Evaluation of SmartNIC Devices for use in Trigger and Data Acquition Systems

Andreas Klavenes Berg, Adam Abed Abud, Enrico Gamberini, Anders Hjelm Poulsen, Evgueny Khartchenko, Giovanna Lehmann Miotto, Graham McKenzie, Roland Sipos, Jeff Zheng

***Abstract*—This paper presents an evaluation of SmartNIC devices in the context of Trigger and Data AcQquisition (TDAQ) systems. SmartNIC devices represent an emerging technology whose aim is to offload network tasks and infrastructure control plane software from the CPU. Such devices are particularly relevant for TDAQ systems where high rates in the orders of TB/s are produced in large detectors such as the DUNE experiment. In this context, the potential use-case of SmartNICs is to perform a quasi real-time reduction of the incoming data streams by identifying only the interesting signals. The goal is to sustain a number of ∼10 Gb data streams aggregated on 100 Gb interfaces, and transmit the results to the host machine. An application was developed to provide a testing environment, measuring the achieved throughput in two cases. In the first case, only the total throughput is considered, and the workload is evenly distributed across the available hardware. In the second case, a constraint of processing a number of discrete data streams is added. The application was shown to handle up to ∼130 Gbps of incoming data when distributing the workload evenly on the available hardware resources of 8 CPU cores. In this contribution, we show the testing results and optimisations and hardware tuning of the technology when performing a workload suitable for TDAQ applications.**

***Index Terms*—Data Acquisition, DPDK, HPC, SmartNIC**

I. Introduction

With the growth of cloud computing, data centres and cloud service providers, this industry has taken a leading role in the development of integrated high performance compute and network equipment. This role was previously held by academia and high energy physics (HEP) experiments like those conducted at CERN, for which custom, state-of-the-art electronics are often developed for Trigger and Data Acquisition (TDAQ) systems, and other specialised tasks. One such device is the custom designed FELIX FPGA-based PCIe card, which was developed within the HEP community, and used by experiments like ATLAS and ProtoDUNE [1].

Paper submitted for review on 19 April 2024.

This work, carried out in the academic year 2022/2023, was made possible thanks to the support of Silicom and Intel providing and hosting the test devices.

A. K. Berg is a master student with the Department of Computer Science at the Norwegian University of Science and Technology, this work was done during a technical studentship at CERN.

A. Abed Abud, E. Gamberini, G. Lehmann Miotto, R. Sipos are with CERN, Geneva, Switzerland.

E. Khatchenko is with Intel.

G. McKenzie is with Altera, an Intel company.

A. Hjelm Poulsen is with Silicom Denmark.

J. Zheng is with Silicom Ltd.

Two areas where cloud industry and HEP are overlapping are networking and high performance computing. Network Interface Cards (NICs) have become very advanced, increasingly offloading networking and infrastructure related tasks from the host CPU. First, as commonly seen in consumer NICs, already widely common tasks such as basic network functions, virtualisation, compression and encryption, are offloaded to the NIC. Next, the term SmartNIC appears when limited programmability is introduced [2]. Advancing further, manufacturers such as both Intel and NVIDIA have started putting increasingly complex processing power on the boards, including FPGAs and general purpose CPUs, making the advanced NIC itself a fully programmable machine. Intel and NVIDIA are marketing this new generation of devices as IPU and DPU, respectively.

The motivation for Intel's Infrastructure Processing Unit (IPU) and NVIDIA's Data Processing Unit (DPU) comes from the observation in cloud service providers that a lot of CPU resources are being used on infrastructure tasks (load balancing, virtualisation, containerisation, etc.). By isolating the infrastructure software to the IPU's on-board processor (SoC), the entire host machine can be leased to customers [3]. In this work, the focus is mainly on testing IPUs from Intel, although an NVIDIA® Bluefield®-2, running an 8 core Arm® Cortex®-A72 processor, was also used to familiarise with the device type. NVIDIA has since then also released the Bluefield®-3, with increased processing and data transfer capabilities, and a roadmap for the Bluefield®-4, both of which could be very interesting devices to consider as well is the future.

FELIX is conceptually similar to a SmartNIC, but has unique features such as the ability to handle a large number of independent point-to-point "low-speed" (∼10 Gbps) links (up to 48) [4]. This feature is necessary when connecting to specialised radiation-hard ASICs with custom protocols. On the other hand, for HEP detectors that can support standard protocols, such as Ethernet, it is now possible to aggregate many links through a switching layer and use multi-100G-NICs for data reception.

At CERN, devices like the IPU, which have been developed for cloud infrastructure and network offload, are being considered for use for the processing workloads of TDAQ systems. An important consideration, in order to best exploit the power of the IPU, is to understand the similarities and the differences of the use-cases – how the devices' properties work as advantages

and as weaknesses for this purpose.

The processor on an IPU is obviously not expected, nor intended, to perform better than a host CPU in a server environment [5]. Its main purpose in a data centre is to offload networking tasks and infrastructure control plane software to provide isolation, so the host CPU can be better utilised by revenue-generating core applications. This means that data processing capabilities when running a filtering application might be the main limitation of the device for a DAQ use-case. However, also for a DAQ application, the advantage of the on-board processor is its location right on the datapath. This allows moving parts of the processing of incoming data out of the host, and can free up resources on the host for other parts of DAQ, such as assembly of events from various links. The on-board processor is also well located to work with the FPGA on the task of transforming and processing a large data stream.

The next section will introduce a potential use-case in the Deep Underground Neutrino Experiment (DUNE) [6], and performance requirements that the testing is based on, followed by an overview of hardware specifications in Section III, and the test application in Section IV. Section V presents the tests with results, together with a discussion of the optimisations carried out to reach those results, and of possible avenues for further improvements.

