Deterministic Networking with TCP-TSN-Cores for 10/25/50/100 Gigabit Ethernet

MLE TB20201203

Summary

We all observe a growing need to connect computers with each other with shorter delays (i.e. lower latencies) and higher bandwidth, in particular for High-Performance Computing (HPC) in the Datacenter and in embedded systems such as advanced industrial robotics or autonomous vehicles. Processing of TCP/IP based network protocols at speeds of 10 Gbps and beyond demand kernel bypass solutions (such as Intel's DPDK or Solarflare's/Xilinx' Onload or Mellanox/NVida VMA) and/or so-called TOEs (TCP Offload Engines).

Domain-Specific Architectures (DSA) use so-called heterogeneous computing elements, also known as Cores with the objective to put the compute burden where it belongs. This is a well established approach going back to the early days when an x86 CPU was partnered with an x87 for better floating-point processing. Today, it is common to deploy various flavors of Cores, for example:

- DSP Cores for digital signal processing in telecommunications
- Shader Cores optimized for image processing, as they can be found in modern Graphics Processing Units (GPU)
- Tensor Processing Units (TPU) Cores which are optimized for Artificial Intelligence and Deep Learning

This is because such (special purpose) fixed-function or programmable function accelerator Cores are optimized for a particular domain and, when properly used, not only take processing load off the (general purpose) CPU but also deliver better overall performance (which is data processed per time) and better efficiency (which is performance per Watt).

Over the following pages we will make a case for processing TCP/IP over TSN over 10/25/50/100 Gigabit Ethernet on dedicated Cores which has significant advantages in particular for real-time Ethernet and Deterministic Networking. These so-called TCP-TSN-Cores can be integrated either in FPGAs or in SoCs (ASIC and ASSP). As we will show, TCP-TSN-Cores are more than just a TOE - the commonly used approach for network protocol acceleration. By running the entire network protocol stack from OSI

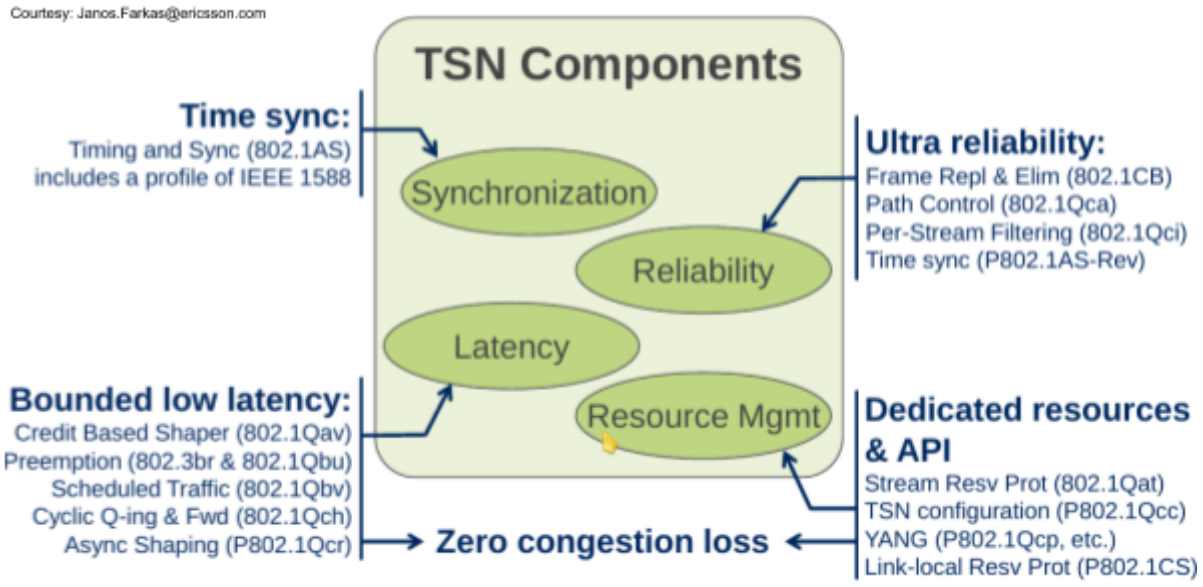
Layer 2 to at least Layer 4 in a dedicated integrated circuit - a so-called Full Accelerator - we can remove (general purpose) CPUs entirely from the datapath.

