High-Capacity FPGA Router for Satellite Backbone Network

STRAHINJA JANKOVIĆ, Member, IEEE
ALEKSANDRA SMILJANIĆ, Member, IEEE
MIHAILO VESOVIĆ, Student Member, IEEE
HASAN REDŽOVIĆ, Member, IEEE
MARIJA BEŽULJ
University of Belgrade, Belgrade, Serbia

ANDREJA RADOŠEVIĆ
Stellar Labs, LLC, Los Angeles, USA

SLAVEN MORO
Facebook, Inc., Menlo Park, CA, USA

Satellite backbone networks provide a viable means of establishing broadband connectivity for remote, sparsely populated areas. In addition, satellite communication systems are well suited for airborne, maritime, and disaster relief environments. Technologies for links are continuing to improve in performance and power efficiency, making onboard regeneration and routing feasible within spacecraft power envelope. In this article, we implement and analyze a spaceborne router design integrated on a field-programmable gate array (FPGA). FPGA provides a flexibility needed to circumvent space radiation effects on chip circuitry, as they can be reconfigured at runtime. We explored scalability of the high-end state-of-the art FPGA chip family, and its ability to support high bit-rate satellite links: 10 Gbps satellite-to-ground links and 100 Gbps intersatellite links. Through implementation and testing, we confirm that the current FPGA technology can support space routers with very high data throughput.

Manuscript received May 28, 2019; revised September 17, 2019; released for publication October 20, 2019. Date of publication November 4, 2019; date of current version August 7, 2020.

DOI. No. 10.1109/TAES.2019.2951187

Refereeing of this contribution was handled by R. Wang.

This work was supported by Facebook Connectivity.

Authors' addresses: S. Janković, A. Smiljanić, M. Vesović, H. Redžović, and M. Bežulj are with the School of Electrical Engineering, University of Belgrade, Belgrade 11000, Serbia, E-mail: (jankovics@etf.bg.ac.rs; aleksandra@etf.bg.ac.rs; mikives@etf.bg.ac.rs; hasan.redzovic@ic.etf.bg.ac.rs; meja@etf.bg.ac.rs), A. Radošević is with the Stellar Labs, LLC, Los Angeles, CA 90035 USA, E-mail: (andreja@stellarlabs.net) S. Moro is with the Facebook, Inc., Menlo Park, CA 94025 USA, E-mail: (smoro@fb.com). *(Corresponding author: Strahinja Janković.)*

I. INTRODUCTION

Satellites are traditionally used for broadcasting and, to a lesser extent, for Internet traffic due to the high latencies in geosynchronous (GSO) satellite constellations [1]. However, large constellations with low Earth orbiting (LEO) satellites provide low latencies, and could be utilized as the Internet backbone network. Recent technology advances have enabled the increase in communication channel capacity between satellites and the Earth. Furthermore, satellite communications have lower entry cost in rural and underdeveloped areas. Also, satellite networks offer crucial robustness during natural and man-made disasters. For these reasons, the research community is seeking to improve satellite backbone capacity for Internet traffic.

Internet service and content providers are interested in satellite backbone networks due to their wide international coverage. Using satellite technologies, Internet access could be provided even to people in isolated remote areas, especially since investments in fixed broadband networks are not profitable in those areas. NASA is developing the next-generation space architecture through its Space Communication and Navigation (SCaN) program, and is looking for partnerships. Google is developing a temporospatial software-defined networking (TS-SDN) platform that is tailored to the specifics of wireless high-altitude communications [2]. This software platform has been designed for high-altitude platforms and can model wireless propagation losses due to atmospheric absorption, rain, clouds, fog, and tropospheric scintillation.

LEO satellite constellations are a very promising solution for high-altitude backbone wireless networks. There are other options as well; however, they include less mature technologies such as unmanned aerial vehicles and stratospheric balloons. In this article, we consider universal high-altitude nodes, having in mind primarily satellite networks that provide ultrawide reach.

Three high-capacity LEO networks will likely be launched within the next three years: SpaceX, Telesat, and OneWeb constellations with 4425, 117, and 720 satellites, respectively. It was shown that all three constellations could provide throughputs over 1 Tbps based on their FCC filings [3]. SpaceX and Telesat constellations use intersatellite links and can connect any two users in the network, while OneWeb relies on terrestrial gateways for communication between different areas that satellites cover. The SpaceX constellation provides the highest throughput as it comprises a much larger number of satellites than the other two constellations.

Fig. 1 shows the architecture of a high-altitude backbone node which could be used in the SpaceX and Telesat constellations. It comprises user modems (UM), user antenna assembly, gateway modems (GM), gateway antennas, optical modems, optical antennas, central processor unit (CPU), telemetry tracking and command (TT&C) system, and space router. A satellite node is linked via UMs and antenna assembly to the customers in the satellite coverage area. It is linked to the backbone terrestrial network via

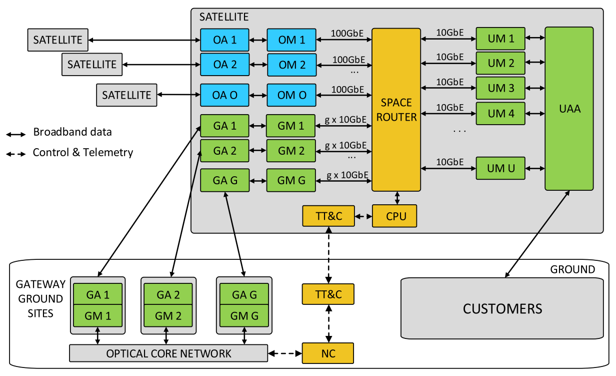

Fig. 1. Architecture of the satellite backbone network.

GMs and antennas. Satellite-to-ground links can provide capacities exceeding 10 Gbps, which are determined by the speeds of UMs and GMs. Satellite nodes are connected to each other via high-throughput free-space optical intersatellite links, which operate at information rates exceeding 100 Gbps [4] using optical coherent modems [5]. Space router should exchange IP packets between lower speed satellite-to-ground and higher speed intersatellite links. Control packets carrying information about network topology are processed in CPU that is connected to the network controller via reliable TT&C links.

Field-programmable gate arrays (FPGAs) have been recognized as suitable for space applications due to their high performance and reconfigurability [6]–[8]. High-end FPGAs comprise large numbers of logic elements and on-chip RAM memory, which allow them to provide high level of parallelization, and, consequently, high-processing throughputs. Obviously, it is not possible to change hardware on satellites, while the functionalities implemented on FPGA could be easily modified and upgraded using the wireless networking resources. At the same time, reconfigurability of FPGA helps mitigation of single and multiple event upsets (SEU and MEU) that are caused by critical radiation effects. FPGA can support protection mechanisms for substitution of the affected modules by the correct ones [8]. Application-specific integrated circuit (ASIC) might be more optimal solution in terms of power consumption or speed; however, ASIC lacks flexibility as it cannot be reconfigured remotely while in space. For this reason, FPGA has a major advantage since it can be reconfigured if mission objectives change or design upgrade is required.

