User Plane Scheduling Algorithms in 5G SOCs

Ahmed Salman

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 13.3.2021

Supervisor

Prof. Jussi Ryynanen

Advisor

MSc Anssi Saari

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Ahmed Salman

Title User Plane Scheduling Algorithms in 5G SOCs

Degree programme Electronics and electrical engineering

Major Micro- and Nanoelectronic Circuit Design **Code of major** ELEC3036

Supervisor Prof. Jussi Ryynanen

Advisor MSc Anssi Saari

Date 13.3.2021 **Number of pages** 50+1 **Language** English

Abstract

While 5G mobile network has received much attention due to the need for faster, more reliable, and more secure mobile communication. 5G uses cases have infiltrated every aspect of both daily and working life. Consequently, Multiple technologies such as Network Function Virtualization (NFV) and Software Defined Networks (SDN) have been introduced to cope with the increasing user requirements.

As a result of NFV and SDN, The networking elements moved from the traditional network element such as routers and switches; to virtualized network elements implemented on computing platforms such as ARM or x86 CPUs. However, the rate of the network packet processing increase is much faster than the rate of performance increase in CPUs, especially with Moore's lawes reaching its end.

Hardware acceleration of network functions especially packet-scheduling, has recently become the main solution for the gap between the performance that CPUs offers and the required processing speed for network packets. On the other hand, software frameworks such as Open Data Plane (ODP) simplify the task of getting the network application running on these different platforms. As ODP offers a unified programming interface for high performance data plane network applications.

This thesis aims to evaluate packet-scheduler feasibility as part of the future Nokia chipset to fulfill 5G requirements by implementing a prototype of the OpenDataPlan (ODP) scheduler. The chipset needs to achieve high performance while maintaining a small size, low power, and low price. Then, this thesis will compare various s trade-offs, including the number of queues, priorities, scheduling algorithms, and memory type, either internal or external.

Keywords Scheduler, System on chip, 5G , SoC

Preface

I want to thank my mother and father,and I want to thank my team at Nokia Petri,Sarri and Mona and my supervisor Ryynänen Jussi for their support.

Otaniemi, 13.5.2021

Ahmed Salman

Contents

Definitions & Abbreviations

Abbreviations

3GPP	3rd generation partnership project
5G	Fifth Generation Mobile System
API	Application programming interface
ASIC	Application specific integrated circuit
CPU	Central processing unit
DMA	Direct memory access
EB	Exabyte
FPGA	Field Programmable Gate Array
PDU	Protocol Data Unit
RAN	Radio Access Network
SoC	System on Chip
TCP	Transmission Control Protocol
UDN	Ultra-Dense Networks
UDP	User Datagram Protocol
UPF	User Plane Function
VM	Virtual Machine

1 Introduction

1.1 Background

During the past decade, the 5G mobile network has received much attention due to the need for faster, more reliable, and more secure mobile communication. Furthermore, applications, such as remote surgeries, massive IoT, 3D videos, and virtual reality, have continued to challenge the capabilities of the current networks. Moreover, global mobile data traffic will reach approximately 226 EB per month by 2026 achieving more than 4 folds growth [1]; in addition, as the price of the cheapest phone supporting the 5G sub 6 GHz band has fallen below the 300 US dollars mark, the number of users has proliferated and thus the demand on 5G networks. By 2026, approximately 54 percent of the word mobile data traffic will be 5G traffic [2]. 5G Mobile networks offer multiple capabilities to serve these future needs: [4]. The first notable capability is enhanced mobile broadband which provides high-speed internet connections up to 10 Gbs. The second concerns ultra-reliable and low latency communication; that provide the user with more certainty regarding different aspects of the network, such as throughput. Massive machine-type communication is another noteworthy feature; this is an essential requirement for the emerging IoT devices that need to be small, cheap, and use minimal power.

5G mobile networks have implemented multiple enablers to achieve these goals. One of these concerns the virtualization of 5G RAN. Network virtualization consists of two parts: Network Function Virtualization (NFV) and Software Defined Networks (SDN). Separating the software and hardware parts of the network and splitting the control and user plane increases the flexibility of the network as changing or adding new functions only updates the software. Additionally, the SDN allows wireless infrastructure-sharing among multiple operators and capacity-sharing between different network parts. Network virtualization moves from the traditional core network (e.g., switches and routers) towards the RAN [5].

Although network virtualization has many benefits, many technical challenges have arisen from this evolution, especially scheduling among the processing cores that run the software and other hardware parts. Resources must be carefully allocated to achieve the same performance as traditional networks. The demand difference between latency, bandwidth, and reliability in content-aware scheduling is considerable. Consequently, scheduling the packets between nodes with quality of service (QoS) requirements is critical with high demanding applications. As single processors processing power doesn't satisfy the networking requirements, parallel processing and hardware acceleration are becoming increasingly attractive due to the parallel nature of the network traffic [6].

1.2 Motivation

This thesis was conducted for Nokia Solutions and Networks Oy from August 2020 to February 2021. Nokia develops 5G HW and SW aiming to deliver ultra-low latency, massive connectivity, extreme capacity in 5G hardware. To meet these requirements, 5G networks need to be highly versatile and change their architecture to support a vast range of services. Nokia develops commercial end-to-end 5G solutions that enable operators to capitalize early on 5G, such as Nokia's AirScale Radio Access, which delivers vast capacity and connectivity required as the network moves towards 5G IoT [7] . Part of Nokia's AirScale Radio Access is the Nokia ReefShark chipset for 5G Baseband which delivers up to 84 Gbps per system module. With multiple modules chained, it can reach throughputs of up to 6 terabits per second [8].

Different hardware IPs in the baseband modules have been developed resulting in substantial improvements and hardware acceleration to support even more bandwidth and better quality of service [8], while simultaneously reducing the size, cost, and power consumption yet still ensuring that they meet 5G requirements. Due to the high cost of deploying the 5G networks, telecommunication companies need to ensure that the radio network can meet user requirements for more than a decade, which requires anticipation of future demands, especially for the hardware modules that depend directly on the application, such as the packet schedule.

The 5G network supports multiple different applications with a different quality of service requirements, from IoT to real-time media service for thousands of clients to more critical applications such as remote surgeries. The network needs to process incoming packets as fast as possible without affecting any critical QoS, including packet loss rate, deadline, and delay variance therefore, hardware acceleration for packet scheduling and processing occupies a central role in the quality of service provided for the user [6].

1.3 Goals and contributions

This thesis aims to evaluate packet-scheduler feasibility as part of the future Nokia chipset to fulfill 5G requirements by implementing a prototype of the OpenDataPlan (ODP) scheduler. The chipset needs to achieve high performance while maintaining a small size, low power, and low price. Then, this thesis will compare various s trade-offs, including the number of queues, priorities, scheduling algorithms, and memory type, either internal or external.

1.4 Implementation

The hardware architectural design was implemented for ODP packet scheduler using Very High Speed Integrated Circuits Hardware Description Language (VHDL). VHDL is a language for describing digital electronic systems which is standardized by IEEE as IEEE Standard 1076 with the first standard version being VHDL-87 [9]. However,

the VHDL-93 revised version is used in our implementation. Hardware Description Languages (HDL) ease the hardware designer's work by providing another level of abstraction instead of transistor-level design, higher abstraction levels such as gate-level, register-transfer-level (RTL), and processor-memory-switch- level. This helps save money and time to market, especially with the current designs that contain billions of transistors. HDLs are used to describe either the structure of the digital circuit uses transistors, gates, and RTL blocks, or it describes the digital circuit using a behavioral model, which is then translated into a netlist by the synthesis tools.
In Application-Specific Integrated Circuit (ASIC) Design, the netlist is mapped to gates standard-cells in the manufacturer's technology file. In the case of Field-programmable gate array (FPGA), the netlist is used for FPGAs to generate the programming file configuring the interconnection and logic of the FPGA fabric. FPGA implementation is used to evaluate the packet scheduler implementation feasibility and compare different parameters for priority levels, the number of groups, and memory size.

1.5 Structure

The thesis is structured as follows: Chapter 2 introduces the overall architecture and use cases of the 5G networks, as well as the improvements offered by the 5G networks to tackle each use case, then describes network virtualization in 5G. Chapter 3 describes the scheduler design, Classic switches and router, then lists multiple scheduling algorithms and reviews OpenDataPlane-based scheduler implementation used in many state-of-the-art 5G L1 SoCs. Afterwards, Chapter 4 discusses the scheduler implementation methods by comparing the software versus hardware-accelerated implementation. Chapter 5 evaluates the feasibility of the scheduler implementation and presents the scheduler prototype implementation. Finally, Chapter 6 presents the conclusion for the thesis and explores possible future improvements.

2 5G Mobile Systems

5G is the fifth generation of mobile networks specified in the 3GPP standard. 5G specifications are set in continuous releases gradually improving different aspects of the mobile network with multiple releases over a decade starting from Release 14 in 2016. Figure 2 shows the evolution time plane for the 5G network implementation starting from providing higher bandwidth speed in 2017 to full 5G eco-system after 2022 including massive IoT, low latency and high reliability network[10]. With the global traffic expected to reach 160 exabytes per month by 2025 [11], 5G is designed to handle this increase, and it was the main focus of its first releases. Currently,5G can be commercially deployed at both sub-6 GHz and mmWave frequency bands.

In this chapter, we will discuss the uses cases for the 5G. Then we will discuss the multiple technology enablers that are introduced in different releases to achieve the required performance for these use cases.

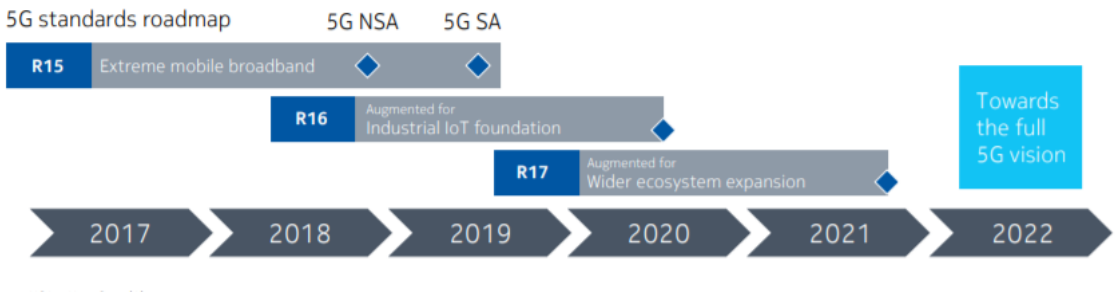

Figure 1: NR evolution time plan[12].

2.1 5G Use Cases

5G mobile communications provide the means to open a new unprecedented application that affects every aspect of daily live,such as video streaming services with Ultra High Definition (UHD) and three-dimensional (3D) videos as well as Augmented and Virtual reality, which require fast internet speed and online meeting, besides online gaming which requires ultra-low latency. A. In addition, it concerns every aspect of working life with the need for remote computing, mobile cloud/desktop cloud, and broadcast services. In addition to participation in more critical applications, such as remote surgeries; 3D connectivity which includes aircraft and drones; and collaborative robots.

IoT applications have added more value to the mobile communication network concerning multiple applications in almost every industry, with tens of thousands of connections required between tens of thousands of small smart devices with low latency and small power. 5G mobile communication supplies the necessary capabilities to achieve these extreme requirements.

When mobile networks were started, the main mobile network functionality was to connect cellular calls. The wireless data connection gradually became a primary part of the mobile networks. IoT has presently expanded the scope to not only include interconnections between IoT devices, but also intrinsic support for IoT-based applications in the network. This support in turn provides the capability required for multiple applications that affect daily living standards as these applications infiltrate every corner of our lives. These applications include mobile health, Internet of Vehicles (IoV), environmental monitoring, smart homes, and industrial control Industrial Automation. IoT applications are spread among different fields, such as smart grid, environmental monitoring, smart agriculture, smart city infrastructure, traffic Management, sensor networks, and remote surgery. Another central application is Automotive driving that will use 5G to produce a much better autonomous driving experience; one other key application is industrial control with machine-to-machine communication.

5G specification consists of multiple requirements to serve the wide applications that 5G intends to support. First, Enhanced Mobile Broadband is the main focus of the average user [3] and the primary upgrade offered by 5G and the first to be released. 5G offers an improved performance and an increasingly seamless user experience. Secondly, Ultra-Reliable and Low Latency Communication focuses on providing throughput, latency, and availability required by critical applications. Such as factory automation, smart grids, traffic managements, remote surgeries etc. Thirdly, Massive Machine-Type communication supports a vast number of low-cost, extended battery life and transmits a relatively low volume of data with flexible delay constraints. Other specifications have been defined and such as vehicular communication, wireless-wireline convergence, network slicing, network automation, enhanced MIMO, time-sensitive communication, cloud gaming, NR-Unlicensed (NR-U), reduced-capability NR devices and private networks. [4].

2.2 5G Architecture

As shown in Figure 3, 5G mobile networks connect devices at the area of coverage. These devices include traditional devices such as phones and tablets; and other devices that have been newly targeted by mobile networks in 5G, such as cars and wireline-wireless coverage. The mobile network consists of the Radio Access Network (RAN) and the Mobile Core. The RAN is the final link between the network and the phone. It includes the antennas on the towers plus multiple base stations. The antenna receives the signal from devices then, is sent to the base station, after which the base station sends it to the mobile core network.

