September 2004

Serial ATA
Staggered Spin-Up

September 2004

Revision 1.0

A WHITEPAPER BY:

Intel Corporation

www.intel.com

Contents

Legal Notices:

1 Summary

Platforms that include numerous Serial ATA hard disk drives may be presented with power system design issues related to the electrical current load presented during system power-up. Staggered spin up provides a simple mechanism by which SATA HBAs can sequence disk drive initialization and spin-up. Specific details with regards to SATA and staggered spin-up can be found in *Serial ATA II: Extensions to Serial ATA 1.0 Rev 1.0* specification (available from the Serial ATA Working Group at: http://www.serialata.org). This paper provides a basis for understanding how the staggered spin-up feature can be supported by both the platform system BIOS and operating system specific device drivers (staggered spin-up aware and non-staggered spin-up aware).

2 Keywords

The following terms will be used throughout this paper:

- **BIOS** – Basic Input/Output System. This is the firmware embedded on a chip (typically FLASH memory) that is responsible for executing the POST sequence and for testing and initializing the system components before booting an operating system.
- **D0**[1] – Device power state in which a device is on and running. It is receiving full power from the system and is delivering full functionality to the user. All devices must support this power state.
    - **D0**$_{active}$ **–** Device power state where the device has been configured and enabled by software and is functional.
    - **D0**$_{uninitialized}$ – A device power state that a device enters as result of PCI RST# being asserted or when instructed by software to transition from D3hot to the D0 state.
- **D1** – Device power state in which the device is in a light sleep power conservation state. This power state is optional and can only be entered from the D0active power state.
- **D2** – Device power state in which the device is in a deeper sleep state than D1 but less than D2hot. This power state is optional and can only be entered from the D0active and D1 states.
- **D3 Full Off** – Device power state in which a device is off. In this state a device loses its context and power is removed from the device. All devices must support this power state.
    - **D3**$_{hot}$ – Device power state that occurs when a device transitions to D3, yet still has Vcc applied.
    - **D3**$_{cold}$ – Device power state that occurs when a device transitions to D3, but Vcc is not applied.
- **Emulation mode** – Optional mode supported by a SATA HBA. Allows non-native SATA aware software to access SATA devices via traditional task file registers.
- **HDD** – Hard Disk Drive.

[1] Device power state definition and behavior can be found in the PCI Specification v2.3 which is available from www.pcisig.com.

- **HBA** – Host Bus Adaptor.
- **Host software** – Refers to software running on the platform and includes system firmware as well as operating system software.
- **IHV** – Independent Hardware Vendor
- **Native SATA aware** – refers to system software (BIOS, option ROM, operating system, etc) that comprehends a particular SATA HBA implementation and understands its programming interface and power management behavior.
- **Non-native SATA aware** - refers to system software (BIOS, option ROM, operating system, etc) that does not comprehend a particular SATA HBA implementation and does not understand its programming interface or power management behavior. Typically, non-native SATA aware software will use a SATA HBA's emulation interface (e.g. task file) to control the HBA and access its devices.
- **Native mode** – Optional mode supported by a SATA HBA. Allows native SATA aware software to access SATA devices via registers that are specific to the HBA.
- **OROM** – Option ROM
- **POST** – Power-on Self Test. This is the part of the BIOS that takes control immediately after the computer is powered-on. The POST code initializes the computer hardware so that an operating system can be booted.
- **System Firmware** – Refers to system BIOS and/or option ROM software.
- **S0**[2] – System power state. While the system is in the S0 state, it is in the system working state. Device states are individually managed by the operating system software and can be in any device state (D0, D1, D2, or D3).
- **S1 –** System Power State. This is a low wake latency sleeping state. The S1 state is defined as a low wakeup latency sleeping state. In this state no system context is lost (CPU or chip set), and the hardware is responsible for maintaining all system context, which includes the context of the CPU, caches, memory, and all chipset I/O.
- **S2 –** System Power State. The S2 state is a low wakeup latency sleep state. This state is similar to the S1 sleeping state; except that the CPU and system cache context is lost (the OS is responsible for maintaining the cache and CPU context). The hardware is responsible for maintaining chipset and memory context.
- **S3 –** System Power State. The S3 state is a low wakeup latency sleep state, where all system context is lost except for system memory. CPU, cache, and device context is lost in this state; the OS and drivers must restore all device context. Hardware must maintain chipset, memory context and restore some CPU and L2 configuration context.
- **S4 –** System Power State. The S4 sleeping state is the lowest power, longest wake latency sleeping state supported. In order to reduce power to a minimum, it is assumed that the hardware platform has powered off all devices ($D3_{cold}$). Platform context is maintained. This system power state is also refered to as the *hibernation* state.
- **S5 –** System Power State**.** The S5 state is similar to the S4 state except that the OS does not save any context. The system is in the "soft" off state and requires a complete boot when it wakes.

