Benefits of Virtualizing the Layer 1 in a RAN Stack

Niall Power
Platform Software Architect

Sindhu Xirasagar
Sr. Director, RAN Solutions

Table of Contents

Overview of Layer 1 Implementation Approaches 1
Software-based L1 on general Purpose CPU-based architecture 1
L1 on purpose-built SoC architecture 2
RAN L1 implementations: Implications to RAN networks. . . .3
Vendor choice for all components (silicon, hardware, OS/virtualization, software). . . .3
Ecosystem size, availability of experienced personnel 3
Velocity and cost of network evolution, e.g., reconfiguration, feature additions and changes 3
Velocity and cost of network infrastructure scaling. 3
Hardware acceleration and cloud native deployments 4
Realizing cloud native RAN benefits. 4
Network Slicing 5
Pooling-enabled optimized infrastructure, network availability and resiliency 5
Energy efficiency and sustainability. .7
Conclusion. 8

Introduction

The mobile network is undergoing a transformation from infrastructure built with purpose-built silicon to fully virtualized platforms running on commercial off-the-shelf (COTS) hardware based on general purpose processors. Early commercial software-based RANs were first deployed in 2018, and since then the pace of new deployments and network expansions continues to grow steadily. In addition, the industry can start deploying new capabilities that will deliver on the full potential of a fully virtualized infrastructure.

In this paper, we refer to a fully virtualized RAN infrastructure as one where all the software-based RAN stack layers (Layer 1 or L1, Layer 2 or L2, and Layer 3 or L3) execute primarily on general purpose processors (CPUs). This approach gives operators a platform to achieve numerous operational and business benefits, including:

- reduced total cost of ownership (TCO).
- continuous innovation for delivering new revenue-generating services.
- energy efficiency, not just from a cost point of view but from overall sustainability considerations and many others.

We first compare major architectures for implementing Layer 1 in the RAN. We also compare their characteristics and the benefits of a fully virtualized, software-based L1 for the ecosystem, operator and the entire RAN industry.

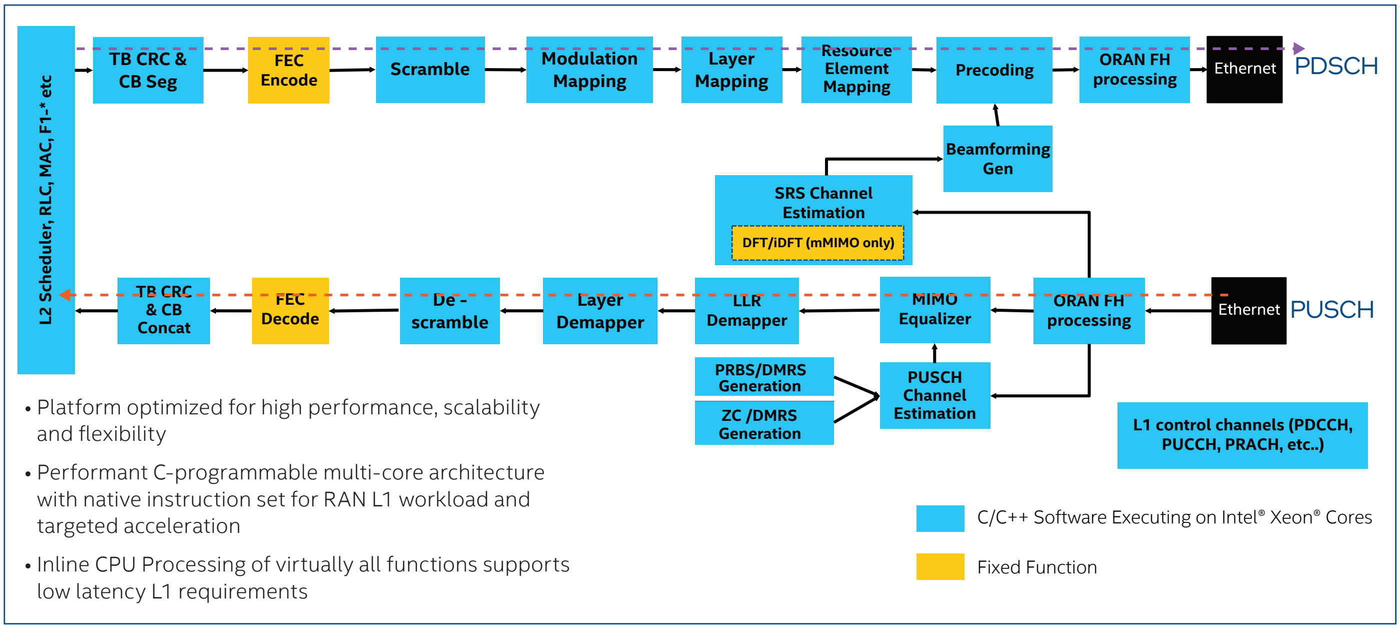
Overview of Layer 1 Implementation Approaches

In this section, we describe the two approaches to implementing the Layer 1 in a RAN.

