FPGA Network Function Acceleration: A Space Division Multiplexing Approach

Xin Dong[1,2], Lizhuang Tan[1,2], Huiling Shi[1,2], Wei Zhang[1,2], Peiying Zhang[1,3]

[1]*Key Laboratory of Computing Power Network and Information Security, Ministry of Education*
Shandong Computer Science Center (National Supercomputer Center in Jinan)
Qilu University of Technology (Shandong Academy of Sciences), Ji'nan, China
[2]*Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing*
Shandong Fundamental Research Center for Computer Science, Jinan, China
[3]*Qingdao Institute of Software, College of Computer Science and Technology*
China University of Petroleum (East China), Qingdao, China
Email: 10431240209@stu.qlu.edu.cn, lzhtan@qlu.edu.cn, shihl@sdas.org, wzhang@sdas.org, zhangpeiying@upc.edu.cn

***Abstract*—FPGA-based SmartNICs are increasingly deployed in cloud and edge data centers to offload cryptography, packet shaping and in-network computing. While recent virtualisation techniques (e.g., SR-IOV) allow multiple tenants to share a single FPGA accelerator, each tenant receives a fixed portion of logic resources and must load a private copy of every required network function. How to improve the resource utilization of FPGA accelerators, meet the stringent service quality constraints of multi-tenants, and provide resource sharing fairness are the main goals of FPGA network function acceleration. We present FPGA-SDM, a virtualized FPGA resource sharing method based on space division multiplexing. In FPGA-SDM, each tenant is issued a private vFPGA domain realised through SR-IOV queues and DMA engines, while a reconfigurable public service pool hosts the library of packet-processing acceleration function that are loaded on demand via partial reconfiguration.Our moduel show that, under tenant activity levels of 0.3–0.6 and sharing ratios of 0.4–0.7, the system achieves up to 1.5× performance improvement over static resource allocation. Moreover, the experimental results confirm that FPGA-SDM achieves line-rate throughput, maintains near-optimal throughput levels under the influence of malicious tenants, and improves average resource utilization up to 49.5%.**

***Index Terms*—Field-Programmable Gate Array, Partial Reconfiguration, Single Root I/O Virtualization, Peripheral Component Interconnect Express.**

I. Introduction

Field-Programmable Gate Array (FPGA) has evolved from application-specific accelerators into first-class citizens of modern data-center networks, where they appear as *SmartNICs* that can offload cryptography, switching and storage functions at line rate [1]. Cloud service providers now seek to expose these capabilities to multiple tenants simultaneously, however, the prevailing slice-based approach—statically carving a device into fixed partitions—struggles to meet the elasticity, fairness and security requirements of multi-tenant deployments [2]. In multi-tenant environments, FPGA virtualization is challenged by.

1) **Elastic allocation vs. Static allocation.** When each tenant is hard-bound to a predetermined share (e.g., 10 % LUTs and 1 Gbps bandwidth), bursts above that quota trigger throttling even if idle resources exist elsewhere, while long idle periods waste silicon [3].
2) **Redundant instantiation.** Tenants loading some copies of the same network function (e.g. AES) dramatically inflate LUT/BRAM usage and configuration latency.
3) **Fairness under adversarial demand.** Malicious or mis-configured tenants can oversubscribe PCIe and on-board DRAM, degrading the tail latency of well-behaved users unless hardware policing is present [4].

To overcome these obstacles, we propose FPGA-SDM, a virtualized FPGA resource sharing method based on space division multiplexing. FPGA-SDM treats accelerators as micro-services and introduces two key abstractions.

- A **private *v*FPGA domain** per tenant, created with SR-IOV over PCIe, guaranteeing hardware isolation of queues and DMA flows.
- A **reconfigurable public service pool** that hosts a library of reusable packet-processing functions (e.g. encryption, firewall, and parsing). Modules are loaded and unloaded at run time via Partial Reconfiguration(PR) within sub-millisecond latency [4].

