Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 98

Stefan Damjančević

A Framework for Analysis and Optimisation of Physical Layer Implementations on Programmable Vector Platforms in the Fifth and Sixth Generation of Mobile Communications

VOGT

Dresden 2023

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Bibliographic Information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2023

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „A Framework for Analysis and Optimisation of Physical Layer Implementations on Programmable Vector Platforms in the Fifth and Sixth Generation of Mobile Communications“ von Stefan Damjančević überein.

Gesetzt vom Autor

ISBN 978-3-95947-062-9

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

Technische Universität Dresden

A Framework for Analysis and Optimisation of Physical Layer Implementations on Programmable Vector Platforms in the Fifth and Sixth Generation of Mobile Communications

M.Sc.
Stefan Damjančević

der Fakultät Elektrotechnik und Informationstechnik der
Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil. Leon Urbas
Gutachter:	Prof. Dr.-Ing. Dr. h. c. Gerhard Fettweis
	Prof. Dr.-Ing. Norbert Wehn

Tag der Einreichung:	06. Januar 2023
Tag der Verteidigung:	11. April 2023

Stefan Damjančević

A Framework for Analysis and Optimisation of Physical Layer Implementations on Programmable Vector Platforms in the Fifth and Sixth Generation of Mobile Communications

Dissertation, April 23, 2023

Vodafone Chair Mobile Communications Systems
Institut für Nachrichtentechnik
Fakultät Elektrotechnik und Informationstechnik
Technische Universität Dresden
01062 Dresden

Abstract

Mobile communication standards with each new generation have enabled a new set of technologies that better the everyday human experience, and for those pioneers treading the unknown - opportunities. The *physical layer* (PHY) specifications of the *fifth generation* of mobile communications introduce variable timing constraints in the form of variable *transmission time interval*, depending on the *subcarrier spacing configuration*, along with the expansion of the throughput requirements. With the *sixth generation* expected to continue stretching the breadth of these requirements - even stricter timing constraints and use case dependant highly variable throughput, the significance of efficient PHY implementations is likely to become an opportunity for differentiation in the modem chip market. One way to handle the increased variability in PHY processing is with the application of the flexible *digital signal processor* (DSP) technology. In that light, the thesis sets its primary goal to analyse the applicability of DSPs in adapting the PHY system to operational circumstances and specific performance requirements. The flexibility of the DSP as a platform enables a high degree of freedom in mapping the PHY functionality and allows a trade-off between latency and efficiency in an implementation given some performance requirements like varied PHY timing constraints and throughput. Therefore, the secondary goal is to analyse the benefits of the latency-efficiency trade-off in PHY mapping on DSPs in respect to PHY specification requirements. To enable an investigation into these two goals the thesis proposes a framework for implementation of PHY signal processing algorithms for the next generation of specifications. To connect the implementation aspects and the specification requirements the framework places every implementation (hardware architecture, clock frequency, software implementation, signal processing algorithm, and workload size combination) on a two-dimensional rate-latency diagram, proposed to be called *kernel timing response*. The kernel timing response is used as a tool that captures changes between different implementations and use case requirements to help select the best-suited implementation for the given use case constraints. Although this method is developed to help analyse the applicability of DSPs, it can be easily adapted for analysis and optimisation of implementation on other machines with varied use case latency and throughput requirements. The proposed framework is then applied to investigate the application aspects of a *very long instruction word* (VLIW) *single instruction, multiple data* (SIMD) DSP in the implementation of *multicarrier modulation filtering* and

channel estimation PHY processing steps, given the requirements at the intersection of the fifth and the sixth generation mobile communications. The analysis shows that the programmable vector processor platforms, like the VLIW SIMD DSP, are indeed well suited for the implementation of some of the key PHY processing steps under high-end requirements. The findings point to a to a high utilisation of parallel processing functional units under a limited clock frequency envelope and ability to mitigate the impact of varying timing constraints on the required frequency clock through low-cost mapping of algorithmic optimisations to specific use cases on software.

Zusammenfassung

Mobile Kommunikationsstandards haben mit jeder neuen Generation eine neue Reihe von Technologien ermöglicht, die die alltägliche menschliche Erfahrung verbessern, und für jene Pioniere, die das Unbekannte betreten - bieten sich unternehmerische Chancen. Die Spezifikationen der *Physical Layer* (PHY) der fünften Generation der Mobilkommunikation führen variable Zeitvorgaben in Form von variablen *Transport Time Interval* ein, in Abhägigkeit von der *Subcarrier Spacing Configuration*, sowie eine Erhöhung der Durchsatzanforderungen. Mit der sechsten Generation werden diese Anforderungen voraussichtlich noch weiter erhöht - noch strengere Zeitvorgaben und ein vom Anwendungsfall abhängiger, hochgradig variabler Durchsatz - Effizientere PHY-Implementierungen werden, zur Differenzierung auf dem Markt für Modemchips an Bedeutung gewinnen. Einer Variante, zur Bewältigung der erhöhten Variabilität in der PHY-Verarbeitung, ist der Einsatz der flexiblen digitalen Signalprozessor (DSP) Technologie. Vor diesem Hintergrund ist das primäre Ziel dieser Arbeit das primäre Ziel, die Anwendbarkeit von DSPs bei der Anpassung des PHY-Systems an verschiedene Betriebsbedingungen und spezifischen Leistungsanforderungen zu analysieren. Die Flexibilität des DSP als Plattform ermöglicht ein hohes Maß an Freiheit bei der Abbildung der PHY-Funktionalität und ermöglicht einen Kompromiss aus Latenz und Effizienz in einer Implementierung bei Leistungsanforderungen sowie unterschiedliche PHY-Zeitvorgaben und Durchsatz. Das sekundäre Ziel dieser Arbeit ist daher die Analyse der Vorteile des Kompromisses zwischen Latenz und Effizienz bei der PHY-Abbildung auf DSPs, im Hinblick auf die Anforderungen der PHY-Spezifikation, zu analysieren. Um eine Untersuchung dieser beiden Ziele zu ermöglichen, schlägt die Arbeit ein Framework für die Implementierung von PHY-Signalverarbeitungsalgorithmen für die nächste Generation von Spezifikationen vor. Um die Implementierungsaspekte und die Spezifikationsanforderungen miteinander zu verbinden, stellt das Framework jede Implementierung (Hardware-Architektur, Taktfrequenz, Software-Implementierung, Signalverarbeitungsalgorithmus und Workload-Größenkombination) in ein zweidimensionales Raten-Latenz-Diagramm, das als *Kernel Timing Response* bezeichnet wird. Die *Kernel-Timing-Response* wird als Werkzeug verwendet, das Änderungen zwischen verschiedenen Implementierungen und Anwendungsfallanforderungen erfasst, um die Auswahl der am besten geeigneten Implementierung für den gegebenen Anwendungsfall zu unterstützen. Obwohl diese Methode entwickelt wurde, um die Anwendbarkeit von DSPs zu analysieren, kann sie leicht für die Analyse und Optimierung von Implementierungen auf anderen Architekturen mit unterschiedlichen Anforderungen an Latenzzeiten und Durchsatz angepasst werden. Das vorgeschlagene Framework wird angewandt, um die Anwendungsaspekte eines*Very Long*

Instruction Word (VLIW) *Single Instruction Multiple Data* (SIMD) DSP bei der Implementierung von Mehrträgermodulationsfilterung und Kanalschätzungs-PHY-Verarbeitungsschritten, angesichts Anforderungen an der Schnittstelle zwischen der fünften und der sechsten Generation der mobilen Kommunikation zu untersuchen. Die Analyse zeigt, dass die programmierbaren Vektorprozessorplattformen, wie der VLIW SIMD DSP, tatsächlich gut für die Implementierung einiger der wichtigsten PHY-Verarbeitungsschritte unter High-End-Anforderungen geeignet sind. Die Ergebnisse zeigen eine hohe Ausnutzung der parallelen Verarbeitungseinheiten, bei gleichzeitig begrenzter Taktfrequenz und die Fähigkeit der Reduzierung des Einflusses der benötigten Taktfrequenz durch eine günstige Abbildung von Software-Optimierungen für spezifische Anwendungsfälle.

Acknowledgement

> *If I have seen further, it is by standing on the shoulders of giants.*
>
> **— Isaac Newton**

Foremost, I would like to thank Professor Dr.-Ing. Dr. h.c. Gerhard Fettweis and Senior Vice-President Dr.-Ing. Yankin Tanurhan for initiating the Industry-University cooperation project "Efficient Implementation of the 5G Baseband Kernels on a Vector Processor". The work I have done within this project serves as the very basis of this thesis. Professor Fettweis chose me to carry it and trusted that I could get the job done successfully. Professor, thank you for that trust. I am grateful for your time, effort, and insistence to produce only excellent work during the entire course of my doctoral research. Your accomplishments, success, and kindness have been and are an inspiration, as a spark of light when things look dim.

Dear Dr. Emil Matúš, you were my group leader and a person I could rely on to have discussions for hours long if needed, as long as it took to clear the matter. You always had time for me, and your enthusiasm for unveiling the unknown is easy to admire and has grown on me. Thank you. I am honoured to have been part of this great team, and I will cherish these memories.

Thank you dear colleagues Dr.-Ing. Seugseok Nam and Luis Godoy, you two guys were there every lunch break making those short respites from science fun, with many memorable highlights. I would like to thank Peter, Faizan, Viktor, Simon, Daniel, Ana, Philipp, Robert, Sadia, Nairuhi, Carolin, Stephan, Jacob, Nick, Ahmad, Kathirn, Sylvia, Rüdiger, Raffael, and all other team members for being there and making my work possible one way or another. It was wonderful meeting and working with you.