Software-based L1 on general Purpose CPU-based architecture

Foundational to this approach is a high-performant, multi-core processor boosted with native instructions for efficient processing of certain L1 functions, such as vector processing. The Intel® Xeon® processor family includes high performance cores and instructions, such as Intel® AVX-512 and the forthcoming enhanced AVX512-FP16 signal processing instructions available in 4th Generation Intel® Xeon® Scalable processors, which are ideal for software-based L1 processing. In the 2nd and 3rd Generation Intel® Xeon® architecture, as shown in Figure 1, L1 functions for each direction are implemented as a simple pipeline with software running on CPU cores. The only exceptions to this pipeline are encode/decode functions associated with forward error correction (FEC), and discrete fourier transforms (DFT), which are performed in fixed-function accelerator blocks. The 4th generation Intel® Xeon® Scalable processor with Intel® vRAN Boost integrates the accelerators.

A key characteristic of software for this architecture is that all layers of the RAN stack are written in standard C/C++ programming language, which leverages generic compiling, debugging and build tools (for example, gcc, LLVM, gdb). The software written for one Intel Xeon CPU generation is therefore reusable in follow-on generations and easily ported to other CPUs.



- Platform optimized for high performance, scalability and flexibility
- Performant C-programmable multi-core architecture with native instruction set for RAN L1 workload and targeted acceleration
- Inline CPU Processing of virtually all functions supports low latency L1 requirements

🟦 C/C++ Software Executing on Intel® Xeon® Cores

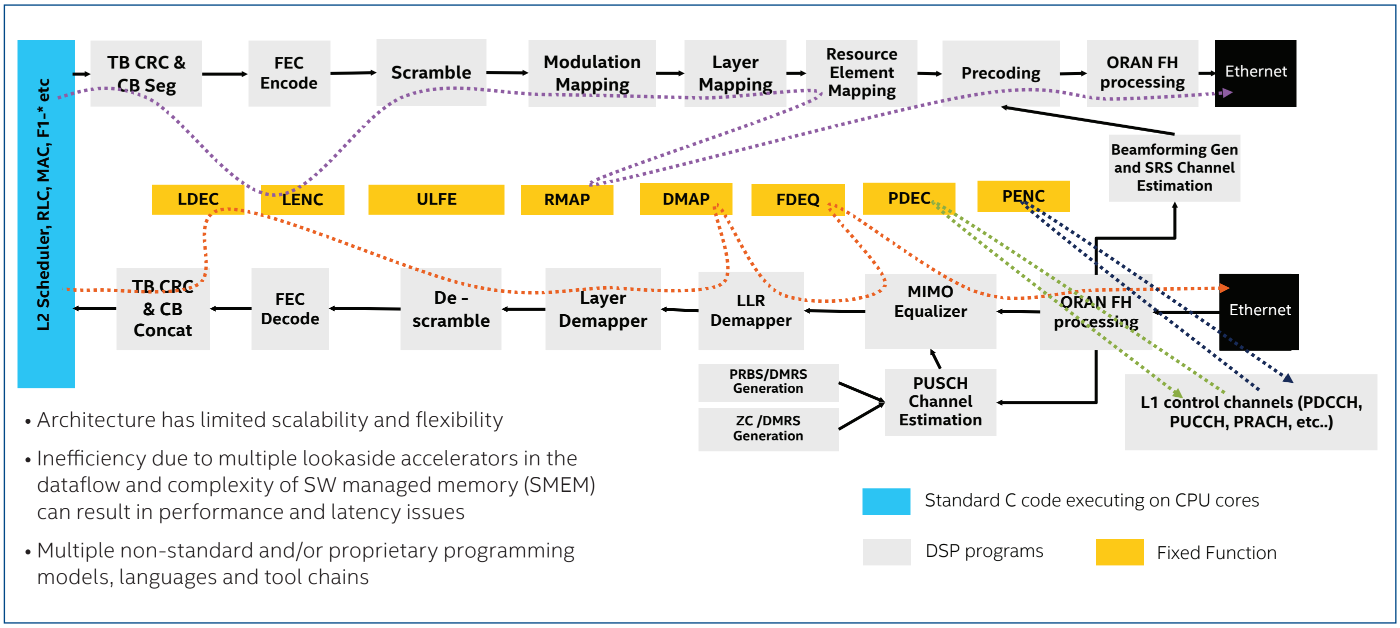
🟨 Fixed Function

Figure 1. RAN DU L1 data flow on Intel® Xeon® Scalable processor with Intel® vRAN Boost

L1 on purpose-built SoC architecture

An alternative to the software-based approach described in the previous section is to implement L1 with purpose-built SoCs. Such SoCs typically include a range of fixed-function IP blocks, each of which may be very specific to a particular L1 function and potentially include multiple types of programmable engines or versions thereof that require specialized and/or proprietary programming models, languages, tools and build environments. In purpose-built silicon, as shown in the following Figure 2, L1 is implemented as a complex mesh of software functions along with multiple hardware accelerators accessed in look-aside mode.