A hardware monitor combines a sliding-window counter with a token-bucket rate limiter: it detects sustained oversubscription, throttles abusive flows, or isolates the culprit preserving fairness under contention or attack.

Compared with prior SmartNIC virtualisation efforts such as SuperNIC [5] and FFIVE [6], FPGA-SDM makes the following contributions.

- **Elastic multiplexing.** Resources migrate on demand between tenants, boosting average utilisation by up to 50% without violating service-level agreements.

This work was supported in part by the Shandong Provincial Natural Science Foundation under Grant No. ZR2022LZH015, ZR2023QF025, ZR2023LZH017 and ZR2024MF066, the National Research Foundation of Korea under Grant NRF-24109 FY2025, the National Natural Science Foundation of China under Grant 62402257 and 62471493, the China University Research Innovation Fund under Grant No.2023IT207, and the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant No.2023RCKY141.

- **Function-level sharing.** A single hardware instance of a network function is transparently reused by many tenants, eliminating redundant LUT/BRAM footprints.
- **Hardware fairness enforcement.** The proposed sliding-window + token-bucket monitor maintains near-optimal throughput levels under the impact of malicious tenants.
- **Prototype and evaluation.** An end-to-end implementation on a Xilinx Alveo U50 shows that, under 50 concurrent tenants, FPGA-SDM preserves over 95 Gbps aggregate throughput.

The remainder of this paper is structured as follows. Section 2 surveys related work. Section 3 details the FPGA-SDM architecture and implementation. Section 4 presents simulation and hardware evaluation results. Section 5 concludes and outlines future work.

II. Related Work

At present, there are various solutions in the field of network function virtualization for FPGA.

SuperNIC maps network tasks to physical chains through PR, providing a multi-tenant SmartNIC, but lacks efficient I/O virtualization. FPGA virtualization technology uses SR-IOV to divide a single FPGA into multiple virtual function (VF) to achieve multi-tenant sharing. FPGAVirt leverages Virtio to implement an efficient communication scheme between virtual machines and the FPGA. However, a fixed resource allocation model is difficult to cope with fluctuations in user demand and is prone to resource idle or bottleneck issues [7]. Like Nimblock [8], managing virtualization resources within a single FPGA, creating isolated and secure environments for each tenant, with each tenant having their own dedicated resources. FFIVE utilizes Kubernetes to deploy virtual connections based on FPGA, enabling FPGA migration in virtual networks as needed.

Partial Reconfiguration (PR) technology [9], allows for on-demand loading of specific functional modules at runtime, improving hardware flexibility, but existing solutions are mostly static deployments and lack efficient dynamic scheduling mechanisms. HiPR [10] allows developers to define partially reconfigurable C/C++ functions as an open-source framework, which facilitates incremental compilation of FPGA. PR-ESP [11] is an open-source platform for a system-level design flow of partially reconfigurable FPGA-based SoC architectures targeting embedded applications that are deployed on resource-constrained FPGAs. S-NIC [12] pervasively virtualizes hardware accelerators, enforces single-owner semantics for each line in on-NIC cache and RAM, and provides dedicated bus bandwidth for each network function.

Network function sharing and malicious detection [13], traditional software scheduling sharing strategies are difficult to achieve precise isolation at the hardware level, and lack timely restrictive measures for malicious user competition. LemonNFV [14] is a novel NFV framework that can consolidate heterogeneous NFs without code modification.

In contrast, the FPGA-SDM solution proposed in this paper combines virtualization, partial reconfiguration, shared reuse, and dynamic scheduling mechanisms at the hardware level, effectively addressing the shortcomings of existing technologies in multi-tenant network function reuse, resource management, and malicious detection.

