ISBN: 978-981-18-1791-5
2021 the 11th International Workshop on Computer Science and Engineering (WCSE 2021)
doi: 10.18178/wcse.2021.06.025

A Novel Testing Method for Interrupt Response Time

Zhen Liu [1], Yirong Shi [2], Guanhua Zhang [2], Bo Hu [1], Fei Ye [2], Hua Zhou [2+]

[1] School of Information Science and Technology , Fudan University, China

[2] Academy of Engineering and Technology , Fudan University, China

Abstract. Interrupt response time is an important index to measure the real-time performance of real-time operating system. Accurate test data can be the evaluation index for selecting real-time operating system. This paper proposes a test method for interrupt response time based on W2 chip. It can test the statistical distribution of interrupt response time quickly and accurately. The results show that the Linux real-time preemptttion patch has better real-time performance than Linux and the two operating systems have better real-time performance under no load. The results also prove that the test method is effective. The test method provides a basis for the development of real-time tasks.

Keywords: interrupt response time,test method, real time performance,Linux.

1. Introduction

Real-time operating systems (RTOSs) [1] such as RTLinux and VxWorks [2] are widely used in industry and aerospace because of their high real-time, reliability and security. One of the most significant features of real-time operating system is real-time. Real-time means that before the external events come, it can response the interrupt in a short time, and then send it to CPU for processing. Interrupt response time [3] is the most important index to evaluate the real-time performance of real-time operating system. Current research on interrupt response time is not deep both at home and abroad, and there is no unified test system or method.

Generally, there are two test methods for interrupt response time. One of the methods is to use an oscilloscope to test interrupt response time. This method is more intuitive and the results are more accurate. However, it is not convenient to repeat the test. Another one is to use software to test the interrupt response time, but the operation of the program itself will affect the timing accuracy. Therefore, it is necessary to design a convenient and accurate interrupt response time test method of RTOS.

2. Interrupt response mechanism

Interrupt response time is related to hardware delay, operating system software scheduling and waiting time for the first instruction of interrupt service routine (ISR) to run. Therefore, it is necessary to analyze its hardware architecture and software architecture.

Taking X86 as an example, the hardware architecture of interrupt response is mainly composed of interrupt controller.The development of interrupt controller mainly includes programmable interrupt controller (PIC) and advanced programmable interrupt controller (APIC) [4]. PIC is an old-fashioned interrupt controller, which is suitable for single CPU. PIC is composed of two 8259A chips. The main chip controls the interrupt signals of irq0 ~ irq7, and the slave chip controls the interrupt signals of irq8 ~ irq15 [5].

Now APIC is more popular than PIC, because APIC is designed for multiprocessor control. In order to maintain compatibility, most PCs still retain PIC. APIC is composed of I/O APIC and Local APIC. I/O APIC was located in south bridge, but it has been integrated into CPU since Pentium p54c. It is responsible for distributing interrupts and sending them to Local APIC. Local APIC is inside CPU. After receiving interrupts, Local APIC assigns the priority of interrupts and passes them to CPU for processing [6].

+ Corresponding author. Tel.: +86 13162217111..
E-mail address: zhouhua@fudan.edu.cn.

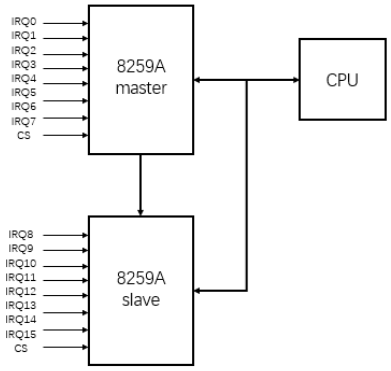


Fig. 1: Architecture of PIC.

