

MLE Technical Brief 20251111

Network Protocol Accelerator Platform

**A stand-alone TCP/UDP/IP Stack Full-Accelerator Subsystem
allowing communication at full line rate and low latency**

Version: 2.8.0
Date: November 2025
Authors: Ulrich Langenbach, Thomas Glatte, F. Maximilian Sokol

1. Features	1
2. Applications	2
3. IP Core Description	3
3.1. Technical Features	3
3.2. Implementation Details	5
3.3. IP Core Deliverables	8
3.4. Network Statistics and Diagnostics	9
3.5. Network Impairment and Bit Error Insertion	13
3.6. Quality Assurance	14
4. TCP/UDP/IP Architecture Aspects and Choices	15
4.1. Picking the “Right” TCP Peer	15
4.2. Picking the Right TCP Rx/Tx Buffer Sizes	16
4.3. Picking the Right Number of TCP Cores	17
4.4. Optimizing NPAP for Linerate Performance	17
4.5. Combining NPAP With FPGA Network Interface Cards (NIC)	18
4.6. Implementing Time-Sensitive Networking (TSN)	19
4.7. Linux Kernel Bypass With NPAP	20
4.8. Latency Analysis Results	21
4.9. Optional Transport Layer Security (TLS)	22
5. NPAP Evaluation Choices	23
5.1. Available Evaluation Reference Designs	23
5.2. Evaluation Reference Design (ERD) Structure	25
6. Resource Estimates for ASIC and FPGA	27
6.1. Resource Estimates for AMD Versal AI Edge Series	27
6.2. Resource Estimates for AMD/Xilinx Ultrascale+ Series	28
6.3. Resource Estimates for AMD/Xilinx 7-Series	29
6.4. Resource Estimates for Altera/Intel Agilex 5E	30
6.5. Resource Estimates for Altera/Intel Stratix-10	30
6.6. Resource Estimates for Lattice Avant-G	32
6.7. Resource Estimates for Microchip Polarfire	33
7. Detailed Protocol Support (RFC1122 excerpt)	34
7.1. Ethernet Layer	34
7.2. IP & ICMP Layer	34
7.3. TCP Layer	36
8. Developer Documentation	38
9. Changelog	55
9.1. NPAP Version 2 Development	55
9.2. NPAP Version 1 Development	66
MLE Contact Info	71

1. Features

MLE's Network Protocol Accelerator Platform (NPAP) for 1/2.5/5/10/25/40/50/100 Gigabit Ethernet is a TCP/UDP/IP network protocol Full-Accelerator subsystem which instantiates the standalone 128 bit TCP/IP Stack technology from German Fraunhofer Heinrich-Hertz-Institute (HHI). This Fraunhofer HHI TCP/IP Stack was designed for embeddable FPGA and ASIC system solutions and offers the following features:

- Interface to 1 / 2.5 / 5 / 10 / 25 / 40 / 50 / 100 Gigabit Ethernet¹
- Full-duplex with 128 bit wide bidirectional datapath
- Full line rate up to 70 Gbps per NPAP instance
- Full line rate >100 Gbps per individual TCP session in ASIC
- Low round trip time NPAP-to-NPAP 700 nanoseconds for 100 Bytes RTT
- TCP session priority management

Designed for maximum flexibility, NPAP implements in programmable logic the most common network communication protocols:

IPv4 The core of the most standards-based networking protocols

TCP Reliable connectivity for direct secured connectivity

UDP Widespread protocol to enable simple direct or multicast communication

ICMP Diagnostic protocol to validate connections

IGMP Enables joining of multicast groups (optional)

Due to the modularity NPAP can easily be enhanced by application specific protocols.

Originally targeted to deliver close to the theoretical line-rate of 10 Gigabit Ethernet, a 128 bit wide datapath in combination with a pipelined architecture allows to scale throughput to line-rates of 50 GbE, and beyond, when using modern FPGA fabric, and up to 100 Gbps for ASIC implementations.

NPAP is available in versions which recently have been merged:

- Version 2 (currently 2.8.0) for new ASIC and AMD/Xilinx Versal, Ultrascale+, Ultrascale, Altera/Intel Agilex-7, Agilex-5E and Stratix-10, Microchip PolarFire and Lattice Avant G/X development, includes many recent resource optimizations plus timing optimizations focused on pipelines in FPGA, including Altera/Intel HyperFlex and Intel HyperFlex2
- Version 1 (currently 1.10.1) back-ports bug fixes from Version 2 for long-term customer support. **Not recommended for new design starts!**

¹ Obviously depending on the FPGA device and speedgrade.

2. Applications

NPAP enhances your real-time application with a leading fast data connectivity. The powerful architecture of the underlying TCP/UDP/IP Stack allows it to transfer data at line-rate with low processing latency without using any CPUs in the data path. The widespread TCP/IP and/or UDP/IP communication protocol suite using industry standard network infrastructure addresses a wide-range of applications:

- FPGA-based SmartNICs
- High-Bandwidth Security with FPGA-based Smart Data Diodes
- In-Network Compute Acceleration (INCA)
- Hardware-only implementation of TCP/IP in FPGA
- PCIe Long Range Extension ²
- Networked storage, such as iSCSI or NVMe/TCP
- Test & Measurement connectivity
- Automotive backbone connectivity based on open standards
- High-speed, low-latency camera interfaces
- Video-over-IP for 3G / 6G / 12G transports
- Bring full TCP/UDP/IP connectivity to FPGAs
- High-speed sensor data acquisition:
stream data out of FPGAs into Network-Attached Storage (NAS)
- High-speed robotics control and machine-to-machine:
Stream data from servers via FPGA into actuators
- Hyper-converged computational storage acceleration for “over-Fabric”
NVMe/TCP
- Deterministic low-latency, high-bandwidth alternative to lwIP or Linux on
embedded CPU

²

<https://www.missinglinkelectronics.com/index.php/menu-products/menu-pcie-connectivity/439-art-pcie-over-xxx>

3. IP Core Description

High performance programmable logic based, standalone TCP/IP stack featuring transparent handling of complete TCP/IP and UDP protocol tasks, e.g. packet encoding, packet decoding, acknowledge generation, link supervision, timeout detection, retransmissions and fault recovery. Complete automatic connection control including tear up and tear down. Compute and manage retransmission timers as in RFC 6298³. Transparent checksum generation and checksum checking, integrated flow control. RFC 9293⁴ compatibility (TCP/IP stack for Windows and Linux). Depending on the project's needs, deliverables can be:

- HDL source code or netlist
- Integrated FPGA system implementation
- Testbenches and scripts for real-life testing
- Comprehensive documentation and interfacing guide
- Development & design-in support

NPAP is optimized to ensure the best bandwidth-delay product performance for your application. The IP core described herein is easy to port to FPGA and ASIC target platforms.

3.1. Technical Features

Feature	Specification
Supported on-chip Interfaces	128 bit wide AXI4-Stream
Compatibility with 3rd party Ethernet PHY interfaces	Standard IEEE Ethernet PHYs with RMII, GMII, XGMII, etc via PCS/PMA via ASIC/FPGA Ethernet Subsystem
Compatibility with 3rd party Ethernet Media Access Controllers	Fraunhofer HHI 10G/25G Low-Latency MAC AMD/Xilinx 10G/25G Ethernet Subsystem (PG210) AMD/Xilinx 100G Ethernet Subsystem (PG165) Intel 10G / 25G Ethernet FPGA IP Microchip PolarFire FPGA 10G Ethernet (UG0727) Lattice 10G Ethernet IP (FPGA-IPUG-02245)
Supported protocols (Hardware based)	Ethernet, ARP, IPv4, ICMPv4, IGMPv4, UDP & TCP, DHCP
Number of simultaneous connections	One per TCP Core instantiation - see "Architecture Choices" below, a TCP Core in NPAP relates to a TCP socket in Linux

³ <https://datatracker.ietf.org/doc/html/rfc6298>

⁴ <https://www.rfc-editor.org/info/rfc9293>

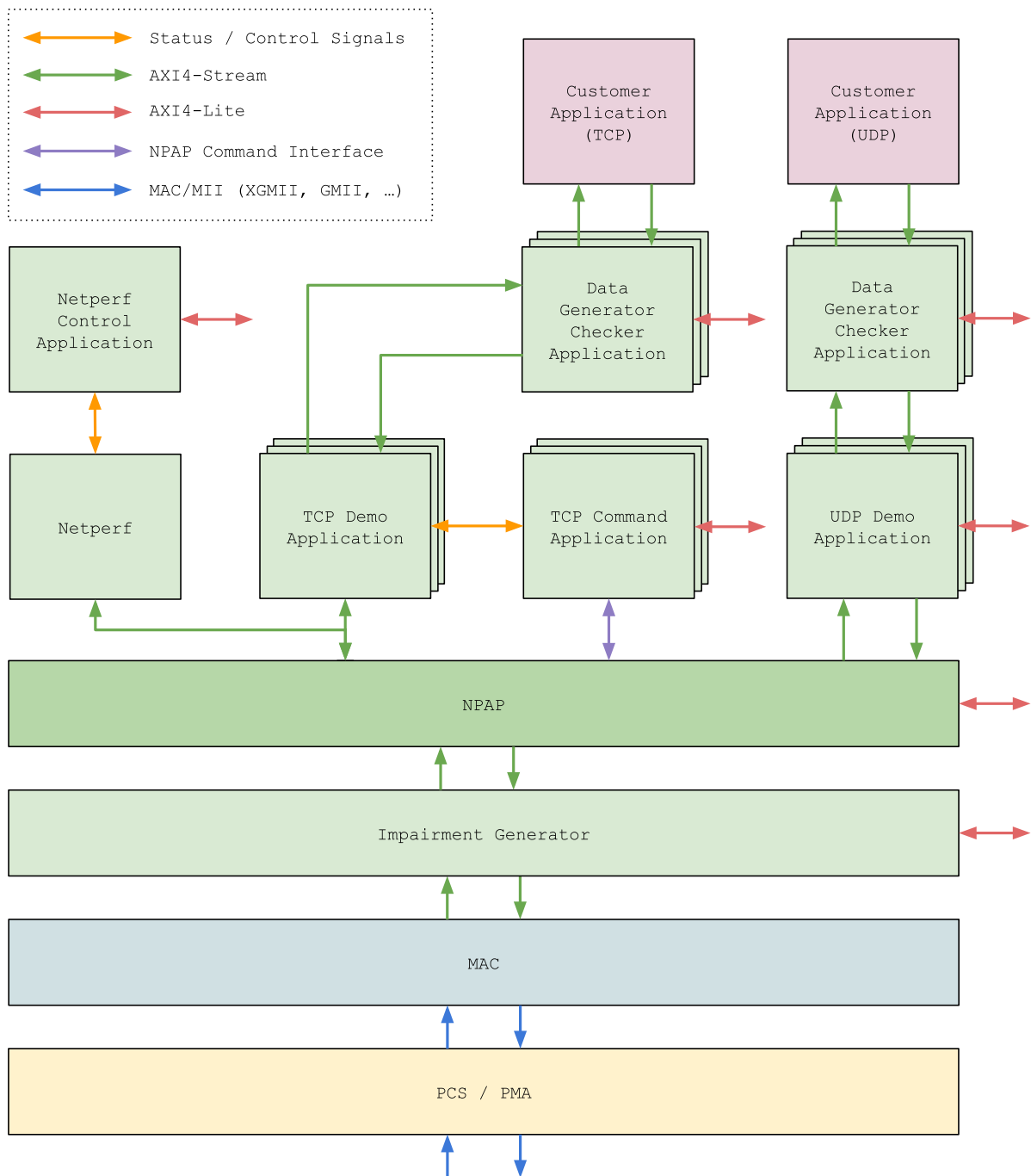
Interface to application	Datapath via AXI4-Stream 128-bit and separate custom TCP command interface
Supported FPGAs	Complete stack uses generic VHDL code (IEEE-1076 2002 or 2008, depending on NPAP version) AMD/Xilinx Virtex 4 to Virtex UltraScale+ AMD/Xilinx Kintex to Kintex UltraScale+ AMD/Xilinx Artix UltraScale+ AMD/Xilinx Zynq-7000 AMD/Xilinx Zynq UltraScale+ MPSoC AMD/Xilinx Zynq UltraScale+ RFSoc AMD/Xilinx Versal ACAP Series Altera Cyclone IV series Altera Cyclone 10 GX series Altera Stratix V Altera Stratix 10 GX series Altera Agilex 5 D, E Series Altera Agilex 7 F, I, M Series Lattice Avant G, X¹ Microchip Polarfire and PolarFire SoC
Performance	Over 60 Gbps line rate for single TCP/IP session (depending on clock rate, see below) Less than 800 ns RTT (depending on clock rate and packet size)
RTL Verification	A full RTL simulation environment, including support for Wireshark PCAP stimulus files can be made available upon special request
Diagnostics & Debug	RTL code plus instrumentations for extensive statistics and diagnostics can be made available upon special request Optional functionality for Network Impairment (a.k.a. Bit Error Insertion) can be made available upon special request

3.2. Implementation Details

NPAP implements a full accelerator, hence all network protocol processing is running as digital logic. NPAP does not instantiate any “soft” CPUs, nor does NPAP rely on external CPUs to fully function. This gives NPAP a very deterministic low latency and makes performance very scalable over circuit area / FPGA resources.

- The (horizontal) datapath is full duplex 128 bits wide using AXI4 Stream
- The (vertical) control path is AXI4 Lite register interfaces with HDL wrappers
- NPAP brings its own 10G/25G Low-Latency Ethernet MAC, but can also interface with Ethernet subsystems from the FPGA vendors
- NPAP implements a complete TCP/UDP/IPv4 stack including functions like ARP, ICMP, IGMP, DHCP
- NPAP is delivered with “Support IP blocks” including design examples for setting MAC addresses and/or IP addresses and/or TCP port numbers either from Programmable Logic / ASIC or via software running on a (Linux) host, either ARM or x86 based
- For each TCP connection that remains open at the same time, there shall be one instance of a TCP Core
- An AXI4-Lite interface may be used to prioritize TCP sessions during runtime
- One and only one single instance of a UDP Core must be instantiated when UDP support is required. If there is no need to process UDP, then the UDP Core can be removed completely
- NPAP is highly parameterizable
 - Number of UDP Cores (zero or one)
 - Number of TCP Cores (zero or many)
 - For each TCP Core: Rx Buffer Size and, separately, Tx Buffer Size
- Other than Rx and Tx buffers and some buffers for clock domain crossings, NPAP hardly uses any buffers at all, which results in very low and deterministic latency
- Your “Layer 7” Application can directly be connected to NPAP via AXI4 Stream which gives you the option of keeping all traffic inside the Programmable Logic / ASIC, and/or to interface with “software” running on (Linux) host, either ARM based or x86 based, via DMA

The following shows the high-level block diagram of a highly modular implementation using standard AXI4 interfaces to support RTL synthesis flows for various FPGA vendors and ASIC RTL design flows.

