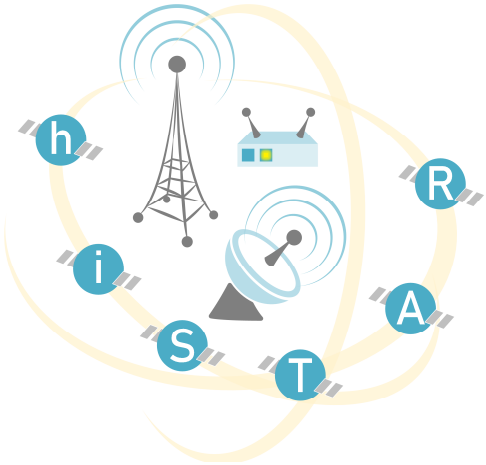
Science Fund
of the Republic of Serbia

Grant Agreement No.: 7750824

Call: IDEJE

Hybrid Integrated Satellite and Terrestrial Access Network



D3.1: Unified SDR framework for DVB-S2X and 5G modems

Work package	WP 3
Subactivity	T3.1
Due date	1/9/2022
Submission date	30/09/2022
Deliverable lead	ETF
Version	1.0
Authors	Dragomir El Mezeni
Reviewers	Vladimir Petrović, Lazar Saranovac

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	30/09/2022	1st version of D3.1	Dragomir El Mezeni

COPYRIGHT NOTICE

ACKNOWLEDGMENT

Science Fund of the Republic of Serbia

This deliverable has been written in the context of hi-STAR project who has received funding from the Science Fund of the Republic of Serbia, Programme IDEJE under grant agreement n° 7750284.

EXECUTIVE SUMMARY

The hi-STAR project addresses one of the most critical challenges for the next generation wireless networks, which is integration of non-terrestrial networks with terrestrial 5G network. The general objective of the project is to develop flexible framework for integrated terrestrial 5G and Low-Earth-Orbit (LEO) satellite networks, where traffic management is performed with assistance of newly developed artificial intelligence methods.

This deliverable is a result of the work done in the context of WP3 Subtask T3.1 – Analysis of existing SDR frameworks for DVB-S2X and 5G. Deliverable D3.1 presents description of physical layer architecture for DVB-S2X and 5G modems. Deliverable D3.2 will present developed FPGA hardware accelerators for both physical layers.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

HUT	**Hybrid User Terminal**
OAI	**Open Air Interface**
SDR	**Software defined radio**
WP	**Work Package**

SECTION 1 - INTRODUCTION

This deliverable is a result of the work done in the context of WP3 Subtask T3.1 – Analysis of existing SDR frameworks for DVB-S2X and 5G. Deliverable D3.1 presents description of physical layer architecture for DVB-S2X and 5G modems. Deliverable D3.2 will present developed FPGA hardware accelerators for both physical layers.

This deliverable is structured as follows: In Section 2 SDR solution for DVB-S2X is presented. It is based on GNU Radio framework. Architecture is developed in order to best use multi-core architecture of modern CPUs considering also that critical blocks will be offloaded to hardware. In Section 3 SDR solution for 5G is presented. It is based on Open Air Interface. Considering control complexity involved in OAI and different, flexible concept of GNU Radio it is decided to separately accelerate those two modems. HUT will have a set of APIs that will enable access to both DVB-S2X and 5G physical channels, acting as one monolithic unified framework. Architecture of such framework is described in Section 4.

SECTION 2 – DVB-S2X SDR SOLUTIONS

For DVB-S2X GNU radio is selected as most appropriate SDR solution. Since increased flexibility usually induces performance degradations, it is challenging to achieve real-time performance with GNU Radio on GPP. Nevertheless, there are many advantages in using such general framework when developing and researching new applications, ranging from community support to easier debugging and reconfiguration of existing designs. GNU Radio supports already developed physical layers of DVB-S2X receiver and transmitter chains. In this section architecture and acceleration of DVB-S2X transmitter is analyzed. However, given conclusions are general and will be similar for the receiver implementation.

There are dedicated benchmarks developed to enable easier selection of appropriate processor that can achieve the best GNU Radio performance. Another approach is to optimize the processing itself to better use available hardware resources. We have analyzed GNU Radio framework and overheads it introduces. In GNU Radio framework each block is implemented in the separate processing thread. Dedicated scheduler is responsible for deploying these threads to the actual processing cores. However, even though the work is split to multiple threads, the parallelization is usually not that much efficient as synchronization is needed between different processing blocks.