II. Use-case performance requirements

This work uses the DUNE DAQ readout system [7] as a case study to demonstrate how an IPU may be used for reception of, and feature extraction from data generated by a detector. Features are extracted by processing the full complement of data and identifying time windows and electronics channels exhibiting activity not compatible with electronics noise: this process is known as hit-finding and results in the forming of trigger primitive information, TPG (Trigger Primitive Generation) [8]. Trigger Primitives (TP) are generated whenever hits are detected, typically at 0.005% of the incoming data frequency, and are then used to signal what data to transmit to storage and higher level analysis. Data frames are buffered for 10 seconds whilst waiting for the result of TP generation and analysis: this long time is not driven by processing constraints but by the sparse arrival of the signals that must be evaluated together in order to detect a Physics event that is worth being captured. In a design with an IPU, the data buffer might be on the host machine, and the TPs are generated from the processor on the IPU and then sent from the IPU to the host when ready.

Each IPU is expected to process data from 2 detector units of 3072 electronics channels each that are producing 14 bit values at a rate of about 2 MHz. This results in an overall data throughput requirement of 172 Gbps that could be injected into an IPU with either 2×100 G or 1×200 G Ethernet links. The electronics channels are handled by so-called Warm Interface Boards (WIB) [7]. Each WIB manages 512 channels and outputs data via two 10 Gbps ports over the Ethernet UDP protocol. The modularity of the detector is such that 6 WIBs are reading out one detector unit. There are 160 detector units to form one of the DUNE far detector modules and there will be in total 4 far detector modules. If the performance goal of the IPU can be achieved, a total of 80×4 IPUs would be needed to readout the full DUNE far detector.

The data used for testing comes in frames of a specific format called WIB2, with a size of 472 B (24 B header added) at a rate of 2 MHz. The frames are sent in chunks of 12, called superchunks, making the payload size of the incoming packets 472 B $\times$ 12 = 5664 B, for a rate of 7.55 Gbps per WIB output link. Including the WIB2's header overhead, the actual throughput requirement for handling 2 detector units then becomes $2\times12\times7.55$ = 182 Gbps, rather than the 172 Gbps of data. This is the format used to readout the detector using the FELIX card and point-to-point links. In order to be handled by the TPG algorithm, the first step after receiving a WIB2 frame is to unpack it. The impact of the unpack stage is discussed in Section V-C. After then running the TPG, any TPs generated, at an expected rate of $\sim$ 100 Hz per channel, need to be transmitted to the down stream TDAQ components.

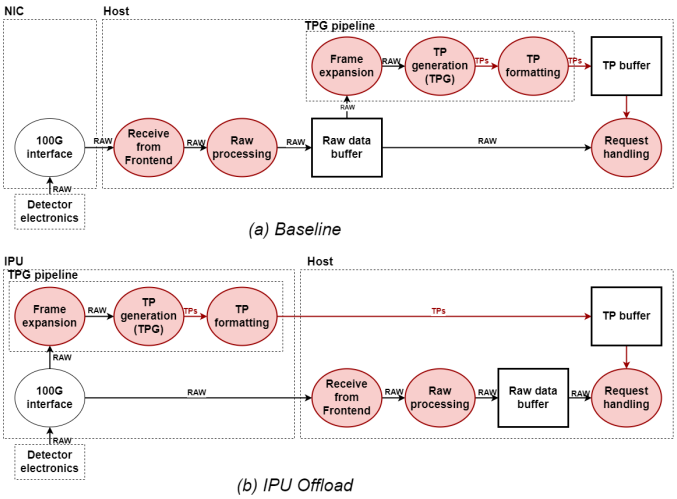

Figure 1: Dataflow diagram for the data processing when executed on IPU compared to full processing performed on the host. Red ellipses are software processes, and bold rectangles are data buffers. Subfigure a shows a simplified schematic of the baseline design. Subfigure b shows the modified design where the TPG filtering task has been relocated to the IPU's on-board processor, and the raw data stream is duplicated on the on-board FPGA.

Figure 1 presents a simplified view of the baseline design of the DUNE TDAQ system, where the entire workload is contained on a conventional server, compared to a solution employing an Intel IPU for offloading the TPG function. In this scenario, the IPU's task is to perform packet processing on a network level, routing the incoming data both to the on-board processor for processing, and to the host for buffering.

Recent developments have been carried out to change the WIB data transmission to use the Ethernet UDP protocol. For this the format has been optimised to send larger data

packets at lower rates. In particular each data stream now groups 64 channels over 64 time samples, resulting into data packets of 7.2 kB sent at 30.5 kHz. One WIB sends 8 such data streams in parallel over two 10 G links. However, for the measurements described in this report new data samples were not yet available.

The version of the TPG algorithm used for this study is heavily simplified, and though it forms a valid proof of concept, it represents only a subset of the required workload for a real-world scenario. This should be taken into consideration when analysing results. Sufficient headroom would be required for considering an IPU for the engineering of the DUNE TDAQ system.

III. Hardware specification

Multiple devices were made available during this project. Early tests were carried out on an Intel® IPU C5000X-PL [9], provided by Silicom. The card has two 25 Gbps interfaces, an FPGA for packet routing and other optional functions, and an on-board processor featuring a 4 core Xeon® D-1612 CPU. Development on this card provided a learning ground for gaining experience with DPDK and the IPU device family. Different methods were tested for receiving and handling the incoming data. When moving to the following devices, the test application was entirely rewritten, improving modularity and readability. The application developed on this platform was built differently and most of the work went into exploring solutions and learning from the experience. Later, an Intel® IPU F2000X-PL Application Development Platform (ADP), the next generation IPU from Intel, was made available. A 6-core ADP provided a head-start whilst waiting for the 8-core ADP, the main target of the testing. Lastly, similar benchmarks were conducted on a server from CERN. Specifications of these two systems are given below.