Dear Dr. Ir. Pieter van der Wolf, you were my superior in Synopsys, someone to whom I would report on a weekly basis. Always precise, perceptive, meticulous, on point, and always on the lookout for the key message, with an impeccable ability to simplify and summarise complex thoughts with a few words - traits you

exemplified and expected from me. Thank you. You brought order to the chaos of research in the nebulous cluster of ideas, directions, and results. I see you as the epitome of professionalism and today, after all these years, a mentor for life.

Dear Dmitry Utyanksy, you were my advisor in all things software and algebra, a person whom I could consult and get an answer from when all other sources could not provide one. Many ideas developed in this thesis would not have been possible without your advice. Thank you. Your humble, yet sceptical demeanour has won me over a hundredfold, my friend.

I would also like to thank the Synopsys teams in Aachen, Eindhoven, and Leuven, for their guidance and support over the years.

Dear Prof. Dr.-Ing. Norbert Wehn, thank you for accepting the role of my thesis reviewer. I am grateful for your time and interest in this topic.

Dear Gabrijela Nikolić, you have been my initial contact in DAAD and you have stood by me on this academic trek in Dresden and Germany as a whole, with this thesis as its crown. Thank you, without you and the DAAD, my life would have been very different and I am happy to give back to the community.

Dear Professors Prof. dr Nadj, Prof. dr Malbaša, and Prof. dr Babković, thank you for instilling in me curiosity and bravery to thread the unexplored and the unfamiliar world, figuratively and literally. With your cheer and example, I have made it thus far.

Dear friends here in Dresden, Dr.-Ing. Shikha Ranjha, Marco Stolba, Christain Alrabbaa, and Javier Acevedo, you folks have been there through the thick of it all. Thank you. Dear friends from far away, Igor Bogdanović, Marko Pilipović, Dr. Stefan Drašković, and Marko Matijašević, thank you for keeping me cheerful, fortitudinous, and not taking myself too seriously, no matter the distance.

Dear father Andrej, in honour of your memory. Dear mother Milica, thank you for all the sacrifices you endure so that I am here today. I hope I made you proud.

Sincerely, your son, friend, colleague, and student,

Stefan Damjančević

Contents

List of Figures

List of Tables

List of Abbreviations

CP *cyclic prefix*

Rx *receiver*

Tx *transmitter*

AWGN *additive white Gaussian noise*

SNR *signal-to-noise-ratio*

CQI *channel quality indicator*

RE *resource element*

$\boldsymbol{H}_0$ *channel frequency response*

$\boldsymbol{h}_0$ *channel impulse response*

DFT *discrete Fourier transform*

i.i.d. *independent identically distributed*

LS *least squares*

MSE *mean square error*

LMMSE *linear minimum mean square error*

ISA *instruction set architecture*

DSP *digital signal processor*

vDSP *vector digital signal processor*

SotA *state-of-the-art*

VLIW *very long instruction word*

HW *hardware*

SW *software*

SIMD *single instruction, multiple data*

ASIP *application-specific instruction set processor*

ISA *instruction set architecture*

FU *functional unit*

LUT *look-up table*

ACC *accumulator*

mac *multiply-accumulate*

3GPP *Third Generation Partnership Project*

5G NR *Fifth Generation New Radio*

4G LTE *Fourth Generation Long Term Evolution*

NR XR *New Radio eXtended Reality*

CP *cyclic prefix*

OFDM *Orthogonal Frequency Division Multiplexing*

DFT *discrete Fourier transform*

iDFT *inverse discrete Fourier transform*

FFT *Fast Fourier Transform*

GFDM *generalised frequency division multiplexing*

UAV *unmanned areal vehicle*

UGV *unmanned ground vehicle*

V2X *vehicle to everything*

TTI *transmission time interval*

QAM *quadrature amplitude modulation*

QPSK *quadrature phase shift keying*

BW *bandwidth*

RB *resource block*

RE *resource element*

MIMO *multiple-input, multiple-output*

FR1 *frequency range 1*

FR2 *frequency range 2*

FR3 *frequency range 3*

MCS *modulation code rate scheme*

CA *carrier aggregation*

HARQ *hybrid automatic repeat request*

MAC-L *media access control layer*

SCS *subcarrier frequency spacing*

URLLC *ultra-reliable low latency communications*

eMBB *enhanced mobile broadband*

mMTC *massive machine type communications*

DBB *digital baseband*

DL *downlink*

CC *carrier component*

PHY *physical layer*

R17 *Release 17*

WLAN *wireless local area network*

ACK *acknowledgement*

NACK *not acknowledgement*

UE *user equipment*

QoS *quality of service*

HS *Hadamard-Schur*

List of Symbols

Constants

e	Euler's number $e = 2.718...$

Number sets

$\mathbb{Z}$	Set of integers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers

Operators

$\circledast$	Circular convolution
$\circ$	multidimensional element-wise multiplication across all dimensions
$\mathcal{O}(\cdot)$	Computational complexity operator
τ	Computational complexity ratio
$f(\cdot)$	Function of
$E(\cdot)$	Estimation operator

Notation

$D(k,l) = D(k+lK)$	Column matrix and array indexing, respectively
$D(k,l)$	Value of the k^th subcarrier (row) and l^{th} OFDM symbol (column)
$\boldsymbol{D}; D$	Matrix or array of data items, a single value data item
$\boldsymbol{D}(:,l)$	Values of the l^{th} OFDM symbol (column)
$\boldsymbol{D}(:,1:3)$	Values of the 1^{st} through 3^{rd} column
$d(n,l)$	n^{th} sample of the l^{th} OFDM symbol
$< \cdot >_{MK}$	modulo indexing with MK
m,l,n	Loop nesting order, loop with m inner, loop with n outer
$\mathbf{m},\underline{l},n$	Vectorised loops m and l

Parameters

μ	Subcarrier spacing configuration scaling factor - numerology
f_{clk}	Clock frequency of a processor
f_0	Clock frequency of a processor at a specific operating point
v_{len}	Vector length - data width of a SIMD FU in number of data items
$scalar$	Data width of a scalar FU in bits
$vector$	Data width of a SIMD FU in bits
$cycles$	Number of cycles needed to complete a task
v, v_{eff}	Effective vector length in data items of a kernel
η_{vec}	Efficiency of v_{len} utilisation in range of 0 to 1
A	SIMD gain
N	Theoretical number of multiplications or macs
η_A	Efficiency of SIMD gain relative to the maximum
a_0, a_1	Constants of the linear function
$\eta_{mem.acc.}$	Efficiency of a SIMD memory access FU
t_{budget}	Available time budget in a kernel
t_{wait}	Time before processing can start
t_p	Time spent processing data
t_{filter}	Processor time spent on filtering
t_{CE}	Processor time spent on channel estimation
t_{free}	Leftover headroom
$t_{p,opt}$	Optimal (deadline) t_p
$t_{filter,opt}$	Optimal (deadline) t_{filter}
$t_{CE,opt}$	Optimal (deadline) t_{CE}
l, m	OFDM symbol index
L	OFDM symbols in a transmission
M	OFDM symbols to filter
k	Subcarrier index
K	Subcarriers in a transmission
u	Spatial layer index
U	Active spatial layers in a transmission (CA$\times$MIMO)
R	Number of receiver antennas
T	Number of transmitter antennas
d	Complex data item value (time domain)
D	Complex data item value (frequency domain)
$D \in \mathbb{C}^{U \times K \times L}$	D defined in a U-by-K-by-L complex space

n	Sample index (time domain)
g_0	Filter coefficients
g	Normalised filter coefficients
x	Filtered data items
δ	Kronecker delta sampling function
$\boldsymbol{p}$	Reference (pilot) symbol sequence
k_p	Subcarrier (row) pilot index
l_p	OFDM symbol (column) pilot index
$\boldsymbol{k}_p, \boldsymbol{k}_l$	arrays of all row and column pilot indices
$\boldsymbol{H}_0$	Channel frequency response
$\tilde{\boldsymbol{H}}$	Estimate of $\boldsymbol{H}_0$
$\boldsymbol{h}_0$	Channel impulse response
$\tilde{\boldsymbol{h}}$	Estimate of $\boldsymbol{h}_0$
$\hat{\boldsymbol{H}}$	Channel estimate after channel measurement channel
$\tilde{\boldsymbol{H}}_s$	Channel estimate after symmetry reconstruction
$\tilde{\boldsymbol{H}}_c$	Channel estimate after column reconstruction
$\tilde{\boldsymbol{H}}_r$	Channel estimate after row reconstruction
e_m	Channel measurement error
e_f	Cumulative error after denoising
e_c	Cumulative error after column reconstruction
$e = e_r$	Cumulative error after row reconstruction and total error
$\boldsymbol{w}$	Independent identically distributed additive white Gaussian noise complex number zero mean sequence and variance σ_w^2
σ_w^2	variance of $\boldsymbol{w}$
$\boldsymbol{W}$	Pre-equalisation/spreading matrix
$\boldsymbol{Y}$	Received sequence
$\boldsymbol{X}$	Transmitted sequence

1 Introduction

> *Verba volant, scripta manent.*
>
> **— Latin Proverb**
> "Words fly away, writings remain."

One of the fathers of modern semiconductors, Gordon E. Moore, wrote in an article [Moo65] for *Electronics* magazine in 1965, that "personal portable communications equipment" are going to be one of the three wonders that integrated circuits bring in the future. Further back in 1926, a famous inventor, Nikola Tesla [Ken26] made a similar prediction regarding handsets, famously saying: "A man will be able to carry one in his vest pocket". And indeed, the mobile phone has profoundly transformed our society. Today, five generations into the evolution of that personal portable communication equipment that fits into our vest pockets' we are in a similar predicament in predicting the future of these devices.