Each part of the network has a distinctive rule. The RAN manages the radio spectrum, ensuring it is efficient and meets every user's quality-of-service requirements. While The core provides access controls ensuring correct authentication for the users and correct routing for calls and data by providing Internet IP connectivity;

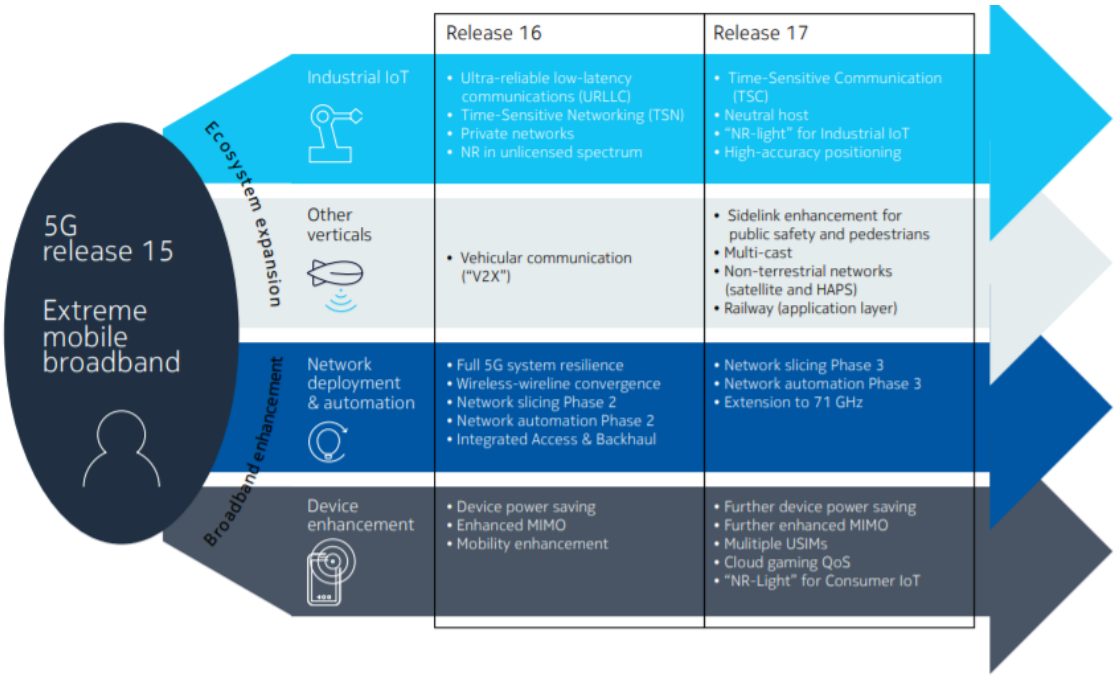

Figure 2: NR evolution time plan[12].

it also enables mobile network operators to charge for phone calls and data usage, as well as connects users to the Internet. It also responsible for handovers as users move from coverage provided by one RAN tower to the next securing uninterrupted service [15, 14].

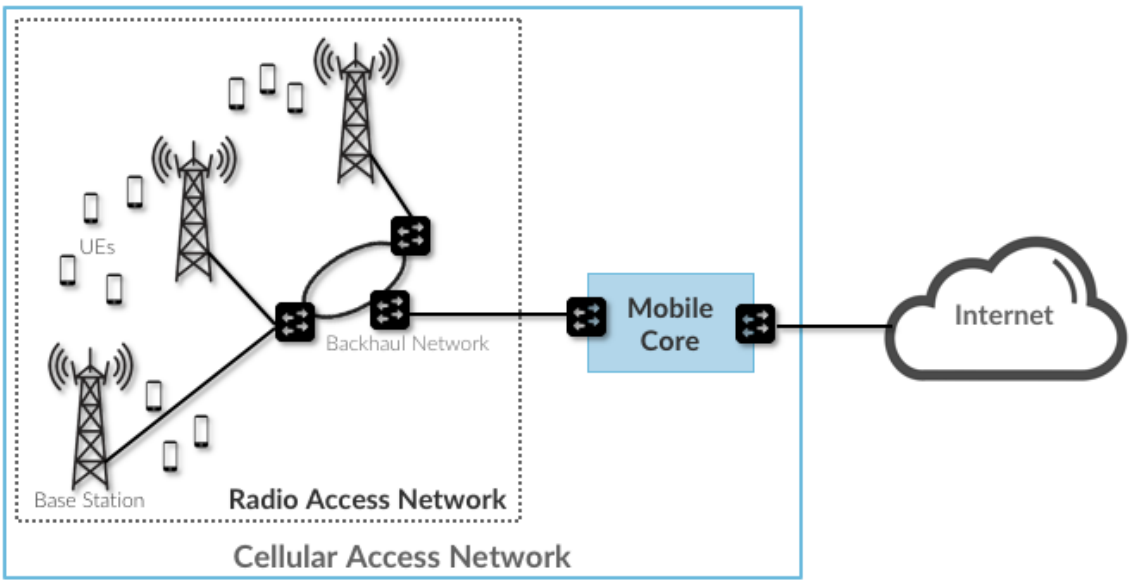

Figure 3: Cellular networks consists of a Radio Access Network (RAN) and a Mobile Core.[14].

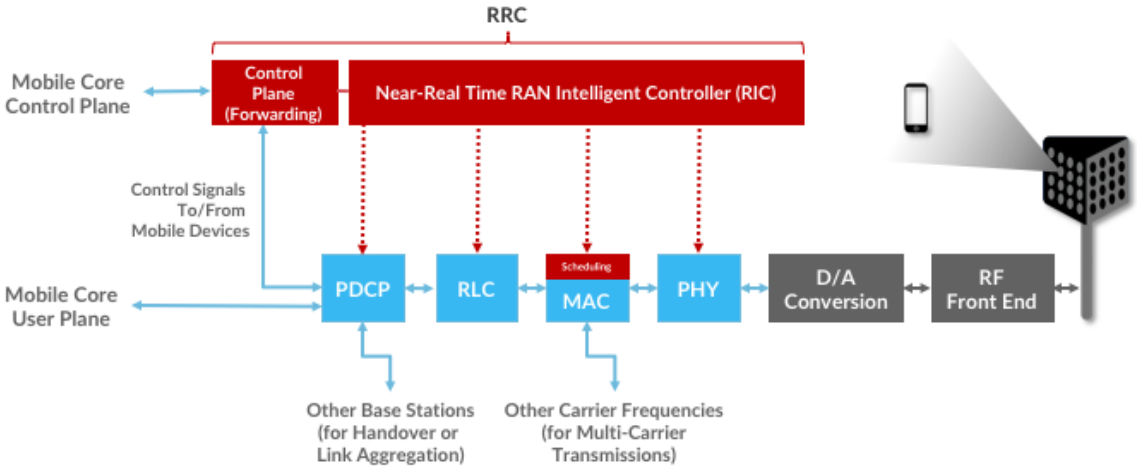

Figure 4: Software Defined Network and Control & User Plane Separation [16].

The Radio Access Network is responsible for receiving the wireless signal and connect the device to the core network. 5G has stringent latency requirements with a minimum of 4ms and going as low as 0.5ms in 3GPP target specifications for ultra-low latency applications.The 5G user plane is required to independently scale without any change to the control plane to efficiently handle the increased data traffic. The separation between the user plane and control plane offers a flatter, centralized, and more user-centered network which leads to ultra-dense networks working efficiently even in the most difficult deployment scenarios.

The user plane is the part of the network that receives and sends packets from an interface and does the required processing, delivering, dropping, or forwarding of these packets [17]. The 5G user plane interface is the link between the 5G-RAN node and the UPF to transmit different network packets such as UDP and TCP between the NG-RAN node and the UPF [18]. Then, the UPF is responsible for Forwarding traffic between RAN and the Internet. Also, it manages policy enforcement, traffic usage monitoring & reporting, , lawful intercept, and QoS policing [4].

The control plane is responsible for user connection management, defining QoS policies,and performing user authentication. The control plane is also responsible for packet routing, using control packets which describe the network topology to correctly route the actual data packets in the user plane. The control packets are handled in the control plane as they are different from the data packets. Although the control packet consists of a small percentage of the traffic, it uses complex signaling protocols, leading to high latency and low performance if they are handled together with the data packets. After parsing the control packets, the control plane can update the data plane information and inject signaling packets into the data plane [19].

2.3 Software-defined Networking (SDN)

Software-Defined Networks [20] is an emerging architecture closing the gap between IT operation and network operation. SDN is used to design, build and manage networks to achieve dynamic, manageable, cost-effective, autonomous, operationally efficient, and adaptable networks. It consists of three different items: SDN separates the network control from the network data plane, such as switches and routers. Secondly, the control plane can be programmed separately, which offers greater flexibility in reconfiguring the network to match user requirements. SDN also safe time by reconfiguring in the software level instead of the hardware level.

Network reachability is one of the main challenges in mobile networks. Each new node or service used to require updating the IP allocation, bandwidth allocation, policy openings and routing changes in every router and every switch in traditional networks. In traditional networks, traffic moves through multiple switches and routers to reach its destination. Each router or switch decides the place to which to send the traffic depending on its routing table, this results in the rerouting task or adding of a new node lasting several hours or days. As shown in Figure 5,SDN centralizes the control plane, which allows the application and network services to directly program the network. turning the task of adding nodes or services into one that only demands seconds. Furthermore, having a centralized control plane increases the network's efficiency and reliability as the router in the traditional network only makes decisions based on its routing tables as well as the router availability and switches adjacent to it. In the SDN control plane, the decision depends on the end-to-end topology of the networks achieving the best efficiency and reliability.

Software control of the network renders the network more agile and flexible as it allows it to meet the dynamic requirements with software routing algorithms. The programmable networks allow the operators to support applications, such as dynamic provisioning of bandwidth, automatic scale-out, automatic scale-in, building protection paths, and adding a new service and nodes to live networks. It also ensures the QoS at any user requirement level. Consequently, SDN is used for reconfiguration and real-time management of the complex network.OpenFlow protocol is one of the main parts of SDN followed by most vendors as it provides a standardized interface between the control plane and the data forwarding plane. SDN uses a net- work controller through OpenFlow to define the behavior of the networking infrastructure and operation, which allows remote configuring of the packet forwarding tables.

2.4 Network Virtualization

The 5G System architecture consists of multiple network functions (NF) such as Authentication Server Function (AUSF), Access and Mobility Management Function (AMF) - Data Network (DN), Network Repository Function (NRF). These functions and their interfaces are defined in the 3GPP specifications. A network function can

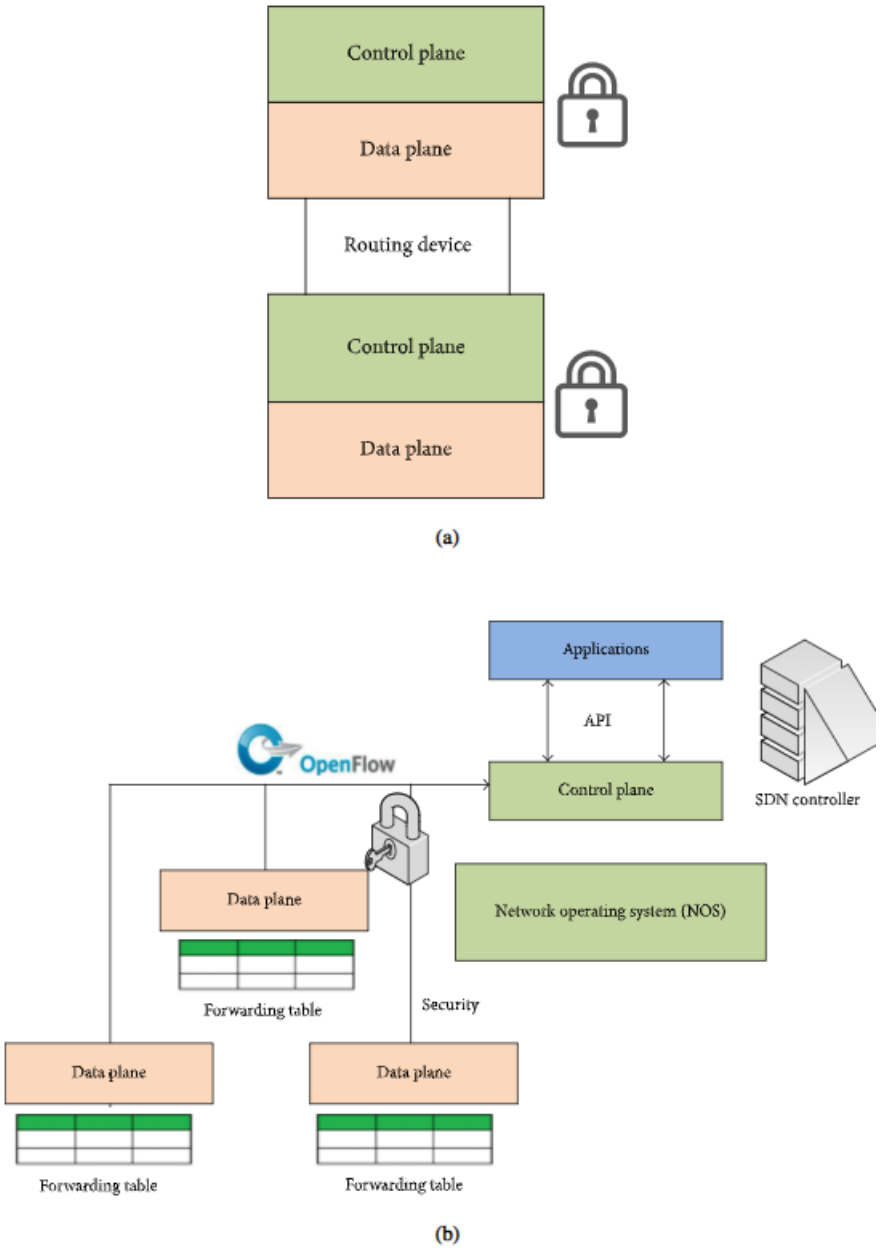

Figure 5: Traditional hardware-based network versus. Software Defined Networks [12].

be implemented as a software instance running on dedicated hardware, which requires proprietary hardware equipment to implement these network functions. Traditional mobile core networks have suffered from slow time to market and costly over-head when cellular operators have tried to replace network functions or add new services. It requires replacing the existing hardware and buying an expensive replacement even if it is still working efficiently for most use cases, in addition to requiring skilled professionals to integrate. The decision to scale the network or add a new service is not an easily made one for the operators. Therefore, all these factors slow the development cycle.

Instead, NFV allows the network function to be virtualized and instantiated on multiple platforms, such as the cloud computing services. Although SDN and NFV are usually confused with one another, both SDN and NFV can be separately implemented. Network functions can be deployed and virtualized without the need for a software-defined network, although using both SDN and NFV effectively decouples network functions. Routing decisions in routers and switches are transformed in a centralized network function at remote network servers or in the cloud through a standardized interface, such as OpenFlow as shown in Figure 5. The result is a highly flexible, fast, adaptive reconfiguration of the network. Furthermore, cloud abstraction provides guaranteed content delivery and cost minimization for the operator. It also reduces power consumption through equipment consolidation. It shortens the operator cycle of innovation thus reducing processing time [21], rendering the network more intelligent, resilient, and scalable. It also provides the carriers with distribution flexibility as they add or remove hardware resources as needed, eliminating performance bottlenecks and idle resources.