Bloessl et al [1] analyzed GNU Radio throughput for a large number of simple processing blocks. They concluded that throughput scales linearly with increasing the number of blocks. However, they also found that significant improvements can be achieved by manual processing organization, thus avoiding framework synchronization and scheduling mechanisms. Becker et al [2] used GNU Radio for real-time wireless signal classification. They found that by using alternative synchronization mechanisms, based on Qt Signals and Slots, performance improvement of up to 78× can be achieved. They also used vector instructions to accelerate separate processing blocks.

In order to efficiently use multiple processing cores, sufficient level of parallelism should be exposed. Recently, Grayver et al [3] presented DVB-S2 demodulator with 4GHz bandwidth using multiple GPP servers. To efficiently use multiple-cores they used multiple data-path processing chains to extract parallelism. Miller also used this way of processing parallelization to develop HDR QPSK modem using GNU Radio [4].

In this deliverable we demonstrate mechanisms how to achieve the HDR performance on a multi-core processor on the GNU Radio DVB-S2X transmitter use case, and analyze the impact of dedicated hardware accelerators on throughput and CPU utilization. First we give a brief overview of GNU radio framework and its synchronization and scheduling mechanisms. Next we describe implementation of accelerated DVB-S2X transmitter architecture. And finally we evaluate performance of fully software and hardware accelerated architectures.

2.1. GNU RADIO FRAMEWORK

GNU Radio represents a framework that can be used for development of different SDR applications. Beside library of already designed DSP blocks with standardized interfaces framework also supports development of custom DSP blocks. These blocks are called out of tree (OOT) modules and can be used to implement new functionalities, which increases the framework flexibility.

Signal processing blocks chain in GNU Radio is called “flowgraph”. Each flowgraph contains at least one source and one sink block. Data flows from source to sink through all DSP blocks in the chain. Every DSP block has input and output buffers and is implemented in a separate processing thread. GNU Radio uses operating system scheduler for managing thread executions. By using affinity parameter of DSP block, thread can be statically scheduled to the specific CPU core. The thread is scheduled for execution if there is enough data in the input and enough space in the output buffer. Otherwise, processing is stalled and neighboring blocks are notified via message passing mechanism. Hence, each block in a chain waits for previous block to fill its input buffer in order to start processing. Consequently, the entire chain throughput is determined by the slowest block.

Since each block is implemented in a separate thread, flowgraph can be scheduled on a multi-core processor. On the other hand, such low granularity can lead to frequent context switching and increase the processing overhead. One recommendation for optimization is to combine several functionalities in one larger block [5]. Efficient use of multiple cores assumes that sufficient number of threads can be executed independently.

2.2. ACCELERATION OF DVB-S2X TRANSMITTER

Flowgraph for DVB-S2X transmitter is shown in Fig. 1. The transmitter is implemented using GNU Radio framework v3.8 on Ubuntu 20.04 using AMD Ryzen 9 5950X with 32 processing cores. Maximal throughput that can be achieved for a single transmitter chain is 131 Mbps with processor utilization of just 160%. This is equivalent to 2 CPU cores, which means that GNU Radio scheduler cannot efficiently utilize multi-core processor when single serial flowgraph is used. The reason for this behavior is that many simple tasks will be blocked while waiting for more complex tasks to be completed. In the DVB-S2X transmitter case, these complex tasks are forward error correction (FEC) encoders, LDPC and BCH [8]. They take almost 70% of total processor utilization. Also, the flowgraph from Fig. 1. is relatively small, having only 6 processing blocks, and scheduler is not able to efficiently use the remaining CPU cores.

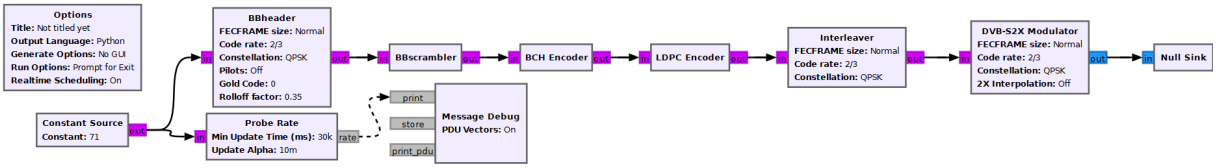


Figure 1: DVB-S2X transmitter flowgraph

In order to increase the CPU utilization and throughput several parallel processing branches can be used [5]. Each parallel branch is processing a different codeword. Interleave and deinterleave blocks from GNU Radio library are used for transferring codewords from serial stream at the input to parallel branches and to combine results into a single serial stream at the output. Parallel architecture of DVB-S2X transmitter for the case of four branches is presented in Fig. 2.

