WHITE PAPER

NFV Acceleration
P4 Compiler

Making Virtualized Mobile Gateways More Efficient

NETCOPE
INNOVATIVE ACCELERATION

Authors

Denis Matoušek
Product Manager,
Netcope Technologies

Viktor Puš
Chief Technology Officer (CTO),
Netcope Technologies

Miroslaw Walukiewicz
Solution Architect,
Intel Corporation

Introduction

This white paper discusses the modern trend of using smart network interface cards (NICs) in the segment of mobile telecommunications networks where they help with performance sustainability in virtualized environments. In 4G networks, many network functions are being implemented in virtual machines or containers instead of using dedicated hardware. Obtained flexibility is redeemed with so high demands on CPU cores that it makes it very difficult to process all the traffic. With the approach of 5G networks, the demands for increased performance because of higher peak data rates and user experienced data rates and reduced latency are merciless. Utilizing the Intel® FPGA Programmable Acceleration Card (Intel FPGA PAC) N3000 can improve overall server density by offloading workloads, the CPU can be used for more complex tasks of the control plane, and thus reduce the overall number of servers and consequently capital and operational expenses. The white paper shows how the Intel FPGA PAC N3000 can be used together with Netcope P4 service to deliver a mobile network core offload in an unprecedentedly short time.

The key idea lies in the separation of control and user plane. The functions of the user plane (i.e. the fast path) are good candidates for offload in the Intel FPGA PAC N3000. These functions are often straightforward but require high performance. There are functions running both within and on the edge of evolved packet core (EPC) in 4G networks or 5G core (5GC) in 5G networks. In the case of 4G networks, the functions are performed in serving gateway (S-GW) and packet data network gateway (PDN-GW) components and in the case of 5G networks, it is the user plane function (UPF) component, but, the functions remain almost the same: packet filtering, quality of service (QoS) enforcement, flow-based charging, data buffering and others. On the edge of the core, GTP-U encapsulation and decapsulation is performed for the access network and firewalling, Network address translation (NAT), Deep Packet Inspection (DPI), parental control, video and web optimization, and others are performed for the external network (i.e. Internet). The situation is depicted in Figure 1.

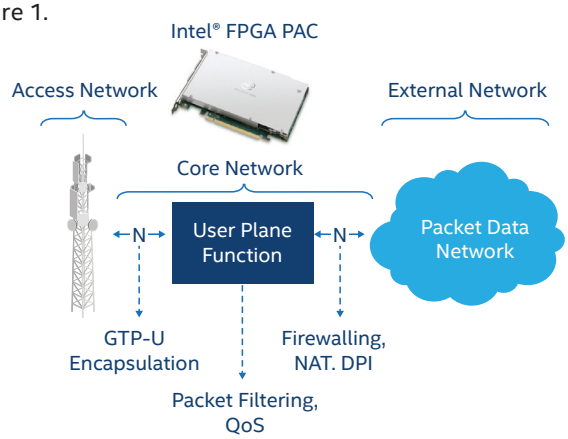

Figure 1. The functions performed on a user plane of a 5G core network for consideration using smart NIC offload

Table of Contents

Shortening Time to Market

To make the Intel FPGA PAC N3000 accessible to a broader audience of network engineers and to shorten the development cycle and overall time to market, Netcope offers a service of converting a program written in P4 language to the firmware that targets the Intel FPGA PAC N3000. P4 is a high-level, domain-specific language targeting network applications. Traditional FPGA design cycles using languages like VHDL or Verilog can be shortened from weeks to days[1]. The conversion process of a P4 program to the firmware for the Intel FPGA PAC N3000 is shown in Figure 2:

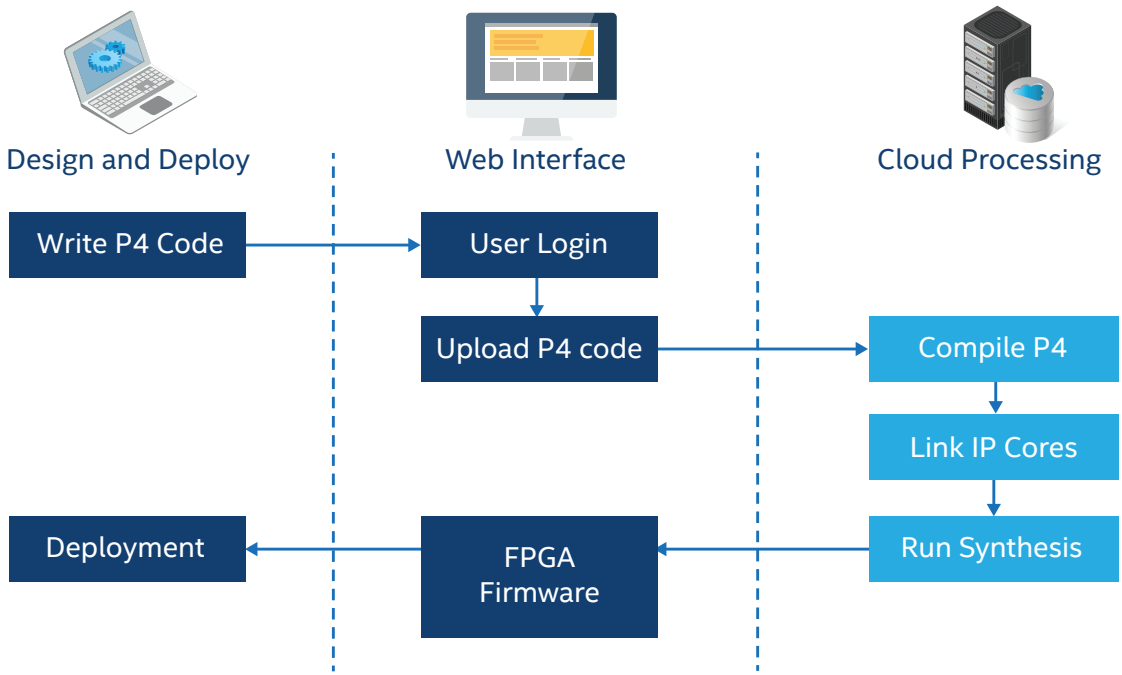

Figure 2. Using Netcope P4 to convert a program written in P4 language to the firmware for the Intel FPGA PAC N3000

Using Netcope P4 is so straightforward in that traditional knowledge of FPGA design is not required. For those who need to integrate their packet processing functionality into their custom hardware platform, there is always an option to generate an intellectual property (IP) core in the form of a netlist.

The functions that are really complex and are out of the scope of P4 language can be included in the form of externs. This feature of P4 language allows using 3rd party IP cores in the P4 processing pipeline. Examples of such functions are QoS, more particularly hierarchical QoS (HQoS), DPI, and encryption (e.g IPsec).

Particular functions that were considered for offload are mapped to the features of P4 language according to Table 1.

Function	Feature of P4 language
Packet filtering	Header field lookup and packet forwarding/drop using match and action tables of P4 language
QoS enforcement	HQoS engine delivered as an extern of P4 language
Flow-based charging	Updating a large array of counters of P4 language instantiated in DRAM or memories
GTP-U encapsulation and decapsulation	Header removal and insertion in P4 language
Firewalling	Header field lookup and packet forwarding or drop using match and action tables of P4 language
NAT	Header field lookup in DRAM memories and header field update including checksum re-computation in P4 language
DPI	Pattern-matching engine as an extern of P4 language

Table 1. Mapping functions considered for offload to the features of P4 language

To demonstrate how straightforward it is to use the P4 language, snippets of the P4 code that do actual GTP-U decapsulation and load balancing are shown in Example 1.

Protocol Headers

```
// GTP-U mandatory fields
header_type gtpu_mandatory_t {
  fields {
    version : 3;
    protocol_type : 1;
    // ... omitted
    messType : 8;
    messLength : 16;
    teid : 32;
  }
}

// GTP-U optional fields
header_type gtpu_optional_t {
  fields {
    seqNumber : 16;
    nPduNumber : 8;
    nextExtHdrFlag : 8;
  }
}

// Header instances
header gtpu_mandatory_t
  gtpu_mandatory;
header gtpu_optional_t
  gtpu_optional;

// Definition of hash fields
field_list gtpu_fields {
  gtpu_mandatory.teid;
  inner_ipv4.srcAddr;
  inner_ipv4.dstAddr;
}
```

