POLITECNICO DI TORINO

Master's degree in Computer Engineering

Master's Degree Thesis

Optimization and Acceleration of 5G Link Layer Simulator

Supervisor

Prof. Luciano Lavagno

Candidate

Nasir Ali Shah

Sep 2019

Abstract

Requirements for higher data rates and wider bandwidth are increasing day by day with the rise and number of digital devices. Modern digital devices and systems such as IoT, autonomous vehicles, smart phones, entertainment systems require a good data rate and quality of service. LTE was presented as a successor of 4G to fulfill this requirement, but it is getting more and more congested with a rapid rise in number of connected devices and bandwidth requirement. To address these challenges, 3GPP announced the specification of a 5G network that will overcome all these issues and provide higher data rates and wider bandwidth by using higher frequencies. This architecture uses millimeter waves instead of the 0.8-2GHz frequencies used up to 4G. The frequency range for millimeter waves is from 30 to 300GHz. The problem with millimeter waves is that they are absorbed and dispersed easily by obstacles and thus can travel less distance. To overcome this issue, base stations need to be spaced closely. This results in a higher number of antennas spaced closely to each other.

When using higher data rates and large number of antennas, parameter tracking becomes computationally more and more complex. These parameters include angle of arrival(AoA), angle of departure(AoD),Zenith Angles of Arrival(ZOA), Zenith Angles of Departure(ZOD),user speed,user direction, antenna correlation. Each time these parameters change, the channel needs to reconfigure its parameters. The case is even more complex for an object moving with a high speed. This results in simulation performance degradation due to this bottleneck.

This work focuses on optimization and acceleration of parameter tracking, calculation and reconfiguration for a 5G channel model. Parameter tracking and re-calculation requires high computation time and large amount of memory. Conventionally, FPGAs are programmed using HDL-based design methodology to accelerate computationally intense applications. FPGAs are frequently used in many real-world applications to

speed up the performance alongside lower power consumption as compared to GPUs and multi-core CPUs. The OpenCL Synthesis tools for Xilinx and Intel FPGAs enable us to accelerate applications that were originally modeled in C/C++ with short development time. This thesis presents an optimized implementation of a 5G NR(New-Radio) link layer simulation model using OpenCL. Performance and power consumption for CPU and FPGA platforms are compared for various channel model configurations.

Acknowledgements

I would like to thank my supervisor, Prof. Luciano Lavagno for giving me the opportunity to work under his supervision . His guidance and feedback has been invaluable for me. I would also like to thank HEC Pakistan for enabling me to fulfil my dream of studying at Politecnico di Torino.
I would also like to thank my parents for their prayers and continuous support. I would like to thank them for encouraging me to fulfil dream of my life. I would like to thank my brother Mr. Tahir Ali Shah for supporting me at every step of my life.

I would like to thank my colleagues for being there for me whenever I needed help. Without their company, it would not be possible for stay away form home for so long.

Contents

List of Figures

List of Acronyms

CPU	Central Processing Unit
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
HDL	Hardware Description Language
IoT	Internet-of-Things
LTE	Long-Term Evolution
LOS	Line-of-Sight
HLS	High Level Synthesis
CLB	Configurable Logic Block
ASIC	Application Specific Integrated Circuit
LUT	Look-Up Table
ALM	Adaptive Logic Modules
5G NR	Fifth Generation New Radio
VHDL	VHSIC Hardware Description Language
Vivado HLS	Vivado High Level Synthesis

Chapter 1

Introduction

With the evolution of always connected systems such as Internet-of-Things, autonomous vehicles, and other digital systems, there was a need of new networking infrastructure that can provide service to all these devices. With a huge rise in number of these devices, the available spectrum is becoming more and more congested. Currently available network infrastructure cannot always meet the demand for data. In highly congested areas or in periods with high usage, consumers may face drop in service quality, unstable connections,slower speeds or even loss of service. Demand of data is rapidly increasing with a high rise in number of connected devices[2]. Long-term evolution (LTE) system which was an incremental improvement to 4G has already reached its maturity. According to visual network index (VNI) reports released by Cisco [3], it can be concluded that wireless data explosion is real and will even continue further. All these makes plain the forecast that an incremental approach to this issue will not be able to meet consumers requirements in near future[4].

There is a need of new network infrastructure that can uninterrupted service to all these devices. The infrastructure will provide stable connections, increased bandwidth and minimal lag. All this requirements set a base for fifth generation of wireless network called 5G. This new standard uses higher frequencies for communication. The usage of higher frequencies enable us to communicate at higher data rates but at lesser

distance. This means that they need to implement multiple antennas to boost its capacity and signal quality. 5G will also enable operators to divide a physical network into multiple virtual networks depending on its usage.

1.1 5G NR

5G uses frequency spectrum above 6GHz which was not usable by cellular services before. The new network uses the currently deployed LTE architecture to support non-standalone (NSA) 5G network which will be then evolved to full scale standalone 5G. The difference between two is in interface to core network and higher layers;the basic radio technology used is same[5].

The Third Generation Partnership Project (3GPP) has specified a new radio interface for 5G which is referred as New Radio(NR) [6]. This new model reuses some of features and structures from LTE. However, this new technology is not required to maintain backward compatibility like LTE. It Use of much higher-frequencies which gives us more spectra for wide bandwidth and higher data-rates. However, communication at higher frequencies is also affected by higher radio-channel attenuation which results in limiting network coverage. This problem is tackled by using multiple antennas for communication which also favors the beam-centric design of NR.

