The role of server software firmware (BIOS, BMC, PSOC, CPLD)

转载 zsmcdut Posted on 2022-08-18 09:25:43 Read 7k Collection 76 Likes 7

Category Column: SATA Article Tags: server network

摘要 This article explores server hardware management in depth and analyzes the workflow of BIOS (Basic Input Output System), including POST self-test, hardware initialization, memory address allocation, MBR loading, etc. At the same time, it introduces the key role of BMC (Baseboard Management Controller) in server monitoring and remote management, such as IPMI protocol, sensor data recording and system event log. In addition, the article also mentions the application of PSoC (Programmable System-on-Chip) in hard disk management and the function of CPLD (Complex Programmable Logic Device) in logic configuration.

The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

Written in front:

serverFunctionally, there is no essential difference between them and the desktops and laptops we use for daily study and office work. They are mainly composed of the three most critical components: CPU, memory , and hard disk. However, the application scenarios of servers are mainly provided to enterprises and other roles to support various businesses. They are not just used to install browsers to access web pages and install players to watch movies. Therefore, servers will use stronger configurations, that is, more powerful CPUs, larger memory, and larger hard disk storage. In terms of appearance, servers have a variety of appearances, such as rack-mounted, blade-mounted, tower-mounted, etc. The main difference is that servers are generally not equipped with monitors, keyboards, mice and other components for near-end operations. They are generally accessed through IP remote connections . Because servers are generally placed in separate computer rooms, and the fan noise will be louder than home computers.

FirmwareGenerally refers to firmware , which is a program written into EPROM (erasable programmable read-only memory) or EEPROM (electrically erasable programmable read-only memory).

Firmware refers to the device "driver" stored inside the device. Through firmware, the operating system can implement the operation of a specific machine according to the standard device driver. For example, optical drives and burners have internal firmware.

Firmware is the software that performs the most basic and lowest-level work of a system. In hardware devices, firmware is the soul of the hardware device, because some hardware devices have no other software components except firmware, so firmware determines the function and performance of the hardware device.

principle

Firmware is generally stored in the EEPROM (Electrically Erasable Programmable ROM) or FLASH chip in the device. It is a program that can be upgraded by the user through a specific refresh program. Generally speaking, the software that performs the most basic and lowest-level work of a digital product can be called firmware, such as the BIOS (Basic Input/output System) on the computer motherboard . In the past, professionals actually called it firmware.

Usually, the programs stored in these hardware cannot be directly read or modified by users. In the past, it was generally not necessary to upgrade the firmware. Even if a serious bug was found in the firmware, professionals had to replace the original machine with a chip with a written program. Early firmware chips generally adopted ROM design. Its firmware code was solidified during the production process and could not be modified by any means. With the continuous development of technology, modifying the firmware to adapt to the constantly updated hardware environment has become an urgent requirement of users. Therefore, rewritable programmable and erasable read-only memory EPROM (Erasable Programmable ROM), EEPROM and flash appeared. These chips are re-flashed, allowing the firmware to be modified and upgraded

.

1. BIOS

1. Interface information

Main interface:

BIOS information: including version, build date, firmware version, etc.

Motherboard information: platform, processor, PCH, etc.

Memory information: memory capacity

System information settings: language, time

ME (Intel Management Engine): Intel ME is the abbreviation of Intel Management Engine, which is translated into Chinese as Intel Management Engine. Intel ME refers to a microprocessor in Intel chips that is independent of the CPU and operating system. ME has a remote management function, which can remotely manage enterprise computers without user control when serious vulnerabilities occur.

Advanced options:

Configuration of CPU, PCH, PCI and USB, etc.

1. Processor configuration:

Prefetch

Cache technology uses the principle of locality to make the faster upper memory a buffer for the lower memory. Based on technical limitations and cost considerations, the capacity of the upper memory is much smaller than that of the lower memory. If the data exists in the upper memory, it can be read and written directly. This situation is called a hit, and the statistical probability of a hit is called the hit rate; if it does not hit, it must involve accessing the lower memory, which is also called a miss.

However, as the scale of applications continues to expand, the proportion of cache misses is increasing, becoming the main factor affecting cache performance. In order to improve the cache hit rate,

prefetch technology issues a prefetch request before the cache may fail by overlapping calculations and memory accesses, so that the data block can be taken into the cache in advance when the data is actually used, thereby avoiding processor stalls caused by cache misses.

2. Common reference codes

3. UPI configuration

4. Memory configuration

5. IIO configuration

The IIO subsystem of the Linux device driver is a kernel subsystem dedicated to analog-to-digital converters (ADCs) and digital-to-analog converters (DACs).