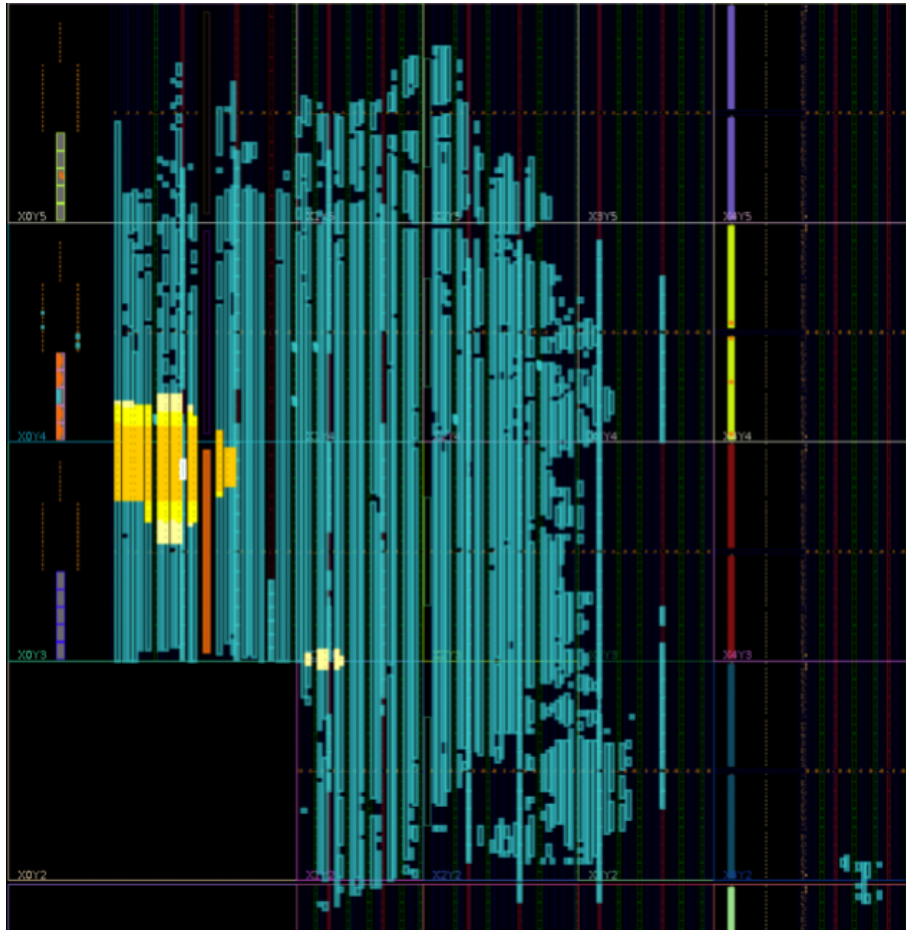


For instantiating NPAP you have two options:

- **Asynchronous clocking** - where the Ethernet MAC block is in a different clock domain than the other functions (IPv4, TCP, UDP block).
- **Synchronous clocking** - where the Ethernet MAC block is within the very same clock domain as the other functions (IPv4, TCP, UDP block).

Both are equally good, but depending on your overall design, the device utilization, your performance requirements, etc., one may be better than the other. Just giving you choices...

Starting with NPAP 2.0, RTL is optimized for FPGA pipelining such as Altera/Intel Hyperflex or AMD/Xilinx IMux. Starting with NPAP 2.0, RTL has further been optimized for Altera/Intel HyperFlex and Altera/Intel HyperFlex2.



3.3. IP Core Deliverables

Deliverables include IEEE 1685 IP-XACT packages which include NPAP Kernel plus NPAP Support IP blocks plus non-IP-XACT FPGA reference design projects. The following design blocks are part of a typical delivery package:

- Low-Latency Ethernet MAC for 10G/25G
- NPAP Kernel with IPv4, TCP, UDP
- NPAP Support IP
 - Data Generator Checker
(great for system-level testing and for performance tuning)
 - NPAP Control Application
 - TCP Command Application
 - TCP Demo Application
 - UDP Demo Application
 - Netperf Control Application (optional)
 - Network Statistics and Diagnostics (optional)
 - Network Impairment and Bit Error Insertion (optional)
- Evaluation Reference Design (as FPGA Design Project Archive)

NPAP is available as VHDL RTL source code or as a netlist suitable for synthesis by 3rd party and FPGA vendor tools:

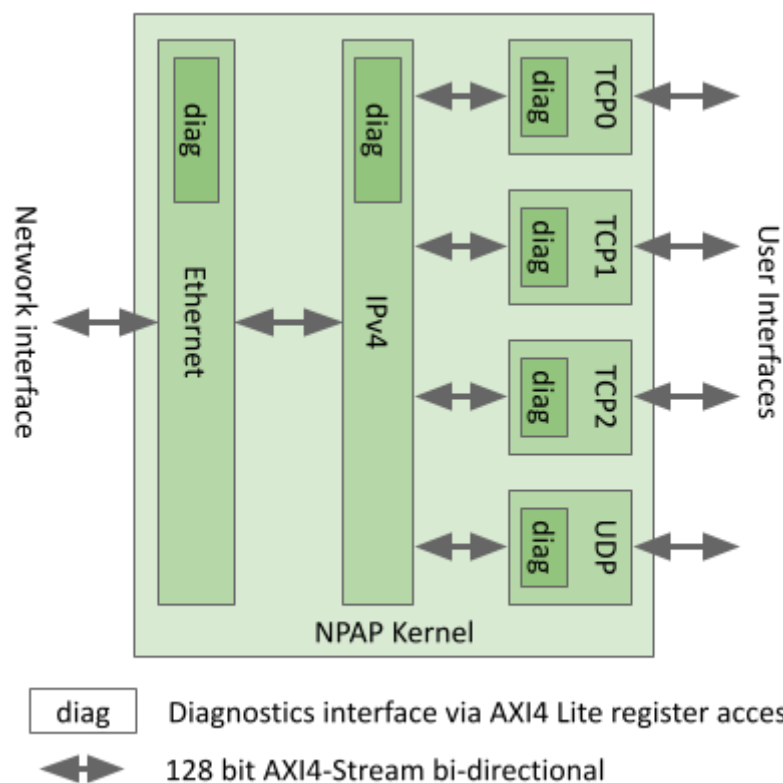
Version	AMD/Xilinx	Altera/Intel	Lattice	Microchip
NPAP 2.5.0	Vivado 2022.2	Quartus Prime Pro 22.4	Radiant 2024.2.1	Libero 2023.1
NPAP 1.10.1	Vivado 2018.3	Quartus Prime Pro 22.1	n/a	Libero 2021.2

Starting with version 2.5.0, NPAP is delivered as VHDL IEEE-1076 2008 RTL source code. This standard is fully supported by all modern FPGA toolchains, including those listed in the table above. However, some older toolchains, such as Intel's Quartus Prime Standard (required for legacy devices like Stratix V), do not offer full VHDL-2008 support. For these specific tools, the latest compatible release is NPAP v2.4.5.

3.4. Network Statistics and Diagnostics

In particular at faster line rates, diagnosing network protocol issues can be time-consuming and challenging. MLE NPAP therefore comes with specialized diagnostics blocks, providing a powerful suite of tools for in-depth network analysis and efficient troubleshooting of network behavior. These diagnostics offer TCP/UDP/IP layer-specific visibility of counters and status information similar to those in Linux or Windows, for example:

- **TCP Core Activity:** Track performance and network usage of individual cores.
- **TCP Peer Balance:** Monitor data flow and identify peer misconfigurations.
- **Buffer Optimization:** Fine-tune buffer settings using diagnostic data and RTT.
- **General Troubleshooting:** Track packets, errors, and pinpoint issue sources.



MLE NPAP's diagnostics blocks are accessed using AXI4-Lite and enable efficient diagnosis and a faster path towards resolving network problems when using hardware acceleration.

3.4.1. Diagnostics Data Acquisition Modes

The MLE NPAP diagnostics system supports multiple data acquisition modes to cater to different debugging and analysis needs. These modes define how the underlying counter and event state registers are updated. The selected mode applies universally

to all readable counter and event state registers within the respective diagnostics block:

Continuous Mode



Registers the current state of events and counters every clock cycle. This mode provides the most immediate, real-time data feedback.

Use Case: Ideal for live monitoring of instantaneous activity and quick verification of data paths.

Note: Reading multiple diagnostics registers in quick succession may lead to incoherent data, as the values reflect different, continuous

points in time.

Snapshot Mode



Takes an instantaneous, synchronous snapshot of all counter and event states upon a user-triggered write to the control register. The resulting data set is coherent across all registers.

Use Case: Essential for event-driven debugging where you need a single, consistent state of the system immediately following a critical event.

Window Mode



Automatically takes periodic snapshots of the current state of events and counters at user-defined time intervals.

Use Case: Perfect for long-term performance trending, resource analysis, and historical comparison where you need to track changes over defined time segments.

3.4.2. Network Diagnostics Resource Costs

For better design visibility and for gathering network statistics and diagnostics MLE NPAP has the option to spend additional FPGA resources.

As a rule-of-thumb basic Network Diagnostics requires ~10% more LUTs and ~20% more FFs!

The following table is an example showing the **additional resources for diagnosing 6 TCP sessions** in an AMD/Xilinx UltraScale+ device. Please compare this to the absolute resource values in Section 6. below!

Module Name	LUTs	FFs
npap_diag_capture_u0	6262	17828
npap_diag_control_eth	137	203
npap_diag_control_ipv4	137	203
npap_diag_control_tcp	137	203
npap_diag_control_udp	137	203
npap_diag_counter_tcp [0]	760	2695
npap_diag_counter_tcp [1]	1016	2695
npap_diag_counter_tcp [2]	1080	2695
npap_diag_counter_tcp [3]	825	2695
npap_diag_counter_tcp [4]	761	2695
npap_diag_counter_tcp [5]	1210	2695
npap_diag_register_u0	793	822
npap_diag_register_ethernet_u0	1	188
npap_diag_register_ipv4_u0	1	188
npap_diag_register_tcp_u0	60	258
npap_diag_register_udp_u0	1	188

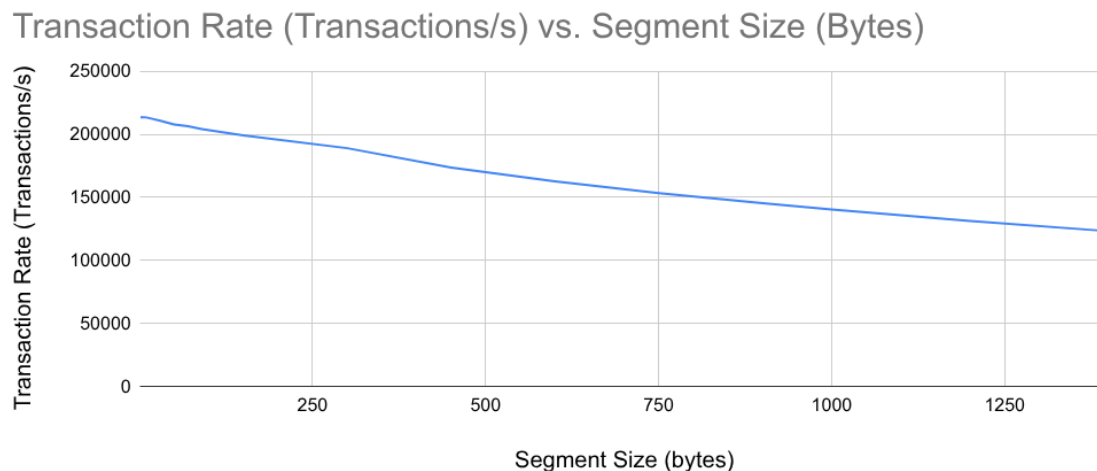
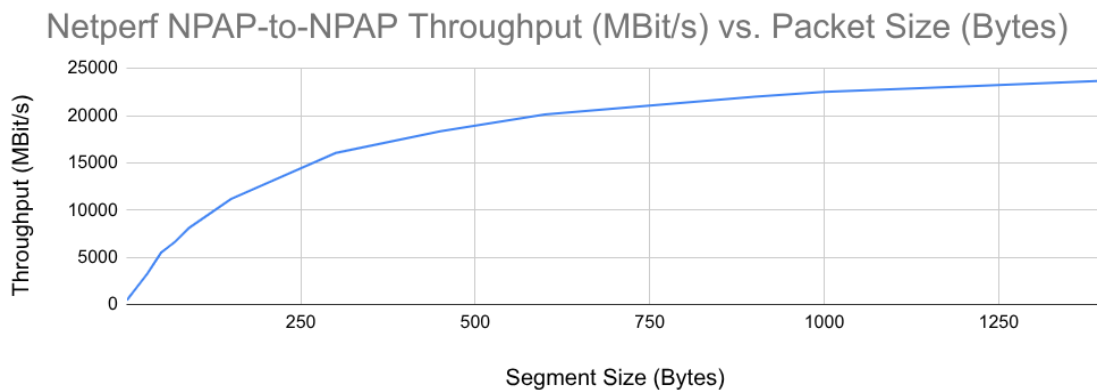
3.4.3. Netperf/Netserver - Benchmark Full Acceleration

MLE NPAP can instantiate a fully accelerated version of Netperf and many of our Evaluation Reference Designs do so (see more details below).

The Netperf/Netserver block is compatible with open source Netperf⁵ 2.6 and can be used for system-wide tuning, for functionality analysis and for performance benchmarking.

The charts below show the results of a 25 GbE NPAP-to-NPAP benchmark, underlining NPAP's superior performance:

- **Consistent High Throughput:** The first chart, measuring a TCP_STREAM test, demonstrates NPAP's ability to achieve consistent, close-to-line-rate throughput.
- **Ultra-Low Latency:** The second chart, measuring a TCP_RR (Request/Response) test, highlights the ultra-high and deterministic transaction rates which is a key advantage that only a full-hardware network stack can deliver due to the required low round-trip-time (RTT) and low packet loss.



⁵ <https://hewlettpackard.github.io/netperf/>

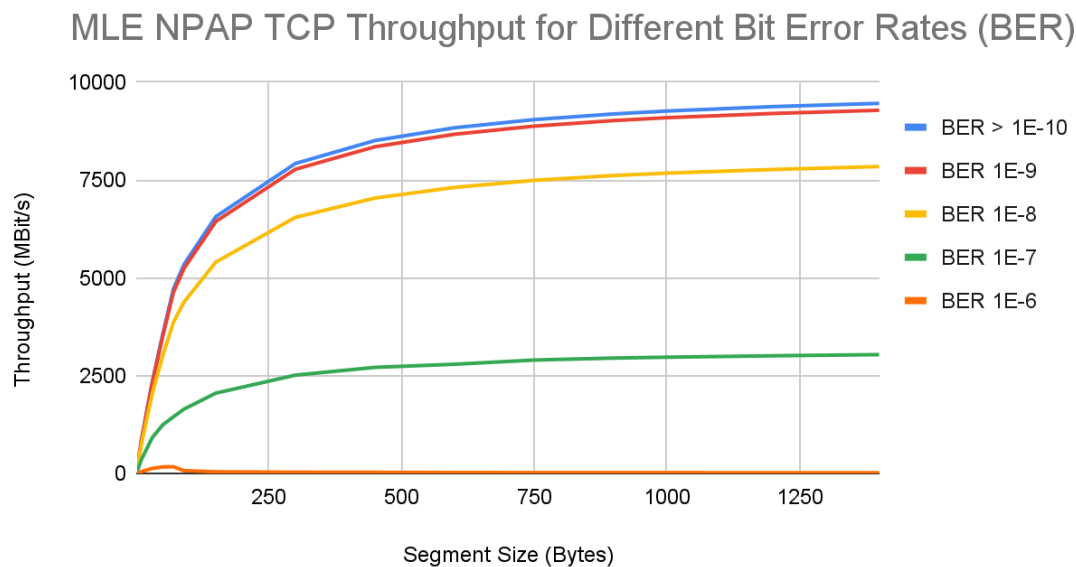
3.5. Network Impairment and Bit Error Insertion

MLE NPAP can be shipped with an optional integrated Network Impairment Generator (Network Impairment is also known as Bit Error Insertion). This helps to rigorously test and validate the robustness of your network by emulating real-world network impairments, such as Bit Error Rates (BER), at line speed.

The Ethernet standard requires a BER no higher than 1×10^{-12} . NPAP's Network Impairment Generator allows you to test your system far beyond this ideal requirement, simulating real-world conditions like those found in high-EMI environments or when using slip rings, for example.

The Network Impairment Generator IP Core sits on the data path between the TCP/UDP/IP layer and the Ethernet MAC layer, allowing for precise, controlled injection of errors. This enables you to perform stress testing and gain valuable insights into system performance under non-ideal conditions, ensuring your implementation remains resilient against link quality degradation.

The graph below illustrates how different levels of Bit Error Rate (BER) over a 10 GBit/s link, emulated using NPAP's Network Impairment Generator, affect the TCP throughput of a TCP connection between two MLE NPAP instances. Obviously, the TCP re-transmissions "eat" into the net data throughput:



This chart shows that MLE's TCP/IP stack can deliver good throughput - even on highly disturbed networks.

3.6. Quality Assurance

MLE ensures the highest quality and reliability for every NPAP release through a rigorous, multi-layered validation strategy built on a robust Continuous Integration (CI) framework. This automated process validates NPAP from initial simulation through final hardware deployment.