1.2 Channel simulation in network planning

Wireless channel model is the most important part in channel planning. It needs to cover a variety of frequency bands, a huge number of design parameters and heterogeneous deployment options. Modeling and simulating the impact of a physical channel in real environment is very important. For wireless systems it is even more crucial

because of high variations in propagation medium with respect to space, time and frequency. Channel simulation is important for evaluating performance of communication systems. Some of the main factors

- Channel simulation is essential in wireless network planning. The results of the simulations are used as input by planning tool (every network operator use these tools) in order determine the network infrastructure (network nodes) and its configuration
- Channel simulation allows to create a theoretical model which can be used to assess the performance of real devices. Many network operators develop these models by running simulations and the test commercial devices in lab and in the field in order to verify that nominal performances are achieved
- 3GPP and other standardization bodies use channel simulator in order to develop theoretical model of communication system and then use them in order to derive technological requirements or event to choose among multiple solutions, architectures, and systems.

1.3 Thesis workflow

In this thesis, this work has been depicted by designing hardware accelerators for simulating channel model of the 5G NR using OpenCL based implementation. The tools used are Vivado and SDAccel from Xilinx [7][8]. Chapter 2 explains about FPGA architecture and flow when design for an FPGA. Chapter 3 will present a brief summery of OpenCL platform and its implementation on FPGA platforms.

Chapter 2

Background

2.1 Field-Programmable Gate Arrays (FPGAs)

FPGAs are a type of programmable semiconductor devices that based on configurable logic blocks(CLBs). These CLBs are organized into a matrix form and are interconnected through programmable connections. As FPGAs are field programmable, they can be configured and programmed according to required functionality after manufacturing. There is present another class of semiconductor devices called Application Specific Integrated Circuits (ASICs). They differ from FPGAs in terms that they are designed for some specific functionality. This makes ASICs best optimized for that specific application but are then limited to that specific functionality. On the other hand we have general-purpose processors that can run almost every application but unoptimized performance. As FPGAs are re-configurable according to required functionality, they are considered more flexible than ASICs but at the cost of power and area. Also they are more efficient than general-purpose processors but the cost to pay is more complex programming and lower flexibility.

When seen at lower level, most of the FPGAs have a fixed architecture. Although there are available in market one-time programmable FPGAs, most of currently used FPGAs

are SRAM based. They are composed of a large number of SRAM cells that form Look-Up Tables(LUTs), a bunch of registers and interconnections in term of programmable routes. This enables the designers to adopt certain design to a different application just by updating contents of LUTs and setting routing configuration accordingly.

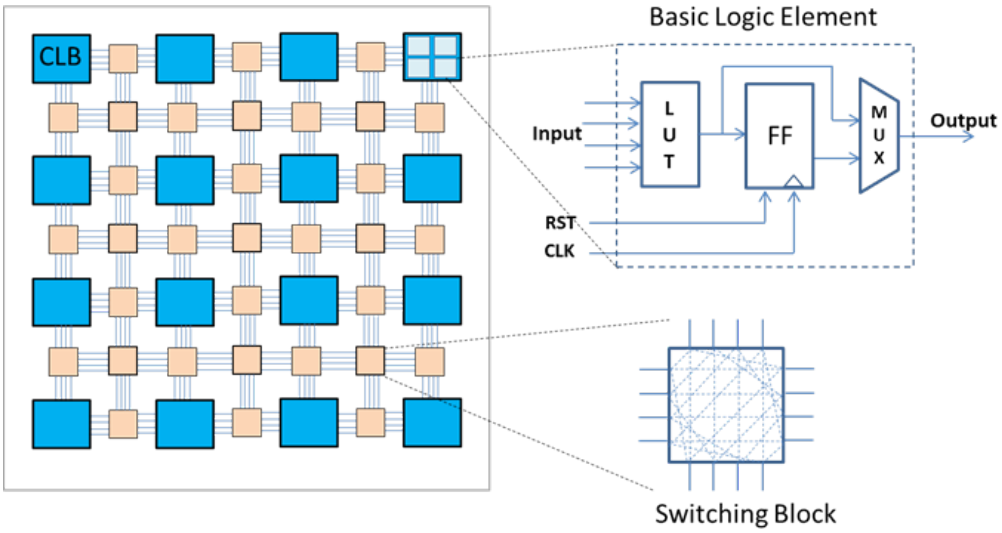

Figure 2.1: Typical FPGA Floorplan

To connect these individual CLBs with each other and external world, some re-configurable interconnection is needed. A matrix of programmable interconnects is used to connect these CLBs with each others and to IO blocks. Besides the LUTs as *soft-logic*, there is also some *hard-logic* components. These components include large memory blocks in form of Block RAM, Digital Signal Processor and a variety of memory controllers. These components are used for implementing some specialized logic functions that will take too much space if LUTs implementation was used.
Fig.2.2 show a top view of Intel Arria 10 FPGA architecture[9].

