AN ADJUSTABLE MEASUREMENT SYSTEM FOR THE REAL TIME CLOCK OF HIGH-END SERVER SYSTEMS

Ying-Wen Bai and Te-Hsin Chen
Department of Electronic Engineering, Fu Jen Catholic University
Taipei, Taiwan, 242, R.O.C.
bai@ee.fju.edu.tw

ABSTRACT
Due to the importance of the accurate measurement of the real time clock (RTC) of high-end server systems such as multimedia server and video server systems during production, we propose the design and implementation of an automatic measurement system which is adjustable to measure the error of the real time clock. Our design uses a higher clock rate, like 10MHz, which provides a clock reference with a high resolution in comparison with the low clock rate of the real time clock of 32.768 KHz of the mainboard of a high-end server system. With the integration of the programming in a microprocessor, the modular CPLD design, the I/O and the interface circuits, we provide an inexpensive, accurate and easy to operate high-performance system for the RTC measurement in the high-end server systems on production lines. The measurement speed and the RTC accuracy are balanced to obtain high measurement efficiency for computer manufacturers. With this design the measurement time is adjusted and limited to within 20 seconds and RTC accuracy obtained to within ±0.1ppm, which betters our previous design whose accuracy stood at between ±11ppm and ±25ppm.

KEY WORDS
Real time clock, accuracy, CPLD, ppm.

1. Introduction
The accuracy of the real time clock (RTC) of the high-end server system is one of the important factors for most IT companies. If the RTC is not accurate, the system clock cannot display an accurate time for the MIS administrator, especially for those network applications which require more accurate time for the synchronous message exchanges. The administrator will need to adjust the server timers very often. Therefore the high-end server system manufacturers need to be able to measure the accuracy of an RTC. We can then provide adjustment methods for the measurement system board so that the RTC will be accurate. Usually the accuracy of the RTC is affected by different batches of components, the hardware circuits, the temperature, and the power voltages of a mainboard in a server. In addition to measuring accuracy, by using our design, the measurement time can be within a couple of seconds of the now shortened manufacturing time. Finally, from the manufacturers’ point of view, a trained operator can easily operate our measurement system [1-4].

To provide the accurate measurement of the RTC in a high-end server system, such as a multimedia server system and a video server system, our design must generate an accurate one-second reference signal for one second and send the reference signal to the mainboard of a server. By recording the difference between the one-second reference signal and the RTC on a mainboard we can determine the difference or the error rate by the number of CPU clocks which can be recorded in the time counter or the 64-bit register in the CPU of the mainboard of a server [5, 6].

This paper is organized as follows. In Section 2 the system design overview for the RTC measurement system is presented. In Section 3 the design and implementation of the RTC measurement system are discussed. In Section 4 the experimental results and the frequency accuracy determination of the RTC system are described. In the final Section our conclusions are presented.

2. System Design Overview for the RTC Measurement System
A typical RTC circuit on a mainboard of a high-end server system is shown in Fig. 1. RTC accuracy can be affected by the following factors: (1) RTC voltages, (2) external capacitance load C1 and C2, (3) RTC circuit layout considerations and (4) environmental conditions. RTC accuracy can also be affected by the voltage of the server system’s battery. In general, when the battery voltage decreases, the RTC accuracy also decreases [7, 8].
There are several methods used to measure RTC accuracy. The first is manpower synchronization. However, since we must know what time it is first, the time indicated by the system or watches may not be accurate enough. The second method is checking the time with the reporting center to obtain the accurate time. Setting the time and checking the error of the system time may take many days or weeks. If we measure the time by visual and manual methods, we must check the time error at least one week later because an error of one ppm is only 0.0864 sec per day. Yet a timing error can cause an inaccuracy that is not what we might expect. Increasing the measuring time is the only way to reduce the timing error, but it is not an efficient method for the production line [9-12].

1 ppm per day and 1 second error per day are shown in Eq. (1).