In a nutshell, 5G or *Third Generation Partnership Project* (3GPP) *Fifth Generation New Radio* is a worldwide international standard regulating mobile communications, aiming to enable wireless connections of all electrically powered devices man has ever created. From the assembly lines, to mobile phones. From simple sensor systems, to complex vehicular networks. Any system that can benefit from multi-device environmental awareness, data transfer, data aggregation or data driven anlytics is a potential 5G application. The application space is broad and hence the requirements on the modems to support many categories of devices too.

Each new generation of mobile communications connects us more, enables new applications and services, automates existing processes, frees up time, stirs up economic growth by creating opportunities for profitable endeavours and innovation, and ultimately improves our day-to-day life and the society as a whole. However, each new generation of mobile communications also increases the complexity of devices - from terminal handsets to network architecture - through new requirements on top of old ones. 5G is a prime example where expanding old requirements - increasing network capacity, data rates, and user mobility, is coupled with new requirements - low latency, reliability, security and localisation - among a few other [ARI+16; Fet12; NGM15; Qua16]. In addition to these diverse requirements, the cost of mass-market devices has to be affordable for the average handset customer.

At the heart of signal processing in terminal devices is the modem chipset like Qualcomm's Snapdragon, Huawei/HiSilicon's Kirin/Balong, or Samsung's Exynos. These systems are developed behind closed doors of R&D departments of large enterprises, and the community is left with little insight into what kind of compute hardware platforms are used in modems and even less insight into the future of these components. So, what is behind this generational technology that changed everything? What are the challenges and requirements that we face today and will define HW of the future? What compute platform could be a good building block of consumer modems tomorrow and at what cost?

1.1 Problem Definition

One way to lower the production cost of a *user equipment* (UE) modem, is to utilise the economy of scale and have a system that supports use cases and applications to effectively compete in a market as broad as possible, such that the mass production of the system for a large enough market size can drive the cost down. Ideally, the production of a larger volume of the modem chip can amortise the non-recurring engineering costs of modem development. Hence, there is a need for the same chip to be used in different products for different market segments with some sort of low-cost customisation. The chip depends upon having sufficient *flexibility* to allow customisation for different market segments. However, modem customisation becomes harder and harder with an increasing number of services and their diverse operational requirements. Therefore, a *flexible*, cost-efficient modem design is continually an open question.

In terms of computational performance requirements and latency constraints, the most critical part of the modem is the *physical layer* (PHY). PHY is a chain of processing steps - algorithmic *kernels - self-contained function blocks*, required for processing bits into a signal for transmission and vice versa. A transmission in 3GPP standards is accomplished within a *transmission time interval* or *transmission time interval* (TTI). TTI is used as the smallest resolution packet possible for transmission generated by the PHY. Prior to 5G, the duration of TTIs was decreasing with new generations, but it was fixed within a generation, e.g. 1 *ms* in 4G [3GP21c]. Starting from 5G, the TTI is variable, presenting a new challenge for modem design as variable timing constraints opposed to fixed timing constraints in previous generations. On the one hand - there is an expansion of the range of data rate - throughput requirements to six orders of magnitude difference between low-end and high-end, compared to two orders of magnitude difference in 4G [FM17]; and on the other hand, 5G introduces three orders of magnitude difference in the range of the TTI duration between low-end and high-end [3GP20b; 3GP20c].

The increased variability in PHY can be dealt with by employing the flexible *digital signal processor* (DSP) technology. Therefore, the primary goal of this thesis is to investigate the applicability of DSPs in adapting the PHY system to specific performance requirements and operational circumstances. For example, depending on the instantaneous channel conditions, a best-suited algorithmic kernel can be selected to trade off computational and communication performance. However, the flexibility also gives the opportunity to trade off latency and efficiency given some performance requirements of algorithmic kernels, as the secondary goal.

The ever-increasing scope of services and complexity expected in future communication protocols beyond 5G [FB21a], [SBC20] is likely going to push for more required variability rather than less. New concepts have been proposed enabling to multiplex a multitude of PHY layers, the so-called "Gearbox PHY" [FB21b] and maturing technologies like "JC&S" [FB21c], [Rah+20], further stretch the *flexibility* requirement of future modems. Another aspect requiring *flexibility* is the need to ensure backward compatibility and cross-standard support. On top, lifecycle support i.e. ease of maintenance and functionality updates over a device's lifetime is another important topic. *Flexibility* through e.g. *software* (SW) updates would also increase the exploitation period of such a system, and enabling those at a low cost is key. However, *flexibility* often comes at a cost of lower *hardware* (HW) specialisation which provides *efficiency* in processing. The device has to retain competitive relevance throughout its lifecycle, hence, some combination of both HW specialisation and *flexibility* is required, which adds another layer of complexity to the requirements. This dynamism on the PHY-level, use-case-level, and protocol-level further requires *flexibility* from the modem.

Besides *flexibility*, there are two more challenges associated with modem development. The implementation needs to be efficient enough to be competitive e.g. in terms of area and power, and offer the required computational *performance* e.g. within a specific clock frequency envelope. It is clear that the modem needs to be at the same time both general enough to cover a variety of deployment, operational and maintenance requirements, specialised enough such that the implementation does not use more silicon area and power than necessary - to maximise *efficiency*, and powerful enough to compute all the processing steps for specified data rates and latency requirements. Therefore, the modem development methodology needs to take *performance* and *efficiency* requirements into account too. Hence, similarly to the delta from 2G to 3G, and the delta 3G to 4G, the delta from 4G to 5G and beyond has the challenge of increased computational workload and broader operational requirements - that need to be supported in the most efficient way possible. However, unlike the previous deltas, from 4G to 5G the performance requirement increase is constrained by the variable timing requirement introduced in 5G - adding a new dimension to the problem.

In summary, the thesis proposes an approach that investigates two aspects in modem PHY design:

1. suitability of DSPs to adapt to HW-SW-algorithm solutions, that best suit the computational-communication performance trade-off, given the *state-of-the-art* (SotA) workload requirements and operational circumstances, and

2. how the flexibility offered by the DSP can exploit the varying timing constraints in 5G to trade off efficiency and latency, given the required performance envelope investigated in step 1.

However, this does not mean that all processing steps in 5G PHY require the same level of *flexibility*. Forward error correction and channel coding and decoding play a crucial role in 5G specification too [3GP22a], yet the algorithmic variability has remained on a similar level compared to previous generations [3GP20a], and the focus is set on providing the most efficient way of dealing with huge computational requirements through dedicated HW accelerators [WSH21] [Her+21], [Kes+20]. Thus a flexible general-purpose DSP implementation is assumed not to be the best approach for these kernels and therefore the investigation of coding and decoding is not further persued in this work.

Flexibility, *efficiency*, and *performance*, are often at odds with each other and require balancing between generalisation, specialisation and brute force, respectively. Therefore, a programmable DSP platform is assumed with *single instruction, multiple data* (SIMD) data-level parallelism and *very long instruction word* (VLIW) with up to an eight pipeline stages instruction-level parallelism[1], as a compromise between *flexibility*, *efficiency* and *performance* requirements. In particular:

- a programmable platform to provide the *flexibility* thorough SW,

- SIMD to provide the necessary horsepower to drive the workloads - *performance*, and

- VLIW *instruction set architecture* (ISA) with DSP instructions to optimise for the domain-specific circumstances known at design time: algorithm and workload data properties, common operations, and structure e.g. available data parallelism, and deterministic scheduling - *efficiency*.

The framework exploration assumes kernels and workloads suited for high-end Release 17 5G NR as the most advanced, SotA, practical reference at the time

[1] See Appendix C in [PH17].

of writing, however, the writing style aims at showing relationships and scaling between parameters as per the two goals described above - for the longevity of the work. Furthermore, the high-end 5G NR PHY is taken as the most relevant benchmark in the context of an intersection towards 6G. Additionally, since the investigation requires delving deeper into all four levels: specifications, algorithm, software, and hardware - the thesis documents many aspects of the analysis as an amalgam of all four levels with many details required for deeper understanding written in footnotes, whilst aiming for a clear and simple main text body.

1.2 3GPP Compliant PHY Kernels

The scope of a whole generational technology such as e.g. *Fifth Generation New Radio* (5G NR) is gargantuan with many layers, aspects and to date 3 major releases[2], covering requirements for UE[3], base stations[4], backhaul, core network, other infrastructure, subsystems, network architecture, services and intertwining protocols that altogether make the network work in harmony [3GP21a]. The needed effort to investigate in detail and efficiently solve all of these aspects requires whole R&D teams, hence, for the purpose of this Ph.D. thesis, the focus is on the 3GPP *digital baseband* (DBB) PHY for UE mobile phone modems.

5G NR, similarly to 4G LTE, is an *Orthogonal Frequency Division Multiplexing* (OFDM) based system with OFDM symbol based transmission. A diagram of a typical OFDM communication system can be seen in Fig. 1.1. Every OFDM symbol is made up of a predetermined number of subcarriers. 3GPP calls subcarriers *resource element*s. To every *resource element* (RE), a *quadrature amplitude modulation* (QAM) symbol is mapped, and in turn every QAM symbol holds a predetermined number of encoded bits. Every group of encoded bits holds payload bits. Payload bits are sometimes referred to as information bits. These payload bits can belong to the control plane which is used for configuring the transmission process or to the user plane which holds majority of data that needs to be communicated. The group of payload bits are refereed to as a transport block in 3GPP terminology for scheduling and channel coding purposes. The transmission process goes as follows: On the *transmitter* (Tx) side the payload bits are received from the upper layers of the protocol stack, encoded, mapped onto QAM symbols, and transformed into OFDM symbols[5]. The OFDM symbols created by the DBB Tx are forwarded to the *radio*

[2] 3GPP Rel15, 3GPP Rel16, 3GPP Rel17, see [3GP21a] for more info.