Space router is an important assembly on the high-altitude platform as it transfers the entire incoming traffic to the corresponding output ports. Therefore, a space router is a potential bottleneck of the high-altitude node capacity. In this article, we present our design and implementation of an internet router on the FPGA. We will analyze our design on a high-end Virtex Ultrascale+ FPGA, in terms of the required chip resources, as well as the maximum achievable data throughputs. We focus on the data plane design and implementation as it is the important part of the system. Our design of data plane is flexible, and it could be connected to the standard routing control plane [9], as well as to the SDN controller using the OpenFlow standard [10].

II. Related Work

Packet processors are an important application of high-end FPGAs. Nowadays, high-end FPGAs support interfaces with bit-rates of up to 100 Gbps. FPGA manufacturers already provide IP cores that implement high-speed ethernet link layers: 10GE, 25GE, 40GE, and 100GE [11]. In addition to these high-speed link layer protocols, Xilinx developed flexible SDNet platform for packet processing at the same speeds [12]. SDNet platform allows implementation of typical network functions using the popular programming language for data plane, p4 [13]. SDNet platform also provides the software module for downloading lookup tables from the control plane. It is, unfortunately, closed software, and it does not include incremental lookup table updates. SDNet platform can implement standard lookup tables used by Internet routers, or OpenFlow lookup tables connected to SDN controller.

Various network functions were designed and implemented in the research literature. Typically, critical network functions were addressed separately as potential bottlenecks, such as lookup, packet classification, or scheduling.

The space router first needs to determine output ports of incoming packets using some type of lookup. IP lookup is the most widely used on the Internet, as it has certain advantages. First, it uses global IP addresses for packet routing. Second, IP lookup tables can be reduced by address aggregation, since IP addresses are flexibly allocated to devices unlike medium access control (MAC) addresses [14]. Finally, information about location of network addresses is distributed using fairly simple protocols such as OSPF (open shortest path first), routing information protocol, or intermediate system to intermediate system. Disadvantage of IP lookup tables is their large size as they store global addresses. Alternatively, multiple protocol label switching (MPLS) uses lookup tables that store labels with local significance. Consequently, MPLS lookup tables are smaller; however, they require additional control protocols, e.g., resource reservation protocol or label distribution protocol, for establishment and maintenance of MPLS tunnels. In our design, we implemented IP lookup tables to avoid complexities of control plane which is beyond the scope of this article.

Sophisticated packet classification is difficult to implement if it should include the large number of fields in packet header. Various solutions were proposed for increasing the throughput of packet classification while reducing its memory requirements [15], [16]. However, high-altitude nodes are not well suited for packet classification, as it will be shown that they need to perform basic network functions with high resource requirements. Packet classification is more appropriate for terrestrial gateway nodes.

SpaceWire and SpaceFibre standards are designed for connecting sensors, memories, processors, and other devices on a spacecraft [17], [18]. These standards focus on reliable link layer with low power consumption, and consider also packet routers appropriate for such link layer. SpaceWire standard includes link speeds up to 200 Mbps, while SpaceFibre standard aims to support link speeds up to 10 Gbps. The routers for onboard networking do not have high throughput requirements as they connect limited number of devices. For the same reason, they do not require large number of addresses in lookup tables. In our article, we aim to achieve packet forwarding with the highest throughput possible and lookup tables that support routing on global Internet.

The space radiation issues, like total ionising dose (TID) and single event upsets (SEU), are important concerns when choosing components for design that will be used in space [19]. For the static random access memory-based FPGAs, like the one used for our design, there are many established methods for mitigating these issues [20], [21].

TID hardening can be provided by FPGA vendor or by additional shielding, but it can be also mitigated by techniques like switched modular redundancy [20] where dynamic reconfiguration is used.

On the other side, SEUs are usually not handled by the FPGA design and manufacturing process. However, there are different techniques that can be applied in order to mitigate the SEU effects [19], [21]. Redundancies can be introduced on global or local level. Triple modular redundancy (TMR) uses voting mechanism at the outputs to mask SEU, while double modular redundancy with comparison (DWC) can be used to detect an SEU event. For memories, error detection and correction (EDAC) can be used to detect and correct errors in memory contents resulting from SEU. Additionally, for the FPGA configuration memories, scrubbing can be performed by rewriting configuration with the reference design without impacting device operation.

In our analysis and implementation, we have focused on implementing a high-throughput router design. Reliability can be increased using the aforementioned techniques. Additionally, reliable communication is also implemented at network and transport layers of IP network, by rerouting traffic around failed devices and packet retransmissions.

Packet routers can have different architectures that mainly depend on the required capacities, and underlying hardware. The lowest capacity routers have either output buffers or shared buffers. Their capacity is limited by the buffer throughput. Routers with input buffers have higher capacities, but they require more complex schedulers for packet forwarding [22], [23]. Finally, multistage routers can provide the highest capacities [24], [25]. However, they have larger memory requirements which are critical on the FPGAs. Also, some architectures, e.g., toroidal, need significant link speedups in order to provide forwarding with no blocking [26]. We will implement the space router with input/output buffers due to the limited throughput of on-chip buffers, as well as their total size.

III. SPACE ROUTER ARCHITECTURE

The space router should be designed to fit the underlying FPGA architecture. We will use the high-end Xilinx chip

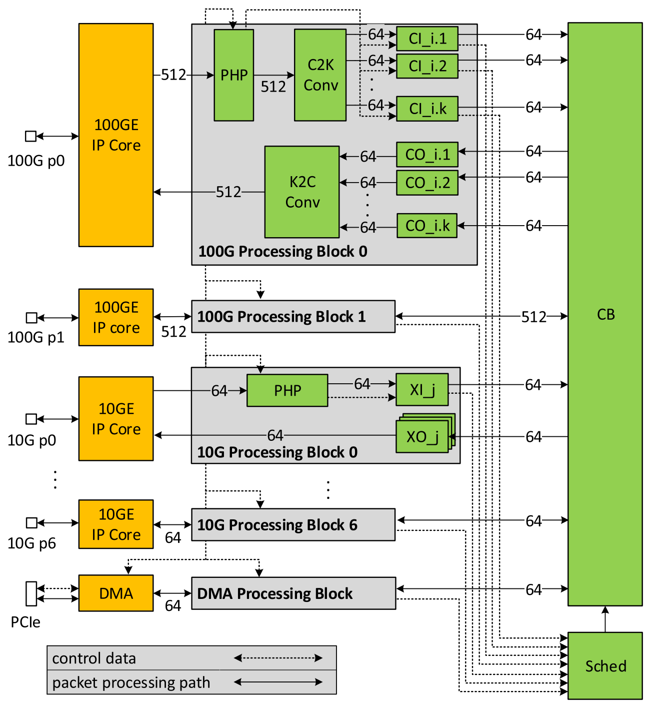

Fig. 2. Architecture of the space router.

family, Virtex Ultrascale+. Xilinx Ultrascale+ chips are the appropriate option as they support sufficient number of integrated 100G ethernet blocks, and large amount of on-chip memory. On-chip memory is a critical resource for the router implementation as it is required for different purposes: lookup tables, storage of packets, as well as for the packet segmentation and reassembly.

