RC32364 Device Errata For Silicon Revisions 1.4 - 1.8 and 2.1

Notes

Supplemental Information

This Device Errata reflects revision 1.4, 1.5, and 1.6 silicon and supplements information in the August 9, 1999 data sheet and the *RC32364 RISController Advanced Architecture 32-bit Embedded Microprocessor Reference Manual, Version 1.0*. Silicon revisions can be identified from the following marking on the device.

ZC - data code = revision 1.4

ZD - data code = revision 1.5

ZE - data code = revision 1.6

ZF - data code = revision 1.7

ZG - data code = revision 1.8

YB - date code = revision 2.1

Revision History

July 1998: First version of errata for 1.2 silicon.

September 1998: Second version of errata for Revision 1.2 silicon.

October 1998: First version of errata for Revision 1.3 silicon.

October 27, 1998: Second version of errata for Revision 1.3 silicon.

September 24, 1999: First version of device errata for versions 1.5, 1.6, and 1.7 silicon revisions.

October 29, 1999: Added item #12.

November 16, 1999: Added item #13.

January 20, 2000: Added items #14 and #15.

February 28, 2000: Revised item #3 to reflect errata item in revision 2.1 silicon.

April 24, 2000: Updated items #4 and #5 and added items #16 through #18.

May 31, 2000: Updated Item #18.

June 8, 2000: Added Notes section.

Descriptions and Workarounds

Item #1 - JTAG

Issue: In revision 1.x silicon, clock buffering in the JTAG block is incorrect, which creates a slight overlap between the JTAG 2 phase clock. Thus, the JTAG state machine is advanced at the wrong time.

Workaround: JTAG should not be used with revision 1.x of the device.

Fix: This will be corrected in Revision 2.0 of the device.

Item #2 - Enhanced JTAG (EJTAG)

Issue: The enhanced JTAG facility for performing in-circuit emulations (ICE) is not implemented in version 1.x of silicon.

Workaround: Traditional debugging techniques, such as IDT/sim, can be used.

Fix: This will be corrected in Revision 2.0 of the device.

Item #3 - Internal Buffer Logic Extends back-to-back Write Timing

Issue: In Revision 1.x silicon prior to 1.8 and revision 2.1 silicon, during a sequence of single datum writes to external memory, the device will insert an additional 2 idle bus cycles between consecutive writes.

Workaround: None. This issue will not cause functional failure and should only affect system performance.

Fix: There are no plans to correct this item.

Item #4 - Store to Wrong Cache Set under Non-blocking Loads

Issue: In revision 1.x silicon, if a "store" that gets a "cache hit" follows a non-blocking load that gets a "cache miss", the "store" data could be written into the wrong set of the data cache. This occurs only when the non-blocking load and the "store" instruction have the same index to the cache, but different tags

Workaround: Either operate with non-blocking loads disabled or avoid the operation sequence discussed above.

Fix: There are no plans to correct this item. See also item #16.

Item #5 - Load to Wrong Source Register under Non-blocking Loads

Issue: When non-blocking loads are enabled in revision 1.x, and if

a. a "load" targets R0 as a destination

b. a "pref" instruction is issued

c. This is followed by an instruction that uses R0 as a source register

Then the "load" data is incorrectly bypassed to the second instruction. This only occurs if both instructions are uncached, or if cached, are separated by two or more instructions.

Workaround: Either operate with non-blocking loads disabled or avoid the operation sequence discussed above.

Fix: There are no plans to correct this errata item. See also item 16.

Item #6 - NMI Behavior

Issue: In revision 1.x silicon, if a non-maskable interrupt (NMI*) occurs while the CPU is processing an exception, the NMI detection logic will fail if the pipeline cancelling logic (the logic needed to stop the execution of the instructions in the pipeline and to load the first instruction of the exception handling routine) and the NMI edge detection logic occur in the same Pipeline Clock (internal clock) cycle. The NMI IS NOT recognized again until it is de-asserted and then re-asserted to form the edge.