Since blocks from different branches are working independently, they can be scheduled on different processing cores more efficiently. The goal of the first experiment in this study was to find an optimal number of parallel branches for DVB-S2X transmitter in the case when different numbers of CPU cores were available.

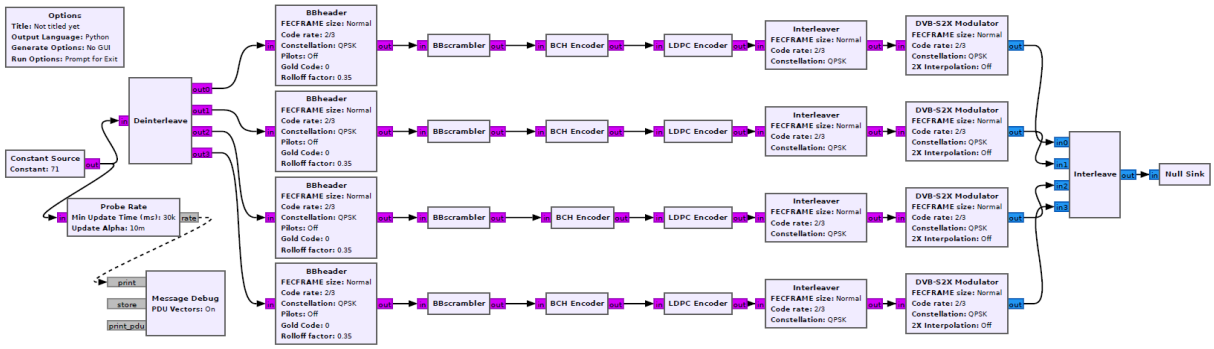


Figure 2: Parallel DVB-S2X transmitter flowgraph

Even though the throughput can be significantly increased by using parallel branches, the most complex blocks like FEC encoders still represent a bottleneck for each branch. To overcome this, the transmitter can be further accelerated by offloading these blocks to dedicated hardware accelerators, eg. PCIe FPGA cards or GPU cards. If accelerator is fast enough, only one can be used to process data from all parallel branches. Interleave and detinterlevae blocks again can be used between parallel and serial stream conversions. These components also emulate data copying to and from the accelerator. If accelerator processes stream of data with the same throughput as the rest of software chain, then it can be modeled as a simple connection between interleave and deinterleave components. Architecture of DVB-S2X transmitter with accelerated FEC blocks is shown in Fig. 3.

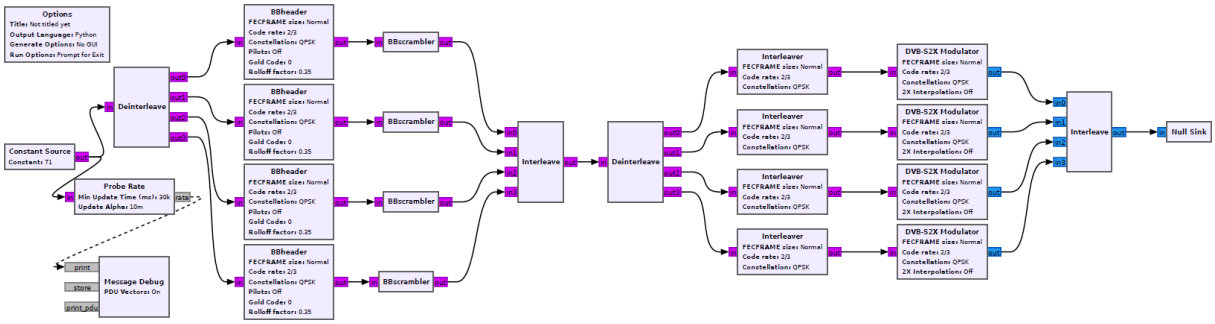


Figure 3: DVB-S2X transmitter with emulation of accelerated FEC blocks

In order to evaluate performance of the examined architectures CPU utilization and achieved transmitter throughput were measured. CPU utilization was measured by using htop tool and it is expressed in percent ranging from 0 to 3200 in case of 32 cores. Number of cores available to GNU Radio application can be limited by using taskset command in Linux. Throughput is measured directly from GNU Radio by using Probe Rate block. Running average gain was set to 0.01 and each measurement was taken by averaging results obtained from different runs of GNU Radio applications. This way the influence of different states of a cache and different states of operating system can be reduced. GNU Radio scheduler was set to Normal priority, and VOLK acceleration is not used.