A key characteristic of software for this architecture is that the software-based functions of Layer 1 of the RAN stack are written (and often hand-coded) in purpose-built and/or proprietary languages suitable for the types of DSP cores embedded in the SoC. They also rely on proprietary tools for compiling, debugging and building applications. The code is usually optimized for specific versions of DSP cores and must be adapted for future generations and even for cores within the same DSP architecture. For other DSP architectures, the code must be essentially rewritten. In addition, the pace of innovation is severely limited with this architecture due to a dearth of human resources specialized in this type of software programming.



- Architecture has limited scalability and flexibility
- Inefficiency due to multiple lookaside accelerators in the dataflow and complexity of SW managed memory (SMEM) can result in performance and latency issues
- Multiple non-standard and/or proprietary programming models, languages and tool chains

🟦 Standard C code executing on CPU cores

⬜ DSP programs

🟨 Fixed Function

Figure 2. RAN DU L1 data flow in Example L1 Purpose-Built SoC

RAN L1 implementations: Implications to RAN networks

There are several material implications of building RANs with the two approaches — a fully virtualized software-based implementation running on standard CPUs and one built using a purpose-built SoC. In this section, we evaluate the relative merits of these two architectures based on side-by-side comparisons of key performance and business considerations relevant to both RAN operators and the industry as a whole. The following sections present side-by-side comparisons of characteristics of the two architectures relative to these criteria.

Vendor choice for all components (silicon, hardware, OS/virtualization, software)

Ideally, each operator should have multiple vendor choices for each system component and be able to independently select each component in the system, making tradeoffs relative to their key network deployment and evolution criteria: cost, performance, energy efficiency, flexibility, scalability, etc.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Silicon offerings with x86 architecture available from multiple ecosystem players • Standardized interfaces; Ethernet connectivity • Hardware/software disaggregation • Many vendors for hardware • Multiple software vendors for all RAN stack layers	• Silicon offerings with proprietary architectures • Non-standard software interfaces create tight L2 dependency • Purpose-built hardware/ software for L1 specific use — few vendors

Ecosystem size, availability of experienced personnel

The ability to rapidly deploy and operate networks is dependent on the availability of experienced human resources, which depends on either the ubiquity or specialized nature of the hardware platforms, reliance on proprietary or open source operating platforms and tools, and software programming models and languages, among other things.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Standard COTS servers • Standard C/C++ programming • Open source toolchains and widely used build environments for test, validation and CI/CD setups • Standard Linux OS and open source Kubernetes (K8s) for orchestration and management	• Special-purpose accelerator cards • Proprietary/hand-coded programming model • DSP toolchain is specific to each architecture and often proprietary • Programmed L1 functions execute on proprietary OS and managed with proprietary mechanisms

Velocity and cost of network evolution, e.g., reconfiguration, feature additions and changes

Operators often need to reconfigure their existing networks to support new RAN technologies. Further, they may also want to add features that enable new business models or reduce operational cost. Software upgrades are easier and have only marginal cost, while hardware upgrades involve truck rolls and can require operator ramp-up to become familiar with proprietary installation and configuration procedures.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Mostly accomplished with software updates only • Features added via software enhancements • Dynamic hardware allocation for RAN functions with software-based re-configuration — fine-grained resource control possible	• Mostly require hardware upgrades • Fixed-function IP blocks typically cannot adapt to support new functions and changes in dimensions • Fixed-function IP blocks are not designed for fine-grain control required for SLA and Slice control

Velocity and cost of network infrastructure scaling

As the number of vRAN users grows, usage patterns and technologies evolve. Operators need to be able to scale and expand coverage of their network infrastructure. Once the hardware capacity is reached, hardware upgrades are required. The two architectures have very different characteristics that directly translate to different speeds and costs for operators.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• SKUs with wide range of core count and performance/power • High fanout for scaling fronthaul I/O • Gen-over-gen software compatibility enables software reuse across hardware upgrades	• Limited SKUs with low performance range, marginal power difference • Very limited fanout for scaling fronthaul I/O • Low compatibility across DSP architectures requires recoding, even for later generations of DSPs sharing a common architecture

Hardware acceleration and cloud native deployments

Some network functions will be limited by the type of hardware acceleration that they employ. The principles applied to cloud native network functions in software must also apply to the hardware accelerators that they use. There are two fundamental requirements that must be considered:

- **Support for multiple cloud native applications:** In order to scale cloud native network functions, the hardware accelerator must be able to support multiple instances of a cloud native application in parallel.
- **Orchestration and management:** The hardware accelerator must be managed and orchestrated from the cloud infrastructure layer, such as with a Kubernetes control plane; it must support the creation and deletion of new network functions dynamically.

Hardware accelerators that cannot support these requirements can still be deployed, but they may limit the network function to being *cloud ready* as opposed to *cloud native*.