6. Advanced power management configuration

CPU P-state control

The following are the specific power management states of the ACPI specification:

Global system states · Visible to users · Divided into 4 states: G0, G1, G2, G3

Sleeping states · Sleeping states in global state G1 (excluding S5) · Divided into 5 states: S1, S2, S3, S4, S5

Device power states · Invisible to users · As long as one configuration is displayed as "on" (startup) state, the other is displayed as "off" (shutdown) state · Divided into 4 states: D0, D1, D2, D3

CPU power states (CPU Power states) · Also called CPU sleep state · In global state G0 · Divided into 5 states: C0, C1, C2, C3, C4 · There will be C6 state (Penryn) in the future

CPU / Device Performance states (CPU / Device Performance states) ·Voltage and clock frequency are determined by workload·The total amount of P-states conforms to CPU/device specifications·For example, the higher the CPU multiplier, the higher the P-states

CPU Thermal Monitor ·When the CPU temperature exceeds the limit, the monitoring system will reduce the performance of the CPU·In TM1, the CPU performance is reduced by changing the cycle·In TM2, the CPU performance is reduced by changing the clock frequency and core voltage (P-state)

Server Management

You can set the BMC's waiting time, preset value, network configuration, user information, and logs. You can also view the BMC's self-test status, version, and system event logs.

Security

Set the administrator password and user password

. Entering the administrator password will allow you to enter the BIOS SETUP; the user password is the power-on password, which allows you to enter the boot list or BIOS interface.

BOOT:

1. setup prompt timeout: Set the waiting time for the screen prompt during startup

2. bootup numlock state: refers to the state of the number lock key on the keyboard during startup.

3. quick boot (quick boot) is recommended to be enabled (enabled), which can increase the boot speed

. Description: The hardware must be detected every time the computer is turned on. Set quick boot in the BIOS, and the computer will not perform a comprehensive self-check when it starts, thereby speeding up the system startup speed. quiet boot (quiet boot)

Description: This item allows you to display the vendor logo on the boot screen. The setting value is: disabled, enabled. The default setting value is "enabled".

4. boot mode select: Select UEFI/Legacy

5. fixed boot order priorities: fixed boot order priorities

6. CSM configuration

CSM (Compatibility Support Module): It is a drop-down sub-item in the Boot option on the BIOS (some old motherboards do not have this option), and it is a parallel item with Secure Boot. Enabling CSM allows support for UEFI boot and non-UEFI boot. If you need to boot a traditional MBR device, you need to enable CSM. If you turn off CSM, it will become a pure UEFI boot, and fully support secure boot. Secure Boot is only applicable to operating systems that use UEFI to boot. In the BIOS of a laptop, the two options Enabled and Disabled are more commonly used. In the case of a computer with Windows 8, Secure Boot is Enabled by default. As a result, CSM (Compatibility Support Module) is Disabled by default, which in turn causes the computer to be unable to boot devices that do not fully support UEFI. To enable the computer to boot devices that do not fully support UEFI, you must turn off Secure Boot and then turn on CSM.

Save and Exit

Select whether to save the previous settings, or restore the default settings and exit BIOS.

2. BIOS stage division

BIOS can be roughly divided into 6 stages, as follows: 1.SEC: Security (handle platform restart events; create a temporary memory area (note: the memory has not been initialized at this time); act as a trusted root in the system; pass information to PEI. Developers can pay little attention to it and rarely involve it) 2.PEI: pre-efi initialization (initialize some permanent memory; memory in HOBs (Hand-off Blocks); and the FV (firmware volume) location in HOBs; pass control to the DXE stage. Developers need to pay attention to it, and it is used quite a lot) 3.DXE: driver execution environment (developers need to pay special attention to it. As the name suggests, the execution environment of the hardware driver on the server is closely related to the use of subsequent peripherals) 4.BDS: boot device select (initialize the console device; load the device driver; try to load and execute the startup item. Developers also need to pay attention to it) 5.RT: run time service (this level is basically the same as OS AL: after life (transition from the OS back to the environment) of system

3. Communication between BIOS and BMC

In x86 servers, BIOS needs to deal with various hardware and chips, including BMC (Baseboard Management Controller).

The communication between BIOS and BMC mainly uses IPMI. There are two stages, PEI and DXE (including those after DXE), which use different IPMI functions (this needs attention). Although IPMI is used, there are two channels, KCS and BT. Generally, the KCS channel is used. Please remember that BMC cannot actively communicate with BIOS; BIOS will send IPMI commands to BMC. If BMC successfully receives them, it will return information to BIOS, so "it looks like BMC can communicate with BIOS".