Throughput and CPU utilization for parallel transmitter architectures with different number of parallel branches, with different number of available CPU cores, are shown in Fig. 4 and Fig. 5 respectively.

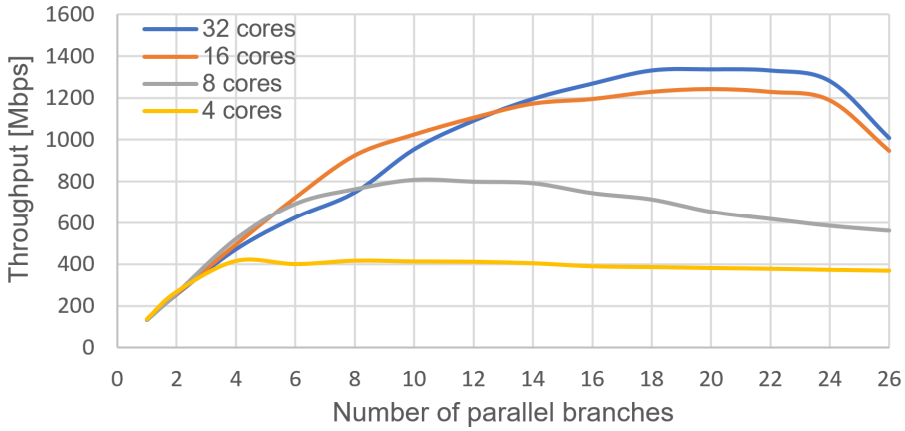


Figure 4: Throughput for parallel DVB-S2X transmitter

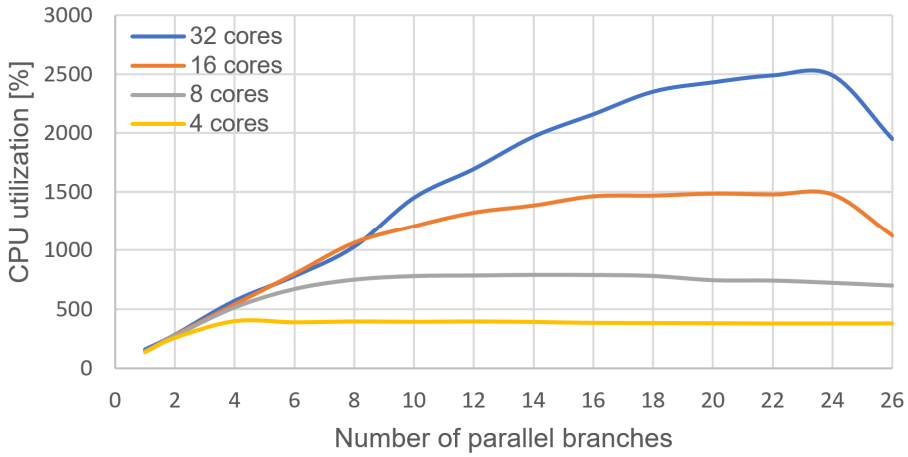


Figure 5: CPU utilization for parallel DVB-S2X transmitter

Maximal throughput for parallel DVB-S2X transmitter was 1.3 Gbps and it is achieved for 20 parallel branches in case when all 32 processing cores were available. However, slightly lower throughput of 1.2 Gbps can be achieved for the same number of parallel branches by using just 16 processor cores. Hence, for this architecture it is worth considering the usage of simpler processor or freeing processor resources for some other processing tasks.