A. Server: 56-core Intel® Xeon® Platinum 8280L with Intel E810 NIC

The alternative to using a SmartNIC or IPU is to run the TPG on servers. For baseline performance benchmark, a server at CERN was also used to run the same application. Its specifications are given in Table I.

Table I: Overview of the machine used from the CERN lab

CPU	Intel® Xeon® Platinum 8280L @ 2.70 GHz (4.0 GHz turbo), 56-core dual socket 32 K L1d, 32 K L1i 1024 K L2 39424 K L3
DRAM	DDR4 16 GB, 2666 MT/s
NIC	Mellanox® ConnectX®-5 MT27800
OS	CentOS 8, Linux kernel 4.18

B. Intel® IPU F2000X-PL - 8 core

The F2000X-PL is current generation IPU from Intel. An ADP was made available, featuring an 8 core Xeon® processor (Table II). Power supply can provide the SoC with 50 W at most [10]. Production cards are now available through OEMs/ODMs (Original Equipment Manufacturer/Original Design Manufacturer).

Table II: Overview of the F2000X-PL 8-core ADP [10]

CPU	Intel® Xeon® D-1736 @ 2.30 GHz (3.40 GHz turbo), 8-core single socket 48 K L1d, 32 K L1i 1280 K L2 15360 K L3
DRAM	DDR4 16 GB, 3200 MT/s
Power	50W external supply or 40W PCIe edge power
OS	CentOS 7, Linux kernel 5.4

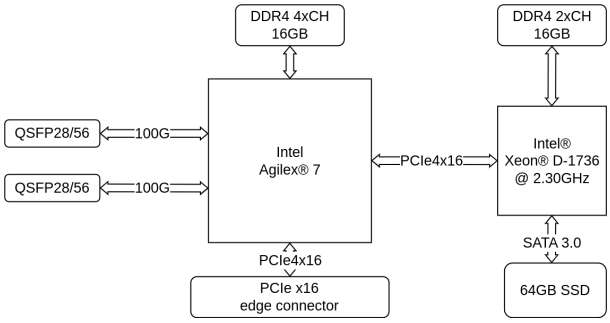

Figure 2: F2000X-PL ADP w/8-core processor [10]

IV. Test application

The test application used for evaluating the IPU relies on DPDK[11] for network functions. The Dataplane Development Kit (DPDK) is an open-source framework initiated at Intel, and managed by the Linux Foundation. DPDK works with drivers to provide user-space access to network devices, giving developers full control of the network handling, and enabling application with strict network bandwidth and latency requirements to bypass the kernel. DPDK is developing fast, and its Application Programmable Interfaces (APIs) are frequently subject to changes.

The testing requires two clients, a transmitter and a receiver. One transmitter, a simple packet generator to send frames at the desired rate and with a configurable range of source IPs, emulates data coming from the detectors. The receiver, the actual benchmarking client, runs on the device under testing, captures the incoming traffic, and processes the packets as fast as possible.

A. Transmitter

The transmitter is kept as close as possible to a minimal example of sending data with DPDK. A number of cores are used to evenly send packets with a range of source IPs, interpreted by the receiver as independent data sources.

On the Intel IPU setups, the external Tx line is another NIC (Intel® E810) on the host, providing 2 100G ports. Only one of these ports were used from the start, but whilst developing the test application on the F2000X-PL ADP, performances beyond

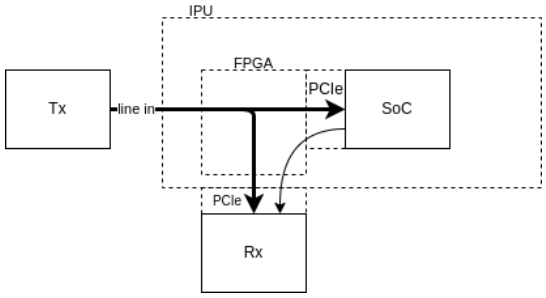

Figure 3: High-level presentation of the network. Packets from a source Tx are duplicated on the FPGA and delivered to host Rx and IPU SoC. Rx would keep a buffer, and route filtered data to further storage or analysis, but in this case simply counts the incoming packets. The SoC receives the packets and processes them as fast as possible, forwarding resulting TPs to Rx.

100 Gbps were eventually reached, and there was a need to add the E810's second 100G port for an input rate of 200 Gbps. As the current card was from a revision with support for only 1×100G, another E810 NIC was sourced in order to keep increasing the rate.

B. Receiver

On the receiving side, a flexible application was created to allow quick switching between testcases. An interface to the DPDK framework was mostly reused from work carried out on the DAQ farm software of the NA62 experiment at CERN [12]. At a first attempt, reception was done in an independent thread, keeping a Single Producer Single Consumer (SPSC) queue for incoming packets and leveraging the available memory to allow a large packet buffer. This however adds a large memory overhead, as packets are copied out of the buffer objects returned by the hardware rings. The less memory heavy solution was to use the hardware ring itself as the only buffer, working directly on the data in buffer objects in bulk, and freeing them when done. Instead of having a reception loop, the processing thread pulls frames when it's ready. This reduces the memory footprint to being almost limited to the scale of the incoming bandwidth. The disadvantages are that the ring size sets a hard limit on the buffer size, and that packets are dropped silently if there is back pressure. Reporting packet drop then becomes difficult in the case of a dynamic environment, if the incoming rate is not static and well known.