Xilinx Ultrascale+ chips have three types of memories that differ according to their size. Packets should be naturally stored in the largest UltraRAM memory cells, or possibly in the Block RAM cells. The maximum width of these memory cells is $M = 64$ (72 without ECC), and the memory throughput is much lower than the required router throughput. For this reason, the space router with output or shared buffers would have the limited capacity, which is too low for satellite backbone network under consideration. On the other side, multistage routers have high memory requirements as packets pass multiple buffers on their way from router inputs to outputs.

The space router with input and output buffers is the appropriate architecture for the given hardware, where each input buffer comprises one or more memory cells. Here, output buffers are used for mitigating the crossbar speedup, and the packet reassembly. Consequently, the output buffers have the same throughputs as the input buffers, and do not represent the router bottleneck.

We will assume that the space router has A 100GE ports, and B 10GE ports.

Fig. 2 shows the architecture of the space router that will be implemented and analyzed in this article. Packets first pass through packet header processors (PHPs), and, then, they are stored in the input buffers with virtual queues. 100GE input buffers are denoted by CI, and 10GE input

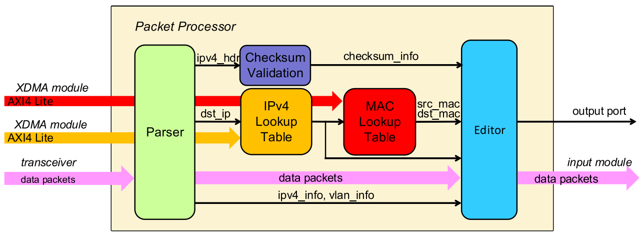

Fig. 3. Packet processor.

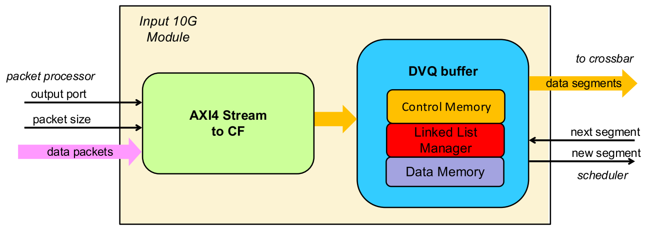

Fig. 4. Input 10G buffering.

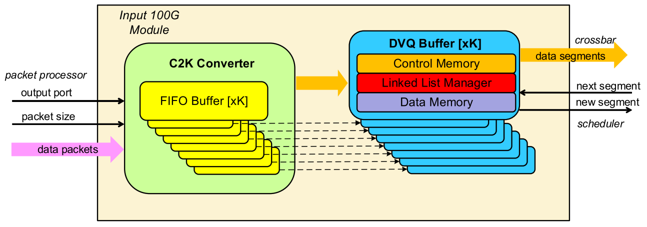

Fig. 5. Input 100G module.

buffers are denoted by XI in Fig. 2. Packet processors receive packets via the local bus (LBUS) interface from the 100GE IP cores. Packet processors are designed to have the same width as the LBUS. The LBUS width is 512 bits, while the buffers with virtual queues use the 64-bit-wide buses. For this reason, a bus converter is needed between the 100GE packet processors and input buffers with virtual queues. Such converter is not needed for 10GE ports, as they receive packets on the 64-bit-wide AXI4-Stream interface. Packets are split into the $L = 512$ bits long segments that are independently forwarded through the crossbar to the output buffers. Output buffers with virtual queues collect segments from the crossbar, and reassemble them into original packets. 100GE output buffers are denoted by CO, and 10GE output buffers are denoted by XO in Fig. 2. The 100GE output port needs a bus converter from the 64- to the 512-bits-wide buses, inverse to the converter of the 100GE input port.

A. Input Modules

Input modules perform processing and storage of packets before they are transferred to the output ports through the crossbar.

When packets arrive to the router, their headers are processed in PHPs with throughputs 100 and 10 Gbps depending on the input port, as shown in Fig. 3. Based on the IP address and lookup table, PHP determines the packet output port, as well as the MAC address of the next-hop device. PHP also determines the packet length that will be needed for packet reassembly. Standard processing is performed by PHP as well: Time to live update, validation of checksum and its update, and MAC address modification.

The Xilinx SDNet platform was used for implementation of the PHP modules. The SDNet platform allows flexible programming of PHP using the P4 language [13], for example. We, however, used the Xilinx language PX for programming of data plane. In our design, we implemented IP lookup as it allows global routing, and requires simple control plane. However, PHP can be easily modified to implement MPLS lookup or other custom-made packet processing.

The input buffering at 10 Gbps is shown in Fig. 4. Before the packets are stored into 10 Gbps buffers, their format needs to be converted from AXI4-Stream to crossbar format (CF). The bus adapter also adds the following control information to each segment: segment input port, packet length, and packet sequence number. Packet segments are, then, stored into virtual queues according to their output ports. Each virtual queue stores segments bound to a specific crossbar output port. These dynamic virtual queues (DVQs) are implemented using the linked list manager. Each queue is defined by pointers to its start and end, which are updated when segments arrive to the queue and leave the queue. One queue is dedicated to empty memory locations. Memory locations from this queue are allocated to new segments, and memory locations of departing segments are returned to the queue of empty segments. Each input buffer with virtual queues is connected to one crossbar port. These input buffers with virtual queues were denoted as XI_j in Fig. 2, where j is the input port number.

Control memory stores the control information of packet segments: segment input port, packet length, and packet sequence number.

The input buffering at 100 Gbps is more complex. Because the 100 Gbps ports have much higher throughputs than crossbar ports, each input 100 Gbps port is adapted to K crossbar ports using a converter, as shown in Fig. 5. The converter, C2K, basically converts the 512-bits-wide LBUS used by PHP to the 64-bits-wide buses of UltraRAM memory cells. Therefore, we chose to allocate $K = 8$ buffers to each port, as eight UltraRAM buffers can store packets at the same speed as they arrive from the LBUS. Converter C2K is implemented using the Distributed RAM memory that can be configured to receive packets from the 512-bits-wide bus, and to transmit packets to the 64-bits-wide bus. C2K also adds the control information to each segment, which is then stored in the control memory: segment input port, packet length, and packet sequence number.

Converter C2K comprises K different Distributed RAM buffers that transmit packets to the corresponding UltraRAM buffers. Incoming packet is stored in the emptiest buffer of C2K, while multiple packets can be stored to different buffers of C2K simultaneously. Each buffer of the converter sends packet segments to the corresponding UltraRAM buffer with DVQs, and each UltraRAM buffer is connected, in turn, to one crossbar port. The input buffer with DVQs has the same design as in the case of input 10GE ports. These input buffers with virtual queues were denoted as CI_$i.j$ in Fig. 2, where i represents the input port number, and j denotes the crossbar port number to which the buffer is attached.

