AHCI and NVMe as Interfaces for SATA Express™ Devices - Overview

By Dave Landsman, SanDisk

Table of Contents

1 Introduction

The Serial ATA (SATA™) storage interface has evolved almost unrecognizably from its beginnings as IDE/ATA (Parallel ATA - PATA) in the late 1980's. SATA is now the most widely used hard disk drive (HDD) interface in the global storage market. SATA has held up incredibly well to the performance demands of HDDs; however, in 2009-2010, solid state drives (SSDs) began to outstrip the capabilities of even 3rd generation SATA, with its 6Gb/s bandwidth.

It was considered impractical to take the SATA interface above 6Gb/s, and so the client storage market needed a way forward to accommodate the huge base of legacy SATA devices, the still vibrant market for new SATA devices, including SATA-based SSDs, as well as to incorporate new classes of very high performance devices, such as PCI Express (PCIe)-based SSDs.

SATA Express™ was created by SATA-IO in 2011 to enable a path beyond SATA 6Gb (see "Why SATA Express") for PCIe-based client SSD's.

The hardware model of SATA Express is that a host can accept a legacy SATA device, or a SATA Express (PCIe) device, per figure 1.

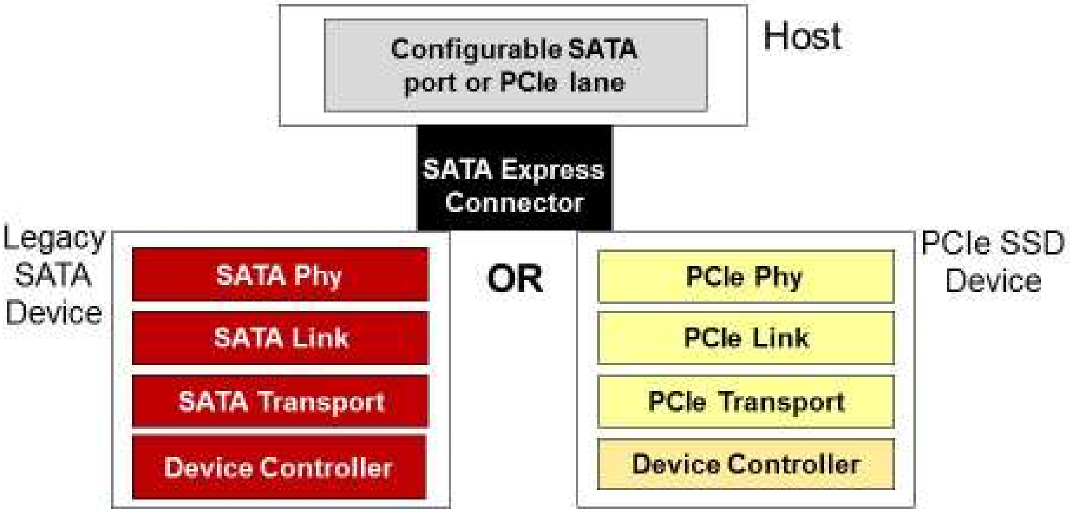

Figure 1 – SATA Express HW System Overview

While the SATA Express hardware model enables SATA and PCIe devices to be used in the same platform electro-mechanical environment, there is also the issue of the logical device interface. There are two standards-based choices for SATA Express device interface, the Advanced Host Controller Interface (AHCI) or NVM Express (NVMe). This paper describes this aspect of the SATA Express vision.

2 SATA Express Interface Architecture

Figure 2 describes the SATA Express software architecture and how SATA legacy, SATA Express/AHCI, and SATA Express/NVMe relate to one another.