The TPG algorithm is implemented based on Intel® AVX2, of which one engine usually exists per physical core, for vectorisation. The workload, being highly homogenous, has also been tuned to minimise memory latency (see Section VI-A) and maximise CPU utilisation. Since the power of Simultaneous Multi-Threading (SMT), known as Hyper-threading in Intel products, mostly comes from intertwining threads that require different parts of the available hardware, there was expectedly little performance gain, observed from placing multiple processing threads on the same physical cores. Keeping SMT enabled was rather observed to be a liability in this case. The application is therefore limited to using the number of physical cores for processing. The remaining logical cores are then available for smaller tasks like control and monitoring, which should be possible to add without a significant performance hit. Meanwhile, the available on-board FPGA can provide capabilities for efficient unpacking of data coming in unaligned structures, saving the SoC some work. For the scope of this study however, the FPGA will be mainly used for packet routing, splitting the path and delivering copies of the incoming data to both the SoC and the host. This splitting allows the host to buffer an unmodified copy of the data, while the SoC might get the pre-processed data, e.g. reformatted or after application of basic noise filtering techniques

C. Machine configurations

Much effort was put into finding and using well-performing settings. On all targets, the governor has been set to performance mode, security features like SELinux have been disabled. On the F2000X-PL ADP, the BIOS was consulted and the "energy efficient turbo" setting disabled in hopes of allowing even more power.

It was noted that the F2000X-PL ADP used distinctively less power than advertised, at only around 37W. This also became apparent in temperature reading being far below the high threshold. BIOS modifications unlocked the full 55W by increasing power limit in socket configurations, finally allowing full core utilisation without throttling the clock rate, with a significant performance improvement.

For most of the time, a pre-installed DPDK 18.11 was used. Later, on the newer F2000X-PL ADP, DPDK was updated to 22.11, yielding a slight performance gain from aggregated API upgrades.

V. Results

For all different configurations used, measurements were done in the range of one single core, up to full processor utilisation at 8 threads running on 8 physical cores. Through monitoring the scaling, potential memory or power related issues could be found. A few different variants of handling data were tested, in order to focus on various aspects of the TPG pipeline. The first test, Section V-A "pure processing", was meant to find the processing limits of the machine. Second test, Section V-B "over network", runs a more realistic pipeline of receiving data on the network port, processing it, and sending out a response. Lastly, Section V-C, an attempt was made to send data in an unpacked format, in order to investigate the impact of the unpack stage.

All the tests performed and presented are shown as four graphs. The first plot show the throughput per core as well as the clock rate of active cores. The plotted rate is counted at the scale of WIB2 frames (lower-level network headers ignored). Where applicable, the incoming data rate has been plotted to

help interpret the result. Top right graph plots show the temperature of the CPU package, measured using `lm_sensors`. The thresholds for high temperature, and critical temperature are marked in yellow and orange dashed lines. The temperature readings provided a clue to issues regarding power supply. The third plots show the total throughput, and, when presented, the fourth plot shows system memory usage, both read and write, measured using `pcm-memory`.

A. Pure processing

The TPG algorithm was tested in an as clean environment as possible, to get an idea of the CPU's processing power and theoretic maximal performance. The process is run repeatedly on a small, static dataset, which should be as ideal for cache as possible. These results give an upper cap to the range of performance when adding more features.

Figure 4 shows the static processing on the F2000X-PL ADP. The clock rate stays more or less constant while operation temperature converges towards 75°C. The throughput per core does still gradually decrease somewhat. Figure 5 shows the same run where the unpacking stage is skipped. The impact of unpacking is discussed in Section V-C.

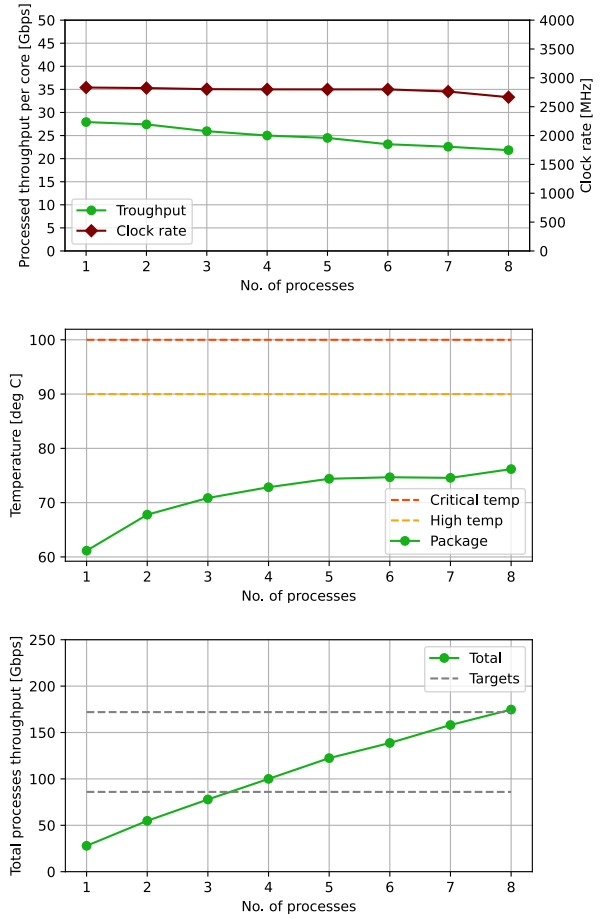

Figure 4: IPU performance scaling in the pure processing scenario.

