Notes

This document is intended to reflect some of the design considerations that need to be applied when designing a system based on the RC32365 device. This document is intended to record subtle behaviors of the RC32365 that should be considered early in the design process to avoid lengthy debug time.

Document Revision History

June 23, 2003: First version of the document.

October 20, 2003: Second paragraph in the Software Recommendations section, changed the following wording: "UARTs for the 16550 are enabled by" to "FIFOs for the 16550 are enabled by." In the second paragraph of the CPU Interrupts section, the reference to TDM Input and DMA11S is replaced by Ethernet 0 Input and DMA0.

November 17, 2003: Added UART Mode section.

March 10, 2004: Added new section, Manually Setting DMA Status Bits...Consequences, to Programming Precautions section.

Hardware Recommendations

Board Reset from EJTAG Probe

The EJTAG specification requires the RST* pin on the EJTAG header to reset the board. Many ICE probes use the RST* pin (which is an open drain output) to sense whether power is applied to the probe. When the EJTAG probe initiates a board reset by bringing RST* low, the signal is driven low solidly.

However, at the end of the reset procedure, the EJTAG probe cannot drive the RST* signal back high because this pin is an open drain output. The EJTAG probe must rely on a pullup resistor supplied by the board.

Even with a resistor as small as 1000 ohms, it takes a full 100ns to pull this signal high even when it is only driving one load. If that single load is an EPLD, the signal tends to oscillate a bit as it crosses the Vih threshold for the EPLD. This causes the EPLD to register the deassertion and reassertion of the RST* signal multiple times. Because all of the IDT parts require continuous reset pulses in the order of 100ms or greater, this oscillation causes the part to fail reset in an unpredictable way.

Two methods for avoiding this problem are listed below.

1. The reset signal from the ICE probe can be routed through the power/reset management IC. Generally, these ICs contain an input which initiates a reset of the circuit. The pulse width of this reset depends on the resistor/capacitor combination attached to the IC.
2. If the signal goes into an EPLD, the assertion of RST* can be used to initiate a cold reset of the processor and board circuit. But when RST* deasserts, a counter with a count length of greater than two microseconds should be used before deasserting the cold reset to the processor and board.

Both the 79EB365 and 79EB336 evaluation boards use the first method.

Software Recommendations

Operation of UART in Polled Transmit Mode

The RC32365 provides 1 UART which is designed to be compatible with both the 16450 and the 16550. The 16550 is identical to the 16450 except that the 16550 provides a 16 byte FIFO on both receive and transmit sides.

FIFOs for the 16550 are enabled by setting bit 0 of the Buffer Control Register, BCR[0].

There are two modes in which users can program the UARTs:

- Interrupt driven mode, when it is supported by the UART controller and the board design
- Polled mode.

The RC32365 UART behaves in compliance with the 16550 specifications in the interrupt driven mode.

FIFOs were introduced in the 16550 to enable the 16 bytes of FIFOs to be filled (transmit) or emptied (receive) in a single execution of the interrupt handler, thereby reducing the load on the CPU. However, these FIFOs can be used in polled mode, although the benefit is less compared to the interrupt driven mode.

Typically, in the polled transmit mode, the Line Status Register (LSR) checks to see if it is appropriate for the software to write the next byte to the UART or the FIFO. The 16550 specifications state that LSR[5] can be guaranteed to be 1 when the Transmit Holding Register (THR) is empty, and LSR[5] can be guaranteed to be a 0 when THR is not empty. A non-empty THR implies at least one byte in the FIFO buffer. Therefore, writing a single byte to the transmit FIFO ought to result in LSR[5] returning a 0. This does not happen on the RC32365.

IDT recommends two possible procedures for ensuring the current operation of the RC32365 device when it is used in polled transmit mode:

- Test LSR[6] bit instead of LSR[5] bit. LSR[6] bit, when set to 1 by the controller, indicates to the user that the transmit buffer as well as the THR are empty. This will allow the user to transmit one byte at a time in the polled mode.
- Set the DMA mode in the Buffer Control Register, BCR[3], to 1. This will activate the TXRDY interrupt signal when the transmit FIFO buffer is completely empty and will deactivate the TXRDY signal when the buffer is completely full. The state of the TXRDY signal can be probed by software through the Expansion Interrupt Controller. Register Group 5 deals with UART channel 0 and Group 6 deals with UART channel 1. TXRDY Interrupt State can be read through the “Interrupt Pending Register” corresponding to the UART channel under consideration. Once a completely empty buffer condition is sensed by polling the Interrupt Pending Register, up to 16 bytes can be written to the transmit FIFO in a single attempt without worrying about the level of fullness of the buffer, etc.

CPU Interrupts

When the RISCore 32300 CPU core processes an interrupt, there are two code components: a generic interrupt handler function, and a specific ISR (Interrupt Service Routine). The generic handler must do the following:

1. Save the current state of the CPU (all registers)
2. Determine the pending interrupt with the highest priority
3. Call the appropriate ISR (somewhere in here, the interrupt source should be cleared)
4. [Optional] return to step 2 if other interrupts are still pending
5. Restore the state of the CPU (all registers)
6. Execute ERET instruction to return to the original PC location.