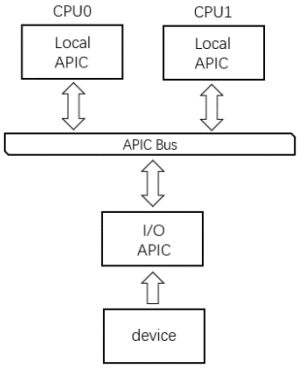


Fig. 2: Architecture of APIC.

The interrupt software architecture of Linux has two layers. The first layer is generic handler, which is responsible for the IRQ line. The second layer is specific handler, which is responsible for all of the devices mounted on the IRQ line. Generic handler is added by irq_set_handler(irq, handle) during initialization. IRQ and handle means IRQ number and first layer processing function. Specific handler is added by request_irq(), which can register interrupt processing function to Linux kernel. The irq_action of the registered device will be added to the end of the IRQ list. The corresponding logical interrupt number IRQ and interrupt name is in the directory of /proc/interrupts [7].

After the interrupt passes through APIC and reaches CPU, Liunx will find the virtual interrupt number IRQ according to the physical interrupt number HWIRQ, which is provided by irq_find_mapping(). If the irq number has been found, Generic_handle_irq() can be called back by do_IRQ(). If the interrupt processing of the first layer is completed, it can traverse the linked list on the IRQ line and execute the irq_action one by one [8].

3. Method of Interrupt Response Time Testing

Interrupt response time refers to the time from the generation of external interrupt to the execution of the first instruction of ISR. It can also be divided into interrupt delay time and system processing time. Interrupt delay time is the time to trigger interrupt and stop current instruction. System processing time is the time to turn off interrupt, stack registers and start executing the first instruction of ISR [9].

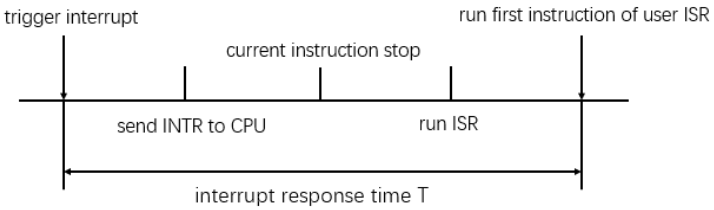


Fig. 3: Interrupt response time.

3.1. Principle of Test Method

In response to these problems, this paper proposes an automatic and effective test method. The method has two parts, which are Linux and test card.

(a)Test card uses GPIO to send a rising edge to X86 hardware and start the timer at the same time;

(b)Linux inserts the test code into the driver function, and sends a falling edge to W2 chip after receiving the interrupt.

(c)After receiving the falling edge signal, the test card immediately turns off the timer and calculates the test results;

(d)Test card sends test results to PC and displays the distribution of interrupt response time on GUI in real time.

3.2. Design and Implementation

The chip used in the test card is W2, which is a 32-bit independent processor. W2 has 128K byte flash

space, and supports 4MHz, 8MHz, 16mhz and 32mhz system clock frequency. It also has three UART modules, five independent timers and 40 GPIO pins.

The hardware of the test method is mainly divided into W2 end and X86. IO2 of W2 sends the rising edge interrupt signal to IO499 of X86. After receiving the interrupt signal, X86 enters the interrupt processing function. The interrupt processing function sends the falling edge signal to IO3 of W2. IO3 stops timing after receiving the confirmation signal and outputs the count value of timer.

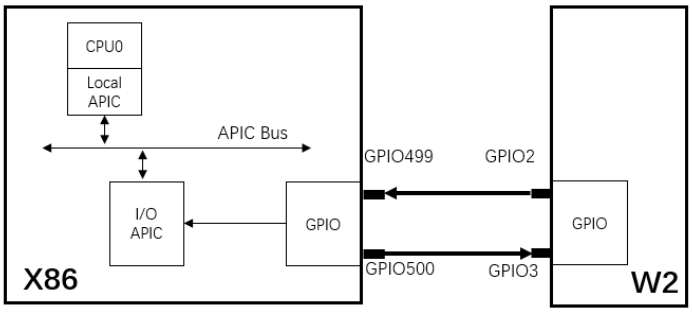


Fig. 4: Hardware Architecture of Test Method.