III. System Architecture

The system architecture of this paper mainly includes three modules, as shown in the Fig. 1, private user resource area: allocating independent virtual function (VF) [15] to each user through the SR-IOV technology on the FPGA network card, each VF has an independent sending and receiving queue and bandwidth limit, achieving hardware level isolation; Public network service resource area: preloading or dynamically loading general network function modules (such as AES encryption and decryption, data compression, firewall, etc.) as needed, shared by multiple VFs, using pipeline and priority scheduling mechanisms to improve resource utilization; Resource Control Center: located on the Physical Function (PF) [16] of the FPGA network card, responsible for maintaining the Hardware Match Table, conducting dynamic resource allocation, scheduling decisions, and malicious user detection, and implementing traffic control through token buckets and sliding window mechanisms.

A. Hardware Design

FPGA-SDM targets network-acceleration workloads around four key components, an SR-IOV PCIe interface, a PCIe DMA controller [17], a packet switch, and a cluster of Network Function accelerators.

- **SR-IOV PCIe**. SR-IOV is used to enable virtualization on the FPGA, partitioning it into one physical function (PF) and multiple virtual functions (VFs). VF not only includes independent vFPGA accelerators to provide to different tenants to meet different needs, but also multiple VF shared functional resource areas. Our framework has shown four VFs corresponding to four different tenants. Tenants can apply for network function acceleration to the public network service component by sending data packets. SR-IOV PCIe uses transaction layer packets to describe the functions, which can achieve more user traffic direct memory access.
- **PCIe DMA Controller**. Determine whether to forward the tenant's data packet through the transaction layer, based on the packet descriptor field, and can manage the interaction between FPGA and host through pipeline to efficiently distribute multi-tenant data packets without additional loss. The load balancer retrieves from the function matching table according to the functional requirements of the data packet. If the task is completed, the data packet is developed to the corresponding vFPGA accelerator for data packet conversion and allocation. Otherwise, the data packet will be blocked or forwarded to Ethernet.

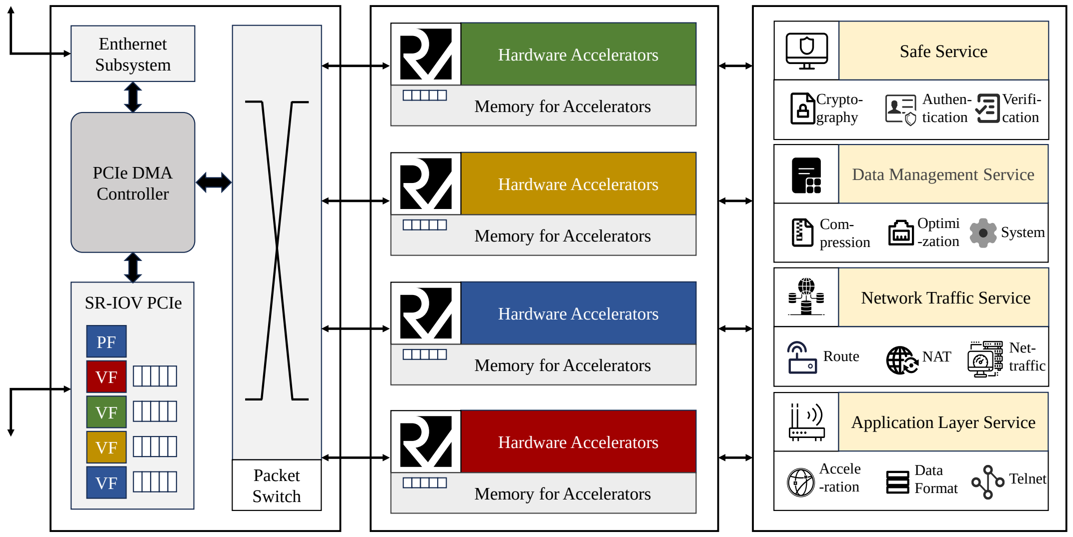

Fig. 1. The hardware of FPGA-SDM framework.

