Integrated Device Technology, Inc.
2975 Stender Way, Santa Clara, CA - 95054
Phone #: (408) 727-6116 Fax #: (408) 727-2328

Errata Notification

EN #: IEN01-03 Errata Revision #: 11/2/01

Issue Date: December 14, 2001 Effective Date: November 2, 2001

Product Affected: IDT79RC32V334 / IDT79RC32V332

ERRATA DETAILS: **ATTACHMENT** ☒ Yes ☐ No

- ☐ Data Sheet Errata
- ☒ Device Errata
- ☐ Device Manual Errata
- ☐ Application Notes Errata
- ☐ Other (Please Specify)

IDT Web Site Location:

http://www.idt.com/products/pages/Integrated_Processors-79RC32334.html

CUSTOMER NOTIFICATION:

As a result of Errata listed above, IDT has decided to notify you through this Errata Notification.
We recommend you to review this notification in details prior to the use of affected product in your application.
Please use acknowledgement below or E-Mail to request additional information.

If you have any questions, please feel free to contact your local IDT distributor or sales representative.

Customer: _____ E-Mail Address: _____

Name/Date: _____ Phone #: _____

Title: _____ Fax # : _____

CUSTOMER COMMENTS: _____

RC32334/RC32332 Device Errata

Notes

Supplemental Information

This Device Errata reflects revision 1.0 silicon and supplements information found in the documentation for this device. Silicon revisions can be identified as follows:

ZA is revision 1.0

ZB is revision 1.1

Note: The RC32334 and RC32332 have an optional clock (*output_clk*, pin #C8 on the RC32334 and pin #25 on the RC32332). This clock should not be used. It should be disabled at boot time. The input clock to the RC32334/RC32332 (*cpu_masterclk*, pin #E13 on the RC32334 and pin #64 on the RC32332) should be used to drive the SDRAM and other memory/IO sub-systems.

Revision History

February 28, 2000: First version of errata for Revision 1.0 silicon.

April 12, 2000: Added items #3 and #4.

May 31, 2000: Deleted item #2 and updated item #3, which had been #4.

October 13, 2000: Added item #4.

November 15, 2000: Revised document to include RC32332.

January 19, 2001: Added items #5 and #6.

February 27, 2001: Added item #7.

May 8, 2001: Added items #8 and #9.

June 6, 2001: Revised the Note under Supplemental Information.

June 22, 2001: Added item #10.

November 2, 2001: Added item #11.

Descriptions and Workarounds

Item #1 - Parity Error

Issue: In the RC32334/RC32332, the PCI command register includes a bit (bit 6) which can be set to request that the devices ignores address parity errors. In the RC32334/RC32332 device, the PCI core will continue to issue an exception even if this bit is set.

Workaround: The RC32334/RC32332 must service the exception and then reset the error condition.

Fix: There are no plans to fix this item.

Item #2 - Operation of CPU Reset through EJTAG register

Issue: The EJTAG control register includes a bit (bit 16) which when set to 1, can force the processor into a software reset. when reset, the CPU should fetch from debug ROM address (ff20-0200). However, in revision 1.0 of the RC32334/RC32332, the CPU fetches from the normal ROM address (BFC0-0000).

Workaround: None

Fix: There are no plans to fix this item.

Item #3 - Boot of CPU into Debug Mode

Issue: The debugboot signal (pin 10) is used during reset to force the RC32334/RC32332 to take a debug exception at the end of the reset sequence instead of a standard reset exception. This enables the CPU to boot from the ICE probe without having external memory connected (or operational). In revision 1.0, when the debugboot signal is asserted, the CPU incorrectly fetches from normal ROM address (BFC0-0000) rather than debug ROM address.

Workaround: On revision 1.0 of the RC32334/RC32332, the debug mode can only be entered once a debug exception is taken. IDT recommends that this signal be pulled low on the board using a 10k ohm resistor.

Fix: There are no plans to fix this item.

Item #4 - DMA Arbitration

Issue: When the DMA arbitration scheme is configured for the round robin algorithm, fair arbitration does not occur if an even number of DMA channels are requesting the bus *and* one of the following situations occurs:

- During the present DMA operation, an uncached CPU 32-bit instruction fetch occurs to the Memory Space with the Bus Turnaround Control Register (BTA) set to a value of either 1 or 0.
- During the present DMA operation, cached CPU instructions are executed (except when a CPU data access followed by an I-cache miss occurs).

Specifically, if two DMA channels request the bus, the arbiter will continue to prioritize the current DMA channel having higher priority until this completes.