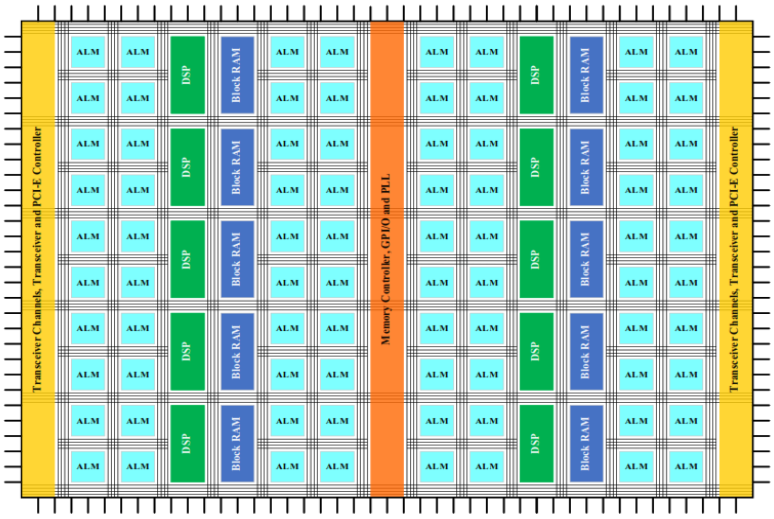

Figure 2.2: Intel Arria 10 FPGA architecture

The components on this FPGA are *soft-logic* called Adaptive Logic Modules (ALM) and there are **hard-logic** components that include DSPs,controllers, block RAMs, Phase-Locked Loops (PLL) and some Transceivers. The logic block on Intel Arria 1- FPGAs consist of multi-input LUTs, registers(Flip-Flops) and some adders and carry logic. Each LUT inside ALM can implement a combination of different functions. Fig.2.3 shows internal architecture of Arria10 ALM.

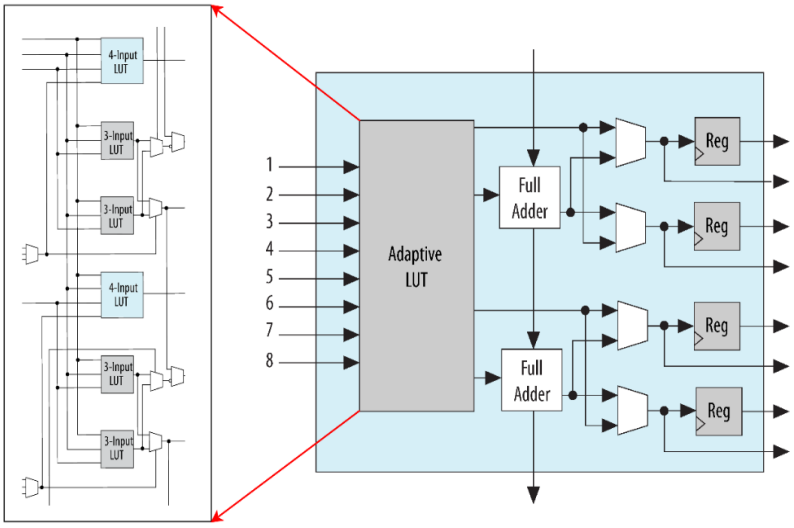

Figure 2.3: Intel Arria10 ALM architecture

Arria 10 FPGA also hosts some DSPs which are used to implement special logic functions. Fig2.4 shows a block diagram architecture for a typical DSP on Arria10 device [10]. These DSPs can perform IEEE-754-complaint single-precision floating-point addition(FADD), Fused Multiply and Add operation (FMA) or floating-point multiplication(FMUL). To implement a further complex operation, these DSPs can be chained together.

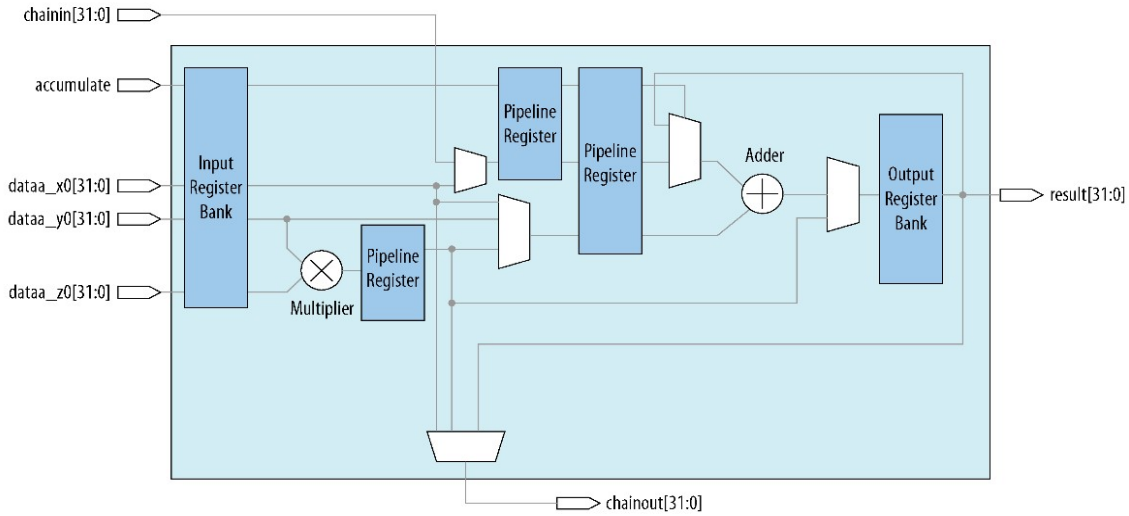

Figure 2.4: Intel Arria10 DSP Architecture

Another component included on Arria10 FPGAs is block RAM called M20K. each of these blocks is capable of storing up to 20 Kb of data. Each block has dual ports and can perform one read and one write operation at same time. Maximum data width for each block is 40 bits and 9 bits are used for address in this case. These memory blocks can also implement FIFO buffers or shift registers. These RAM blocks can be chained together in case a larger buffer is required.

2.1.1 FPGA Design Flow

The FPGA based design flow comprises of different phases including design entry, design synthesis, implementation, and finally programming the device. These phases are

explained below in details.

Design Entry

There are several techniques for design entry. It can be done starting from schematic or through some Hardware Description Language or a combination of both. For a system with high complexity, it is recommended to start with HDL instead of working with schematics at lower level. HDL based approach for design are fast and easy to implement and are most commonly used today for FPGA based design.