- **Packet Switch**. It can transmit data packets to the corresponding vFPGA accelerator, and the Ethernet subsystem supports network access and is embedded in a Layer 2 switch. It allocates rules for other PF based on the destination MAC address of the incoming data packet. It is used to perform control plane tasks and can centrally manage and control FPGA resources.
- **Hardware Accelerator**. RISC-V [18] can run in the form of a soft-core on FPGA, offering high flexibility and customizable instruction sets. In a multi-tenant environment, FPGAs need to be dynamically allocated to different users while ensuring isolation and performance. RISC-V's virtualization technology can be used to load different bitstreams into the FPGA sections corresponding to each tenant, and it can ensure that different FPGA accelerators cannot access beyond their boundaries.

B. Software Design

The FPGA-SDM complements the hardware datapath with two software-defined partial reconfigurable public functions and tenant-aware resource management that jointly deliver elasticity, isolation, and real-time mitigation of abusive workloads.

- **Reconfigurable Public Service Pool**. In FPGA, resource sharing and reuse represent prevalent resource optimization strategies, primarily categorized into time multiplexing and space multiplexing. In this context, spatial multiplexing is employed to partition a segment of FPGA hardware resources as a shared public resource domain among multiple tenants. Within this domain, the resources are further divided into multiple independent functional regions, each capable of operating autonomously. In particular, each partition can load distinct network acceleration bitstreams and be invoked by heterogeneous hardware accelerators to implement diverse network acceleration functionalities.
  Additionally, this shared domain is designated as a dynamically partial reconfigurable region. Each PR partition within this region exclusively occupies dedicated Block Random Access Memory (BRAM) [19] and logic resources, thereby eliminating contention risks. Hardware accelerators are directly routed to specified PR region in real time, aligning with tenant-specific acceleration demands.
- **User Management**. In order to realize dynamic function sharing and resource scheduling in the above system, this paper implements partial reconfiguration and shared scheduling algorithms in hardware, shown in Algorithm 1. The main algorithms include: parsing the TLP descriptor of the data packet, extracting the user ID, function ID and priority information from it, using the symmetric hash algorithm to determine the queue to which the data packet belongs, and deciding whether to call the existing public function module or trigger partial reconfiguration to load a new module based on the matching table. Malicious user detection algorithm, the sliding window countingand token bucket mechanism are used to monitor the request rate of each user in real time. If the preset threshold is exceeded in multiple consecutive windows, the token generation rate of the user is dynamically reduced or its request is transferred to a low-priority queue, and its resources are directly isolated if necessary. In addition, this paper also provides a hardware acceleration solution based on the AES encryption and decryption module and the RSA authentication module implemented on the FPGA network card to reduce the pressure of

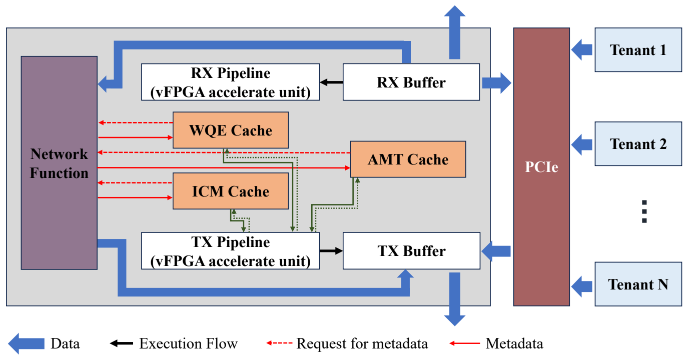

Fig. 2. The service function chain of FPGA-SDM.

network encryption operations on the host CPU and support high-speed processing of security protocols such as QUIC.

C. Service Function Chain

In the data reception pipeline illustrated in Fig. 2, data from multiple tenants is first transferred to the FPGA via the PCIe interface. It is then buffered in the RX module, which acts as a staging area for subsequent processing stages. Next, the cached data stream enters the RX Pipeline, serving as the vFPGA acceleration unit, where the hardware acceleration capabilities of the FPGA are utilized to perform targeted processing on the data, thereby enhancing the processing efficiency. During this process, based on processing requirements, metadata requests are sent to the work queue cache, address mapping table cache, infiniBand connection manager cache, and the obtained metadata is returned to the Network Function module to assist with processing. Finally, the data enters the Network Function module for subsequent processing operations related to network functions.