The virtualization of the network resources on the cloud ensures service reliability and stability. It eliminates local resource failure threats with backup in multiple sites and reduces the total cost of ownership (TCO). Network operators can explore new markets for their business and use various new services offered by the 5G networks. The carriers can provide extra services, such as online gaming, 3D videos, augmented reality, Massive IoT, and more at request, which is due to SDN and NFV [20]. This provides an additional source of earning for the carrier as there is a limit to what the carriers can accumulate on the regular data and call subscription, rendering it financially unviable as 5G deployment is expensive and requires ultra-dense network nodes to provide an ideal user experience. 5G fuels the launch of innovative services in different fields to generate new revenue. Telecommunications, energy & utilities, agriculture, automotive, banking, media & entertainment, and consumer electronics are expected to participate more in the revenue and have a much more significant share than regular data and calls subscriptions. 5G market mainly depends on IT and telecom. However, by 2020 5G services contributed with more than 24.5% of the market. other services are expected to surpass it during the next decade, as shown in Figure 6.

Network function virtualization also improves network service provisioning flexibility and reduces the time to market new services, which is a significant advantage

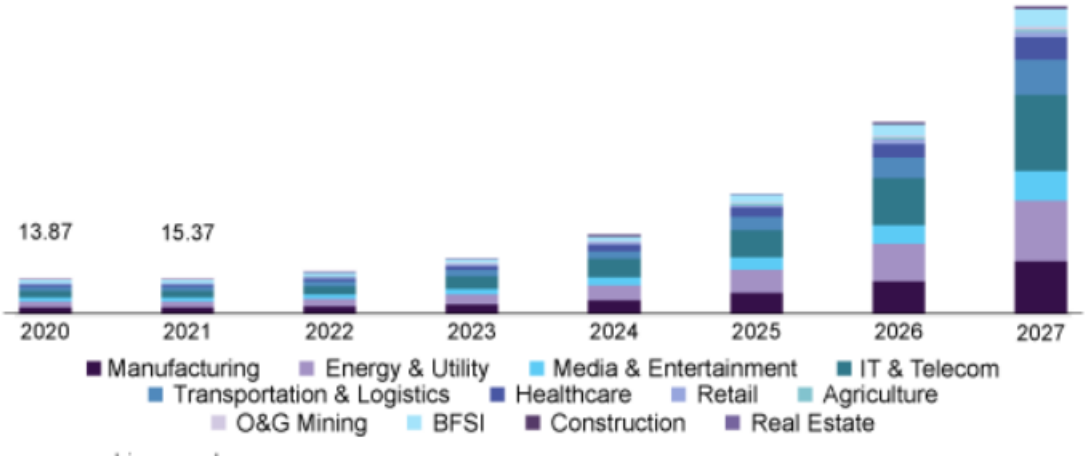

Figure 6: Vertical 5G Revenue Growth Opportunities [22].

to operators. General-purpose servers, such as Amazon's EC2, and off-the-shelf hardware, such as storage and switch, can be used as the building blocks of the network functions. This became possible as NFV separates the software implementation of the network function from the underlying hardware platform [21]. The use of separate software instances instead of hardware opens new ways to architect the network to provide faster networking service provisioning using network function virtualization. Network functions are implemented on servers through software virtualization techniques and supported by technologies such as Docker and Kubernetes. Carriers can acquire these new servers from providers on-demand, Less money and time are spent as no installation of new equipment is required. These servers can be integrated with other network parts quickly. Network operators may implement a software-based virtual firewall, vEPC (Virtual Evolved Packet Core) function, or any other network function. The network functions can be implemented using virtual machines (VM) on an x86 or ARM platform with a general-purpose processor-based platform. NFV is a step to lower-cost agile network infrastructure that brings many benefits to users and carriers [21]. NFV has a massive effect on the telecommunication industry as it reduces capital investment and energy consumption by eliminating the need to buy networking devices and decrease the time to market a new service shortening the average innovation cycle. The development becomes based on software-based service deployment instead of dedicated hardware. Hence, Operators can provide personalized and targeted services to satisfy the customers' needs.
.[22].

2.5 Quality of Service

Communication networks has multiple types of traffic. Data traffic and mails are not delay-sensitive. But voice and video are real-time, so it requires a specific quality of service. Also, applications such as IoT, mobile broadband internet, autonomous

driving, remote surgeries, and machine to machine, some services like broadcasts require real-time and demand minimum delay. The best-effort delivery model can cause random delays or drop packets, or do not arrive in order. This results in bad video quality, video and audio out-of-sync, pausing, or choppy videos. Also, audio can be distorted, or voice breakups [23].

First, there are three main problems. The first problem concerns the lack of bandwidth when the bandwidth is surpassed, packets are dropped, and the solution is to increase the bandwidth, which isn't always possible; therefore, QoS policing and queueing is discussed later. The second one is the latency and jitters, with a latency of 4ms target of 5G, primary sources of latency are propagation delay, processing delay, and queueing delay. The final and third is that packet loss happens when the input buffer size is exceeded. This can be fixed by introducing queueing, but this can cause jitters, also implementing traffic policing by dropping low-priority packets or traffic shaping by delaying queues [24][25][26].Implementing QoS is essential to increase spectral efficiency and reduce the latency. QoS implementation is split into three steps. First, identify the traffic and its requirements. Secondly, it defines the classes with similar priority, finally determining the QoS for each class. Classes can be defined as such with voice being the highest priority, then video, then after that is data services applications, followed by best effort application, such as emails, then less critical applications, such as torrent.

QoS has three modes. The first is the best effort, which means QoS is disabled, and the best-effort model is used instead. The best-effort model is similar to the traditional internet, in which the network cannot classify the type of data transmitted. Secondly, the guaranteed flow bit rate (GBR QoS Flows) or integrated services for which the application signals to the network that it requires special treatment. The third one uses differentiated services with a non-guaranteed bit rate (non-GBR QoS Flows). The network identifies the classes with similar QoS requirements and applies the related QoS level, rendering it very scalable. However, it does not guarantee a specific treatment for the packets [18]. Classification of the classes can be in either layers 2, 3, or 4 of the protocol, or in the parameters, such as COS (class-of-sevice bits) or Source and destination IP addresses or protocols, including TCP or UDP.

Congestion can occur anywhere in the network in which there is a speed mismatch between input and output. The overflow of arriving traffic can be handled by queuing algorithms and then choosing from the queue depending on the priority. Congestion management inserts packets in queues and then schedules them for transmission when the output buffer is ready to transmit them. Queue temporarily stores the packets until either they can be scheduled to transmission or queue drop packet when the queue is full. Packets in software-defined networks are mapped to queues depending on multiple factors. First, the protocol they belong to, such as UDP and TCP. Secondly, the priority requirements, and finally the ordering requirements.

Queueing algorithms include FIFO (First In First Out), priority queueing, round-

robin, and weighted round robin. FIFO is the most straightforward queuing algorithm as it just adds data to the FIFO, all software queues are mapped to the same hardware queue. Secondly, priority queuing using multiple queues to allow prioritizations of different events. However, the first queue dispatches, and others may starve. Thirdly, the round-robin uses multiple queues but offers no prioritization as it dispatches one packet from a different queue each time. After that, there is a weighted round-robin in which the packets are accessed in the round-robin, but weight is added to each queue to ensure that multiple packets can be dispatched from the same queue at the same round.

Finally, weighted fair queuing (WFQ) is flow-based queuing. First, WFQ schedules the high priority queues to the front of the queue and prevents high-volume flows from taking over the bandwidth, then flows can be transmitted by order, and higher importance flow gets higher priorities. It cannot provide fixed guarantees for traffic flow. Consequently, custom queues are introduced for high priority queues such as voice traffic, introducing classes that define these classes priority. This offers fine granularity scheduling based on the QoS Flow.

When congestion happens, packets are stored in the queues, but when the queues are full, the packets start to be dropped, which may cause noticeable service degradation. Queues must respect the synchronization policy, such as the TCP synchronization. Also, long queues can cause an overly long latency, and aggressive flows can cause other flows to starve. Consequently, the quality of service has to put some thresholds in order to slow the very aggressive flow, then when reaching the max threshold, it stops this flow. SDN and NFV allow a nearly instant response for configuring the QoS requirement in the network depending on the traffic and network performance. This can help achieve the best performance in forwarding behavior and minimize packet loss rate and packet delay.

3 Packet Scheduler

IT was difficult for packet processing to achieve more than 10 Gb/s using the conventional methods on software-based platforms. Network processing with such rates proved to be a challenge for both CPUs and Operating systems' capabilities. Consequently, ODP hardware-acceleration, DPDk and net map, and other solutions were proposed. This chapter discusses Packet processingg by explaining the features of the packet , and then it reviews classic switches and routers, as well as their operation. Then, different scheduling algorithms are evaluated followed by a description of the OpenDataPlane (ODP) scheduler concept. This chapter concludes with discussing other packet scheduling frameworks currently available in industry.

3.1 Packets Processing

Networks process transport data packets at several parts of the communication networks. This processing was traditionally done using routers and switches. " A packet is A group of bits that includes data plus control information [27]." The protocol data unit (PDU) in the network layer (OSI layer 3) is generally referred to as a data packet. A packet consists of control information and user data(payload). This control information signals to the network element multiple parameters such as the sending host address and destination address, error detection codes, sequencing information, hop limit, packet length, quality of service requirements, and others. This control information is located in the headers and trailers of the packet. The transmitter adds these control data at all the different layers of the protocol stack. The transmitter calculates error detection and correction such as checksum, parity bits, or cyclic redundancy before sending the data. The receiver side recalculates these bits and compares them to the received bits to detect any error during transmission. If inconsistencies are found, the receiver can discard or correct the message depending on the protocol.Afterwards, the network protocol deals with the discarded packets by requesting packets retransmission or by informing the transmitter or resuming operations. Furthermore, some network elements, such as packet filters, may also modify the network packets. Consequently, checksums must be recalculated.

Some parameters, including packet length, are optional and can be ignored depending on the used protocol as it might imply the packet length, in addition to, parameters, such as COS (class-of-sevice), MPLS EXP (Multiprotocol Label Switching Experimental), IPP (Ip precedence, differentiated services code point(DSCP), Explicit Congestion Notification (ECN), and app signature via NBAR. This information in different layers of the protocol prioritizes some types of packets above others. as well as indicates the priority of the packet when congestion happens. In the final part of the packet, the payload (transmitted data) is usually a variable length up to a maximum length. For example, the Maximum Transmission Unit of the Ethernet protocol is 1500 bytes.

3.2 Classic Switches and Routers

Traditional networks consist of hardware devices with a specific function, such as switches, routers, and application delivery controllers. These functions work together and support the network. As these devices are implemented in hardware, they usually sustain their speed, but traditional networks lack flexibility. Most of the router and switch do not expose most of the provisioning APIs; moreover, the software and hardware are proprietary, and licenses are expensive [13]. Although the proprietary provisioning software effectively manages the hardware, it still requires a great deal of time and direct access to the router or the switch. Traditional network elements are mainly implemented as Application Specific Integrated Circuits (ASIC). ASICs arevery fast as they are specially made for a specific target. Still, they are not flexible nor customizable as they are usually produced in thousands or millions of pieces.

3.3 Open Data Plane

OpenDataPlane (ODP) is an open-source project built to provide a unified programming interface for high performance data plane applications in software-defined networks. The main advantages of ODP are that it is simultaneously easy to use and offers a very high-performance. It is also flexible and accelerates innovation as it can interface various networking SoCs of different instruction set architectures.The ODP environment consists of common APIs, configuration files, services, and functions optimized for specific hardware acceleration. The ODP target provides cross-platform compatibility. ODP works across all different networking SoCs with various hardware acceleration. As packet scheduler implementation depends on the vendor ranging from complete software implementation to intensive hardware acceleration of all of its functions and everything in between, which are present in most modern networking Systems on Chip(SoCs)[28]. ODP can achieve that because it provides a level of abstraction ensuring that the application can run efficiently regardless of the way that the ODP functions are implemented [29].

ODP provides a functional model for data plane applications covering all the data-plane application programming operations, such as receiving, manipulating, and transmitting packet data. ODP does not define the performance of its functions; consequently, these functions can be implemented using different ODP implementations produced by various companies. ODP functions are described using abstract data types that are defined depending on the ODP implementation [30]. The main challenge is that the application will use different functions and require different algorithms depending on the hardware-acceleration. However, the goal of ODP is to build applications that run on multiple SoC implementations with a simple recompilation [28].

Data plane application aims to provide extremely high-performance networking, including packet receiving and transmission, in addition to examining and manipulating these packets to achieve optimum network performance. ODP is designed as a

portable framework that primarily depends on the Layer 2 and Layer 3 parameters for packet networking.At the top level, The ODP Application is a program that utilizes at least one ODP APIs. ODP is a framework, not just a programming environment, and applications can use other APIs that could provide similar API capabilities.ODP applications differ regarding their functionalities and operation. However, when all is said in done, some characteristics are shared between all of them. First, they consist of at least one thread, and these threads execute in parallel. Secondly, these threads can communicate and synchronize using synchronization mechanisms, such as atomic variables. Packets are received for the I/O interfaces, and they are examined,then queued until scheduled, afterwards, they are sent to the packet I/O interfaces [29].

The central part of ODP is data packets. Packets are the primary data type that data plane applications manipulate, and they are transmitted and received by PktIO. Packets are defined from pools of type ODP_POOL_PACKET. ODP parcels incorporate semantics that permits review and control in complex manners and support metadata in addition to user metadata. User metadata enables applications to add application-dependent bits of side data with every packet for describing the way it should be used. Handles of abstract type odp_packet_t represent packets.

ODP consists of multiple-build units. First, the ODP application main building unit is the thread, as the ODP applications consist of multiple threads that coordinate together in order to complete the required task. These threads can be either a control thread or a worker thread; these threads can also share the memory. The control thread manages the worker threads and controls the resource allocation. On the other hand, worker threads use a run-to-completion model to do the main processing logic of the application. Multiple processing cores are dedicated for worker threads to run onto. However, multiple threads on a single core can be multitasked, which can lead to performance degradation. Threads also contain attributes that specify the task that the thread executes including the thread mask and scheduler group.