1 day = 24 hours * 60 min / hour * 60 sec / min = 86,400 sec
1 ppm (1 day) = (1* 86,400) / 1,000,000 = 0.0864 sec
1 second = 1 ppm / 0.0864 = 11.57407 ppm                (1)

The standard value of the error for the current level of accuracy is under ± 1 sec per day (± 11.57ppm). If the timing error set by the tester is ± 0.3 sec, it means that there is almost a 30% measurement error in one day. This traditional method is quite unsuitable for the production line.

The third measurement method is counting the time error of the RTC via an accurate time scale and an internal CPU clock. First we provide a very accurate outside one-second signal. Then the system GHz (1x10^9 Hz) CPU clock counts the difference between the outside accurate one-second reference signal and one second from the RTC of the server system itself and immediately displays the error by ppm of the mainboard. Thus from the Unit Under Test (UUT) we can derive the error of the RTC of the server system per day via mathematical calculation, as well as the error of the UUT in seconds per day. By using this method on the production line we can speedily see whether the error value goes beyond the standard set for the RTC of the mainboard or the UUT.

3. Design and Implementation of the Measurement System

There are several modules in the design as shown in Figure 2: (1) the oven-controlled crystal oscillator (OCXO), (2) the generation of the accurate one-second reference signal from the CPLD block, (3) the digital temperature sensor, (4) the 8051 microcomputer for monitoring the temperature and compensating for the frequency deviation of the clock, (5) the LCD module for displaying the current frequency and temperature and (6) the RS232 for the signal transmission to the mainboard or unit under test (UUT).

The major modules of the design are briefly explained as follows.

(1) The oven-controlled crystal oscillator (OCXO)
We know that the outside temperature affects the crystal easily. In general the frequency stability of the common crystal is up to 25 ppm in a 20 - 50° C testing environment. The frequency is not sufficiently stable for RTC accuracy which requires a value of less than 11 ppm. Our goal is to control the frequency stability under ±0.1 ppm. The OCXO is suitable for our requirements and reduces the influence of an environment change while under constant temperature control. The block diagram of the OCXO is shown in Figure 3.

Although we can control the frequency stability, perfect frequency stability cannot be expected. As we use the
10MHz OCXO, we find that the frequency deviation can be -1 ppm (9,999,990 Hz) as measured by a high-quality frequency counter. However, the frequency stability can be up to 10 ppm if the oscillation is unstable due to environment temperature deviation. Therefore we need to use additional technology such as an CPLD programmable solution as shown in Figure 4 for compensation for the frequency deviation to achieve better accuracy and stability at 10MHz ± 0.1 ppm.

(2) The generation of the one-second reference signal from the CPLD module

The role of the CPLD module is to count the clocks to send the one-second reference signal to the UUT. Ideally the 10MHz oscillator should output a 10,000,000 Hz frequency every second, but due to the physical limits of the material it is impossible to reach this level. If there is a -1 ppm frequency error of the oscillator, then the frequency output is only 9,999,990 Hz per second. To provide an accurate one-second reference signal, the CPLD module should know the actual one second of this reference oscillation first and then compensate for the output of the one-second reference signal. As the known 10MHz (-1 ppm) is equal to 9,999,990 Hz (10 clocks missing), the outside DIP switch of the CPLD could be set to -10 in advance until the inside counter (32-bits counter) of the CPLD counts to 9,999,990 and the CPLD outputs the one-second signal via the I/O port. The compensation setting would increase both frequency accuracy and stability.

To both improve the resolution of the adjusting switch setting and to obtain the compensation, we increase the reference oscillator in the RTC measurement system. For example, adjusting one switch increases the resolution for a 10MHz oscillator to 0.1 ppm and for a 100MHz oscillator to 0.01 ppm.