[2] System power state definition and behavior can be found in the ACPI 2.0b specification which is available from www.acpi.info

3 Supporting Staggered Spin-up

For platforms that are staggered spin-up enabled, system firmware should always provide staggered spin-up support because:

1. On a multi-drive platform, the system firmware may not comprehend which attached HDD contains the operating system that the user wishes to boot to. Typical PC system firmware requires that the HDDs are spun-up and ready prior to presenting the user with a list of bootable devices. This system firmware requirement combined with the power supply limitations associated with a staggered spin-up enabled platform requires that any attached SATA HDDs be spun-up in a controlled manner. Doing so will prevent over-loading of the platform power supply.

2. System firmware may not comprehend if the target operating system natively supports the staggered spin up feature and as such requires that all attached devices are in a spun-up and ready state prior to OS boot. This is particularly important when the target operating system is non-native SATA aware; as it will not comprehend the staggered spin-up feature which may lead to inaccessible devices. This operating system requirement combined with the power supply limitations associated with a staggered spin-up enabled platform requires that any attached SATA HDDs be spun-up in a controlled manner. Doing so will prevent over-loading of the platform power supply.

Complete support of the staggered spin-up feature requires that host software implement staggered spin-up support for the following power states:

- Power-on Self Test (POST)
- Resume from $D3_{cold}$
- Resume from $D3_{hot}$

3.1 Performing the Staggered Spin-up Sequence During POST

For each SATA port implemented by the SATA HBA, the host performs the staggered spin-up sequence as follows:

Note: Because HBA implementation may vary in design, all examples in this paper assume a generic HBA with register/bits that provide spin-up control, device presence detection and device ‘readiness’ detection.

1. The host attempts to establish PHY communications with the device attached to the SATA port. PHY communications is initiated via a host issued COMRESET. It is the establishment of PHY communications that causes the attached device to begin rotation of its spindle. (Refer to the Serial ATA II extensions to Serial ATA 1.0a specification for the exact host to device signal exchange).

2. The host then waits for a positive indication that a device is attached to the port. The exact details of how a host detects device presence are beyond the scope of this paper. However, as per the Serial ATA 1.0a Specification, a device must respond with COMINIT within 10ms of detecting a host COMRESET. As such, it is reasonable to expect that an HBA will take advantage of this behavior and provide an appropriate mechanism for host software to use. If the host does not implement a

device detection mechanism accessible by host firmware, then waiting for the (potential) device to become 'ready' is an alternate mechanism.

3. If it is determined that no device is present, then the host proceeds to step 5. Otherwise, the host waits for indication that the attached SATA drive is ready. The device indicates 'readiness' when BSY and DRQ are both clear (0)[3]. This is identical to how device presence/readiness is determined for P-ATA devices. It is reasonable to expect that an HBA will provide a host software accessible mechanism whereby device readiness can be determined (e.g. emulated task file registers or HBA specific registers).

4. If the SATA device indicates readiness, then the host continues the staggered spin-up sequence using the next available port (go to step 1).

5. Cleanup. If no device is detected or the device does not indicate 'ready', then the host performs whatever steps are required by the HBA implementation and leaves the port in a state that is expected by the operating system. Since HBA implementations and operating systems vary in design, an exact description of this is beyond the scope of this paper. The host then continues the staggered spin-up sequence using the next available port (go to step 1).

Note: Because the staggered spin-up sequence described in this section is highly sequential, it does not represent the most optimized solution available. This sequence illustrates the fundamental requirements needed to support staggered spin-up. Any optimizations are beyond the scope of this paper and are left as an exercise for the implementer.

3.2 Supporting Staggered Spin-Up During Resume from the $D3_{cold}$ Device Power State

This section describes how system firmware supports staggered spin-up during resume from the $D3_{cold}$ device power state. The term 'device' in this context refers to a SATA HBA and not an attached SATA HDD.

3.2.1 HBA Behavioral Assumptions

A SATA HBA is in the $D3_{cold}$ device power state when its Vcc is removed as a result of the system being placed in the suspend state (S3), hibernation state (S4) or power off state (S5). Typically, system firmware does not differentiate between the S4 and S5 states and as such it will follow the sequence outlined in section *3.1 Performing the Staggered Spin-up Sequence During POST*.

When the HBA transitions from the $D3_{cold}$ power state to the D0 device power state, the following is true:

- The HBA is placed into the $D0_{uninitialized}$ state (causes a change in device power state, $D3_{cold}$->$D0_{uninitialized}$). As per the PCI v2.3 Specification, the result of this transition is:
  - The HBA PCI Configuration space is reset
  - The HBA memory mapped space is reset
- Register restoration is the responsibility of one or more of the following: system firmware, the operating system, and device driver software
- System firmware can participate in resume path processing.

[3] The maximum amount of time that a SATA device is permitted to take before indicating readiness is specified in the ATA/ATAPI-6 specification which is available from the T13 Technical Committee at www.t13.org).

- Note that this is not true for SATA HBA option ROMs; option ROMs (regardless of device type) only get initialized during initial system power on. As such, the operating system and/or device driver has full responsibility for re-initializing the SATA HBA.

Note: Power to any attached SATA devices will be lost once a SATA HBA transitions into the $D3_{cold}$ state. As such, any attached SATA devices will lose their programming and will require subsequent reprogramming by system software once the SATA HBA transitions into the $D0_{active}$ state.