Linux driver registers interrupt handling function in kernel firstly, and then generates falling edge by interrupt handling function.

The code of the driver is as follows:

```
//GPIO application
gpio_ request(gpioNo499, "mygpiopin499");
gpio_ request(gpioNo500, "mygpiopin500");
//GPIO settings
gpio_ direction_ input(gpioNo499);
gpio_ direction_ output(gpioNo500, 1);
//Application interrupt number
gpio_ to_ irq(gpioNo499);
//Register interrupt handler
request_ irq(irqno, myirq_ handler, IRQF_ TRIGGER_ RISING, "mygpiopin", NULL);
```

The code of W2 is as follows:

```
//Set the clock frequency to 32mhz
MemoryOr32(0x1f800701,0x0200); MemoryOr32(0x1f800702,0x2000);
MemoryWrite32(0x1f800704,0x4);
MemoryWrite32(0x1f800705,0x0);
//Set timer
MemoryAnd32(T0_ CTL0_ REG, ~(1<<7));
MemoryWrite32(T0_ CLK_ REG, 1);
MemoryWrite32(T0_ REF_ REG, 0);
MemoryOr32(T0_ CTL0_ REG, (0x02 | (0 << 7)));
MemoryOr32(SYS_ CTL0_ REG, 1);
```

4. Interrupt Response Time Test

The test method has tested and analyzed the interrupt response time. The hardware platform and Linux version used in the test are as follows: Advantech MIO-5850 embedded motherboard, atom e3845 CPU, 1.91 GHz frequency (725mhz CPU for interrupt), and 3.75 GB memory; The Linux version is Ubuntu 18.04.1, the Linux kernel is official Linux 4.13.13, and Linux real-time preemption patch is patch-4.13.13-rt5.

The interrupt response time has been tested for 5000 times. The interrupt response time of Linux and Linux real-time preempt patch have been tested under full load and no load respectively.

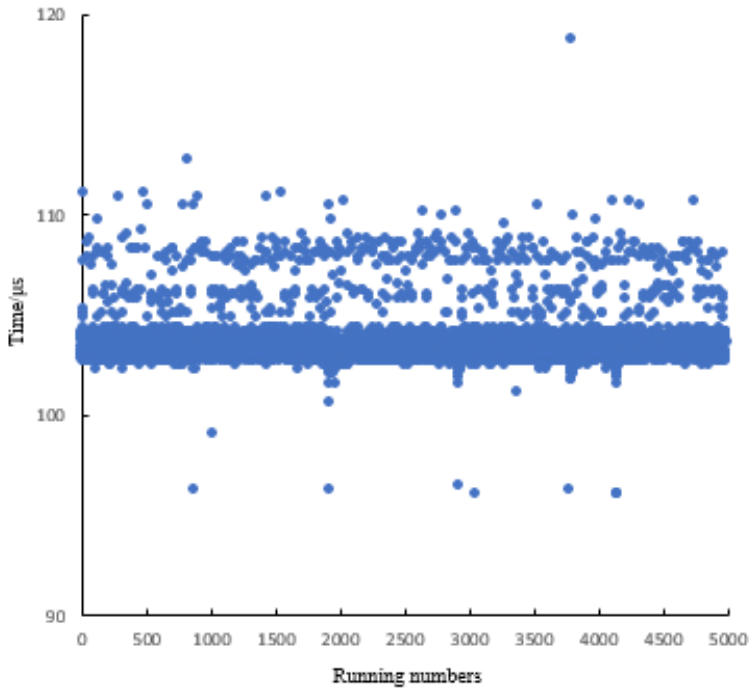


Fig. 5: Interrupt response time of Linux kernel under no laod.

