catch up. It also prevents the whole 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 bit will depend on the types of devices used, it is important that the optimum value be calculated by trial and error.

Rtimer bits [7:0]

The retry timer default value is 16, per PCI 2.2 Specification. However, a good number of devices respond in 20 PCI clock cycles. Instead of issuing a retry after 16 clock cycles, the limit can be increased to 20 clock cycles, and the whole request / grant process can be avoided from being run again. In certain system applications, IDT has noticed a 2x throughput when the Rtimer was set to 24.

Other Features

The RC3233x provides other features that the system designer should be familiar with optimizing the PCI throughput and the overall system throughput. These features are discussed below.

PCI Arbitration Register

Although the PCI arbitration register located at 0x1800_20E0 is primarily used for choosing internal or external arbiters and arbitration modes, it provides control bits that affect the PCI performance to some extent. Features that control idle bus behavior and arbiter parking have been added in the RC3233x.

Arbiter Idle Grant Mode bit [4]

The arbiter idle grant mode bit changes the number of cycles that a master is allowed to hold the bus without initiating a transfer. Bit 4 (arbiter idle grant mode) can be used in systems that have multiple devices of which one is a high priority master and the other device is a slower device. The arbiter may take away the grant from a master and issue it to the other master. If the arbiter idle grant mode bit is not used, the higher priority master will have to wait until the timeout occurs for the slower, lower priority device.

Arbiter Park Mode Enable bit [3]

The arbiter park mode enable 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, should the master again require use of the bus. The master will not need to assert its REQ# and can initiate the transaction by sampling the bus idle and having its GNT# asserted.

Notes

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

In some cases, enabling the Arbiter Idle Grant Mode may actually result in undesirable consequences. Although the likelihood of such a case is rare, system designers should be familiar with the functionality of the Arbiter Idle Grant Mode. For a more detailed discussion of this issue, refer to the Arbiter Idle Grant Mode section in the RC3233x Design Guide located at www.idt.com.

Summary of Guidelines for Optimizing PCI Performance on the RC3233x

- Use advanced commands such as MRM, MRL, and MWI.
- Use longer bursts whenever possible.
- The arbitration method between different masters should ensure that it does not adversely affect the system performance.
- The prefetch bit in the BAR registers must be used when advanced commands are to be used.
- To fully utilize the Eager Prefetch mode, the target disconnect timer (Dtimer) should be set high enough to avoid practically all disconnects, and the master should issue memory read multiple commands using large blocks.
- Since the MW/MWI bit and the Threshold bits in the PCI Target Control Register 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, the delayed read and write bits should be considered.
- The Dtimer is used in conjunction with the prefetching and Eager prefetching modes, 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 the overall system performance.

Conclusion

Although a significant amount of PCI efficiency in a system depends on the PCI cards used in the system, configuring the RC3233x as described in this document should help improve overall system performance. Some of the general guidelines to improve PCI efficiency were also covered along with specific system configurations. These steps can be used to achieve maximum throughput and utilize the PCI interface in an efficient manner.

Acknowledgements

The author would like to thank the following contributors who provided relevant technical information for this document:

Paul Snell and Kasi Chopperla: Integrated Device Technology.

References

1. IDT AN-350: RC32334/RC32332 Differences Between Z and Y Revision. Author, Paul Snell.
2. IDT TN-45: Software Configuration of PCI Bridge on RC3233x Integrated Processors. Author, Harpinder Singh.
3. 79RC3233x User Reference Manual.
4. PCI Local Bus Specification. Rev. 2.2 - PCI special interest group.
5. 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 Jan 2024)

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.