Our Quality Assurance process is documented in the MLE Quality Manual (QAM001 Rev. 13a) and involves:

- **Extensive Simulation-Based Verification:** We conduct comprehensive RTL simulations (testing NPAP-to-NPAP) as well as advanced, system-level co-simulations (testing NPAP-to-software). This automated test suite validates all aspects of protocol behavior, from register access and TCP connection management to complex corner-case scenarios like retransmissions, backpressure, and priority handling.
- **Rigorous Hardware Validation:** Simulation is followed by automated nightly hardware regression testing on a large, physical test farm. We validate NPAP across a wide array of target platforms from all major FPGA vendors, including AMD, Intel, Lattice, and Microchip.
- **Real-World Interoperability Testing:** Hardware tests cover numerous real-world use cases, including board-to-board latency and performance benchmarks, and board-to-PC interoperability against standard Linux/Windows stacks using tools like **Netperf**. We also perform stress tests under adverse link conditions using the integrated Network Impairment Generator (see Section 3.5).

This comprehensive approach guarantees that NPAP delivers deterministic, high-performance, and reliable operation across all supported target technologies.

4. TCP/UDP/IP Architecture Aspects and Choices

Team MLE has gained long and deep experiences with integrating NPAP into systems and will support you in identifying, implementing and testing the right architecture choice. Key aspects are outlined below. Please contact us for more details.

4.1. Picking the “Right” TCP Peer

MLE NPAP has a clear focus on highest performance in a very resource efficient FPGA implementation which is reliable (the key reason to use TCP) and interoperable with (almost) any other TCP/UDP/IP stack.

When you optimize data transports towards low latency and/or high bandwidth, keep in mind how both, TCP flow control and TCP congestion control, function and do parameterize both peers accordingly.

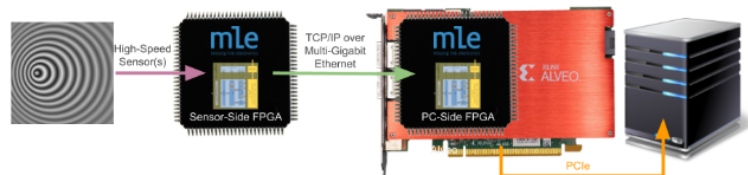
NPAP runs the entire protocol stack as a digital circuit. So, when NPAP is on the receiving side TCP packets will be checked, and acknowledged (ACK’ed), in a very short time and at a very high rate (close to line rate). That may challenge a “slow sender”. Similarly, when NPAP is on the sending side, TCP packets will be generated and sent in a very short time and at a very high rate (close to line rate). That may challenge a “slow receiver”.

Experimenting, and tweaking parameters on either side, is key to deliver good performance. Predictable high bandwidth and low latency is typically delivered by a “balanced” TCP connection such as putting NPAP on both sides.

When implementing so-called High-Speed Data Acquisition systems⁶ we do recommend investigating NPAP not only for sensor-side TCP/IP acceleration but also for PC-side:

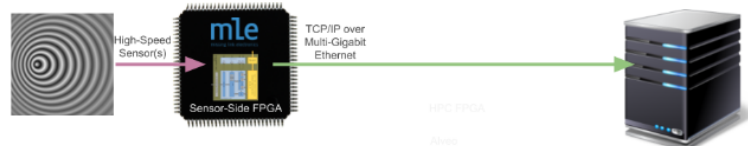
With PC-side FPGA Accelerators:

- All PC CPU cores can fully be used for sensor data processing
- Round-trip time (RTT) is minimal, so only small Tx buffers are needed in the sensor-side FPGA
- All data streams between sensor and server can be prioritized using QoS, scheduling, traffic shaping etc



Unaccelerated PC:

- Precious CPU cores are consumed for network protocol processing
- Round-trip time (RTT) is high which eats up BRAM resources in the sensor-side FPGA
- TCP scheduling and congestion control restricted to Operating System capabilities



⁶ <https://devzone.missinglinkelectronics.com/application-notes/high-speed-data-acquisition-systems/>

4.2. Picking the Right TCP Rx/Tx Buffer Sizes

Obviously, larger Tx and Rx buffers deliver higher bandwidths, but at the cost of transport latency. Worse, NPAP Tx and Rx buffers require expensive FPGA BRAM resources. To help you pick a good tradeoff, here is the metric to determine TCP buffer sizes for TCP (keep in mind, that TCP buffers are placed on both ends: Tx side and Rx side):

- Buffer size (in bits) = Bandwidth (in bits-per-second) * RTT (in seconds)

RTT is the Round-Trip Time which is the time for the sender to transmit the data plus the time-of-flight for the data, plus the time it takes the recipient to check for packet correctness (CRC), plus the time for the recipient to send out the ACK, plus the time-of-flight for the ACK, plus the time it takes the sender to process the ACK and release the buffer. Here examples:

1. Assuming a direct connection with the recipient being NPAP, and a 25 GBit/s link, ACK times can be assumed to be less than 1 microsecond. This translates to minimal buffer sizes of 25 Kbits, which is equivalent to a single 36 Kbit FPGA BRAM.
2. Assuming a 10 GBit/s link and the recipient being software, then RTT can be much longer, mostly due to the longer processing times in the OS on the recipient side. For a modern Linux we can assume RTTs of 50 microseconds, or longer (you can run 'Netperf' on your machine to find out). Means buffer sizes shall be around 500K bits, or the 64K Bytes of BRAM we typically instantiate.

NPAP allows to set Tx and Rx buffer sizes individually and per each TCP Core, to facilitate optimizations for better performance/resource trade-offs in more unidirectional dataflows.

To ensure these configured receive buffers are used effectively, especially when they exceed the standard 65,535-byte TCP window, MLE NPAP implements the TCP Window Scale option (RFC 7323). This capability is essential for achieving line-rate performance on high-latency networks (i.e., those with a large Bandwidth-Delay Product). MLE NPAP automatically calculates the correct scale factor based on the specified Rx Buffer Size for each TCP core. It then advertises this full receive capacity during the connection handshake, enabling the remote peer to send enough data to fill the pipe and maximize throughput.

4.3. Picking the Right Number of TCP Cores

In typical software systems, the cost of opening a TCP connection is quite CPU expensive, and may take a long time because of RTT and processing times in the operating system. Therefore, most software driven systems keep a TCP connection alive “forever” rather than closing it. The low costs of system RAM for storing each TCP connections’ state are not worth the CPU processing costs.

For NPAP, each TCP connection which is open at the same time requires a dedicated TCP Core, which costs FPGA / ASIC resources. However, if RTT is low such as in a LAN, and with the very low costs of opening and closing a TCP connection in NPAP (a few hundred FPGA clock cycles), “time sharing” TCP Cores can save a lot of FPGA resources without any negatives. Please refer to the [MLE Technical Brief TB20201203](#) “Deterministic Networking with TSN-10/25/50/100G” for more information on “time sharing” TCP Cores⁷.

4.4. Optimizing NPAP for Linerate Performance

MLE has been working with FPGA vendors to constantly improve NPAP clock frequency. While NPAP originally was designed for ASIC implementation, MLE has adopted NPAP for efficient implementation using modern FPGA devices. Unlike other TCP stacks for FPGA, NPAP features a 128 bit wide bi-directional datapath which puts NPAP into a unique position for realizing high-bandwidth FPGA-based SmartNICs.

Larger bit widths, 512 bits or more, cause “bloat” which is wasting FPGA resources. Smaller bit widths, 64 bits or less, do require unrealistic high clock frequencies to deliver high line rates as the following table shows:

	10 Gbps	25 Gbps	50 Gbps	100 Gbps
32 bit	312.5 MHz	781.25 MHz	1,562.5 MHz	3,125.0 MHz
64 bits	156.25 MHz	390.625 MHz	781.25 MHz	1,562.5 MHz
128 bits	78.125 MHz	195.3 MHz	390.625 MHz	781.25 MHz
512 bits	19.5 MHz	48.8 MHz	97.7 MHz	195.3 MHz

Please refer to [MLE Technical Brief TB20230523](#) “Put a TCP/UDP/IP Turbo Into Your FPGA-SmartNIC”⁸ as this discusses pipelining the data paths to avoid “FPGA bloat”.

⁷ <https://devzone.missinglinkelectronics.com/application-notes/deterministic-networking-with-tsn-10-25-50-100g/>

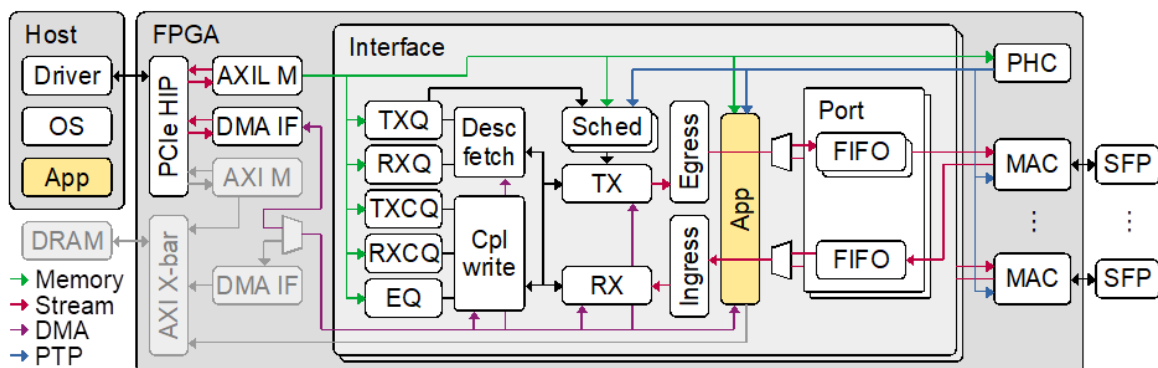
⁸ <https://devzone.missinglinkelectronics.com/application-notes/put-a-tcp-udp-ip-turbo-into-your-fpga-smartnic/>

4.5. Combining NPAP With FPGA Network Interface Cards (NIC)

An FPGA Network Interface Card (NIC) does some network packet handling in FPGA logic and then DMA's the data into a (Linux) host computer. At MLE we have been using (and contributing to) the Corundum project: <http://corundum.io>

Corundum is an open-source, high-performance FPGA-based NIC and platform for In-Network Compute. Features include a high performance datapath, 10G/25G/100G Ethernet, PCIe connectivity to the host, a custom, high performance, tightly-integrated PCIe DMA engine, many (1000+) transmit, receive, completion, and event queues, scatter/gather DMA, MSI, multiple interfaces, multiple ports per interface, per-port transmit scheduling including high precision TDMA, flow hashing, RSS, checksum offloading, and **native IEEE 1588 PTP timestamping**. A Linux driver is included that integrates with the Linux networking stack. Development and debugging is facilitated by an extensive simulation framework that covers the entire system from a simulation model of the driver and PCI express interface on one side to the Ethernet interfaces on the other side (<https://docs.corundum.io/en/latest/contents.html>).

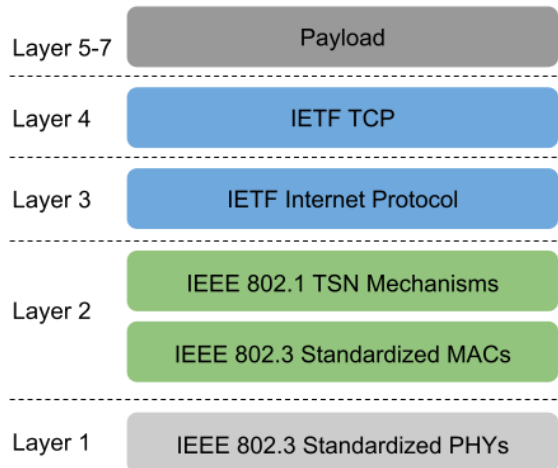
One of the key advantages of the Corundum architecture is support for In-Network Processing inside the FPGA logic, shown as “App” in the block diagram:



These Corundum “Apps” can serve as a “turbo”, and one of those turbos can be NPAP. This is quickly evolving so please contact us for more information!

4.6. Implementing Time-Sensitive Networking (TSN)

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. TSN and TCP can be combined according to the OSI Layers.



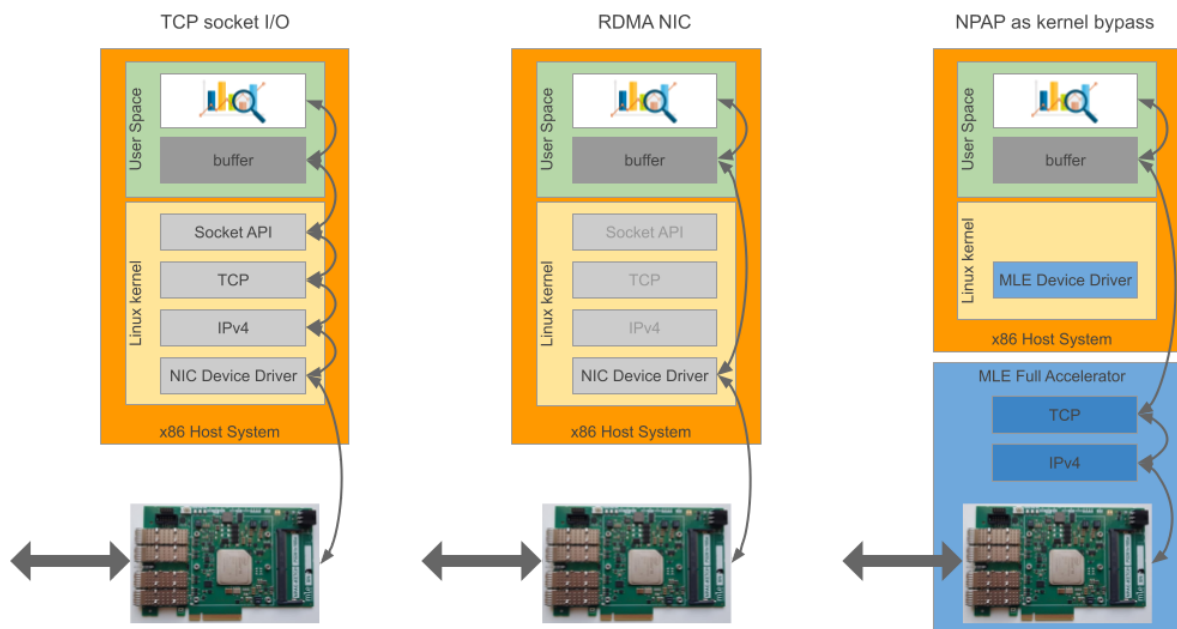
The outcome is a deterministic and reliable network protocol, which makes TCP/IP over TSN a very good candidate for all networking where IT (Information Technology) and OT (Operations Technology) converge, or in Systems-of-Systems backbones. TSN itself is quickly evolving so please contact us for more information!

4.7. Linux Kernel Bypass With NPAP

Increased Ethernet speeds push a need to offload CPUs from the burden of TCP/UDP/IP processing. Various kernel bypass options exist, some include so-called RDMA (Remote DMA). NPAP can provide architecture choices for implementing such kernel bypass.

Normally, from a CPU's perspective, TCP socket I/O means sending and/or receiving (raw) network data between the host CPU and the NIC. The kernel runs network protocol processing for IPv4, TCP, and socket APIs. User space applications that generate and/or digest the network data add to the CPU processing burden.

RDMA skips most steps and a so-called rNIC (RDMA NIC) device driver directly interfaces with user space memory, effectively bypassing the kernel.