If a communication failure occurs, how to solve it: (1) Confirm whether the BIOS has sent an IPMI command to the BMC. You can check the completion code returned by the BMC; (2) Confirm whether the BMC has received the IPMI command sent by the BIOS; (3) If the BIOS has sent an IPMI command but the BMC has not received it: it may be that the BMC's IPMI process is busy, so it has lost the IPMI command (if the BIOS fails to send the command, you can try to send it multiple times; in addition, you can slightly increase the delay of the KCS channel); of course, it is also possible that the BMC's IPMI process has crashed. (4) If it involves the specific implementation of the underlying protocol, an IPMI command usually involves two underlying implementation functions, SendDataToBmcPort() and ReceiveBmcDataFromPort(). When receiving, the BIOS reads data from the I/O port of the KCS. After reading, it will check the OBF status register in the KCS register. If the BMC has not written data, it will count down by 1, delay 5ms, and then try again; if the BMC has not written data, the count reaches 0, and it is considered that the BMC has a fault and returns Device Error.

4. BIOS works during computer startup

4.1 Basic Concepts

4.1.1 BIOS (Basic Input Output System)

BIOS directly interacts with hardware and provides the operating system with basic functions for controlling hardware devices.

BIOS is divided into system BIOS (mainboard BIOS, the main code that controls computer startup), graphics card BIOS and other device BIOS (IDE controller, SCSI card or network card).

BIOS is generally stored in ROM (read-only memory chip), and these codes will not disappear after shutting down or losing power.

4.1.2 Memory Address

Each byte of memory is assigned an address for CPU access. The original 8086 processor can only access a maximum of 1MB of memory (0 to FFFFFH): the low-end 640KB is called basic memory; A0000H to BFFFFH is reserved for the video card memory; C0000H to FFFFFH is reserved for BIOS, where the graphics card BIOS is generally at C0000H to C7FFFH, the IDE controller BIOS is at C8000H to CBFFFH, and the system BIOS generally occupies the last 64KB or more.

4.1.3 MBR

Disk refers to devices such as hard disks, floppy disks, USB flash drives, CD-ROMs, etc. For each type of disk, there are MBR and partitions to make up the disk. According to the standard, each disk can have up to 4 primary partitions and 1 extended partition. The extended partition can be divided into multiple logical partitions. The first sector of each partition of the disk is used to store special information, such as the boot loader, and is not used to store files and other information (the first sector of each partition cannot be accessed through the file system). Note: MBR does not belong to any partition. In addition to storing the boot loader, the MBR also stores the partition table of this disk in the last 64 bytes.

4.2 BIOS boot process

4.2.1

When the power switch is pressed, the power supply starts to supply power to the motherboard and other devices. At this time, the voltage is not stable yet, and the control chipset on the motherboard will send and maintain a RESET signal to the CPU, allowing the CPU to automatically restore to its initial state, but the CPU will not immediately execute instructions at this moment. When the chipset detects that the power supply has begun to stabilize (of course, the process from unstable to stable is only a matter of seconds), it will remove the RESET signal (if you manually press the Reset button on the computer panel to restart the machine, then the chipset will remove the RESET signal when you release the button). The CPU immediately starts to execute instructions from address FFFF0H. This address is the address range of the system BIOS, and here is just a jump instruction to jump to the real startup code in the system BIOS.

4.2.2 The system BIOS boot code first performs POST (Power-on self test)

POST mainly detects whether some key devices in the system can work properly, such as memory and graphics card.

Since POST is the first detection process, the graphics card is not initialized at this time. If the system BIOS finds a fatal error during the POST process, such as: no memory found or memory problem (only 640k conventional memory is checked at this time), then the system BIOS will directly control the speaker to sound an error report. Different

sounds indicate different error types.
Under normal circumstances, POST is very fast, and after completion, other codes will be called to perform more complete hardware detection.

4.2.3 System BIOS searches for graphics card BIOS

After the system BIOS finds the graphics card BIOS at C0000H, it calls its initialization code, and the graphics card BIOS initializes the graphics card. At this time, some initialization information (manufacturer, graphics chip, etc.) will be displayed on the screen.
The system BIOS will then search for the BIOS programs of other devices, and after finding them, it will also call the initialization codes inside these BIOS to complete the device initialization.

4.2.5

The system BIOS will then detect the CPU type and operating frequency, test all RAM, and display the progress of the memory test on the screen. You can decide in the CMOS settings to use a simple test method that takes less time or a detailed test method that takes more time.