Protocol Parser, GTP-U Decapsulation and Load Balancing

```
// Definition of hash for load balancing
calculation field_list_calculation gtpu_csum {
  input {
    gtpu_fields;
  }
  algorithm : csum16;
  output_width : 8;
}

parser parse_gtpu_mandatory {
  // Extract mandatory part of GTP-U
  extract(gtpu_mandatory);

  // Parse optional GTP-U header if present
  return select(latest.extHdrFlag,
    latest.seqNoFlag, latest.nPduNoFlag) {
      1 mask 1, 2 mask 2, 4 mask 4 :
        parse_gtpu_optional;
      default : parse_inner_ipv4;
  }
}

action decapsulateAndDistributeGtpU() {
  // Hash header fields
  modify_field_with_hash_based_offset(
    gtpu_metadata.hash, 0, gtpu_csum, 65536);
  // Remove GTP-U headers
  remove_header(gtpu_mandatory);
  remove_header(gtpu_optional);
  // Perform load balancing over 32 channels
  modify_field(intrinsic_metadata.egress_port,
    gtpu_metadata.hash, 31);
}
```

Example 1. Snippets of the P4 code that do actual GTP-U decapsulation and load balancing

Advanced Features

Netcope P4 supports external components that are available on the Intel FPGA PAC N3000 platform including DDR and QDR memories and Intel Ethernet controller XL710.

Storing items of match and action tables in the FPGA chip leads to very fast access with very low latency but provides relatively low capacity. If the items are stored in external memories, the capacity is significantly bigger depending on mounted memory modules. The situation is depicted in Figure 3.

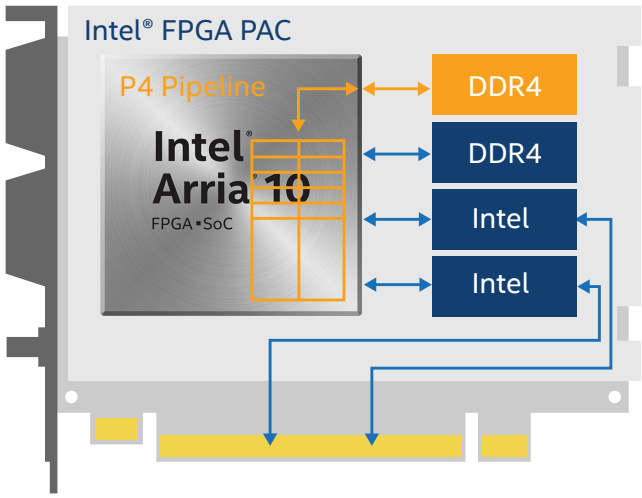

Figure 3. Intel FPGA PAC N3000 with P4 pipeline with match and action table using external on-board DDR4 memory (shown in orange) to extend the capacity

In the case of a mobile core network, external memories are suitable for storing a large array of counters, which are used for flow-based charging, or for storing NAT flow table.

Intel Ethernet controllers XL710 supplement the onboard FPGA chip and implement fundamental functionality that is used in NICs most often including various receive filters, LAN engine, support of single root I/O virtualization (SR-IOV), receive-side scaling (RSS), and so on. With this approach, you do not need to bother with implementing such basic functionality so that you can focus on the offload itself and leverage massive parallelism that is available in the Intel Arria® 10 FPGA.

Conclusion

A summary of the key ideas we discussed in the white paper are as follow:

- Solving performance bottlenecks of virtualized functions by offloading the bottlenecks to the Intel FPGA PAC N3000
- The Intel FPGA PAC N3000 has the ability for firmware upgrades and achieving unparalleled flexibility
- Offloading to the Intel FPGA PAC N3000 saves CPU cores for more complex tasks of the control plane, thus reducing the overall number of servers and consequently capital expenditures (CapEx) and operating expenses (OpEx)
- Shortening time to market by using high-level language for the description of the hardware offload
- Mapping features functions of a mobile network core to the features of P4 language
- Using Netcope P4 to generate high-performance firmware for Intel FPGA PAC N3000

More information please visit:

- www.intel.com/pac-n3000
- www.netcope.com/en/products/netcopep4

[1] For details about the shortened FPGA design cycles or development time, refer to the Building a PoC of Segment Routing at 100G Using FPGA Smart NIC and P4 Language White Paper.

WP-01296-1.0