B. Switching Fabric and Scheduler

Packets are divided into the segments that are stored in the input buffers with virtual queues. These segments are $L = 512$ bits long. Information about incoming segments is sent to the scheduler that determines when these segments will be transferred to the output ports through the crossbar. Based on this information, scheduler stores the number of unscheduled segments in each input virtual queue.

Generally, the crossbar matrix connects each input port to at most one output port, and each output port to at most one input port. Time is divided into slots, and the scheduler determines the input–output pairs that will be connected in each slot. The duration of a time slot equals the segment transmission time. Scheduler also sends the information to the input buffers about segments to be sent in each time slot, if any.

In the presented design, we will use the SLIP algorithm for the segment scheduling, which is used in Cisco routers [27]. The SLIP algorithm is iterative, and each iteration comprises three steps. In the first step, inputs send requests to the outputs for which they have segments. Each output selects one input port from which it received a request, and sends an acknowledgment to it. In the final step, each input selects one output from which it received an acknowledgment, and this input will send a segment to the selected output in the next time slot.

In order to simplify reassembly of packets at high speeds, input buffer CI_$i.m$ is connected to output port j via output buffer CO_$j.m$. Also, input buffer XI_i is connected to output port j via output buffer CO_$j.(i \mod K)$. In this way, reassembly of packets at the router outputs is simplified since their segments wait in the same input and output virtual queues. Still, packets can be reordered at the router outputs, as they can experience different delays at the crossbar ports of some 100GE port. So, a packet coming to the router earlier might depart its input virtual queue later, and, therefore, arrive later to the corresponding output buffer. Consequently, getting packets to their original order will be needed at the space router outputs.

The number of crossbar ports in our design can be calculated as follows:

$$S = A \cdot K + B \tag{1}$$

where A is the number of 100GE ports, and B is the number of 10GE ports, as mentioned before. Since each input buffer is connected to $A + B$ crossbar output ports, it needs to have $A + B$ virtual queues.

C. Output Modules

Reassembly of segments into packets is implemented at the output ports. Since packets could be reordered while passing the crossbar, they also must be brought back to the original order at the output ports. For these reasons, buffers are also needed at the output ports.

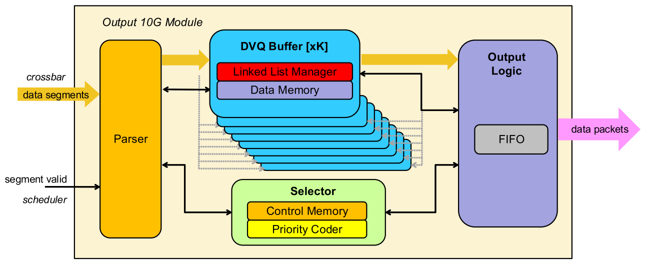

Fig. 6. Output 10G module.

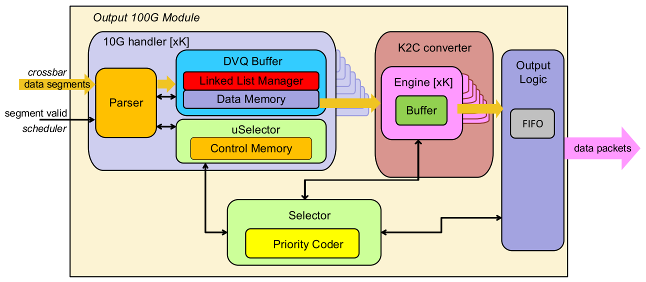

Fig. 7. Output 100G module.

The 10GE port receives packets from a dedicated crossbar output port, and it is shown in Fig. 6. This crossbar port is connected to K output buffers with DVQs. Packets are stored into the virtual queues according to their input ports. One virtual queue stores the segments coming from a specific crossbar input port that is determined by the parser. DVQs are implemented in the same way as in the input buffers. Virtual queues are assigned evenly to these K output buffers, so that each buffer stores $(A \cdot K + B)/K$ virtual queues, where B is divisible with K in our case. Selector stores the control information about segments (segment input port, packet length, and packet sequence number). Based on this information, the selector calculates the packets eligible for transmissions from virtual queues. Packets are eligible if all of their segments have arrived, and all previous packets from the same input port have been transmitted. In this way, original packet ordering is maintained. The priority encoder chooses one of the eligible packets for transmission in a round-robin fashion. This packet is then transmitted through the FIFO buffer to the network.

The output 100GE port comprises also K output buffers, as shown in Fig. 7. Each output buffer of the 100GE port is connected to one crossbar output. It stores DVQs implemented as in the input buffers. Similarly, as in the case of 10GE port, each output buffer has $(A \cdot K + B)/K$ virtual queues. Each virtual queue stores packets coming from a specific input port. Segments from one output buffer are sent to the corresponding buffer of the K2C converter. Because the 100Gbps ports have much higher throughput than crossbar ports, each output 100Gbps port is adapted to K crossbar ports using the K2C converter. The K2C converter, basically, converts the 64-bits-wide buses used by crossbar ports to the 512-bits-wide bus of the output port. K2C uses Distributed RAM because its cells can receive segments from the 64-bits-wide buses, and send segments to the 512-bits-wide bus.

The uSelector associated to an output buffer determines head-of-line packets in its virtual queues. Global selector has the information about eligible packets in all virtual queues at the output port, and decides which packets should be passed to the converter buffers. Packet is eligible if all of its segments have arrived, all previous packets from the same input have been transmitted to FIFO, and if the corresponding converter buffer is not busy. The priority encoder of the selector decides on the next packet that will be moved to converter buffer. The engines are used to move packets from virtual queues of output buffers to converter buffers. Segments are transmitted from the converter buffers through the FIFO buffer to the network, in the order in which they were written to the converter buffers.

D. Control Plane

We use the Quagga routing suite for implementation of the control plane. In particular, IP lookup tables will be calculated according to the OSPF protocol of the Quagga suite. We developed a network driver that links Quagga with the data plane via the direct memory access (DMA) module as shown in Fig. 2. DMA is connected to other ports via one 10 Gbps port. We also integrated Quagga with the SDNet software module for downloading lookup tables to the data plane.

SDNet platform is flexible, so our data plane could use other types of lookup tables and control planes. For example, MPLS lookup tables could reduce on-chip memory requirements at the expense of more complex control plane.

SDNet platform also supports centralized SDN controllers that communicate with the data plane according to the OpenFlow protocol. A distributed control plane seems more appropriate for satellite networks with large distances and delays between nodes, because reactions of a distant central controller to failures and network congestions are likely be too slow [28].

It is important to also provide reliable TT&C connections between satellite nodes and network on the ground for transmission of critical control information as shown in Fig. 1.

IV. PERFORMANCE ANALYSIS

We implemented the space router using the VHDL programming language, and it can be used on a wide range of FPGA chips. We chose the most advanced Xilinx Ultrascale+ chip provided on a development board, in order to analyze scalability of space routers. Before implementation and verification on actual hardware, individual components as well as the entire design have been thoroughly tested through simulations.

