Software Defined Radio platform to evaluate processing latency of 5G NR MIMO functions

Karen Caloyannis, Anaïs Vergne, Philippe Martins

Software Defined Radio platform to evaluate processing latency of 5G NR MIMO functions

Karen Caloyannis
Télécom Paris
France
karen.caloyannis@telecom-paris.fr

Anaïs Vergne
Télécom Paris
France
anais.vergne@telecom-paris.fr

Philippe Martins
Télécom Paris
France
philippe.martins@telecom-paris.fr

ABSTRACT

This paper presents a Software Defined Radio (SDR) implementation in C++ of a Multiple Input Multiple Output (MIMO) transceiver platform using the open-source 5G physical layer *free5GRAN*. The platform evaluates the processing latency of physical layer functions at the receiver on a x86 processor, required for 2 and 4 layer MIMO transmissions. The goal is to verify if the functions can be implemented in software, and integrated into the *free5GRAN* 5G physical layer. The implemented MIMO transmission schemes are Spatial multiplexing based on V-BLAST and Alamouti Space Frequency Block Coding based on LTE Transmission Mode 2. Results show that MIMO decoding adds significant processing latency that may not respect the time budget for decoding. To reduce latency, functions have been implemented using AVX2 instructions and processing times between sequential and AVX2 execution are compared. Measurements are performed in a Faraday cage and the code will be open-source to provide reproductible results.

KEYWORDS

MIMO, MIMO-OFDM, Software Defined Radio, 5G, free5GRAN

1 INTRODUCTION

The work in [8] presents *free5GRAN*, a software library implemeting a 5G physical layer. It currently supports Single Input Single Output reception but aims at implementing other features like MIMO. To that end, this paper describes a software implementation of a MIMO platform, based on the 5G specification and using the *free5GRAN* library. MIMO has already been implemented in other software defined radio projects. In [3], a MIMO receiver integrated to *OpenAirInterface* (OAI) using precoding and Zero-Forcing is presented. It is limited to 2 layers in a real-time setup due to its processing speed but can decode up to 4 layers. Physical and transport layers involve complex computations and thus require specific architectures and optimizations to reduce their processing latency as described in [6, 7]. Improving the throughput of heavy computation tasks can be done by hardware acceleration, as done for transport layer decoding in [10]. Adding MIMO procedures might increase processing latency because they will depend upon the number of layers to be decoded and require more complex equalization algorithms. Processing latency must be limited to leave enough time for other functions and benefit from throughput increase. In previous works, no precise evaluation of MIMO processing latency at the physical layer is available.

This paper focuses on evaluating latency of MIMO processing functions at the physical layer, used to decode 2 and 4 MIMO layers. The goal is to verify if their implementation on a given platform is possible on a x86 processor or if hardware acceleration is required. The MIMO transmission scheme implemented is spatial multiplexing. It is based on V-BLAST and the platform implements up to 4 layers. Transmit diversity using Alamouti Space Frequency Block Coding (SFBC) is also implemented. It is not described in detail as it cannot be applied in 5G. Algorithms from *free5GRAN* are reused and *libuhd* is used to drive the SDR devices. Evaluation is performed inside a Faraday cage by transmitting and decoding a radio frame containing simulated data to be decoded.

Section 2 describes the MIMO transmission schemes implemented and their extension to MIMO-OFDM. Section 3 describes the platform and implementation details. Section 4 presents the results.

2 MIMO TRANSMISSION SCHEMES IMPLEMENTED

2.1 Spatial Multiplexing using V-BLAST

V-BLAST was introduced in [14] and is used for spatial multiplexing to increase data rates. The stream of symbols to be transmitted is demultiplexed into n_t independent layers that are sent separately on each transmit antenna. This transmission mode is modelled by

$$\mathbf{r} = \mathbf{H} \cdot \mathbf{s} + \mathbf{n}$$

where n_r and n_t denote respectively the number of receive and transmit antennas, $\mathbf{r} \in \mathbb{C}^{n_r}$ the vector of received signals, $\mathbf{s} \in \mathbb{C}^{n_t}$ the vector of sent symbols, $\mathbf{H} \in \mathbb{C}^{n_r \times n_t}$ the channel matrix where entries h_{ij} are the fading coefficients between receive antenna i and transmit antenna j, and $\mathbf{n} \in \mathbb{C}^{n_r}$ denotes the noise.

Different algorithms can be used at the receiver to cancel inter-symbol interference and recover each data stream. The decoding algorithm proposed in [14] is based on Zero-Forcing (ZF) combined with Successive Interference Cancellation (SIC). It computes the pseudo-inverse of a deflated version of $\mathbf{H}$ at each iteration which increases complexity. Linear equalizers like ZF have lower computational complexity but at the cost of performance degradation at low SNR. They also require a well-conditionned channel matrix.