Hence, TCP-TSN-Cores can deliver very low bounded and deterministic latency with predictable scalability needed for 10/25/50/100 Gigabit Deterministic Networking.

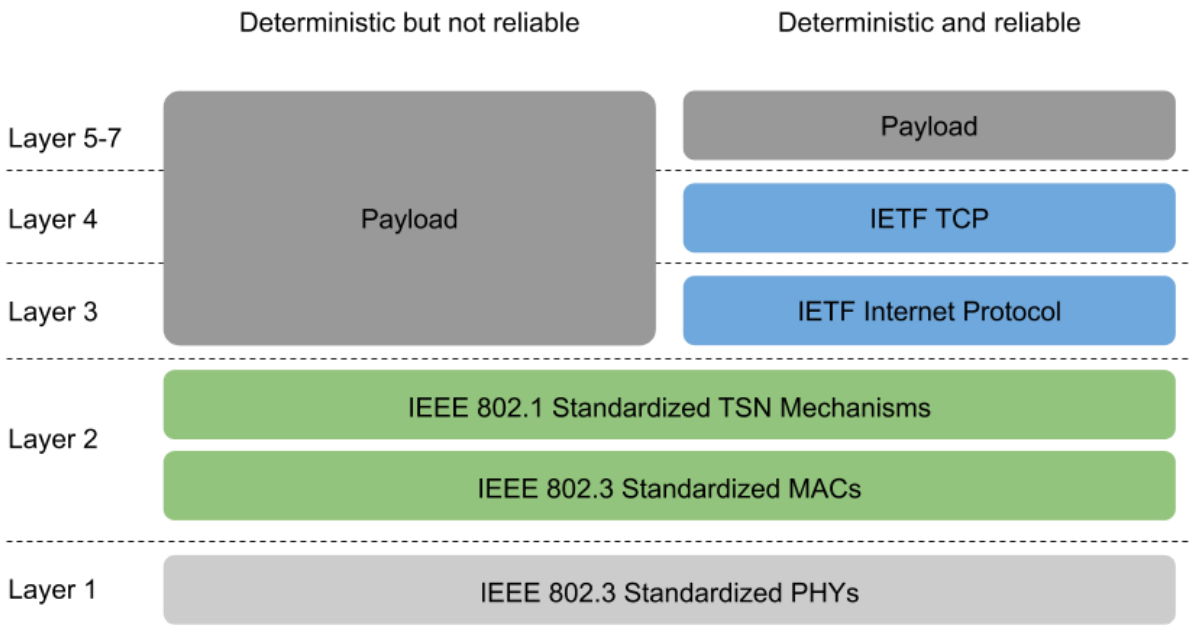
1. Deterministic Networking

Deterministic Networking (DetNet), according to RFC8655 from the IETF, combines the Internet Protocol (IP) with Time-Sensitive Networking (TSN). Applications are Data Center Local Area Network (LAN) as well as (Embedded) Systems-of-Systems used in Industrial Robotic Systems or Autonomous Vehicles or modern Automotive Zone-Based Architectures.

TSN started as AVB (Audio-Video Broadcasting), a networking initiative aimed to improve audio/video transports by better management of network latencies. Today, TSN has become a set of emerging, open IEEE standards with momentum in industrial markets (for 10/100/1000 Mbps speed) and in next-generation Automotive Zone architectures (for 10/25/50 Gbps speeds). Aspects such as Time-Aware Traffic Shaping also find application in telecommunication, Provider Back-Bone (PBB) Switching or Software-Define Wide Area Networks (SD-WAN), for example.