4.2.6

After the memory test passes, the system BIOS will begin to detect some standard hardware devices installed in the system (hard disk, CD-ROM, serial port, parallel port and floppy drive, etc.). Newer versions of the system BIOS will also automatically detect and set the memory timing parameters, hard disk parameters and access modes during this process.

4.2.7

After the standard device detection is completed, the plug-and-play code in the system BIOS starts to detect and configure the plug-and-play devices installed in the system. Every time a device is found, the system BIOS will display the device's name, model, etc. on the screen, and allocate resources such as interrupts, DMA channels, and I/O ports to the device.

4.2.8

After all hardware detection and configuration are completed, most system BIOS will clear the screen and display a table at the top of the screen, which briefly lists the various standard hardware devices installed in the system, the resources they use, and some related working parameters.

4.2.9

Then the system BIOS will update ESCD (Extended System Configuration Data). ESCD is a method used by the system BIOS to exchange hardware configuration information with the operating system. This data is stored in CMOS (a small piece of special RAM powered by the battery on the motherboard). Usually ESCD data is updated only after the system hardware configuration changes, so you will not see the message "Update ESCD... Success" every time you start the machine. However, the system BIOS of some motherboards uses a different data format from Windows 9x when saving ESCD data, so Windows 9x will modify the ESCD data to its own format during its own startup process. But the next time you start the machine, even if the hardware configuration has not changed, the system BIOS will change the ESCD data format back. This cycle will cause the system BIOS to update ESCD every time you start the machine, which is why some machines will display relevant information every time they start.

4.2.10

After the ESCD is updated, the system BIOS boot code will do its last job: boot the MBR from the floppy disk, hard disk or optical drive according to the boot order specified by the user. The system BIOS will read the master boot record MBR on the disk and put it into the memory at the specified location (0x7c00). Then the BIOS will hand over control to the MBR. The master boot record consists of two parts: code and partition table. The MBR code first checks other codes (such as checking whether there is a "55AA" valid mark), then finds the first active partition from the partition table, reads and executes the partition boot record of this active partition.

The above are all the initialization tasks that the computer needs to complete when it turns on the power switch for cold boot. If you choose to restart the computer from Windows to warm boot, the post will be skipped and start directly from step 3. In addition, the CPU and memory test in step 5 will not be performed.