The design was implemented on the Xilinx VCU118 [29] evaluation board with VU9P Virtex Ultrascale+ chip. The board has dual QSFP28 interfaces, FMC and FMC+ card interfaces, and PCIe header. Since the board has only two QSFP28 interfaces, the HTG-FMC-X6QSFP28 FMC+ card [30] was used for additional QSFP28 interfaces. The QSFP28 interfaces are used for implementation of both 10GE and 100GE ports. Some of the QSFP28 interfaces were used to connect up to four 10GE ports by means of the QSFP+ to SFP+ splitter cables. The VU9P has a total of 345.9 Mbit of on-chip memory comprising UltraRAM, Block RAM, and Distributed RAM. The IP lookup tables are configured to have a depth of $2^{14} - 1$. Each buffer with virtual queues at input and output ports is implemented within one UltraRAM memory cell and can store up to 512 packet segments of 512 bits.

The space router was implemented using Vivado 2017.1 and PHPs were created using SDNet 2017.1. The Ultrascale+ integrated 100G ethernet subsystem IP [31] was used for implementing the 100GE interface, and, the 10G/25G ethernet subsystem IP [32] was used for the 10GE interface.

A. Resource Requirements

Table I represents utilization of the resources on the Xilinx VU9P chip after the implementation. Used resources are given as the percentages of the total available resources on the VU9P Virtex Ultrascale+ chip. Additionally, the total resources available on VU9P and VU13P are presented in the table. The VU13P is added for comparison with VU9P, as it is the UltraScale+ FGPA chip with the largest on-chip memory available.

Table I shows that the Block RAM memory is the resource with the highest utilization. Most of the Block RAM memory is used for lookup tables in PHPs, as expected. The depth of lookup tables could be, possibly, further increased since 55% of Block RAM memory is unused. Alternatively, there would be room for additional ports on the VU9P chip. More ports could be added to the design if the lookup tables were reduced by replacing IP addresses with MPLS labels.

The UltraRAM memory is used only for the components associated with input and output buffers with virtual queues. Since only around 11% of the UltraRAM memory is used for our design, it is not a critical resource and can allow larger buffers, and more ports. With the increase of the number of traffic flows, the chance that more packets will arrive to the buffer simultaneously increases, so the larger buffers are needed. Consequently, the buffer size increase enables support for finer granularity flows through the buffer.

If reliability of the design needs to be improved, several mitigation mechanisms can be applied. Memory scrubbing can be used for the FPGA configuration memory. Reliability of on-chip memories can be improved by using the integrated EDAC functionality. Local redundancies (TMR or DWC) can be implemented for core components of the design, the scheduler, crossbar, and DVQ management in input and output ports. The global TMR is not necessary for our design, since packet losses will be handled by OSPF and link layer protocols. The VU9P and VU13P FPGA chips

TABLE I
Resource Utilization [%]

Component	CLB LUTs	CLB flip-flops	Distributed RAM	Block RAM	UltraRAM
10G PHP	9.35%	6.14%	3.78%	25.41%	0%
100G PHP	3.39%	2.57%	1.32%	9.44%	0%
10G input	1.08%	0.84%	0.01%	1.28%	0.83%
100G input	2.31%	1.39%	0.90%	1.12%	1.67%
10G output	5.41%	1.86%	0.01%	2.56%	6.67%
100G output	2.31%	1.48%	0.80%	1.90%	1.67%
Other (+IP cores)	6.92%	3.49%	0.55%	3.15%	0%
Total	30.77%	17.77%	7.37%	44.86%	10.83%
VU9P available	1,182K	2,364K	36.1 Mb	75.9 Mb	270.0 Mb
VU13P available	1,728K	3,456K	48.3 Mb	94.5 Mb	360.0 Mb

have sufficient available resources for different levels of redundancy as shown in Table I.

B. Packet Throughput

Our design of the space router was tested in different scenarios, and bit rates were recorded. The VCU118 board was plugged into a PCIe slot of a computer. As we mentioned earlier, the Linux device driver and application were developed for loading lookup tables to PHPs. Computers used for testing were equipped with Mellanox ConnectX-4 MCX415A-CCAT cards with 100GE ports, and Intel X710-DA4 cards with 10GE ports, while the VCU118 board has QSFP28 interfaces used for both 100GE and 10GE ports.

The Pktgen application from Data Plane Development Kit (DPDK) was used to generate traffic with high packet rates. Pktgen allows generation of high-rate traffic with configurable packet sizes, IP addresses, and ports. The DPDK version 16.04.0 was used for Mellanox ConnectX-4 MCX415A-CCAT card and the DPDK version 17.11.3 was used for Intel X710-DA4 card, according to their capabilities.

Three different tests were performed. In test 1, each port forwards packets only to one port of the same speed. Test 2 combines traffic from multiple 10GE ports to one 100GE port, and spreads the traffic from the other 100GE port to all 10GE ports. Test 3 stresses the crossbar by forwarding packets from all input ports to all output ports.

In test 1, the Pktgen application is configured so that the packets are sent from one port only to one other port with the same port speed, as shown in Fig. 8.

If 100GE ports are denoted as $C_i,\ i \in \{0, 1\}$, and 10GE ports as $X_j,\ \ j \in \{0..6\}$, then, the crossbar connections for test 1 are made in the following manner:

$$C_0 \leftrightarrow C_1$$
$$X_i \rightarrow X_{(i+1)\bmod 7}.$$

The resulting throughputs summed across all output ports, 100GE ports, and 10GE ports are shown in Fig. 9.

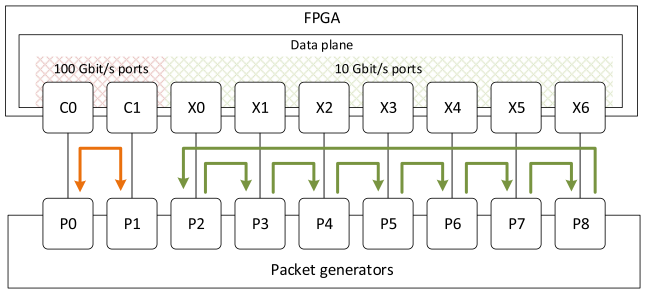

Fig. 8. Traffic pattern in test 1.

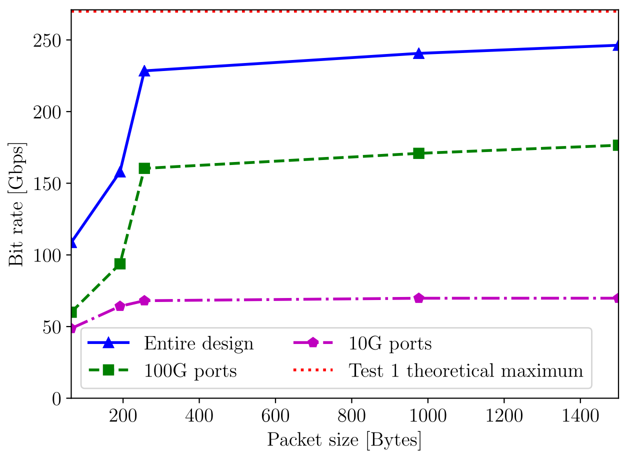

Fig. 9. Achieved throughputs in test 1.