Design Synthesis

Design synthesis is the phase where the code is translated into actual circuit with lower level implementation such as gates, flip flops, adders. The input design is converted into net-list which describes components used and interconnections among them.
The process of design synthesis starts with syntax check once it is provided with HDL based design as input. The logic is then optimized by using different optimization techniques such as redundant logic elimination, logic reduction, size reduction and simultaneously making its implementation faster. After possible optimization is carried-out, the design is mapped on the technology and information for timing and area is extracted.

Implementation

This step determines the layout of final design. It is composed of three different steps; translate, mapping and finally place and route. The tool gathers all the design constraints together with the design net-lists. The tool then maps the resources available on FPGA according to requirements specified in the design. After the mapping, the next step is to route the signals and connection of IOs.

Programming FPGA Device

The output of implementation step is bit-stream file that contains the footprint of final design on device. This file is then uploaded to FPGA to finally implement the design on actual FPGA.

Chapter 3

OpenCL Overview

Open Computing Language which is commonly referred as OpenCL is one of the very first industry standard framework for heterogeneous computing systems[11]. It is a royalty-free open-source standard. OpenCL is also compatible with High Level Synthesis(HLS). It is updated and maintained by Khronos Group in collaboration with major hardware design companies[12]. Normally heterogeneous platforms include CPUs, DSPs and GPUs. HLS supports also adds FPGAs into heterogeneous platforms list. OpenCL is based on C99 programming standard. It also provides Application Programming Interface(API) for controlling the accelerator and its communication with host processor. An OpenCL-based application is mainly composed of two separate parts. The part which is not computational intensive and usually contains algorithm implementation and executes on CPU is called *host* code. It is independent of programming language as long as there is compatible compiler. The other part contains C-based code called device code. It implements the kernel. As for OpenCL framework, it can be divided into four different models to understand its way of work.

1. Platform Model
2. Execution Model
3. Memory Model

4. Programming Model

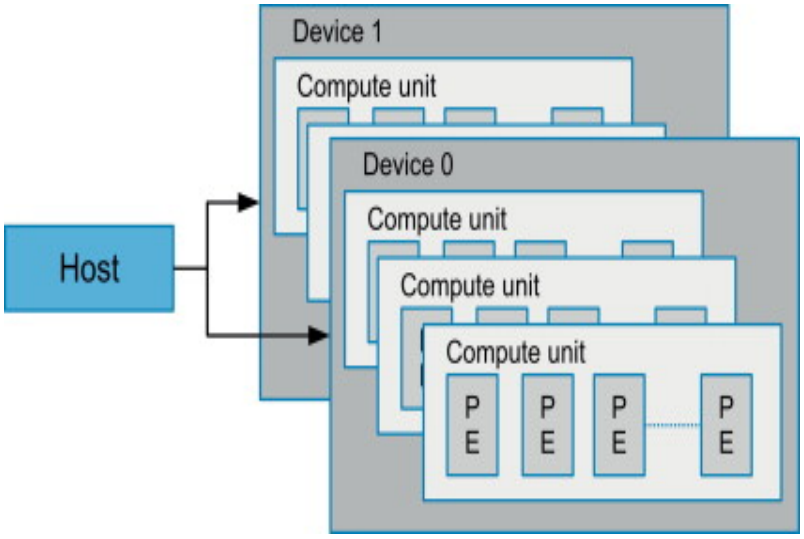

Figure 3.1: OpenCL Platform Model

3.0.1 Platform Model

A brief overview of OpenCL platform model is presented in Fig.3.1. An OpenCL based application executes on the host system and mostly implements the control part of application. There can be any number of devices connected to the host. Type of connectivity of supported device with the host is not specified by OpenCL standard. The OpenCL implementation specific to that device is considered responsible to perform the communication and is usually hidden for application developer. OpenCL devices contain one or more Compute Units(CU) which further contains one or more Processing Elements(PE). Inside an OpenCL device, the actual computation is performed by PEs.

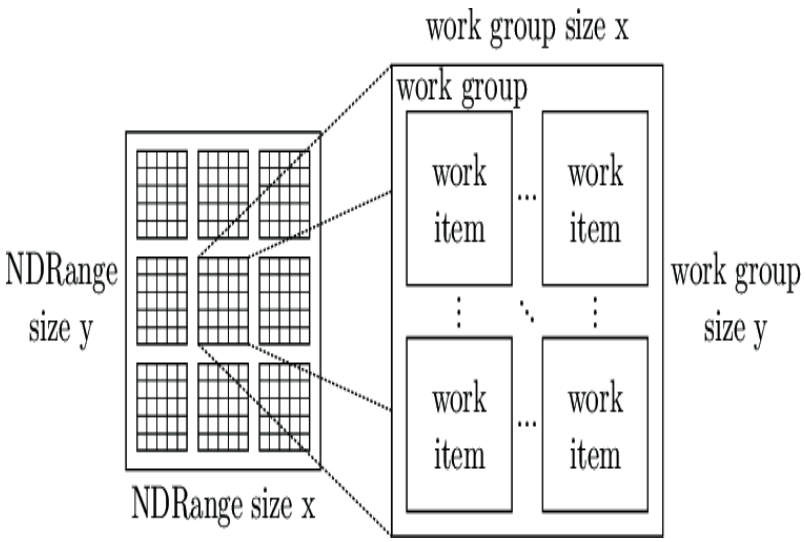

Figure 3.2: OpenCL Execution Model

3.0.2 Execution Model

An OpenCL application is composed of host code and device code. Host code is written with API to manage program objects, memory objects and command queues. Kernel code mainly implements the computationally intensive part of the application and is executed on OpenCL device. The following set of parameters is required to setup and OpenCL execution environment.