2. BMC

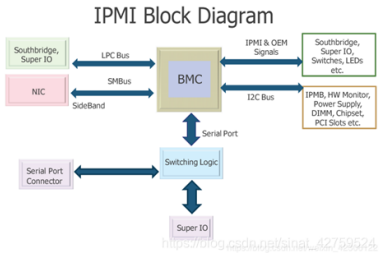
Before introducing BMC, we need to understand a concept, namely platform management. Platform management refers to a series of monitoring and control functions, and the object of operation is the system hardware. For example, by monitoring the system's temperature, voltage, fan, power supply, etc., and making corresponding adjustments, the system is kept in a healthy state. Of course, if the system is really abnormal, the system can also be restarted by resetting. At the same time, platform management is also responsible for recording various hardware information and log records to prompt users and locate subsequent problems. The above functions can be integrated into a controller, which is called the Baseboard Manager Controller (BMC for short).
BMC is an independent system. It does not rely on other hardware on the system (such as CPU, memory, etc.), nor does it rely on BIOS, OS, etc. (but BMC can interact with BIOS and OS, which can play a better role in platform management. There is system management software under OS that can work with BMC to achieve better management results). Generally, our computers do not have BMCs because they are not very useful. The CPU is enough to control some temperature, power supply, etc. management. However, for devices with high system requirements, such as servers, BMCs will be used. Of course, because BMC is an independent system, for some embedded devices, other processors may not be needed, and a BMC alone can complete the work. In the final analysis, BMC itself is also a small system with a processor (usually an ARM processor), and it is completely possible to use it alone to handle certain tasks.
BMC stands for Baseboard Management Controller. In layman's terms, BMC is the steward of the entire server board. From the moment a server is powered on, all its components are responsible for and managed by BMC. BMC is a small operating system independent of the server system. Its function is to facilitate remote management, monitoring, installation, restart, etc. of the server. BMC is a chip integrated on the motherboard (it can also be inserted into the motherboard in various forms such as PCIE). Its external manifestation is just a standard RJ45 network port with an independent IP. For ordinary maintenance, you only need to use a browser to access the IP:PORT login management interface. Server clusters generally use BMC instructions for large-scale unattended operations. Generally, the BMC network port of a server is independent. Look carefully and you will see the word BMC. There are also small servers where the BMC network port and communication network port are two-in-one.
BMC is a software that runs when the server AC is powered on. It runs on a separate ARM chip on the server. This ARM chip is the CPU of the BMC software. At the same time, the chip will be configured with its own RAM, Flash and other devices. As long as the server is plugged in, the BMC software will run quickly. At this time, it is possible that the OS on the x86 server side in our usual sense has not been installed. BMC is the steward of the entire server. It is mainly used to detect the temperature, voltage and other health status of each component of the server (CPU, memory, hard disk, fan, chassis, etc.). At the same time, it adjusts the fan speed in real time according to the situation of each temperature collection point to ensure that the server does not overheat and the overall power consumption is not too high. If any abnormality occurs in the single board component, the information will be reported to the upper-level network management in a timely manner through various industry-wide specifications such as SNMP protocol, SMTP protocol, Redfish protocol, etc., so that the operation and maintenance personnel can handle it in time to ensure that the business is not damaged.
Here we introduce the concepts of out-of-band management and in-band. Usually, the operation and maintenance management actions performed on the x86 side are called in-band methods, and BMC is a server management software that runs independently from the x86 side, which is called out-of-band management software. BMC can collect any information on any server except the private business itself running on the x86 side. However, please note that there are various solutions for BMC out-of-band management software in the industry, which can collect any information you want. Because even if the information cannot be obtained from out-of-band, there are solutions in the industry that combine out-of-band and in-band monitoring and provide unified out-of-band interfaces, which can achieve any information you want to obtain through BMC.
From a functional point of view, BMC is mainly used to collect various information on a single server and provide it to the upper-level operation and maintenance network management software. There are two main methods. The first is that BMC will provide various interfaces for upper-level network management to query, such as human-machine interfaces such as web and command line, SNMP, IPMI, Restful and other machine-machine interfaces; the second is active reporting. When a fault is detected, BMC can report to the server of the upper-level network management software through SNMP trap messages, SMTP email messages, Redfish http json messages, etc., so that the operation and maintenance personnel can identify and handle the fault in time. In general, the message reported by the BMC software will clearly indicate which component has caused the fault, what the handling suggestions are, etc.
SNMP: Simple Network Management Protocol (SNMP) is a standard protocol designed specifically for managing network nodes (servers, workstations, routers, switches, HUBS, etc.) in IP networks. It is an application layer protocol. SNMP enables network administrators to manage network performance, discover and solve network problems, and plan network growth. Network management systems are informed of network problems by receiving random messages (and event reports) through SNMP.
RESTFUL: A design style and development method for network applications, based on HTTP, which can be defined in XML format or JSON format. RESTFUL is suitable for scenarios where mobile Internet manufacturers use it as a business enabling interface to implement the function of third-party OTT calling mobile network resources. The action types are adding, changing, and deleting the called resources.
BMC is connected to other components in the system through different interfaces. Various management controllers are connected to the **IPMB (Intelligent Platform Management Bus)** bus, each of which performs different functions. Some I2C devices are also connected to the IPMB bus, which are used as sensor interfaces so that system management software can read sensor data through IPMB. At the same time, the specific configuration information of these sensors, such as alarm thresholds, whether event triggering is allowed, etc., are all stored in a set of data called SDR (Sensor Data Record). The alarm events generated by the sensors are stored in a set of data called SEL (Sensor Event Log). On the IPMB bus, there is an ICMB (Intelligent Chassis Management Bus) bridge connected, through which it can communicate with another remote management platform. In addition, on the IPMB bus, other user boards can be connected externally to expand the functions of the IPMI management platform.
The BMC chip is equivalent to the central processing unit in the computer. Through a pair of SMBus interfaces on the BMC chip to connect to the network, users can access the network to achieve out-of-band management of remote server takeover (Out-of-band), such as remote takeover of the server (Pre-OS), and achieve complete takeover of the remote server on the client; through the RS-232 interface to connect to the Modem, when the remote server is down, users can dial to obtain SDR and SEL data and analyze and diagnose the cause of the fault; BMC accesses the SMC on the module fan backplane, power backplane, etc. through the IPMB interface to achieve key parameter management of temperature, voltage, fan speed, etc. of various backplanes; BMC implements the IPMI message transmission mechanism through the system interface (mostly using SMIC: Server Management Interface Chip), controls LCD display and realizes communication between upper-layer software and lower-layer F/W, and realizes alarm and data collection. SDR (Sensor Data Repository), SEL System Event Log), FRU (Field Replacement Unit) physical entities can be storage bodies built in the chip or external E2PROM.