A stateless function such as FEC can be implemented as an external accelerator and still be cloud native. On the other hand, it is more difficult to apply cloud native principles for purpose-built L1 SOCs which include more complex fixed function accelerators requiring state information and interaction between software services to service requirements.

Realizing cloud native RAN benefits

Cloud-based orchestration and management have greatly benefited the IT and CSP industries. These principles can be applied to a fully virtualized RAN (including L1) to further increase flexibility, scalability and improve ease of deployment. Cloud native-compliant RAN can achieve better control of both the quality and reliability of service across the network.

In a Cloud RAN, applications are designed as microservices and the infrastructure that orchestrates these microservices use Kubernetes (K8s). As noted in the following table, the virtualized L1 offers the ability to implement end-to-end Cloud RAN.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Cloud native principles are easily adapted for all layers of the RAN • L1, L2 and L3 applications can be deployed as K8s microservices • Common cloud platform and management for entire DU (L1, L2 and L3)	• L1 must be configured and managed by other mechanisms • L1 cannot be managed as a K8s microservice • Dependency between SoC L1 and L2 software components may not allow for L2 deployment as a K8s microservice • Non-uniform ways to manage different layers of the stack

Many benefits can be realized by implementing an end-to-end Cloud RAN, including the full implementation of the Open RAN (O-RAN) architecture:

- **Service assurance / High availability:** In the event of a failure, provides the ability to spawn new instances in the current data center or shift traffic to different data centers; can be accomplished with concurrency and auto scaling.
- **Resource optimization:** Ensures optimum use of resources, with dynamic scaling and no up-front costs but with real-time automated response to scaling needs.
- **Observability:** Enables logs, metrics and tracing of the cloud native application with central control.
- **Quality of service (QoS):** Enables end-to-end security, throttling, compliancy and versioning across applications.
- **Central control plane (SMO):** Provides a central place to manage every aspect of the cloud native application.
- **Resource provisioning:** Manages resource allocations (CPU, memory, storage, network) for each application.
- **Multi-cloud support:** Provides the ability to manage and run the application across several cloud environments — including private, hybrid and public clouds — because a given application may require components and services from multiple cloud providers.
- **Advanced revenue generating and TCO optimization features:** Use cases like network slicing, pooling and power management are enabled end-to-end across the entire network using cloud orchestrated networking to realize the full benefits; these use cases are discussed in some detail further below.

Network Slicing

The *O-RAN Use Cases Detailed Specification*, published by O-RAN ALLIANCE, defines several use cases that require a cloud ready or cloud native deployment.

O-RAN cloud ready use case example 1: Multi-vendor slices

This use case enables multiple slices with functions provided from multiple vendors. In the example highlighted in Figure 3, slice #1 is composed with a distributed unit (DU) and centralized unit (CU) provided from vendor A, and slice #2 is composed with DU and CU provided from vendor B. Without the ability to share resources between network functions, this multi-vendor slice use case would not be possible. Each vendor's network functions could still be a monolithic software entity and not require cloud native microservice architecture. However, it still benefits from hardware and software disaggregation, hardware pooling, zero touch provisioning and a common standard orchestration and management layer.

Network Slice Subnet Instance (NSSI) resource allocation optimization requires a cloud native approach to open distributed unit (O-DU) and open central unit (O-CU) deployments as this use case requires scaling of the cloud native network function and zero touch automation.