Context

As there can be more than one OpenCL supported platforms, we need to specify which of these we want to be used for execution. The context consists of all necessary information to setup one or more devices for execution by using OpenCL API.

Kernel

Kernel is the device code that implements the actual computation part. The same kernel can be executed by all the PEs inside a CU.

Work-Items and Work-Groups

For execution of kernel, a virtual N-dimensional space is defined. For each of these points, an instance of kernel is executed. These kernel instances are called work-items. The execute the same code but on different set of data. Each of these work-items is being assigned a unique ID and can be identified across the whole N-dimensional space. work-items are then grouped together in a work-group. All work-groups have the same dimensions. Work-items within a work-group has access to shared local memory.

Program Objects

Program object is mainly composed of source code and binary implementation of kernel code. The binary implementation of kernel can be either generated from source code, or it may be available pre-compiled. It can be considered as a dynamic library of kernels because there can be multiple kernels inside a program object.

Memory Object

Memory objects are used for data exchange between host and devices. They are visible to both the kernel and the host. They work as input and output objects for kernels.

Command Queue

It is used by host to manage command execution on kernel. It can be either a memory command to data transfer, or a kernel command to launch kernel or synchronization command to create points of synchronization.

3.0.3 Memory Model

Based on the access pattern, the OpenCL memory model is divided into four categories.Fig.3.3 shows the division of memory model for OpenCL[13].

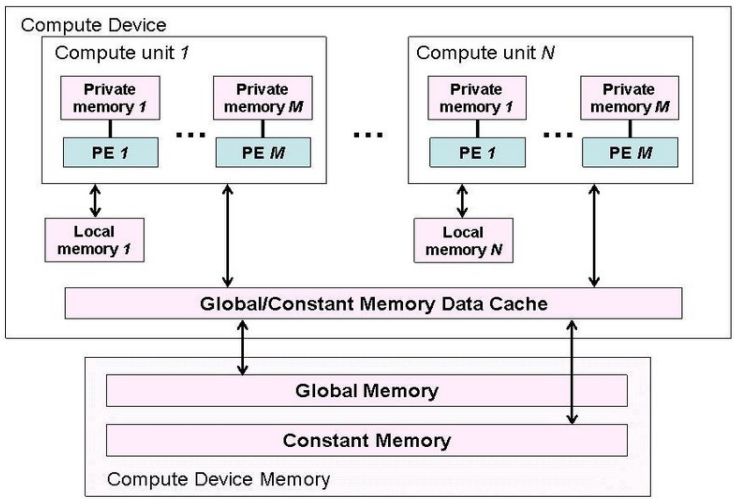

Figure 3.3: OpenCL Memory Model

Global Memory

This memory area is accessible to all work-items and they can perform read-write operations on this region. In OpenCL-based applications, usually host writes input data for work-items in this region. After computation is complete, results are written back in this memory area by kernel for the host.

Constant Memory

This part of memory is also accessible globally to all work-items but Read-only access. This memory is allocated and initialized by host.

Local Memory

This part of memory is local to each work-group. It is accessible to all work-items inside that work-group. work-items use this memory to communicate within a work-group.

Private Memory

In this memory resides the local variables for a kernel instance. Each work-item has its own private memory area and it only visible to that work-item.

3.0.4 Programming Model

OpenCL mainly benefits from two types of parallelism; data parallelism and task parallelism. In data parallelism, each work-items inside a work-group performs the same operation but on different data. It falls under the category of Single Instruction Multiple Data(SIMD) and Single Program Multiple Data(SPMD) streams. Task parallelism is where multiple large kernel instances contain a single work-item that are executed concurrently. GPUs are mostly considered suitable for data parallelism, however due to its architecture FPGA can take benefit from both types of parallelism.

OpenCL platforms usually support four types of memory systems. Memory models can be described as following based on where they are located.

Global Memory

It is the external memory for FPGA. It can be either one of Double Data Rate (DDR) Synchronous Dynamic Random Access Memories or a Quad Data Rate (QDR) Static Random Access Memory (SRAM). It usually has large capacity but also long latency.

Constant Memory

It is also considered as global memory but it contains part of data that does not change during application execution.

Local Memory

It resides inside FPGA and has lower capacity but higher bandwidth and less latency. subsubsectionPrivate Memory It is assigned to the on-chip registers on FPGA and has

lowest latency but very high bandwidth.

3.1 Modeling Software workflow

As the goal of this work is hardware generation to accelerate execution, the OpenCL code is translated into Register-Transfer-Level (RTL) hardware. High level synthesis(HLS) tools enables us to perform direct translation of a design modeled in some high level language into RTL design. The translation of OpenCL kernels into RTL design to execute on FPGA devices is performed through HLS tools. RTL version is obtained either in Verilog or VHDL language. The HLS tools used here are from Xilinx.

3.1.1 Xilinx Vivado HLS

It is an HLS tool provided by Xilinx for synthesis of C++/OpenCL/C and SystemsC based designs into RTL designs[7]. The transformation is performed through automated tools which consists or three major stages;

- **Operation Scheduling:** This stage performs scheduling of operation performed each clock cycle. It takes into account clock frequency and also latency of each operation to be performed. Depending on the target frequency, the tool may schedule more then one operations per clock cycle if it meets the design constraints.

- **Resource binding:** This stage is about assignment of resources to each scheduled operation. Knowing the target platform, Vivado HLS tool performs this operation in an optimized way.

- **Control logic:** Control logic is generated for the implemented as a finite state machine to take care of scheduling and binding.