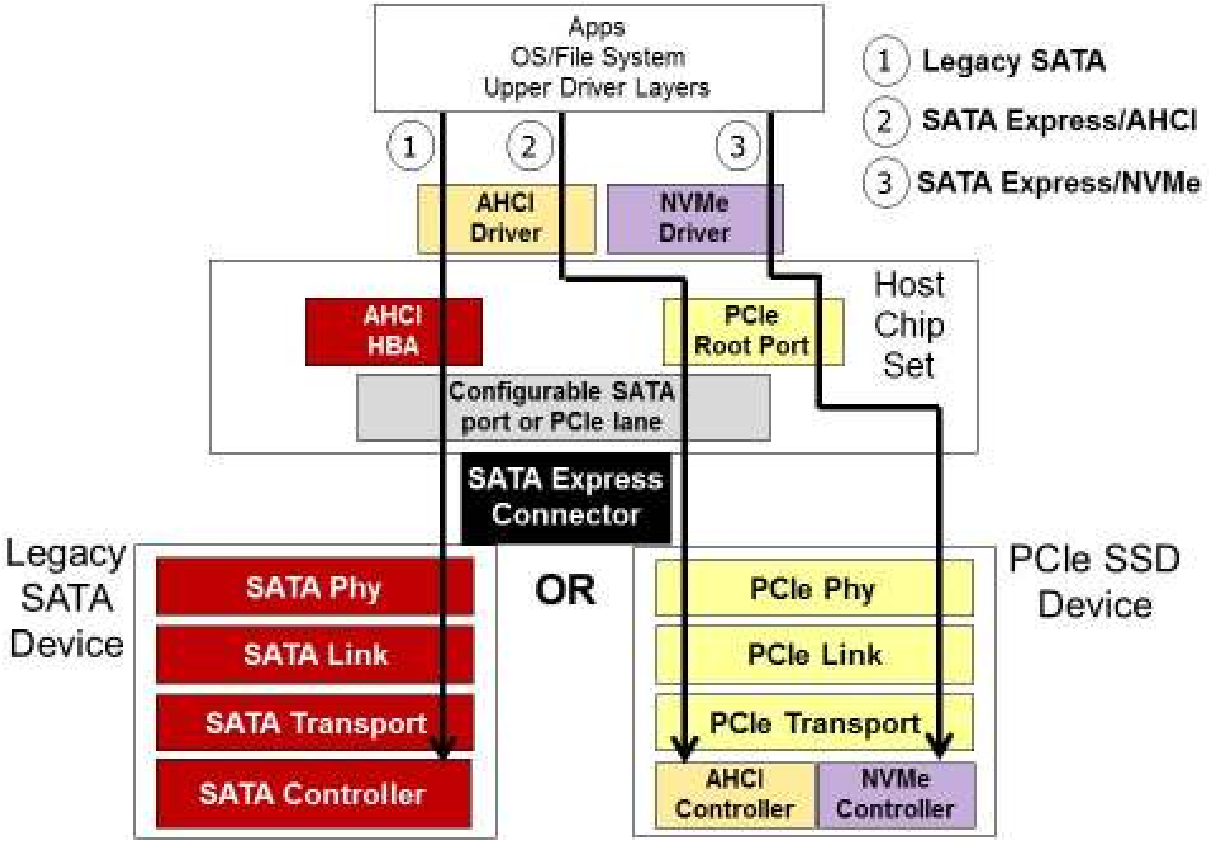

Figure 2 – SATA Express High Level Architecture

In a legacy SATA implementation (left side of figure) the AHCI interface is implemented as a Host Bus Adapter (HBA), and often built into the host chip set. In this case, applications talk to the host side of the AHCI HBA via PCIe or an internal system bus. The device side of the HBA connects to the SATA device over the legacy SATA Link and PHY channel.

In a SATA Express case (right side of figure) two standard device interface options are possible, AHCI and NVMe. In both cases, the SATA Express device appears to the system as an attached PCIe device through the host's PCIe Root Complex Port. The difference is mainly in the device driver and, of course, the device controller in the device itself.

In path (2) the PCIe endpoint, the SSD, appears to the system as an AHCI device and is accessed through the AHCI/SATA storage device stack, supported in nearly all client platforms by a standard in-box device driver.

In path (3), the endpoint is accessed through the NVMe driver. Note that in the case of NVMe, an NVMe driver must be installed on the system with the device (see Driver Availability).

3 NVMe and AHCI Comparison

While SATA Express/AHCI has the benefit of legacy software compatibility, the AHCI interface does not deliver optimal performance when talking to a PCIe SSD. This is because AHCI was developed at a time when the purpose of the HBA in a system was to connect the CPU/Memory subsystem with the much slower rotating media-based storage subsystem. Such an interface has some inherent inefficiency when applied to SSD devices, which behave much more like DRAM than spinning media.