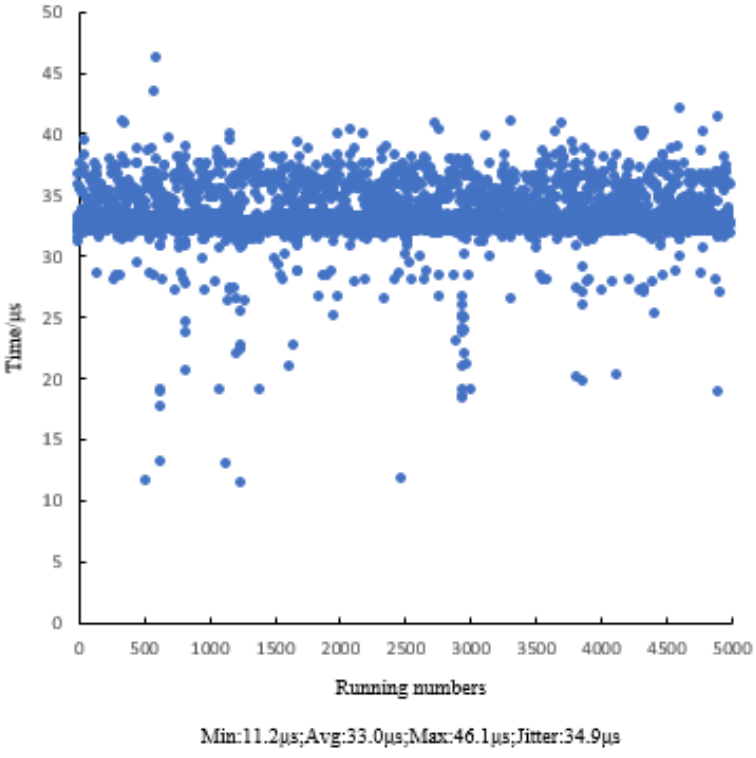


Fig. 6: Interrupt response time of rt kernel under no laod.

Under no load condition, the interrupt response time of Linux kernel is about 102.8 μs -110.7 μs, and the maximum is 118.7 μs. The interrupt response time of Linux real-time preemption patch is 26.9 μs-40.9 μs, and the maximum is 46.1 μs. Comparing the experimental results, it is clear that Linux real-time preemption patch has better real-time performance than standard kernel.

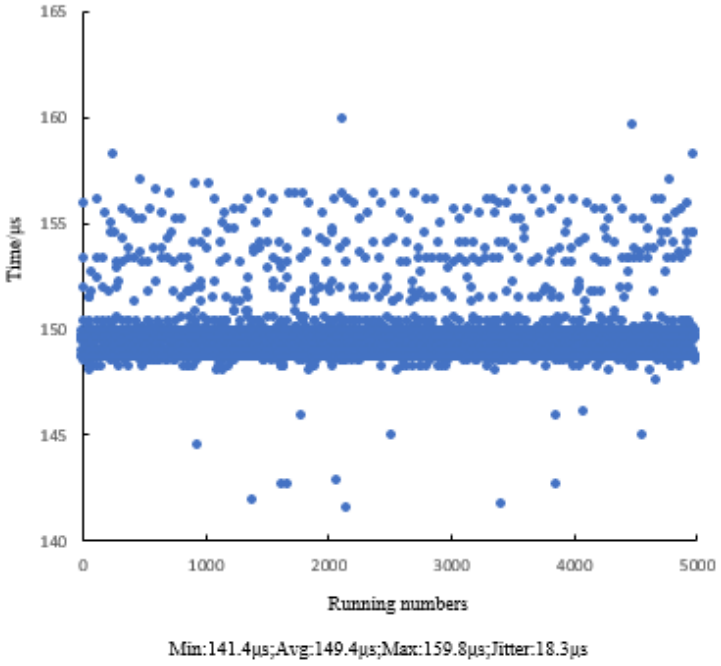
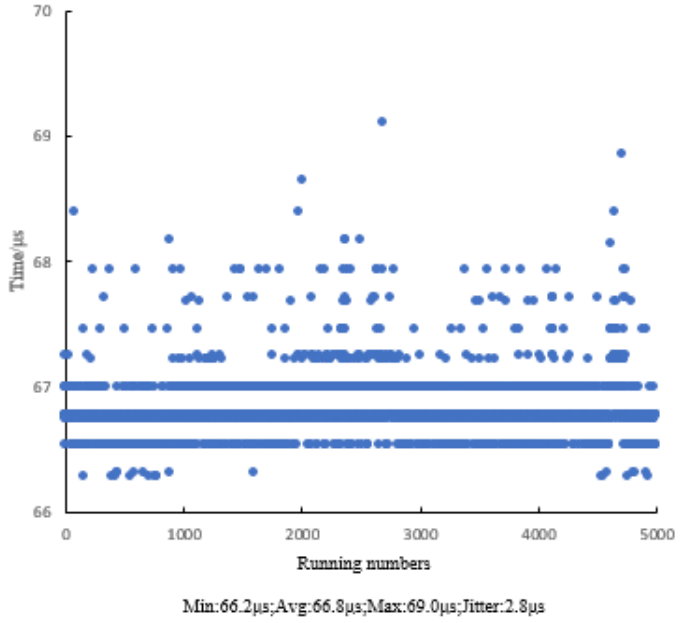