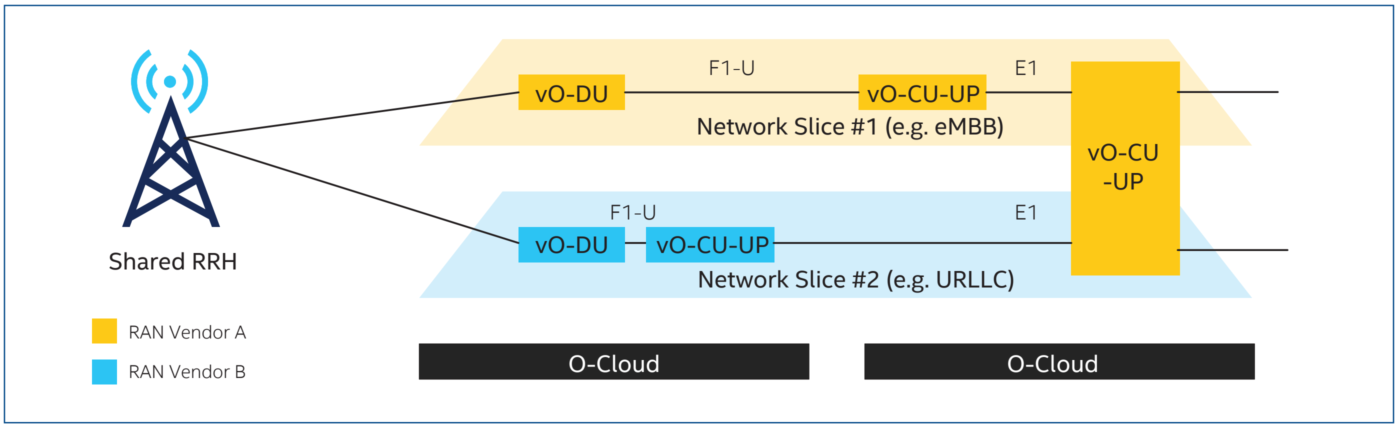


Figure 3. *O-RAN Multi-vendor Network Slice* sharing O-Cloud compute resources

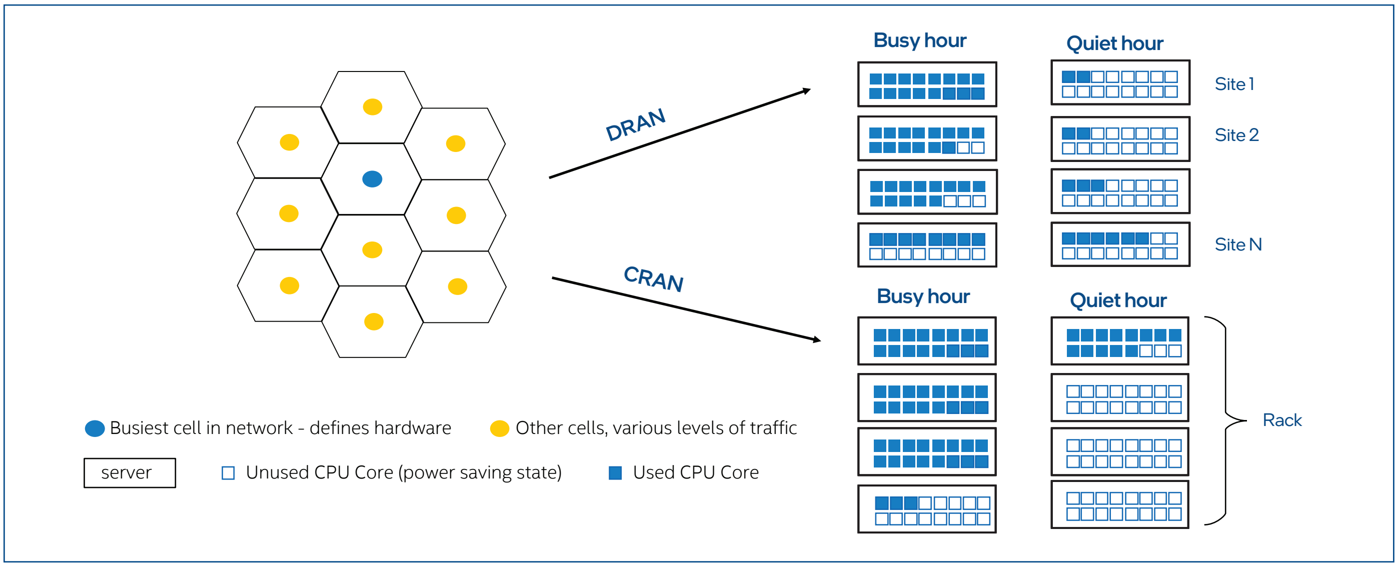

DRAN
CRAN
Busy hour
Quiet hour
Site 1
Site 2
Site N
Busy hour
Quiet hour
Rack
Busiest cell in network - defines hardware
Other cells, various levels of traffic
server
Unused CPU Core (power saving state)
Used CPU Core

Figure 4. Example Resource Profile with pooling within a server and across a rack of servers

Pooling-enabled optimized infrastructure, network availability and resiliency

Network infrastructure is typically dimensioned for peak usage. However, the number of users and usage patterns can vary dramatically over a given 24-hour period. Typically, the number of users is much higher during the day than at night. Usages (video, calls, text, gaming, etc.) also change over the course of the day. Some of these variations are predictable and some are not. Also, the horsepower requirements for workloads can vary considerably depending on the traffic patterns. For example, the L1 workload is heavy during periods when users consume high bandwidth. The L2 workload depends on the number of simultaneous users as well. There are additional functions in a DU, such as statistics collection and processing, AI inferencing models and others that need to be run from time to time and have flexibility in when they can be executed.

Pooling enables dynamic allocation of the right number of cores to each workload; the hardware resources may be within a server blade or across server blades. Pooling benefits are maximized when the same type of cores (for example, general purpose CPUs) are used for all functions: L1, L2 and additional DU functions.

Pooling can be implemented within a server, pooling the CPU cores, memory and I/O for multiple cells and also implemented within a rack of servers. These are not mutually exclusive and can be leveraged concurrently to improve power consumption depending on the deployment scenario. Distributed RAN (DRAN) can achieve use cases where there is a single server pooling of resources. For DRAN deployments with more than one server and for centralized RAN (CRAN) deployments, pooling can be achieved within the server and within the rack of servers. As the compute resources scale depending on the user load, power benchmarking should take this into account and provide a 24-hour power consumption rather than peak or low power consumption. This is depicted in Figure 5.