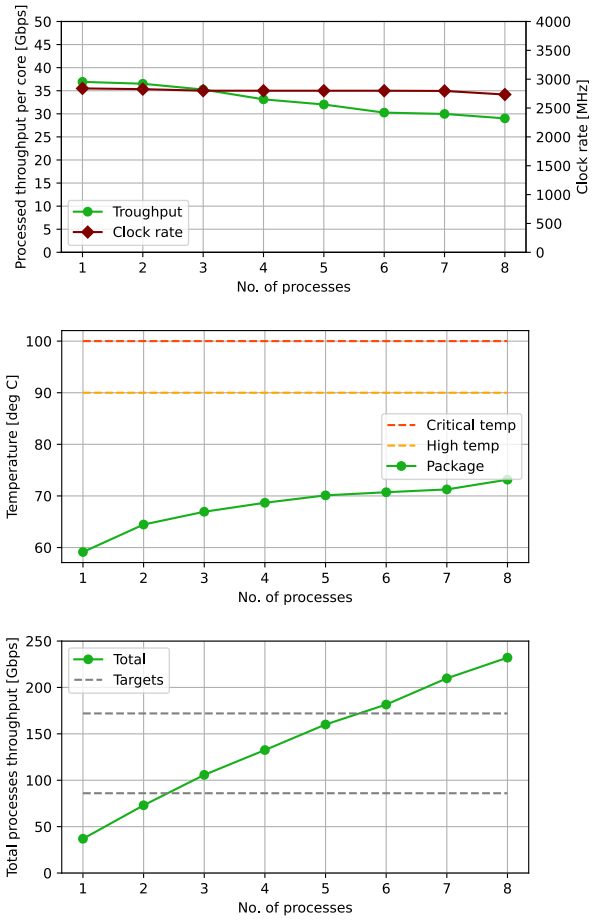

Figure 5: IPU performance scaling in the pure processing scenario, when skipping the unpacking stage.

The alternative of offloading to SmartNICs or IPUs is to continue running on conventional servers. The test application was also run on one of the servers of the CERN lab, described in Section III-A, first up to 8 cores, then in up to full utilisation, with TPG running on all 56 cores divided on 2 sockets. The per-core throughput and clock rate is shown in Figure 6. As expected, the throughput starts at just slightly higher than on the embedded CPU of the IPU, and a similar scaling is seen in the per-core throughput.

B. Over network

The most realistic case is when receiving data over network, and sending a response afterwards. The incoming data saturates the link, and distributes over the number of active cores (grey line t.l.). Instead of using a software queue as buffer, memory usage has here been cut in half by instead reading directly from the hardware ring. The disadvantage being that the extent of buffering is limited to the ring's capacity (32k packets, meaning 0.2 seconds in this case), and that packets are dropped silently if back pressure is too high, due to not having been read before being overwritten.

Figure 7 shows the full DPDK and TPG on the F2000X-PL ADP. Similarly to the previous test, there is now almost

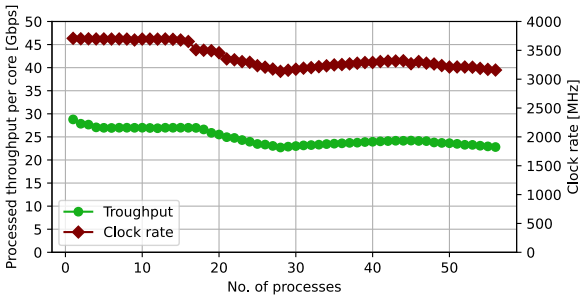

Figure 6: Offline TPG on the server from the CERN lab. The average clock rate and scaling decrease as the cores of one socket are being filled past halfway, then rises for a moment when cores on the next socket are included.

no throttling of clock rate. Note that a no memory usage is reported below 5 active threads. Section VI-A describes DDIO, a technology that allows incoming data to be passed straight to last-level cache (LLC). As the number of processing threads, and receiving rings, increase, memory usage starts increasing, as the effect of DDIO is reduced if the cache capacity is exceeded. Various BIOS settings increased available power and overall performance, as mentioned in Section IV-C. At very high rates, fine-tuning of burst parameters becomes crucial. Lack of synchronisation was observed to limit performance, as a large burst would be dropped even if only a few packets from it would be missed. This tuning restricts the sending rate, but targets the threshold where traffic flows smoothly, and missed packets don't further affect performance. The jumps in incoming rate are due to the traffic being sent on two distinct lines, each with a limit of 100Gbps, as well as the sending burst size being reduced from 256 to 64 starting from 6 cores.

After seeing these results, it is also interesting to approach the real scenario even further. Some aspects presented in Section II have not been respected in the tests above. The data from one module comes from 12 distinct links which need to be processed sequentially. Second, the transmission of the produced TPs to where a buffer is allocated in order to signal what data to keep. Beginning with the distinct links, when the number of threads does not map cleanly to the number of physical cores, 3 threads were created for each 2 cores.By tuning packet burst sizes, ring sizes and burst intervals, a rate of 8.3 Gbps was achieved on each thread, with the single loaded cores running at about 60% utilisation, and the double loaded cores running at about 85% utilisation.