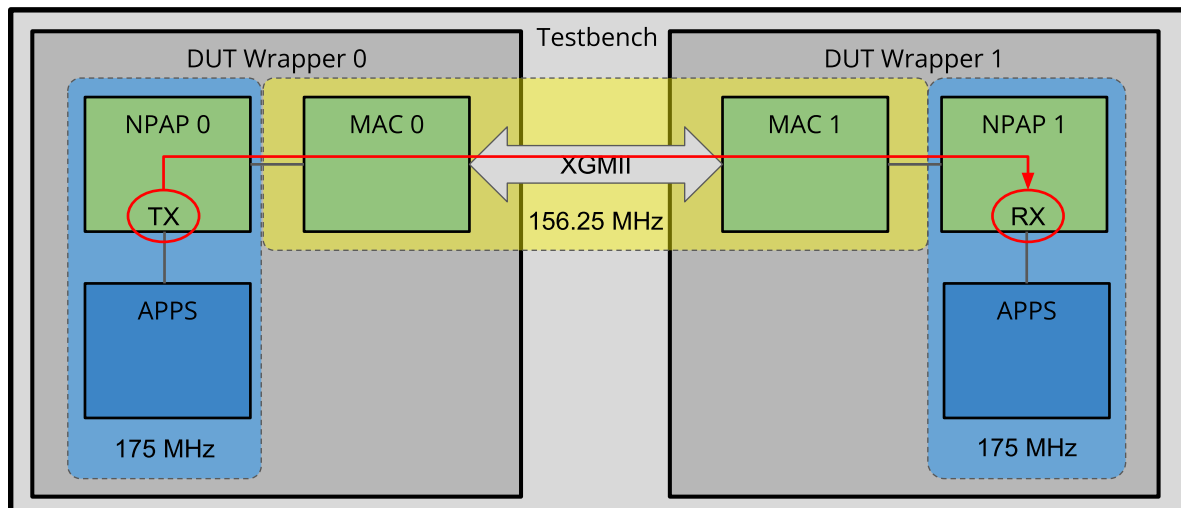
NPAP can operate in a similar way, where NPAP does all TCP/IP processing in dedicated ASIC / FPGA logic and then a (Linux) device driver copies the payload data between user space memory and NPAP.

4.8. Latency Analysis Results

MLE analyzed processing latency using RTL simulation of two instances of NPAP (using different clock speeds) connected via 10G LL MAC via XGMII (clocked at 156.25 MHz).

TCP Payload Size [Byte]	Clock cycles	RTT [ns] at 175 MHz	RTT [ns] at 322 MHz	RTT [ns] at 550 MHz
1	62	708.6	385.1	225.5
32	67	765.7	416.1	243.6
64	73	834.3	453.4	265.5
160	91	1,040.0	565.2	330.9
448	145	1,657.1	900.6	527.3
960	241	2,754.3	1,496.9	876.4
1216	289	3,302.9	1,795.0	1,050.9
1456	334	3,817.1	2,074.5	1,214.5

Latency was measured “door-to-door”, i.e. we measured the time difference between sending payload data from one NPAP instance via TCP/IP until receiving that payload data at the other instance of NPAP, see the system-level block diagram:



Obviously, increasing the NPAP clock frequency will reduce RTT latency for asynchronous NPAP subsystems. More information on dependable latency numbers can be found in our Technical Brief “Myth-Busting Latency Numbers for TCP Offload Engines”⁹

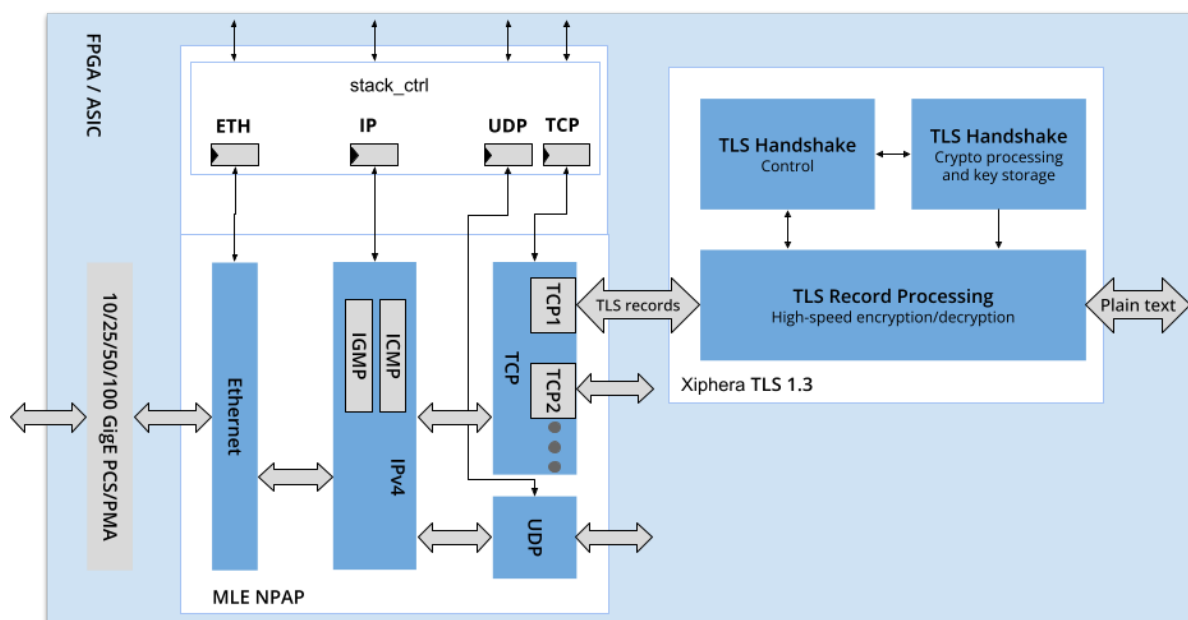
⁹ <https://devzone.missinglinkelectronics.com/application-notes/myth-busting-latency-numbers-for-tcp-offload-engines/>

4.9. Optional Transport Layer Security (TLS)

MLE has been working with partner Xiphra to integrate NPAP with Xiphra's TLS IP Cores for FPGA. Successful integration has been delivered to first customers.

TLS is a cryptographic protocol that provides end-to-end data security, on top of the Transmission Control Protocol (TCP) layer. Implementing TLS has become a standard practice for building secure web apps. With growing needs for security compliance, for example under IEC 62443, TLS is also an option for protecting sensitive data transported using MLE NPAP.

Collaboration within the FPGA ecosystem created joint solutions combining MLE's [TCP/IP Network Protocol Accelerator Platform \(NPAP\)](#) with Xiphra's [TLS 1.3](#) to ensure secure and reliable connection between devices over LAN and WAN.



Since the TCP/IP stack and the TLS 1.3 security protocol – including importantly both key exchange and key management – are both executed entirely in hardware, the joint solution has both scalable high-speed performance and minimizes attack surface, especially when compared to a software-based approach. The FPGA hereby utilizes the hardware Root of Trust, best suited for applications like critical communication in defense, space technology, and energy production and distribution .

5. NPAP Evaluation Choices

MLE offers multiple ways to evaluate and benchmark NPAP:

- Free-of-charge Evaluation Reference Designs are available for a limited set of off-the-shelf hardware (see section [Evaluation Reference Designs](#) for more info on available development kits)
- A “Developers License” is a highly discounted extended evaluation license which gives you full source code access to integrate and run NPAP within your target hardware

In [MLEs Developer Zone](#) you will find an in depth guide of how to evaluate NPAP using the Evaluation Reference Designs.

5.1. Available Evaluation Reference Designs

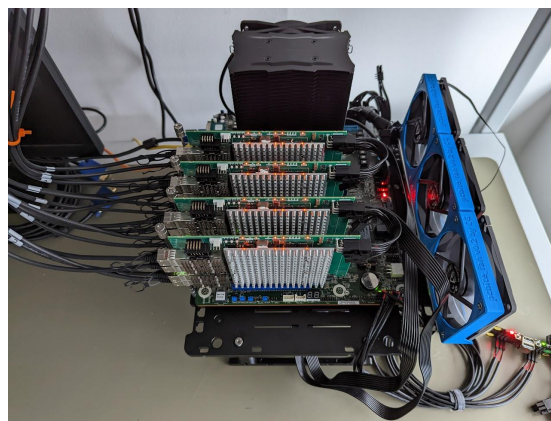
For evaluating the functioning and the performance of NPAP MLE provides Evaluation Reference Designs (ERD) for several FPGA Development Kits:

- MLE’s Network Protocol Accelerator Card “Ketch” with Intel Stratix 10 GX
- NPAP-1G on AMD/Xilinx ZCU102 with Zynq Ultrascale+ MPSoC ZU9EG
- NPAP-10G on AMD/Xilinx ZCU102 with Zynq Ultrascale+ MPSoC ZU9EG
- NPAP-10G on AMD/Xilinx Artix Ultrascale+ FPGA AU15P
- NPAP-10G on AMD/Xilinx ZCU111 with Zynq Ultrascale+ RFSoc ZU28EG
- NPAP-25G on AMD/Xilinx ZCU111 with Zynq Ultrascale+ RFSoc ZU28EG
- NPAP-100G on AMD/Xilinx ZCU111 with Zynq Ultrascale+ RFSoc ZU28EG
- NPAP-25G on AMD Alveo U200
- NPAP-25G on AMD Alveo U55C
- NPAP-25G on Trenz Electronics TE0950 with AMD Versal AI Edge VE2302
- NPAP-25G on Intel Agilex 5 E-series A5E 043B / 065B
- NPAP-10G on Microchip PolarFire MPF300-EVAL-KIT
- NPAP-10G on Lattice Avant-G Versa Board
- NPAP-25G on Lattice Avant-X Versa Board (in development)

The objective of an ERD is to facilitate design-in via a complete design project running on selected FPGA boards.

Each ERD typically instantiates the full stack including MAC, Ethernet, IPv4 (with ICMP and IGMP), plus one or more TCP session instances, the UDP block, a Netperf/Netserver implementation and an Network Impairment generator in programmable logic.

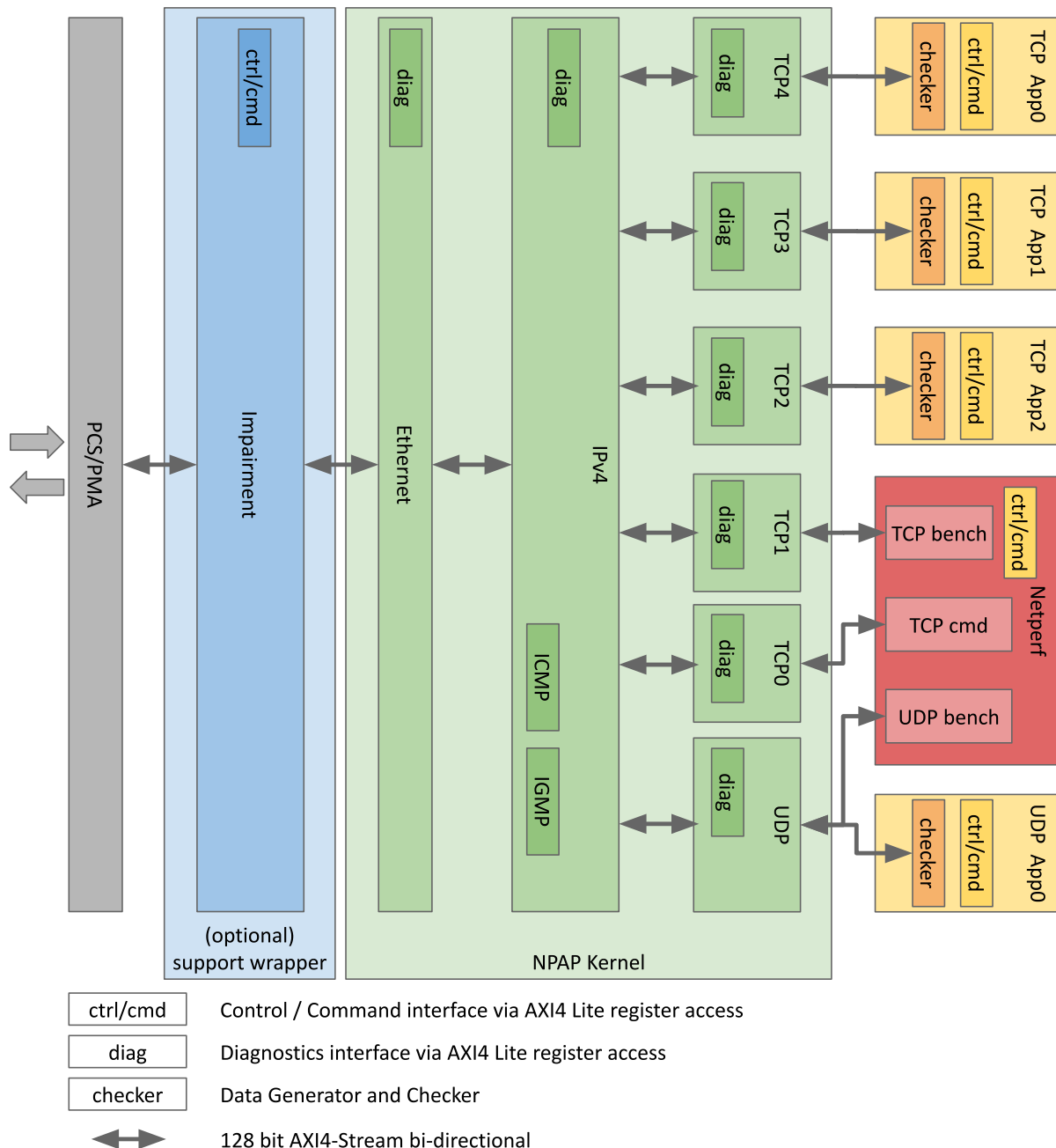
Here some exemplary setups from MLE's NPAP Test Lab:



5.2. Evaluation Reference Design (ERD) Structure

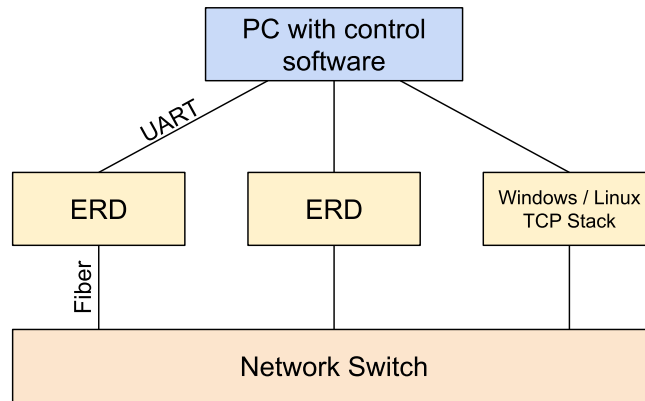
A hierarchical design philosophy is used to integrate MLE NPAP together with the auxiliary blocks. Related blocks are packaged together in a “support wrapper”. These are nested and joined within a top-level wrapper.

The following shows an example of an ERD, including multiple TCP cores with data generators and data checkers, plus an instance of Netperf.



The AXI4-Lite protocol is used to manage all IPs within the NPAP ecosystem. In some ERD implementations, the top-level AXI4-Lite connects to an AXI4-Lite to XFCP

converter. This converter, controlled via UART, simplifies the interface between a PC and one or more ERDs:



The ERD can be used to evaluate MLE NPAP and as a starting point for your own integration of NPAP.

A more in-depth documentation of the workings of the ERD and the support IPs is available under license. Please contact your MLE sales representative for more information.

6. Resource Estimates for ASIC and FPGA

While NPAP has been implemented using re-targetable RTL HDL code we just cannot provide resource estimates for all possible target technologies. If you are interested in integrating NPAP into another target technology, please contact us.



Resource estimates are directly tied to the specific configuration of NPAP. Due to its high configurability, a single NPAP instance can range from a minimum of 10K LUTs to, literally, 1 million LUTs (for hundred TCP cores). For inquiries about resource estimates tailored to your specific part, please contact your MLE sales representative.