[3] Simplified: covers a broad range of end-user devices such as modems for mobile phones and/or another type of equipment that can be connected to the network e.g. sensors/actuators etc.

[4] Referred to as NodeB in 3G UTRA/EDGE/HSPA, eNodeB in *Fourth Generation Long Term Evolution* (4G LTE) and gNodeB in 5G NR.

[5] In general this symbol could also be a product of a variation to the OFDM modulation, like DFT-S-OFDM, GFDM, FBMC, etc.

frequency (RF) *back end*[6] and transmitted. After passing through the environment i.e. the channel, the radio waves are received and processed into the digital form again by the RF *front end*[7]. The DBB *receiver* (Rx) essentially, performs inverse operations to the DBB Tx, however, the signal gets corrupted and distorted in the channel, hence the DBB Rx is not completely symmetrical to the Tx and needs additional processing steps to account for these distortions. The channel is a source of variability, for which on the Rx side, the additional steps are *channel estimation* and *channel equalisation*, through which the effects of the channel are in the ideal case identified and suppressed, respectively. The *synchronisation step* is also needed since the Rx does not have a priori knowledge about the Tx and effects of the channel onto the timing and frequency drift. Optionally, the DBB Tx can also employ *channel pre-equalisation* to predistort the signal provided it receives some feedback from the Rx about the channel or uses the channel estimation information from the time it was acting as the Rx. Lastly, the payload bits, once decoded at the Rx side, are forwarded to the upper layers of the protocol stack, which in part[8], eventually end up as data for *services* and *applications* running on the mobile.

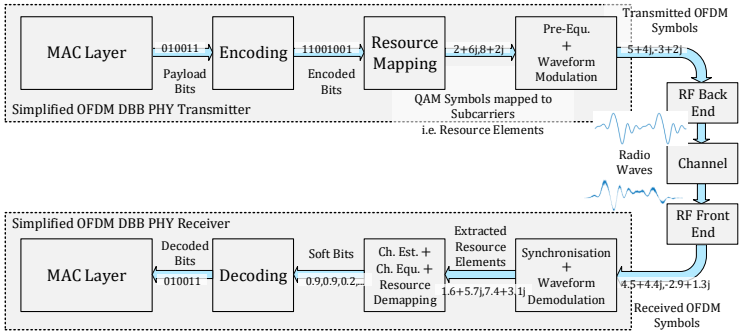

Fig. 1.1: Simplified Diagram of a General OFDM System with Illustrated Data Types between Steps.

With the basics of OFDM transmission introduced, let us focus on the details of PHY for 5G NR. Figure 1.2 and Fig. 1.3, first shown in [Dam+21b], present system diagrams for Tx and Rx parts of PHY, respectively, of a mobile phone or any 5G NR compliant device. Signal transmission to the UE is called downlink, and transmission from the device towards the base station i.e. cell tower is called uplink, therefore Fig. 1.2 and Fig. 1.3 are called uplink Tx and downlink Rx[9]. The

[6] Simplified: Circuitry responsible for converting the complex-numbered OFDM symbol samples into a radio wave transmitted over an antenna.

[7] Simplified: Inverse to the RF *back end*.

[8] There is an additional overhead encapsulation of data throughout the protocol stack.

[9] Not to be confused with downlink Tx and uplink Rx related to base station processing. The UE uplink Tx and downlink Rx, in 5G NR, can also double as sidelink Tx and Rx, provided sufficient *flexibility* in the *waveform modulation* block [RAN21].

Tx/Rx diagrams are not standard specified, rather they emerge from understanding and interpreting the specifications. The processing steps are divided among subsystems that serve towards a functional whole i.e. *coding*, *waveform modulation*, *synchronisation*, *channel estimation* etc. In both diagrams, several processing steps are highlighted in a darker shade of grey, compared to the rest. Due to previously mentioned dynamic channel conditions and transmission parameter variability, some processing steps may need to adapt their algorithms in real-time. *Waveform modulation* and *channel estimation* are good examples where a *flexible* approach to PHY is needed. In other words, these are the key processing steps, since they are not fixed in terms of algorithms used to perform the processing steps, and at the same time, these steps are dramatically influenced by the broad range of workload requirements[10]. Other processing steps are not investigated in-depth, but are mentioned for the sake of completeness of a 3GPP DBB PHY 5G NR system.

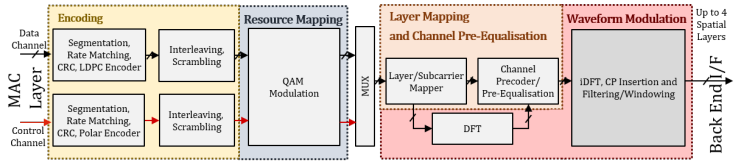

Fig. 1.2: 3GPP DBB PHY Uplink Transmitter System Diagram.

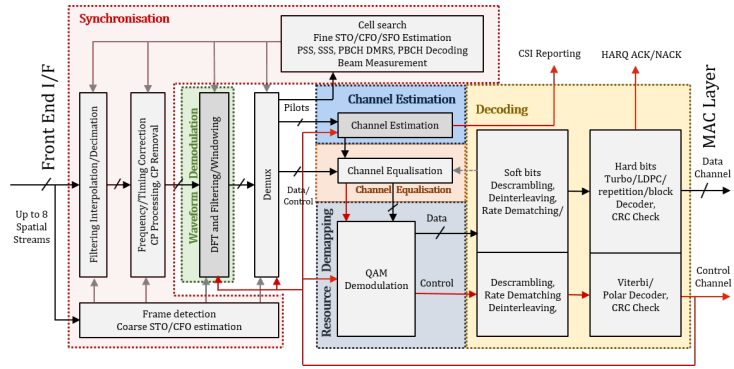

Fig. 1.3: 3GPP DBB PHY Downlink Receiver System Diagram.

[10]On the *efficiency* and *performance* front, the algorithms used in these processing steps are well suited for a specific type of fine-grained *parallel processing - SIMD processing*, however, more on that topic in Appendix A: Section 5.2.

Some of these processing steps are required per *carrier component* (CC) *bandwidth* (BW) [11] of a MIMO layer[12] or of a *carrier aggregation* (CA)[13]. This means that a PHY instance can potentially be scaled to a set of several PHY instances depending on the MIMO layer and CA configuration, determined by the base station. A single PHY instance can be servicing a large workload, but there may be several PHY instances servicing a small workload each needed to support several CC BWs via MIMO layers or CA [3GP20b],[3GP20c]. To meet this scalable workload *flexibility* is key. In addition to scaling the number of BW CCs for CA and MIMO layers, there are numerous other standard specified parameters, like the number of subcarriers and sampling rate, governing every CC BW to a point of high variability.

1.3 3GPP Compliant Workloads

Latency and throughput can be used as two aspects that infer the performance requirement, and the question is: how to find and relate latency and throughput in PHY specifications?

Knowing how many subcarriers are involved in a transmission translates, albeit in a convoluted way, can be converted into throughput of bits per second or bps. 5G NR specifications [3GP20b], [3GP20c] define, in a convoluted way, how many subcarriers are involved in transmission. This number is presented in units called *resource block*s or RBs, which are associated with a grid section of subcarriers belonging to several neighbouring OFDM symbols. The grid of a single *resource block* (RB) is spun by 12 neighbouring subcarriers of 14 neighbouring OFDM symbols, with some exemptions. The naming for subcarriers as *resource element* or RE comes from unit elements of this grid, and there are 12x14 = 168 REs in a RB.

Every subcarrier occupies a certain bandwidth in the frequency spectrum called *subcarrier frequency spacing*. All subcarriers in an OFDM symbol share the same *subcarrier frequency spacing*[14], which due to the duality of time and frequency domains determines also the duration of that OFDM symbol. The duration of several OFDM symbols bundled together form a TTI. For the purpose of this work 14 OFDM symbols bundled together form a TTI. Bundling 14 OFDM symbols serves also to

[11] Simplified: An established communication link, in a sense of occupancy of the channel for communication. Caution: Not to be confused with wireless channel state or *channel* for short, which describes the changes the signal goes through whilst occupying the channel.

[12] Simplified: Generating a communication channel by transmitting on another antenna in parallel i.e. several spatial links. Caution: Not to be confused with massive *multiple-input, multiple-output* (MIMO), which generates one or more MIMO layers through constructive and deconstructive interference of multiple radio waves.

[13] Simplified: Generating a communication channel by transmitting on a different radio channel in parallel i.e. several spectral links.

[14] Referred to as numerology in 3GPP jargon.

form a basic data packet that aligns with the size of the RB, therefore simplifying further discussion[15].

The mentioned 5G NR specifications [3GP20b; 3GP20c] targeting PHY of 3GPP compliant UE, determine the number of RBs in a transmission i.e. in a 14 symbol TTI and the *component carrier bandwidth configuration*[16] (CC BW), associated with that number of RBs. This configuration in turn defines how many subcarriers are there within an OFDM symbol, the *subcarrier frequency spacing* and the duration of the OFDM symbol, which also defines the TTI duration. Finally, with the TTI duration and the number of RBs in transmission, we can determine the required throughput for 5G NR transmission modes in terms of RB per second. The 3GPP PHY processing deadline is limited by the *hybrid automatic repeat request* (HARQ) *media access control layer* (MAC-L) procedure which requires a UE response within 3 TTI [Dam+19], effectively acting as a maximum latency constraint. Therefore, in 3GPP specifications we can find, a parallel measure to throughput and latency requirements from the required number of RBs per TTI and TTI duration.