A single sorted QR decomposition (SQRD) of the channel matrix has been used in [5, 15] jointly with a SIC detector. The SIC detector still suffers from error propagation that is limited by detection reordering. In [14] reordering is based on column norms in

5G-MeMU '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0301-0/23/09…$15.00
https://doi.org/10.1145/3609382.3610512

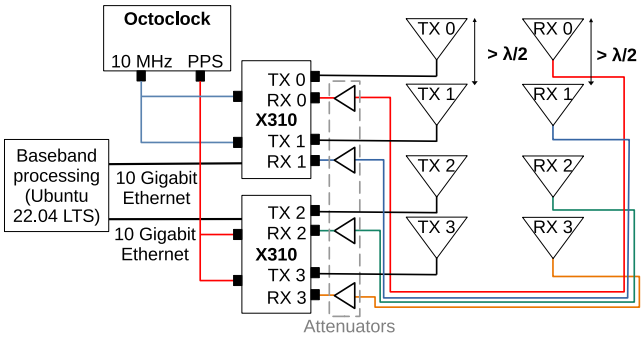

Figure 1: MIMO testbed

Table 1: X310 USRP configuration parameters

TX Gain MIMO 4TX [dB]	30
RX Gain MIMO 4RX [dB]	30
TX Gain MIMO 2TX [dB] and SISO	31.5 (max.)
RX Gain MIMO 2RX [dB]	33
RX Gain SISO 1TX/1RX [dB]	35
Center Frequency [GHz]	3.800
Bandwidth [MHz]	15.36
Sampling Rate [MHz]	15.36
Master Clock Rate [MHz]	184.32
Receive Frame Size [bytes]	8000
Send Frame Size [bytes]	8000
`num_recv_frames` (receive buffers)	256
`num_send_frames` (sending buffers)	256

the pseudo-inverse of the deflated **H** matrix. In [15] column norm reordering is integrated into the QR decomposition and tries to sort the diagonal elements of **R** in ascending order.

2.2 Alamouti SFBC

Alamouti SFBC [4] is based on LTE Transmission Mode 2 [1] for 2 and 4 transmit antennas. The system for two antenna ports transmitting two symbols s_0 and s_1 is given by

$$\mathbf{R} = \mathbf{HS} + \mathbf{N}$$

where $\mathbf{H} \in \mathbb{C}^{2\times n_r}$ denotes the channel matrix defined in 2.1 assumed to be constant over two subcarriers, $\mathbf{R} \in \mathbb{C}^{2\times n_r}$ the received symbols, $\mathbf{N} \in \mathbb{C}^{2\times n_r}$ the noise and

$$\mathbf{S} = \begin{bmatrix} s_0 & s_1 \\ -s_1^H & s_0^H \end{bmatrix}$$

denotes the encoded symbols. Precoding for 4 symbols and 4 antenna ports is similar, each pair of symbols being encoded on separate subcarriers and antennas. Element s_{ij} in **S** denotes the symbol transmitted on subcarrier i by antenna j. Element r_{ij} in matrix **R** denotes the received signal on antenna i and subcarrier j. Given that the precoding matrix is known and neglecting the noise, the receiver can recover symbols $\tilde{s_0}$ and $\tilde{s_1}$ by computing

$$\begin{aligned} \tilde{s_0} &= \frac{\sum_{i=0}^{n_r} h_{i0}^H r_{i0} + h_{i1} r_{i1}^H}{\sum_{i=0}^{n_r} |h_{i0}|^2 + |h_{i1}|^2} \\ \tilde{s_1} &= \frac{\sum_{i=0}^{n_r} h_{i0}^H r_{i1} - h_{i1} r_{i0}^H}{\sum_{i=0}^{n_r} |h_{i0}|^2 + |h_{i1}|^2} . \end{aligned} \tag{1}$$

The same method is used to recover 4 symbols transmitted by 4 antennas ports, each pair of symbols being encoded over different subcarriers and antennas.

2.3 Extension to MIMO-OFDM

The transmission schemes described previously assume that the channel remains constant during each transmission interval. When used in a wideband scenario, they are implemented jointly with OFDM (Orthogonal Frequency Division Multiplexing) and Cyclic Prefix addition to ensure this condition and avoid interference between the transmission intervals. Assuming the subcarrier spacing is smaller than the coherence bandwidth of the channel, independent MIMO transmissions on one OFDM symbol are perfomed on each subcarrier. Alamouti SFBC is applied on each pair of subcarriers and V-BLAST on each subcarrier. At the receiver, this implies that MIMO decoding has to be performed for every symbol and subcarrier, hence its computational cost must be limited, for instance by chosing suboptimal but faster algorithms.