6.1. Resource Estimates for AMD Versal AI Edge Series

The following table shows resources for AMD/Xilinx Versal AI Edge Series fabric (xcve2302-sfva784-1LP-e-S-es1) compiled with AMD/Xilinx Vivado/Vitis 2024.1 - instantiating the following design features

- Ethernet block
- IPv4 block
- 1 UDP block
- 4 instances of TCP blocks
- Diagnostics blocks

	Total	Logic	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	URAM	DSP
	LUTs	LUTs							Blocks
npap_tcp_udp_wrapper_u0	70944	69010	1914	20	62052	25	1	70	4
npap_tcp_udp_top_u0	70944	69010	1914	20	62052	25	1	70	4
axis_register_udp_u0	228	228	0	0	430	0	0	0	0
axis_register_udp_u1	292	292	0	0	431	0	0	0	0
gen_hhi_to_axis_adapter	1214	1214	0	0	1978	0	0	0	0
npap_configuration_block_u0	103	103	0	0	339	0	0	0	0
npap_priority_manager_configurati	29	29	0	0	43	0	0	0	0
npap_visibility_block	9500	9480	0	20	13324	0	0	0	0
npap_diag_capture_u0	8297	8277	0	20	12511	0	0	0	0
(npap_diag_capture_u0)	46	26	0	20	694	0	0	0	0
npap_diag_control_eth	157	157	0	0	278	0	0	0	0
npap_diag_control_ipv4	157	157	0	0	278	0	0	0	0
npap_diag_control_tcp	157	157	0	0	203	0	0	0	0
npap_diag_control_udp	157	157	0	0	278	0	0	0	0
npap_diag_counter_tcp_loop[0]	1803	1803	0	0	2695	0	0	0	0
npap_diag_counter_tcp_loop[1]	1961	1961	0	0	2695	0	0	0	0
npap_diag_counter_tcp_loop[2]	1936	1936	0	0	2695	0	0	0	0
npap_diag_counter_tcp_loop[3]	1923	1923	0	0	2695	0	0	0	0
npap_diag_register_u0	1204	1204	0	0	813	0	0	0	0
(npap_diag_register_u0)	809	809	0	0	0	0	0	0	0
npap_diag_register_ethernet_u	1	1	0	0	190	0	0	0	0
npap_diag_register_ipv4_u0	1	1	0	0	189	0	0	0	0
npap_diag_register_tcp_u0	393	393	0	0	245	0	0	0	0
npap_diag_register_udp_u0	1	1	0	0	189	0	0	0	0
wrapper_ll_ip_tcp_u0	59583	57669	1914	0	45507	25	1	70	4
(wrapper_ll_ip_tcp_u0)	633	633	0	0	765	0	0	0	0
g0.i_tcpTxDmux	0	0	0	0	2	0	0	0	0
gen_WithUdp.i_udp	4864	4864	0	0	4265	5	0	8	0
gen_tcpConnections[0].i_tcp	12007	11573	434	0	8830	4	0	17	1
gen_tcpConnections[1].i_tcp	11174	10764	410	0	8365	1	0	11	1
gen_tcpConnections[2].i_tcp	12007	11573	436	0	8846	4	0	17	1
gen_tcpConnections[3].i_tcp	12011	11575	436	0	8931	4	0	17	1
iBusScheduler8	531	531	0	0	36	0	0	0	0
i_internetLayer	3383	3265	118	0	3108	0	0	0	0
i_networkLayer	2947	2867	80	0	2404	7	1	0	0
i_resetStretchStack	54	54	0	0	45	0	0	0	0

6.2. Resource Estimates for AMD/Xilinx Ultrascale+ Series

The following table shows resources for AMD/Xilinx Zynq Ultrascale+ MPSoC ZU9EG compiled with AMD/Xilinx Vivado 2022.2 - instantiating the following design features:

- 10 GigE Low-Latency MAC from Fraunhofer HHI
- Ethernet block
- IPv4 block
- UDP block
- 10 instances of TCP blocks
- Diagnostics blocks

Instance	Total LUTs	Logic LUTs	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	URAM	DSP Blocks
top	111324	106686	4540	0	129415	707	0	0	10
mac_10_gbe_wrapper_u0	1790	1780	10	0	1016	0	0	0	0
mac10gbe_top_u0	1790	1780	10	0	1016	0	0	0	0
mac10Gbe_struct_u0	1790	1780	10	0	1016	0	0	0	0
mac_interface_adapter_generic_64_u	161	161	0	0	370	0	0	0	0
axis_dwidth_converter_128_to_64_u	119	119	0	0	222	0	0	0	0
axis_dwidth_converter_64_to_128_u	42	42	0	0	148	0	0	0	0
reset_stretch_reset_rx_mac	6	6	0	0	17	0	0	0	0
reset_stretch_reset_tx_mac	6	6	0	0	17	0	0	0	0
reset_sync_reset_system_to_reset_m	0	0	0	0	2	0	0	0	0
reset_sync_reset_system_to_reset_m	0	0	0	0	2	0	0	0	0
npap_tcp_udp_wrapper_u0	109534	104906	4530	98	128399	707	6	0	10
npap_tcp_udp_top_u0	109534	104906	4530	98	128399	707	6	0	10
axis_register_udp_u0	100	100	0	0	428	0	0	0	0
axis_register_udp_u1	263	263	0	0	424	0	0	0	0
gen_hhi_to_axis_adapter	3112	3112	0	0	266	0	0	0	0
npap_configuration_block_u0	135	135	0	0	338	0	0	0	0
npap_priority_manager_confi	52	52	0	0	52	0	0	0	0
npap_visibility_block	11118	11068	0	50	29770	0	0	0	0
npap_diag_capture_u0	10311	10261	0	50	28912	0	0	0	0
(npap_diag_capture_u0)	115	65	0	50	1150	0	0	0	0
npap_diag_control_eth	137	137	0	0	203	0	0	0	0
npap_diag_control_ipv4	137	137	0	0	203	0	0	0	0
npap_diag_control_tcp	137	137	0	0	203	0	0	0	0
npap_diag_control_udp	137	137	0	0	203	0	0	0	0
npap_diag_counter_tcp_l	9651	9651	0	0	2695	0	0	0	0
npap_diag_register_u0	808	808	0	0	858	0	0	0	0
wrapper_ll_ip_tcp_u0	94739	90161	4530	48	91959	707	6	0	10
(wrapper_ll_ip_tcp_u0)	800	800	0	0	192	0	0	0	0
g0.i_tcpTxMux	0	0	0	0	4	0	0	0	0
gen_WidthUdp.i_udp	3649	3601	0	48	3889	47	5	0	0
gen_tcpConnections[0].i_t	8403	7977	426	0	8153	68	0	0	1
gen_tcpConnections[1].i_t	8052	7618	434	0	7792	41	0	0	1
gen_tcpConnections[2].i_t	9720	9286	434	0	8410	68	0	0	1
gen_tcpConnections[3].i_t	8440	8006	434	0	8297	68	0	0	1
gen_tcpConnections[4].i_t	8449	8015	434	0	8296	68	0	0	1
gen_tcpConnections[5].i_t	8446	8012	434	0	8297	68	0	0	1
gen_tcpConnections[6].i_t	8446	8012	434	0	8300	68	0	0	1
gen_tcpConnections[7].i_t	8442	8008	434	0	8311	68	0	0	1
gen_tcpConnections[8].i_t	8456	8022	434	0	8297	68	0	0	1
gen_tcpConnections[9].i_t	8397	8003	434	0	8298	68	0	0	1
iBusScheduler8	1108	1108	0	0	69	0	0	0	0
i_internetLayer	1954	1836	118	0	3043	0	0	0	0
i_networkLayer	1894	1814	80	0	2276	7	1	0	0
i_resetStretchStack	48	48	0	0	35	0	0	0	0

6.3. Resource Estimates for AMD/Xilinx 7-Series

The following table shows resources for AMD/Xilinx 7-Series Kintex fabric (XC7Z045-2) compiled with AMD/Xilinx Vivado 2014.4 - instantiating the following design features:

- 10 GigE Low-Latency MAC from Fraunhofer HHI
- Ethernet block
- IPv4 block
- UDP block
- 2 instances of TCP blocks
- No diagnostics

Instance	Module	Total LUTs	Logic LUTs	LUT RAMs	SRLs	FFs	RAMB36	RAMB18	DSP48 Blocks
wrapper_mac_10gStack	(top)	29459	28425	904	130	25727	50	6	4
(wrapper_mac_10gStack)	(top)	241	241	0	0	0	0	0	0
i_10GStack	Wrapper_LL_IP_TCP_parameterized0	27045	26013	904	128	24644	50	6	4
g0.i_tcpTxMux	TcpTxMux_parameterized0	0	0	0	0	3	0	0	0
gen_WithUdp.i_udp	wrapper_udp_parameterized0	3538	3350	92	96	3732	14	0	0
gen_tcpConnections[0].i	Wrapper_TCP_parameterized0	9347	8963	384	0	7938	16	3	2
gen_tcpConnections[1].i	Wrapper_TCP_parameterized0_1	9349	8965	384	0	7938	16	3	2
iBusScheduler8	BusScheduler8_parameterized0	568	568	0	0	31	0	0	0
i_internetLayer	Wrapper_IP_parameterized0	2989	2913	44	32	3437	2	0	0
i_netLayerConv	MacNetworkLayerConversion_parameterized0	139	139	0	0	72	0	0	0
i_networkLayer	Wrapper_NetworkLayer_parameterized0	1125	1125	0	0	1491	2	0	0
i_txLinkResetSync	ResetSync_parameterized0_2	0	0	0	0	2	0	0	0
i_ResetStretch_aux	ResetStretch_parameterized0	75	74	0	1	35	0	0	0
i_ResetStretch_stack	ResetStretch_parameterized2	75	74	0	1	34	0	0	0
i_clk_stretch	clk_stretch	1	1	0	0	29	0	0	0
i_gen100ms	GenClk100ms_parameterized0	69	69	0	0	30	0	0	0
i_mac10GbE	mac10GbE_wrapper_parameterized0	1957	1957	0	0	955	0	0	0
(i_mac10GbE)	mac10GbE_wrapper_parameterized0	0	0	0	0	1	0	0	0
i_mac10GbE	mac10GbE_parameterized0	1957	1957	0	0	954	0	0	0

6.4. Resource Estimates for Altera/Intel Agilex 5E

The following table shows resources synthesized for Agilex 5E A5ED065BB32AE4SR0 using Altera Quartus v24.2 - instantiating the following design features:

- Ethernet block
- IPv4 block
- UDP block
- 7 instances of TCP blocks
- Diagnostics blocks

Compilation Hierarchy Node	ALMs used in final placement	Combi- national ALUTs	Dedicated Logic Registers	Block Memory Bits	M20Ks
npap_tcp_udp_wrapper_u0	78569.5	103514	117217	18305712	1046
npap_tcp_udp_top_u0	78569.5	103514	117217	18305712	1046
axis_register_udp_u0	191.0	223	486	0	0
axis_register_udp_u1	172.8	222	427	0	0
gen_hhi_to_axis_adapter[0-6]	1736.1	2240	3920	0	0
hhiudp_axis_adapter_u0	20.1	42	4	0	0
npap_configuration_block_u0	222.2	254	735	0	0
npap_priority_manager_configuration_b	47.3	68	60	0	0
npap_visibility_block	13698.9	15617	22381	0	0
npap_diag_capture_u0	12937.2	14732	21099	0	0
npap_diag_control_eth	125.7	247	240	0	0
npap_diag_control_ipv4	128.3	247	246	0	0
npap_diag_control_tcp	127.7	247	257	0	0
npap_diag_control_udp	124.9	247	249	0	0
npap_diag_counter_tcp_loop[0-6]	9046.8	10535	18609	0	0
npap_diag_register_u0	761.7	885	1282	0	0
npap_diag_register_ethernet_u0	191.4	252	381	0	0
npap_diag_register_ipv4_u0	144.8	150	232	0	0
npap_diag_register_tcp_u0	277.2	349	478	0	0
npap_diag_register_udp_u0	148.3	134	191	0	0
wrapper_ll_ip_tcp_u0	62479.9	84846	89199	18305712	1046
g0.i_tcpTxMux	8.8	13	4	0	0
gen_withUdp.i_udp	3102.2	4519	5322	1753088	91
gen_tcpConnections[0].i_tcp	7926.0	10966	11470	2482096	140
gen_tcpConnections[1].i_tcp	6987.8	9801	9836	1375184	85
gen_tcpConnections[2].i_tcp	7970.8	10884	11148	2482688	141
gen_tcpConnections[3].i_tcp	8033.7	10912	10900	2482688	141
gen_tcpConnections[4].i_tcp	8061.9	10914	11347	2482688	141
gen_tcpConnections[5].i_tcp	7976.0	10891	11336	2482688	141
gen_tcpConnections[6].i_tcp	8015.6	10887	11038	2482688	141
i_internetLayer	2305.1	2955	2961	5312	7
i_networkLayer	1905.1	1897	3351	276592	18
i_resetStretchStack	24.5	42	47	0	0

6.5. Resource Estimates for Altera/Intel Stratix-10

The following table shows resources for compiled with Altera Quartus Prime v22.4 for 1SX280HN2F43E2VG - instantiating the following design features:

- 10 GigE Low-Latency MAC from Fraunhofer HHI
- Ethernet block
- IPv4 block
- UDP block
- 3 instances of TCP blocks
- No diagnostics

Compilation Hierarchy Node	ALMs used in final placement	ALMs used for memory	Dedicated Combinational ALUTs	Logic Registers	Block Memory Bits	M20Ks	DSP Blocks
gen_loopback_tcp_interface_wrapper_top[0].u	35194.0 (4.2)	200.0 (0.0)	44839 (18)	34020 (0)	2112512	163	3
gen_loopbackServer.i_loopbackServer	100.2 (0.0)	0.0 (0.0)	57 (0)	172 (0)	0	0	0
gen_loopbackServer.i_loopbackServer	100.2 (100.2)	0.0 (0.0)	57 (57)	172 (172)	0	0	0
gen_loopback_tcp_interface_wrapper_top[1].u	99.1 (0.0)	0.0 (0.0)	57 (0)	172 (0)	0	0	0
gen_loopbackServer.i_loopbackServer	99.1 (99.1)	0.0 (0.0)	57 (57)	172 (172)	0	0	0
gen_loopback_tcp_interface_wrapper_top[2].u	101.4 (0.0)	0.0 (0.0)	57 (0)	172 (0)	0	0	0
gen_loopbackServer.i_loopbackServer	101.4 (101.4)	0.0 (0.0)	57 (57)	172 (172)	0	0	0
mac_i0_gbe_wrapper_u0	2089.5 (0.0)	0.0 (0.0)	2926 (0)	1378 (0)	0	0	0
mac_i0_gbe_top_u0	2089.5 (0.0)	0.0 (0.0)	2926 (0)	1378 (0)	0	0	0
npap_tcp_udp_wrapper_u0	32624.9 (0.0)	200.0 (0.0)	41429 (0)	31867 (0)	2112512	163	3
npap_tcp_udp_top_u0	32624.9 (2.2)	200.0 (0.0)	41429 (4)	31867 (1)	2112512	163	3
wrapper_ll_ip_tcp_u0	32622.7 (67.2)	200.0 (0.0)	41425 (1)	31866 (196)	2112512	163	3
GEN_MAC_NET_CONV_HHI.i_netLayerConv	1948.2 (9.2)	0.0 (0.0)	785 (15)	2655 (0)	0	0	0
gen_64bitAlign.i_rxAlign	74.2 (74.2)	0.0 (0.0)	13 (13)	145 (145)	0	0	0
gen_rxSyncFifo.i_fifoCDC	1855.4 (1855.4)	0.0 (0.0)	744 (744)	2507 (2507)	0	0	0
i_n1ToMacMux	9.3 (9.3)	0.0 (0.0)	13 (13)	3 (3)	0	0	0
g0.i_tcpTxMux	1.7 (1.7)	0.0 (0.0)	3 (3)	3 (3)	0	0	0
gen_WithUdp.i_udp	485.7 (0.0)	0.0 (0.0)	626 (0)	988 (2)	278560	18	0
gen_tcpConnections[0].i_tcp	8480.2 (35.9)	60.0 (0.0)	11408 (12)	7936 (87)	593568	45	1
gen_tcpConnections[1].i_tcp	8460.5 (40.8)	60.0 (0.0)	11357 (12)	7600 (87)	593568	45	1
gen_tcpConnections[2].i_tcp	8497.9 (34.8)	60.0 (0.0)	11376 (12)	7801 (86)	593568	45	1
iBusScheduler8	86.2 (86.2)	0.0 (0.0)	136 (136)	55 (55)	0	0	0
i_internetLayer	2741.3 (6.2)	20.0 (0.0)	3560 (10)	2657 (5)	17920	6	0
i_networkLayer	1851.8 (0.0)	0.0 (0.0)	2173 (0)	2071 (0)	35328	4	0
i_rxLinkResetSync	1.0 (1.0)	0.0 (0.0)	0 (0)	2 (2)	0	0	0
i_txLinkResetSync	1.0 (1.0)	0.0 (0.0)	0 (0)	2 (2)	0	0	0