NVMe has been designed from the ground up to exploit the low latency of today's PCIe-based SSD's, and the parallelism of today's CPU's, platforms, and applications.

It is expected that individual applications and the overall system will see an immediate performance boost on NVMe-based systems vs. AHCI-based systems, though, as of this writing, it is still early in the NVMe life cycle and performance data will become available over time. Moreover, there are advanced features in the NVMe specification which may not be fully exposed to applications in early implementations. These should add further application and system performance improvement over AHCI as they become main-stream.

At a high level, the basic advantages of NVMe over AHCI relate to the ability to exploit parallelism in host hardware and software, manifested by differences in depth of command queues, interrupt processing, the number of un-cacheable register accesses, etc. The table below summarizes the high level differences between the basic NVMe and AHCI device interfaces, which are expected to bring immediate performance increases. For more in-depth information about NVMe and AHCI, see the References at the end of this paper.

	AHCI	NVMe
Maximum Queue Depth	1 command queue 32 commands per Q	64K queues 64K Commands per Q
Un-cacheable register accesses (2K cycles each)	6 per non-queued command 9 per queued command	2 per command
MXI-X and Interrupt Steering	Single interrupt; no steering	2K MSI-X interrupts
Parallelism & Multiple Threads	Requires synchronization lock to issue command	No locking
Efficiency for 4KB Commands	Command parameters require two serialized host DRAM fetches	Command parameters in one 64B fetch

4 Driver Availability

In today's client SATA platforms, the vast majority of which are Windows based systems, the SATA software stack ships with Windows; thus, the SATA Express/AHCI solution, which utilizes legacy drivers and software, "just works" out of the box.

NVMe was first defined in 2007, as the Non-volatile Host Controller Interface (NVMHCI), and while the specification is mature, the ecosystem is still evolving and in-box Windows drivers are not yet available. However, standard Windows NVMe drivers are available at https://www.openfabrics.org/resources/developer-tools/nvme-windows-development.html, and vendors will ship drivers with their devices until/if NVMe is supported natively in Windows. Linux and VMware NVMe drivers are also available. Moreover, the University of New Hampshire Interoperability Laboratory recently added an NVMe compatibility program to begin testing devices and software for NVMe plug and play (see https://www.iol.unh.edu/services/testing/NVMe/).

5 Summary

The SATA/AHCI interface evolved to address the needs of high performance HDD devices and has served this use case ably, enabling SATA devices to be used ubiquitously from laptops to servers. The parallelism built into AHCI, while not fully enabling the parallelism available in today's host platforms, is more than sufficient for the relatively slower SATA devices it is intended to serve. SATA/AHCI is even sufficient for 1st generation SSD devices, especially in relatively lower end mobile platforms such as tablets and ultra-lite laptops.

However, in recognition of the evolution of storage requirements, and due to the technical challenges involved in taking the SATA PHY from its current maximum of 6Gb/s to 12Gb/s, SATA-IO chose to adopt PCIe, with its 8GT/s link, and capability of multi-lane implementations, as the physical interface for client storage. SATA Express hosts may accommodate both legacy SATA and SATA Express devices based on standard connectors now being defined, but even with electromechanical interoperation, there remains a choice of logical device interface.

Device vendors may implement AHCI or NVMe device interfaces for SATA Express. Both AHCI and NVMe bring advantages and costs. AHCI brings compatibility, with attendant performance inefficiencies due to legacy architecture. NVMe brings higher performance at the near term cost of in-box driver compatibility. Together, the two device interfaces enable options for SATA Express devices that provide a path from the legacy SATA environment to a PCIe based solution.

6 References

1) Why SATA Express (www.sata-io.org/technology/sataexpress.asp)
2) NVMe and AHCI – A Technical Overview (www.sata-io.org/technology/sataexpress.asp)
3) Serial ATA Revision 3.1 (www.sata-io.org)
4) Serial ATA Advanced Host Controller Interface (AHCI) 1.3 (www.intel.com/content/www/us/en/io/serial-ata/ahci.html)
5) PCI Express® Base Specification Revision 3.0 (www.pcisig.com)
6) NVM Express Revision 1.0c (www.nvmexpress.org)

This Page intentionally left Blank