As shown in Fig. 2, once the data is processed by the Network Function module, it is forwarded to the TX pipeline. This stage serves as the vFPGA acceleration unit, performing pre-transmission optimization to enable efficient data delivery. Then, the data that has undergone acceleration processing flows into the TX Buffer for caching, ensuring that the data is arranged in an orderly manner before being sent to the tenants and preventing transmission congestion. Finally, the processed data is accurately sent back to the corresponding Tenant N via the PCIe interface, completing the entire data processing and transmission process.

IV. Simulation and Experimental Evaluation

To quantitatively assess the performance benefits of our proposed FPGA-SDM dynamic resource sharing framework, we developed a comprehensive simulation model focusing on how resource multiplexing affects system utilization and performance under different operational conditions.

Algorithm 1 Sliding Window Token Bucket for Malicious Tenant Detection

```
Require: $packet$: incoming packet, $tenant$: tenant state object
Ensure: Boolean: whether the tenant is malicious
 1: $currentTime = \text{get\_timestamp}()$
 2: $elapsed = currentTime - tenant.lastUpdate$
 3: $tenant.tokens = \min(tenant.tokenCapacity, tenant.tokens + elapsed \times tenant.fillRate)$
 4: $tenant.lastUpdate = currentTime$
 5: if $packet.size \leq tenant.tokens$ then
 6:     $tenant.tokens = tenant.tokens - packet.size$
 7:     $consumed = packet.size$
 8: else
 9:     $consumed = 0$
10:     $tenant.anomalyCount = tenant.anomalyCount + 1$
11: end if
12: while $currentTime \geq tenant.windowEnd$ do
13:     Remove $tenant.windows[0]$
14:     Append 0 to $tenant.windows$
15:     $tenant.windowEnd = tenant.windowEnd + tenant.windowInterval$
16: end while
17: $tenant.windows[-1] = tenant.windows[-1] + consumed$
18: if $\sum(tenant.windows) > tenant.globalThreshold$ or $tenant.anomalyCount \geq tenant.consecutiveLimit$ then
19:     return True
20: else
21:     return False
22: end if
```

A. Simulation Setup

The simulation targets a multi-tenant environment where each tenant's resource activity level varies dynamically. Key parameters are set as follows.

- **Overall performance gain** (G): The net profit obtained by the system through resource sharing, between -1 and 2.
- **Sharing Ratio** (p): The proportion of FPGA resources designated for elastic sharing, varied from 0.1 to 0.9 in increments of 0.1.
- **Tenant Activity Level** (a): Represents the average load or utilization of each tenant, ranging from 0.05 (light load) to 1.0 (full load), sampled at 20 intervals.
- **Tenant Number** (N): Fixed at 10 for baseline experiments, but also evaluated under $N = 5$, 20, and 50 to explore scalability impacts.

Resource contention and coordination overhead are incorporated into the gain model, with the final system performance gain adjusted by a logarithmic function of the tenant number.

The normalized performance gain of our FPGA-SDM mul-

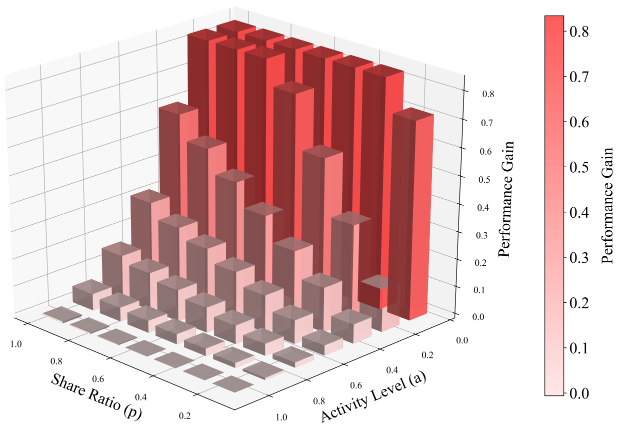

Fig. 3. Performance gain under different conditions of sharing ratio and activity level.