In 5G Physical Downlink Shared Channel (PDSCH) processing, MIMO corresponds to the layer mapping and precoding steps [2, 9, 12]. Symbols of a PDSCH codeword are distributed onto layers that are fed to their corresponding DMRS port. Each DMRS port carries its associated PDSCH layer and DMRS used for channel estimation at the receiver. Interference between DMRS sent by different ports is avoided by using Code Division Multiplexing (CDM) groups so that the receiver can estimate the channel between each transmit and receive port. DMRS ports are grouped in CDM groups. The maximum number of CDM groups and DMRS ports per CDM group depend on the DMRS configuration. DMRS sent by ports in different CDM groups occupy a different set of Resource Elements (RE) and do not interfere with each other. DMRS sent by ports in the same CDM group occupy the same set of REs but are encoded by an Orthogonal Cover Code (OCC), used to separate DMRS at the receiver. A PDSCH codeword can be distributed over 4 layers at most. The base station can transmit up to 8 layers if it transmits simultaneously 2 PDSCH codewords, each codeword being distributed over 4 layers. Layers can be precoded but this step is not precised in the standard. Precoding is transparent to the receiver because DMRS are precoded along with the PDSCH. An Alamouti scheme cannot be applied because the receiver must know the precoding matrix. At the receiver, MIMO decoding is performed on each RE of the allocated PDSCH.

3 IMPLEMENTATION

3.1 Setup

The setup is depicted in Figure 1. The radio frontend is composed of two X310 USRPs that provide 4 transmit and receive channels. Transmission and reception are performed on the same devices. The channels of the USRPs are synchronized with an external reference that provides a 10 MHz reference clock and a Pulse Per Second (PPS) signal. The configuration values are given in Table 1.

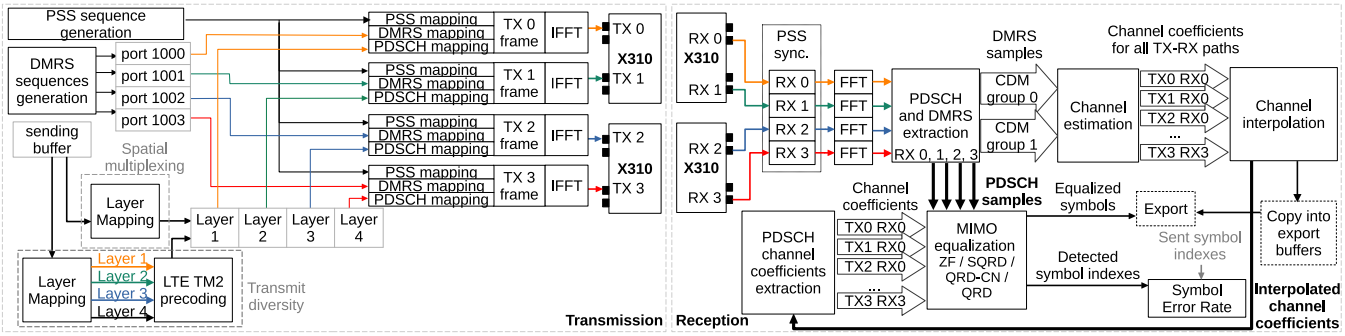

Figure 2: Transmitter and receiver processing performed on the same SDR device

Table 2: Radio frame parameters

Bandwidth [MHz]	15.36
Subcarrier spacing [kHz]	30
FFT size	512
Numerology index μ	1
Constellation	QPSK
FFT scaling factor	0.02
DMRS ports for 2 layers	1000,1001
DMRS ports for 4 layers	1000,1001,1002,1003
PDSCH slots per frame	19
DMRS configuration type	type 1 single symbol
PDSCH mapping type	Mapping type A
PDSCH length	14 symbols
PDSCH start symbol	0
DMRS l_0 (first DMRS symbol)	2
DMRS additional positions	3

USRPs are connected to the baseband through a dedicated 10 Gigabit Ethernet interface to avoid sharing bandwidth between them and to limit packet loss. Baseband processing runs on a *Ubuntu 22.04 LTS* remote server that uses a *Intel Core i9-10900K* processor. To reduce latency of baseband processing, CPUs have been set to *performance mode* and run at a frequency between 5 GHz and 5.3 GHz. Hyper-threading has been deactivated to limit core sharing between processes. The code is compiled using `-Ofast`, `-march=native` and `-ffast-math` options to enable compiler optimizations. `-march=native` generates code specifically optimized for the target processor. Because AVX and AVX2 are available, they are enabled by default and may be used implicitly when activating this flag.