Fig. 7: Interrupt response time of Linux kernel under full laod



.Fig. 8: Interrupt response time of rt kernel under full laod.

Under full load condition, the interrupt response time of Linux kernel is 148.1μs -150.5 μs, and the maximum is 159.8 μs. The interrupt response time of Linux real-time preemption patch is 66.4 μs - 67.2 μs, and the maximum is 69.0 μs. The comparative analysis shows that Linux real-time preemption patch still has better real-time performance in the case of full load, and the jitter of Linux interrupt response time in the condition of full load is smaller than no load.

5. Conclusion

This paper proposes an automated test method, which can test the interrupt response time of real-time operating systems quickly and accurately. In order to prove the effectiveness of this test method, this paper

also tests the interrupt response time of Linux and Linux real-time preemption patch under the condition of no load and full load. The test method and the test results provide the basis for the design of the real-time tasks. The next work is to make the test method more integrated and add more functions.

6. References

[1] Stankovic J A, Rajkumar R. Real-time operating systems[J]. Real-Time Systems, 2004, 28(2-3): 237-253.

[2] Zhangjin W, Mc Guire N. Porting RT-preempt to Loongson2F[C]//Eleventh Real-Time Linux Workshop. 169.

[3] Lackorzynski A, Weinhold C, Härtig H. Predictable low-latency interrupt response with general-purpose systems[C]//Proceedings of OSPERT2017, the 13th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications OSPERT 2017. 2017: 19-24.

[4] Sia C Y, Rosdi B A, Lee M C. Synchronous design of 8259 programmable interrupt controller[C]//2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE). IEEE, 2011: 195-200.

[5] Buchanan W. Hardware Interrupts[M]//Mastering Pascal and Delphi Programming. Palgrave, London, 1998: 197-212.

[6] Cruz T, Simoes P, Monteiro E. Virtualizing programmable logic controllers: Toward a convergent approach[J]. IEEE Embedded Systems Letters, 2016, 8(4): 69-72.

[7] Wang Q. An interrupt management scheme based on application in embedded system[C]//2008 International Conference on MultiMedia and Information Technology. IEEE, 2008: 449-452.

[8] She W, Yang F, Wang S, et al. The Research of an On-demand Threaded Interrupt Technology of Serial Port Based on the Power Edge Smart Device[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1828(1): 012091.

[9] Ganssle J. Interrupt Latency[M]//The Firmware Handbook. Newnes, 2004: 245-249.