tiplexing scheme Equation as follow.

$$G(p,a,N) = \underbrace{\left[p\left(\frac{1}{\bar{a}}-1\right) - c\left(\frac{p}{\bar{a}}\right)^k\right]}_{g_{\text{net}}} \times \frac{1}{\ln(N+1)} \tag{1}$$

- g_{net} is the actual net income,
- $\bar{a} = \frac{1}{N}\sum_i a_i$ is the mean activity,
- $c > 0$ and $k \geq 2$ model contention penalties,
- the factor $1/\ln(N+1)$ accounts for coordination overhead as the tenant count N grows.

g_{net} captures the trade-off between reusable function benefits, $p(1/\bar{a}-1)$, and contention penalties, $c(p/\bar{a})^k$. We then clamp g_{net} into the interval $[-1, 2]$ to avoid unbounded extremes, and finally scale by $1/\ln(N+1)$ to reflect increasing coordination cost as more tenants share the pool.

B. Evaluation Metrics

The core evaluation metric is the **performance gain** achieved by dynamic resource sharing, defined as the relative improvement in effective resource utilization compared to a static resource allocation baseline. The model captures two factors.

1) **Statistical multiplexing benefits**: Reduced resource wastage when tenant loads are not fully overlapping.
2) **Resource contention penalties**: Performance degradation due to excessive simultaneous demand.

C. Simulation Results

Simulation results are visualized through Fig. 3, where the X-axis represents tenant activity level (a), the Y-axis represents sharing ratio (p), and the Z-axis shows the resulting normalized performance gain.

Key observations include:

- **Positive Gains Region**: When tenants exhibit moderate activity levels ($a \approx 0.3$–0.6) and moderate sharing ratios ($p \approx 0.4$–0.7), the system achieves noticeable performance gains up to 1.5X compared to static allocation.
- **Over-sharing Risks**: As p increases beyond 0.8, performance gains start to decline, highlighting the adverse impact of excessive sharing-induced contention.
- **Low-activity Limitation**: Extremely low activity ($a < 0.1$) yields marginal gains because base resource demands are too small to exploit multiplexing benefits effectively.
- **Scalability Analysis**: Experiments with larger tenant numbers ($N = 20, 50$) show that while multiplexing opportunities increase, so does coordination overhead, requiring careful design of control mechanisms.

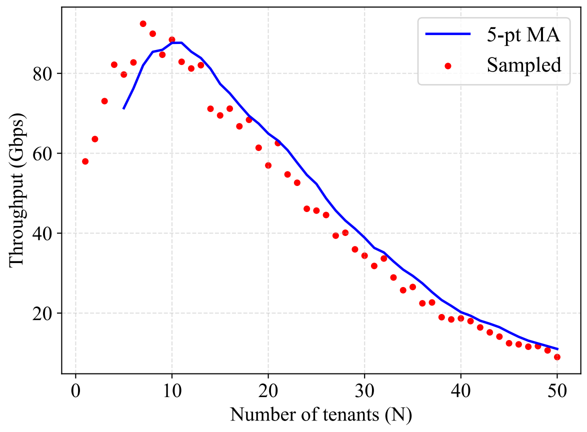

Fig. 4. Throughput comparison among normal and malicious tenants.

D. Experimental

To evaluate the performance and fairness of the proposed FPGA-SDM resource sharing framework, we conduct comprehensive simulations under different tenant behavior and workload conditions.

1) Impact of Malicious Tenants on Throughput: We first simulate the throughput performance for three types of tenants: normal tenants with stable requests, malicious tenants generating excessive requests, and tenants managed under our proposed solution. As shown in Fig. 4, under increasing request sizes, malicious tenants significantly degrade system throughput compared to normal tenants. However, our framework effectively mitigates this degradation, maintaining near-optimal throughput levels even under adversarial behaviors.