O-RAN ALLIANCE also defines a Baseband Unit (BBU) Pooling use case to achieve RAN Elasticity and presents three different classes for pooling: Class 0, Class 1 and Class 2. In Class 0 pooling, the remote radio head (RRH) is directly assigned to a single DU during run time.[1] Pooling is achieved through centralization of multiple cells to a single DU instance running in the cloud platform. Class 1 and 2 are more aggressive types of pooling and allow a looser coupling of the RRH to the DU network function. In both these cases, the RRH is a fixed physical network function and cannot be changed. Therefore, it is the DU network function that must support cloud native design principles such as scaling, zero touch automation, orchestration and management, automation and more.

Pooling can deliver key cost and power benefits toward achieving resiliency and high availability:

- Minimizes total number of deployed hardware resources (CPU cores and servers).
- Enables regular maintenance to be conducted without interrupting traffic.
- Makes it possible to monitor server health and do preemptive maintenance by moving traffic to other servers.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Software-based control of number of processing cores/ servers relative to cells/UEs • Since all workloads (L1, L2, L3) run on the same processor, additional pooling benefits can be realized across workloads	• Fixed-function IP blocks and DSP cores in a single SoC architecture do not offer much flexibility to fluidly move L1 workload among all resources, and hence are not as amenable to pooling implementation • Within the L1 SOC, pooling gains do not extend beyond the L1 workload.

As a simple example, a benchmark was executed using Intel FlexRAN™ Reference software where multiple 5G NR cells were executing in BBU Pooled scenario and the number of cores and power consumption measured. Figure 5 shows that as the number of active cells increases, optimal core deployment can be achieved via pooling techniques in a fully virtualized RAN executing on 3rd Generation Intel Xeon processors, as shown for an example use case below.

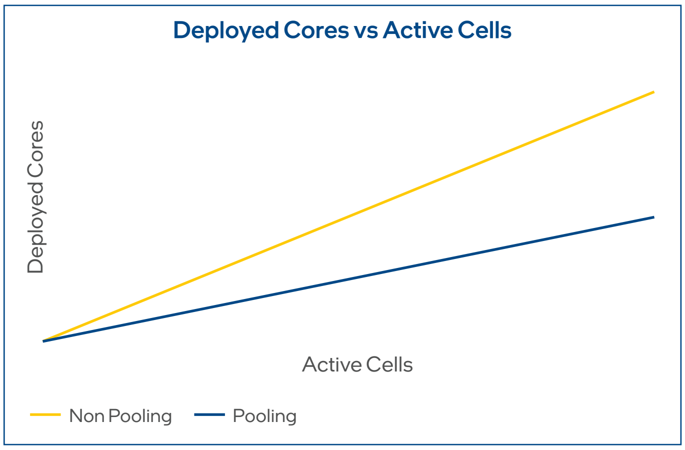


Figure 5. Pooling enables fewer cores to support required capacity

Figure 6 shows the associated reduction in power achieved due to pooling.

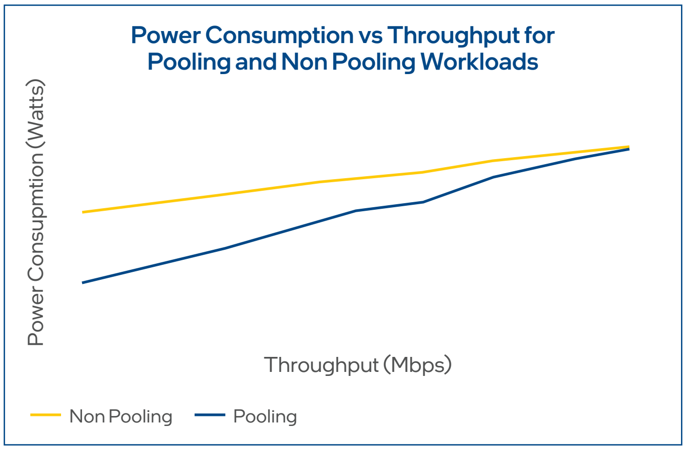


Figure 6. Pooling results in lower power to support required capacity

Energy efficiency and sustainability

Energy expenditures are a significant component of RAN operational costs, and network sustainability is an area of focus for the entire RAN ecosystem.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Telemetry information for monitoring CPU resource utilization • Ability to control frequency and power of silicon resources (cores) • Ability to put cores in a sleep state during periods of low user activity	• Fixed-function IP blocks cannot be controlled for reducing power during periods of low traffic • DSP cores typically do not support the fine-grain power state control (e.g, micro-sleep and hibernate) similar to those available in CPUs like the Intel Xeon Processors.