Workaround: Use the following sequence to assert NMI*

1. Assert NMI* for one system clock cycle
2. Deassert NMI* for one system clock cycle
3. Assert NMI*

The double assertion of the NMI* signal will result in only one NMI cycle. If the CPU recognizes the first NMI* assertion, the deassertion and re-assertion of the signal will be transparent to the CPU. If the 1st assertion of NMI* is not recognized due to the presence of an exception as described above, the 2nd assertion of the signal will be recognized.

Fix: This will be corrected in Revision 2.0 of the device.

Item #7 - "Wired" Register

Issue: In revision 1.x, CP0 register 6, known as the "wired" register, specifies the boundary between the nonreplaceable ("wired") and replaceable (random) entries in the TLB. A value is written to the lower 6 bits to indicate the number of wired TLB entries required. On the RC32364 processor, the device will reserve one less than this number. As an example, if the value of 5 is programmed into this register, only 4 TLB entries will be defined as nonreplaceable. In this case, the fifth TLB entry would be denoted as a random entry and could be overwritten.

Workaround: Ensure that the wired register is programmed with a value one greater than the number of wired TLB entries required.

Fix: This will be corrected in Revision 2.0 of the device.

Item #8 - Incorrect Cache Refill after Store Instruction Causes TLB Miss

Issue: In revision 1.5 and prior revisions of the RC32364, the CPU will erroneously cause the index for the load miss cache refill to change to an incorrect value under the following conditions:

A. A load instruction results in a cache miss, which is processed using "early restart" such that the cache is refilling simultaneous with other operations.

B. A store instruction follows, which generates a TLB miss exception for the store target address.

Specifically, the original cache line which was being refilled will be refilled incorrectly/incompletely. In addition, another line in the data cache will be corrupted, as the cache index will have taken an incorrect value.

Workaround: This errata is only associated with the use of the on-chip TLB. Manage the code to insure that there are no stores during a load refill (e.g. perform an uncached load, targeting register r0, with NBL disabled, before a store instruction, to cause all cache miss processing to complete prior to the store).

Fix: This is fixed in Revision 1.6 and subsequent silicon revisions.

Item #9 - Incorrect Cache Refill

Issue: In revision 1.6 and prior revisions of the RC32364, an incorrect cache refill will occur if a cache line refill coincides with a load that hits in the cache during a stall cycle (that occurred due to the write buffer being full).

Specifically, the cache refilled data is mistakenly written to the cache line of the load hit and to the load target register. The refilled cache line is then loaded with unknown data.

Example:

The following series of instructions will cause the CPU to fail in this mode.

Assume r10 = 80002000

r12 = 80002020

Memory content at 2000 = 11111111 22222222 33333333 44444444

Cache content at 2020 = 55555555 66666666 77777777 88888888

```
lw   r5, (r10)   # this load is a miss which causes a cache line refill

sw   r2, 0x0(r11) # lots of stores that cause the store buffer full stall
sw   r2, 0x4(r11)
sw   r2, 0x8(r11)
```

```
sw   r2, 0xc(r11)
sw   r2, 0x10(r11)

lw   r7, 0x8(r12)     # this load is hit in the cache
```

After this code:

Cache content at 2020 = 55555555 66666666 33333333 88888888
======== <---------- Bad Data

r7 = 33333333 <----- Bad Result

Cache content at 2000 = 11111111 22222222 xxxxxxxx 44444444
======== <---------- Unknown data

Workaround: None.

Fix: This will be fixed in Revision 1.7 and subsequent revisions.

Item #10 - Warm Reset

Issue: In revision 1.x silicon, when an external component issues a warm reset exclusively to the RC32364 after the CPU has completely booted, the CPU fails to exit out of reset.

Workaround: Issue a cold reset sequence to reset the CPU.

Fix: This will be fixed in Revision 2.0 of the device.

Item #11 - Simultaneous Clearing of UM/ERL/EXL Bits in Status Register 12 During Interrupt Service Routine

Issue: In revision 1.x RC32364 silicon, when a program running in kernel mode executing an interrupt routine attempts to simultaneously clear the User Mode (UM), Error Level (ERL) and Exception Level (EXL) bits in status register 12, the UM bit remains set, causing the RC32364 to enter User Mode for one pipeline clock cycle. If the subsequent instruction access kseg0 memory space, an instruction fetch Address Error Exception [Exception code value 4] will occur.