To reduce channel correlation, the antenna spacing value is set superior to half the wavelength of the carrier frequency. Attenuators with a 40 dB reduction have been added because they improved signal quality at the receiver.

3.2 Transmission and reception procedures

The transmission and reception procedures are depicted in Figure 2. Note that they are performed on the same SDR device.

The transmitter prepares the frames to be sent following the parameters given in Table 2. The same Primary Synchronization Signal (PSS) sequence is generated for each transmit channel and placed on the first symbol of each frame. DMRS sequences of each port for the whole frame are generated and encoded with their respective OCC. A one-to-one mapping is considered between DMRS ports and physical ports of the SDR device, so the number of layers is equal to the number of used ports. Random PDSCH modulation symbols are then distributed over layers which have a one-to-one mapping with DMRS ports. Layers are precoded if transmit diversity is used. They are not precoded when using Spatial Multiplexing. RE mapping of PSS, DMRS and PDSCH is then performed and the same frame is emitted continuously. Note that the first slot of each frame contains only the PSS sequence, and that only 19 slots contain PDSCH.

The number of receive ports is set equal to the number of transmit ports. Receive buffers (one buffer for each receive port) store the equivalent of two frames to recover at least one entire frame. The frame synchronization algorithm is performed only on the first half of the receive buffers. The receiver then extracts the PDSCH and DMRS samples. DMRS samples are separated into their respective CDM groups and are used to estimate the channel coefficients. Then they are interpolated to get an estimate of the channel on the whole PDSCH allocation. The estimated coefficients on the PDSCH are then extracted and fed to the MIMO decoder. After the decoding stage, the equalized symbols and the channel coefficients are exported to be plotted and the symbol error rate is computed.

Execution times of reception procedures depicted in Figure 2 are measured with the method described in [13] by placing the corresponding sections of code in-between `RDTSC` and `RDTSCP` assembly instructions.

3.3 Physical layer algorithms

3.3.1 Synchronization. The synchronization algorithms already implemented in *free5GRAN* [8] are reused. Frame synchronization is performed by cross-correlating the PSS of the Synchronization Signal Block with the received signal. To avoid performing it for each receive port, the start of the frame determined on one receive port is reused for other ports, which is possible as the receive channels are synchronized. Moreover, synchronization is performed for only one value of $N_{ID}^{(2)}$, assumed to be determined before PDSCH reception (e.g. acquired during random access procedure) to avoid performing correlation for all possible values. Cross-correlation has also been reimplemented using AVX2 instructions to reduce its latency.