After these stages has been done, Vivado HLS then creates an optimal implementation based on the input design constraints. The performance metrics reports give information about the timing and area estimation of implemented design. User can then adjust the input constraints accordingly to reach the best optimized implementation. After possible optimization has been done, the design can be exported into a synthesized block implementing the kernel functionality. This step is followed by a logic synthesis step which transforms the RTL design into a netlist. Implementation phase performs the place and route for the target device. After this stage, the design is converted into a bitstream which represents the the resources used and interconnection among them in binary format. This bitstream file is then combined with other platform files to form Hardware Platform Specification File (*.hdf) which is used to program the FPGA fabric.

3.1.2 SDAccel Development Environment

It is a software tool from Xilinx to analyze the host code and cross-compiler kernel management for FPGAs[8]. Is performs the standard compilation and linking for host and device(kernel) codes. Build process of SDAccel environment is visualized in Fig.3.4

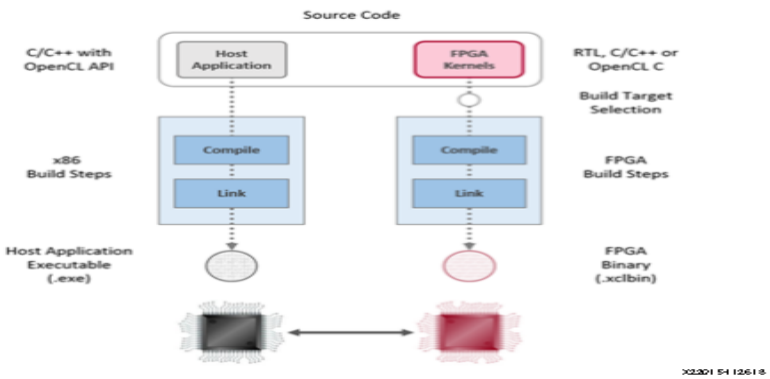

Figure 3.4: SDAccel Host/Device code building process[1]

The host code compilation is performed using xccp which is a varient of GCC from Xilinx. The host code compilation results into a host executable application. The FPGA

specific code called kernel (C/C++ or OpenCL based) is compiled using Xilinx *"xocc"* compiler. The compiler generated object files are then combines with FPGA platform files producing **.xclbin* binary file.

SDAccel provides three different types of build targets; *Software Emulation*, *Hardware Emulation* and *System*. The code functionality verification and source-level debugging can be performed in a much faster way using software emulation. Hardware emulation gives more accurate view of RTL implementation behaviour.

Chapter 4

Link Layer Simulator Design

The main part of this thesis work is to optimization and accelerate NR channel model used for System Link Simulation. The channel model used in this design is a Space Channel Model (SCM) described and calibrated in[6].Tapped Delay Line(TDL) and Cluster Delay Line(CDL) description of SCM are used for Link Simulator. There are certain requirements which characterises this channel model.

- **Mobility/Spatial consistency:** All channel characteristics for this model geometry specific and they change continuously as the UT moves. With the traditional channel model, it is very difficult to achieve full consistency because cluster locations and visibility regions are not defined.
- **Path-losses and diffusion:** This may lead to a totally different channel characteristics for propagation[14]. SCM channel model introduces sub-paths around the dominant component which can be used to model diffuse tail of propagation paths.
- **Large antenna arrays:** When using a large number of antenna arrays, an accurate model is required to track the directional characteristics. They are subject to large scale variations in propagation parameters. This model uses 20 different sub-paths which when resolved in angular domain can provide information

about change in parameters.

4.1 summarises different steps involved in this channel model.[6]

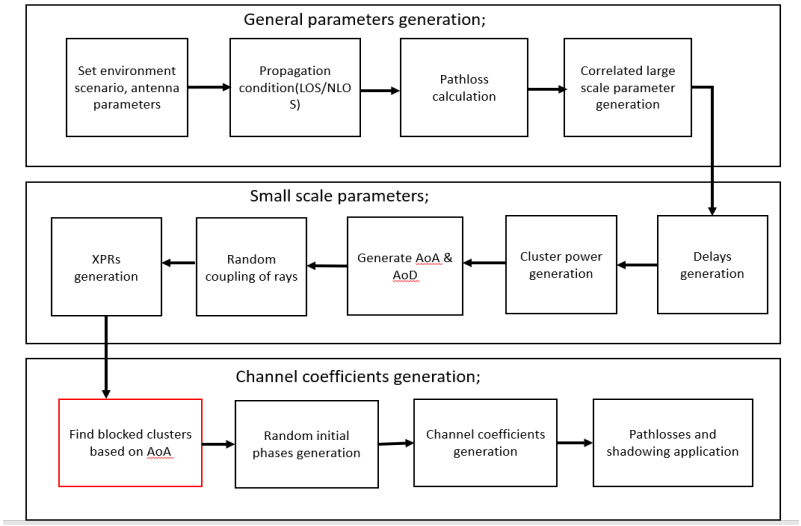

Figure 4.1: Channel Model for 5G

The steps mentioned in the above figure are explained in details in the following sections.

4.1 Environment,antenna array parameters and network layout

As this design supports different scenarios for setting the environment, we have to choose one while setting it. Each of these layouts has its own parameters to set for these scenarios. These parameters include Cell layout, height of BS antenna,user terminal(UT) location, UT mobility, minimum UT distance and its distribution.

4.1.1 Urban Micro(UMi)-street canyon and Urban Macro(UMa)

This scenario is for Urban Micro and Urban Macro environment. UMi is focused on densely populated areas e.g. city center where most of the UT are probably smart phones. The different parameter settings for these environment are listed below;

Urban Micro(UMi)-street canyon

- Cell layout:Hexagonal with 19 micro sites and 3 sectors per site 200m apart
- BS antenna height: 10m
- UT Location: Outdoor and indoor,LOS and NLOS
- Indoor UT ratio: 80%
- UT mobility: 3km/h
- Minimum UT distance: 10m
- Distribution UT: Uniform

The above environment settings will be used when Urban Micro(UMi)-street scenario is selected. The same is the case for UMa but with different values of environment parameters.