TCP is the Transmission Control Protocol specified by the IETF (RFC793). Together with UDP (User Datagram Protocol - RFC768) it belongs to the well-known, well understood, ubiquitous, easy to deploy and to maintain transport protocols the Internet depends on. Unlike UDP, TCP is considered a reliable protocol because of the provisions to re-transmit packets lost in transport. TSN and TCP can be combined according to the OSI Layers:

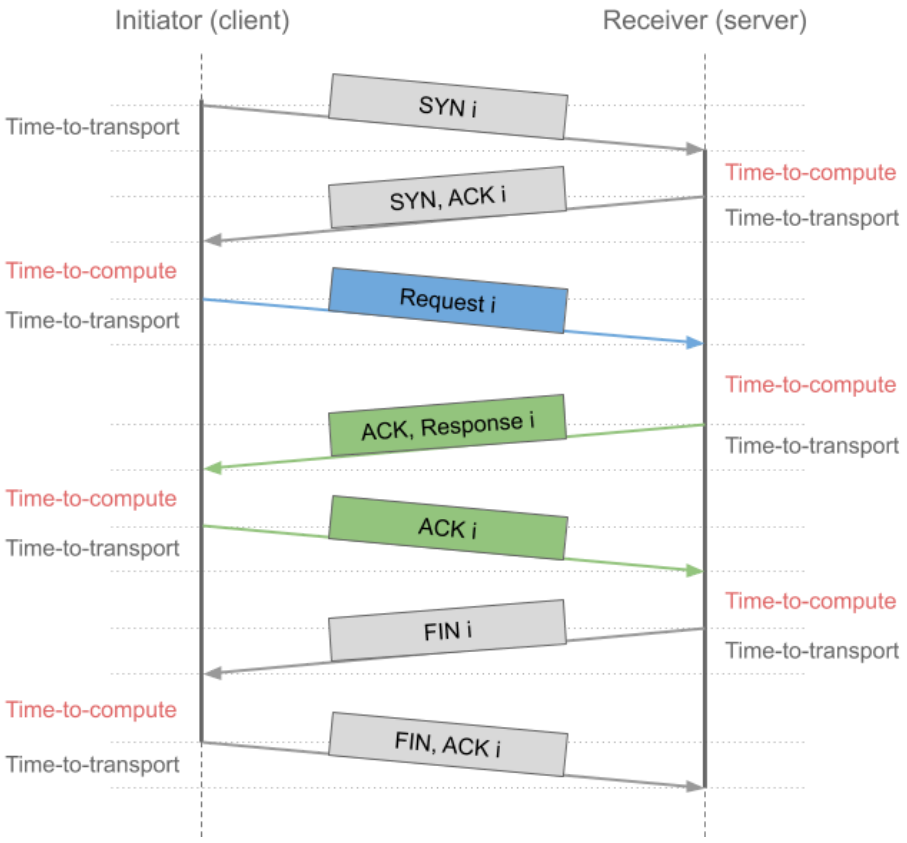


The outcome is a deterministic and reliable network protocol, which makes TCP/IP over TSN a very good candidate for all networking where IT (Information Technology) and OT (Operations Technology) converge, or in Systems-of-Systems backbones such as Automotive Zone Architectures.

The key challenge, however, lies in the computational burden when running Deterministic Networking in software on a (general purpose) CPU, as the anatomy of a TCP connection (sometimes referred to as a TCP socket) shows.

2. Anatomy of TCP/IP Processing

As Ethernet line rates continue to increase, the bottlenecks in network protocol processing software become more visible. Basically, the time-to-transport has improved much more than the time-to-compute, as the following Figure shows:



For 10 Mbps Ethernet the time-to-transport for example a TCP SYN packet (40 Bytes) took 32 microseconds which, time-wise, was equivalent to 3200 instructions on a (then) 100 MIPS CPU. For 100 Gbps Ethernet the time-to-transport for the same TCP SYN packet has shortened to 3.2 nanoseconds, or the equivalent of 10 instructions on a 3000 MIPS CPU - assuming best-case with no cache misses. This disparity between the time-to-transport and the time-to-compute in combination with the time required for the initial handshake for TCP is one of the reasons why applications are looking at UDP-based protocols such as IETF QUIC.

However, many (not all but more than a few) applications require a reliable transport, industrial and automotive networking for example, and will benefit from Deterministic Networking run on dedicated TCP-TSN-Cores.