Table 3: Sequential versus AVX2 processing times

		Sequential [µs]			AVX2 [µs]			AVX2 Gain
		Mean	Min.	Max..	Mean	Min.	Max.	
2 layers	PSS synchronization known $N_{ID}^{(2)}$	6145	6060	6221	1205	1188	1240	5.10
	FFT [1]	452	398	468	447	391	462	NA
	PDSCH and DMRS extraction	10.2	8.07	18.8	9.34	7.76	20.1	none
	Channel estimation on DMRS	5.03	4.62	12.06	2.36	1.95	4.61	2.13
	Channel interpolation (ZF)	12.5	11.9	23.7	7.40	6.8	21.3	1.69
	PDSCH channel coefficients extraction (ZF)	8.33	7.69	16.12	8.59	8.12	16.5	none
	ZF	23.1	22.4	27.0	14.1	13.4	17.6	1.64
	ML detection [1]	41.9	40.5	46.0	41.6	40.6	58.3	NA
	Time to decode one slot ZF [2]	101	97.4	129	83.4	79.5	114	1.21
	SQRD	293	290	306	156	154	175	1.88
	Time to decode one slot SQRD [2]	329	325	372	184	180	218	1.79
	QRD-CN	290	287	300	159	157	181	1.82
	Time to decode one slot QRD-CN [2]	326	321	370	187	183	227	1.74
	QRD [1]	283	281	360	NA	NA	NA	NA
	Time to decode one slot QRD [2]	320	315	360	NA	NA	NA	NA
4 layers	PSS synchronization known $N_{ID}^{(2)}$	6146	6085	6246	1192	1140	1393	5.16
	FFT [1]	931	778	954	901	766	945	NA
	PDSCH and DMRS extraction	24.1	20.4	35.7	21.6	18.5	41.6	none
	Channel estimation on DMRS	19.2	18.0	23.9	8.58	7.54	17.0	2.24
	Channel interpolation (ZF)	52.2	48.4	96.5	29.8	27.5	65.4	1.75
	PDSCH channel coefficients extraction (ZF)	25.2	21.3	66.4	23.5	21	54.2	none
	ZF	289	287	307	94.6	92.1	135	3.05
	ML detection [1]	71.3	68.5	130	69.9	67.9	77.5	NA
	Time to decode one slot ZF [2]	481	467	575	248	238	348	1.94
	SQRD	851	842	896	361	355	412	2.36
	Time to decode one slot SQRD [2]	975	956	1096	446	431	577	2.19
	QRD col. norm.	956	944	996	360	354	479	2.66
	Time to decode one slot QRD-CN [2]	1081	1061	1174	444	432	647	2.43
	QRD [1]	804	798	865	NA	NA	NA	NA
	Time to decode one slot QRD [2]	928	913	1015	NA	NA	NA	NA
1 layer	PSS synchronization known $N_{ID}^{(2)}$	6309	6184	6613	1219	1192	1280	5.18
	FFT [1]	223	198	246	220	189	251	NA
	PDSCH and DMRS extraction	5.30	3.76	20.9	4.63	3.80	7.97	none
	Channel estimation on DMRS	0.79	0.57	6.26	0.78	0.60	1.83	none
	Channel interpolation (ZF)	2.06	1.80	5.51	1.88	1.62	6.79	none
	Extract PDSCH channel coefficients	2.10	1.78	19.0	1.89	1.68	4.40	none
	ZF	8.41	7.80	24.3	3.63	3.14	20.6	2.32
	ML detection [1]	21.5	20.1	37.9	21.1	20.1	38.5	NA
	Time to decode one slot ZF [2]	40.1	36.7	55.7	33.9	31.6	50.7	1.18
Alamouti	Alamouti 2×2 equalization [1]	58.1	55.6	77.4	NA	NA	NA	NA
	Alamouti 4×4 equalization [1]	71.8	70.4	89.6	NA	NA	NA	NA

[1] No AVX2 optimizations applied explicitly. Values in all columns correspond to sequential execution.
[2] Does not equal the sum of values listed in the table. Values are computed separately.

3.3.2 Channel Estimation and Interpolation. Channel estimation on DMRS and interpolation reimplement the Least Squares method described in [8]. Because reference symbols have unit norm, the channel on one RE is estimated by multiplying the received DMRS sample by the conjugate of the reference DMRS symbol. Interference between DMRS sent by ports in the same CDM group is suppressed by descrambling the OCC. For the configuration given in Table 2, this is done by considering the channel constant over two consecutive DMRS encoded by OCC in frequency, and by computing the mean between their estimated channels. This cannot detect variations of the channel between the two encoded DMRS. However, it enables optimizations depicted in Appendix A for interpolation. REs in between encoded DMRS subcarriers and on the edge of the band are not interpolated and the channel is set directly to the value estimated on DMRS. AVX2 optimizations do not take advantage of this structure and are used to perform computations on 4 REs at once.

3.3.3 MIMO equalization. ZF, SQRD [5] and QR decomposition with column norm reordering (QRD-CN) are implemented both in sequential and AVX2 versions. AVX2 implementation performs computations on 4 REs at once. QR decomposition without reordering (QRD) only uses a sequential algorithm.

Table 4: Variation of execution times for 2 and 4 layers versus 1 layer

	Sequential		AVX2	
	2 layers	4 layers	2 layers	4 layers
PSS synchronization	×1	×1	×1	×1
FFT [1]	×2.0	×4.2	×2.0	×4.1
PDSCH and DMRS extraction	×1.9	×4.5	×2.0	×4.7
Channel estimation	×6.4	×24	×3.0	×11
Channel interpolation	×6.1	×25	×3.9	×16
PDSCH channel coefficients extraction	×4.0	×12	×4.5	×12
ZF	×2.7	×34	×3.9	×26
ML detection [1]	×1.9	×3.3	×2.0	×3.3
SQRD	×35	×101	×43	×99
QRD-CN	×35	×114	×44	×99
QRD	×34	×96	NA	NA

[1] No AVX2 optimizations applied explicitly. Values in all columns correspond to sequential execution.