The CPLD module in Figure 4 provides two functions. First it generates a calibrated one-second reference signal by using two 24-bit counters. The first counter transfers the state when the counter reaches 4,999,995 and the second counter transfers the state when the first counter reaches 9,999,990. This two-state transition sends out the calibrated one-second reference signal for the measurement. In addition the two 24-bit counters are cleared for the next use of the one-second reference signal. The second function of the CPLD module is measuring the accuracy of the one-second reference signal from the RTC circuit of the server system. The 32-bit counter of the CPLD module counts the number of clocks from the RTC circuit. The value of the 32-bit counter is read back from the registers of the 8051 microcomputer through the address A0-A3 as the variable C2. In addition to the real frequency of the oscillator of 10 MHz, C1, we obtain the error rate of the RTC circuit by [(C2 − C1) / C1] x 10^6.

For example, if we obtain 10,000,000 clocks per second, then C1 is equal to 10,000,000. In addition, if we read the 1 Hz from outside, the 32-bit counter shows that the number of clocks from the RTC circuit is 10,000,510, which means C2 is equal to 10,000,510. Then we obtain [(10,000,510-10,000,000) / 10,000,000] x 10^6 = +51 ppm. In other words, the clock rate of the RTC circuit is 51 ppm faster than the reference clock from the oscillator. Equivalently, the clock of the RTC circuit in the server system is 4.4064 seconds, that is, 26.8 minutes per year faster than the reference time. Figure 5 shows the block diagram of the frequency deviation of the calibrated signal.

(3) The digital temperature sensor

This senses the surface temperature of the OCXO and sends the temperature to the 8051 microcomputer.

(4) The 8051 microcomputer module

This module senses the temperature variation of the surroundings and sends it to the CPLD module which is
pre-stored for the characteristics of the temperature compensation for the crystal oscillator.

Another connection of this module is to the PC for long-time monitoring. The 8051 sends the collection data of both the temperature and the output frequency to the PC for storing and analyzing.

(5) The LCD Display
The function of this module is real-time display of temperature and output frequency.

(6) The transmission interface of the RS232
The system design is for test use on the production line, so the CPLD module sends out a one-second reference signal to all UUT via the RS232.

The flowchart of the measurement program shown in Figure 6 comprises four functions: (1) disable all interrupt functions of the system, (2) set the reading rate of the RS232 port, (3) read the 64-bit register function of the timing clocks and (4) the 64-bit arithmetic computation.

![Figure 6 Test program flowchart of the RTC measurement](image)

(1) Disable all interrupt functions of the system
Because the time interval technique is used for the high measuring accuracy of the RTC, the testing units must disable all interrupt functions. The testing program and the tested CPU must concentrate on clock counting without interference from the interrupt of the external events. The testing program is executed under a DOS environment with a single tasking. If our design is working in the multitasking environment in Windows, the reading for the CPU clocks cannot be accurate because the communication port is not used by a single task.

(2) Set the reading rate of the RS232 port

According to the testing procedure as shown in the flowchart in Fig. 6 the system continues to send a one-second reference signal to the RS-232 port of the tested unit. The tested unit executes the measurement program. This program first disables all interrupts, continues reading the status of the RS232 port until it deletes a rising edge and then executes the instruction RDTSC to transfer the values of the registers EDX: EAX into the

The system architecture must have very accurate timing as its base, so any event which interrupts the reading of the RS232 port can affect the accuracy of measuring. For instance, the tested UUT needs to read the one-second reference signal, so the status of the RS232 port can affect the accuracy of measuring. From the experimental results we learn that a 2GHz CPU is 520 thousand times as fast as the reading of the RS232 port. The maximum accuracy can be less than 2 ppm; hence, to increase the accuracy of the reading of the RS232 port we need to extend the measuring time to 10 seconds. The error in reading the one-second reference signal from the RS232 port can occur at the first clock or the last clock. In comparison with a 2 GHz clock of the server system CPU, the error in reading the status of the RS232 port can last for 3846 CPU clocks. Therefore, to increase the frequency accuracy to 0.2 ppm or 0.1 ppm we need to extend the measuring time to 10 or 20 seconds respectively.

