HOW TO SCULPT A 16-CORE 2 GHZ PROCESSOR

By Ben Eckermann

Our company (Freescale at the time, before we formed part of NXP) was faced with a challenge: We had built and were sampling initial silicon of an 8-core 2 GHz processor, the LS2088A, but customers always want more. Our discussions went something like, "How much more? Do we really need more? And, if the market does need more, more of what?" What to do next? We had to pinpoint what was missing in the market and in our portfolio, and most importantly, which emerging applications and use cases did we want to pursue?

Just throwing more of everything onto the next chip was one option, but a more measured approach won out. This article dives into the thought processes behind the LX2160A, and how we solved important challenges. It might be a 2 GHz-class processor, and it might have double the number of cores of the LS2088A, but it definitely is not just "more of everything." We'll describe how we approached I/O interfaces, based on the applications we wanted the processor SoC to excel in, and how that drove aspects of the SoC architecture such as cache, SRAM and DRAM bandwidth, and more. We will discuss why we partitioned specific workloads between accelerators and general-purpose compute cores and, in doing so, how the LX2160A achieves performance goals in workloads ranging from wireless transport protocol processing to software defined storage systems for use in cloud data centers.

OUR STARTING POINT: LS2088A

The LS2088A device has eight Arm Cortex-A72 cores, each running at 2.0 GHz. In 2016, the Cortex-A72 was Arm's highest performing core. It also had the bonus of consuming less power than its predecessor Cortex-A57. It is rare for a core to improve upon the diverging targets of both performance and power simultaneously.

For each pair of Arm cores, we allocated 1 MB of L2 cache. Complementing this 4 MB of total L2 cache, the LS2088A has 1 MB of L3 platform cache. We chose a relatively small L3 cache in LS2088A so that we could devote as much of the chip's area as possible to the L2 caches, which are lower latency when accessed by the core and therefore have a larger impact on core performance than the L3. To access main memory, we endowed the LS2088A with two 72-bit DDR4 controllers, each operating at 2.1 GT/s, plus a third 36-bit DDR4 controller intended for use by the Ethernet datapath. In the LS2088A's target applications of network packet processing, significant bandwidth is needed by data structures (both packet data and routing tables) private to the Ethernet datapath, which led to the presence of the third DDR4 controller.

The LS2088A design had 16 SerDes lanes for external high-speed I/O, which could be configured to support up to four PCIe Gen3 controllers (PCIe Gen3 being the fastest speed available at the time), and 16 Ethernet MACs (eight of which supported up to 10 Gbps). In terms of acceleration, the LS2088A has specialized compression, decompression, pattern matching, and security coprocessors, as well as a programmable AIOP (Advanced IO Processor) for autonomous packet processing. The AIOP was targeted at network routing and forwarding applications where Ethernet packets went through potentially multiple table lookups and header manipulations.

With that architecture, LS2088A is capable of aggregate core performance of approximately 100,000 CoreMark or SPEC CPU2006-Int of 81. It is capable of 40 Gbps of DPDK IPv4 simple forwarding at 128-byte packet size. Or, utilizing the AIOP, the device is capable of 19.4 Gbps of complex IPv4 forwarding (complex forwarding being a use case with three exact match lookups, one longest prefix match lookup, and one 5-tuple access control list lookup per packet). Most importantly, the AIOP could achieve this rate fully offloaded from the CPU datapath, with zero loading of the Cortex-A72 cores.

EMERGING APPLICATIONS

For the LX2160A, we were looking to provide support for a range of emerging applications. Whereas we optimized the LS2088A for networking and wireless infrastructure, we wanted our next product to serve well in wireless infrastructure (which is moving from 4G LTE to 5G), network function virtualization (NFV), mobile edge computing, and new types of datacenter offload and storage applications.

DRAM

For these applications, we knew that core performance would remain important. Regardless of an application's ultimate use, it is rare to see extra core cycles go unloved. Both NFV and edge computing applications place an increasing requirement on core performance for higher-level applications in addition to the tasks of data or network packet movement of the device. However, if too much core performance was added, it would not be usable because all of those cores would be waiting for access to the memory subsystem. Often, this is referred to as "hitting the memory wall." We therefore had to first calculate how much DRAM bandwidth our target applications required within our cost constraints. The goal became: provide as much core performance as possible to utilize that bandwidth.