The impact of TP sending was measured by sending packets with a fixed size at a given rate. The expected TP rate is for each channel to produce a hit at a frequency of 100 Hz. With data coming in at 2 MHz per channel, that means 1 in 20000 values (0.005%) produce a hit. Figure 8 shows the incoming data rate as a function of the fraction of ADC inducing TPs. With an incoming WIB throughput of 7.9 Gbps (corresponding to a rate of 2.1 MHz, slightly above requirement), the figure shows that a fall of the data rate occurs starting from a fraction of 4% (90 kHz). Note for this case that the implementation of

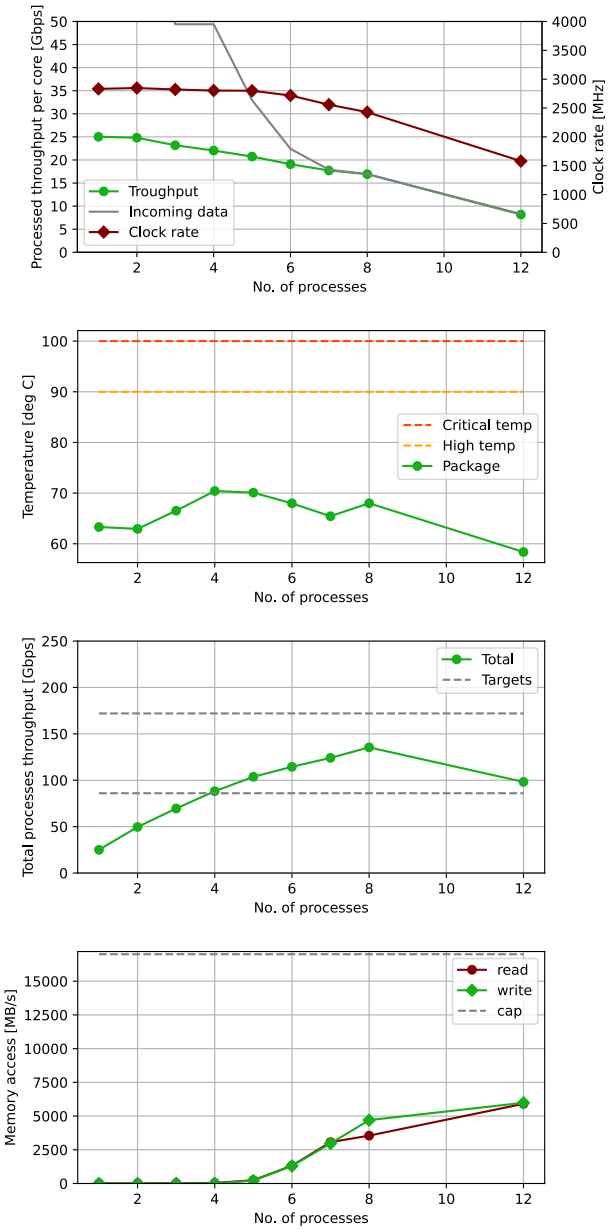

Figure 7: IPU performance scaling in the realistic scenario. The incoming data, marked as a grey line in the first plot, drops as the total data is divided on multiple data streams.

sending is not optimal; one packet is sent at the time, with a fixed rate. In a practical scenario, TPs will be generated at a more dynamic rate, and there will also be an occasion to implement a burst sending, to the extent memory limitations will allow it.

C. Impact of the unpack stage

The TPG algorithm in itself is not very memory intensive, it will just use as much as it is allocated. However, the unpack stage does an explicit copy when rearranging the data, leaving a larger memory footprint. This stage is also identified as computationally rather simple, stable, and highly parallel, as well as working on unaligned data formats. It has been

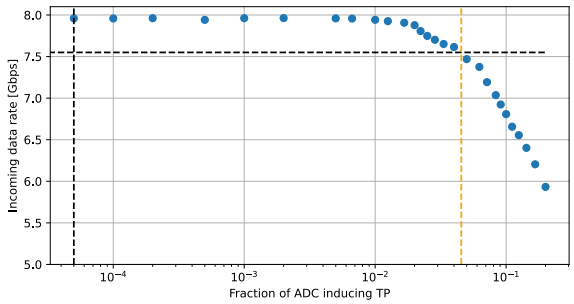

Figure 8: Performance as the hit rate of TPG is increased beyond the expected value. Incoming data rate was set slightly above the nominal rate of 2 MHz (7.55 Gbps). The dashed horizontal line show nominal rate. The black, dashed, vertical line shows the expected TP rate of 100 Hz (0.005%) per channel, the yellow, dashed vertical line shows the intersection of the TP rate that could be sustained at nominal TPG rate, at 90 kHz (4%), 900× expected rate.

suggested to consider moving this stage to FPGA, as the function should be well suited for it. Although this was outside the scope of this work, it is interesting to investigate the potential performance impact of offloading the unpack stage to hardware.

To simulate an environment where the unpacking is done in hardware, on the data stream, as it arrives to the NIC, a test was done with sending the data already unpacked. On the transmitter side, a set of WIB2 frames are created as before, and then stored in the unpacked format. These unpacked frames are then sent instead, needing a higher data rate to send the same number of frames. On the receiver side, since the data is already in the unpacked format, the unpacking stage is skipped and only TPG is run.

Figure 9 shows the scaling when using two lines to achieve close to 200Gbps input. Figure 5 shows the same run on a static dataset, as with the testing in Section V-A, for a comparison with the TPG processing in Figure 4. Note that the temperature and clock development does not change significantly. This can suggest that the comparison is fair, as the device is equally stressed in both cases. Meanwhile, the memory throughput in this case is growing faster than previously, suggesting that there is still some potential of improvement by tuning the cache hits for the unpacked frame size. This could regain the advantages of DDIO seen earlier.

Figure 10 compares the three configurations tested on IPU. Not surprisingly, skipping unpacking gains throughput for data over network, with performance in the two configurations correlating well. The fact that it also gains throughput compared to the static set at low core utilisation, but reaches a cross-over at full utilisation, seems consistent with earlier discussion on cache utilisation and DDIO, and could suggest that processing capacity is initially the limiting factor.