3. TCP-TSN-Core Architecture

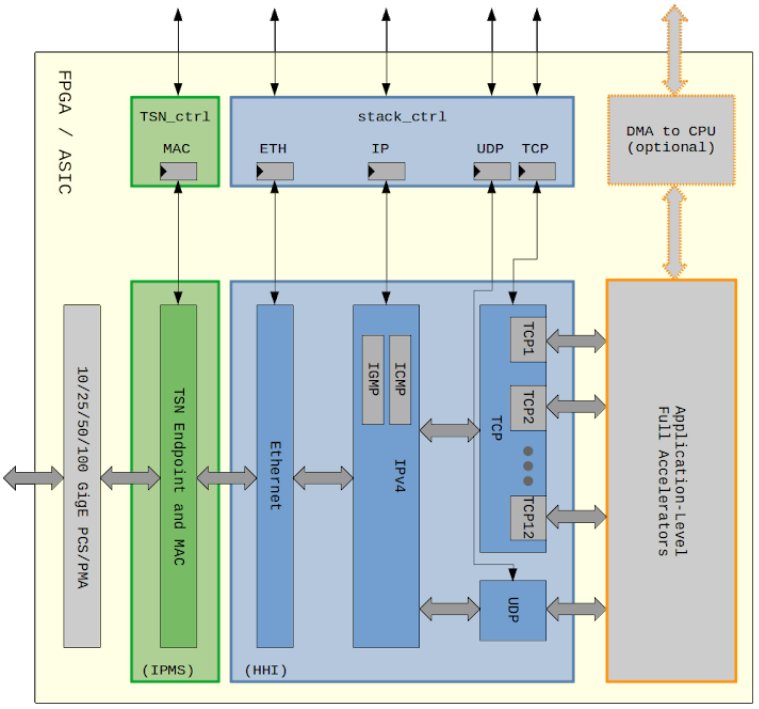
The use of Full Accelerators for TCP/UDP/IP processing, a key ingredient in TCP-TSN-Cores, goes back to 2010 when the Fraunhofer Heinrich-Hertz Institute (HHI) in

Berlin, Germany, started implementing Network Protocol Accelerator Platforms (NPAP) in ASIC and FPGA. Unlike in TOEs where significant processing of TCP still happens in adjacent CPUs (with or without kernel bypasses), Full Accelerators run the entire TCP stack in a dedicated digital circuit.

3.1. TCP/UDP/IP Full Accelerators

The key advantage of TCP-TSN-Cores is the high level of determinism because processing throughput can be predicted at nano-second accuracy as the Full Accelerator circuit typically runs at multiple 100s of MHz clock speed and needs very little internal FIFO buffers.

Such a TCP-TSN-Core is shown in the following block diagram:



The datapath goes horizontally, with ingress and egress 10/25/50/100 Gigabit Ethernet on the left into/out of dedicated FPGA/ASIC hardware blocks:

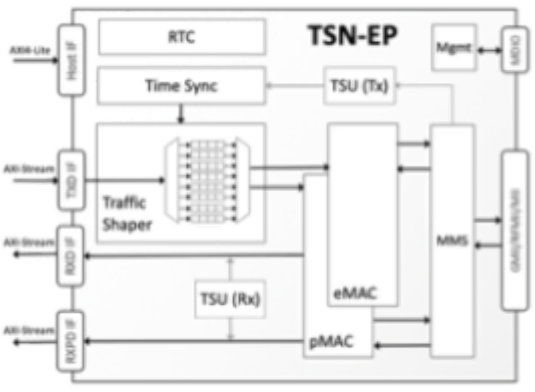
- TSN and Ethernet Media Access Controller (MAC)
- Ethernet management (including ARP)
- IPv4 (including ICMP for network management and diagnostics plus IGMP for group management)