Energy efficiency is another advantage of cloud native network functions executing on Intel x86 CPUs where they can leverage the existing investment into IT domain energy efficiency solutions. There are three different categories for energy efficiency in the IT domain that can be applied to cloud native RAN solutions, and these are:

1. **Application-level power control**
   Application-level power control allows the application to take advantage of the C-states on the 3rd and 4th Generation Intel® Xeon® Scalable CPUs with reduced latency transitions. The application can sleep the threads when there is no work to do and the CPU will enter C-states to save power.
2. **Server Level Power Control**
   Server level power control allows a privileged server executing on the cloud to change the platform power controls, such as P-states, which allows a per-core CPU frequency scaling. The cloud platform can reduce power consumption or tailor the CPU core frequency for the network function.

3. **Cluster Level Power Control**
   When running in a rack of servers or in a cluster that supports cloud native RAN network functions, it is possible to scale and consolidate the workload onto fewer servers and power off unused servers to save energy. Without cloud native network functions for RAN, cluster-level power control will not be possible.

As one example, Deutsche Telekom used its own network traffic profiles to demonstrate approximately 30% energy savings in L1 processing in a Deutsche Telekom laboratory setup. Key aspects of the demo, which was presented at Mobile World Congress, 2022 in Barcelona, include:

- Virtualized L1 implemented in Intel FlexRAN software running on Intel 3rd generation Xeon processor.
- Telemetry capabilities and dynamic power controls available in the Xeon processors used to turn on/off cores as needed with varying traffic conditions.
- Intelligent L2 MAC scheduler in Intel FlexRAN software detected periods of low traffic activity and scheduled fewer cores per radio frame.
- Unused cores transitioned to a deep power down state with savings of multiple W per core.

Additional power savings can be achieved via pooling technologies applied in DUs based on virtualized L1. System-wide power savings can be realized by extending this approach of combining monitoring telemetry information and applying CPU power controls to other stack layers and base station functions, for example, L2, transport, control and OAM.

Additional details of the use cases and results are presented in a video that can be viewed at: https://www.intel.com/content/www/us/en/events/mobile-world-congress.html

As another example, at the O-RAN Spring Plugfest held in June and July 2022, Intel collaborated with Vodafone, Wind River, Radisys and Keysight to demonstrate up to 12% power savings with the Vodafone network traffic profile using a limited application of the power management capabilities in a RAN DU+CU based on the 3rd Generation Intel Xeon processor, without any application changes.

In both demonstrations, traffic throughput was not impaired while achieving the energy efficiency gains.

Enabling the Intelligent RAN

Service providers are increasingly looking to build intelligence into the network to maximize their return on investment (ROI) and minimize operational costs.

This is also the goal of the O-RAN Alliance and the RAN Intelligent Controller (RIC) architecture. The Non-Real time and Near-Real time RAN Intelligent Controllers (RIC) allows for combining platform telemetry through O2 interface with application telemetry through the O1 and E2 interfaces. xApps and rApps executing in the Near-RT RIC and Non-RT RIC respectively leverage Artificial Intelligence/Machine Learning (AI/ML) technologies to implement a variety of use cases, some examples below:

- Maximize spectral efficiency and use of deployed radio resources with active antenna management and orchestration
- Enhance real-time response, for example, dynamically adjust resource allocation, as network conditions change
- Closed loop automation for best configuration settings or configuration to meet the given workload requirement with minimal resources
- Faster new user provisioning
- Predictive analytics
- Network anomaly detection and intervention at scale
- Security anomalies and intrusion detection, and automated application of corrections
- Automated cell deployment