We had a few DRAM technologies to choose from: LPDDR4, GDDR, HBM, and DDR4. LPDDR4 provided good bandwidth, but because it is a point-to-point technology with no concept of multiple banks of chip selects sharing a common data bus, and also because it is fundamentally a ×32 technology (data bus width per chip) rather than the ×16, ×8, or ×4 technology of DDR4, the maximum DRAM system capacity achievable with LPDDR4 would be too small for our needs. The various GDDR flavors also provided good bandwidth, but their stumbling block was that they only achieve full bandwidth for long sequential accesses, something that could not be guaranteed in this system, with core-initiated cache-line-sized transactions and potentially small Ethernet packets.

We needed the DRAM technology to also have good performance when accessed coherently and when accessed from the core (both of which result in cache-line-sized transactions) – hence consideration of this in DRAM technology selection. We also needed this DRAM to also operate well with small Ethernet packets, as in many applications, the system needs to be able to respond with sufficient performance for any size Ethernet packets that may be received. 3D-stacked DRAM on the same package substrate as the main SoC (such as HBM) was also considered, but it would have added substantial package cost, and, because DRAM would need to be embedded within the package (rather than being a system-dependent design parameter), it would also limit DRAM capacity. Also, because our processors go into various applications, predicting the exact size of the memory prior to production was a challenge.

That left DDR4, which we were already experienced with, as the technology of choice. The good news is that since the launch of the LS2088A, 3.2 GT/s DDR4 was on the horizon. The same number and width of DDR4 interfaces could therefore deliver 50% more throughput that the 2.1 GT/s DDR4 interfaces of the LS2088A. To keep both our customers' PCB design costs and product costs reasonable, more than two DRAM controllers was not viable. We, therefore, decided to support two 72-bit DDR4 interfaces at 3.2 GT/s.

CORES & CACHE

Back to core selection. For the core architecture, we looked again at the Cortex-A72, as well as some other cores that were released after it. Whatever we selected needed to remain Arm-v8 AARCH64-compatible for software compatibility with the LS2088A and other products in our portfolio. The performance of each core had to remain on par with the Cortex-A72 in LS2088A. A step backwards for single-threaded code was unacceptable. Despite how much an application is designed to be parallelized across multiple cores, there is always some amount of single-threaded code (such as for exception handling or the networking control plane) in an application.

ECC protection on the caches (as on Cortex-A72) remained a must-have for reliability. Within a given amount of die area (and therefore cost) devoted to the CPUs, the performance couldn't drop. Remarkably, with that set of requirements, the optimal core remained the Cortex-A72. We tweaked the speed up slightly – from 2.0 GHz to 2.2 GHz – but it was essentially the same core.

Having established the DRAM bandwidth of the chip and the core we're using, we computed how many cores we would have by analyzing a range of benchmarks, both public and in-house. The public benchmarks all stressed the CPU, cache, and DRAM subsystems in subtly different ways. The in-house benchmarks included multiple proof-point applications we had created for packet processing. These results converged to be consistent with the results of the portions of the SPEC CPU2006-Int suite that have fewer cache hits and thus stress DRAM the most (xalankbmk and gcc for example). A rule of thumb emerged: that approximately 0.9 GB/s per GHz CPU clock was required. Using this rule of thumb, and after considering DRAM utilization efficiencies, this meant that we could fit 16 Cortex-A72 cores (16 × 0.9 GB/s × 2.2 GHz) in the DRAM's practically available 31 GB/s bandwidth (3.2 GT/s × 8 bytes/cycle × 2 DDR controllers × 60% average DRAM utilization).

To make better use of precious DRAM bandwidth, we chose to significantly increase the size of the platform cache in the coherent interconnect – to 8 MB – from the LS2088A's 1 MB. While this cache does provide some benefit for core traffic (acting as a victim cache from the core clusters' eight 1 MB L2 caches), the primary benefit of the LX2160 platform cache is to reduce pressure on DRAM bandwidth for the I/O initiators (Ethernet and PCI Express). This data is written to DRAM by one initiator and then may be read by another initiator before it is evicted from the cache. In general, a small cache is thrashed (previously written data is replaced with new data) more often than is the case with a larger cache. Therefore, LX2160's larger cache meant that data stayed in the cache longer. The longer the data is in the cache, the more likely it is to still be in the cache later when the data needs to be consumed.

PCIE, SERDES, AND COMPRESSION

Determining what was needed for core-dominated workloads was, relatively speaking, the easy part. The harder job was working out what else was required for additional workloads.

One of the challenges that we wanted to tackle head on was designing the right combination of performance per watt, offload for datacenters, as well as enterprise storage solutions. We needed to adhere to the stringent high availability designs and dual-controller architectures of existing storage systems, which incidentally also happen to be the basic design for next-gen high end hot-storage solutions, as well as the newer object storage systems and highly scalable software designed storage (SDS) systems.