Increasing number of parallel branches exposes more parallelism and enables higher throughput. However, when processing load exceeds available resources CPU utilization enters saturation. In the case of 4 CPU cores, utilization reaches 400% for 4 parallel branches and throughput increase stops. When there are 8 CPU cores available, CPU utilization reaches 785% for 10 parallel branches and stays in this high utilization region. For 16 and 32 CPU cores, CPU utilization saturates before CPU reaching maximal values of 1600% and 3200% and even starts to drop when number of parallel branches is further increased. for this behavior is a new bottleneck introduced in parallel architectures. Namely, although there are parallel branches enabling larger throughput, Interleave and deinterleave components still operate in serial manner. Their load increases with throughput and they become bottlenecks when more than 24 parallel branches are used. Interleave and deinterleave components can be implemented as simple data switches, and thus their complexity should be minimal. In software they just need to forward data pointers for current codeword to the appropriate processing branch. However, in GNU Radio, these blocks are copying all data elements from memory to component buffers. Copy operation takes significant amount of time especially in case with large number of parallel branches. Hence, implementing OOT module that work with pointers instead of using library components can potentially increase throughput.

For smaller number of parallel branches lower number of processing cores can provide higher throughput. For example, in the case of 6 parallel branches, throughput of 692 Mbps can be achieved with 8 processing cores while it drops to 626 Mbps when all 32 cores are used. This can

be explained by inefficiency of GNU Radio scheduler when number of cores largely exceeds processing load. In this case, scheduler will more frequently switch processing tasks from one core to another, introducing additional overhead and thus decreasing the throughput. This can be confirmed by statically scheduling processing blocks using affinity parameter of GNU Radio blocks. In this case increasing number of cores will not decrease the throughput. However, throughput achieved with static scheduling is lower than by dynamic scheduling mechanism.

Throughput and CPU utilization for accelerated DVB S2X transmitter architecture are shown in Fig. 6 and Fig. 7 respectively.

Maximal throughput for accelerated DVB-S2X transmitter was 3.4 Gbps and it is achieved for 8 parallel branches in a case when just 8 processing cores were used. Furthermore, the achieved throughput for 8 cores is larger than for any other number of cores for all configurations of parallel branches. This result is very interesting and shows that there exists optimal architecture for a specific flowgraph. In accelerated DVB-S2X transmitter there are 4 simple blocks in parallel branches and 4 serial blocks (interleave and deinterleave). This architecture can optimally fit 8 cores when serial tasks are implemented in the separate cores while parallel blocks share the remaining 4 cores. Since number of cores is limited there will be less thread and context switching. By examining CPU utilization, we can notice that in the case of 8 cores, it saturates to lower level than in the case of DVB-S2X transmitter without acceleration.

When only 4 CPU cores are available, the utilization reaches 379% for 4 parallel branches and maximal achieved throughput is 1.8 Gbps. This is still higher than maximal throughput achieved for all-software transmitter with 32 cores. For 16 CPU cores maximal achieved throughput is 2.9 Gbps for 6 parallel branches, while throughput of 2.2 Gbps is achieved for 8 parallel branches and 32 CPU cores. Further increase of parallelism leads to lower throughput since interleave and deinterleave components

By accelerating the most complex blocks, the rest of the flowgraph becomes more balanced and can be efficiently scheduled. The goal of the second experiment was to find the number of CPU cores needed to achieve the maximal throughput of the transmitter with FEC acceleration.