3.2.2 System Firmware Considerations

To support staggered spin-up on the platform when the SATA HBA is resuming from the device $D3_{cold}$ power state and the system firmware is participating in the resume path (e.g. S3), the system firmware performs the following:

1. Those registers required for normal operating system operation (i.e. capability bits) are initialized.

2. If the system firmware is resuming to an operating system that provides native-SATA support, then the system firmware's participation in the staggered spin-up sequence is complete. It is expected that the operating system will complete the remaining steps required to spin up the drives. Eliminating the actual spin-up of the drives (from a system firmware perspective) will expedite the resume process (e.g. reduce resume latencies). Otherwise, the user could be presented with a blank screen during the interval when the system firmware is sequentially spinning up each drive in the platform.

3. If the system firmware is resuming to an operating system that does not provide native-SATA support (i.e. the device sub-class code is IDE (01) and as a result, the operating system uses the emulation capabilities of the SATA HBA) then system firmware completes the steps outlined in section *Supporting Staggered Spin-up During System Post*. System firmware must perform the complete staggered spin up sequence for this case as a non-Native SATA aware operating system will not be cognizant of the staggered spin-up feature.

3.2.3 Operating System Considerations

To support staggered spin-up on the platform when the SATA HBA is resuming from a $D3_{cold}$ device power state, an operating system device driver that is native SATA aware implements one of the following spin-up strategies:

Note: Because option ROMs are not re-initialized when a device is resuming from the $D3_{cold}$ device power state, the operating system device driver assumes that the SATA HBA's spin-up control bit can be found in its default state (off).

1. Do not spin-up any of the drives until an access request is made (e.g. read/write).

   - Care must be taken when simultaneously accessing any remaining spun down drives (i.e. must be accessed in a serialized manner until all remaining drives are spun up).

2. Spin up only the drive that contains the operating system or operating system key data (i.e. page file) as this drive is typically accessed first.

   - Care must be taken when simultaneously accessing any remaining spun down drives (i.e. must be accessed in a serialized manner until all remaining drives are spun up).

3. Sequentially spin up all of the drives during resume phase.

Each strategy described above has its own advantages and short comings. The choice of which strategy to implement may be dependent upon many factors including (but not exclusively): operating system power management policy, device driver architecture, host controller capabilities, SATA device capabilities, etc.

3.3 Supporting Staggered Spin-Up During Resume from the $D3_{hot}$ System Power State

This section describes how system software supports staggered spin-up during resume from the D3hot device power state.

3.3.1 HBA Behavioral Assumptions

A SATA HBA will enter the $D3_{hot}$ device power state whenever:

- The system is in the S0 system power state and the SATA HBA has been programmed to enter the D3 device power state.
- The system is transitioning to the S1 system power state and the SATA HBA has been programmed to enter the D3 device power state.

Upon resuming from the S1 system power state, the following is true:

- The SATA HBA will experience a $D3_{hot}$->D0 transition
    - Results in the HBA being placed into a $D0_{active}$.
    - SATA HBA PCI Configuration space may or may not be reset.
    - SATA HBA memory mapped space may or may not be reset.
- System firmware cannot participate in resume path processing.
- Any connected SATA device will remain powered and will not lose its programming.

3.3.2 System Firmware Considerations

Since system firmware cannot participate in the D3->D0 resume path, it has no requirements for supporting staggered spin-up.

3.3.3 Non-native SATA Aware OS Considerations

Since SATA devices maintain power when the SATA HBA is in the S1 system power state, each device will spin up upon being accessed (e.g. a request is made that that requires media access), provided the device was placed into standby/sleep via an ATA command issued by an operating system device driver. As such, when resuming from a $D3_{hot}$ device state, the OS must ensure that it does not permit simultaneous access to all attached devices until they have been accessed serially, at least once. For non-native SATA aware operating system device drivers, this can be managed if the devices are re-enumerated sequentially during resume processing (e.g. any device previously placed into STANDBY/SLEEP state will begin spinning up upon receipt of an ATA command).

3.3.4 Native SATA Aware OS Considerations

Same as resume from $D3_{cold}$, except no programming of the SATA devices is required as power to the devices is maintained in the S0 and S1 system power states.

3.3.5 HBA D3->D0 Implementation Considerations

If the SATA HBA implements a power management policy that is in strict compliance with the PCI v2.3 specification, then supporting staggered spin-up on a platform that is booting to a non-native SATA aware operating system may not be possible. This is due to how PCI devices are expected to perform when experiencing a D3->D0 transition. The PCI spec states that this transition is supposed to place the device into a $D0_{uninitialized}$ state, where PCI configuration space and I/O space is reset. Because a non-native SATA aware operating system will not comprehend the programming interface specific to the SATA HBA, it may not be able to restore the SATA HBA to a working state. Therefore, to maintain compatibility with older operating software, SATA HBA IHVs may want to take this into consideration when designing their hardware.

The impact to operating systems that are native SATA aware is less than for those that are not native SATA aware. Since a native SATA aware operating system will comprehend both the programming interface and behavioral aspects of the HW, it is therefore reasonable to expect that the operating system can restore the SATA HBA to a functional state (following a D3->D0 transition).