6.6. Resource Estimates for Lattice Avant-G

The following table shows resources synthesized for Lattice Avant-G LAT-AT-G70ES-3LFG1156C using Lattice Radiant Version 2024.2.1 - instantiating the following design features:

- Ethernet block
- IPv4 block
- UDP block
- 3 instances of TCP blocks
- Diagnostics blocks

Instance	Logic	DRAM	Ripple Logic	Registers	EBR
top	61003	1374	18962	53694	272
npap_tcp_udp_wrapper_u0	57027	1374	18814	52958	270
npap_tcp_udp_top_u0	57027	1374	18814	52958	270
axis_register_udp_u0	224	0	0	425	0
axis_register_udp_u1	219	0	0	413	0
gen_hhi_to_axis_adapter	808	0	0	1258	0
hhiudp_axis_adapter_u0	47	0	0	4	0
npap_configuration_block_u0	349	0	0	338	0
npap_priority_manager_co...	60	0	10	35	0
npap_visibility_block	6756	0	4160	10387	0
npap_diag_capture_u0	5645	0	4160	9620	0
npap_diag_register_u0	1111	0	0	767	0
wrapper_ll_ip_tcp_u0	48518	1374	14644	40098	270
g0.i_tcpTxMux	469	0	0	29	0
gen_WithUdp.i_udp	4884	24	1434	6142	52
gen_tcpConnections[0]...	12646	264	4120	9838	81
gen_tcpConnections[1]...	11406	276	3976	8597	48
gen_tcpConnections[2]...	12769	312	3980	9786	81
iBusScheduler8	93	0	14	31	0
i_internetLayer	2836	498	776	3252	0
i_networkLayer	3358	0	216	2093	8
i_resetStretchStack	14	0	32	34	0
mac_phy_u0	3673	0	148	3066	2
mac_interface_adapter_u0	303	0	0	370	0

6.7. Resource Estimates for Microchip Polarfire

The following table shows resources synthesized for Microchip PolarFire MPF300TS-1FCG1152I using Microchip Libero 2024.2 - instantiating the following design features:

- Ethernet block
- IPv4 block
- UDP block
- 3 instances of TCP blocks
- No diagnostics

Instance	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	uSRAM 1K	LSRAM 18K	Math 18x18	Chip Global
npap_tcp_udp_wrapper_u0	60339	31116	6792	6792	122	146	2	12
npap_tcp_udp_top_u0	60339	31116	6792	6792	122	146	2	12
Primitives	13	1	0	0	0	0	0	0
interface_adapter (all)	57	1	0	0	0	0	0	0
wrapper_ll_ip_tcp_u0	60269	31115	6792	6792	122	146	2	12
Primitives	349	195	0	0	0	0	0	0
i_netLayerConv	249	206	0	0	0	0	0	0
i_tcpTxMux	123	2	0	0	0	0	0	0
gen_WithUdp.i_udp	5598	3997	1548	0	24	35	0	0
gen_tcpConnections[0].i	15560	7037	1740	1740	31	37	0	3
gen_tcpConnections[1].i	13287	6649	1068	1068	26	19	2	3
gen_tcpConnections[2].i	15707	7112	1752	1752	35	37	0	3
iBusScheduler8	106	34	0	0	0	0	0	0
i_internetLayer	3717	2995	216	216	6	4	0	1
i_networkLayer	5306	2665	504	504	0	14	0	0

7. Detailed Protocol Support (RFC1122 excerpt)

7.1. Ethernet Layer

Feature	Section	Must	Must not	Implemented
Send Trailers by default without negotiation	2.3.1		x	x
ARP	2.3.2			
Flush out-of-date ARP cache entries	2.3.2.1	x		(x)
Prevent ARP floods	2.3.2.1	x		(x)
Ethernet and IEEE 802 Encapsulation	2.3.3			
Host able to:	2.3.3			
Send & receive RFC-894 encapsulation	2.3.3	x		x
Send K1=6 encapsulation	2.3.3		x	
Use ARP on Ethernet and IEEE 802 nets	2.3.3	x		x
Link layer report b'casts to IP layer	2.4	x		
IP layer pass TOS to link layer	2.4	x		
No ARP cache entry treated as Dest. Unreach.	2.4		x	x

7.2. IP & ICMP Layer

Feature	Section	Must	Must not	Implemented
Implement IP and ICMP	3.1	x		x
Handle remote multihoming in application layer	3.1	x		x
Meet gateway specs if forward datagrams	3.1	x		-
Silently discard Version != 4	3.2.1.1	x		x
Verify IP checksum, silently discard bad dgram	3.2.1.2	x		x
Addressing:				
Subnet addressing (RFC-950)	3.2.1.3	x		-
Src address must be host's own IP address	3.2.1.3	x		x
Silently discard datagram with bad dest addr	3.2.1.3	x		x
Silently discard datagram with bad src addr	3.2.1.3	x		x
Support reassembly	3.2.1.4	x		-
TOS:				

Allow transport layer to set TOS	3.2.1.6	x		-
TTL:				
Send packet with TTL of 0	3.2.1.7		x	x
Discard received packets with TTL > 2	3.2.1.7		x	-
Allow transport layer to set TTL	3.2.1.7	x		-
Fixed TTL is configurable	3.2.1.7	x		x
IP Options:				
Allow transport layer to send IP options	3.2.1.8	x		-
Pass all IP options rcvd to higher layer	3.2.1.8	x		-
IP layer silently ignore unknown options	3.2.1.8	x		x
Silently ignore Stream Identifier option	3.2.1.8b	x		x
Source Route Option:				
Originate & terminate Source Route options	3.2.1.8c	x		-
Datagram with completed SR passed up to TL	3.2.1.8c	x		-
Build correct (non-redundant) return route	3.2.1.8c	x		-
Send multiple SR options in one header	3.2.1.8c	x		-
ROUTING OUTBOUND DATAGRAMS:				
Use address mask in local/remote decision	3.3.1.1	x		x
Operate with no gateways on conn network	3.3.1.1	x		x
Maintain "route cache" of next-hop gateways	3.3.1.2	x		-
If no cache entry, use default gateway	3.3.1.2	x		x
Support multiple default gateways	3.3.1.2	x		-
Able to detect failure of next-hop gateway	3.3.1.4	x		-
Ping gateways continuously	3.3.1.4		x	-
Ping only when traffic being sent	3.4.1.4	x		-
Ping only when no positive indication	3.3.1.4	x		-
Switch from failed default g'way to another	3.3.1.5	x		-
Manual method of entering config info	3.3.1.6	x		-
REASSEMBLY and FRAGMENTATION:				
Able to reassemble incoming datagrams	3.3.2	x		-
Transport layer able to learn MMS _R	3.3.2	x		-
Send ICMP Time Exceeded on reassembly timeout	3.3.2	x		-
Pass MMS _S to higher layers	3.3.3	x		-
MULTIHOMING:				
Allow application to choose local IP addr	3.3.4.2	x		x
BROADCAST:				
Broadcast addr as IP source addr	3.2.1.3		x	-
Recognize all broadcast address formats	3.3.6	x		-
Use IP b'cast/m'cast addr in link-layer b'cast	3.3.6	x		-

INTERFACE:				
Allow transport layer to use all IP mechanisms	3.4	x		-
Pass interface ident up to transport layer	3.4	x		-
Pass all IP options up to transport layer	3.4	x		-
Transport layer can send certain ICMP messages	3.4	x		-
Pass spec'd ICMP messages up to transp. layer	3.4	x		-
Include IP hdr+8 octets or more from orig.	3.4	x		-
ICMP:				
Echo server	3.2.2.6	x		-
Echo client	3.2.2.6	x		x
Use specific-dest addr as Echo Reply src	3.2.2.6	x		x
Send same data in Echo Reply	3.2.2.6	x		x
Pass Echo Reply to higher layer	3.2.2.6	x		x
Reverse and reflect Source Route option	3.2.2.6	x		-
Use IP b'cast/m'cast addr in link-layer b'cast	3.3.6	x		-

7.3. TCP Layer

Feature	Section	Must	Must not	Implemented
Push flag				
ESEND call can specify PUSH	4.2.2.2			-
If cannot: sender buffer indefinitely	4.2.2.2		x	
If cannot: PSH last segment	4.2.2.2	x		x
Window				
Treat as unsigned number	4.2.2.3	x		x
Robust against shrinking window	4.2.2.16	x		-
Sender probe zero window	4.2.2.17	x		(x)
Allow window stay zero indefinitely	4.2.2.17	x		x
Sender timeout OK conn with zero wind	4.2.2.17		x	x
TCP Options				
Receive TCP option in any segment	4.2.2.5	x		x
Ignore unsupported options	4.2.2.5	x		x
Cope with illegal option length	4.2.2.5	x		-
Implement sending & receiving MSS option	4.2.2.6	x		x
Send-MSS default is 536	4.2.2.6	x		x

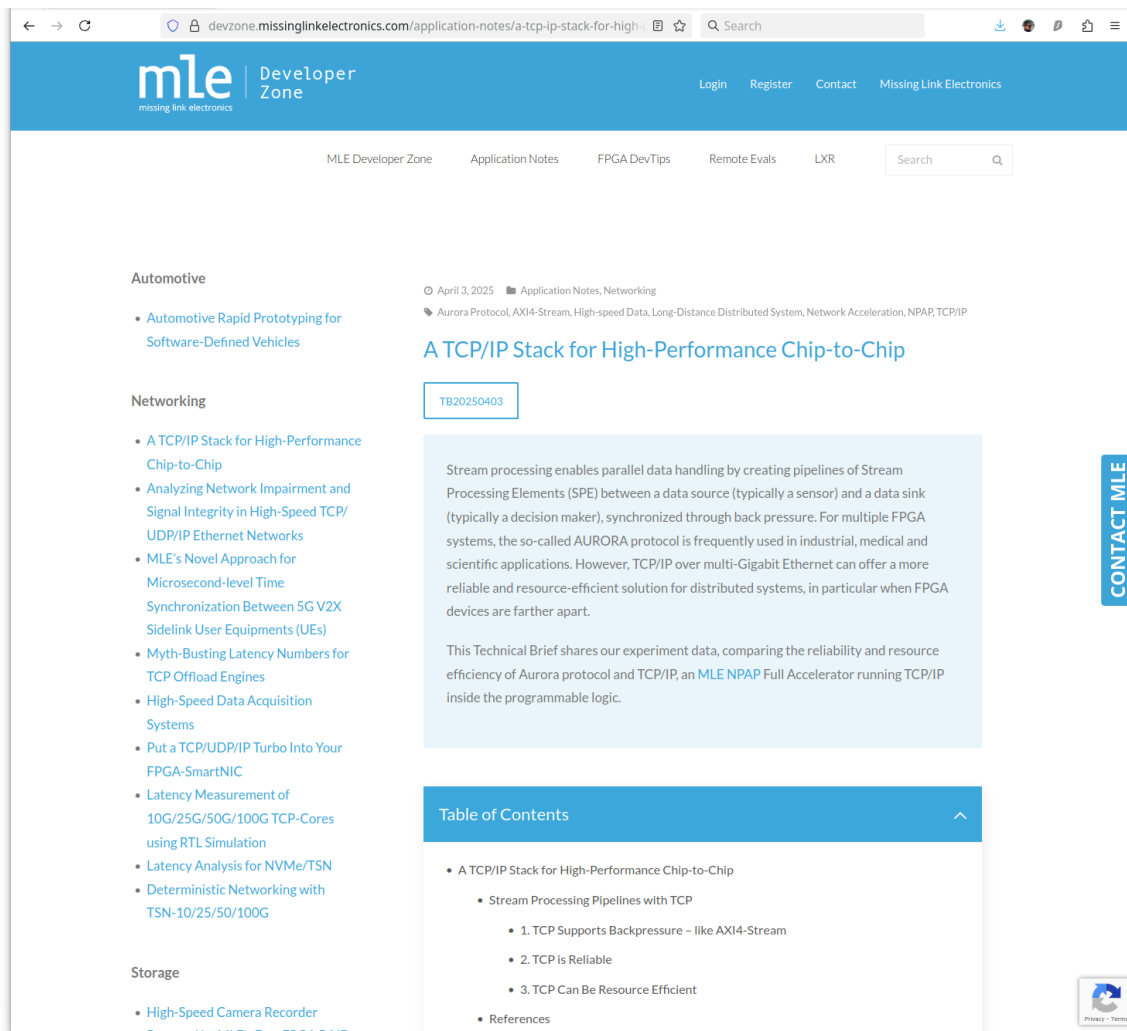
Calculate effective send seg size	4.2.2.6	x		x
TCP Checksums				
Sender compute checksum	4.2.2.7	x		x
Receiver check checksum	4.2.2.7	x		x
Use clock-driven ISN selection	4.2.2.9	x		x
Opening Connections				
Support simultaneous open attempts	4.2.2.10	x		-
SYN-RCVD remembers last state	4.2.2.11	x		-
Passive Open call interfere with others	4.2.2.18		x	-
Function: simultan. LISTENs for same port	4.2.2.18	x		
Ask IP for src address for SYN if necc.	4.2.3.7	x		x
Otherwise, use local addr of conn.	4.2.3.7	x		x
OPEN to broadcast/multicast IP Address	4.2.3.14		x	-
Silently discard seg to bcast/mcast addr	4.2.3.14	x		-
Closing Connections				
Inform application of aborted conn	4.2.2.13	x		x
In TIME-WAIT state for 2 x MSL seconds	4.2.2.13	x		x
Retransmissions				
Jacobson Slow Start algorithm	4.2.2.15	x		-
Jacobson Congestion-Avoidance algorithm	4.2.2.15	x		-
Karn's algorithm	4.2.3.1	x		-
Jacobson's RTO estimation alg.	4.2.3.1	x		-
Exponential backoff	4.2.3.1			
Generating ACK's:				
Process all Q'd before send ACK	4.2.2.20	x		x
Receiver SWS-Avoidance Algorithm	4.2.3.3	x		-

8. Developer Documentation

A comprehensive product design guide and a detailed description of the support IPs and their documentation is available under license:

- NPAP Product Guide
- NPAP Control Application (NCA) Product Guide
- TCP Control (TCA) and Demo (TDA) Application Product Guides
- UDP Demo Application (UDA) Product Guide
- Netperf Control Application (NCA) Product Guide
- Data Generator / Checker (DGC) Product Guide
- Impairment Generator Application (IPA) Product Guide

We also suggest to visit [MLE's Developer Zone](#) where you can find a growing list of Technical Briefs with expert insights into Networking.



The following pages are a preview of the documents you will receive when purchasing MLE NPAP.