By deploying a self-developed Agent on the x86-side OS, I can also collect all kinds of information I want. For a small amount of information that is strongly related to hardware, I can query and complete it through the BMC interface on the in-band OS. Isn't this in-band combined with out-of-band management method also good?
Full out-of-band management is a single-board operation and maintenance management solution currently launched by major server manufacturers. Major server manufacturers currently have their own unified in-band Agent information completion tool, which ultimately aggregates information to the BMC side to provide a unified external interface. In

theory, it can meet the demands of single-board operation and maintenance management. For server users, this is a zero-cost solution, which is strongly recommended. Why bother developing another set of your own to increase costs?

However, since it is called BMC here, the focus is generally on platform management, so this article mainly talks about the BMC in the server.

The position of BMC in the system is roughly shown in the following figure:



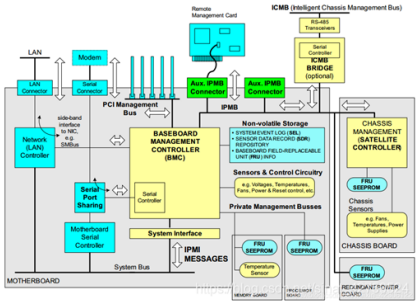
BMC is connected to other components in the system through different interfaces. LPC, I2C, SMBUS, Serial, etc., these are relatively basic interfaces, while IPMI is a bus that matches BMC. All BMCs need to implement this interface, so it needs special introduction here.

IPMI

The full name of IPMI is Intelligent Platform Management Interface. IPMI stipulates many things, BMC is the most important part of it. In addition, there are some "satellite" controllers connected to BMC through IPMB. These "satellite" controllers generally control specific devices.

IPMB stands for Intelligent Platform Management Bus, which is an I2C-based serial bus. It is used for communication between BMC and "satellite" controllers, and IPMI commands are transmitted on it.

The following figure describes the various modules related to IPMI:



Intelligent Platform Management Interface (IPMI) is an open standard hardware management interface specification that defines a specific method for embedded management subsystems to communicate. IPMI information is communicated through the baseboard management controller (BMC) (located on the hardware component of the IPMI specification). Using low-level hardware intelligence management instead of operating system management has two main advantages: First, this configuration allows out-of-band server management; second, the operating system does not have to bear the task of transmitting system status data. Users can use IPMI to monitor the physical health characteristics of the server, such as temperature, voltage, fan working status, power supply status, etc.

The core of IPMI is a dedicated chip/controller (called a server processor or baseboard management controller (BMC)), which does not rely on the server's processor, BIOS or operating system to work. It is an agentless management subsystem that runs independently in the system. It can start working as long as there is BMC and IPMI firmware. BMC is usually a separate board installed on the server motherboard. IPMI's good autonomous characteristics overcome the limitations of previous operating system-based management methods. For example, it can still perform operations such as power on and off, information extraction, etc. when the operating system is not responding or not loaded.

When working, all IPMI functions are completed by sending commands to the BMC, which receives and records event messages in the system event log and maintains sensor data records describing the sensor conditions in the system.

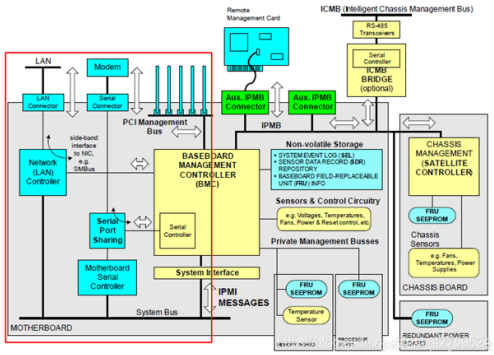
Generally speaking, BMC has the following functions:

1. Access through the system's serial port
2. Fault logging and SNMP alert sending
3. Access to the System Event Log (SEL) and sensor status
4. Control including power on and off
5. Support independent of system power or working status
6. Text console redirection for system settings, text-based utilities and operating system consoles

Through IPMI, users can actively monitor the status of components to ensure that preset thresholds, such as server temperature, are not exceeded. In this way, by avoiding unscheduled power outages, it helps maintain the operating time of IT resources. IPMI's ability to predict failures also helps manage IT cycles. By checking the System Event Log (SEL), it is easier to pre-determine faulty components. As long as the server is connected to the network and the power of the server is not disconnected, users can be allowed to remotely manage the server through the network regardless of the state of the server (power on, shutdown, restart).