If four DMA channels request the bus, the arbitration incorrectly jumps between either the even priority channels or the odd priority channels.

Workarounds:

- Operate with an odd number of active DMA channels.
- Operate the device using the fixed arbitration scheme.

Fix: There are no plans to fix this item.

Item #5 - Use of Fixed Priority Arbitration Scheme In Internal PCI Arbiter

Issue: The RC32334 and RC32332 integrated processors include an on-chip arbiter in the PCI mode that is used to grant the PCI bus to additional PCI peripherals wishing to take control of the PCI bus. For a system with two external PCI bus master devices, A and B, with device A assigned the higher priority, an issue occurs when device B requests the PCI bus and device A issues a retry. In this situation, the on-chip arbiter will not grant the bus to device B. Instead, it will wait 16 PCI clock cycles and then allocate the PCI bus ownership to the highest priority device requesting the bus. If device A issues a bus request inside 16 PCI clock cycles, it will again be granted the bus.

Workaround: This is expected to be an issue only when two or more high bandwidth peripherals are connected to the PCI Bus. In cases where the higher priority Masters are not often idle, then the use of the rotating priority arbitration mode of the PCI Arbiter, instead of the fixed priority arbitration mode, is advised to ensure data fairness.

Fix: There are no plans to fix this item.

Item #6 - PCI Arbitration Back to Back Grants

Issue: When the PCI bus is in the Idle State (pci_frame_n and pci_irdy_n de-asserted), the PCI arbiter should wait at least one clock after de-asserting a grant before asserting a different grant. This prevents a potential address stepping master from bus contention with the previous master or with a higher priority master.

In the case where a new, higher priority request occurs on the last clock of a transaction, the PCI arbiter then allows concurrent grant de-assertion and assertion to occur 2.0 clocks later, even though 2.0 clocks later the PCI bus is in the Idle State.

Workaround: This problem can be fixed externally. There are two basic approaches, depending on the system:

a. Ignore the problem and, if possible, turn address stepping off. Note that address stepping typically occurs during configuration reads and writes which can be sequenced by the PCI host to ensure they do not collide with other masters.

b. Add a combinational logic fix externally. This approach requires margin on pci_gnt_n setup times.

```
// note: Actual implementation should include pci_rst_n, which is
//       omitted here for the sake of clarity.
assign new_pci_gnt_n[0] = ~(~pci_gnt_n[0] & ~idle_fix[0]);
always @(posedge pci_clk) begin
  idle_fix[0] <= (pci_frame_n & pci_irdy_n)
                & (~pci_gnt_n[2]|~pci_gnt_n[1]);
end
```

Fix: There are no plans to fix this item.

Item #7 - Reset Must Be Synchronous to cpu_masterclk.

Issue: If the rising edge of cpu_coldreset_n leads cpu_masterclk by 3.75 ns $\pm$ 0.25 ns during the initial power-on reset, the boot width mode bit may be read incorrectly in the RC32334 and RC32332.

Workaround: In order to guarantee proper operation after power-on reset, use a synchronized reset on both the RC32334 and RC32332. This can be accomplished by double registering the reset through an EPLD or two D-flops and clocking the reset signal out with cpu_masterclk.

If cpu_masterclk is used to clock-out the reset signal changes, then the reset level transition will lag cpu_masterclk, and the problem condition will be avoided.

Fix: There are no plans to fix this item.

Item #8 - The UART Receive FIFO Timeout Interrupt

Issue: The UART receive FIFO timeout interrupt does not work as stated. Specifically, the UART receive FIFO timeout interrupts are not masked/unmasked with the IER register, field RDA (bit [0]).

Workaround: The UART receive FIFO time-outs are incorrectly masked/unmasked with IER register, field RLS (bit [2]) instead of RDA (bit [0]). So, to enable the receive timeout interrupt in FIFO mode, set IER bit [2] to one. A FIFO timeout interrupt will then occur if at least one character is in the receive FIFO and the most recent character was received more than 4 character times ago. This prevents bytes from sitting in the FIFO for long periods of time before the FIFO trigger level is reached.

Fix: There are no plans to fix this item.

Item #9 - UART Line Status Register: THR Bit

Issue: When the UART is in the 16550 (buffered/FIFO) mode, the user can expect the Line Status Register, field THR Empty (bit[5]), to be set to 1 when the transmit channel is ready to accept a new character for transmission. The 16550 specifications state that THR Empty can be guaranteed to be 1 when the Transmit Holding Register is empty and guaranteed to be a 0 when the Transmit Holding Register is not empty. However, this does not occur in the RC32334/RC32332.

