X3T10/94-240R0

Duncan Penman
Zadian Technologies
Rev 1.0, 11/10/94

Comparison of Proposals for Overlapped Command Execution on ATA Bus

There are 3 outstanding proposals on ways to allow overlapped execution of commands between the 2 devices on an ATA cable. All were presented as part of larger proposals which included command queueing. Two of the proposals were for traditional IDE devices, while one was for ATAPI devices; the suggested command queueing implementations are not compatible across the 2 device types, but the protocol for overlapped execution (i.e. concurrent execution of commands on both devices with interleaved use of the bus) potentially is common.

The information in this writeup is derived from the following documents:

1. ATA/ATAPI Multi Threading, rev 0.2, dated 9/14/94, by Western Digital
2. ATA Command Queueing, rev 5, dated 10/13/94, by Quantum
3. ATA Tagged Command Queueing, v1.0, dated 10/28/94, by Conner Peripherals

This discussion is limited to the portions of the proposals that apply to overlapped command execution. I believe it is desirable to find a protocol that both IDE and ATAPI can implement so that the two device types can share a cable, even though only one may use command queueing.

The contents of this writeup are my interpretation of the 3 source documents. This has not been reviewed by the authors of the 3 proposals.

Traditional ATA - The Baseline

The basic ATA behavior model is that there will only be one command active from the host at any one time. (There has been limited use of overlapped seek, but nothing more that I am aware of.) In consequence, a drive stays 'selected' from the time it receives a command until the command is finished. Any interrupts or DMA requests must have come from that drive, so no form of bus arbitration or selection is needed.

Overlapped Command Execution

General Description

WD - Commands are issued with interrupts disabled in 3F6. The host re-enables for interrupts and does something else while waiting for an interrupt. (The host can also leave interrupts disabled and poll for status changes.) When an interrupt occurs, the host disables interrupts again and reads both status registers to see which device posted the interrupt. It is possible for both devices to have posted an interrupt, in which case both need service. In that case, the host handles one device at a time.

Quantum - The host locks the devices when issueing a command (i.e. they cannot update their interface registers or assert interrupt request.) During command execution the drives arbitrate between themselves for use of the interrupt request line (INTRQ) and then the winner locks the host until its interrupt has been serviced. The Quantum proposal redefines register 3F6 to be 16 bits wide instead of 8, and uses the high order byte for locking, arbitration control, and queue tags.

Conner - Commands are issued as they are in ordinary ATA. During command execution the drives arbitrate between themselves for use of the interrupt request line. The arbitration signals, PDIAG- and DASP-, also tell the host which drive generatedthe interrupt. Three status register bits are redefined to tell the host whether the interrupt was for data transfer, bus disconnect, or command completion.

New or Changed Commands

WD - SELECT DEVICE grants the selected device use of the bus. (Note: I'm not sure why the DEV bit in 1F6 isn't sufficient for this.) Also, new feature register values are identified for the SET FEATURES command to Enable/Disable Shared IRQ Protocol and Enable/Disable Open Collector IRQ Protocol. Not real clear what the difference between these is, nor why two different commands are needed.

Quantum - New feature register values are identified for the SET FEATURES command to Enable/Disable Command Queueing. This is needed to activate the 16 bit width of register 3F6 where locking is controlled, even if queueing per se isn't used.

Conner - New feature register values are identified for the SET FEATURES command to Enable/Disable Command Queueing. Presumably this is needed to activate the register remapping and arbitration protocol, even if queueing per se isn't used.

Register Definition Changes

WD - Status Register

Bit 1 (IDX) becomes ATN to indicate that the drive has an outstanding interrupt request. INTRQ **may** have been asserted on the interface, depending on whether interrupts are enabled.

Bit 4 (DSC in ATA, Vendor Specific in ATA-2) becomes POST, indicating command completion.

Note: other register usage is based on the ATAPI Task File definitions. Not described here, but I don't believe these affect overlapped command execution.

Quantum - Device Control Register is redefined as a 16 bit register.
Bit 13 (Read) becomes Drive # (which one is requesting a lock)

Bit 14 (Write) becomes LRLC, Lock Request/Lock Clear
Bit 14 (Read) becomes LG, Lock Granted, set by Drive 0 for both drives

Bit 15 (WR/Rd) becomes AE, Arbitration Enable(d).

Status Register - No changes noted.

Conner - Status Register
Bit 5 (DF) becomes DISC, indicating the drive is temporarily giving up control of the bus.

Bit 4 (Vendor Specific) becomes CCPT, command complete.

Bit 3 (DRQ) becomes DTPH, Data Phase. Meaning & usage is the same as or similar to DRQ.

Features Register/Precomp Register presumably retains its meaning for the SET FEATURES command; is remapped for others.
Bit 6 - HACK, Host Acknowledge tells the drive that the host has seen the interrupt and the drive can now modify the Task File.

<u>New Interface Signal Usage</u>

WD - INTRQ may be driven concurrently by both drives.

Quantum - PDIAG- used to arbitrate for use of INTRQ. Drive 0 appears to have high priority.

Conner - DASP- and PDIAG- used to arbitrate for use of INTRQ and to notify host which drive is the source of an interrupt.

<u>How a Device 'Disconnects'</u>

WD - No explicit control. Each interrupt is for a specified service with a specified transfer length. When that service is complete, the bus is again available.

Quantum - The lock condition is cleared by the host based on PIO transfer length or DMARQ/DMACK both being deasserted.

Conner - An interrupt with DISC set in the status register tells the host that the bus is now available.

How a Device 'Reconnects'

WD - Drive asserts INTRQ and sets ATN in the status register. Other bits identify the emulated SCSI bus phase for this interrupt request (ATAPI protocol). The host issues SELECT DEVICE command to grant one device or the other access to the bus.

Quantum - Drives arbitrate. Winner asserts INTRQ or DMARQ.

Conner - Drives arbitrate. Winner asserts INTRQ or DMARQ and keeps its arbitration line asserted. DASP- for Drive 0; PDIAG- for Drive 1.

How a Command is Completed

WD - Uses ATAPI Command Phase for ATAPI devices, undefined for IDE devices but presumably unchanged.

Quantum - Nothing defined different from basic ATA.

Conner - Drive sets CCPT in status register before asserting INTRQ.

Other Required Changes

The Conner proposal identified the following items which may apply to the other proposals as well.

Arbitration requires that a drive be aware of whether the other device is asserting INTRQ or DMARQ.

Only the selected drive should clear an interrupt request on a status register read by the host. (Some drives today don't check.)

On receiving a command, the selected drive should stay busy until it has saved all necessary information from the Task File. Once BSY is turned off, the host may start writing to the Task File again.

Not a complete analysis, but I hope it will help toward a workable common protocol.