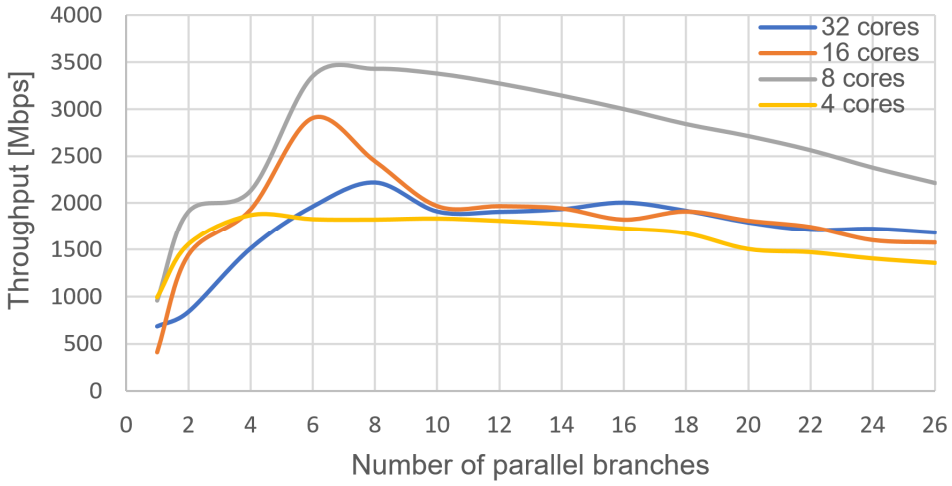


Figure 6: Throughput for accelerated DVB-S2X transmitter

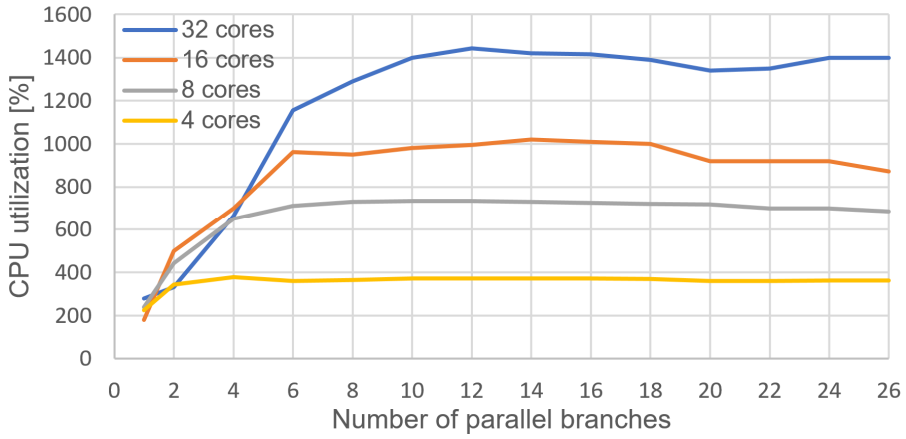


Figure 7: CPU utilization for accelerated DVB-S2X transmitter

In this deliverable we examined the influence of varying number of parallel branches and number of available CPU cores to the performance of DVB-S2X transmitter with and without hardware acceleration.

We found that in case of fully software implementation, transmitter achieves throughput of 1.2 Gbps for 16 CPU cores and 20 parallel branches. This is just a slight penalty when compared to the case when all 32 cores were used. Hence, there is a limit of number of parallel branches and processing cores for this architecture.

When FEC modules are accelerated in hardware, aforementioned effects are much stronger since the remaining flowgraph is more balanced. Maximal throughput of 3.4 Gbps is achieved for just 8 parallel branches and 8 processing cores. Accelerated architecture provides larger throughput for 4 cores than software architecture with all 32 cores.

Accelerating DVB-S2X transmitter by implementing FEC encoders in hardware can significantly increase a throughput and enable usage of less powerful processors which are usually available in SoC solutions.

Architectural recommendation derived from the example of DVB-S2X transmitter are general and similar conclusions will hold for the case of DVB-S2X receiver.

Next deliverable D3.2 will provide accelerators for critical functionalities and software architecture will be organized based on the previous recommendations. This deliverable will also include software implementation of missing processing blocks that will complete standalone physical layer implementation of DVB-S2X transmitter and receivers.

References:

[1] B. Bloessl, M. Müller and M. Hollick. "Benchmarking and Profiling the GNURadio Scheduler." In Proceedings of the GNU Radio Conference, vol. 4, no. 1. 2019.

[2] C. Becker, A. Baset, S. Kasera, K. Derr and S. Ramirez, "Experiences with using GNU Radio for real-time wireless signal classification." In Proceedings of the GNU Radio Conference, vol. 3, no. 1. 2018.

[3] E. Grayver and A. Utter. "Extreme Software Defined Radio–GHz in Real Time." In 2020 IEEE Aerospace Conference, 2020, pp. 1-10.

[4] D. Miller, "Demonstration of GNU Radio High Data Rate QPSK Modem at 15.0 Mbps Real-Time with Multi-Core General Purpose Processor", In Proceedings of the GNU Radio Conference, 2022.

[5] T. W. Rondeau, O. Holland, H. Bogucka, and A. Medeisis. "On the GNU radio ecosystem." Opportunistic Spectrum Sharing and White Space Access: The Practical Reality, 2015, pp. 25-48.