Workaround: Do not use THR Empty by itself. Instead, use the THR Empty and TX Engine Status (bit[6]) fields together. Use the following logic:

TX Engine Status = 0, THR Empty = 0
This means that Transmit Shift Register (TSR) and THR both have bytes in them (if in FIFO mode, at least one byte is in the FIFO).

TX Engine Status = 0, THR Empty = 1
Do not use this combination. This does not work.

TX Engine Status = 1, THR Empty = 0
Not valid.

TX Engine Status = 1, THR Empty = 1
TSR and THR are both empty. OK to send next byte of data

Fix: There are no plans to fix this item.

Item #10 - PCI Interface May Corrupt Data When Using DMA

Issue: DMA writes from memory to PCI may corrupt data in PCI write FIFO.

It is very possible that a situation occurs where a DMA write transaction overwrites a valid word in the internal write FIFO of the PCI interface under the following conditions:

- A DMA channel is programmed to use 4-word burst size
- *And* while the DMA channel is doing 16 bytes (4 words) DMA write from memory to the PCI write FIFO
- *And* when the RC32334/332 initiates the transaction.

This errata occurs in both Master and satellite PCI modes. However, it only affects slow PCI systems where the target of the write transaction does not empty the RC32334/332 write FIFO in a burst fashion.

Under the above conditions, a DMA write to PCI transaction is allowed to be granted even when there is not sufficient space in the PCI write FIFO for the DMA transaction to complete.

As an example, if the internal write FIFO is full (more than 6 words) and DMA request for a burst of 4 words occur, the DMA will stall waiting for space to become available. Once the FIFO has two or more words of space available, the DMA is allowed to continue. Due to internal pipelining, there is a possibility that the DMA will not be stopped in time, once the FIFO becomes full again, and the FIFO pointer will wrap around and overwrite the first word in the FIFO.

This errata only affects the DMA write operation from memory to PCI that is initiated by the RC32334/332. It does not affect the rest of the standard data transfers using the DMA controller.

Workaround: There are two possible workarounds:

- Set the DMA channel to use only single words transfers. This will prevent the rapid burst of words from the DMA to occur and the FIFO will signal in time to the DMA channel when it is full to stop any new transfer.
- Another solution is to use the DMA_READY_N signal to control the DMA channel operation and prevent this from happening as explained below. This solution only applies when the RC32334/332 is used in satellite mode. It does not apply when it is used in Master mode.

  The DMA flow can be controlled by monitoring the PCI_REQ_N signal. The PCI_REQ_N signal must be fed into an EPLD that counts the number of cycles that PCI_REQ_N is deasserted. If it is

deasserted for more than a certain number of clock cycles, the EPLD then toggles DMA_READY_N low for one clock cycle. The DMA controller on the RC32334/332 will latch this in, and it will then start a 16-byte burst of data. By gating the DMA this way, it is possible to guarantee that there is never more than four words in the PCI write FIFO. This has the added benefit that it keeps the internal bus from stalling with a DMA in progress waiting for the PCI FIFO to empty.

It is important to note that the number of cycles of deassertion can vary depending on targeted PCI device. There is a possibility for this to fail if the target device stretches out the transaction. This happens because the PCI request will deassert once the transaction starts, and it will not reassert until the transaction completes. Obviously, if the target device is slow and withholds TRDY for too long, then the deassertion counter will time out and start another DMA transaction on top of the pending one. In this case, it is recommended to increase the counter value until it is large enough to cope with the longest valid delay.

Baseline EPLD equations in ALTERA HDL syntax:

```
IF (pci_epld_req_n1) THEN
  IF (pci_req_cnt[3..0].q > 10) THEN
       pci_req_cnt[3..0].d = GND;
       dma_ready_n0= GND;
  ELSE
       pci_req_cnt[3..0].d = (pci_req_cnt[3..0].q + 1);
       dma_ready_n0=VCC
  END IF;
ELSE
  pci_req_cnt[3..0].d = GND;
  dma_ready_n0=VCC
END IF;
```

Fix: There are no plans to fix this errata

Item #11 - BTA Control Register

Issue: The BTA Control Register (see Chapter 8), located at virtual address 0xFFFF-E204 within the processor, has no effect on the bus turnaround time. The processor will always have a turnaround time equivalent to a setting of 4 in the BTA register regardless of what value BTA is set to.

Workaround: There is no workaround. The resulting architecture of the chip is such that it is not possible to reduce the back-to-back delays below four cycles. Although the BTA register is technically functional, it is overshadowed by this architectural limitation and thus appears to have no effect on the bus turnaround time.

Fix: There are no plans to fix this item.