The following is a brief introduction to each part.

- MOTHERBOARD



is the lower left part in the picture, and its name is Mother Board.

Usually, in the server, this part is the protagonist, and it contains the main components such as CPU, PCH, etc. It connects several components: network card, serial port and IPMI bus, and there is actually a part in the PCI bus in the middle of the top of the picture.

Network card: The server needs to use the network card, and the focus is actually on the connection from BMC to the network card, which will be introduced later.

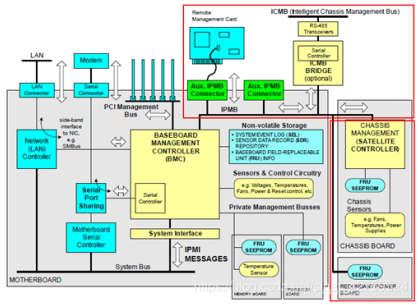
Serial port: The serial port is used to output the debugging information of the server. It is worth noting that Serial Port Sharing allows the serial port output of the server to be output directly or to the BMC. As for why it is output to the BMC, it is actually necessary to pay attention to a common scenario here. The server is located in the computer room, and the staff usually do not operate directly in the computer room, but operate through the network (this is why the BMC is connected to the network card). At this time, if you need to obtain the serial port information of the server, it is not convenient to go directly to the computer room. At this time, it is a good idea to obtain the serial port information of the server through the BMC.

IPMI bus: This is the main body for BMC to communicate with the server and control it, which is of course indispensable.

PCI bus: This part is very similar to the serial port. In addition to outputting serial port information, the server also needs to output things like graphical interfaces. From the server side, it is connected to a graphics card through PCI, through which it outputs the display.

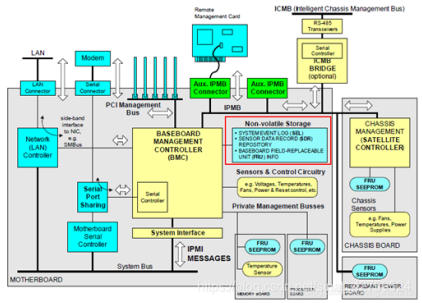
- IPMB

is in the upper right corner of the picture, which describes the devices connected through IPMB.



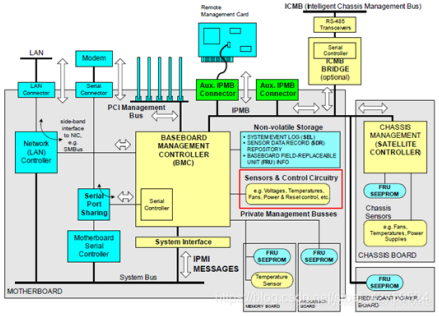
These devices are similar to BMC and are also used to manage chips. They are a supplement to BMC, thus expanding the functions of BMC.

- Non-volatile Storage



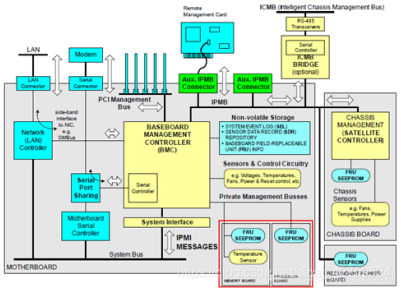
We know that BMC is actually an independent chip, so it must also run a system. A Unix-like system runs in BMC, and the system is stored in Non-volatile Storage, usually in SPI Flash. There is no essential difference from general storage media. In addition to the system itself, it also contains a series of information that BMC will store. For example, serial port information obtained from the server; system alarm information; FRU information, etc.

- Although the Sensors & Control Circuitry



section only occupies a small part in the figure, it is the most basic function of BMC: obtaining information and controlling the environment. BMC will obtain the temperature of the device through buses such as I2C/PECI, and then adjust the temperature according to the pre-set strategy. There are two ways to adjust the temperature. One is to adjust the fan, which is active cooling; the other is to adjust the power supply, such as the P state of the CPU, or shut down unnecessary hard disks, which is passive cooling.

- FRU



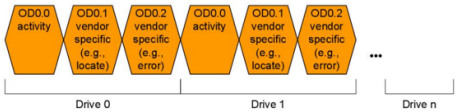
The full name of FRU is Field Replaceable Unit.
As can be seen from the figure, memory sticks, CPUs, etc. are FRUs, which are usually replaceable in servers.
BMC will detect these devices and save relevant information.
When the status of these devices changes, BMC will generate relevant alarms.