Secondly, events represent an action that happens in the network which needs to be scheduled. This can be the arrival of a new packet that requires handling or the execution of requests that have been asynchronously performed, such as interrupts. The event also reflects notification from the timers or notifications for status change in the components adjacent to the scheduler. Events have an event type in the API specifications, which describes the synchronization mode it represents, and can be created or consumed by a thread as well as events processed by it. Threads can forward the events for other components to continue the processing after it finishes. Thirdly, the queue represents the channel which holds the events. Events are added to a queue using enqueue operations or removed from a queue using dequeue operations. There are two main queue types:plain queues and scheduled queues. Plain queues are just a memory space in which the software directly manages to dequeue events. In contrast,the scheduled queued are managed by hardware accelerators, and events are chosen from it depending on the scheduling algorithm. Queues may also have states that describe the associated context, such as configured, allocated, and ready.

ODP uses a shared memory to store events and packets that exceed the size of its internal physical queueThe shared memory represents raw blocks of storage that are sharable between the ODP threads. ODP application uses memory pools and buffers to define packets and events, and these pools are part of the shared memory. Handles of the abstract type odp_shm_t represents a shared memory. In addition to bools, buffers are another memory abstract used as a shared storage of size by the ODP application. Buffers are allocated from the pools inside the shared memory.

The main strength of ODP is that anyone can create their own ODP implementation catered specifically to their needs. Despite ODP being open-source, businesses can make their own implementation open or closed source depending on the application needs. ODP Allows HW and SW innovation as the API works on multiple platforms and SoCs. ODP implementation does not define the way that each function will be implemented in detail. Instead, it declares the abstract data type; however, it is each business' own decision as to the way each abstract type is represented in a specific platform, and the ODP function is realized. As a result, the business can define ODP APIs as just some software instructions that run on the CPU or offload this function to a co-processor. The platform can be changed without the programmer needing to fully understand the internal hardware. The ODP's ability to work independently is mainly valuable for virtual network functions as they can be built to run on a different platform with different vendors, which renders it more financially beneficial [30].

3.4 Data Plane Development Kit

While ODP is an abstraction that is of a sufficiently high level to allow platform abstraction without restriction or overhead on any network SoC implementation, in contrast, the Data Plane Development Kit (DPDK) is a specific implementation of a scheduling framework. It is a complete software implementation without hardware acceleration, which is supported for x86, ARM, and PowerPC architecture. DPDK has some issues in comparison, for example, it supports a single packet size of 2k bytes, which wastes a significant amount of memory and requires chaining of buffers for packets larger than 2k, which in turn wastes CPU cycles. DPDK also does nothave the abstraction level required for hardware acceleration. Moreover, it uses lock-free queues, resulting in different threads blocking each other, which significantly affects the performance [31].

4 Hardware Accelration

Hardware design is one of the core industries today, with Moore's law reaching its end as we are approaching the physical limits of transistor size with 5nm and 7 nm technology, it is becoming harder and much more expensive to go beyond that. Also, increasing the clock speed of hardware devices causes overheating of the electronic devices because the power dissipation is transistor switching. As a result, achieving better performance in many of today's applications in general-purpose computing platforms has become challenging. Consequently, hardware acceleration has gained huge interest recently as the way to achieve the wanted performance. In this chapter, we will discuss technologies that enable adequate hardware acceleration.

4.1 FPGA

Hardware acceleration comes with many challenges, like a long time to market and considerable initial cost. Application-specific integrated circuits (ASIC) and Field Programable gate array(FPGA) are the two main hardware acceleration choices. ASIC is an Integrated circuit developed for one particular use. It offers the best performance compared to FPGA and general-purpose integrated circuits, but they have the longest time to market that might be years. On the other hand, FPGA offers a middle solution between the ASIC and general-purpose solutions. FPGA takes much less time to market than ASIC, though it does not require as much money as ASIC to use an FPGA, the cost for a single piece is much more expensive than ASIC.

FPGA consists of a matrix of reconfigurable pieces of logic and interconnects, and these logic pieces can consist of lookup tables(LUTs) that can be configured to any required logic. Also, much more complex blocks like memory controller, ram, DMA, and high-speed interfaces like SATA, USB, PCIe in addition to a digital signal processing unit. Some FPGA contains a general-purpose CPU such as arm cortex core integrated with them to integrate software part of the application with the hardware parts.

Xilinx developed ZYNQ System on Chip (SoC). ZYNQ includes much more than the FPGA logic to ease the use of FPGAs for embedded systems development. Firstly, a programable ARM processor capable of running Linux OS, Secondly The Advanced Extensible Interface (AXI) interconnections provide a standardized high-speed interface between the ARM cores and the rest of the FPGA and other IP inside the FPGAs.Finally, it provides hardware overlays to ease the development of new hardware acceleration by providing hardware libraries that can be interfaced with software languages such as python. With the advantages of microprocessors and FPGA, ZYNQ builds very productive applications. AXI interface with the Direct Memory Access (DMA) interface connects the ARM core with the FPGA logic to maximize throughput and minimize latency between the two blocks to achieve the needed performance. AXI DMA transfers streams of data up to two one data word

per clock cycle between the DDR memory connected to the ARM core and the FPGA.

4.2 Hardware Design Process

The hardware design process for hardware accelerators starts by defining the top-level specifications and constraints for the required performance. Functional and hardware requirements are then developed to describe in detail the hardware and software components and their connections and interactions with each other. After that, the RTL design of the hardware blocks is implemented and integrated with other IPs. RTL design can be done in multiple ways VHDL, Verilog, or high-level synthesis. While VHDL and Verilog are much more time-consuming than high-level synthesis, VHDL and Verilog offer more control of the underlying logic and achieve much better performance results. VHDL is strongly typed, natural in use, and easy to read the language, but it has some disadvantages. First, it is very verbose compared to Verilog. Secondly, Verilog is much more similar to the C programming language, so it is easier for people to start writing in Verilog. Also, With System Verilog being the most popular way of testing, Verilog is much easier to integrate with test benches.

After writing the RTL, the testing starts to ensure that every block works correctly independently with unit test and then complete test benches, usually UVM or System Verilog after the Integration. After that, there are two routes to be taken. The first is FPGA implementation, so the FPGA synthesis is done, then the place and routing with power and area optimization is done. This is mainly done by the tools and does not require too much time. The other route is ASIC implementation, which starts by testing for the target technology library, then starts down the back-end route, which consists of multiple steps: place and route, timing, power, and after place and route testing.

Strongly Typed	Weakly Typed
Easier to understand	Less code to write
More natural in use	More of a Hardware Modeling Language
verpose	succinct
Not like C language	Similar to C language
Less testing friendly	Testing Friendly

Table 1: Comparing VHDL and Verilog [32]

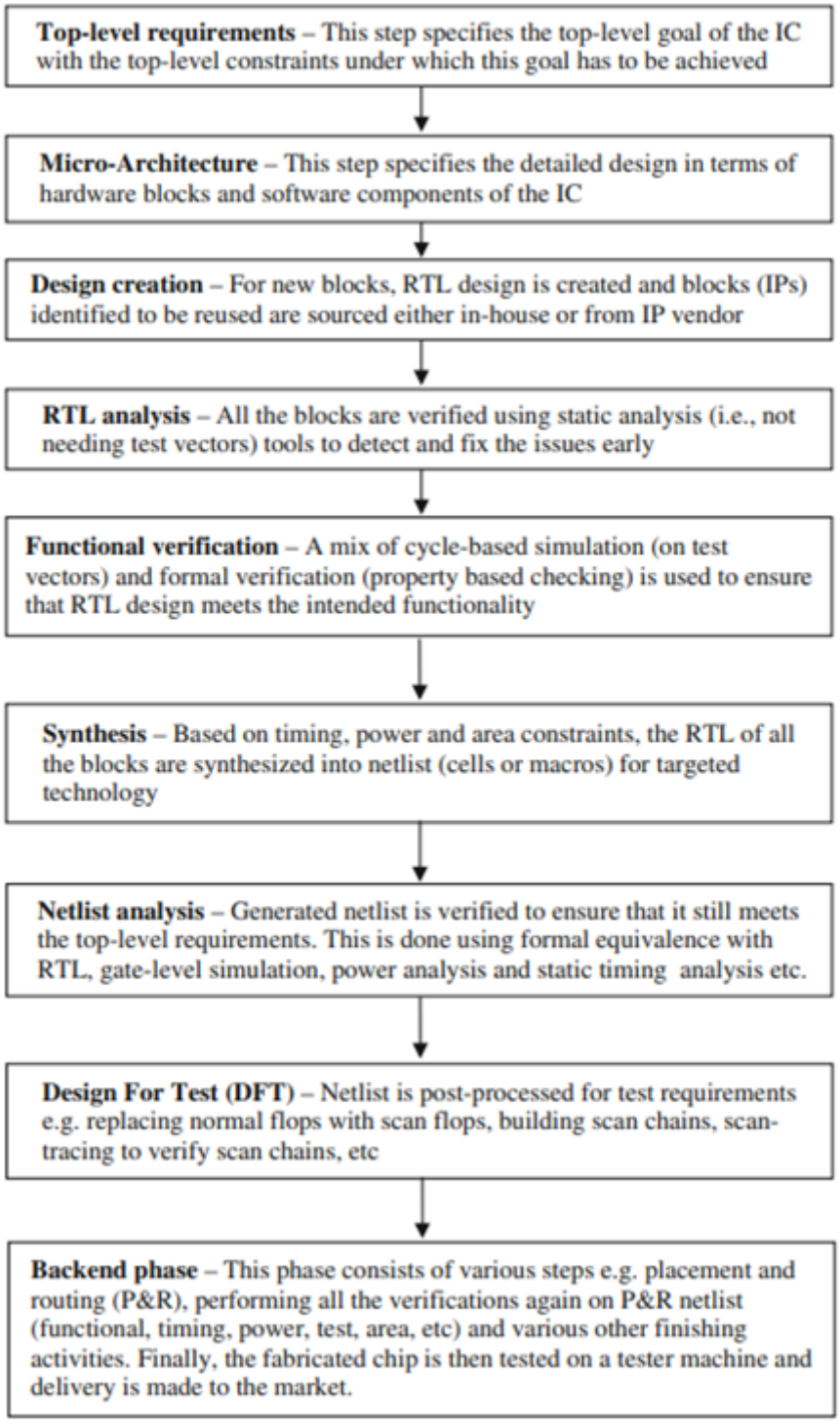

Figure 7: Hardware Design process[12].

4.3 AXI 4

Interconnection between the CPU cores, the packet scheduler, and other parts of the system has been implemented using the AXI4 interface. The configuration of the packet scheduler is being done of the AXI4-Lite interface. The Packet scheduler requires high-frequency near 1 GHz, also requires high-bandwidth and low-latency, which renders AMBA AXI4 a suitable choice.

Furthermore, AMBA specifications for interface standard improves IP reusability significantly. Therefore, thousands of SoCs, and IP products, are using AMBA interfaces, making it easier to integrate other modules from different sources[34]. Separation of the address and data in The AXI protocol offers a very high data throughput, especially with burst-based transactions that require only the start address.AXI also supports unaligned data transfers, using byte strobes and separate read and write data channels that can render a low-cost Direct Memory Access (DMA). AXI also supports multiple outstanding requests the can be used for non-blocking caching.

AXI specification does not just define an interconnect between IPs; instead, it defines the IPs interfaces themself. AXI interfaces are symmetrical and can be master or slave, containing the same set of signals. AXI also offers IPs to connect multiple master and slaves, such as AXI interconnect or AXI SmartConnect. but The direct connection between the master and slave IP with no extra logic provides the maximum bandwidth.

There are three types of AXI4 interfaces. First, AXI4 provides high-performance memory-mapped requirements. The full AXI feature's supported, such as bursts and multiple outstanding requests, is usually used in a complex system, a very high throughput system, or interfacing memories and caches. Secondly, AXI4-Lite is a simpler version of AXI4. AXI4-Lite is used for simple and requires much fewer wires and logic. Still, it is used in lower-throughput memory-mapped communication as it has some disadvantages, such as all transactions are of burst length one and data bus width is either 32-bit or 64-bit. Finally, AXI4-Stream is used for high-speed streaming data that does not require memory mapping [35].

5 OpenDataPlane

The Compound Annual Growth Rate (CAGR) of the Mobile data traffic is 46 percent from 2017 to 2022, reaching a seven-fold increase in this period and a total mobile network traffic of 77.5 exabytes per month by 2022 [36]. Moreover, billions of IoT end-nodes are being introduced to the network. On the other hand, the CPU performance increase is much slower due to the power and transistor size challenges that better performance CPUs face. The increasing difference between the data traffic and the CPU performance creates a forwarding gab. Therefore SoCs optimized for the network forwarding workload is currently playing a critical role in achieving the network requirements in 5G such as lower power, better user experience and higher throughput.

This Chapter discusses the architecture top-level of the ODP accelerator imple-

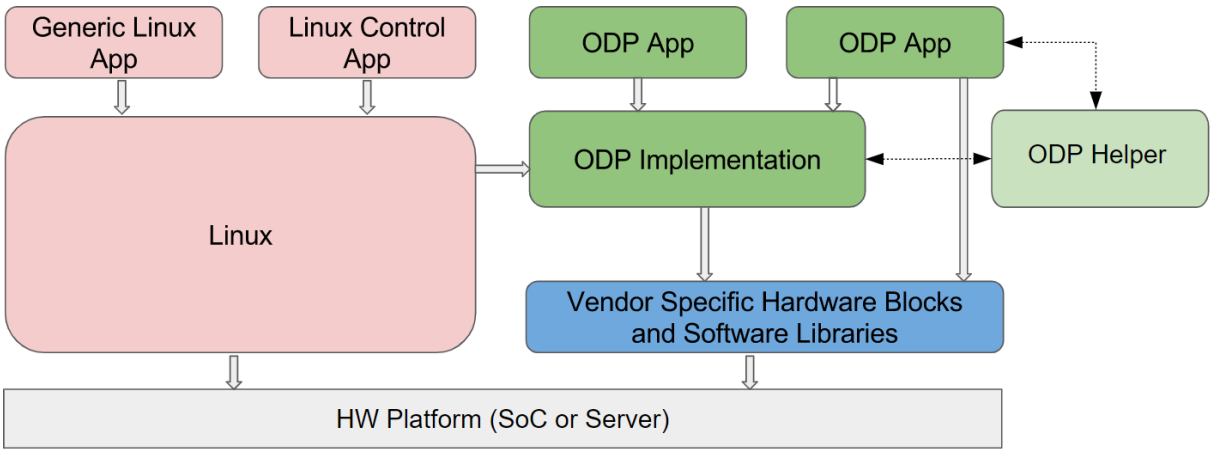

Figure 8: Overview of a system running ODP applications [29].

mented in the this thesis. The scheduling process is complex and has multiple components, as shown in figure 8. First, there are the servers, either x86 or ARM-based architecture. Secondly, there is the ODP software application. Finally, there is the hardware accelerator for some of the ODP function. The hardware implementation accelerates specific ODP functions, and the software handles the rest. Functionalities such as odp_schedule_group_join, odp_schedule_ group_create ,odp_schedule_group_destroy are handled by the software. The software controls functionalities, including creating new groups, queues, and threads, and the hardware does not get involved. This ODP implementation accelerates specific ODP functions such as scheduling, enqueuing, acquiring, and releasing the lock. Each function call is represented as a memory-mapped write operation where e.g. a write to address 0x20 represents an enqueue operation to group 2. The ODP implementation contains an input queue for scheduled events and a plain queue for the software-managed events. then the scheduler applies the specific scheduling algorithm to choose the events to dispatch to each thread.