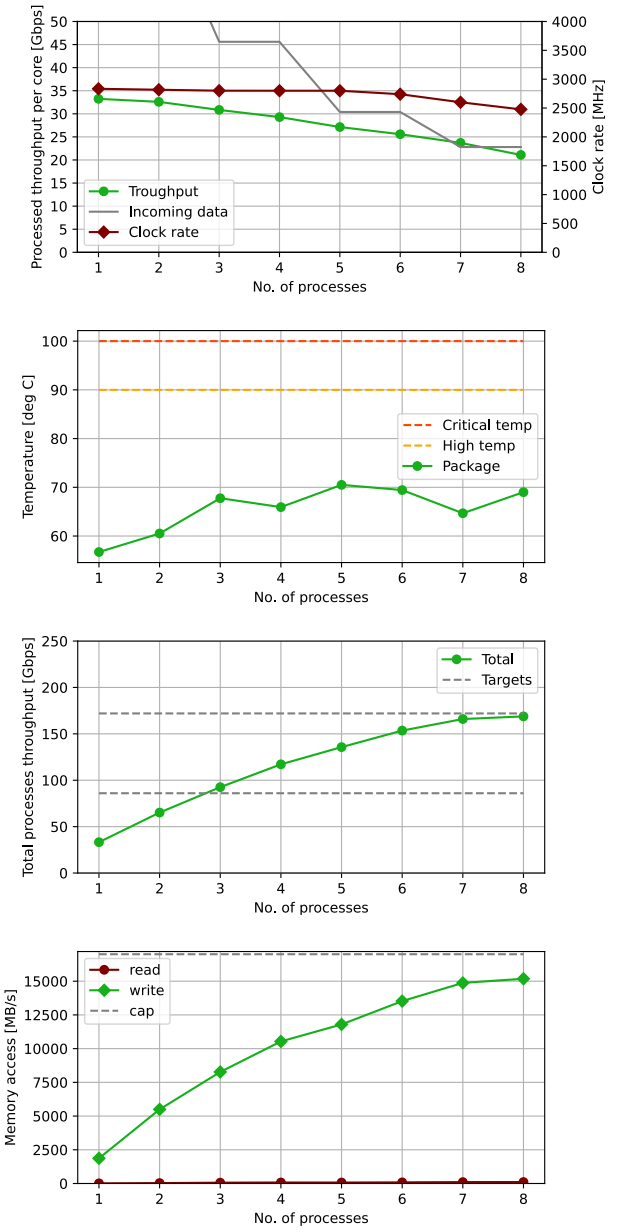

Figure 9: DPDK + TPG with data sent as unpacked frames. A grey line in the top left plot marks the incoming data rate.

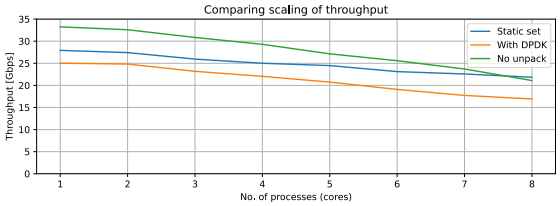

Figure 10: Comparison of throughput rate as the number of cores is scaled, for the pure processing, normal DPDK implementation, and simulated offload of unpack.

VI. Notable Technologies and Further Work

Much time was spent exploring various OS and BIOS configurations to tune the device for the specified workload. Starting

tests at a single core, it can be observed what the device is capable of at best. Then, one also needs to take into account the scaling penalty on memory consumption, power, temperature and clock rate.

The need to rely on the hardware rings as buffers leave little room for jitter, so a sufficient margin in performance will be needed in order to be a reliable solution. It should nevertheless be noted that in the studied use case data arrive with a fixed size and at fixed rate, thus having naturally very little jitter.

A. DDIO: Data Direct IO

Data Direct I/O (DDIO), introduced with the Intel® Xeon® E5 processor family, makes a direct connection between IO and processor cache, rather than having all IO reads and writes go through main memory [13]. This technology has become possible with the improvements in IO speed and increase of last level cache sizes. With the memory consumption of the TPG algorithm, the size of the incoming packets, and the requirement of high throughput, this should be able to increase the performance by a lot.

Extensive testing showed that network packets were consistently missing L3 despite DDIO available on this CPU. This issue caused excessive memory traffic limiting application performance. Tuning IO-dedicated LLC and reducing ring buffer sizes allowed to land about 80% of network data in L3 cache even at the highest rate, and gave a significant performance boost.

B. Programmable hardware datapath via FPGA

Throughout this work the FPGA IPU has been programmed with Intel provided NIC and packet processor functionality. This configuration has been used as if it is a fixed function with the FPGA performing the packet forwarding, duplication and assignment of queues within the `virtio` DMA that provides the interface between host and FPGA and SoC and FPGA. The packet processor is software programmable via a management tool that runs on the SoC to configure match-action packet processing, traffic shaping, QoS and miscellaneous functions. This functionality takes a fraction of the FPGA resources, leaving plenty of room for value-add customisation to the data path.

The unpacking of 14 bit data into a more CPU friendly, 16 bit aligned format has already been identified as a notably heavy task within the software running on the SoC on the IPU. This functionality is both a fixed and trivial function to implement in hardware, making it a prime target for FPGA acceleration. By offloading this processing to the FPGA, the SoC will have more capacity to execute the TPG algorithm, leading to performance benefits like suggested from the results presented in Section V-C. This gives algorithm developers the benefit of FPGA acceleration as well as the flexibility and convenience of software development on a standard CPU.

C. Format and size: WIBEthernet

As briefly mentioned in Section II, the WIB2 format used for this report has in the meantime been replaced with a new format named WIBEthernet. This new format packs more data into one frame, thus changing the units of transfer. In the future it would be interesting to adapt the current test to the new format, experiment with how other parameters will have to be adjusted, and to see if the new format can improve the capacity further. The WIBEthernet frame will be 25% larger than a superchunk, still with the same network headers, which could reduce the overhead workload needed for handling packets.