The RB per second is parallel to throughput and TTI duration is parallel to required latency. In Fig. 1.4, you can see calculated values of all active and under-investigation configurations[17]. There are 3 different carrier frequency ranges: the *frequency range 1* (FR1) $(0.41\,GHz - 7.125\,GHz)$ [3GP20b], the *frequency range 2* (FR2)$(24.25\,GHz - 52.6\,GHz)$ [3GP20c], and the *frequency range 3* (FR3) $(52.6\,GHz - 71\,GHz)$ [3GP21b][18]. The FR1 overlaps in operation with 4G LTE and other legacy standards, whilst FR2 and FR3 are nuances in terms of using a new part of the electromagnetic spectrum i.e. the so-called cm and mm waves, for the purpose of 3GPP mobile communications.

The Fig. 1.4 shows the *throughput - TTI duration* relationship for a single CC BW. For the sake of better understanding, the measure RBs per TTI from the specifications is converted into RBs per millisecond, whilst the TTI duration is also expressed in milliseconds too. Calculating the values for a multitude of CC BWs transmitted e.g.

[15] There are exceptions to the duration of 14 symbols forming a TTI rule and there are exceptions to 14 symbols forming a data packet rule. In a more rigorous view of 3GPP specification, the TTI can consist of less or more than 14 OFDM symbols and the size of the data packet need not match 14 OFDM symbols e.g. several data packets in terms of a transport block codewords can be multiplexed within a single TTI [Dam+21b].

[16] Simplified: Frequency bandwidth and subcarrier spacing setup of a communication channel or a communication link and a possible operating point in 3GPP 4G LTE and 5G NR operation i.e. a use case.

[17] See appendix Chapter 6 for formulas and methodology of calculations

[18] At the moment of writing this the FR3 is planned as part of a future 3GPP *Release 17* (R17) for 5G NR.

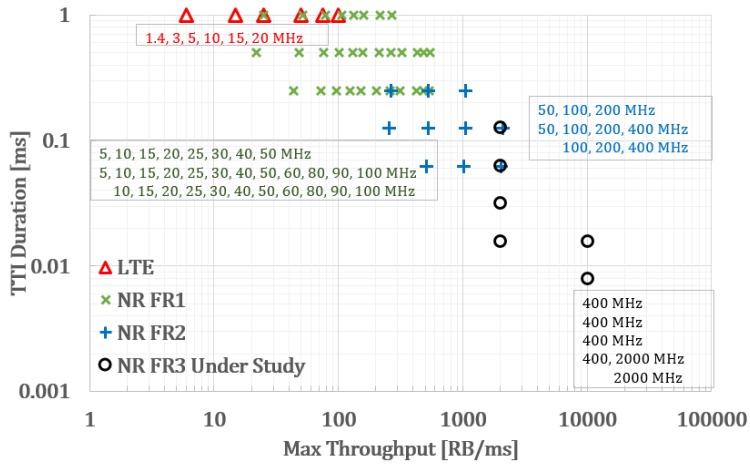

Fig. 1.4: Standard Specified Use Cases: Maximum RB Throughput and TTI Duration for all 3GPP Specified and Under Study Use Cases per *Component Carrier Bandwidth* Configuration.

simultaneously through techniques of CA and MIMO layers is a matter of linear scaling of the appropriate configuration[19].

We can see that the specification requirements, similarly to 5G/6G service requirements, also reflect the need for a broad range of throughput and latency requirements, namely a throughput difference in the range of 6 to 10000 RBs per millisecond, and TTI duration difference in the range of 1 millisecond to 7.8 microseconds, between low-end LTE and high-end FR3 corners, respectively[20]. The discrete steps between different use cases for TTI duration are regulated by a fixed scaling factor or numerology 2^{μ} - where μ is an integer[21]. More precisely, μ is regulating the subcarrier spacing, through it the OFDM symbol duration, and through OFDM symbol duration also the slot and TTI duration.

For the modem design discussion, the large breadth of throughput and latency requirements means that the structures that can provide *flexibility* and *performance* on the required scale i.e. *computational performance* have to be considered. Now

[19]For example, to calculate the throughput for an 8 layered MIMO, which is the maximum for current 5G NR [3GP20b], the selected throughput value in Fig. 1.4 needs to be multiplied by a factor 8. For *carrier aggregation* of 2 *carrier component bandwidths*, the throughput value needs to be scaled by 2 and so on.

[20]Note that 3GPP specifies categories of UE devices, some of which need to cover only a subset of those requirements, hence for some niches the workload breadth does not have to be as large. However, for the purpose of this work a general category of UEs covering the whole presented specification range is assumed.

[21]For LTE legacy $\mu = 0$, for FR1 $\mu = 0, 1, 2$, for FR2 $\mu = 2, 3, 4$, for FR4 $\mu = 3, 4, 5, 6, 7$

the question becomes how do these specification requirements, expressed in *TTI duration* and RB *throughput* translate into deadlines and bit rates of 3GPP DBB PHY processing steps and subsystems on the one hand and what HW structures can provide this flexibility.

A quick calculation of how many bits of information are transferred per CC BW can be made to check if the applications from the 5G/6G vision can be supported with current specifications. From Appendix A: Table 5.3, we can see that 5G NR can use up to 0.952 payload bits per encoded bit per 256 QAM symbol[22]. One QAM symbol is mapped to 1 RE and there are 168 RE per RB. Altogether yielding a maximum of 12.44 Gb/s for FR3, 2.49 Gb/s for FR2, and 0.622 Gb/s for FR1. The LTE maximum is at 124 Mb/s. Consulting Fig. 5.1 of the Appendix A, we see that with 5G NR we can fully support up to lower ranges of *New Radio eXtended Reality* and large file transfers in terms of required throughput. However, for the full support of 2 Tb/s we will still need an 160x increase in throughput compared to what the FR3 maximum can offer. Even with multiple CC BWs transmitted via techniques like *carrier aggregation* (*CA*) and *MIMO* (see Table 5.3 of the Appendix A), which still have to be specified, we would not even theoretically achieve the 160x additional throughput required[23], to cover the complete 5G/6G vision within 5G NR. Since current devices operate on FR1 [24], we are still far away in terms of market readiness, from making the 5G/6G vision a reality, which in turn makes this work ideally positioned for the next step into FR2, FR3 and beyond. Hence, it is not just that high *flexibility* is required, but also servicing a high workload. Figure 1.5 shows the calculated information bit throughput from specification use cases overlapped with 5G/6G *services'* and *applications'* requirements. In Fig. 1.5 the per CC BW notation is used since CA and MIMO for FR2 and FR3 are still not fully specified, and partially including CA and MIMO would not be fair[25,26]. On the latency front, the short TTI duration in 5G NR high-end means that the 5G NR PHY as per specifications should not be the main bottleneck for the latency requirement of 5G/6G *applications* and *services*. Yet, the TTI duration does give us an orientation point for calculating the maximum allowed latency for PHY processing.

[22] See [3GP22a] for possible *modulation code rate scheme* (MCS) combinations.

[23] So far CA up to 4×400 MHz in FR2 has been specified [3GP20c] and MIMO up to 8 spatial data layers [3GP22b] in FR1 have been specified. However, we cannot exclude the expansion of CA and MIMO specification into FR2, FR3 and beyond.

[24] Huawei's HiSilicon Kirin/Balong and Qualcomm's Snapdragon reporting 7.5 Gb/s [HiS20], [Qua20], and Samsung's Exynos reporting 7.3 Gb/s [Sam20] as *downlink* peak data rates including CA and MIMO on FR1.

[25] If we were to take SotA modems mentioned above, we can see that these would be able to support a single user with a single CC BW easily. However, the issue comes when needing to support several CC BWs via CA and MIMO.

[26] The reader can infer the total aggregated throughput by multiplying with the number of *component carrier bandwidths* supported. One of the appendices is a calculator in *.xlsx format to calculate these values under different configurations.

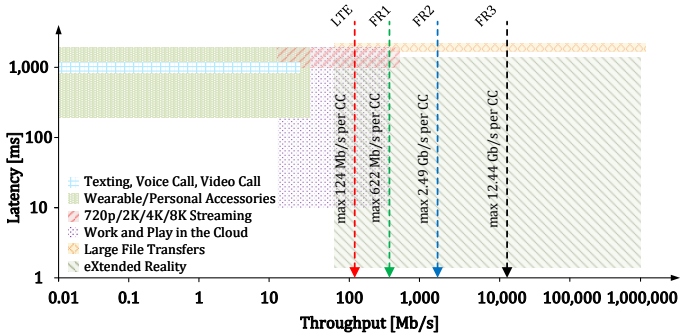

Fig. 1.5: Calculated Maximum Information Bit Throughput per *Component Carrier Bandwidth* of LTE and NR PHY overlapping Application and Service Requirements. Application and Service Requirements adapted from [FM17].

1.3.1 Computational Performance

For the next step closer to HW requirements, we want to translate the communication performance requirement into a computational performance requirement for PHY i.e. throughput and deadline requirements of individual PHY processing steps.

The selected kernels are on the critical data path, hence they adhere to the latency constraint set by the HARQ MAC-L procedure of 3x *TTI duration*, meaning that data items arriving at the Front End I/F have a processing time window anywhere between 23.44 μs and 3 ms untill the related *acknowledgement* (ACK)/*not acknowledgement* (NACK) message has to be ready on the Back End I/F [Dam+19]. This value represents the combined deadline distributed among the critical path. The ACK/NACK message size compared to the communication data items is very small, and as a simplification, its generation and transmission are not counted i.e. considered negligible towards the total deadline budget. Hence, in further discussion, the deadline budget is distributed among the receiver processing steps. To further distribute the delay budget the thesis assumes three task groups. The first group consists of the non-highlighted steps: *Synchronisation*, *Equalisation*, *Demapping* and *Decoding*; the second group holds *Waveform Demodulation*; and the third group consists of *Channel Estimation*. There is no rule of thumb or reference on how to group or distribute the deadline among these processing taks since the actual SotA implementations are vendor-specific and kept private by those vendors. Therefore, as a first-order approximation, this work assumes 1 *TTI duration* per processing task group as a deadline. Actually, these deadlines are even shorter to account

for additional overheads, however using more conservative deadlines would not highlight new issues[27]. Hence, if workload balancing issues exist with the current bound, these will surely exist with shorter bound estimates too and adding another layer of complexity will not add to the discussion.