Network Protocol Acceleration Platform

***Product Guide - npap-pg - version
2.8.0, NPAP IP Core version 2.8.0***

2025-09-30, g49ded3b



Contents

Acronyms	1
1 Document History	7
2 NPAP IP Changelog	9
2.1 Internal Submodules Changelog	16
2.1.1 NPAP Configuration Block Changelog	16
2.1.2 Priority Manager Configuration Block Changelog	17
2.1.3 Ethernet Diagnostics Block Changelog	17
2.1.4 Internet Protocol (IP) Diagnostics Block Changelog	17
2.1.5 User Datagram Protocol (UDP) Diagnostics Block Changelog	17
2.1.6 Transmission Control Protocol (TCP) Diagnostics Block Changelog	17
3 Summary	19
3.1 Core Benefits	19
3.2 Document Structure	19
4 Nomenclature	21
4.1 Protocol Related Nomenclature	21
4.2 IP Core Nomenclature	23
5 Introduction	25
5.1 Overview	25
5.2 Applications	27
5.3 Features	27
5.4 Limitations	29
6 Product Specification	31
6.1 Performance	31
6.2 Resource Usage	31
6.3 Attributes	31
6.4 Ports	36
6.4.1 Aggregated Ports	36
6.4.2 Clock, Reset, Configuration and Status Ports	37
6.4.3 AXI4-Lite interface ports	38
6.4.4 TCP Interfaces	39
6.4.5 UDP Interfaces	41
6.4.6 Network Interfaces	43
7 Designing with the Core	45
7.1 Buffer Structure	45
7.1.1 Buffer TCP	46
7.1.1.1 Buffer Size	46
7.1.1.2 Segment Size	46
7.1.1.3 Frame Number	47
7.2 Clocking	47
7.3 Reset	49



7.4	Network Side Interfaces	49
7.4.1	Network Side Receive Interface	49
7.4.2	Network Side Transmit Interface	49
7.5	User Side Interfaces	49
7.5.1	TCP	52
7.5.1.1	TCP Command Interface	52
7.5.1.2	TCP Status Interface	56
7.5.1.3	TCP Error Interface	57
7.5.1.4	TCP Data Interface RX	58
7.5.1.5	TCP Data Interface TX	58
7.5.2	DHCP	58
7.5.2.1	Allocate Network Address	58
7.5.2.2	Renew Network Address	60
7.5.2.3	Release Network Address	60
7.6	Diagnostics	60
7.6.1	Paging	61
7.6.1.1	Primary-Level Paging	61
7.6.1.2	Secondary-Level Paging	61
7.6.2	Diagnostics Acquisition Modes	61
7.7	Internal Submodules	63
7.7.1	ARP	64
7.7.1.1	ARP for UDP	65
7.7.1.2	ARP for TCP	65
7.7.2	ICMPv4	65
7.7.3	NPAP Configuration Block	65
7.7.3.1	Register map	65
7.7.4	Priority Manager	71
7.7.4.1	Register map	72
7.7.5	Ethernet Diagnostics Block	74
7.7.5.1	Register map	74
7.7.6	Internet Protocol version 4 (IPv4) Diagnostics Block	84
7.7.6.1	Register map	84
7.7.7	UDP Diagnostics Block	94
7.7.7.1	Register map	94
7.7.8	TCP Diagnostics Block	103
7.7.8.1	Register map	103
8	Example Design	119
A	Important Legal Information	121
B	References	123
	References	125

Network Protocol Acceleration Platform

***Product Guide -
data-gen-checker-product-guide -
version 1.1.0, Data Generator and
Checker IP Core version 2.0.0***

2025-09-30, gdf9f856



Contents

1 Document History	4
2 Data generator and checker IP Changelog	5
3 Summary	7
4 Product Specification	8
4.1 Attributes	8
4.2 Ports	10
5 Register Access	13
6 Register Map	14
6.1 0x000 VERSION	15
6.2 0x004 ID	15
6.3 0x008 CLOCK_FREQUENCY_PS	16
6.4 0x00C DATA_WIDTH	16
6.5 0x020 CONFIG	16
6.6 0x040 STATUS	16
6.7 0x040 CAPABILITY	17
6.8 0x100 TIMESTAMP_L	17
6.9 0x104 TIMESTAMP_H	17
6.10 0x400 GENERATOR_CONFIG	18
6.11 0x410 GENERATOR_DATA_PROCESSING	18
6.12 0x420 GENERATOR_SEED	19
6.13 0x440 GENERATOR_CONFIG_N_FRAMES	19
6.14 0x444 GENERATOR_CONFIG_N_BYTES	20
6.15 0x448 GENERATOR_CONFIG_N_PAUSE_BEATS	20
6.16 0x500 GENERATOR_STATUS	20
6.17 0x510 GENERATOR_ERROR_TYPE	20
6.18 0x600 GENERATOR_COUNTER_PACKET_L	21
6.19 0x604 GENERATOR_COUNTER_PACKET_H	21
6.20 0x610 GENERATOR_COUNTER_BEAT_L	21
6.21 0x614 GENERATOR_COUNTER_BEAT_H	21
6.22 0x620 GENERATOR_COUNTER_BYTE_L	22
6.23 0x624 GENERATOR_COUNTER_BYTE_H	22
6.24 0x800 CHECKER_CONFIG	22
6.25 0x810 CHECKER_DATA_PROCESSING	23
6.26 0x820 CHECKER_SEED	24
6.27 0x840 CHECKER_CONFIG_N_FRAMES	24
6.28 0x844 CHECKER_CONFIG_N_BYTES	24
6.29 0x848 CHECKER_CONFIG_N_PAUSE_BEATS	24
6.30 0x900 CHECKER_STATUS	25
6.31 0x910 CHECKER_ERROR_TYPE	25
6.32 0x940 CHECKER_ERROR_VALUE_LL	26
6.33 0x944 CHECKER_ERROR_VALUE_LH	26

2025-09-30, gdf9f856

Licensed by MLE

i

NPAP IP Product Guide - data-gen-checker-product-guide - version 1.1.0, Data Generator and Checker IP Core version 2.0.0



6.34	0x948 CHECKER_ERROR_VALUE_HL	26
6.35	0x94C CHECKER_ERROR_VALUE_HH	26
6.36	0x950 CHECKER_ERROR_EXPECTED_VALUE_LL	27
6.37	0x954 CHECKER_ERROR_EXPECTED_VALUE_LH	27
6.38	0x958 CHECKER_ERROR_EXPECTED_VALUE_HL	27
6.39	0x95C CHECKER_ERROR_EXPECTED_VALUE_HH	27
6.40	0xA00 CHECKER_COUNTER_PACKET_L	28
6.41	0xA04 CHECKER_COUNTER_PACKET_H	28
6.42	0xA10 CHECKER_COUNTER_BEAT_L	28
6.43	0xA14 CHECKER_COUNTER_BEAT_H	28
6.44	0xA20 CHECKER_COUNTER_BYTE_L	29
6.45	0xA24 CHECKER_COUNTER_BYTE_H	29
6.46	0xB00 LATENCY_FB_TX_FB_RX_L	29
6.47	0xB04 LATENCY_FB_TX_FB_RX_H	29
6.48	0xB00 LATENCY_FB_RX_LB_RX_L	30
6.49	0xB04 LATENCY_FB_RX_LB_RX_H	30
7	Designing with the Core	31
7.1	Overview	31
7.2	Connecting the IP-Core	31
7.3	Configuration	33
7.4	Data Generation and Checking Modes	34
7.4.1	Off	34
7.4.2	External	34
7.4.3	Beats / N Bytes (BNB)	34
7.4.4	LFSR	35
7.4.5	Timestamp	35
7.5	Starting a cycle	36
7.6	Detecting errors	36
7.7	Current data count	37
7.8	Measuring Round Trip Time	37
A	Important Legal Information	38
B	References	39

Network Protocol Acceleration Platform

***Product Guide -
impairment-generator-product-guide -
version 1.1.0, Impairment Generator
IP Core version 1.0.0***

2025-09-30, gdf9f856



Contents

1 Document History	4
2 IP Core Changelog	5
3 Summary	6
4 Product Specification	7
4.1 Attributes	7
4.2 Ports	8
5 Register Access	11
6 Register Map	12
6.1 0x000 VERSION	12
6.2 0x004 ID	13
6.3 0x008 CLOCK_FREQUENCY_PS	13
6.4 0x00C AXIS_DATA_WIDTH	13
6.5 0x020 CONFIG	13
6.6 0x060 CAPABILITY	14
6.7 0x080 ERROR	14
6.8 0x100 BER_LOW	14
6.9 0x104 BER_HIGH	15
6.10 0x400 GEN_BIT_ERROR_CNT	15
6.11 0x404 BIT_ERROR_CNT	15
7 Designing with the Core	16
7.1 Overview	16
7.2 Connecting the IP-Core	16
7.3 Operation	17
7.4 Configuration	18
7.4.1 Compile-Time Configuration (Generics)	18
7.4.2 Run-Time Configuration	19
A Important Legal Information	21
B References	22

Network Protocol Acceleration Platform

***Product Guide - npap-control-
application-product-guide - version
1.0.4, NPAP Control Application IP
Core version 2.0.3***

2025-09-30, gdf9f856



Contents

1 Document History	4
2 NPAP Control Application IP Changelog	5
3 Summary	6
4 Product Specification	7
4.1 Attributes	7
4.2 Ports	8
5 Register Access	11
6 Register Map	12
6.1 0x000 VERSION	12
6.2 0x004 ID	13
6.3 0x008 CLOCK_FREQUENCY_PS	13
6.4 0x00C DATA_WIDTH	13
6.5 0x020 CONFIG	13
6.6 0x040 STATUS	14
6.7 0x200 LOCAL_IP	14
6.8 0x204 GATEWAY_IP	14
6.9 0x208 SUBNET_MASK	14
6.10 0x20C MAC_LSB	15
6.11 0x210 MAC_MSB	15
6.12 0x400 LOCAL_DHCP_IP	15
6.13 0x404 DHCP_TRY_COUNTER	15
A Important Legal Information	17
B References	18
References	19

Network Protocol Acceleration Platform

***Product Guide - tcp-command-
application-product-guide - version
1.1.3, TCP Command Application IP
Core version 4.0.2***

2025-09-30, gdf9f856



Contents

1 Document History	4
2 IP Core Changelog	5
3 Summary	6
4 Product Specification	7
4.1 Attributes	7
4.2 Ports	8
5 Register Access	12
6 Register Map	13
6.1 0x000 VERSION	13
6.2 0x004 ID	14
6.3 0x008 CLOCK_FREQUENCY_PS	14
6.4 0x020 CONFIG	14
6.5 0x040 STATUS	14
6.6 0x200 START_DELAY	15
6.7 0x204 STOP_DELAY	15
7 Designing with the Core	16
7.1 Reset	16
7.2 TCP Command Interface	16
A Important Legal Information	18
B References	19
References	20

Network Protocol Acceleration Platform

***Product Guide -
tcp-demo-application-product-guide -
version 1.1.6, TCP Demo Application IP
Core version 4.0.3***

2025-09-30, gdf9f856



Contents

Acronyms	1
1 Document History	4
2 TCP Demo Application IP Changelog	5
3 Summary	7
4 Product Specification	8
4.1 Attributes	8
4.2 Ports	9
5 Register Access	13
6 Register Map	14
6.1 0x000 VERSION	14
6.2 0x004 ID	15
6.3 0x008 CLOCK_FREQUENCY_PS	15
6.4 0x00C DATA_WIDTH	15
6.5 0x020 CONFIG	15
6.6 0x040 STATUS	16
6.7 0x044 TCP_STATUS	16
6.8 0x080 TCP_ERROR	16
6.9 0x100 DATA_PROCESSING	17
6.10 0x200 LOCAL_PORT	17
6.11 0x204 REMOTE_PORT	18
6.12 0x208 REMOTE_IP	18
6.13 0x300 MINIMUM_RTO	18
6.14 0x304 MAXIMUM_RTO	19
6.15 0x308 STATUS_RTO	19
7 Designing with the Core	20
7.1 Overview	20
7.2 Data Path Modes	20
7.2.1 Discard	20
7.2.2 Loopback	20
7.2.3 External	20
A Important Legal Information	21
B References	22
References	23

Network Protocol Acceleration Platform

***Product Guide -
udp-demo-application-product-guide -
version 1.1.2, UDP Demo Application IP
Core version 2.0.1***

2025-09-30, gdf9f856



Contents

1 Document History	4
2 UDP Demo Application IP Changelog	5
3 Summary	6
4 Product Specification	7
4.1 Attributes	7
4.2 Ports	8
5 Register Access	12
6 Register Map	13
6.1 0x000 VERSION	13
6.2 0x004 ID	14
6.3 0x008 CLOCK_FREQUENCY_PS	14
6.4 0x00C DATA_WIDTH	14
6.5 0x020 CONFIG	15
6.6 0x040 STATUS	15
6.7 0x100 DATA_PROCESSING	15
6.8 0x200 LOCAL_PORT	16
6.9 0x204 REMOTE_PORT	17
6.10 0x208 REMOTE_IP	17
6.11 0x400 COUNTER_RX_CONFIG	17
6.12 0x410 COUNTER_RX_PACKET_L	17
6.13 0x414 COUNTER_RX_PACKET_H	18
6.14 0x420 COUNTER_RX_BEAT_L	18
6.15 0x424 COUNTER_RX_BEAT_H	18
6.16 0x430 COUNTER_RX_BYTE_L	19
6.17 0x434 COUNTER_RX_BYTE_H	19
6.18 0x500 COUNTER_TX_CONFIG	19
6.19 0x510 COUNTER_TX_PACKET_L	19
6.20 0x514 COUNTER_TX_PACKET_H	20
6.21 0x520 COUNTER_TX_BEAT_L	20
6.22 0x524 COUNTER_TX_BEAT_H	20
6.23 0x530 COUNTER_TX_BYTE_L	21
6.24 0x534 COUNTER_TX_BYTE_H	21
7 Designing with the Core	22
7.1 Overview	22
7.2 Data Path Modes	22
7.2.1 Discard	22
7.2.2 Loopback	22
7.2.3 External	23
A Important Legal Information	24

9. Changelog

The following lists MLE's engineering changelog for NPAP. With the release of NPAP v2.2.0 the development cycle has changed from 1.x to 2.x.



Version 2.4.5 uses the least amount of resources and includes no diagnostics.

Version 2.5.0 and onwards requires VHDL-2008. VHDL-2008 is not fully supported in Quartus Prime Standard, for example. Please inquire with any questions regarding tool support.

9.1. NPAP Version 2 Development

- 20251031
 - ERD v3.4.7
 - all
 - bump all ERD to v3.4.7
 - amd_zcu111_source_full_25g_sync_async_npap2
 - #7289 - ERD deprecated
 - amd_zcu111_source_full_25g_async
 - #7289 - new ERD 3 variant
 - amd_zcu111_source_full_10g_sync_async_npap2
 - #7289 - enable sync NPAP variant by default
- 20253009
 - NPAP Kernel v2.8.0
 - GENERAL
 - #7246 - fix synthesis error in case G_TCP_CONNECTIONS is set to 0
 - DIAGNOSTICS
 - #7362 - implement diagnostic results reset on snapshot and on window capture
 - TCP
 - #5662 - adapt window scaling algorithm to match the compile time specified buffer sizes as described in RFC 7323
 - #7440 - fix RFC 7323 conformity on window scale option in <SYN,ACK> segment

- #7457 - fix rare RX window underflow when an unaligned payload nearly fills the window
 - ERD v3.4.6
 - #7246 - fix UDP only ERD generation
 - bump all ERDs to v3.4.6
- 20250820
 - Updated documentation and datasheet
- 20250731
 - NPAP Kernel v2.7.0
 - DIAGNOSTICS
 - #7147 - Implement diagnostics registers for TCP TX RTO min/max value
 - #7148 - Implement memory-mapped reset for diagnostics results reset (either all session or single session only)
 - ERD v 3.4.5
 - bump all ERDs to v3.4.5
- 20250615
 - NPAP Kernel v2.6.1
 - GENERAL
 - #6836 - split FIFO implementation into separate synchronous FIFO and asynchronous FIFO modules
 - TCP
 - #6927 - fix half-closed TCP session handling
 - UDP Demo Application v2.0.1
 - #7298 - fix tvalid being asserted one clock cycle too late in loopback mode
 - Netperf v4.0.0
 - #6836 - split FIFO implementation into separate synchronous FIFO and asynchronous FIFO modules
 - #6927 - update error handling to work with NPAP v2.6.1 update
 - ERD v3.4.4
 - bump all ERDs to v3.4.4
- 20250531
 - NPAP Kernel v2.6.0
 - DIAGNOSTICS

- #7145 - improve diagnostics: add window control SW read active bit, remove shadow registers for TCP COUNT registers, reduce number of register stages
 - #7146 - change diagnostics window duration load
 - ERD v3.4.3
 - bump all ERDs to v3.4.3
- 20250430
 - ERD v3.4.2
 - bump all ERDs to v3.4.2
 - lattice_avant_g_source_full_10g_sync
 - #6987 - add new Lattice Avant G ERD
- 20250402
 - NPAP Kernel v2.5.0
 - DIAGNOSTICS
 - #6941 - add diagnostics register space for reporting events and counters thereof
 - ERD v3.4.1
 - bump all ERDs to v3.4.1
- 20250320
 - NPAP Kernel v2.4.5
 - GENERAL
 - #7074 - fix internal NPAP version register
- 20250228
 - ERD v3.4.0
 - bump all ERDs to v3.4.0
 - #6745 - add Impairment Generator Application (IPA)
 - IP
 - add IPA v1.0.0
- 20250130
 - ERD v3.3.0
 - bump all ERD to v3.3.0
 - #6291 - update ERD setting and add application loopback
 - amd_u55c_source_full_25g_async_npap4
 - #6521 - add new Alveo U55C ERD