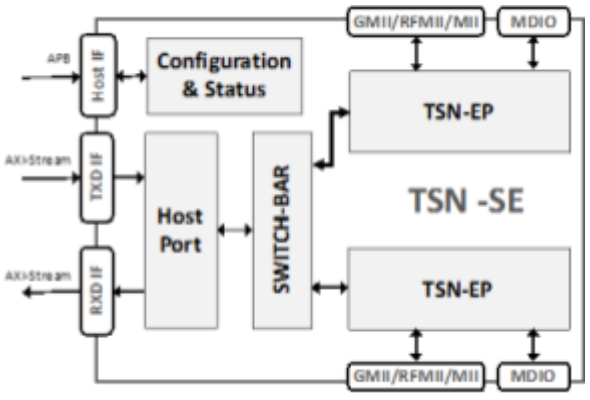
- one UDP block (taking care of all UDP packets)
- one TCP block per TCP connection (picture shows 12)

Connectivity between these hardware blocks is via bi-directional 128-bit wide AXI4-Streams, which can scale to 50 Gbps line-rate in FPGA and 100 Gbps line-rate in ASIC. The control flow is shown here vertically with means to set MAC addresses, IP addresses, plus a command interface to manage (open, configure and close) TCP connections. Control flow including TSN configuration can be implemented via hardware state machines or via software. On-chip Full Accelerators (bottom right in the block diagram) can either implement application-level data processing, or data can be sent to and/or received from adjacent CPUs, SoC-style with integrated CPUs or via PCIe-connected Host CPUs.

3.2. TSN Subsystem

A typical TSN subsystem is hardware blocks plus software for system configuration. The following shows an exemplary implementation of such a hardware block from Fraunhofer IPMS, Germany, which shows the TSN Endpoint (which handles Time Synchronization, does Traffic Shaping, etc) as well as a so-called TSN Switched Endpoint with two ports. TSN Switched Endpoints allow daisy-chaining and facilitate redundancy needed for Functional Safety or High Availability.





Real-time configuration needs little compute performance and runs in software either on the Host CPU (which can be PCIe-connected) or SoC-style on an embedded CPU. Unlike in a typical CPU/software implementation where the “upstream” port of the TSN Switched Endpoint connects to a DMA engine, in a TCP-TSN-Core this is connected to

the TCP/IP Full Accelerator (which, obviously, must be aware of multiple streams and shall support traffic preshaping).

4. Architectures with TCP-TSN-Cores

The key to implementing cost-efficient architectures with TCP-TSN-Cores is to complement both approaches:

1. Processing in hardware on dedicated TCP-TSN-Cores - which has the advantage that TCP/IP processing can happen at line rate. The downside is that TCP-TSN-Cores do require precious silicon real estate so we are limited by the number of parallel open TCP connections at any given time. And, you can only process those protocols for which resource efficient Application-Layer Full Accelerators exist.
2. Processing in software on (general purpose) CPUs - which can be in the operating system (Linux) kernel with or without a “Bypass” such as DPDK, for example. The strength is that you can have millions of open TCP connections at any time, and that software covers all application-layer protocol functionalities needed. The downside is much higher latency than running TCP/IP on dedicated TCP-Cores.