3. PSoC

Programmable System-on-Chip (PSoC) is a programmable mixed signal array architecture controlled by a built-in microcontroller (MCU) on a chip. It integrates configurable analog and digital circuits, including UART, timer, amplifier, comparator, digital-to-analog converter (ADC), pulse width modulation (PWM), filter, as well as SPI, GPIO, I2C and dozens of other components to help customers save R&D time.

Introduction

Programmable System-on-Chip (PSoC) is a programmable mixed signal array architecture controlled by a microcontroller (MCU) built into the chip. It integrates configurable analog and digital circuits, including UART, timer, amplifier, comparator, digital-to-analog converter (ADC), pulse width modulation (PWM), filter, as well as dozens of components such as SPI, GPIO, I2C, etc., to help customers save research and development time.
Altera, Atmel, Xilinx, and Lattice have all launched PSoC products. There are two ways to implement PSoC: using FPGA/CPLD; the other is to add programmable modules to ASIC.



PSOC firmware main functions:
SATA, SAS hard disk LED control;
BMC obtains PSOC version information;
BMC obtains hard disk status through PSOC;

PSOC 4 is a reconfigurable and scalable platform architecture, a programmable embedded system controller that includes an ARM Cortex-M0 CPU. Through flexible automatic routing resources, it combines programmable and reconfigurable analog modules with digital modules. The PSOC 4200-L product series based on this platform combines microcontrollers with digital programmable logic, programmable analog, programmable interconnect and off-chip memory expansion security, high-performance analog-to-digital conversion, operational amplifiers supporting comparator mode, and standard communication and timing peripherals. Targeting new applications and design requirements. PSOC 4200-L products are fully upward compatible with PSOC 4 platform series products. Programmable analog and digital subsystems support flexible and field debug designs.

BMC updates PSOC firmware.

PSOC firmware can be updated through BMC's WEBUI. PSOC firmware online update files are 'cyacd' format files. BMC cannot update PSOC's hex files. PSOC's hex format firmware files need to be burned offline at the factory through a burner or burned using tools provided by Cypress.

BMC WEBUI updates PSOC path: Maintenance->Firmware Update->Update PSOC Firmware

When the user updates PSOC in BMC's WEBUI, BMC sends an update command to PSOC, PSOC enters online update mode, and then BMC transfers PSOC's update file to PSOC, completing the update inside PSOC.

BMC sends I2C command: 0x01 0x02

PSOC returns data: 0x01/0x00

When PSOC returns 0x01, it means that PSOC has entered the online update mode, and then BMC and PSOC interact to complete the firmware transmission, and PSOC completes the firmware update action internally. If you want to confirm whether the upgraded firmware is correct, you can check the current PSOC version number in the BMC WEBUI.

4. CPLD

CPLD uses programming technologies such as CMOS EPROM, EEPROM, flash memory and SRAM to form a high-density, high-speed and low-power programmable logic device.

CPLD is mainly composed of three parts: logic block, programmable interconnection channel and I/O block.

The logic block in CPLD is similar to a small-scale PLD. Usually, a logic block contains 4 to 20 macro cells. Each macro cell is generally composed of a product term array, product term allocation and programmable register. Each macro cell has multiple configuration methods, and each macro cell can also be used in cascade, so it can realize more complex combinational logic and sequential logic functions. For CPLDs with higher integration, embedded array blocks with on-chip RAM/ROM are usually provided.

The programmable interconnection channel mainly provides an interconnection network between logic blocks, macro cells and input/output pins. The input/output block (I/O block) provides an interface between the internal logic and the device I/O pins.

CPLDs with larger logic scale generally also have JTAG boundary scan test circuits, which can perform comprehensive and thorough system testing on programmed high-density programmable logic devices. In addition, in-system programming can also be performed through the JTAG interface.

Due to differences in integration processes, integration scales and manufacturers, various CPLD partition structures, logic units, etc. also have significant differences.

Application

The emergence of reconfigurable PLDs (programmable logic devices) based on SRAM (static random access memory) has created conditions for system designers to dynamically change the logic functions of PLDs in running circuits. PLDs use SRAM cells to store configuration data. These configuration data determine the interconnection relationship and logic functions within the PLD. Changing these data also changes the logic functions of the device. Since SRAM data is volatile, these data must be stored in non-volatile memories such as EPROM, EEPROM or FLASH ROM outside the PLD device so that the system can download them to the SRAM cells of the PLD at the appropriate time, thereby realizing in-circuit reconfigurability ICR (In-Circuit Reconfigurability).