The dotted red line in Fig. 9 marks the maximum bit rate of the design with two 100GE ports, and seven 10GE ports, which is 270 Gbps. This test measures maximum throughputs of ports for different packet lengths. Bit rates are lower (40–60% of the maximum throughputs) for packet lengths under 256 B and reach 90% of the maximum throughputs for larger packet lengths. It can be observed that the 100GE ports exhibit a sharp drop in efficiency for packets shorter than 256 B, while 10GE ports process shorter packets more efficiently.

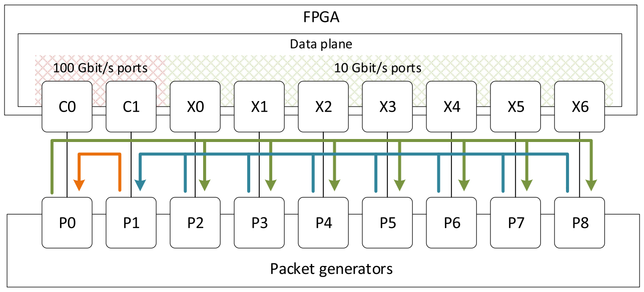

Fig. 10. Traffic pattern in test 2.

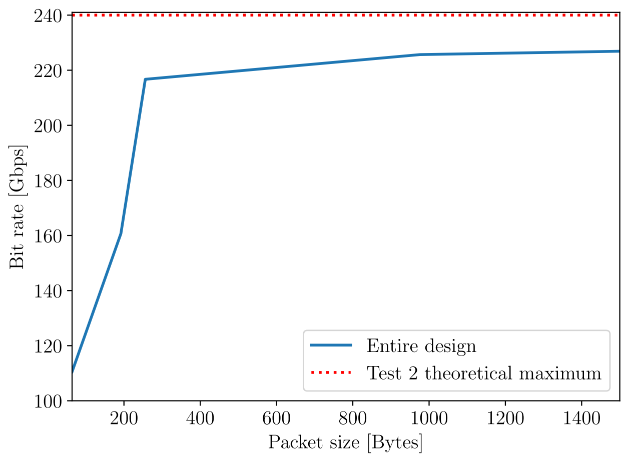

Fig. 11. Achieved throughput in test 2.

In the second test (Fig. 10), the first 100GE port, C_0, forwards packets to all 10GE ports, all 10GE ports forward packets to the second 100GE port, C_1, and port C_1 forwards packets to port C_0:

$$C_0 \rightarrow X_i,\ i \in \{0..6\}$$
$$X_i \rightarrow C_1,\ i \in \{0..6\}$$
$$C_1 \rightarrow C_0.$$

Theoretical maximum throughput for this configuration is 240 Gbps.

Shape of the curve in Fig. 11 is similar to the shape of the curve in Fig. 9. However, distribution of bit rates among specific ports is different. Comparison of bit rates for different traffic paths in test 2 is given in Fig. 12.

The maximum throughput of one 100GE port is obviously higher than of 7×10GE ports. But, it can be observed that the total bit rates obtained at the outputs of 10GE ports for shorter packets are much higher than the bit rate obtained at the output of the 100GE port C_0. Utilization of 100GE ports increases with packet length, but it is lower than the utilization of 10GE ports for all packet lengths.

In test 3, packets from all input ports are forwarded to all output ports, and this is a stress test for the crossbar scheduler (Fig. 13).

Traffic distribution is configured so that eight times more packets are directed to the 100GE output ports than to the 10GE output ports, because each 100GE port is implemented by $K = 8$ times more input buffers. If a port in the design, regardless of its speed, is denoted as I_p, $I_p \in$

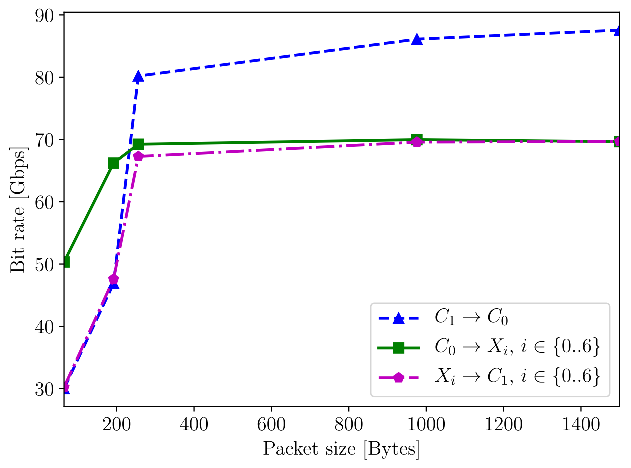

Fig. 12. Bit rates for different traffic paths in test 2.

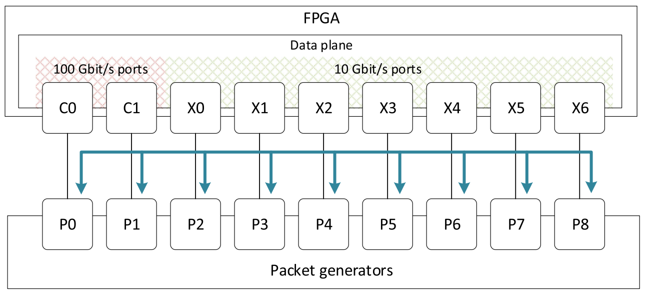

Fig. 13. Test 3 setup.

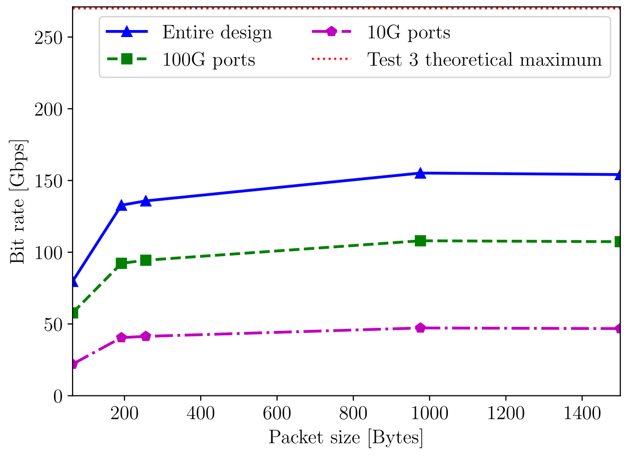

Fig. 14. Achieved throughputs in test 3.