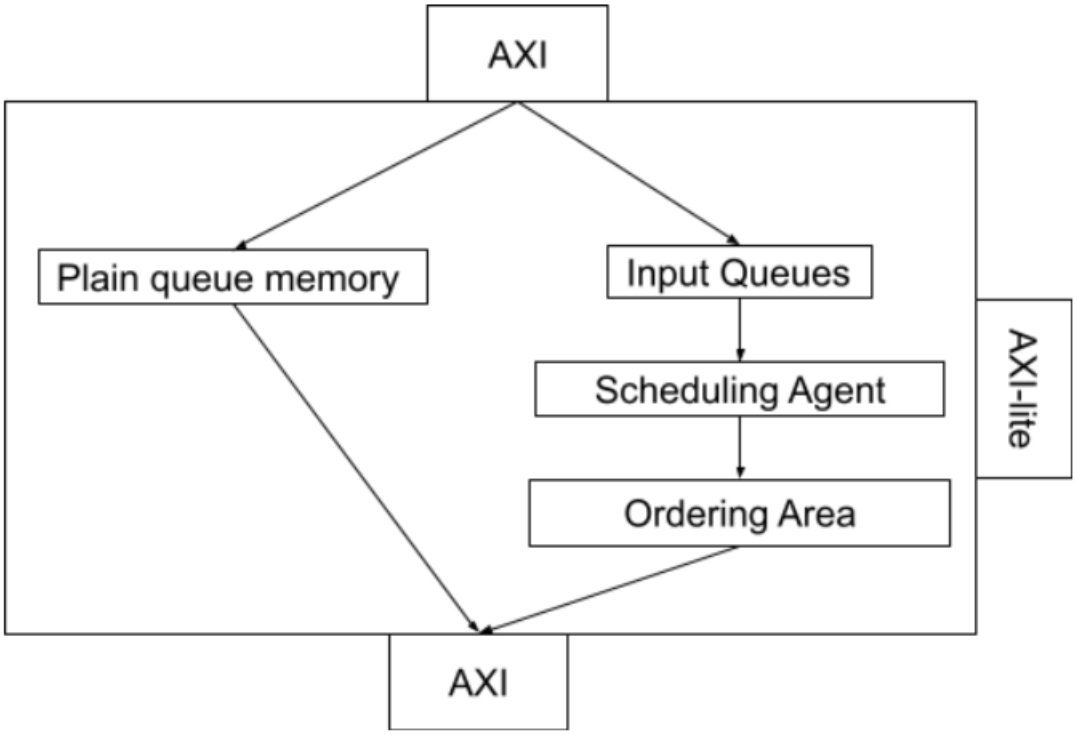

Figure 9: ODP Accelerator Architecture

5.1 Classifier

The classifier is one of the key elements in the ODP framework. It enables classification and prioritization of the packets allowing for optimum scheduling and processing in both hardware and software levels. Moreover, it is critical for satisfying the QoS requirements. The classifier is part of each operation executed in the ODP application, so it has to add minimal delay to the accelerator. The classifier is implemented as a singles cycle lookup table that applies the classification rules to the headers of incoming packets such as IP version, ethertype, IP protocol, transport layer port numbers, IP DiffServ, VLAN id 802.1p priority. The classifier is also responsible for validating the packet data integrity and correctness by checking different parameters such as checksum and length fields and store the validation result. After the classifier parses the incoming packet, it defines multiple parameters such as the ODP queue that this packet belongs to, the flow ID, Scheduling group, and priority level, then stores these parameters as packet metadata for the application.

As we are working on a software-defined network, the classification process is configurable and can be controlled by the ODP application class-of-service based on its Layer-2 802.1P/902.1Q VLAN tag priority fields that define priority levels also can the class-of-sevice can be determined using IP DiffServ header field in layer 3. Also, it defines whether to use Layer-3 priority or Layer-2 priority in a packet where both headers present. Finally, the application can specify the class-of-sevice based on

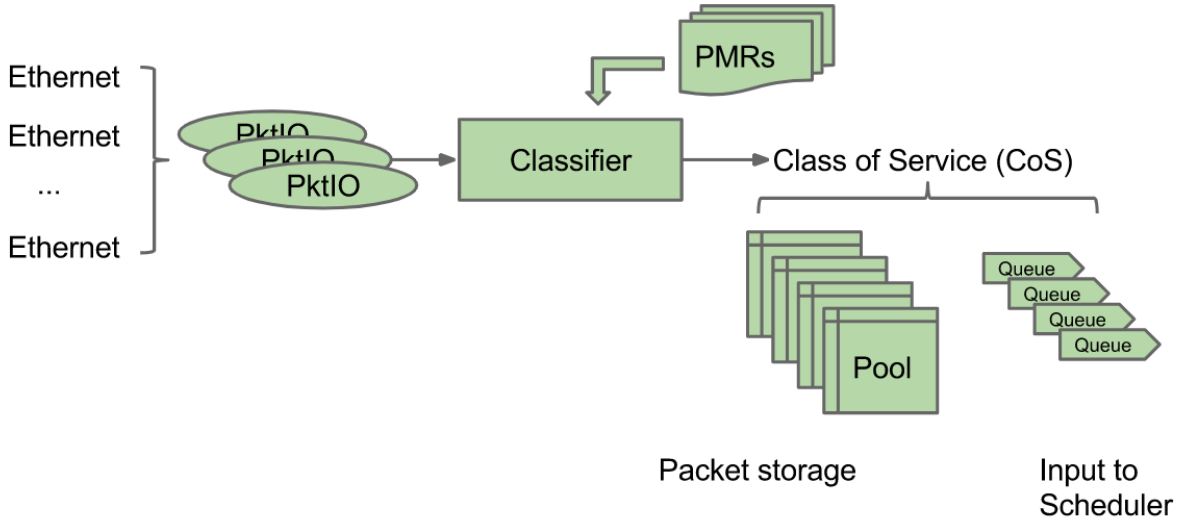

Figure 10: Classifier procedure in ODP Scheduler [29].

defined pattern matching rules that consider several factors such as the transmitting, receiving address, and transmission protocols. Using these classification tools, the classifier identifies incoming packets. Such as VoIP traffic and video streaming packets, destination, and source port numbers. Consequently, these packets are assigned to a class-of-sevice that maps to the highest priority allowing optimum processing for the voice packets and minimizing the latency and packet drops.

5.2 Input Queue Implementation

In our design, after the Classifier determines the packet's class-of-sevice and their meta-data, the packets are arranged in Events. Events are messages passed through queues. Classifiers determine the ODP queue that the event belongs. Events are inserted in a queue using the enqueue operation and dispatched using the dequeue operation. Queues are software instances that have two types two major types: plain and scheduled. Plain queues are handled totally by the software without any interaction from the hardware acceleration. The hardware accelerator manages scheduled queues based on the defined scheduling algorithm.

After the Enqueue instruction, the Event goes in one of two directions. First, The plain queue which represented as a memory managed by the software. When the software issue enqueues instruction for an event to plain queue, the plain queue's memory address is sent with the event. Also, when it issues a dequeue instruction, it sends the memory address to read from. Secondly, the Scheduled queues these queues are managed by the hardware. The software can have a massive number of ODP queues, and these software queues are mapped to hardware queues in the input queues. Input queue stores the event in groups, and the scheduler can choose

which event to schedule only from the head of these groups.ODP predefines several scheduler groups. These include default group (ODP_SCHED_GROUP_ALL), worker threads groups (ODP_SCHED_GROU_WORKER), and control threads group (ODP_SCHED_GROUP_CONTROL). The application can create additional scheduler groups for multiple purposes, but a maximum number of groups is supported by the hardware and cannot be exceeded.

5.3 Plain Queue

Plain queues are controlled directly by the ODP application, and events are enqueued and dequeued by software instructions. This offers flexibility for the software to bypass the scheduling algorithm. ODP programmers can use the flexibility of the plain queue to support multiple functionalities such as guaranteed bit rate (GBR) QoS flows. The plain queue can be considered as a tool that can be controlled directly by the software-defined network. The plain queue is implemented as two-port memory. The enqueue is a memory write operation, and the dequeue is a memory read operation. Any dequeue or enqueue operation takes one cycle at the ODP scheduler.

5.4 Scheduling agent

Scheduling agent is considered the main building block of the implemented design, as it is the primary location for performance upgrades. In the Scheduling agent Each queue has a priority parameter determining the queue's scheduling priority and which scheduler group is selected for dispatching events. According to application needs, though multiple priority levels are supported by QoS 3GPP specification, The hardware implementation supports eight priority levels. The software maps these hardware priorities to the different QoS priority levels. The software threads requests events then the Scheduling agent selects and dispatches one or more events to this thread. The Scheduling agent dispatch events based on multiple factors such as Event selection is based the queues containing schedulable events and the thread making an odp_schedule() call, and previously served events. The Threads that request events from the scheduler can be on the same processing core or in different cores.

Each ODP queue has a specific scheduling priority selected to meet the QoS by choosing which event should be scheduled first among the rest of the queues. Queue meta-data also contain Queues a scheduler group id that is used to provide event specific to the thread requesting the schedule function. as a result, the events are grouped into classes and threads can process events from specific classes, software can change scheduler groups by adding or removing a thread from a group dynamically, which allows the application to better handle demand increase.

The scheduling agent implements flow-aware scheduling. A flow consists of is a group of events that with the same context and belong to the same application. TCP flow is an example of flow-aware scheduling. TCP is one of the most popular internet protocols. TCP protocol requires reliability, error checking, and maintaining packet order. The TCP connection consists of multiple events such as payload data packets and timeout events and other transmission control packets. These packets are logically connected and must be received in the same order to be meaningful to the receiving application. Typically the single flow is represented as an ODP queue as all events from the same flow stored in the same ODPqueue. So the queue id and the flow-id are synonymous for those events. But as Queues are complex objects providing synchronizations in addition to user contexts for events. As a result, Flow aware scheduling is defined where the number of flows is more than the maximum number of queues defined in (max_queues). In our case, we support a 1024 ODP queue, but it contains 32 bits flow-id that can support up to 4G of flows. In flow-aware mode, the ODP tracks the flow-id separately as each flow-id is stored with the event. The application is responsible for assigning and interpreting the flow-id.A combination of flow-id and queue id in making scheduling decisions in flow-aware mode.

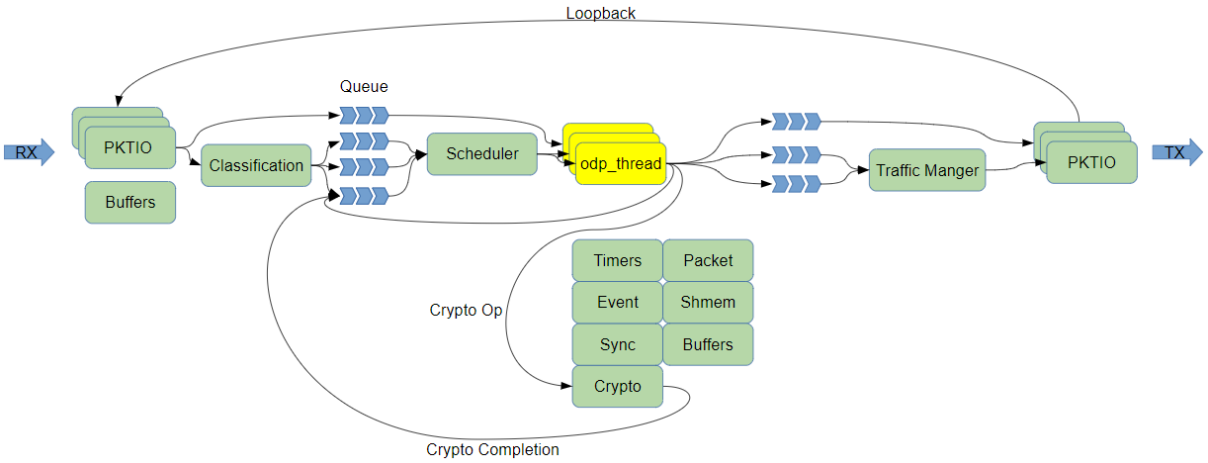

Figure 11: ODP Application Packet Flow Overview [29].

The packets' payload are usually stored in memory with a caching mechanism. The scheduling agent should also fully utilize the cache; this is implemented by giving higher priority to the events that belong to the same queue that have been scheduled recently.First, odp_schedule runs the event scheduler to find the next highest priority event, which is available for the calling thread. Only queues that have been created with ODP_QUEUE_TYPE_SCHED type are connected to the scheduler. Optionally, outputs the source queue of the event. If there's no event available, waits for an event according to the wait parameter setting. Returns ODP_EVEN_INVALID if it reaches the end of the waiting period. The scheduling agent takes 128-bit wide

input from the input queue called input_queue_group_empty. Each bit refers to one of the hardware groups as 1 is empty and 0 is not. After that, the scheduling agent applies the thread mask. The thread mask is a 128-bit register representing which groups the thread issuing the schedule can receive events from. The scheduler implementation supports up to 32 hardware threads. the scheduling agent contains 128 thread mask for each hardware thread. then scheduling agent uses the 128-bit priority register to choose which of the events are eligible to be dispatched to the threads. cache_hot_reg is 128 by 4 register, the cache_hot_reg is reset to all ones. with each 4bits represent a counter for one of the groups that have been recently dispatched. after that, we choose events with the highest cache_hot_reg value we subtract one from the value corresponding to the chosen group, then random event from the qualified events is dispatched from the scheduler.

