Leveraging PCIe SSD performance with a full hardware NVMe

Introduction

With the recent introduction of the NVM Express (NVMe) specification, an important milestone has been reached in the race to higher performance PCIe SSDs. NVMe technology will provide the best solution for a direct access between the PCIe bus and NandFlash memories. The second milestone is the way to implement NVM Express. This Tech Talk will cover different solutions and the impact on the performances (latency and throughput).

High Performance storage market.

The demand in higher performance storage is coming from a various range of application such as cloud computing, high performance computing and video broadcast.

For each of them, there is a specific memory access profile where the performance requirements are not the same. For example, on a database search, the latency is the main important criteria, for a video broadcast system, the throughput is the key parameter.

The best choice is to use PCIe SSDs. Such devices are the only devices able to provide high throughput with a low latency, thanks to the PCIe bus (high frequency and multiple lanes). The current solutions are based on two different architectures. The first one is using SATA bridges (PCIe to SATA, then traditional SATA SSD interface): not too much difficult to design but the SATA itself is the bottleneck in this system (for both latency and throughput). The second one is using chips with direct access from PCIe to NandFlash memories, but all the NandFlash management is done on the host side, therefore adding overload on the host processor.

NVM Express was developed to reduce latency and provide faster performance with support for security and end-to-end data protection. Defined by 80+ NVM Express Work Group members, the specification, published in March, 2011, provides a flexible architecture for Enterprise and Client platforms. NVM Express is an optimized, high performance, scalable host controller interface with a streamlined register interface and command set designed for Enterprise and Client systems that use PCI Express SSDs.

NVMe write access example

NVMe is a protocol encapsulated in PCIe data packet. Here is a simplified write access description:

1) The host sets the configuration space of the device in order to inform it that there is a new submission queue ready
2) The device reads the submission queue into the host memory. The submission queue data includes the write access description: source and destination address, data size, priority…
3) The device manages the data transfer
4) The device sends a completion queue to the host

In a running system, there are multiple read and write accesses at the same time. In these cases, the device manages it through an arbitration mechanism (weighted round robin).

How the performance is impacted by this protocol? The system latency will be the sum of: the host driver latency, the multiple PCIe accesses latencies, the NVMe processing latency and the memory read/write latency. It also depends on the queue priority. The maximum throughput will depend on: the protocol overhead, the maximum data packet size, and how the NVMe device will be able to access to the host memory.

Software implementation

A software solution is an easy and fast way to design a NVM Express device. The NVM Express packets which are encapsulated in PCIe data, are stored in the memory buffer, then it is processed by a local driver.

Single CPU architecture- The embedded CPU reads the buffer, decodes the NVMe packets and processes it. From a performance point of view, the NVMe device will deliver high latency because of the clock cycles that take to the processor to do all this job.

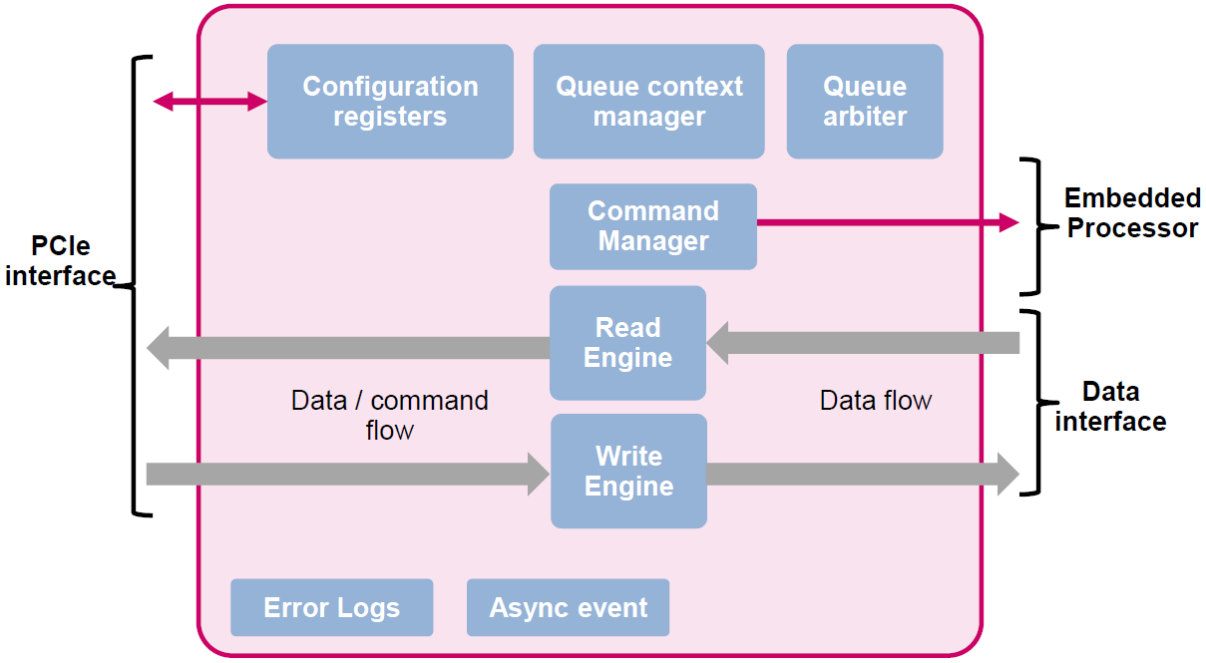
Multi-core architecture- A smartest improvement leads to design a multi-core architecture to increase performance thanks to a parallelism processing, but adding software complexity and expensive design with a higher silicon cost.

For both architectures, the throughput will depend on the memory access management integrated in the embedded CPU. If there are no specific mechanism, the CPU will be overloaded by memory access functions leading to performance decrease or a higher power consumption due to a higher frequency speed.

Even if a software solution provides flexibility, it is on the opposite strategy of the NVM Express specification where the goals are to reduce the latency and to increase the throughput.

Hardware implementation

NVM Express has been introduced to increase performance, an hardware solution seems to be an obvious way to design it. Below is the architecture of the NVMe IP from IP-Maker. All the different part required by the NVMe specification have been designed through multiple hardware blocks, including configuration space, queue context management, queue arbitration and read/write engines.



NVMe IP core architecture from IP-Maker

Each of the hardware blocks takes only few clock cycles to be processed, therefore reducing dramatically the NVMe processing latency. So the impact of the NVMe processing on the system latency is very low compared to the other latency parameters.

The data transfer rate is accelerated with the **multi DMA channels** integrated in the read and write engines. So, the PCIe bus is always used by NVMe accesses. The maximum throughput is defined by the PCIe configuration: number of lanes and speed generation.

A processor interface has been included. That allows to connect an embedded CPU for additional embedded processing such as Nandflash management including flash translation layer (FTL), wear leveling and garbage collection.

This full hardware NVMe architecture is ideal for other Non-Volatile Memories such as DDR-based storage system using super capacitors or new MRAM memories. In this case, the memory management is easy and direct therefore removing the need of embedded CPU. A simple state machine may be used to synchronize the NVMe protocol and the memory controller.

Conclusion

The combination of the NVM Express specification and a full hardware implementation is definitely the best solution to leverage PCIe SSDs performance with a reduced latency and a very high throughput. The system latency mainly consists in the host driver, PCIe and memory access latencies and is not affected by the NVMe decoding. In addition to the NVMe DMAs which manage the data transfer, an embedded processor may compute all the required memory management, therefore completely offloading the host processor.