2) Throughput Trends with Tenant Scaling: Fig. 5 illustrates the system throughput as the number of tenants increases from 1 to 50. Due to natural resource contention and coordination overhead, throughput does not scale linearly with tenant count. We introduce stochastic variations to mimic real-world system jitter. A general increasing trend is observed up to a saturation point (around 30 tenants), beyond which throughput gains diminish.

3) Effect of Packet Sizes and Tenant Numbers: To further explore performance under variable packet sizes, we simulate throughput across 64B to 1500B packets combined with

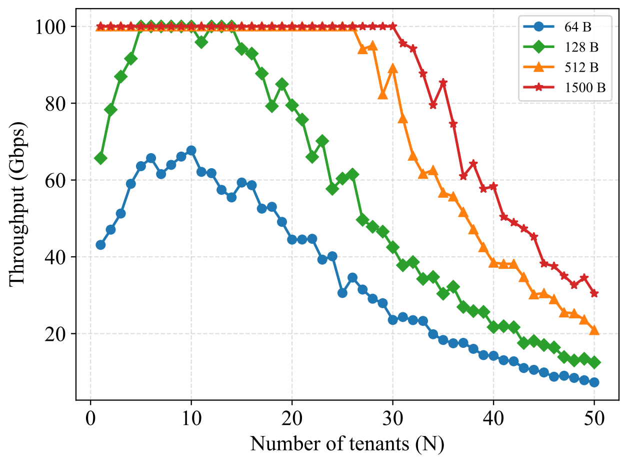

Fig. 5. Throughput variation with increasing tenant numbers, including random system jitter.

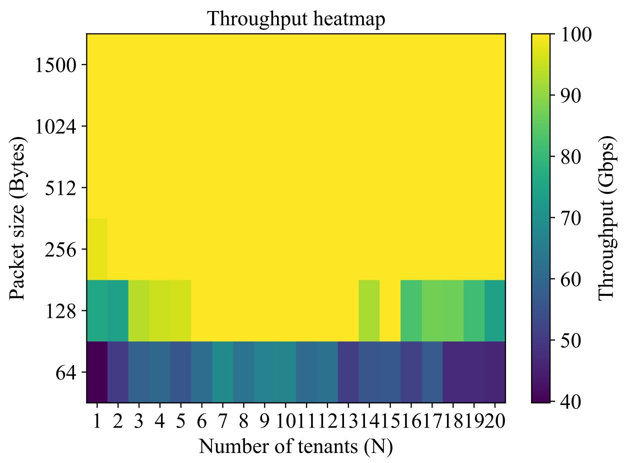

Fig. 6. Heatmap of throughput versus tenant numbers and packet sizes.

different tenant numbers. As shown in Fig. 6, larger packet sizes lead to higher achievable throughput due to improved efficiency in network transmission. However, when tenant numbers exceed 20, even large packet scenarios experience resource saturation effects.

4) Summary: These simulations demonstrate that our proposed dynamic sharing mechanism can maintain high throughput and robustness against malicious tenants, effectively balancing resource efficiency and fairness even under multi-tenant conditions.

V. Conclusion

We have introduced FPGA-SDM, an FPGA Network Function as a Service architecture that unlocks elastic sharing of SmartNIC resources across multi-tenant workloads. FPGA-SDM achieves high utilization through space division multiplexing and ensures strict performance fairness even in the presence of bursty or malicious tenants. Experiments show up the throughput can maintain a normal level under the occupation of malicious users, and can maintain a certain performance gain in multi tenant scenarios. Looking forward, we plan to (i) extend the service pool with stateful transport accelerators (e.g., QUIC offload) and (ii) integrate FPGA-SDM into a Kubernetes-compatible orchestration layer so that cloud operators can deploy FPGA micro-services using the same abstractions as CPU/GPU functions. We believe these steps will further cement SmartNICs in heterogeneous cloud infrastructures.

References

[1] M. H. Quraishi, E. B. Tavakoli, and F. Ren, "A Survey of System Architectures and Techniques for FPGA Virtualization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2216–2230, 2021.