- IP
 - update DGC to v2.0.0
 - update NCA to version 1.0.4
 - update UDA to version 2.0.0
- 20241220
 - NPAP Kernel v2.4.4
 - GENERAL
 - #6731 - add fifo variant for Microchip FPGA
 - ERD v3.2.2
 - microchip_mpf300_eval_kit_source_full_10g_sync
 - #6731 - add HHI fifo variant for Microchip
 - intel_axe5_eagle_source_full_10g_async
 - #6685 - add new Agilex 5 ERD
- 20241130
 - ERD v3.2.1
 - amd_te0950_source_full_25g_async
 - #6734 - update ERD 8 (TE0950) to non es silicon
- 20241028
 - NPAP Kernel v2.4.3
 - TCP
 - #6479 - fix possible TCP session lock in half closed state if priority manager is enabled
 - #6613 - increase internal debug support by adding new debug signals
 - ERD v3.2.1
 - #5739 - add new register to top level register space
 - amd_zcu102_source_full_10g_sync
 - #5739 - add top level info block to ER
 - microchip_mpf300_eval_kit_source_full_10g_sync
 - #6581 - add XFCP
- 20240930
 - NPAP Kernel v2.4.2
 - TCP

- #6593 - fix consistency of TCP TX ram type implementation
 - #6645 - fix session stall corner case when the congestion window is full, the last packet does not get ACKed, and no more packets are written into the TX buffer
 - #6649 - fix session stall corner case when an ACK for a retransmitted packet is lost, and the TX buffer on the ACK side is empty
- ERD v3.1.6
 - #6119 - change UDP default datagram size to 1472 byte
 - amd_zcu111_source_full_25g_sync_async_npap2
 - #6634 - fix DGC default setting
 - amd_zcu111_source_full_100g_sync
 - #6634 - fix DGC default setting
 - IP
 - update TDA to v4.0.4
 - update MAC to v1.2.0
- 20240813
 - NPAP Kernel v2.4.1
 - GENERAL
 - #6590 - rewrite FIFO constraints from xdc to tcl
- 20240731
 - NPAP Kernel v2.4.0
 - GENERAL
 - #5878 - update generic fifo constraints
 - #6512 - multiple internal coding style fixes and signal clean-ups
 - #6592 - fix npap_configuration_block register reset
 - TCP
 - #4682 - rework of receive window handling
 - #6494 - change window update calculation naming
 - #6513 - add support for receive side Keep-Alive segments
 - #6514 - add support for receive side Zero-Window-Probe
 - ERD v3.1.5
 - IP

- update DGC to v1.4.0
- 20240630
 - ERD v3.1.4
 - microchip_mpf300_eval_kit_source_full_10g_sync
 - #5877 - replace reset stretch and sync module
- 20240430
 - NPAP Kernel v2.3.4
 - GENERAL
 - #6413 - fix bus scheduler early release hang
 - ERD v3.1.3
 - #6248 - DGC modes are configurable in top level
 - amd_zcu102_source_full_1g_sync_async_npap2
 - #6063 - add 1G ERD
 - amd_zcu102_source_full_1g_10g_sync_async_npap4
 - #6063 - deprecate ERD
- 20240331
 - ERD v3.1.2
 - amd_te0950_source_full_25g_async
 - add TE0950 25 ERD
- 20240311
 - NPAP Kernel v2.3.3
 - GENERAL
 - #6399 - fix possible UDP checksum mismatch
 - #6377 - fix UDP only design
- 20240229
 - NPAP Kernel v2.3.2
 - GENERAL
 - #4909 - fix bus scheduler fairness
 - #5987 - fix G_PRIORITY_WIDTH synthesis error
 - #6243 - fix G_ENABLE_UDP = 0 timing loop
 - ERD v3.1.1
 - #6087 - fix UDP session - 1 configuration error
 - #6093 - fix Netperf UDP test

- #6201 - fix default value settings
 - #6235 - support no DGC operation
 - #6282 - support UDP and TCP only operation
 - #6298 - fix local port not assigned any more
 - amd_au200_source_full_25g_async
 - add AU200 25G ERD
 - amd_zcu111_source_full_100g_sync
 - add ZCU111 100G ERD
- IP
 - update Netperf to v3.0.2
 - update DGC to v1.1.5
 - update TCA to v4.0.3
- 20240101
 - NPAP Kernel v2.3.1
 - GENERAL
 - #6005 - update IP packaging to support Versal FPGA
 - ERD v3.1.0
 - microchip_mpf300_eval_kit_source_full_10g_sync
 - build with obfuscated MAC license
 - +1 commit to ERD release tag
 - intel_npac_ketch_source_full_10g_async_npap2
 - add NPAC Ketch 10G ERD
 - intel_npac_ketch_source_full_10g_async_npap3
 - deprecated
 - IP
 - update Netperf to v3.0.1
 - update MAC 10/25 G to 1.1.2 (some ERDs)
 - update DGC to v1.1.4
 - update NCA to v1.0.2
 - update TCA to v4.0.2
 - update TDA to v4.0.2
 - update UDA to v1.1.1
- 20231129

- NPAP Kernel v2.3.0
 - GENERAL
 - #5743 - add AXIL-Lite control and status interface (NPAP Configuration Block)
 - #5948 - update Priority Manager to new AXI4-Lite structure
- 20231031
 - ERD v3.0.5
 - #5891 - update generic and clock structure
 - microchip_mpf300_eval_kit_source_full_10g_sync
 - add MPF300 10G ERD
- 20230930
 - NPAP Kernel v2.2.5
 - GENERAL
 - #5963 - update timeouts to be clock depended
- 20230919
 - NPAP Kernel v2.2.4
 - TCP
 - #5954 - harden TCP against lost ACK
- 20230831
 - NPAP Kernel v2.2.3
 - GENERAL
 - #5880 - remove unused reset stretch modules
 - ERD v3.0.3
 - amd_zcu102_source_full_1g_10g_sync_async_npap4
 - #5899 - fix AXI-L connectivity
 - amd_zcu111_source_full_25g_sync_async_npap2
 - #5899 - fix AXI-L connectivity
 - #5908 - increase system clock
 - intel_npac_ketch_source_full_10g_async_npap3
 - #5880 - update constraints and reset structure
- 20230731
 - NPAP Kernel v2.2.2

- TCP
 - #5805 - fix out of order received ACK handling
- ERD v3.0.2
 - amd_zcu102_source_full_1g_10g_sync_async_npap4
 - #5784 - add zcu102 1G / 10G ERD
 - amd_zcu111_source_full_25g_sync_async_npap2
 - #5816 - add zcu111 25G ERD
- 20230630
 - NPAP Kernel v2.2.1
 - UDP
 - #5707 - fix UDP meta data handling
 - ERD v3.0.1
 - amd_zcu102_source_full_10g_sync
 - #5814 - fix generic names
 - IP
 - #5815 - update NPAP v2.2.1
 - #5859 - update DGC v1.1.2
 - #5860 - update NCA v1.0.1
 - #5861 - update Netperf v3.0.0
- 20230531
 - NPAP Kernel v2.2.0
 - GENERAL
 - #5652 - internal code merge and structure update (technically NPAP v2.2.0 is identically with v2.1.2, the code history has a different merge / rebase history)
 - ERD v3.0.0
 - amd_zcu102_source_full_10g_sync
 - #4775 - add ZCU102 10G ERD
 - intel_npac_ketch_source_full_10g_async_npap3
 - #5708 - add NPAC Ketch 10G ERD
 - IP
 - #5760 - update Netperf v2.0.0
 - #5761 - update UDA v1.1.0

- #5763 - update DGC v1.1.1
- 20230411
 - NPAP Kernel v2.1.2
 - TCP
 - #5562 - add support for partially ACKed packages
- 20230331
 - NPAP Kernel v2.1.1
 - version changes not included
 - sync with NPAP 1.10.1
 - IPGUI
 - #5560 - remove AXI4-S MAC TDATA width setting from customisation GUI
 - #5569 - buffer size: make IP customisation GUI default the same as HDL default
 - GENERAL
 - #5668 - clock are now associated to the AXI4-L interface
- 20230306
 - NPAP Kernel v2.1.0
 - TCP
 - #5554 - fix signal overflow on TCP transmit controller
 - #5555 - fix signal overflow on TCP receive buffer calculation signal
- 20220822
 - NPAP Kernel v2.0.1
 - version changes not included
 - NPAP 1.9.2 to 1.10.0
 - GENERAL
 - #4836 - add priority scheduler
- 20220715
 - NPAP Kernel v2.0.0
 - version changes not included
 - NPAP 1.9.2 to 1.10.0
 - GENERAL

- #2854 - remove NPAP application CDCs from code base
- #3944 - NPAP Performance Enhancement and Clean Up (parts)
- #4398 - remove 8bit data path
- #4399 - remove dma code fragments
- #4834 - add QoS interfaces and generics
- #4835 - add register interface for priority settings
- ETHERNET
 - #4601 - change MAC interface to standard 128 Bit AXIS

9.2. NPAP Version 1 Development

Not recommended for new design starts!

- 1.10.1 (20230331)
 - TCP
 - #5554 - fix overflow on allow payload size register
 - #5557 - fix minimum value for G_TCP_RX_MAX_FRAME_NUMBER and G_TCP_TX_MAX_FRAME_NUMBER
 - #5639 - fix TCP window calculation after window scale is set
- 1.10.0 (20220930)
 - TCP
 - #4389 - add TCP Cmd TcpCmdSetTcpPsh
 - #4888 - fix TCP session handling with same destination port
 - #4916 - fix TCP splitter generating overlong packages
 - #5007 - fix bug where changing RTO values could lead to TCP Cmd interface to hang
- 1.9.2 (20220718)
 - GENERAL
 - #4805 - TCP session do not transfer data reliably on first connection (Microchip only)
- 1.9.1 (20220503)
 - GENERAL
 - #4741 - fix wrong license header
- 1.9.0 (20220430)
 - TCP
 - #4700 - add per TCP session configurable buffer sizes and configurable MSS
 - UDP
 - #4700 - add new parameter G_TX_MAX_DATAGRAM_SIZE
- 1.8.0 (20220331)
 - ETHERNET
 - #4606 - add padding for frames smaller than 60 Bytes
 - TCP
 - #4609 - remove maximum TCP Session limit

- UDP
 - #4557 - fix length assignment in UDP Interface Adapter
- 1.7.1 (20211220)
 - GENERAL
 - #3951 - add missing reset signal
 - #4412 - fix buffer generic ranges
 - #4416 - fix subnet mask assignment for no UDP setup
 - TCP
 - #4367 - fix tcp space available calculation for buffer sizes above 64KB
- 1.7.0 (20211101)
 - GENERAL
 - #3951 - remove unused altera_attribute
 - #4139 - allow set off RTO values in TCP session establish state
- 1.6.2 (20210913)
 - GENERAL
 - #3981 - remove unused altera_attribute
- 1.6.1 (20210801)
 - UDP
 - #3969 - fix use of ceil function
- 1.6.0 (20210701)
 - GENERAL
 - #3916 - rename / fix generic names
- 1.5.3 (20210301)
 - UDP
 - #3549 - fix bus scheduler bus handling
- 1.5.2 (20201101)
 - TCP
 - #3091 - fix ack command fifo interface handling
- 1.5.1 (20201001)
 - ARP
 - #2924 - fix handshake between ARP and bus scheduler
 - TCP

- #2539 - fix possible wrong MAC usage in multi session configuration
 - #2923 - fix data acceptance criteria from application
- 1.5.0 (20200901)
 - GENERAL
 - #2254 - change 100ms clock not generated inside NPAP wrapper
 - #2848 - change application clocking by removing TCP and UDP application clock
 - IP
 - #2239 - change default configuration for IP filter, now enabled
 - DHCP
 - #2858 - change increase DHCP usability by adding try counter, valid signal and new timeout behaviour
 - #2737 - fix DHCP lease calculation
 - TCP
 - #2703 - fix payload length update behaviour in transmit controller
 - #2727 - fix acknowledgment number update behaviour in transmit controller
 - #2831 - fix FSM handshake in transmit controller which could lead to TCP session freeze
- 1.4.9 (20200702)
 - IGMP
 - #2704 - change disable IGMP per default
 - #2496 - fix compiler warning about latch implementation
 - TCP
 - #2706 - change TCP tx data path in asynchronous mode
 - #2627, #2688, #2693, #2701, #2702 - fix TCP tx splitter, rework after multiple bugs
 - #2692 - fix used TCP tx splitter generic
 - #2711 - fix transmit controller fsm reset generation
- 1.4.8 (20200430)
 - TCP
 - #2477 - fix TCP retransmission buffer delete handling
- 1.4.7 (20200331)

-
- TCP
 - #2180 - change TCP tx splitter to work with byte granularity
 - #2097 - fix TCP multi session reset synchronization
 - #2339 - fix TCP application reset clock domain crossing
 - 1.4.6 (20200303)
 - GENERAL
 - #2469 - fix default gateway IP address usage
 - TCP
 - #2468 - add register stage to TCP TX application interface to ease timing on Virtex 6
 - 1.4.5 (20200220)
 - GENERAL
 - #2449 - fix Xilinx ISE 14.7 workflow
 - 1.4.4 (20200213)
 - TCP
 - #2086 - fix retransmission lockup
 - #2112 - fix fsm lockup in transmit controller
 - 1.4.3 (20200116)
 - ARP
 - #2179 - fix ARP cache ip address lookup
 - 1.4.2 (20200113)
 - GENERAL
 - #2294 - change delivered IP XACT constraint file
 - 1.4.1 (20200107)
 - TCP
 - fix data type and IP core GUI handling of TCP sequence number initialization
 - 1.4.0 (20191126)
 - GENERAL
 - #1930 - add AXI4-Stream TCP application interface
 - #2151 - add customized block design configuration gui
 - #2166 - add example constrain file to IP XACT packaging
 - #2148 - fix block design gui NPAP name generic
 - TCP

- #1939 - fix space available calculation which lead to duplicated data beat
 - #2181 - fix TCP tx splitter timeout
- 1.3.0 (20191002)
 - GENERAL
 - #1682 - add IP XACT TCP/UDP wrapper and packaging
- 1.2.0 (20190920)
 - GENERAL
 - #2075 - add packaging infrastructure for TCP source code release
- 1.1.0 (20190705)
 - ARP
 - #1719 - add new ARP cache size generic
 - #1698 - fix internal race condition during initialization
 - UDP
 - #1720 - fix retry mechanism on failed ARP lookup
- 1.0.0 (20181023)
 - GENERAL
 - #1320 - add NPAP to MLE Vivado build toolchain
 - #1326 - add IP XACT UDP wrapper and packaging
 - #1133 - fix bus scheduler grant timeout
 - ETHERNET
 - #1323 - add 64 and 128 bit AXI4-Stream interface option
 - IP
 - #1328 - fix IP header decoder data valid calculation for payloads of 1 to 3 byte
 - UDP
 - #1322 - add AXI4-Stream UDP application interface
 - #1324 - add new generic to disable UDP TX aligner
 - #1132 - fix UDP tx fifo write count calculation
 - #1133 - fix UDP header encoder fsm timeout for ARP
 - #1327 - fix UDP throughput bottleneck for payload sizes less than 100 byte
 - #1330 - fix corrupt UDP data multiplexing for zero TCP connections

MLE Contact Info

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

Missing Link Electronics GmbH
Industriestraße 10
89231 Neu-Ulm
Tel. +49 731 141149-0

Email: sales-web@mlecorp.com

<http://www.missinglinkelectronics.com>



Founded in 1949, the German Fraunhofer-Gesellschaft undertakes applied research of direct utility to private and public enterprise and of wide benefit to society. With a workforce of over 23,000, the Fraunhofer-Gesellschaft is Europe's biggest organization for applied research, and currently operates a total of 67 institutes and research units. The organization's core task is to carry out research of practical utility in close cooperation with its customers from industry and the public sector.

[Fraunhofer HHI](#) was founded in 1928 as "Heinrich-Hertz-Institut für Schwingungsforschung" and joined in 2003 the Fraunhofer-Gesellschaft as the "Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut". Today it is the leading research institute for networking and telecommunications technology, "Driving the Gigabit Society" .