Assuming a total 32-bit precision i.e. 16-bit real and 16-bit imaginary per data item[28], deadline distribution of 1 *TTI duration* per task group, and specification use cases depicted in Fig. 1.4, the per task group deadline and throughput in *bps* can be calculated. The calculated values for individual use cases per *component carrier* can be seen in Fig. 1.6[29]. The span stays the same as in Fig. 1.4, however, the data rates that reach the upper layers of the protocol stack to be used by the top layer applications can vary vastly due to different redundancy mechanisms like the QAM modulations, and ratios between information and encoded bits i.e. depending on the MCS used[30] on top of the *carrier component bandwidth* configuration. See Table 5.3 and Fig. 1.1.

In Fig. 1.6 we can see a multitude of standard specified CC BW *configurations* i.e. use cases. From Fig. 1.6 we can see that FR3 requires up to approximately 50 Gb/s, FR2 up to approximately 10 Gb/s, FR1 up to approximately 3 Gb/s, and LTE up to approximately 500 Mb/s. A ballpark estimate is that we will need thousands of instructions for every bit of throughput, a simple calculation would yield that any processor from Appendix A:Fig. 5.2 would need to run at tens of THz processor clock, which is impossible at present.

1.4 Processor Architecture

The concept of vDSPs or *vector digital signal processor*s is widely known and intuitively understood by electrical engineers or computer scientists. However, it can mean a lot of different things to different people depending on their previous experience with vDSPs[31], usually vague and inconsistent between any two readers, which

[27] For example, the conclusion from Chapter 3 states the need to trade-off throughput and latency to avoid HW overprovisioning in 5G/6G high-end assumes a 1 *TTI duration* deadline, would be reached even earlier in workloads with shorter deadlines. A shorter deadline exaggerates this effect.

[28] Even though the information is coded with up to 8 bits (256 QAM) per QAM symbol, the resolution at which we measure the incoming signal has to be higher than that. See [Dam+19] for a discussion on precision and data item resolution.

[29] See Appendix B: Chapter 6 for formulas and methodology of calculations.

[30] Simplified: MCS is a configuration of two parameters: 1) encoded bits per QAM symbol (up to 10 encoded bits per QAM symbol) and 2) ratio of information to encoded bits i.e. the encoding procedure increases the total number of bits before QAM mapping to enable error detection and correction on the receiver side (up to 0.926 information bits per encoded bit). For more on MCS used in 5G NR see [3GP20a; 3GP22a].

[31] As of late RISC V Vector Extensions seem to affect common trade terminology, where "vector processing" means "pipelined processing of vectors of arbitrary/configurable length". Whereas in

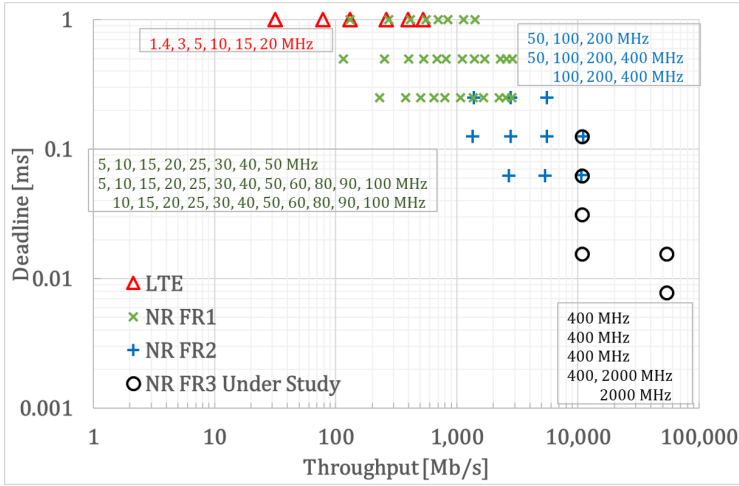

Fig. 1.6: Maximum Throughput in Mbps and Processing Step's Related Execution Deadline for all 3GPP Specified and Under Study Use Cases per Component Carrier Bandwidth Configuration under the Assumption of 32-bit (16-bit real, 16-bit imaginary) Precision per Data Item - *do not confuse with information bit throughput*.

in turn leads to not understanding the results or worse misinterpreting those same results. So let us set the fine print, and define what is meant by terms associated with the vDSP, what are the vDSP parameters and their values, and what is the methodology under which the results have been obtained.

The section describes the vector processor model used as a base HW platform in this work on a level necessary for understanding the work and results obtained using it. Some information about the processor model developed within Synopsys Inc. deemed not relevant to the understanding of the work are not disclosed and represent a trade secret owned by Synopsys Inc. For reference, the vDSP model is named CDSP. CDSP is developed and simulated in the cycle-accurate simulation using the ASIP-Designer Tool Suite [Syn]. ASIP-Designer suite comes with a C-compiler and tools for debugging and profiling SW kernels compiled for the CDSP, as well as tools to remodel and reconfigure the vector processor, such that a different configuration of the CDSP model can be emulated. The vDSP can be reconfigured in cases of significant bottlenecks in SW execution that could potentially come from the intricacies of kernel algorithmic and SW requirements that cannot be solved with algorithmic or SW optimisation. Lastly, using the ASIP-Designer, cycle measurements of kernels are taken and used for calculation of the required

older literature the term "vector processing" would refer to "SIMD processing", which is processing using fixed-width relatively short vectors, one instruction per vector operation. See [PH17, chap. 4, app. M.6].

processor frequency budget to run the kernels as per requirements of use cases shown in Fig. 1.6.

1.4.1 Block Diagram

Figure 1.7 shows a simplified block diagram of the CDSP *vector digital signal processor* (vDSP) model used in the work and simulations. The CDSP model is a wide vector SIMD processor with VLIW ISA and domain-specific instructions. The data intended for processing is accessed by the *memory access functional unit*s (FUs) either in scalar or vector fashion, however, the VLIW *configuration* can be set to allow parallel memory accesses by several memory access units. After accessing the data in *memory*, the access units store the accessed data in the *register file*. Once the data is in the *register file*, it then can be accessed by the *computational* FUs, which in turn store the result of computations back in the *register file*. Based on the VLIW configuration several *computational* FUs can be instantiated to compute the data in parallel. CDSP uses six configurable VLIW slots and up to eight pipeline stages depending on the instruction. Computed data from the *register file* is then stored back into *memory* via *memory access* FUs, assuming it is no longer needed in successive computations or some more urgent data needs to be loaded into the *register file*.

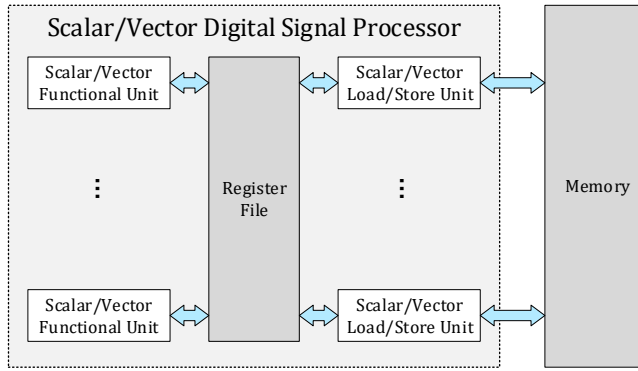

Fig. 1.7: Block Diagram of the Vector Processor Used.

1.4.2 SIMD

SIMD or *single instruction, multiple data* is used as an enabler for data-level parallelism. The principle is that the machine's SIMD FUs, *computational* or *memory access*

type, work on a set of data items, so-called *vectors*, performing the same identical operation over the whole vector, that gives the name *single instruction, multiple data*. Data items in the vector are referred to as *scalars*. The *register file* supports both scalar and vector data types matching the ones needed by either scalar or SIMD FUs. Actually, depending on the VLIW *configuration*, there can be several parallel FUs both scalar and SIMD. The SIMD vector assumed in this work is 512 bits i.e. vector data items are consisting of smaller scalar data items with a grand total of 512 bits for the whole vector. Most commonly though the instructions operate on sixteen scalar complex data items in parallel, each data item consisting of 16-bit real and 16-bit imaginary parts. The ISA supports integer and fixed-point arithmetic instructions, including shuffle and multiply-accumulate instructions for real and complex data types. The number of scalars in a vector is defined as:

$$v_{len} = \frac{vector\ [\text{bits}]}{scalar\ [\text{bits}]} \tag{1.1}$$

and referred to as *vector length*. Ideally, with SIMD processing the SW kernel can be performed with $v_{len}\times$ less cycles. However, this is not the case in practice, due to Amdahl's law [Rod85], numerous adverse effects, overheads and additional constraints, the $v_{len}\times$ gain is dampened. Additionally, a conscious loop vectorisation choice can cause the SIMD processor to be utilising less then the full v_{len} of data per computation, requiring in total more computations and lowering the achievable SIMD gain further[32] on top of the original unwanted overhead. The effective vector length can be described as:

$$v = v_{eff} = \eta_{vec}\, v_{len}, \tag{1.2}$$

where η_{vec} is the utilisation of the SIMD vector.