$\{C_i, X_j\},\ i \in \{0, 1\},\ j \in \{0..6\}$, then, test 3 can be described as

$$I_a \rightarrow I_b$$
$$\mathrm{P}\,(I_b = C_i | I_a) = 8/23$$
$$\mathrm{P}\left(I_b = X_j | I_a\right) = 1/23.$$

The resulting throughputs summed across all output ports, 100GE ports, and 10GE ports in test 3 are shown in Fig. 14. The throughput of 100GE ports for packet lengths up to 200 B is the same as in test 1, which implies that it is limited by packet processing. For longer packets, the throughput of 100 GE is lower than in test 1 as the packet

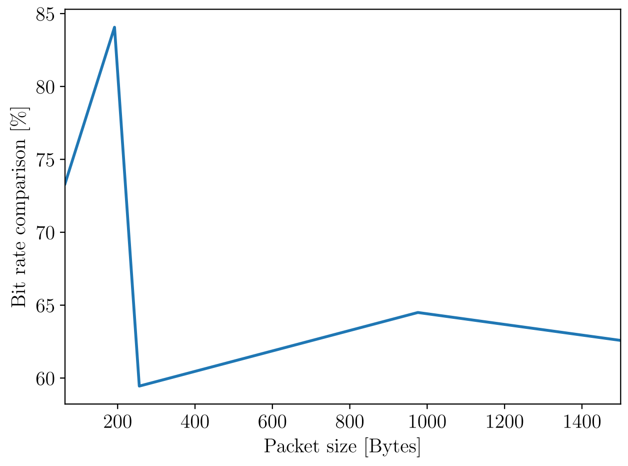

Fig. 15. Ratio of bit rates in test 3 and test 1.

scheduler becomes a bottleneck. The packet scheduler reduces throughput of 10GE ports, and their total throughput is lower than in test 1 for all packet lengths.

Fig. 15 shows the ratio of throughputs in test 3 and test 1. This ratio is between 60% and 65% for the packets longer than 256 B which matches the efficiency of the one-iteration SLIP algorithm for uniform traffic [27]. For the packet lengths below 256 B, the ratio of throughputs in tests 3 and 1 is much higher as the utilization of 100GE ports is limited by packet processing in both tests.

V. CONCLUSION

In this article, we presented an implementation of the space router for a satellite node. We showed that the router with two 100GE ports and seven 10GE ports can be implemented on the Xilinx Ultrascale+ chip with intermediate capabilities available in the currently most advanced development platform. The Ultrascale+ chips with more resources are likely to support the space router with significantly larger throughputs.

The implemented router has input buffers with 10 Gbps throughputs in order to fit the FPGA architecture. For this reason, 100GE ports use multiple input and output buffers, and packets might take different paths through the space router. So, we had to implement elaborate segmentation and reassembly mechanisms at high speeds. Our router is designed for the proposed satellite backbone network. As the satellite network will be connected to the terrestrial IP network, it is desirable that our router is an IP router as well. For this reason, it can be used as a ground router as well. It is implemented on the FPGA chip as these chips are configurable, and can adapt in the case of radiation effect like SEU, or if design specifications change. If more reliable reaction to disturbances is needed, redundancies can be readily introduced to our design.

We showed that the implemented space router can achieve throughputs up to 240 Gbps for the longest packets. The achieved throughputs exceeded 200 Gbps for packets longer than 256 B.

REFERENCES

[1] R. Sorace
Overview of multiple satellite communication networks
IEEE Trans. Aerosp. Electron. Syst., vol. 35, no. 4, pp. 1362–1368, Oct. 1999.

[2] B. Barritt and V. Cerf
Loon SDN: Applicability to NASAs next-generation space communications architecture
In *Proc. IEEE Aerosp. Conf.*, 2014, pp. 349–359.

[3] I. del Portillo, B. G. Cameron, and E. F. Crawley
A technical comparison of three low earth orbit satellite constellation systems to provide global broadband
In *Proc. 69th Int. Astronautical Congr.*, IAF, 2018, pp. IAC–18–B2.1.7.

[4] B. Patnaik and P. Sahu
Inter-satellite optical wireless communication system design and simulation
IET Commun., vol. 6, no. 16, pp. 2561–2567, Nov. 2012.

[5] R. J. Aniceto *et al.*
Heavy ion radiation assessment of a 100G/200G commercial optical coherent DSP ASIC
In *Free-Space Laser Communications XXXI*, H. Hemmati and D. M. Boroson Eds. SPIE, Mar. 2019, p. 37. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10910/2513244/Heavy-Ion-radiation-assessment-of-a-100G-200G-commercial-optical/10.1117/12.2513244.full

[6] A. Fernández-León
Field programmable gate arrays in space
IEEE Instrum. Meas. Mag., vol. 6, no. 4, pp. 42–48, Dec. 2003.

[7] S. Felix, T. Vladimirova, J. Ilstad, and O. Emam
Availability analysis for satellite data processing systems based on SRAM FPGAs
IEEE Trans. Aerosp. Electron. Syst., vol. 52, no. 3, pp. 977–989, Jun. 2016.

[8] L. Sterpone, M. Porrmann, and J. Hegemeyer
A novel fault tolerant and runtime reconfigurable platform for satellite payload processing
IEEE Trans. Comput., vol. 62, no. 8, pp. 1508–1525, Aug. 2013.

[9] R. Perlman
Interconnections: Bridges, Routers, Switches, and Internetworking Protocols. 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[10] N. Zilberman, P. M. Watts, C. Rotsos, and A. W. Moore
Reconfigurable
network systems and software-defined networking
Proc. IEEE, vol. 103, no. 7, pp. 1102–1124, Jul. 2015.

[11] H. Toyodo, G. Ono, and S. Nishimura
100Gbe PHY and MAC layer implementations
IEEE Commun. Mag., vol. 48, no. 3, pp. 541–547, Mar. 2010.

[12] K. Yamazaki, N. Yoshihiro, H. Takahiro, H. Takahashi, and M. Akihiko
Innovating a reprogrammable network with SDNet
Xcell Softw., no. 2, pp. 24–29, 2015.

[13] P. E. A. Bosshart
P4: Programming protocol-independent packet processors
ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 87–95, 2014.

[14] A. Smiljanić and Z. Čiča
A comparative review of scalable lookup algorithms for IPv6
Comput. Netw., vol. 56, no. 13, pp. 3040–3054, 2012.

[15] W. Jiang and V. Prasanna
Scalable packet classification on FPGA
IEEE Trans. VLSI Syst., vol. 20, no. 9, pp. 1668–1680, Sep. 2012.

[16] Y. R. Qu and V. Prasanna
High-performance and dynamically updatable packet classification engine on FPGA
IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 1, pp. 197–209, Jan. 2016.

[17] S. Saponara, L. Fanucci, M. Tonarelli, and E. Petri
Radiation tolerant spacewire router for satellite on-board networking
IEEE Aerosp. Electron. Syst. Mag., vol. 22, no. 5, pp. 3–12, May 2007.

[18] S. Parkes, C. McClements, D. McLaren, A. F. Florit, and A. G. Villafranca
Spacefibre networks: Spacefibre, long paper
In *Proc. Int. SpaceWire Conf.*, Oct. 2016, pp. 1–6.