Table 5: Remaining time budgets after MIMO processing

		Sequential [µs]		AVX2 [µs]	
		Frame	Slot	Frame	Slot
2 layers	ZF + ML	1484	399	6763	416
	SQRD	-2848	171	4852	316
	QRD-CN	-2791	174	4795	313
	QRD	-2677	180	NA	NA
4 layers	ZF + ML	-6216	19	3195	252
	SQRD	-15602	-475	-567	54
	QRD-CN	-17616	-581	-529	56
	QRD	-14709	-428	NA	NA

SQRD is performed directly on the transposed matrix $\mathbf{H}^T$. The matrix has been transposed to perform dot products in *row-major* order which offers better performance in C and C++. Permuting columns in $\mathbf{Q}$, $\mathbf{R}$ and the detection order $\mathbf{p}$ as given in Figure 2 from [15] is suppressed. Multiplying the received symbols by $\mathbf{Q}^H$ and SIC are performed following the computed detection order. The norm of remaining columns to be computed in $\mathbf{Q}$ is updated using the same method as step (15), Algorithm 1 in [16] to avoid computing column norms at each iteration.

In QRD-CN, column norms of $\mathbf{H}^T$ are computed before QR decomposition and the detection order is determined in descending order of their value. The first layer to be decoded has the highest norm. Then QR decomposition is performed based on the detection order. When using AVX2 for SQRD and QRD-CN, reordering is based on the mean of the column norms over 4 REs.

ZF for 4 layers is based on [11]. The channel matrix is reduced to the hermitian matrix $\mathbf{H}^H\mathbf{H}$ to reduce the number of computations and use a faster algorithm to compute the inverse of $\mathbf{H}$. Only the upper half coefficients of $\mathbf{H}^H\mathbf{H}$ need to be computed due to its symmetry. Its inverse can be computed based on its LDL decomposition into an upper triangular matrix $\mathbf{R}$ and a diagonal matrix $\mathbf{D}$ as described in [11] to perform computations in a *row-major* order. LDL decomposition and computation of $\mathbf{R}^{-1}$ and $\mathbf{D}^{-1}$ are performed in-place inside the same memory location as $\mathbf{H}^H\mathbf{H}$. ZF for 2 layers uses a closed form inversion and ZF for 1 layer is implemented following the method described in [8]. Alamouti decoding is performed following equation (1) using a sequential algorithm only.

4 RESULTS

4.1 Execution times

Table 3 gives the execution times computed for 20 frames processed successively at the receiver. It compares between sequential and AVX2 execution, except for *FFT* and *ML detection*. *PSS synchronization* and *FFT* are executed once per frame, while other functions are executed at each slot. AVX2 gain is provided in the last column and corresponds to the ratio between sequential versus AVX2 mean execution time.

Time to decode one slot is computed for each slot by adding *extraction* functions, *channel estimation*, *interpolation* and MIMO decoding, excluding *PSS synchronization* and *FFT*. The mean, minimum and maximum values over all the frames are determined at the end of the simulation. *It is not equal to the sum of execution times given in Table 3.* For ZF, *channel estimation*, *interpolation* and *PDSCH channel coefficient extraction* are identified by *(ZF)*. Those for SQRD, QRD-CN and QRD are performed separately because the channel matrix stored in memory is transposed, but computation times are the same as for ZF. *NA* indicates that AVX2 optimizations are not applied explicitly so values presented in *AVX2* column correspond to a sequential execution. *FFT* is performed by an external library that may use AVX/AVX2 internally. *ML detection*, and QRD are not optimized. Table 4 gives the ratio between the mean execution time for a given number of layers versus one layer.

According to Table 2, the time budgets for frame and slot processing are set respectively to 10 ms and 500 µs. The goal is to reduce the function processing times below these time budgets and to leave enough time for other processing functions. The remaining frame processing time in Table 5 is computed by substracting the frame time budget by *PSS synchronization*, *FFT* and the number of PDSCH slots per frame (from Table 2) multiplied by *Time to decode one slot*. The remaining slot time budget is obtained by substracting the slot time budget by *Time to decode one slot*. Negative values indicate that the receiver is late.

4.2 Discussion

Table 3 indicates that AVX2 gains vary depending on the function. Execution times are not stable as shown by the min. and max. values. AVX2 optimizations divide *PSS synchronization* time by 5, but reduction for other functions is lower, between 1.6 and 3. No gains are visible for *extraction* functions that use the built-in function `memcpy()`. The most time consuming tasks at the frame and slot level are respectively *PSS synchronization* and MIMO equalization.

Table 4 shows the variation when increasing the number of layers compared to 1 layer. *PSS synchronization* does not vary with the number of layers but *FFT* increases linearly. MIMO equalization, channel estimation and interpolation have the largest variations.

Table 5 gives the remaining frame and slot time budgets after decoding. For sequential execution, no budget is left when increasing the number of layers. AVX2 optimizations reduce *PSS synchronization* time which leaves more frame budget than sequential execution

when using the ZF decoder. Although QR based decoders using AVX2 respect the slot time budget for 4 layers the budget left for other functions is insufficient.