[2] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, "SoK: Secure FPGA Multi-tenancy in the Cloud: Challenges and Opportunities," in *EuroS&P'21*. IEEE, 2021, pp. 487–506.

[3] J. Mbongue, F. Hategekimana, D. Tchuinkou Kwadjo, D. Andrews, and C. Bobda, "FPGAVirt: A Novel Virtualization Framework for FPGAs in the Cloud," in *CLOUD'18*. ACM, 2018, pp. 862–865.

[4] J. Chaudhuri and K. Chakrabarty, "Diagnosis of malicious bitstreams in cloud computing FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 3651–3664, 2023.

[5] W. Lin, Y. Shan, R. Kosta, A. Krishnamurthy, and Y. Zhang, "SuperNIC: An FPGA-based, cloud-oriented SmartNIC," in *FPGA '24*. ACM, 2024, pp. 130–141.

[6] M. Ewais, J. C. Vega, A. Leon-Garcia, and P. Chow, "A Framework Integrating FPGAs in VNF Networks," in *NoF'21*. IEEE, 2021, pp. 1–9.

[7] J. M. Mbongue, D. T. Kwadjo, A. Shuping, and C. Bobda, "Deploying multi-tenant FPGAs within Linux-based cloud infrastructure," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 2, pp. 1–31, 2021.

[8] M. Mandava, P. Reckamp, and D. Chen, "Nimblock: Scheduling for fine-grained FPGA sharing through virtualization," in *ISCA'23*. ACM, 2023, pp. 1–13.

[9] D. Park, Y. Xiao, and A. DeHon, "Fast and Flexible FPGA Development using Hierarchical Partial Reconfiguration," in *FPT'22*, 2022, pp. 1–10.

[10] Y. Xiao, A. Hota, D. Park, and A. DeHon, "HiPR: High-level Partial Reconfiguration for Fast Incremental FPGA Compilation," in *FPL'22*. IEEE, 2022, pp. 70–78.

[11] B. Seyoum, D. Giri, K.-L. Chiu, B. Natter, and L. Carloni, "PR-ESP: An Open-Source Platform for Design and Programming of Partially Reconfigurable SoCs," in *DATE'23*. ACM, 2023, pp. 1–6.

[12] Y. Zhou, M. Wilkening, J. Mickens, and M. Yu, "SmartNIC security isolation in the cloud with S-NIC," in *EuroSys'24*. ACM, 2024, pp. 851–869.

[13] X. Kong, J. Chen, W. Bai, Y. Xu, M. Elhaddad, S. Raindel, J. Padhye, A. R. Lebeck, and D. Zhuo, "Understanding RDMA microarchitecture resources for performance isolation," in *NSDI'23*. USENIX, 2023, pp. 31–48.

[14] H. Li, Y. Dang, G. Sun, G. Liu, D. Shan, and P. Zhang, "LemonNFV: Consolidating Heterogeneous Network Functions at Line Speed," in *NSDI'23*. USENIX, 2023, pp. 1451–1468.

[15] J. Li, S. Xue, W. Zhang, R. Ma, Z. Qi, and H. Guan, "When I/O interrupt becomes system bottleneck: Efficiency and scalability enhancement for SR-IOV network virtualization," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1183–1196, 2017.

[16] S. Cirici, M. Paolino, and D. Raho, "SVFF: An automated framework for SR-IOV virtual function management in FPGA accelerated virtualized environments," in *CITS'23*. IEEE, 2023, pp. 1–6.

[17] S. N. Nag, "Technical analysis of pcie to pcie 6: A next-generation interface evolution," *World Journal of Engineering and Technology*, vol. 11, no. 3, pp. 504–525, 2023.

[18] E. Cui, T. Li, and Q. Wei, "Risc-v instruction set architecture extensions: A survey," *IEEE Access*, vol. 11, pp. 24 696–24 711, 2023.

[19] Y. Chen and M. S. Abdelfattah, "Bramac: Compute-in-bram architectures for multiply-accumulate on fpgas," in *FCCM'23*. IEEE, 2023, pp. 52–62.