(3) Read the 64-bit register function of the timing clocks
There is a 64-bit MSR (Model Specific Register) in a Pentium CPU to represent and count the number of clocks when the system power is on. Hence we need not use the external hardware to count the CPU clocks. When the PC executes the instruction RDTSC, the number of CPU clocks is stored into the registers EDX: EAX which represent the 64-bit registers where EDX and EAX represent upper and lower 32-bit registers respectively. To obtain the numbers of the CPU clocks within one second, the CPU must first execute the instruction RDTSC and record the values of the registers EDX and EAX. Then, after one second, the CPU must execute the instruction RDTSC again and record the new values of the registers EDX and EAX. By subtracting the old from the new values in the registers EDX and EAX we obtain the number of CPU clocks within one second. For example, the difference between the new and the old values can be around 2,000,000,000 by using 2 GHz CPU. In addition there are $2^{32}$ / $(2 \times 10^9)$ seconds, 292 years for the first overflow in the 64-bit register. Even for a 3 GHz CPU there are 194 years for the counting in a 64-bit register. Hence we need not worry about the overflow problem in a 64-bit register for counting the CPU clocks.

(4) 64-bit arithmetic computation
Under the DOS environment of the tested unit there is no 64-bit arithmetic computation. So the test program requires a special arrangement by merging the several smaller integer computations for the 64-bit arithmetic computation.

According to the testing procedure as shown in the flowchart in Fig. 6 the system continues to send a one-second reference signal to the RS-232 port of the tested unit. The tested unit executes the measurement program. This program first disables all interrupts, continues reading the status of the RS232 port until it deletes a rising edge and then executes the instruction RDTSC to transfer the values of the registers EDX: EAX into the
register R1. The measurement program continues reading the status of the RS232 port for 10 seconds, which represents ten times of LOW to HIGH for the status of the RS232 port. The measurement program again executes the instruction RDTSC to transfer the values of the registers EDX: EAX into the register R2. From the difference between R1 and R2 we learn the number of CPU clocks, C1=R2-R1, in the tested unit during the 10 seconds. By using the same procedure, we obtain R3 and R4 and the difference between R3 and R4, or C2=R4-R3. When we obtain C1 and C2 we can obtain the amount of error between the tested unit and the reference signal as shown in Eq. (2).

\[
\text{Inaccuracy in ppm of UUT} = \left(\frac{C2 - C1}{C1}\right) \times 10^6 \text{ ppm}
\]

(2)

4. Experimental Results

To determine the speed and accuracy of measuring, we set the measuring time to 20 seconds, so that we can enhance the measurement accuracy to ±0.1 ppm. The present requirement of the server system in production is accuracy in the RTC of about ±11 ppm - ±25 ppm. By using our testing design we provide a higher accuracy for the measurement of RTC of a high quality server system.

Figure 7 shows the test prototype and actual measurement by a high-accuracy frequency counter. We use a high-accurate frequency counter to calibrate the one-second reference signal from the output of the CPLD module and set the switch to obtain an accurate one-second reference signal. In Figure 7 we obtain an accuracy of under 0.0035 ppm by careful setting. Equivalently, the measuring accuracy can reach 1.0000000035Hz for a one-second reference signal. Due to the aging of the crystal oscillator we need to calibrate the system with a highly accurate frequency counter, because the purpose of the design is to generate an accurate one-second reference signal.

![Image](image.png)

Figure 7 The RTC measurement system calibrated by the frequency counter with high accuracy

4.1 Design Consideration and Enhancement of Frequency Accuracy

For rapid development and simulation we use CPLD only for real-time prototype and system validation. In the design phase, CPLD offering more flexibility and capacity, an increasing number of projects previously done in ASIC are being started, even taken to production, in programmable logic. While CPLD programming, debugging and fixing has negligible cost compared to an ASIC turn, this is one of the best ways to get an CPLD design to implement a function earlier. The key to fast debugging is visibility.