In the context of this work SIMD gain is formulated as:

$$\begin{aligned} A &= \frac{N}{cycles} = \frac{N}{f(HW, alg, SW, N) + \frac{N}{v}} \\ &= \frac{v\,N}{v\,f(HW, alg, SW, N) + N} = \frac{v}{\frac{v}{N}\,f(HW, alg, SW, N) + 1} \end{aligned} \tag{1.3}$$

where N is the *theoretical number of multiplications (or macs)* of the processing step divided by the number of cycles for that specific kernel implementation. As per Amdahl [Rod85], the cycle count consists of two parts: 1) The non-parallelisable $f(HW, alg, SW, N)$ resulting from the coupling of the HW-SW-Algorithm-Workload when implementing an algorithm on a vector machine even if the scalar version of that algorithm was 'perfectly' paralellisable; and 2) parallisable N/v, where v is

[32] This can be done to reduce another implementation cost like the number of memory accesses or, lower the number of registers needed or, lower latency.

the effective vector length. N is taken as the reference[33] under the assumption that the machine can compute a single multiplication (mac) every cycle. The expected number of cycles is at least N/v in the ideal case, but most likely requires additional cycles $f(HW, alg, SW, N)$. If $f \ll N$ the speedup is converging to v.

The speed up efficiency of the specific kernel implementation is formulated as:

$$\begin{aligned} \eta_A &= \frac{A}{v_{len}} = \frac{\frac{\eta_{vec} v_{len}}{v_{len}}}{\frac{\eta_{vec} v_{len}}{N} f(HW, alg, SW, Workload) + 1} \\ &= \frac{1}{\frac{v_{len}}{N} f(HW, alg, SW, Workload) + \frac{1}{\eta_{vec}}} \end{aligned} \tag{1.4}$$

where A is the SIMD gain from Eq. (1.3). If $v = v_{len}$ and $f \ll N$ the efficiency approaches 1. In practice $f \ll N$ may not be achievable always and f may also scale with N. However, due to Gustafson's law [Gus88] the increasing number of theoretical operations results in an increase in gain efficiency[34]. Since the thesis investigates kernels supporting different workload sizes, we can look for this efficiency increase based on the workload size in the experimental results. To illustrate what kind of behaviour we can expect from η_A, we can model f as a simple 1^{st} order polynomial[35] $f = a_0 + a_1 N$ dependant N and for simplicity sake assume $v = v_{len}$. In this case, η_A converges to an asymptote at $1/(va_1)$ for large N and starts at $1/(v(a_0 + a_1) + 1)$, for N = 1, or:

$$\eta_A = \frac{1}{\frac{v}{N}(a_0 + a_1 N) + 1}, \tag{1.5}$$

Therefore, according to this model, if in a kernel workload (N) is varied we can expect that the gain efficiency behaves similarly as the (1.5) curve, where a_0 and a_1 may vary between alternative kernel implementations. Figure 1.8 shows the model for two alternative kernel implementations.

There are many possible mechanisms of interaction between HW-SW-algorithms-workload, however, a simple model like this one can be easily verified with experimental results. The key points of the model are: 1) There exists an efficiency asymptote $1/(va_1)$ resulting in an offset in efficiency at large workloads for two alternative implementations, 2) there exists an efficiency offset $1/(v(a_0 + a_1) + 1)$ at N=1, and

[33] Alternatively, a scalar implementation can be taken as a reference, however, a bad scalar implementation could inflate the achieved speedup and make the implementation results incomparable to other scalar references.

[34] Why? Based on the consequence of Gustafson's law: for a fixed parallelism and increasing workload - speedup increases.

[35] Why 1st order polynomial? Some cycles are independent of workload size e.g. loop setup and control hence a_0, and some cycles are added that scale with increasing workload e.g. a data dependency in the main loop requiring stalls hence $a_1 N$.

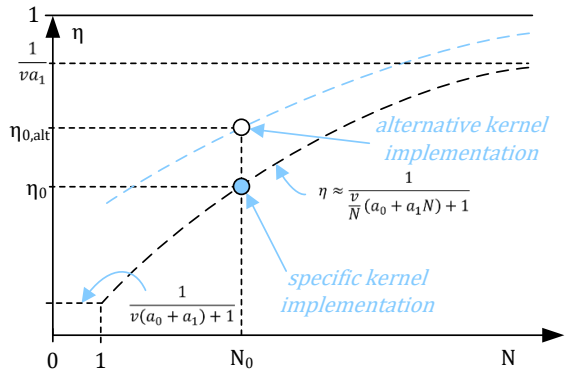

Fig. 1.8: Expected Efficiency Scaling with Workload Size - According to the Model with 1^{st} Order Polynomial Scaling of the Added Cycles with Workload.

3) the efficiency rises with increasing workload according to $1/((v/N)(a_0 + a_1N) + 1)$.

1.4.3 Functional Units and Register File

There are many FUs, some of the most important ones worth noting are the ones facilitating scalar/vector: memory access, addition (*add/vadd*), subtraction (*sub/vsub*), multiplication (*mpy/vmpy*), multiply-accumulation (*mac/vmac*), division (*div/vdiv*), and intra-vector data item reordering (*shuffle*).

Since the FUs support integer and fixed-point arithmetic, the *register file* also supports accumulator type scalar/vector data items e.g. 40-bit, 80-bit or 1280-bit word length depending on the operand data type. In addition, the *computational* FUs saturate to maximum or minimum values, in case of a computation overflow or underflow, respectively.

Memory access FUs work with non-accumulator data item sizes exclusively, e.g. 16-bit, 32-bit or 512-bit. There are two types of scalar/vector memory access FUs used: *LOAD/vLOAD* facilitating *laod/vload* instructions and *LD/ST/vLD/ST* facilitating both *load/vload* and *store/store*. The differentiation between the two variants come from a common theme in algorithms that an operation like *mac/vmac* needs two new data items sized operands from memory as input per computation, whilst producing a single output only after a set of iterations. Since there are disproportionately more *loads/vloads* than *stores/vstores*, a *STORE/vSTORE* FU facilitating exclusively *store/vstore* is omitted.

The number of memory accesses can vary between kernel variant implementations of the same algorithm even under a fixed HW architecture and VLIW configuration due to algorithmic tweaks that may require accessing certain data items several times. To compare memory access efficiency between the kernel variants, access efficiency $\eta_{mem.acc.}$ is defined as:

$$\eta_{mem.acc.} = \frac{unique\ data\ items\ accessed}{v_{len} \times kernel\ memory\ accesses} \tag{1.6}$$

where the *unique data items accessed* is the number of different data items the processor has to access[36], v_{len} is the width of the *memory access vLD/ST unit* and *kernel memory accesses* is the actual count of memory accesses.

1.4.4 VLIW

VLIW or *very long instruction word* processing approach provides instruction-level parallelism and effectively allows for several SIMD or scalar operations to be performed in parallel, depending on the processor's configuration, which in turn determines the number and layout of FUs. The number of FUs that can run in parallel is determined by the number of so-called VLIW *lanes* or VLIW *instruction slots*. A certain FU is tied to a specific VLIW *instruction slot* and can be called by scheduling a corresponding functional instruction word on that VLIW *instruction slot*. When scheduled right, the proper use of the register pipeline and VLIW *instruction slots* further decrease the number of cycles required for a SW kernel beyond the possible $v_{len}\times$ gain from SIMD processing. The number of VLIW *lanes* or their exact configuration is not disclosed, but for the interest of the presentation of the work, the following can be disclosed:

- There are two *vector memory access* FUs enabling a *vload+vload* or *vload+vstore* in parallel;
- The *memory access* FUs are matched with several *registers* and *computational* FUs to facilitate a subset of *vadd*, *vsub*, *vmpy*, *vmac*, *vdiv* and *shuffle* in parallel.

The design goal when defining a VLIW *configuration* is to find the smallest possible set of FUs and their parallel execution that achieves the desired throughput, whilst balancing used design area and power consumption.

[36] Number of operands and results of the theoretical number of multiplications (or macs) minus the number of overlapping operands and results that can be reused by another theoretical multiplication (or mac) in that algorithm.

1.5 Solution Approach

As mentioned, the implementation of PHY baseband consists of a chain of processing steps with dependencies and deadlines. How the varying timing requirements introduced by 5G impact PHY processing steps can be seen by expanding a timing diagram. Figure 1.9 assumes the following generalised timing diagram of a processing step. The processing step i.e. *kernel* requires a certain amount of time to process t_p, there is a certain time till deadline t_{free} that is the leftover headroom, a certain amount of waiting time for the previous step to produce enough data t_{wait} so that the current step can start processing, and a deadline at witch the next processing step starts. The whole processing step happens within t_{budget} assumed to be TTI duration long[37] - abbreviated TTI, out of which the total available time for processing is $t_{p,opt}$.

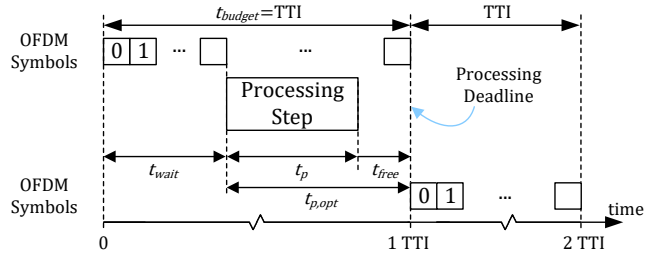

Fig. 1.9: General Timing Diagram of a 5G/6G PHY Processing Step.

From Fig. 1.9 the following relationships can be inferred:

$$t_{budget} = t_{wait} + t_p + t_{free} = TTI, \tag{1.7}$$

and

$$t_{p,opt} = t_p + t_{free} \tag{1.8}$$

where $t_{p,opt}$ is the optimal processing time in the sense of meeting the deadline for that TTI, such that if $t_p < t_{p,opt}$ then the implementation is over-provisioned requiring more resources than it needs, and if $t_p > t_{p,opt}$ then the implementation is under-provisioned and not viable.

The parameter t_p is dependant on the processor clock frequency f_{clk} and the number of cycles it takes for the task to complete as in:

$$t_p = \frac{cycles}{f_{clk}} \ [ms] \tag{1.9}$$

[37] See Section 1.3.1.