The LX2160A was designed so it could enable high performance data-path architectures where I/O connects via new high bandwidth interfaces like 25G/50G/100G Converged Network

Adapters (CNAs) from the networking side, Serial Attached SCSI (SAS) controllers for enterprise storage, or PCIe switches for NVMe SSDs. The DMA controllers were also sized to enable controller-to-controller data copies to support NTB (non-transparent bridging) functionality.

With the combination of 16 Cortex-A72 cores and the ability of hardware acceleration to compress, encrypt, and dedupe, we targeted the sweet spot of performance, power, and functionality required for newer datacenter challenges like compute offload and compute-closer-to-storage requirements like intelligent SSDs. With the increase in virtualization requirements for solid state storage, and SSD capacities growing, compute-closer-to-storage becomes more of a necessity.

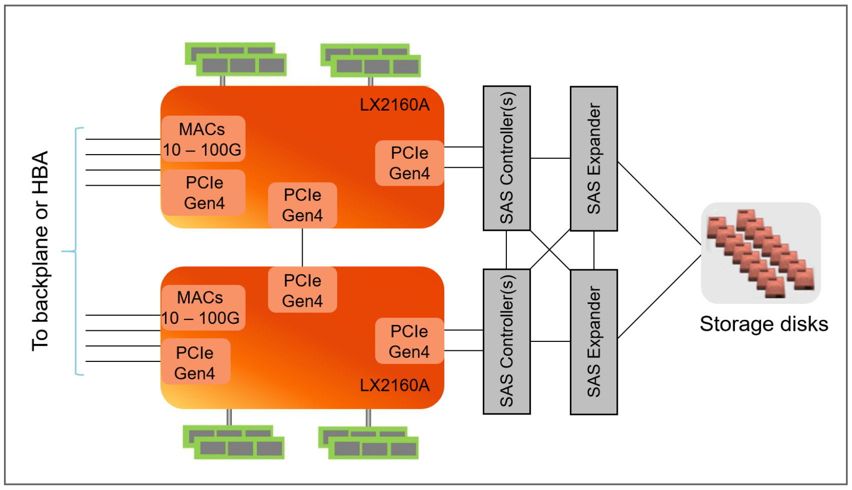

Figure 1: Storage Controller

In such applications, the LX2160A acts as a storage controller sitting between a server processor (connected through a host bus adaptor via PCIe) and the storage devices (HDD or SSD connected via a PCIe-based IO controller). Because system redundancy is important in such enterprise systems, there can be multiple copies of these storage controllers, with redundant copies of data being transmitted between the storage controllers, so that if one system fails, no data is lost. Fundamentally, there is a lot of data going into and out of the SoC through PCIe (and to some extent also Ethernet), as well as encryption/decryption, compression/decompression, and DMA copies of data from one location to another. Block sizes are relatively large (of the order of kilobytes, not bytes), and so DRAM efficiencies can be higher than for networking transactions, and the overhead to determine what to do with a packet or block of data is minimal compared to the time to transfer the data. This allowed us to focus on optimizing the LX2160A for moving large blocks of data into and out of the SoC for the PCIe Gen4 interfaces.

During LX2160A definition, we knew that PCIe Gen4 would be commonly used during the life of the LX2160A, which made its inclusion a non-negotiable feature. It offers a theoretical 16 Gb/s per lane, compared to the 8 Gb/s of the Gen3 included on the LS2088A. The LS2088A has 16 SerDes lanes shared between PCIe and Ethernet. Even with the doubled data rate, 16 SerDes lanes didn't seem enough for the new processor. Driven by the storage use case, the product target was to support eight PCIe Gen4 lanes for traffic to or from a server. This traffic also needed to ultimately go out to storage, and therefore an additional eight lanes were needed to connect the I/O controllers to the storage (HDD or SSD). Yet another eight SerDes lanes were required to connect to an identical LX2160A for redundancy. Ultimately, this meant that the LX2160A needed not just PCIe Gen4, but a total of 24 lanes at this speed.

With traffic coming in from the server, plus potentially from an identical redundant LX2160A, our design had to support a significant amount compression/decompression traffic, which drove the size of the hardware data compression and decompression engine. Although the security engine was used in the storage use case, its sizing is best explained through the DPDK IPv4 forward with IPSec use case, described next.

ETHERNET AND SECURITY

Another key use case we chose to optimize the LX2160A for was DPDK IPv4 forward with IPSec, particularly for applications such as 5G wireless network infrastructure. IPSec is very similar to the DPDK IPv4 forward use case that the LS2088A was optimized for, except that each packet is also needs encryption or decryption. As more and more private data is stored in the cloud and transferred over the Internet, security becomes even more paramount. IPSec ensures that all payload data is encrypted, preventing a rouge entity from intercepting confidential information.