[19] H. Quinn
Radiation effects in reconfigurable FPGAs
Semicond. Sci. Technol., vol. 32, no. 4, Apr. 2017, Art. no. 044001.

[20] F. Smith and S. Mostert
Reconfigurable FPGA computing to mitigate for total ionizing dose effects
In *Proc. IEEE Aerosp. Conf.*, Mar. 2007, pp. 1–13.

[21] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam
Mitigation of radiation effects in SRAM-based FPGAs for space applications
ACM Comput. Surv., vol. 47, no. 2, pp. 37:1–37:34, Jan. 2015.

[22] A. Smiljanić
Flexible bandwidth allocation in high-capacity packet switches
IEEE/ACM Trans. Netw., vol. 10, no. 2, pp. 287–293, 2002.

[23] M. Petrović, A. Smiljanić, and M. Blagojević
Design of the switching controller for the high-capacity non-blocking internet router
IEEE/ACM Trans. VLSI Syst., vol. 17, no. 8, pp. 1157–1161, Aug. 2009.

[24] A. Smiljanić
Rate and delay guarantees provided by Clos packet switches with load balancing
IEEE/ACM Trans. Netw., vol. 16, no. 1, pp. 170–181, Feb. 2008.

[25] M. Binesh Marvasti and T. H. Szymanski
An analysis of hypermesh NoCs in FPGAs
IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 10, pp. 2643–2656, Oct. 2015.

[26] M. Antić and A. Smiljanić
Throughput of reliable networks with load balanced shortest path routing
In *Proc. IEEE IEEE Global Telecommun. Conf.*, 2010, pp. 1–5.

[27] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellarsick, and M. Horowitz
Tiny tera: A packet switch core
IEEE Micro, vol. 17, no. 1, pp. 26–33, Jan./Feb. 1997.

[28] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford
Central control over distributed routing
In *Proc. ACM SIGCOMM (Best Paper Award)*, 2015, pp. 43–56.

[29] Xilinx Virtex UltraScale+ FPGA VCU118 evaluation kit. (2018). [Online]. Available: https://www.xilinx.com/products/boards-and-kits/vcu118.html

[30] 6-port QSFP28 (6 × 100G) / QSFP+ (6 × 40G or 6 × 56G) FMC+ module (Vita57.4). (2018). [Online]. Available: http://www.hitechglobal.com/FMCModules/x6QSFP28.htm

[31] UltraScale+ integrated 100 G ethernet subsystem. (2018). [Online]. Available: https://www.xilinx.com/products/intellectual-property/cmac_usplus.html

[32] 10G/25G ethernet subsystem. (2018). [Online]. Available: https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

Strahinja Janković received the B.Sc. and M.Sc. degrees in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, in 2011 and 2012, respectively. He is currently working toward the Ph.D. degree in embedded systems with the Department of Electronics at the same school.

He works as a Teaching and Research Assistant with the School of Electrical Engineering, University of Belgrade. His research interests include power management of embedded systems, and FPGA design and operating systems.

Aleksandra Smiljanić [M'96] received the B.Sc. degree from Belgrade University, Serbia, in 1993, and the M.A. and Ph.D. degrees from the Princeton University, Princeton, NJ, USA, in 1996 and 1999, respectively, all in electrical engineering.

She currently works as a Professor with the Belgrade University. She worked for AT&T Labs Research from 1999 until 2004. She is the author of numerous conference and journal papers in the area of high-performance switching and routing, and is the inventor of 11 U.S. patents in this area. She was mentor on four papers that received awards at scientific conferences. Her research interests include high-performance switching and routing.

Dr. Smiljanić is the author of the Best Papers at the IEEE Conference on High Performance Switching and Routing in 2000 and 2002. She received the Research Excellence Award at AT&T Labs in 2000. She served as an Editor of the IEEE COMMUNICATION LETTERS and *OSA Journal on Optical Networking*. She has also served as a Guest Editor of the IEEE JSAC Special Issue on Switching and Routing for Scalable and Energy Efficient Data Center Networks.

Mihailo Vesović received the B.Sc. and M.Sc. degrees in electrical and computer engineering, in 2013 and 2015, respectively, from the School of Electrical Engineering, Belgrade University, where he is currently working toward the Ph.D. degree.

He is a Research Assistant with the School of Electrical Engineering. His research interests include networking, algorithms, and FPGA design.

Mr. Vesović received a scholarship from the Serbian Ministry of Science and Education as one of the best students in his class. In 2015, he received the Best Student Paper Award in the communication section of the international IcETRAN conference.

Hasan Redžović received the B.Sc. and M.Sc. degrees in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, in 2013 and 2014, respectively. He is currently working toward the Ph.D. degree in the communication area.

He is a Research Assistant with the Innovation Center, School of Electrical Engineering. He is the author of numerous conference and journal papers, as well as technical solutions. His research interests include Internet routers, software-defined networking, and network optimization and security.

Marija Bežulj received the B.Sc. and M.Sc. degrees in electrical and computer engineering, in 2013 and 2014, respectively, from the School of Electrical Engineering, University of Belgrade, Serbia where she currently working toward the Ph.D. degree.

After several years of developing computer vision algorithms in industry, she joined the Department of Electronics, School of Electrical Engineering, University of Belgrade, where she is currently a Teaching and Research Assistant. Her research interests include machine vision, machine learning, algorithms, and FPGA design.

Andreja Radošević received the Ph.D. degree from the UC San Diego's Department of Electrical and Computer Engineering, in 2012.

In 2012, he joined Qualcomm, San Diego, CA, USA, where he was one of the key contributors to development of baseband modems supporting 4G-LTE, WCDMA, and TD-SCDMA standards. His role included algorithm designs for time and frequency synchronization, turbo decoding, power control, and interference cancellation. In 2016, he joined FocusMotion, Seatle, WA, USA, a company that develops machine learning and sensor fusion algorithms for human motion recognition. In 2018, he founded Stellar Labs, LLC in an effort to join the wireless and Internet-of-Things industry frontier by delivering disruptive and state-of-the-art system solutions. His research interests include information and coding theory, adaptive modulation, channel estimation and equalization, and spectrally efficient MIMO-OFDMA transmission schemes, mainly in the context of underwater acoustic communications.

Slaven Moro received the Ph.D. degree in optoelectronics from the Department of Electrical and Computer Engineering, University of California–San Diego, in 2010, for his pioneering work in wideband low-noise optical parametric amplification and wavelength conversion in nonlinear fibers.

He worked as a Senior Scientist with the Ziva Corporation, where he was responsible for the design and implementation of optical physical layer decryption systems utilizing blind source separation, independent components analysis, nonlinear/nonstationary source separation in convolutive mixtures and statistical randomness testing. From 2011 to 2015, he led the teams working on high-energy lasers, high-power coherent fiber amplifiers, and coherent LIDAR sensors at General Atomics. In June 2015, he joined Facebook Connectivity, where he has been leading a team developing novel high-capacity optical and millimeter-wave technologies for communication systems.