79RC32334/RC32332 Interrupt Modus Operandi

Notes

By Harold Gomard

Revision History

November 5, 2001: Initial publication.

Interrupt Scheme

The Expansion Interrupt Controller extends the RC32300™ CPU Core CP0 interrupt control by collating the RC32334 generated interrupts into a single CPU interrupt - cpu_int_n[3]. In addition, interrupts can be independently masked by the Expansion Interrupt Mask Register.

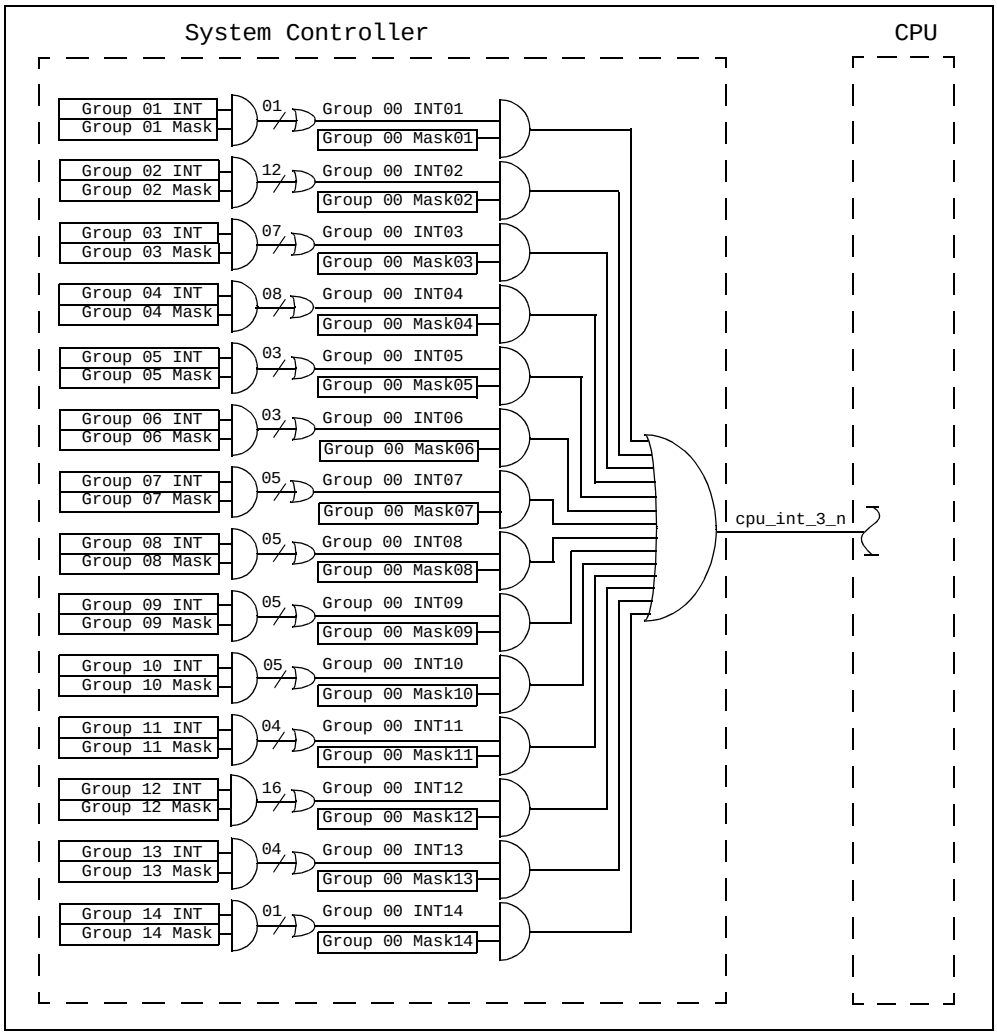

Figure 1

Notes

As shown in Figure 1, the System Controller, Group 1 interrupts are ANDed with their masks and then ORed into one signal. This ORed signal then goes to bit 1 of the Group 0 pending register. Group 2 interrupts are ANDed with their respective masks, and then ORed into one signal which goes to bit 2 of the Group 0 pending register, etc. There are fourteen of these groups--Group 1 to Group 14. Each group gets ORed down to a single bit in the Group 0 register.

The 14 bits of the group 0 pending register are then ANDed with their respective masks and then ORed down to a single signal which internally feeds the CPU INT3 signal. This is why there is no INT3 signal external to the CPU.