In the ISR, the code should clear the source of the interrupt. For example, when a DMA interrupt for Ethernet 0 Input is being processed, the ISR should clear the DMA0 register bits before exiting the ISR. If this is not done, the same interrupt will be given as soon as processing is done (i.e. go from step 6 directly back to step 1). A problem occurs because even though the source of the interrupt in the ISR is cleared the system still gives an additional interrupt after completing step 6. When handling this interrupt (in step 2), no interrupts pending can be found. As a result, an error is displayed on exit.

The system gives an additional interrupt because the register write to clear the source of the interrupt gets delayed going from the YY Bus to the IP Bus. The delay occurs because an external master has been granted the bus, so the IP Bus cannot accept the transaction from the BIU until the external arbiter has completed. Although the delay here occurred due to the external arbiter, it could just as easily have occurred if the DMA had been granted the bus. In either case, during this delay the CPU continues onto steps 5 and 6. The instructions and the data for these steps are already in the cache. As a result, the CPU is able to complete these steps very quickly with no need to perform a transaction over the YY Bus (Remember, the YY Bus is tied up now with the pending register write transaction that cannot complete until the external arbiter has released the bus). Immediately after completing step 6, the CPU branches to the interrupt vector and begins step 1. By the time it gets to step 2, the write transaction has completed and the interrupt source has been cleared. Since no pending interrupts are active, an error is displayed.

To avoid this problem, make sure the transaction in the ISR that clears the interrupt source completes before exiting the ISR. Since the RC32365 can only queue up one external write at a time, a system designer could simply perform the write twice in a row before exiting the ISR. If the first one is delayed (due to the external arbiter or DMA being granted the bus), the CPU will have to stall on the second write.

Ethernet Controller software reset

The two RC32365 ethernet controllers can be reset manually by clearing the EN bit in the corresponding ETHxINTFC register. When the EN bit is cleared, an Ethernet interface reset is generated and the RIP bit is set to indicate that an Ethernet reset is in progress. The reset may take several clock cycles to complete due to the crossing of multiple clock domains. When the reset has completed, the RIP bit is cleared and the Ethernet interface may be re-enabled by setting the EN bit.

However, if the MII clocks disappear or are not present during the Ethernet interface reset, the RIP bit will remain set, preventing the Ethernet interface from being reset until an MII clock is provided. The only other way to clear the RIP bit is to generate a cold_reset. The Ethernet PHY normally drives the MII clocks. Consequently, the behavior described above can be avoided by ensuring that any software which resets the RC32365 Ethernet controller does not reset the Ethernet PHY at the same time.

UART Mode

Do not change between UART 16450 and 16550 modes (bit [0] of the FIFO control register) and then flush the FIFOs while there are characters still in the FIFO TX buffer waiting to be sent. This may result in a UART transmit status malfunction. If this happens, the TE and THR bits in the UARTxLS register may be permanently cleared until the interface is reset. Characters can otherwise still be transmitted and the other status signals will continue to function.

Programming Cautions

Invalid Security Engine Command Descriptors

If a command descriptor is sent to the security engine with the wrong number of bytes causing an error, the security engine will ignore the command and issue an output descriptor with the error bit set. However, if this is then immediately followed with a valid command, the security engine may lock up. This condition can only occur in the case of an improperly written security engine driver. In the event the security engine stops responding during driver development and debug, check the last output descriptor returned to determine if the error bit was set. Correcting the security engine driver will fix this problem.

ST_CNTXT or ST_IV Security Engine Commands with Invalid Contexts

If a Store Context (ST_CNTXT) or Store Initialization Vector (ST_IV) command with an invalid context identifier is sent to the encryption unit, the encryption unit will output a zero length status descriptor with the Context Invalid (CE) bit set, indicating that the context is invalid. It will also output a second descriptor as well. The second descriptor has the CE bit set, but it also contains 24 words of data for the ST_CNTXT case and 4 words for the ST_IV case. In the event an unexpected duplicated output descriptor is found, check for a valid context in the offending command which generated the duplicate. Invalid contexts are the result of a programming error in the security engine driver and will not be encountered in a production worthy system. Correcting the security engine driver will fix this problem.

Some Hash Misconfigurations Will Not be Flagged

If the HOFFSET is greater than the data stream length, the Hash Misconfiguration Error (HME) bit will not be set in the descriptor DEVCS field. If Hash Checking (HC) is set in the associated security context, the Hash Mismatch (HM) bit will be set. If the HOFFSET + HLENGTH are greater than the data stream length AND the HLENGTH = 1, 2, or 3 bytes, the HME bit will not be set in the DEVCS field. If HC is enabled, the HM bit will be set. The security engine will lock up under the following conditions:

- If HOFFSET + HLENGTH are greater than the data stream length AND
- Data stream length - HOFFSET = 3, 2, or 1 byte AND
- Data stream length is an integral number of 32-bit words (DSL % 4 bytes = 0).

This will never happen in a properly written driver. Correcting the security engine driver will fix this problem.

Manually Setting DMA Status Bits May Have Undesired Consequences

If the done bit "D" is set in a descriptor which is fetched from memory as part of a DMA transaction, both the CPU and DMA may attempt to drive the internal bus at the same time, thereby corrupting the address or data for that transaction. This can have highly unpredictable consequences depending on what the data or address impacted happens to be associated with. The DMA status bits ("D", "F", "T") must be cleared before a descriptor is used. Once the DMA sets the bits, the descriptor cannot be reused until the software handler reconditions it by clearing the status bits, appropriately setting the byte count, etc.