5.5 Synchronization

ODP Scheduler accelerator does not only dispatch event but it is also responsible for synchronization between events running on multiple threads, which provides automatic scalability to ODP applications in the many-core environment. ODP scheduler is also responsible for providing event synchronization methods to simplify programming parallel processing applications. Also, it assists parallel processing and increases parallel processing performance significantly. The sync mode per queue can be Parallel, Atomic, or Ordered. These synchronization modes are used to determine how multiple events from the same queue are ordered when multiple events are dispatched from the same queue.

First, Parallel Queues ODP_SCHED_SYNC_PARALLEL is the mode Scheduled queues, where no synchronization is required between the events. Parallel events can be scheduled to any thread without any restrictions and execute simultaneously. Synchronization required between parallel events is handled by the software. Parallel events have the best throughput as no synchronization is required and can fully utilize all the available threads, but it shifts the synchronization work to the application. parallel queues do not require extra logic in the hardware implementation as the dispatched event is not tracked once it is scheduled in the case of parallel sync mode.

Secondly, Atomic queues simplify event synchronization which implies better performance in the ODP application as atomic synchronization can be cumbersome and degrade the performance noticeably when implemented on the software level. In Atomic sync mode, only one event from the same queue can be processed by a thread at the same time. consequently, the software can process atomic events lock-free. The atomic sync is implemented by providing the number of atomic locks equal to the max number of the thread supported by the ODP scheduler. When the scheduler dispatches event of sync mode atomic the scheduler checks the atomic locks first and if the scheduler is operating in the flow-aware mode it compares both the queue id

and the flow-id with the flow and queue ids stored at the atomic locks if flow-aware scheduling is disabled only the queue id is compared. flow-aware scheduling provides better performance for the atomic events as only events the belong to the same flow can block each other not all the events in the same ODP queue. The synchronization is only for events that belong to the same atomic queue and no synchronization between events belonging to different queues. The atomic queue is locked until either the next call to the scheduler releases the lock using odp_schedule_release_atomic(). odp_schedule_release_atomic Release the current atomic context. This is called by the software once the critical section that required order has completed execution but the thread does not require another event yet. after that the scheduler can signal other threads that are waiting for this ordered event to proceed with their critical section However, the context may still be held until the next odp_schedule() as this call allows but does not force the scheduler to release the context early. though this function is not a must with the support for the voluntary atomic release the performance can increase significantly.

Though ordered queue is the most complex to implement, it provides very useful capabilities as it is a trade-off between parallel and atomic queues by providing higher throughput closer to parallel queues while maintaining synchronization between events. In ordered queue, the scheduler is free to dispatch multiple events to different threads at the same time similar to parallel queues, but it also keeps track of the order of the dispatched packets, so the output queue order is the same as the source queue order. Similar to atomic queues, the ordering is maintained between events belonging to the same queue, and no ordering is tracked between events belonging to different queues. The ordering is implemented using ordered locks with one or more lock specified per queue by lock_count parameter at queue create time.

Ordered locks in Ordered queues provide more efficient synchronization than atomic queues. As it allows events to be processed in parallel while keeping the order only for the critical sections that require synchronization. Ordered locks are efficient when used with events that contain only part of the code that has to be executed sequentially and in order which is called the critical section, while the rest of the event can be executed in parallel. When these threads need to synchronize, Orderd locks are used to ensure that the critical sections are executed in the correct queue order. Multiple ordered locks can be used to support multiple critical sections but this implementation contains only one order lock per thread. odp_schedule_release_ordered()is similar to the odp_schedule_release_atomic (). The thread calls this function to notify the scheduler that it no longer requires the ordered lock as it finished the critical section and all the required enqueues to keep the order are complete. but the context may still be held by the lock as this function just hints at the scheduler. The next odp_schedule() call releases the context if it is still held. The release function is used to increase the parallelism and the overall system performance.

When the ODP application starts executing a critical section that needs to be

ordered, The application issues odp_schedule_order_lock() to acquire the ordered lock. If it is the event turn to acquire the ordered lock the thread starts executing the critical section. However, if it is not the event turn the thread is blocked until preceding events have released this ordered lock. Threads can release the context of the ordered lock without using it to handle a critical section. lock_count parameter sets the number of ordered locks per queue and this parameter is passed to the queue when it is created using odp_queue_create(). Max_ordered_locks defines the maximum number of locks available for all the queues. The same lock cannot be used multiple times for the same event. If a thread contains multiple critical sections in processing an event, then multiple ordered locks need to be defined as the ordered lock can be acquired and released once. If events are processed in multiple critical sections an ordered lock is defined for each section. With multiple ordered locks for an event, a smaller granularity is used for parallelism which increases the performance significantly.

Each thread contains a pointer that points to the event dispatched to it, for the atomic event the thread also contains a pointer to the blocked events so once the context is released the blocked events are again ready for scheduling. The ordered queues on the other hand are stored in a ring buffer with a pointer that indicates the first event in the ring buffer. The order lock cannot be acquired unless the thread that trying to acquire the lock is the first event in the ring buffer. once the first event releases the context the ring buffer is updated, and that event is discarded.

6 Results

The ODP scheduler hardware accelerator is implemented on a Zynq Ultra-Scale+ FPGA (xczu9eg-ffvb1156). The Implementation consists of four main parts First, the Instruction Decoder that is connected to the AXI bus and provide all the required signals to the other blocks. Secondly Input Queue which stores all the incoming events in an array of FIFOs until they are scheduled. The scheduling agent provide the scheduling mechanism that aims to increase the performance and satisfy the quality of service requirements. Finally the synchronization agent which make sure that all the results are in the correct order.

6.1 Instruction Decoder

Instructions through AXI bus are memory mapped writing to different addresses implements different function as address from 0x1 to 0x80 represent a enqueue to specific input queue as 0x1 represent an enqueue to group 0 and write to address 0x80 represents an enqueue to group 128. An queue and dequeue to plain queue is implemented using a write to address 0x100 to 0x500 as the plain queue is 1024 event deep the first slot is accessed using address 0x100 and the last is addressed using address. A R/W bit is added with the instruction to define a read or write

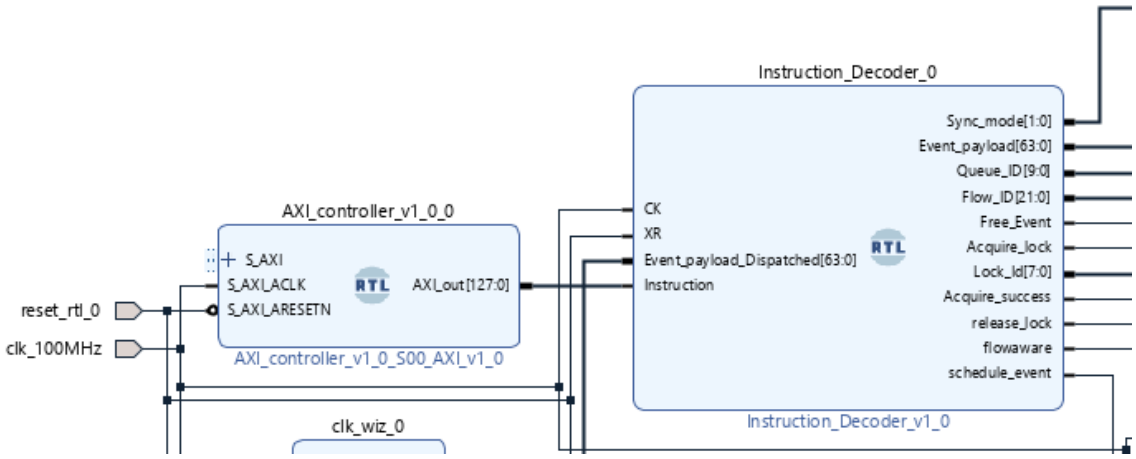

Figure 12: Instruction Decoder Interface.

operation. The Scheduler also contains a 1024 lock that belong to address 0x600 to 0x1000 an acess to this address is used to acquire/release these locks. Finally the scheduler supports 16 different threads with registers at address 0x1100 to 0x1140. The instruction decoder handles the memory mapped requests to the register address and passes the information required to all the other modules as shown in figure 12.

6.2 Input Queue

The input queue is implemented as an array of 128 FIFO, where each FIFO represents one hardware group. The order between events that belong to the same hardware group is maintained, while no guarantee for ordering between events from different groups. Each queue belongs to a specific group. The Input Queue is also responsible for choosing packets to drop when it gets full. It does so by keeping track of the ODP queues usage of the total size and slows or stops the queues that are taking more than its fair share of the traffic. As Shown in figure 13, The input queue provide a list of the groups that contains events to the scheduling agent, and after the scheduler agent completes the algorithm and chooses the group to schedule from it sends the groups ID back to the input queue which in turn sends it to the synchronization agent.

6.3 Synchronization Agent

Synchronization Agent is implemented to keep track of the ordering requirements between different Event that belong to the same queue. Starting from the parallel queues, any request for a parallel queue will be successful as different threads don't block each other. As shown in figure 14,sync_mode 0 refers to parallel event type.

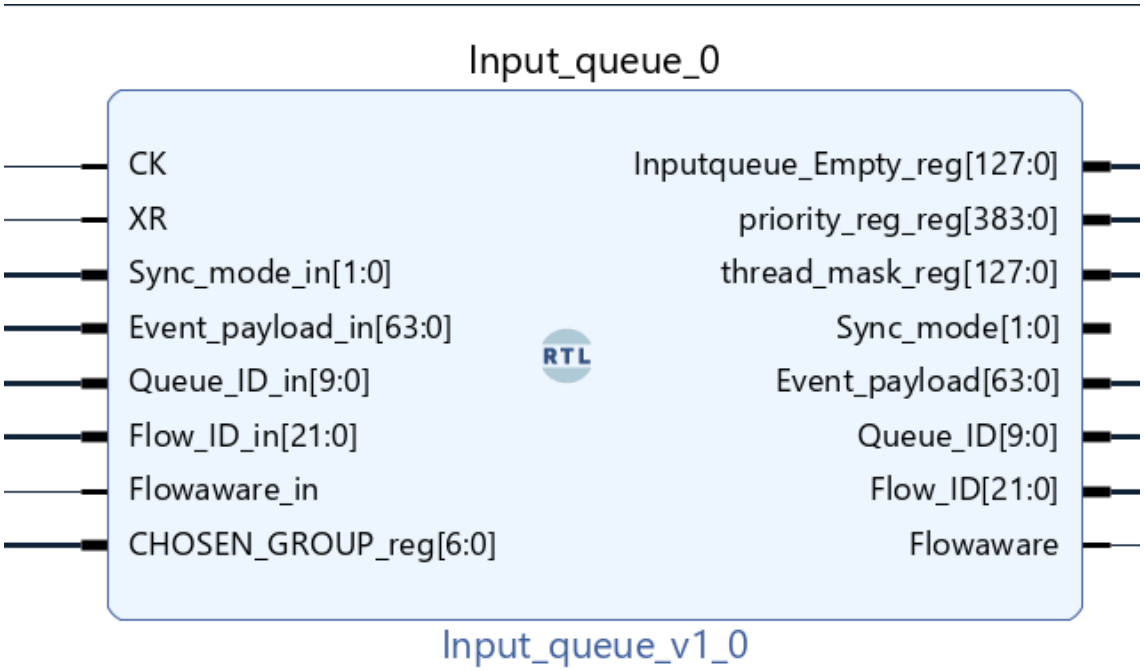

Figure 13: Input Queue Interface.

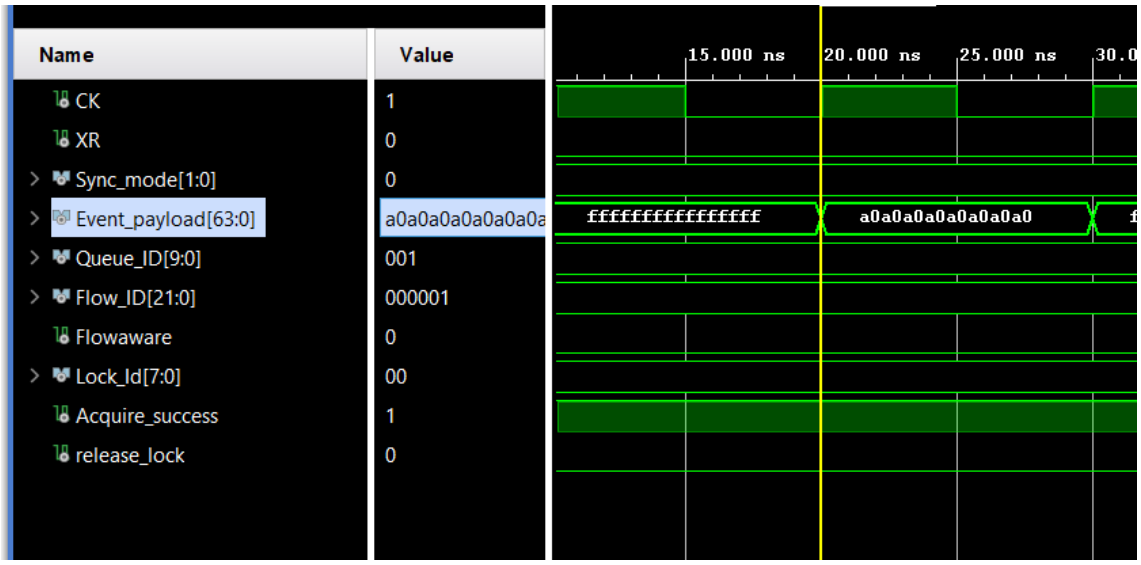

Figure 14: Parallel Events doesn't block each other.

Although both the requests access the same queue no blocking happened for the parallel Event.