Results show that the most demanding tasks at the frame and slot level are respectively frame synchronization and MIMO equalization. Increasing the number of layers adds significant processing latency. AVX2 gains are variable and their current implementation does not bring significant gains.

Some points shoud be noted regarding the results. They are not exhaustive and are given for a specific configuration. They will vary depending on other parameters such as bandwidth, frame structure, hardware and algorithm implementation. The platform does not implement other functions and procedures decribed in [8], as radio transmission parameters are assummed to be known at the receiver, so not all the Downlink procedures to decode data are included in the evaluation. The code is not fully portable since AVX2 is not supported by all processors. AVX512 optimizations could be implemented to further reduce the processing latency, but the same problem arises. Other solutions to reduce latency include multithreading and hardware acceleration.

5 CONCLUSION

This paper presents a MIMO transceiver platform based on the 5G specification and using the open-source library *free5GRAN*. The platform and the implemented algorithms are described. Processing time of physical layer functions at the receiver are measured on a x86 processor. Results show that MIMO decoding consumes significant processing time when increasing the number of layers, and that it may not leave enough time for other functions. They also suggest that MIMO processing is suitable for hardware acceleration when decoding a higher number of layers. The code will be published in the *free5GRAN* repository [1] to provide reproductible results. The next step is to integrate the algorithms into *free5GRAN* and its receiver to measure their performance within the whole physical layer processing.

[1] *free5GRAN* github repository : https://github.com/free5G/free5GRAN

REFERENCES

[1] 3rd Generation Partnership Project. 2017. TS36.211 : LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation. V14.2.0.

[2] 3rd Generation Partnership Project. 2020. TS38.211 : 5G; NR; Physical channels and modulation. V16.2.0.

[3] Khodr A. Saaifan, Thomas Schlichter, and Thomas Heyn. 2021. NR MIMO Feature Implementation into OpenAirInterface. In *WSA 2021; 25th International ITG Workshop on Smart Antennas*. VDE Verlag GmbH, 10-12 November 2021, French Riviera, France, 1 – 6. https://ieeexplore.ieee.org/document/9739185

[4] Siavash M. Alamouti. 1998. A simple transmit diversity technique for wireless communications. *IEEE Journal on Selected Areas in Communications* 16, 8 (Oct. 1998), 1451–1458. https://doi.org/10.1109/49.730453

[5] Ronald Bohnke, Dirk Wübben, Volker Kuhn, and Karl-Dirk Kammeyer. 2003. Reduced complexity MMSE detection for BLAST architectures. In *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*. IEEE, 01-05 December 2003, San Francisco, CA, USA, 2258–2262. https://doi.org/10.1109/GLOCOM.2003.1258637

[6] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. 2016. srsLTE: an open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM, October 3 - 7, 2016, New York City New York, 25–32. https://doi.org/10.1145/2980159.2980163

[7] Wang Tsu Han and Raymond Knopp. 2018. OpenAirInterface: A Pipeline Structure for 5G. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE, 19-21 November 2018, Shanghai, China, 1–4. https://doi.org/10.1109/ICDSP.2018.8631835

[8] Aymeric de Javel, Jean-Sébastien Gomez, Philippe Martins, Jean-Louis Rougier, and Patrice Nivaggioli. 2021. Towards a new open-source 5G development framework: an introduction to free5GRAN. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, 25-28 April 2021, Helsinki, Finland, 1–5. https://doi.org/10.1109/VTC2021-Spring51267.2021.9448964

[9] Chris Johnson. 2019. *5G new radio in bullets* (1st edition (colour) ed.). Chris W. Johnson, Independently published. http://www.5g-bullets.com/index.html OCLC: 1344338439.

[10] Florian Kaltenberger, Guy De Souza, Raymond Knopp, and Hongzhi Wang. 2019. The OpenAirInterface 5G New Radio Implementation: Current Status and Roadmap. In *WSA 2019; 23rd International ITG Workshop on Smart Antennas*. VDE Verlag GmbH, 24-26 April 2019, Vienna, Austria, 1 – 5. https://ieeexplore.ieee.org/document/8727207/authors#authors

[11] Aravindh Krishnamoorthy and Deepak Menon. 2013. Matrix Inversion Using Cholesky Decomposition. In *2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. IEEE, 26-28 September 2013, Poznan, Poland, 1–3. https://ieeexplore.ieee.org/abstract/document/6710599