A code example that will cause the failure described is as follows.

```
#define CLI
        mfc0 t0,CP0_STATUS;
        li   t1,ST0_CU0|0x1f;
        or   t0,t1;
        xori t0,0x1f;
        mtc0 t0,CP0_STATUS
```

The transition of bits UM, ERL, EXL when executing this code is shown below:

UM 1 -> 1 -> 0

ERL	0	->	0	-> 0
EXL	1	->	0	-> 0

Mode State Kernel User Kernel

|

|__ Faulty User Mode for 1 Pipeline clock cycle

Workaround: Ensure that the UM bit is cleared first, before clearing the ERL/EXL bits several instructions later. The following code example provides a workaround to the problem described above.

```
#define CLI
        mfc0 t0,CP0_STATUS;
        li      t1,0x10;
        or      t0,t1;
        xor     t0,t1;
        mtc0 t0,CP0_STATUS;
        li   t1,ST0_CU0|0x1f;
        or   t0,t1;
        xori t0,0x1f;
        mtc0 t0,CP0_STATUS;
```

Fix: This will be fixed in revision 2.0 of the silicon.

Item #12 - BusErr* Recognition in x3 to x8 Clock Multiplier Mode

Issue: In revision 1.x RC32364 silicon, when the CPU is programmed to use the x3, x4, x5, x6, x7, x8 clock multiplier, it doesn't recognize the assertion of the BusErr* signal.

Workaround: None.

Fix: This will be fixed in revision 1.8 and 2.1 of the device.

Item #13 - Cache Line Corruption Following Cache Miss

Issue: In silicon revisions up to and including 1.7, when the RC32364 has its on-chip data cache configured in write-back or write-through with write allocate and a store operation misses the cache, the device will bring in a line from main memory in order to update it. However, this update is not correctly performed. The CPU correctly performs the update to the critical word, but the subsequent word in the cache line becomes corrupted, being changed to an undefined value.

Workaround: Operate the CPU in write through, non write allocate mode.

Fix: This errata is fixed in revision 1.8 and revision 2.1 of the device.

Item #14 - Occurrence of Bus Error (BusErr*) during a Partial Datum Transfer

Issue: In Revision 1.x and 2.0 of the device, if a bus error (BusErr*) is asserted during a partial datum transfer (i.e the requested data is bigger than the port width), the CPU tries to complete the data transfer prior to handling the Bus Error through an exception. Consequently, the CPU will wait for additional acknowledge receiving data (Ack*) signals from the external agent.

Workaround: Ensure external logic asserts the required number of ACK* signals back to the CPU.

Fix: This errata is fixed in revision 2.1 of the device.

Item #15 - Bus Error (BusErr*) during CPU Write Cycle

Issue: In Revision 1.8 of the device, if a Bus Error (BusErr*) is asserted during a write cycle (for example, if the software attempts to write to a non existent memory device), the CPU will cause the system to hang.

Workaround: None

Fix: This errata is fixed in revision 2.1 of the device.

Item #16 - Non-blocking Loads

Issue: In all revisions of RC32364, problems have been identified with the non-blocking loads feature.

Workaround: The non-blocking loads should be disabled by ensuring the NBL bit in status register (CP0 register 12, bit 27) is set to 0.

Fix: There are no plans to correct this errata item.

Item #17 - 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) when the debugboot pin is asserted. However, in revisions 2.0 and 2.1 of the RC32364, the CPU fetches from the normal ROM address (BFC0-0000).

Workaround:

Fix: There are no plans to fix this errata item.

Item #18 - Boot of CPU into Debug Mode

Issue: The debugboot signal (pin 10) is used during reset. If the signal is high, it should force the RC32364 to take a debug exception at the end of the reset sequence instead of a reset exception. This enables the CPU to boot from the ICE probe without having external memory connected (or operational). In revisions 2.0 and 2.1, when in debugboot mode the CPU incorrectly fetches from normal ROM address (BFC0-0000) rather than debug ROM address.

Workaround: On revision 2.0 and 2.1 of the RC32364, 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 errata item.

Notes

1. The RC32364 data sheet has been modified to remove 5V tolerance on all input pins as a feature.