6.4 Scheduling Agent

The Scheduling Agent is activated when a schedule instruction is issued, the send a scheduler signal to the scheduling agent as shown in figure 16, The scheduling agent also receive a list of all the groups in the input queue that are not empty stored in a 128 bits vector called Inputqueue_Empty as shown in figure 17, also a 128 bit vector blocked_list_Empty representing whether blocked events in the synchronization area have became ready.First, Scheduling Agent prioritize the previously blocked events as shown in figure 17 in order to avoid deadlocks and increase the performance.

Secondly, The 128 bit thread mask supplied by the requesting core is applied to the result of the previous stat as shown in figure 18. The events eligible to continue for the next step are then stored in 128 bit vector FIFO_Masked. Thirdly, the groups that have the higher priority are passed and the lower priority groups are excluded. Figure 19 provides an example for the events selection in this stage as the highest priority in this case was 4 only event belonging to groups with priority of 4 are passed and the rest are masked out. Finally a random selection between all the eligible events is executed as shown in the next section. Figure 21 shows a complete example of the process inside the scheduling agent.

Secondly, Ordered Events a linked list contains all the events associated with the a specific a queue, Similar to the parallel queues the acquire of event always succeed but when the thread enters the critical section it request the ordered lock and the request is completed only when the event is the top of the list otherwise the lock isn't acquired and the thread is blocked.
Finally, Atomic Queues in which the acquire isn't completed as long as an event from that same queue is being executed.

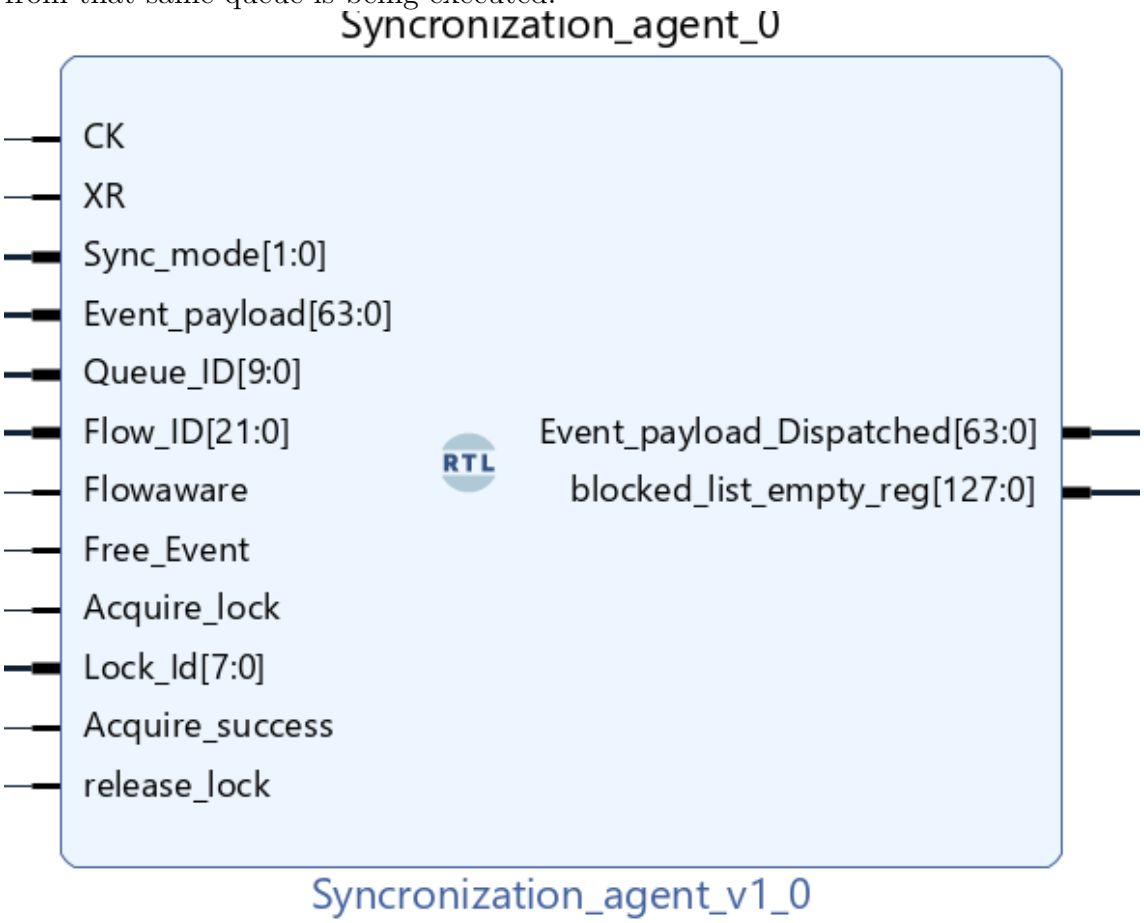

Figure 15: Synchronization Agent Interface.

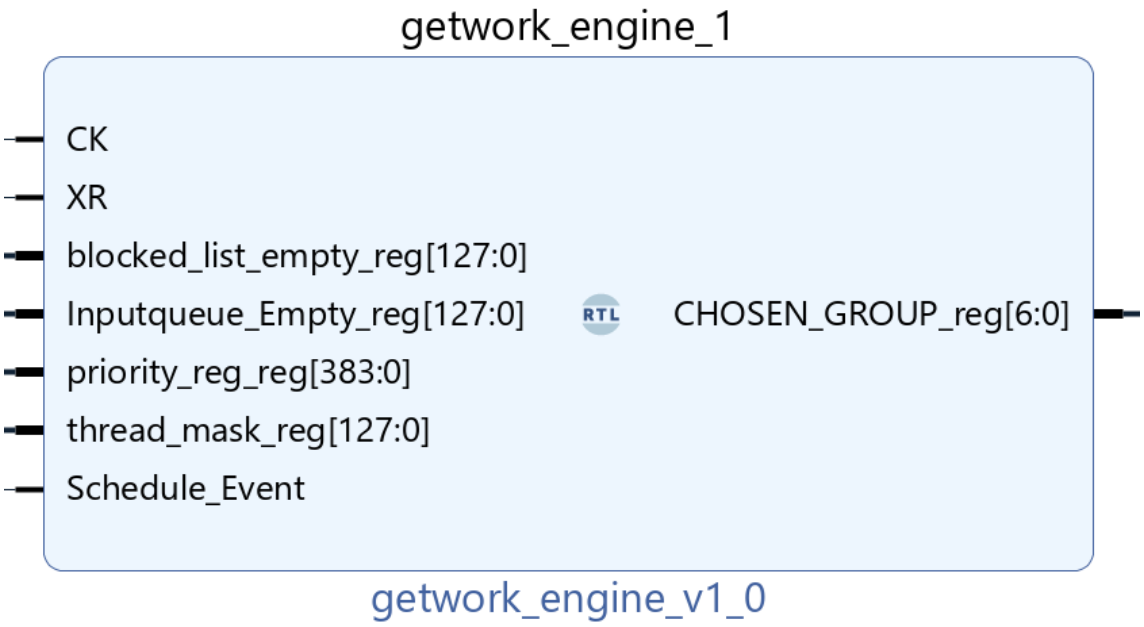

Figure 16: Scheduling Agent Interface.

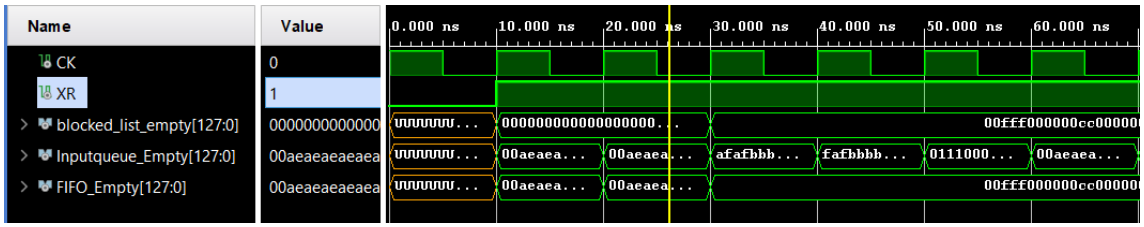

Figure 17: Blocked Event that became ready are prioritized by the scheduling agent.

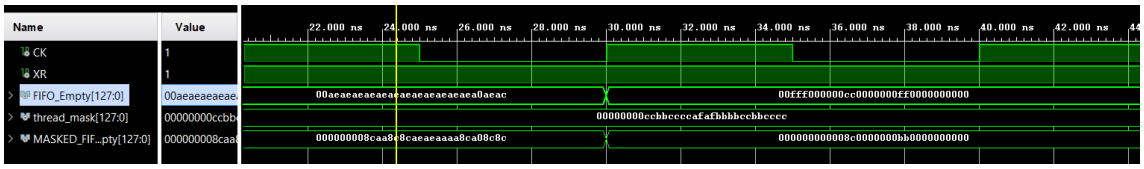

Figure 18: Only Events from Groups allowed by the requesting Mask can be scheduled.

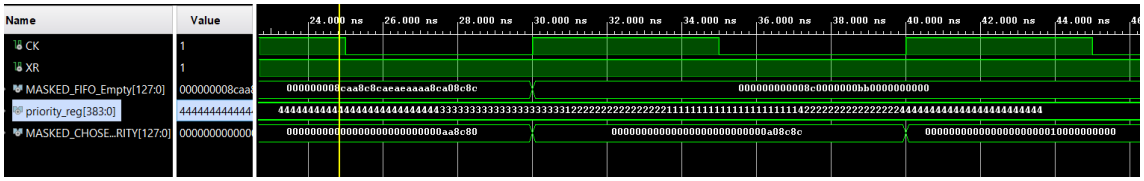

Figure 19: Only Events in Highest priority groups are allowed to be scheduled.

Resource	Utilization	Available	Utilization %
LUT	2867	274080	1.05
FF	689	548160	0.13

Figure 20: Resources used by the scheduler Agent.

6.5 Random Selection

In the Scheduler design implementation that is developed in the thesis after the applying the priority, the mask and the cache hot register, a number of events will to remain so the scheduler should chose one of them to dispatch to the requesting thread. The number of the eligible events and their location is the random depending on the state of the input queue, the descheduling request and the state of the scheduling agents. Consequently, choosing event randomly from the eligible events isn't easy. As show in figure 21, The random generation is implemented using a wrap around 15 bit counter that is increased by one each clock cycle, The events remaining from the previous scheduling stages are represented 128 signal MASKED_CHOSEN_AFTER_WGT where each bit represent a group, in which one represents that this group has an eligible event and zero represent that it doesn't. The MASKED_CHOSEN_AFTER_WGT is shifted by the count modulus 128 which is the number of groups. After that the index of the first bit that equals '1' is calculated. Finally, the calculated index is added to count modulus 128 to calculate the index of the group that an event is going to be dispatched from.

6.6 Configuration

As flexibility is one of the key features of the ODP scheduler, applying the configuration and increasing the software on control on the accelerator performance is crucial. AXI-Lite port is used for configuration. The configuration is changed by memory-mapped write to the chosen register. this configuration register can be used to control the QoS performance and the scheduling priority. The registers that can be configured are sched_priority_reg setting the priority for each group, also defining the cache_hot_max_reg which defines the max number of the cache_hot_max_reg that is used to set the max counter value for the cache_hot_reg. Also, input_queue_max which defines the max number of event per-queue that can be at the input queue at the same time and after that packets from that queue is dropped which is one of the key factors of the QoS and prevent one queue from starving the others.. another register such as input_queue_threshold is also configured by the AXI-Lite port. Flow-aware scheduling can also be turned on/off through the AXI-Lite port.

AXI-Lite port also provides an API that can be used by the software to find out

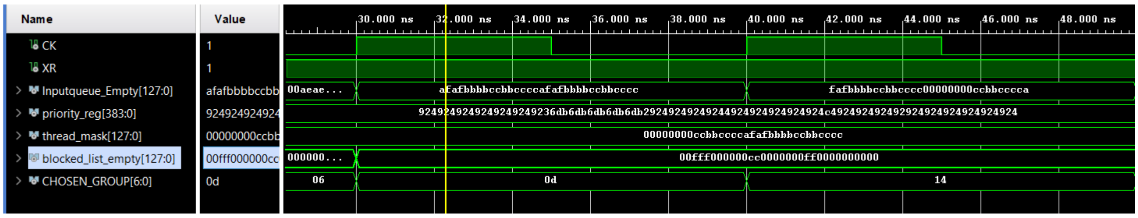

Figure 21: Simulation of random selection.

Site Type	Used	Fixed	Available	Util%
CLB LUTs*	716	0	274080	0.26
LUT as Logic	716	0	274080	0.26
LUT as Memory	0	0	144000	0.00
CLB Registers	19	0	548160	<0.01
Register as Flip Flop	19	0	548160	<0.01
Register as Latch	0	0	548160	0.00

Figure 22: Random Selection Resources Usage.

all the supported functionalities in the ODP accelerator as the same ODP software application work independent of the hardware implementation. The Software can use this API to inquiry about the number of hardware threads that the ODP queue supported, the number of ordered events that can be scheduled at the same time, the number of the ordered locks, the size of the input queue, the number of hardware groups.

This implementation is as shown in figure 22 is simple and uses only 716 LUTs and 19 Flip flop, It doesn't provide perfect synchronization, Using this algorithm event in the lower groups starting from 0 have a higher chance in getting chosen than events at the back as it can be noticed in figure 23.

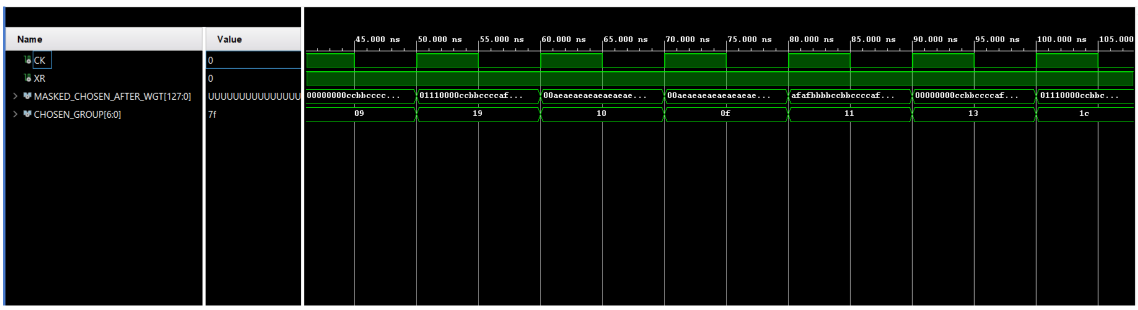

Figure 23: Simulation of random selection.