[6] ETSI, “Digital Video Broadcasting (DVB)”, https://www.dvb.org/standards/dvb-s2

SECTION 3 – 5G SDR SOLUTIONS

For 5G NR only Open Air Interface supports limited implementation of standard communication stack. This framework provides implementation for UE, gNB and core network. For this deliverable implementation of UE is of interest. Open Air Interface is developed by several project groups: 5G RAN, 5G Core network and MOSAIC5G. Focus of this deliverable is on 5G Radio access network and on standalone UE implementation.

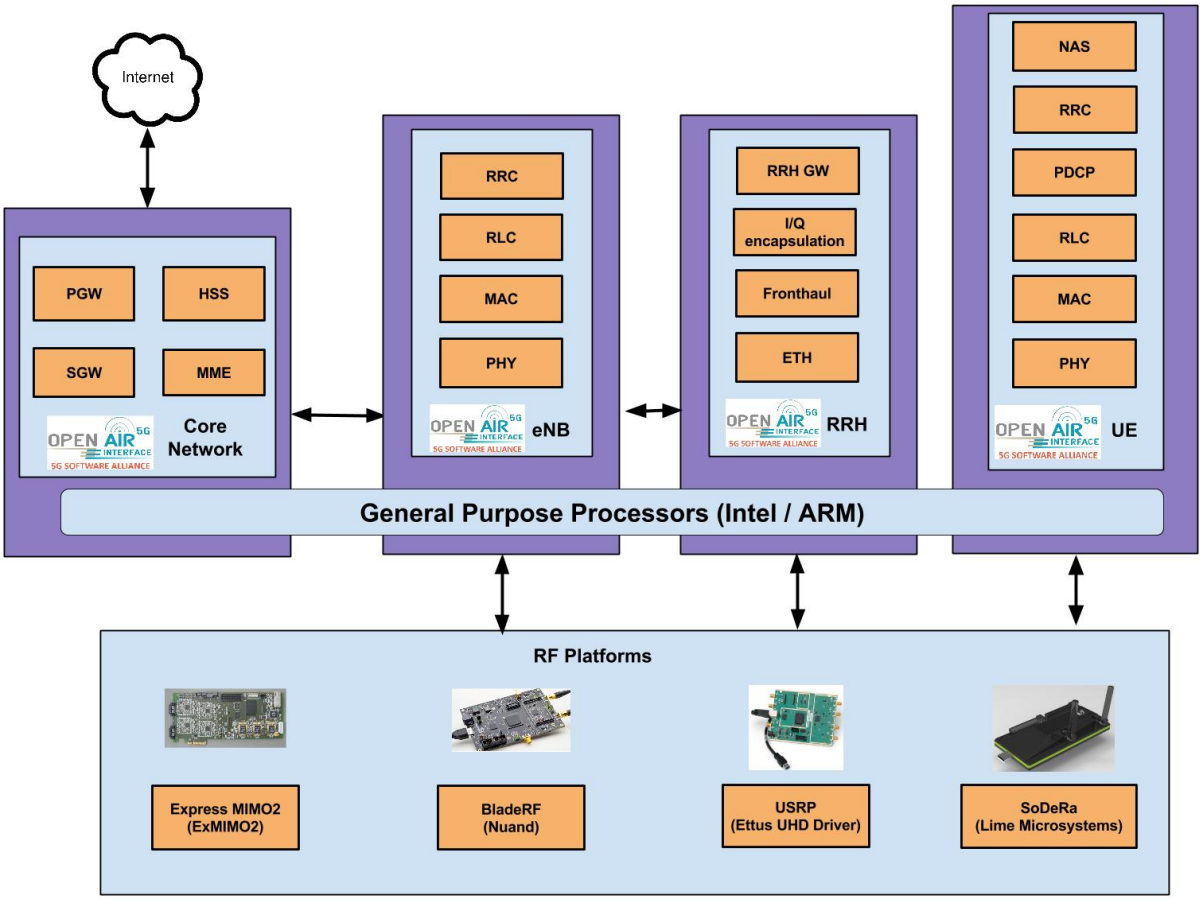


Figure 8: Open Air Interface UE demonstrator - source: openairinteface.org

OAI 5G RAN project supports L1-simulation framework that provides channel models that can be incorporated together with satellite channel models as described in the next section. OAI will be used for UE physical layer implementation on RF-SoC development board.