[12] Xingqin Lin, Asbjorn Grovlen, Karl Werner, Jingya Li, Robert Baldemair, Jung-Fu Thomas Cheng, Stefan Parkvall, Daniel Chen Larsson, Havish Koorapaty, Mattias Frenne, and Sorour Falahati. 2019. 5G New Radio: Unveiling the Essentials of the Next Generation Wireless Access Technology. *IEEE Communications Standards Magazine* 3, 3 (Sept. 2019), 30–37. https://doi.org/10.1109/MCOMSTD.001.1800036

[13] Gabriele Paoloni. 2010. How to Benchmark Code Execution Times on Intel® IA-32 and IA-64 Instruction Set Architectures, Intel Whitepaper. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-32-ia-64-benchmark-code-execution-paper.pdf

[14] Peter W. Wolniansky, Gerard J. Foschini, Glenn D. Golden, and Reinaldo A. Valenzuela. 1998. V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel. In *1998 URSI International Symposium on Signals, Systems, and Electronics. Conference Proceedings (Cat. No.98EX167)*. IEEE, 2 October 1998, Pisa, Italy, 295–300. https://doi.org/10.1109/ISSSE.1998.738086

[15] Dirk Wübben, Ronald Böhnke, Jürgen Rinas, Volker Kühn, and Karl-Dirk Kammeyer. 2001. Efficient algorithm for decoding layered space-time codes. *Electronics Letters* 37, 22 (2001), 1348 – 1350. https://doi.org/10.1049/el:20010899

[16] Dirk Wübben and Karl-Dirk Kammeyer. 2006. Low Complexity Successive Interference Cancellation for Per-Antenna-Coded MIMO-OFDM Schemes by Applying Parallel-SQRD. In *2006 IEEE 63rd Vehicular Technology Conference*. IEEE, 03-07 April 2006, Melbourne, VIC, Australia, 2183–2187. https://doi.org/10.1109/VETECS.2006.1683243

A SEQUENTIAL INTERPOLATION METHOD

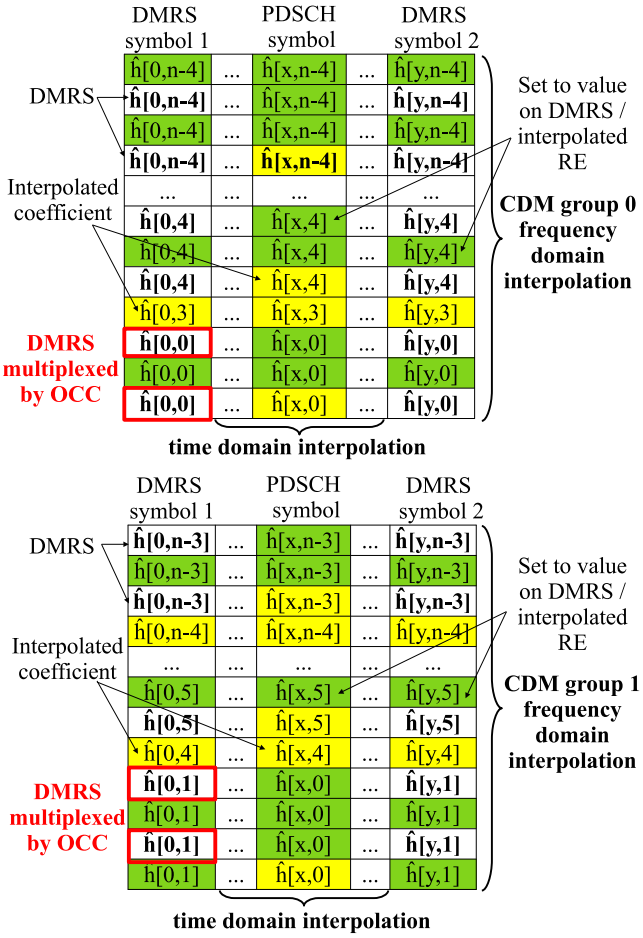

Figure 3: CDM group 0 and 1 sequential interpolation

Separating the DMRS as described in section 3.3.2 is suboptimal as it cannot detect variations of the channel between DMRS subcarriers, but it simplifies interpolation. Interpolation for CDM group 0 and CDM group 1 is depicted in Figure 3. It is only applicable to DMRS configuration type 1 single symbol. Interpolation is first performed in the frequency domain. Channel on REs in-between DMRS subcarriers encoded by OCC is set to the value on the DMRS. On the edge of the band, a similar simplification is done for the first or the last subcarrier depending on the CDM group. Time domain interpolation is then performed in-between DMRS symbols. REs that were interpolated in frequency domain are interpolated in time domain. Within groups of REs having their coefficient set to the same value during frequency domain interpolation, only one RE is interpolated in the time domain and the same value is applied to the remaining REs.