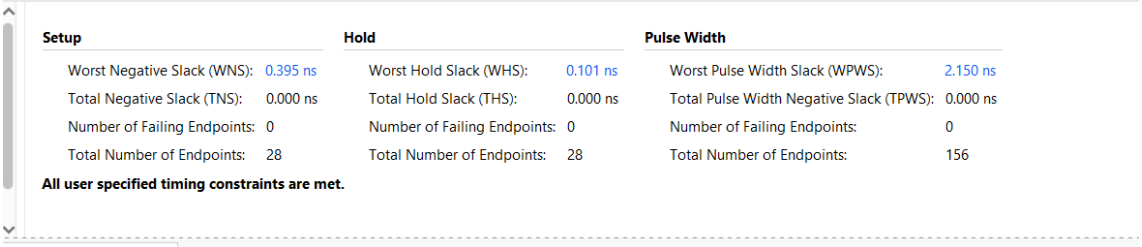

Figure 24: Timing passes at 200Mhz.

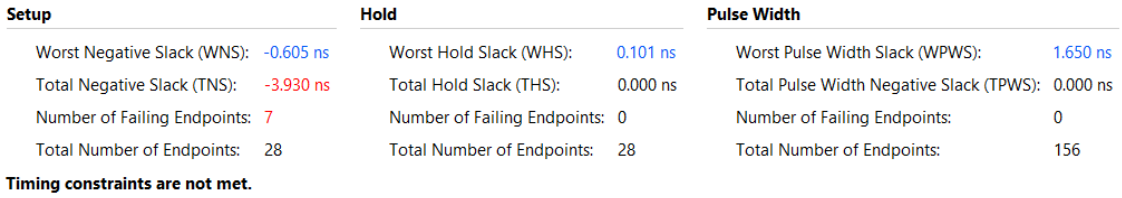

Figure 25: Timing fails at 250Mhz.

6.7 Conclusion

A clock of 200MHz was achieved. However, a setup time violation occurred when increasing the frequency to 250 MHz as shown in figure 24, 25. Although 200MHz is not ideal for the scheduler. The scheduler implementation targets ASIC implementation, rendering it possible to achieve the required clock frequency. With the scheduling agent containing the longest clock path. The minimum delay for an event that is enqueued to the input queue to be dispatched to one of the threads is three cycles as the reading and writing from the input queue takes one cycle, the scheduling agent takes one other cycle to choose the event.

The scheduler implementation offers multiple trade offs and has a lot of locations to improve. First, A group also contain a thread mask which defines which thread can process event from this group. Threads can be located on different cores. As a result, the number of groups defines the scheduler's level of parallelism, which provides the first trade-off in the scheduler design. Although extra parallelism is desirable and leads to better performance, the number of groups can create extra control overhead.

The second trade-off faced in the input queue design is the use of external memory to store the input events. The trade-off depends on two factors of the network firstly is the number of packets dropped, and secondly is the latency requirements. With external memory access, the input queue can virtually store any number of the events with zero packets lost. But 5G has very low latency requirements, which decreases the importance of packets delivered too late. With a target clock of 500MHz queue that can store 10k events has a maximum delay of 20us. The trade-off is between the importance of keeping the delayed packets.this design a queue of size 32768 events in

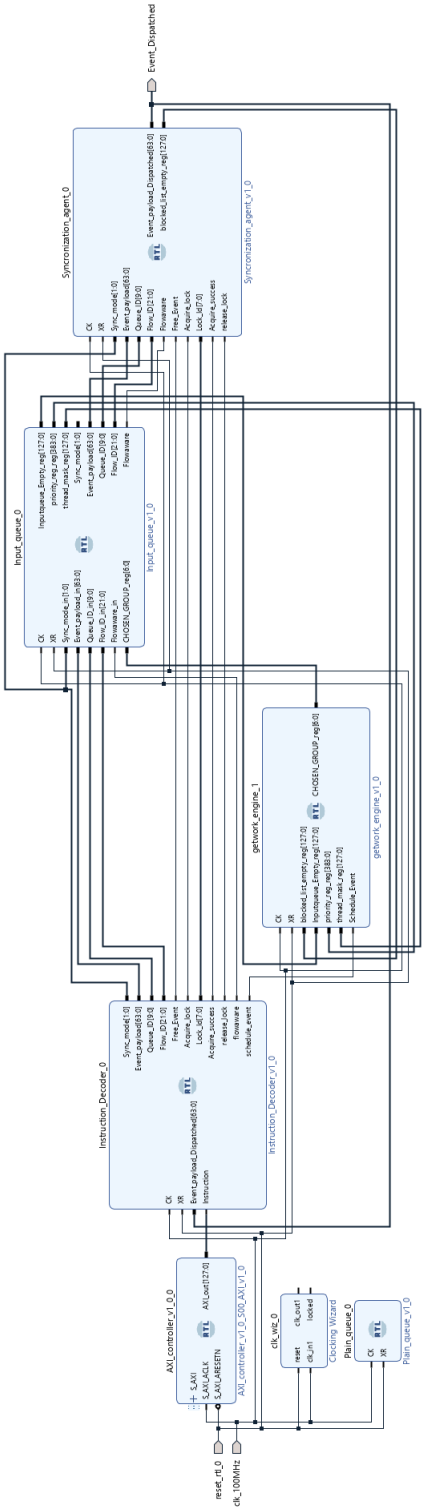

Figure 26: Simulation of random selection.

which packets can take up to 64us delay, and other packets are dropped. Quality of service requirements are provided at the input queue with a maximum threshold configured for each queue, preventing any queue from filling the input queue and blocking other queues.

Thirdly the number of locks, Though having multiple locks is very useful especially in using multiple critical sections for ordered events in the thread. Increasing the number of events increases in turn the logic used to check with a queue is locked or not and the time to acquire a lock consequently increasing the clock speed thus decreasing the performance.

Vast number of locations are available for improvement. First, the random selection implementing more fair algorithm in choosing events randomly. Secondly, Improving the cache utilization by making sure that the events from the same queues are executed multiple times in a row to keep hitting the cache. Finally, smaller granularity in the Input queue instead of splitting it into groups with equal size which can was queue size in case of the traffic isn't equally split between all the groups.

References

[1] www.ericsson.com. 2020. Mobile data traffic outlook. [online] Available at: /urlhttps://www.ericsson.com/en/mobility-report/dataforecasts/mobile-traffic-forecast [Accessed 10 January 2021].

[2] Jonsson, P. and Carson, S., 2021. Ericsson Mobility Report November 2020. [online] ericsson.com. Available at: urlhttps://www.ericsson.com/4adc87/assets/local/mobility-report/documents/2020/november-2020-ericsson-mobility-report.pdf [Accessed 9 October 2020].

[3] Nokia. 2021. Survey: When it comes to 5G, this is what consumers want | Nokia. [online] Available at: https://www.nokia.com/blog/survey-comes-5g-consumers-want [Accessed 4 March 2021].

[4] W. Xiang, K. Zheng, and X. Shen, 5G Mobile Communications. Cham, Switzerland: Springer International Publishing, 2017

[5] J. Rodriguez, Fundamentals of 5G mobile networks. Chichester, UK: John Wiley & Sons, 2015.

[6] J. Yao, J. Guo and L. N. Bhuyan, "Ordered Round-Robin: An Efficient Sequence Preserving Packet Scheduler," in IEEE Transactions on Computers, vol. 57, no. 12, pp. 1690-1703, Dec. 2008, doi: 10.1109/TC.2008.88.

[7] Nokia. 2021. AirScale Radio Access | Nokia. [online] Available at: https://www.nokia.com/networks/solutions/airscale-radio-access [Accessed 4 November 2020].

[8] Nokia. 2021. Nokia launches ReefShark chipsets that deliver massive performance gain in 5G networks | Nokia. [online] Available at: https://www.nokia.com/about-us/news/releases/2018/01/29/nokia-launches-reefshark-chipsets-that-deliver-massive-performance-gain-in-5g-n [Accessed 12 October 2020].

[9] Ashenden, P. J. The Designer's Guide to VHDL. 3. Edition. Burlington, Morgan Kaufmann Publishers, 2009

[10] "5G evolution: 3GPP releases 16 & 17 overview", ericsson.com, 2020. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-nr-evolution. [Accessed: 21-Dec- 2020].

[11] www.ericsson.com. 2019. Mobile data traffic outlook. [online] Available at: /urlhttps://www.ericsson.com/en/mobility-report/reports [Accessed 10 November 2020].

[12] "5G Releases 16 and 17 in 3GPP", Onestore.nokia.com, 2020. [Online]. Available: https://onestore.nokia.com/asset/i/207276. [Accessed: 12- Nov- 2020].

[13] Ibm.com. 2021. SDN Versus Traditional Networking Explained. [online] Available at: <https://www.ibm.com/services/network/sdn-versus-traditional-networking> [Accessed 4 March 2021].

[14] Peterson, L. and Sunay, O., 2019. Chapter 3: Basic Architecture — 5G Mobile Networks: A Systems Approach Version 1.1-dev documentation. [online] 5g.systemsapproach.org. Available at: https://5g.systemsapproach.org/arch.html#:~:text=by%20the%20spec.-,5G%20Mobile%20Core,blocks%20and%20not%20an%20implementation. [Accessed 5 March 2021].

[15] "Open RAN explained | Nokia", Nokia, 2021. [Online]. Available: https://www.nokia.com/about-us/newsroom/articles/open-ran-explained/. [Accessed: 03- Feb- 2021].

[16] U.Fattore, F.Giust, M.Liebsch, "5GC+: an Experimental Proof of a Programmable Mobile Core for 5G", 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona 2018.

[17] European Telecommunications Standards Institute, "NG-RAN: Architecture description," European Telecommunications Standards Institute, ETSI TS-138-401 2010. [Online]. Available: http://www.etsi.org. [Accessed: December. 17, 2020].

[18] "5G;NR;Overall description;Stage-2", Etsi.org, 2021. [Online]. Available:https://www.etsi.org/deliver/etsi_ts/138300_138399/138300/15.08.00_60/ts_138300v150800p.pdf. [Accessed: 13- Oct- 2020].

[19] H. Jonathan Chao and Bin Liu. High Performance Switches and Routers. Wiley-IEEE Press, 2007. ISBN 0470053674.

[20] N. Le, M. Hossain, A. Islam, D. Kim, Y. Choi and Y. Jang, "Survey of Promising Technologies for 5G Networks", Mobile Information Systems, vol. 2016, pp. 1-25, 2016. Available: 10.1155/2016/2676589 [Accessed 1 January 2021].

[21] B. Han, V. Gopalakrishnan, L. Ji and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," in IEEE Communications Magazine, vol. 53, no. 2, pp. 90-97, Feb. 2015, doi: 10.1109/MCOM.2015.7045396.

[22] Grandviewresearch.com. 2021. Global 5G Services Market Size Report, 2021-2027. [online] Available at: <https://www.grandviewresearch.com/industry-analysis/5g-services-market> [Accessed 9 March 2021].

[23] W. Shi, X. Zhuang, I. Paul and K. Schwan, "Efficient Implementation of Packet Scheduling Algorithm on High-Speed Programmable Network Processors", Management of Multimedia on the Internet, pp. 184-197, 2002. Available:10.1007/3-540-45812-3_15 [Accessed 7 January 2021].

[24] C. F. Müller, G. Galaviz, Á. G. Andrade, I. Kaiser, and W. Fengler, "Evaluation of Scheduling Algorithms for 5G Mobile Systems," Computer Science and Engineering—Theory and Applications Studies in Systems, Decision and Control, pp. 213–233, 2018.

[25] algorithms for LTE using weights," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, 2016, pp. 264-269, doi: 10.1109/ICATCCT.2016.7912005.

[26] F. Heliot, M. A. Imran and R. Tafazolli, "Low-Complexity Energy-Efficient Resource Allocation for the Downlink of Cellular Systems," in IEEE Transactions on Communications, vol. 61, no. 6, pp. 2271-2281, June 2013, doi: 10.1109/TCOMM.2013.042313.120516.

[27] Stallings, William (2001). "Glossary". Business Data Communication (4 ed.). Upper Saddle River, New Jersey, USA: Prentice-Hall, Inc. p. 632. ISBN 0-13-088263-1.

[28] "OpenDataPlane™ Introduction and Overview," Linaro Networking Group (LNG) January 2014. [Online]. Available: [Accessed: December. 20, 2020]. http://www.opendataplane.org.

[29] "OpenDataPlane (ODP) Users-Guide", Opendataplane.github.io, 2019. [Online]. Available: https://opendataplane.github.io/odp/users-guide/. [Accessed: 01- Oct- 2020].

[30] "Technical Overview", Opendataplane.org, 2021. [Online]. Available: https://opendataplane.org/index.php/service/technicaloverview/. [Accessed: 08- Sep- 2020].

[31] Nagarahalli, H., 2018. HKG18-409 - DPDK vs ODP: A comparison. [online] Slideshare.net. Available at: https://www.slideshare.net/linaroorg/hkg18409-dpdk-vs-odp-a-comparison [Accessed 14 March 2021].

[32]] S. Brown and Z. Vranesic, Fundamentals of Digital Logic with VHDL Design. McGraw-Hill, 2009.

[33] OpenFastPath, "OpenFastPath-An Open Source Accelerated IP Fast Path," openfastpath.org March 2013. [Online]. Available:https://openfastpath.org/index.php/service/technicaloverview/ [Accessed: December. 1, 2020].

[34] "Introduction to AMBA AXI", Developer.arm.com, 2021. [Online]. Available: https://developer.arm.com/architectures/learn-the-architecture/introduction-to-amba-axi/single-page. [Accessed: 18- Nov- 2020].

[35] "AXI Reference Guide", Xilinx.com, 2021. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf. [Accessed: 20- Oct- 2020].

[36] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022", https://cisco.com/, 2019. [Online]. Available: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1955935://davidellis.ca/wp-content/uploads/2019/12/cisco-vni-mobile-data-traffic-feb-2019.pdf. [Accessed: 29- Jan- 2021].