The main issue with Open Air Interface is that it is optimized solution and it is very difficult to adapt it to some custom use case. Also since data-link layer is technology dependent it cannot be easily separated from physical layer in order to use it in the unified solution. On the other hand, OAI can be used in real-time with user signals.

Next deliverable D3.2 will provide hardware acceleration of LDPC decoder that can significantly increase throughput of OAI. Other authors showed that accelerated OAI is able to achieve throughput in the gigabit range using general purpose processors [1].

References:

[1] E. A. Papatheofanous, D. Reisis and K. Nikitopoulos, "LDPC Hardware Acceleration in 5G Open Radio Access Network Platforms," in IEEE Access, vol. 9, pp. 152960-152971, 2021, doi: 10.1109/ACCESS.2021.3127039.

SECTION 4 – UNIFIED FRAMEWORK

Architecture of Hybrid User Terminal proof of concept is presented in Fig 9.

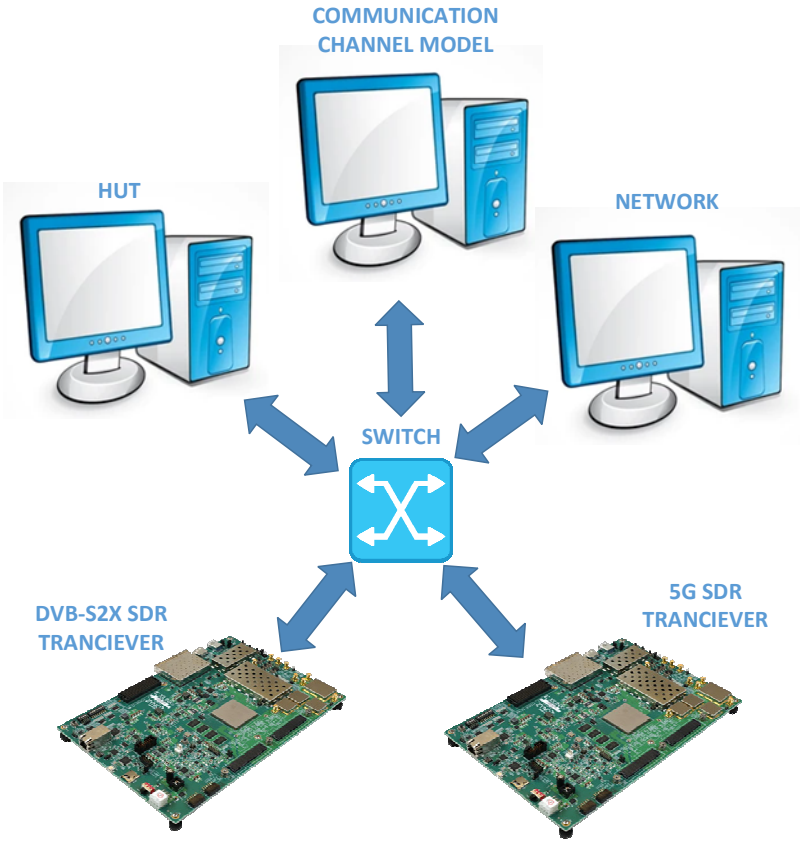


Figure 9: Logical design of hybrid user terminal proof of concept

The main functionality that should be provided to the HUT is the access to the both satellite and terrestrial networks. Since there is no single framework that can support both standards, separate acceleration of DVB-S2X and 5G modems is proposed. Those modems will be implemented on the RF-SoC platforms, where each platform will be used for one standard. Development boards are connected together using local switch that enables HUT to access both communication channels.

Proof of concept demo will function in the following way. HUT decides where data should be sent and use the common steering application which can interface with both communication standards. Data is packed in the Ethernet frame and sent to the one of the development boards. Data is processed through the transmitter chain and sent to the Communication channel model implemented on the PC. Data is then transferred back to the development board to be processed by receiver communication chain and sent to the Network PC. Network PC implements core network functionalities and should aggregate packets that come from both satellite and terrestrial communication channels.

CONCLUSIONS

This document D3.1 describes unified framework for satellite and terrestrial communication that will be used for hybrid terminal demonstration. Since there was no single framework that can support both communication standards, unification is done using application with the common API that can steer data to each of the communication channels. Because of these circumstances and since procurement of development boards is delayed, given framework still needs to be verified in order to accomplish milestone M3.1.