Internal to the CPU, the CPU timer interrupt, which is masked by one of the boot vectors, is ORed with INT5 coming from outside the chip. If the user wishes to use both the timer and interrupt 5 simultaneously, then some external means must be provided to allow the user to determine if an external interrupt is present (i.e., a readable register of some type). By reading this register, the user can determine if the interrupt is coming in from outside the chip. If this register does not show an interrupt pending, then the interrupt was caused by the timer. Generally, most users will either use the timer function or INT5, but not both.

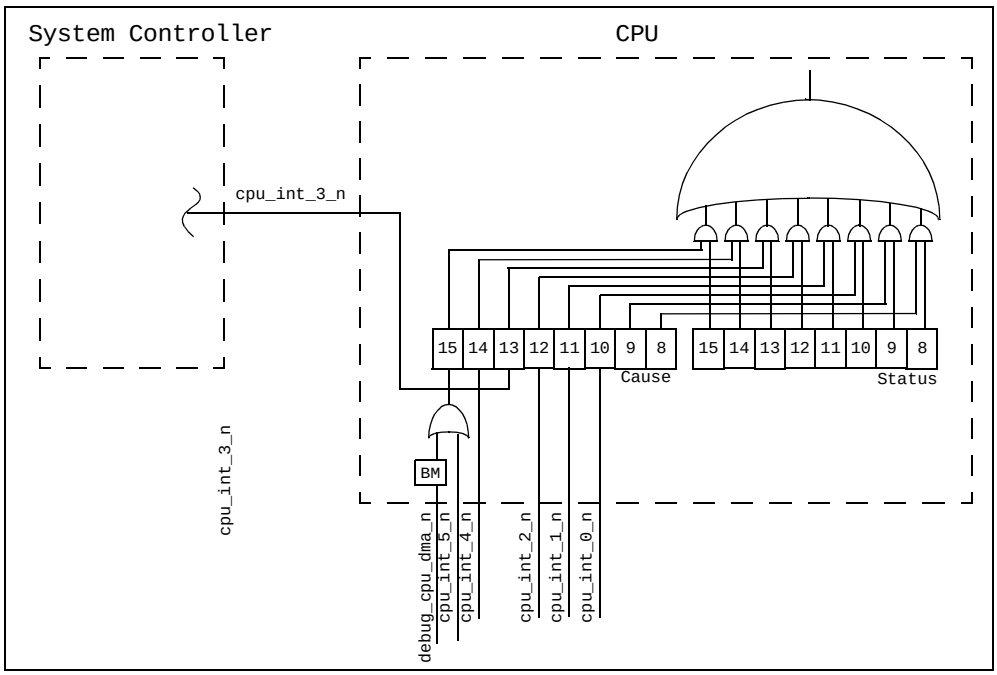

Figure 2

As shown in Figure 2, inside the CPU, INT5 goes to the cause register, bit 15. INT5 is ORed with (Timer & Bootmask) -> Cause[15], INT4 -> Cause[14], internal3 (internal from system controller) -> Cause[13], INT2 -> Cause[12], INT1 -> Cause[11], INT0 -> Cause[10]. Cause[9] & Cause[8] are software interrupts. Cause[15:8] and the Status Register (Mask) [15:8] are ANDed together, and then ORed to a final signal which goes to the CPU and causes it to take exceptions.

RC32334/RC32332 Interrupt Latency

When an instruction cache is running and a CPU pin interrupt occurs, two different scenarios must be considered in order to estimate the interrupt latency properly.

1. Interrupt handler (0x80000180) is already resident in a valid instruction cache location.

In this case, the interrupt handler is fetched 3.0 clocks after the interrupt occurs and takes 1.0 system clock (in divide by 2 mode) to reach the ALU stage where it is "executed/run". Therefore, the interrupt latency is 4.0 clocks.

Notes

2. Interrupt handler (0x8000180) misses the cache and must be fetched.

In this case, the interrupt handler is fetched (debug_ads_n is clocked occurs) 8.0 clocks after the interrupt occurs. This accounts for the time for the memory access (4 wordburst reads must be added). Then, 2.0 extra clocks are required until the CPU pipeline is restarted.

Finally, 1.0 extra system clock is needed to reach the ALU stage where it is "executed/run". Thus, if a page-miss burst read occurs from SDRAM (11.0 extra clocks in addition to the usual 8.0 clocks described above), the interrupt latency accounts for a total of 22.0 clocks.