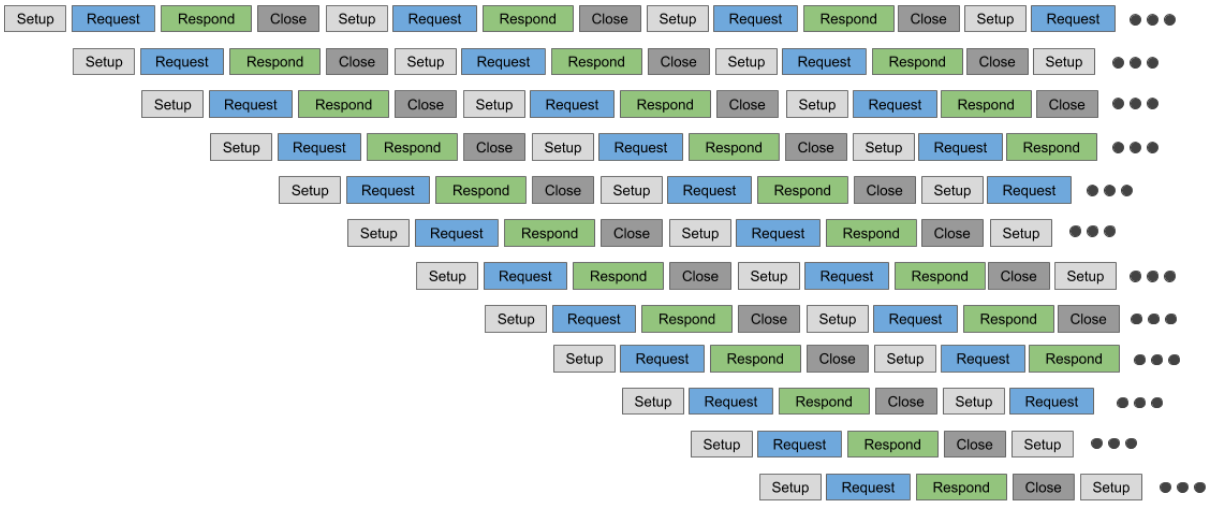
While most of the hardware blocks can be shared among a single Ethernet port, the Full Accelerator approach from Fraunhofer HHI requires multiple TCP blocks. To process the state information of a TCP connection a dedicated TCP block has to be available in hardware. Obviously, a single TCP block can process many TCP connections sequentially, as soon as the prior TCP connection is closed, which leads us to the concept of a pipelining architecture.

4.1. Pipelining with TCP-TSN-Cores

TCP-TSN-Cores need more hardware resources than a TOE which made quite a difference in 2010. However, today most applicable FPGAs such as Stratix-10 or Agilex from Intel or Virtex UltraScale+ or Versal Prime from Xilinx have plenty of resources and allow implementing TCP-TSN-Cores which can process dozens of TCP connections at the same time (as a rule of thumb, an extra TCP block needs approx. 10k ALMs in Intel

Stratix-10 or 10k LUTs in Xilinx UltraScale+). Hence, you can process dozens of TCP connections open at the same time, but not millions.

To make TCP-TSN-Cores resource efficient it is important to “time-share” the TCP blocks in the TCP-TSN-Cores: Once a TCP connection has been closed, the corresponding TCP block can serve another TCP connection. This is shown in the following Figure with a TCP-TCN-Core which features a total of 12 instances of TCP blocks:



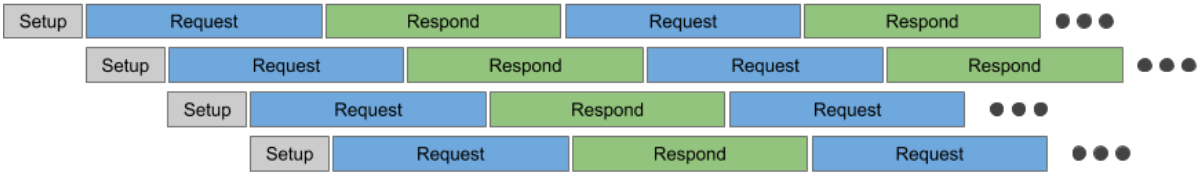
Implementing such a pipelining approach is facilitated by yet another advantage of TCP-TSN-Cores: Opening, processing and closing takes significantly less time compared to software, if the network is fast, and if the counterpart is also fast. Best is to have similar TCP-TSN-Cores on either side of the TCP connection.

Fast processing of many very short TCP connections is needed in many Datacenter applications, for example in database caching such as Memcached or other Key-Value Stores including NoSQL databases.

4.2. Stream Processing with TCP-TSN-Cores

Other sweet-spots for TCP-TSN-Cores are network applications which require few long-lasting and very compute intense TCP connections. Examples are distributed data-in-motion processing as we see this in video streaming applications or in networked computational storage (CSx). This is referred to as Stream Processing and for such applications you combine both approaches.

This can be visualized by the following Figure which shows a TCP-TSN-Core with 4 instances of a TCP block processing two separate compute intensive applications.