We learn that a crystal oscillator even in an oven is affected by the surrounding temperature. Figure 8 shows the relationship between frequency deviation and surrounding temperature. When the surrounding temperature changes, the frequency deviation of the oscillator occurs as shown in Figure 8. If a deviation is -30 ppm from the 10MHz crystal oscillator, then there is shortage of 300 clocks per second. This deviation can affect the accurate frequency of the one Hz reference signal from the CPLD module.

![Image](image.png)

Figure 8 The relationship between frequency deviation and surrounding temperature

Even when the crystal is placed in an oven with a constant temperature, there is still a little deviation from the oscillation frequency. Table 1 shows the measurement results of the frequency deviation ranging from 0.028 ppm to 1.297 ppm. We record the relationship in Table 1 and prestore it in the 8051 module. If the sensor detects a temperature of 38°C, then the 8051 module sends the prestored value with +0.724 ppm to the CPLD module. The CPLD module deducts seven clocks from the external 10MHz clock signal and then the frequency deviation of the crystal oscillator is reduced to 0.024 ppm from 0.724 ppm. In other words, by using the pre-stored table we can improve the frequency accuracy by ten times [5, 6]. Table 1 shows the relationship between both the output frequency and the temperature, and Table 2 shows the design specification.

<table>
<thead>
<tr>
<th>Temperature of Chamber</th>
<th>Temperature of OCXO</th>
<th>Output Frequency</th>
<th>PPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>43</td>
<td>1.000,001,297</td>
<td>1.297</td>
</tr>
<tr>
<td>35</td>
<td>38</td>
<td>1.000,000,724</td>
<td>0.724</td>
</tr>
<tr>
<td>30</td>
<td>34</td>
<td>1.000,000,350</td>
<td>0.350</td>
</tr>
<tr>
<td>25</td>
<td>28</td>
<td>1.000,000,093</td>
<td>0.093</td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>1.000,000,028</td>
<td>0.028</td>
</tr>
</tbody>
</table>

![Image](image.png)

Table 1 The relationship between output frequency and temperature
From the comparison of the measurement results, we confirm that our CPLD engine works correctly, meets our design as expected and provides a mechanism in a multimedia server or high end server system that requires accurate real time clock synchronization.

Table 2 The design specifications of the RTC measurement system

<table>
<thead>
<tr>
<th>Frequency output</th>
<th>1Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency vs. Operating Temperature Range</td>
<td>1.269ppm max. (20 – 40° C)</td>
</tr>
<tr>
<td>Output Waveform</td>
<td>Square wave</td>
</tr>
<tr>
<td>Frequency Adjustment Range</td>
<td>+/- 0.1ppm</td>
</tr>
<tr>
<td>Frequency Stability</td>
<td>&lt; +/- 0.1ppm</td>
</tr>
<tr>
<td>Power for to Stabilization Time</td>
<td>&lt; 5Min. at +25 °C</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>+10° C - +60° C</td>
</tr>
<tr>
<td>Operating Humidity Range</td>
<td>+20% - +80% R.H.</td>
</tr>
<tr>
<td>Operating Voltage Range</td>
<td>+5V (+/-5%)</td>
</tr>
<tr>
<td>Support UUT for Manufacture</td>
<td>144 test servers max.</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper our modular and programmable design has demonstrated highly efficient, adjustable and automatic measurement of the accuracy of the RTC of the mainboard of high-end server system production. Our RTC measurement system can be calibrated by setting the adjustable switches and using a high-accuracy frequency counter. After calibration, within a 20-second testing period, our design measures the RTC with an accuracy of +/- 0.1 ppm which is good enough as a reference for the accuracy of 11 ppm at the current quality level of the RTC. In the future we will increase the clock rate of the measurement system so as to increase measuring accuracy and reduce measuring time. We will also increase the CPU frequency of the mainboard in a server to increase RTC accuracy.

References