Cycle counts depend on a more convoluted set of factors. In broad terms the cycles depend on four factors, out of which three are selected by the designer and one is given by the specification. These are: the processor and HW architecture, the algorithm, its SW implementation - designer defined, and the workload parameter 'slot configuration[38]' - defined by the specification. For example, doubling the SIMD length or the number of cores tends to reduce the required number of cycles or, opting for a more computationally complex algorithm tends to increase the cycle count. A higher workload tends to increase the required cycles, and having a highly SW pipelined kernel tends to reduce the cycle count. Hence, these factors also impact t_p in addition to f_{clk}. Therefore, t_p can be designed with four out of these five factors - knobs: clock frequency f_{clk}, HW architecture, the algorithms used, and SW optimisations; whilst the slot configuration[39] is variable - but defined by the standard. In previous generations fixing these four knobs once to produce a t_p matching the deadline was largely sufficient for a PHY implementation, however with 5G the subcarrier spacing [40] became variable meaning that the duration of a TTI became variable. Therefore the number of TTIs and data for processing per unit of time became variable too. For this reason TTI rate can be defined as:

$$TTI\ rate = \frac{1}{TTI} \sim 2^{\mu} \tag{1.10}$$

where the TTI rate scales exponentially with the scaling factor[41] μ and is inversely proportional to the TTI duration. In 4G the TTI rate is fixed. Starting from 5G μ is variable. Assuming a kernel with five fixed knobs - f_{clk}, HW, algorithm, SW, and slot configuration, the changing TTI rate influences the workload by modulating the number of TTIs that need to be processed per unit of time. The additional workload variability has an impact on t_p through the number of cycles and scales exponentially with μ, same as the TTI rate. Or putting it in terms of TTI variability - t_p scales linearly with the TTI rate for a specific kernel implementation (fixed knobs):

$$t_p \sim TTI\ rate\,[\%]\,. \tag{1.11}$$

The parameter t_{wait} is dependant on the algorithm used, the slot configuration, and TTI duration[42]. Whereas in previous generations the variability of t_{wait} came from the choice of the algorithm used[43], 5G introduces terms like slot configuration and TTI duration as variable parameters. For example, a TTI slot can differ in the

[38] Number of Resource Blocks (RBs) per OFDM symbol and number of OFDM symbols per TTI.

[39] See Fig. 1.4 showing different *component carrier bandwidth* configurations (RBs per OFDM) and 14 OFDM symbol TTI.

[40] Along with different slot configurations.

[41] See note [21].

[42] See TTI duration in different *component carrier bandwidth* configurations in Fig. 1.4.

[43] As in requiring different granularity in terms of needed number of OFDM symbols buffered before the start of processing.

number of OFDM symbols or the position of important OFDM symbols like the ones carrying reference pilots can vary or the duration of OFDM symbols varies with changing scaling factor μ. For a fixed slot configuration t_{wait} scales linearly with TTI duration and inversely with TTI rate:

$$t_{wait} \sim \frac{1}{TTI\ rate}. \tag{1.12}$$

The parameter t_{budget} is limited by the processing deadline and is assumed to be the TTI duration hence it is also inverse to the TTI rate:

$$t_{budget} = \frac{1}{TTI\ rate}. \tag{1.13}$$

Inserting (1.13), (1.12) and (1.8) into (1.7) yields the scaling of $t_{p,opt}$:

$$t_{p,opt} \sim \frac{1}{TTI\ rate}. \tag{1.14}$$

The linear scaling of t_p with TTI rate and inverse scaling of $t_{p,opt}$ with TTI rate poses a challenge for the implementation of 5G/6G PHY.

The interplay of these two processes, t_p and $t_{p,opt}$, can be seen in Fig. 1.10 defined as *kernel timing response*. The relationship holds true for any kernel implementation. The $t_{p,opt}$ curve can move in case there are a different number of OFDM symbols in the slot configuration, whilst the rest of the parameters are fixed. An alternative kernel implementation can be achieved by changing one of the four knobs, and with it, the slope of the t_p line is changed as well. Out of the four it is perhaps most intuitive to scale the clock frequency f_{clk}, however there are other parameters that can be optimised too: HW architecture, the algorithm, or the SW optimisations, whilst taking into account the fifth static knob - slot configuration for that TTI rate.

So which design parameter can be scaled? There is no single answer. Increasing f_{clk} decreases t_p inversely as shown in Fig. 1.10, but there are limitations to practically plausible clock frequency ranges. Changing parallelism of the architecture impacts the cycle counts, but the HW needs to stay general enough to process all PHY processing steps. Changing the algorithm can lead to different costs in terms of memory accesses patterns, latency, or may require specific HW or SW solutions. Changing SW to better fit the HW and workloads can always be tried, but it is

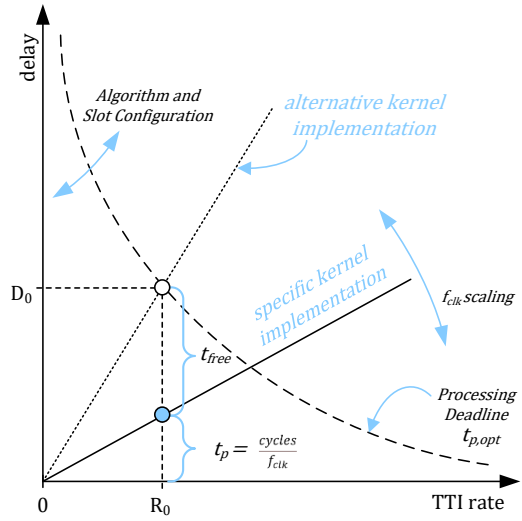

Fig. 1.10: Kernel Timing Response: Dependence of the 5G/6G PHY Timing Parameters on System Throughput.

hard to predict to which degree it will yield improvements to the cycle count. In previous generations there was no easy solution to metaphorically set these four out of five knobs for a fixed deadline, and 5G introduced a new dimension to the problem - a moving target. With TTI rates expected to continue rising in 6G, the implementation challenge rises in complexity and significance. It may not be sufficient anymore to turn one of the knobs, but rather a combination of a few. Out of the five design parameters - knobs: clock frequency f_{clk}, HW architecture, algorithm choice, SW optimisation, and workload related slot configuration, this thesis assumes the HW knob to be fixed, and the rest of the knobs are parametrised between alternative kernel implementations. The parameterisation of knobs is done according to a defined optimisation criteria and the interaction between knobs and the *kernel timing response* is investigated.

The motivation for having alternative kernel implementations is in minimising some cost function or optimisation criteria. Whilst the plane of the *kernel timing response* represent the space of plausible solutions, the third dimension of the graph can be cost, providing a cost function for operating points of the *kernel timing response* from Fig. 1.10. This means that alternative kernel implementations with larger slopes (more cycles for a constant f_{clk}) can also be viable solutions if they optimise some other criteria defined by the cost function inviting to consider possible trade-offs.

In regards to possible trade-offs and possible trade-offs and optimisation criteria, from Fig. 1.10, the observation can also be made about over- and under-provisioning based on where the line intersects the hyperbole. If for a fixed TTI rate R_0, $t_p = t_{p,opt} = D_0$, the implementation is optimal in the sense of meeting the deadline for that TTI rate, otherwise - if it is below the optimal point then the implementation is over-provisioned requiring more resources than it needs, and if it is above the intersection point - the implementation is under-provisioned and not viable. This means that there is an optimal point for every TTI rate, and turning the knobs introduces a completely new problem space that can be investigated.

Given the target HW architecture - a VLIW SIMD DSP, constraints related to variable timing and throughput requirements of 5G/6G workloads, likely 5G/6G signal processing algorithms, this thesis investigates the suitability of DSP platforms to provide alternative kernel implementations optimised for different cost functions, and provides a method to select the best suited alternative kernel implementation adapted to the instantaneous use case requirements. Additionally, as a showcase of the proposed framework, the thesis investigates the newly introduced concept of over- and under-provisioning kernel implementations for a given TTI rate R_0, as a methodical approach to trading latency and efficiency.

For this purpose, in the coming chapters, the thesis investigates Waveform Modulation via GFDM and Channel Estimation algorithms.

Waveform Modulation is chosen for investigation due to its highly regular processing structure and a high ratio of theoretical *multiply-accumulate* (*mac*) operations to unique operands, where maximising *mac*s per cycle and minimising memory accesses to only unique operands are selected as two optimisation criteria. Channel Estimation is chosen for investigation due to its comparatively short kernel processing deadlines and high algorithmic variability, where minimising the processing time and minimising the overall required clock frequency f_{clk} are the two optimisation criteria.

The thesis offers a contribution on the road to the next generation of mobile communications, bridging the rift between 5G and 6G. On the one hand, the dissertation offers a framework of implementing 5G/6G algorithms suitable for vector processing on vector DSPs with respect to standard specified workload requirements. And on the other hand, the dissertation analyses the capabilities of future vector DSP technology for implementing 5G/6G signal processing tasks.

1.6 Thesis Outline

This thesis is a compilation, extension and putting it all together of previous work, published in [Dam+19; Dam+21c; Dam+21b; Dam+21a] and selected areas from [Wit+19; Fet+19b; Fet+19a; Qur+22].

The rest of the thesis is organised in five chapters: In Chapter 2, and Chapter 3, the thesis follows the implementation of the Waveform Modulation and Channel Estimation kernels, respectively, discussing their unique algorithmic challenges, relevant optimisation criteria, and draws conclusions based on numerical results. The thesis is concluded in Chapter 4. Appendix A: Chapter 5 gives further background for the flexibility, performance and efficiency requirements in 5G/6G, whilst the Appendix B: Chapter 6 holds further details on parameter computation relevant to 5G/6G and the thesis.