Urban Macro(UMa)

UMa scenario is also focused on urban area but less populated. In these areas, they user terminals may be moving vehicles mostly. For this scenario of environment, the parameter values are changed in some cases. The values considered for this case are;

- Cell layout:Same as in UMi scenario but 500m apart
- BS antenna height: 25m

- UT Location: Same as in UMi scenario
- Indoor UT ratio: 80%
- UT mobility: 3km/h
- Minimum UT distance: 35m
- Distribution UT: Uniform

Rural Macro(RMa)

This scenario focuses on rural areas for larger and continuous coverage. In rural areas, UE elements are not spaced closely as those in urban areas, and hence more area will be covered by the same BS element. The key highlighting feature of this scenario is continuous wide area coverage and thus supports UT moving at high speeds. Noise and/or interference is limited in scenario by using macro Transmission Reception Points(TRPs). Parameters considered for this scenario are;

- Cell layout:Same as in UMi and UMa scenarios but at 1732-5000m
- BS antenna height: 35m
- Carrier Frequency: up to 7Ghz
- Indoor/Outdoor: 50% indoor and 50% moving UT
- Minimum UT distance: 35m

The above scenarios consider broader areas and UT at longer distances. To consider the environment inside the buildings, we have to different parameters and therefore a separate environment is needed. The following scenario considers the indoor environment.

Indoor-office

This scenario considers user elements inside the buildings. They are not directly in Line-of-Sight with the BS element and hence uses indoor antenna configuration. For this scenario of environment, the parameter values are changed in some cases. The values considered for this case are;

- Cell layout:Equal to room size(WxLxH) and each 20m apart
- BS antenna height: ceiling(3m)
- UT Location: LOS and NLOS with height of 1m
- UT mobility: 3km/h
- Minimum UT distance: 0m
- Distribution UT: Uniform

The Indoor-office environment can be open-office or mixed-office where the only difference is in line-of-sight(LOS) probability.

4.1.2 Antenna modelling

The next step in this channel design is antenna modeling. This part of the design captures the antenna array structures which are considered for calibration.
The base station antennas are modelled as an array of uniform rectangular panels. They are spaced uniformly at a distance in vertical and horizontal direction. The antenna panel can either be uni-polar or bipolar. Fig.4.2 visualizes the antenna configuration in a 2D plan.

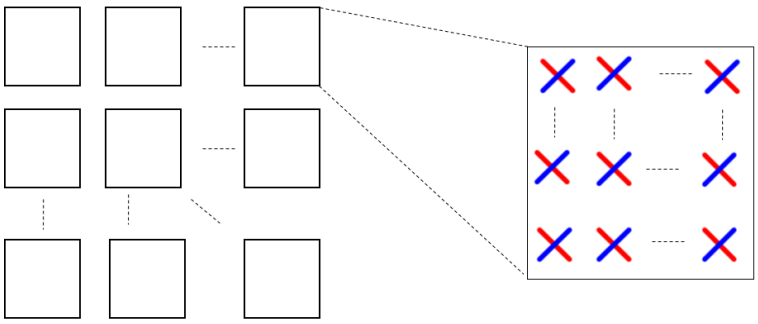

Figure 4.2: Antenna configuration

4.1.3 Pathlosss and LOS probability

In this stage, The pathloss and LOS probabilities are calculated. This will help in focusing the beam and setting the channel parameters accordingly. Fig. 4.3 shows distance definitions for two different scenarios. On the left, the out-door scenario is considered where UT and BS are in LOS. The right side of this figure visualizes the in-door scenario where the UT and BS are not in LOS and hence NLOS conditions are applied.

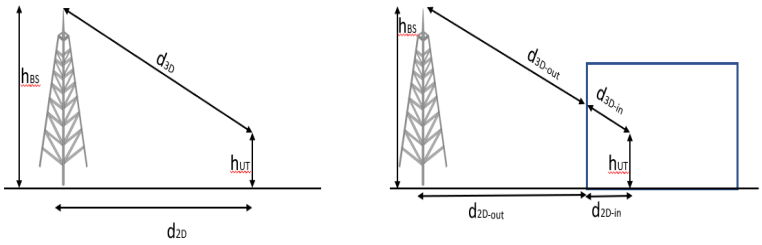

Figure 4.3: Distance definition for in-door and outdoor scenarios

4.2 Workflow of channel model

- **Environment Parameters:** The first step is selection of one the scenarios(UMi-Street Canyon,UMa, In-Office or RMa) to set the environment accordingly. Next step is the selection of global coordinate system. Number of Base stations and user terminal are set. Next is determining angle of arrival(AoA), zenith angle of arrival(ZoA),angle of departure(AoD), and zenith angle of departure(ZoD) in 3D space. This will give information about array patters for BS in UT in global coordinate system. Next step is determining the speed of UT and its direction of motion.

- **Large scale parameters:** This stage is about assigning different propagation conditions. These condition include LOS/NLOS and in-door/outdoor. Path-losses are calculated for the selected propagation conditions. Large scale parameters are then generated taking into account the correlation. These parameters include angular spread, delay spread and shadow fading.