D. Further work

The work done for this report has provided good experience with IPUs, and sparks a number of questions going forward. In relation to the use-case of DUNE, the testing in this report has been in many aspects somewhat disconnected from the real system. The focus has been on the on-board workload, whilst the surrounding task, that of sensor readouts and data transmission on host, have been ignored or simulated. In the future, it could be interesting to build on this to set up a prototype of the entire pipeline of receiving real data, handling the buffers and forwarding of data of the host machine. It would also be interesting to step-wise add more of the features to the TPG, to measure each feature's weight, and find the limits of the current IPU's capacity. The DUNE detector prototypes installed at CERN and scheduled for realistic operation in 2024 would be a perfect playground to fully validate the potential use of IPUs in HEP.

VII. Conclusions

The work summarised in this report has built experience with DPDK as a platform, and with the usage of advanced SmartNICs through Intel® IPUs, mainly with the F2000X-PL application development platform. These devices merge processing and network handling in a new way, originally intended for data centre infrastructure and network functions. By testing IPUs in a different environment, an attempt was made to uncover another market for these devices in the TDAQ design for HEP experiments. The WIB2 frame format and the TPG algorithm from the specifications of the DUNE experiment were used as practical examples, and for evaluating the idea of using IPUs in the TDAQ system of DUNE.

Measurements presented here indicate that the F2000X-PL shows a potential, being able to handle the minimal requirement of one electronics board comprising 12 incoming links at 7.55 Gbps (2 MHz) of WIB2, running TPG, and transmitting trigger primitives at two orders of magnitude higher than the expected 235 Mbps (100 Hz), still with some margin that should allow a more slightly more complex application to handle one electronics board per IPU. Looking at the maximum acquired throughput, without considering constraints of sequential tasks, the test application achieved a 130 Gbps throughput, 70% of the throughput needed to support 2 electronics boards of 3072 channels each. Further optimisations could help reach this target with the current generation device,

however with a performance margin that won't allow adding much more complexity to the workload.

Throughout the project, it has repeatedly become clear that memory consumption is the biggest challenge in the tested use-case. All major improvements in throughput have come from memory optimisations, such as avoiding software buffers, using small packet bursts on the network link, and minimising the ring size of the network port. Memory also showed to be a limiting factor when attempting to factorise the 130 Gbps throughput to 12 links rather than 8, which required the available memory to be divided over an increasing number of data streams. In addition, the mismatch between the number of processing threads and the number of physical cores were adding a context switching overhead, and making an even load distribution difficult. The available processing power could be better used if those numbers were properly aligned. While a working example of processing on 12 links of 8 Gbps each, the load of one electronics board, has been presented, it is still important to remember that the tests have been using a minimal workload, a highly simplified version of the TPG. More complexity, as well as more advanced control and monitoring, will have to be added on top to make a useful application.

The FPGA available on the F2000X-PL and other IPUs has so far only been used for packet routing, but some of the measurements presented should argue for the potential that it provides. Offloading stable functions to the FPGA frees up SoC bandwidth to be used on other tasks improving overall system efficiency. The path splitting, which already saves a good amount of work spent on copying data, would allow preparing of data, like unpacking, reorganising, noise filtering, to be done at low cost, and with efficiencies that would not be possible if the original data could not in parallel be transmitted to the host.

REFERENCES

[1] G. Lehmann Miotto. 'The FELIX Project.' (Jun. 2022), [Online]. Available: https://ep-news.web.cern.ch/content/felix-project (visited on 22/05/2023).

[2] K. Deierling. 'What is a SmartNIC?' (Oct. 2021), [Online]. Available: https://blogs.nvidia.com/blog/2021/10/29/what-is-a-smartnic/ (visited on 30/05/2023).

[3] A. Moore and J. Henrys, 'IPU-based Cloud Infrastructure: The Fulcrum for Digital Business,' Intel Corporation, Tech. Rep., 2021. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/programmable/ipu-based-cloud-infrastructure-white-paper.html (visited on 22/05/2023).

[4] A. Borga, E. Church, F. Filthaus *et al.*, 'FELIX-Based Readout of the Single-Phase ProtoDUNE Detector,' *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, Jul. 2019.

[5] K. Deierling. 'What Is a DPU?' (20th May 2020), [Online]. Available: https://blogs.nvidia.com/blog/2020/05/20/whats-a-dpu-data-processing-unit/ (visited on 02/06/2023).

[6] *DUNE Homepage*. [Online]. Available: https://dunescience.org.

[7] R. Sipos, 'DUNE DAQ Readout Final Design Review,' Tech. Rep. [Online]. Available: https://edms.cern.ch/ui/file/2826457/1/DUNE-DAQ-FDR-Readout.pdf.

[8] DUNE-DAQ fdreadoutlibs, Github. [Online]. Available: https://github.com/DUNE-DAQ/fdreadoutlibs (visited on 22/05/2023).

[9] *C5010X Product Information*. [Online]. Available: https://www.silicom.dk/product-details/silicom-c5010x-data-center-nic/ (visited on 22/05/2023).

[10] *Intel IPU F2000X-PL Data Sheet*, Version 2023.03.06, Intel, Apr. 2023.

[11] *DPDK Homepage*. [Online]. Available: https://dpdk.org.

[12] *NA62-Farm repository*, Gitlab, CERN, 2023. [Online]. Available: https://gitlab.cern.ch/NA62FW/na62-farm/na62-farm-lib-networking.

[13] Intel, 'Intel Data Direct I/O Technology: A Primer,' Intel Corporation, Tech. Rep., 2012. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/data-direct-i-o-technology-brief.pdf (visited on 07/07/2023).