Upfront, you select those network protocols which shall run in hardware on dedicated TCP-TSN-Cores. TCP-TSN-Cores work best when processing needs to happen at line-rate and for protocols for which resource efficient Application-Layer Full Accelerators exist. This allows to keep the “heavy traffic” away from the Host CPUs. When computational storage using NVMe SSDs is involved, PCIe Peer-to-Peer between the TCP-TSN-Core and the NVMe SSD can further improve the overall system performance and efficiency.

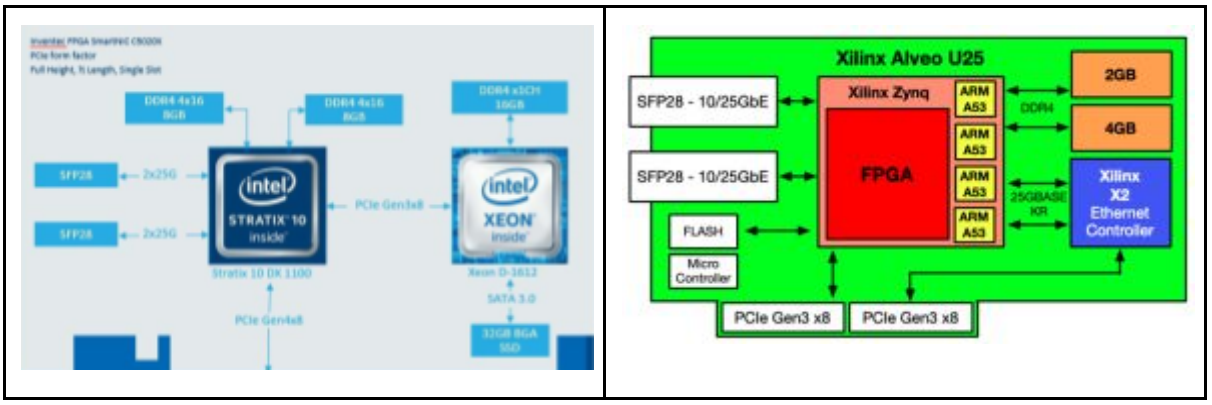
Obviously, all other network traffic runs in software (with or without kernel bypass, with or without TOEs) as we would do without any TCP-TSN-Cores.

While this allows you to support millions of open TCP connections at any time, and to support all application-layer protocols, such an architecture which assigns certain protocols to TCP-TSN-Cores and others to software may be a bit too static. Which leads us to the concept of Hybrid Acceleration.

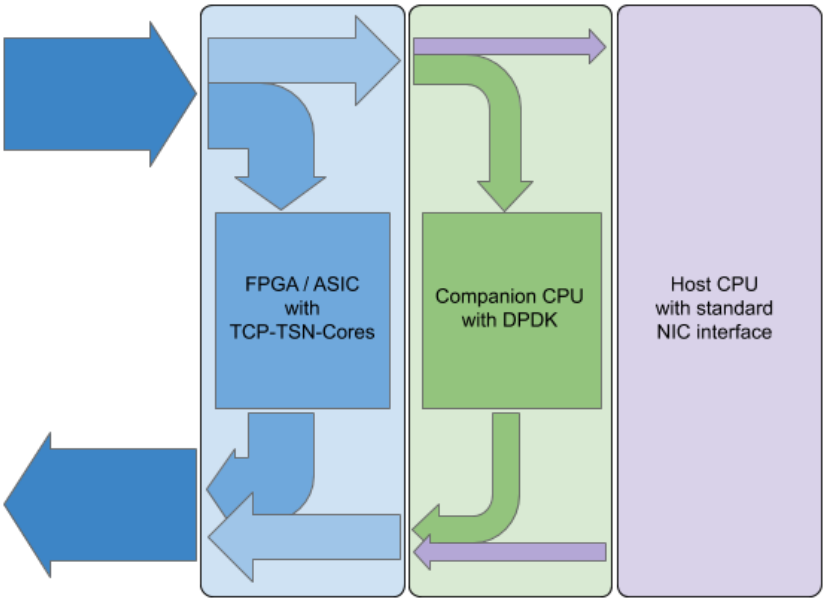
4.3. Hybrid Acceleration with TCP-TSN-Cores

Hybrid Acceleration is a concept where TCP/UDP/IP processing can either be run in software on a (general purpose) CPU or in hardware on TCP-TSN-Cores. For UDP streams this is relatively easy. But TCP keeps state information for each connection (including the TCP session quintuple), thus Hybrid Acceleration must be able to “swap” this TCP state information between the software stack and the corresponding TCP-Core. TCP-TSN-Cores do support storing and loading TCP states in general. The opacity of DPDK makes integrating such a mechanism for “swapping TCP states” feasible.