Unlike the LS2088A though, this product was deliberately not optimized for complex forwarding and its up-to-five table lookups. This is because we found that in many applications, flow caching can be used so that only the first packet in a flow needs to go through the table lookups of complex forwarding. If, on average, there are many packets in the lifetime of a flow, then that flow can be cached to avoid the need to do the lookups on each packet. Because of this application refocusing, the AIOP's table-lookup acceleration was not required on this product, allowing the LX2160A design to focus more on its other key use cases.

Furthermore, whereas we designed the LS2088A to process a wide range of packet sizes, we optimized the LX2160A to achieve peak IPsec data rates with larger packet sizes. The new processor can perform IPv4 forwarding with IPsec at up to 50 Gb/s. This is 2.5 times the maximum data rate of the LS2088A. We targeted this design point because, while some network processing markets (such as those the LS2088A was optimized for) may have extended situations with smaller average packet sizes, the storage and 5G wireless network infrastructure applications LX2160A was optimized for tend to have much larger average packet sizes. Aligning the LX2160A design with different applications and keeping the packet rate lower allowed the chip to be optimized for these high data rates without overburdening it with packet processing.

From an Ethernet standpoint however, performance targets were up and to the right of the 10 Gb/s rate supported by LS2088A's MACs. 25G, 40G, 50G, and 100G Ethernet were becoming commonplace, and we needed to support them. Two Ethernet MACs were provided with support for 40G, 50G, and 100G. This maximum Ethernet speed was a 10× increase over the 10G of the LS2088A, adding potential for high bandwidth use cases that were just not possible previously. Two ports at the maximum speed are often needed in enterprise applications – for redundancy, or to support a "bump in the wire", with one Ethernet port in and another out. Sixteen MACs in total were provided on the LX2160A. This supported use cases such as 16 1G Ethernet for connection to a wide range of devices in a system (with L2 switching supported in the processor's hardware), or other use cases with up to six 25G Ethernet for applications such as datacenters where 25G Ethernet is often deployed.

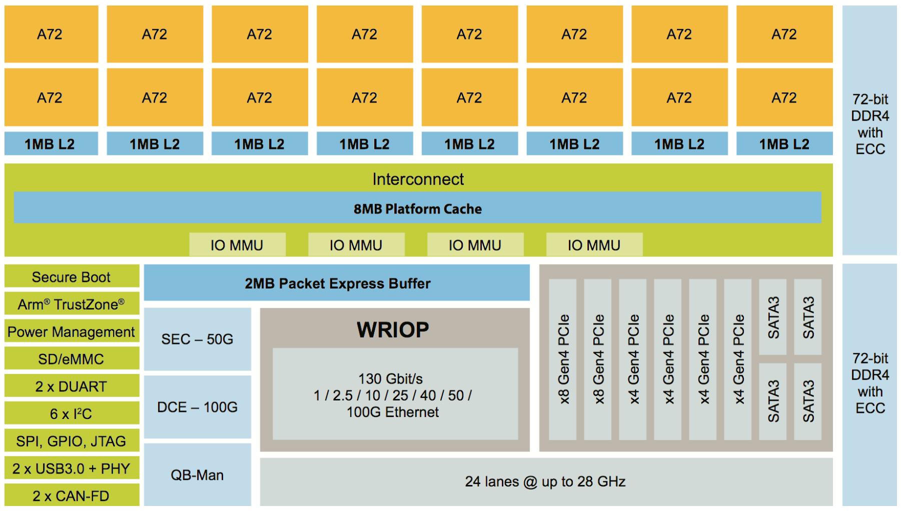

Figure 2: The final chip design

SUMMARY

Thoughtful analysis and evaluation of customer requirements resulted in the birth of the LX2160A. Compared to the LS2088A, it has double the number of cores, eight times the platform cache, 1.5× the DRAM bandwidth, three times the PCIe bandwidth, and 10 times the maximum Ethernet speed. The combined set of features does not reflect a simple "more of every-thing" approach. Instead, we thoroughly analyzed current and future use cases, and hopefully created an SoC to enable them. You can learn more about the processor at our LX2160A site.

Ben Eckermann is a Technical Director and Systems Architect for Digital Networking at NXP. Ben is currently leading systems architecture and technical requirements for QorIQ processors built on Arm technology. He has designed and architected low-power products for NXP (and formerly Freescale and Motorola) for over 15 years.