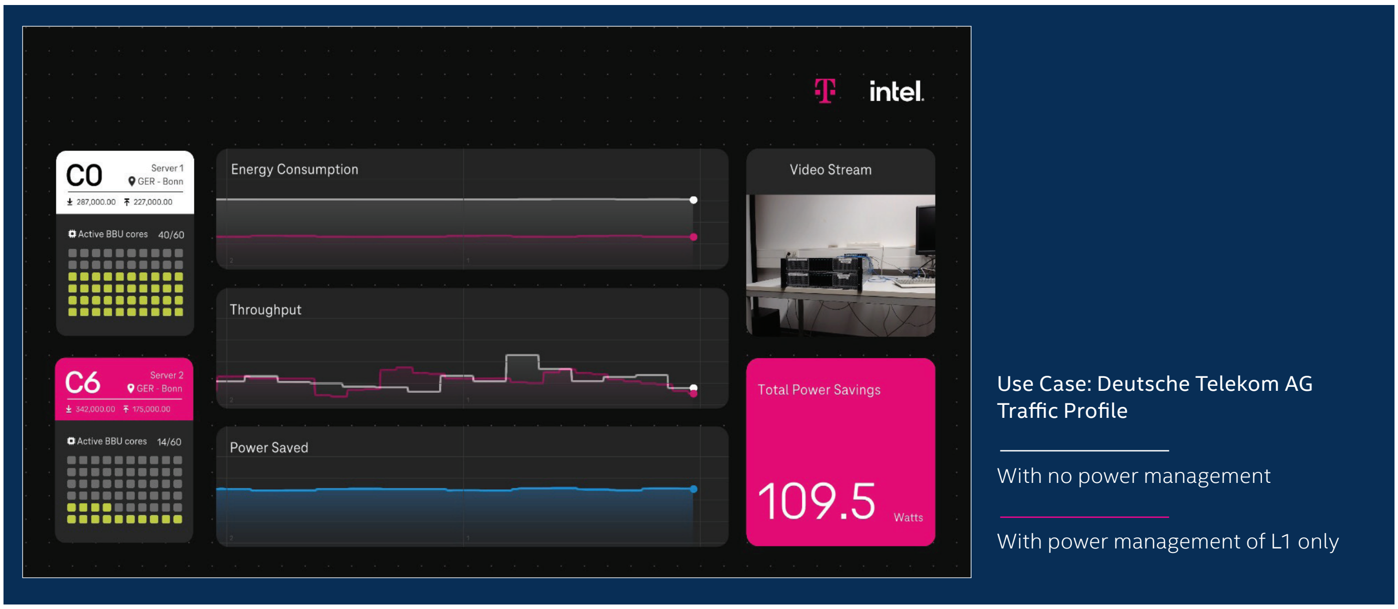


Figure 7. Energy efficiency gains with power management of L1 application in a RAN DU

Application telemetry for many of these use cases include dynamic resource and network performance information from the RAN stack, including L1. Key performance indicators (KPMs) and resource information may be at the antenna-level, cell-level and user-level. Beam level information and physical resource block (PRB) utilization are some examples of information that the L1 functions can harvest for use by xApps and rApps.

Moreover, since the use cases, xApps, and rApps continue to evolve, we simply do not know all the data that needs to be collected from the L1. Consequently, it is not possible to pro-actively build all needed capabilities in the L1 implementation today, and L1 implementations in provisioned networks must be able to evolve.

Virtualized L1 on CPUs with open architecture, e.g., Intel Architecture	L1 implemented on purpose-built SoC
• Actionable, real-time, granular platform telemetry information for monitoring CPU resource utilization, power consumption, fault detection, and performance • L1 software can be readily modified to report dynamic network KPMs which can be customized for each xApp and rApp • As xApps and rApps evolve, L1 application can be evolved to match the needs, avoiding expensive hardware replacement	• Heterogenous functional blocks makes it difficult for Silicon to report actionable platform telemetry information and in a timely fashion • Fixed function or configurable IP blocks cannot be modified to provide information needed. • Since only some of the functions are in software running on DSP cores, the flexibility needed to collect all such information needed by xApps & rApps is not available. Memory is often a constraint in purpose built SoC's restricting the amount, type and frequency of data that can be exported • New requirements by evolving xApps and rApps cannot be accommodated in deployed purpose-built SOCs and will often require expensive Silicon revisions and hardware upgrades.

Several ecosystem players are developing and demonstrating the value of such applications.

One example of automated deployment with telemetry abstraction from the DU to the Cloud services is from demo of Microsoft for Telecommunications at MWC 2022 Hybrid cloud platform for operators – Azure Operator Distributed Services. This demonstrates the automated deployment of DU and CU services and also demonstrated how User Equipment (UE) telemetry can be exported to review and help debug parameters like per user Signal Interference to Noise Ratio (SINR).

Another example is recent MWC demo with an AI powered vRAN application (rAPP) for massive MIMO beam management that can optimize the quality of user experience (QOE) and 5G coverage dynamically as user traffic changes throughout the day. As a standard C/C++ sw application and running on COTS hardware the Intel FlexRAN L1 SW was modified to extract and export the beamforming counters and data to enable the rApp.

A video of this demonstration is available at: Intel at Mobile World Congress 5G vRAN & AI powered RIC Applications on Xeon.

Conclusion

Full virtualization of the RAN offers many technical and business benefits. Software-based RAN can be made more agile, leading to flexible network deployments that are amenable to dynamic orchestration and software-defined coordination. In combination with the centralization of network architecture, dynamic resource pooling and load balancing can be used to improve network resource utilization, gain energy efficiencies and achieve higher resiliency. vRAN is also more agile and scalable to meet network capacity requirements based on demand, compared to a network infrastructure based on fixed-function devices. Failures in network operation can be handled without truck rolls (zero touch) by merely moving the network workload to a different server, resulting in lower operational expenses (OpEx). The flexibility of software-based RAN also enables rapid, simple evolution of the infrastructure to optimize network resources, reduce TCO and support new capabilities, such as network slicing, to enable new revenue-generating business models for operators at optimal cost. Further, a fully virtualized architecture enables implementation of innovations such as machine learning algorithms across all functions in the RAN, including L1, evolving the RAN network to deliver ever-more capabilities at optimized cost.

[1] See O-RAN.WG1.Use-Cases-Analysis-Report-v08.00.pdf available at https://orandownloadsweb.azurewebsites.net/specifications