Furthermore, today there are some interesting choices for FPGA-based SmartNICs which can implement such a Hybrid Acceleration, and which can effectively deliver a multi-stage processing approach for network protocol acceleration. Examples are Inventec’s C5020X or the Alveo U25 SmartNIC from Xilinx.



When using such FPGA-Based SmartNICs, network protocol processing can be performed on three stages using three different cores, as the following Figures shows:



Here, all network traffic enters (and leaves) the system via a powerful FPGA. Within the FPGA TCP-TSN-Cores act as a first processing stage and immediately take care of certain “heavy” network traffic or of the “real-time” data. This can be done using pipelined architectures of TCP-TSN-Cores or by Stream Processing. In a second processing stage, which complements the TCP-TSN-Cores, a companion CPU runs an optimized TCP/IP stack such as DPDK. Hybrid Acceleration allows to swap TCP states between the first two stages. Finally, all network traffic that has not been handled by those two stages is sent to the Host CPU via PCIe. For this PCIe drivers basically implement a Network Interface Card (NIC) function.

Having this companion CPU in the SmartNic provides several advantages:

- Network traffic can be handled very predictably because the companion CPU never has to process user applications with “unexpected” load
- The software stack of the Host CPU becomes independent of “special” network functionality such as kernel bypasses
- System maintenance gets easier as the software for the companion CPU can be optimized and hardened on its own
- System security is improved as the network stack for the SmartNIC can additionally be hardened
- Hardware security features can be deployed without having to make the Host system trusted

5. Conclusion and Backgrounder

TCP-TSN-Cores integrate TSN Switched Endpoints for Time Sensitive Networking functionality with Full Accelerators for TCP/UDP/IP processing in hardware. The outcome delivers not only very deterministic transport at bounded, low-latency, but also very high throughput that scales with 10/25/50/100 Gigabit Ethernet line-rates.

When TCP-TSN-Cores are complemented with state-of-the-art kernel bypass (DPDK, for example) and/or TCP Offload Engines (TOE) they enable Hybrid Acceleration for all relevant network protocols. This takes heavy and/or latency-sensitive network traffic off the Host CPU.

TSN is Time Sensitive Networking, an IP Core for FPGA/ASIC from Fraunhofer Institute for Photonic Microsystems (IMPS). Fraunhofer IPMS is a worldwide leader in research and development services for electronic and photonic microsystems in the fields of Smart Industrial Solutions, Medical & Health applications and Improved Quality of Life and is located in Dresden, Germany.

NPAP is the Network Protocol Accelerator Platform, an IP Core for FPGA/ASIC from Fraunhofer Heinrich-Hertz Institute (HHI). Fraunhofer HHI focuses on 10 to 100 Gbit transmission in the field of high-performance telecom components and on mobile broadband systems. Fraunhofer HHI is located in Berlin, Germany.

MLE (Missing Link Electronics) is offering technologies and solutions for Domain-Specific Architectures, which focus on heterogeneous computing using FPGAs. MLE is headquartered in Silicon Valley with offices in Neu-Ulm, Germany.

Authors and Contact Information

Ulrich Langenbach, Dir. Engineering, Missing Link Electronics GmbH
Endric Schubert, PhD, CTO, Missing Link Electronics, Inc.

Missing Link Electronics, Inc.
2880 Zanker Road, Suite 203
San Jose, CA 95134, USA
+1-408-475-1490

Missing Link Electronics GmbH
Industriestrasse 10
89231 Neu-Ulm
Germany

www.missinglinkelectronics.com