- **Small scale parameters:** First step in this part is generation of cluster delays. Delays for LOS conditions are scaled to compensate LOS peak addition effect. This step is followed by cluster power generation. They depend on the delay distribution for different environment scenarios. Cluster with power -25db lower than maximum cluster power are eliminated. AoA and AoD are then generated for elevation and azimuth. AoA and AoD angles within a cluster are then coupled randomly. The same is done for azimuth angles. Cross polarization of power ratios is then generated for each ray within a cluster.

- **Coefficients generation:** This is the part where actual coefficient generation takes place. It computes the impulse response of the channel taking into account all those input parameters calculated in previous steps or from external environment. Once the impulse response of the channel is known, it can be applied to

the channel design.

The Cluster Delay Line (CDL) used in this design is similar to Space Channel Model (SCM) model but it simulates only one link generating a CDL which represents the link channel. The CDL model allows better representation of beam-forming as compared to Tapped Delay Line (TDL) because the direction of signal in space is modeled. TDL model is based on the impulse response of the channel while CDL uses description of arrival and departure directions in space.

Chapter 5

Implementation and Optimization

The chapter explains the steps and methodology adopted to perform optimization and acceleration of channel design for 5G NR. The starting point of thesis was a model developed in C environment and co-simulated with matlab. To optimize the design, it was required to have it working in a homogeneous environment to take benefits of High-Level-Synthesis optimizations.

5.1 Initial Design

The input design was based on co-simulated model. Co-simulation here meas that a part of model runs in one environment and reaming in some other environment. The two environments used in this design are matlab and a C++ based design. They communicate with each other through MEX environment.

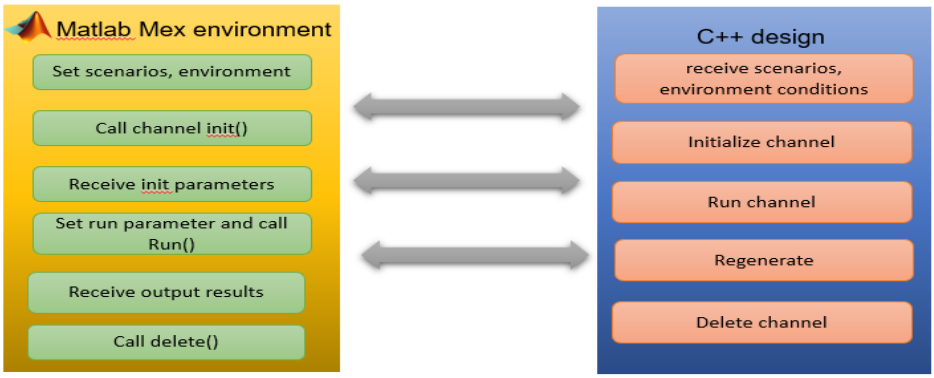

Figure 5.1: MEX Matalb and C++ co-simulation concept

The channel parameters are set using matlab environment. The channel design implements the underlying base functionalities of the channel which is implemented in C++. The co-simulated model consists of different steps to perform the simulation.

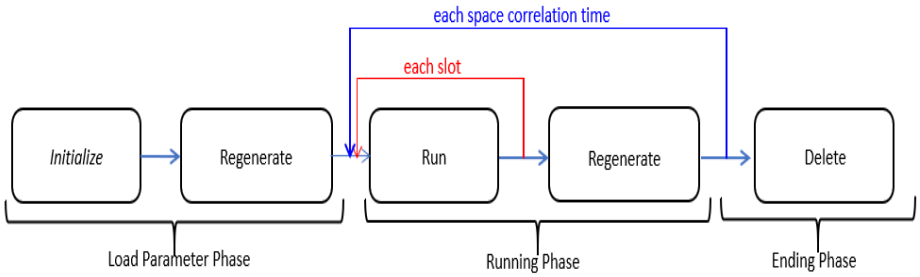

Figure 5.2: Channel Model in Matlab and C++ co-simulation environment

5.1.1 Initialization

In the initial design, this part was totally performed in matlab. The main functionality of this part is initialization of parameters. It takes as input the environment conditions and sets the channel parameters accordingly. The input environment conditions

include number of polarization, AoA,AoD, CDL type, correlation model, delay spread model, number of transmitter and receiver antennas and other environment specific parameters. These input conditions are then used to calculate channel parameters such as number of fading samples in each NR slot, user element direction type, and channel state, antenna configuration, and angular direction info. These parameters are then passed to channel design modeled in C++ which performs the initialization of the channel. Random variables are set and coefficients are initialized accordingly.

5.1.2 Regenerate

This steps performs the regeneration of channel for new CDL directions. Considering the new correlation and arrival and departure angle, it determines the new location of UT in space and sets the channel parameters accordingly.

5.1.3 Run

This step performs the actual running operation of channel. Input to this step environment conditions, channel parameters, input symbols and current time. The design first sets its parameters according to these input conditions. It then retrieves CDL parameters and start the computation. The new coefficients and channel parameters are initialized for newly calculated conditions. When the parameter calculation is done, it applies them to the channel link. After the channel application phase is completed, it returns the output vectors which contains impulse response of the channel.

The channel keeps running and calculates channel coefficients and parameters for each slot. When the time advances, the channel recalculates the correlation parameters and calculates new coefficients and parameters for new time instance. This computation is performed for each slot. The channel regeneration takes place each time there is a change in space time correlation.

5.1.4 Delete

It is the ending phase of a channel link. When the link is no more required, the allocated space and other resources can be freed for use in other slots.This part of the model is invoked with some input parameters which tells the model which slots are to be removed. The model then finds the resources which are related to that specific link. It clears up the coefficients calculated for that specific slot and also old coefficients associated with it.

5.2 Isolating the simulation environment

The next step was to remove the matlab dependency from co-simulation environment. The co-simulation model was not optimized and user a lot of resources and time. Also a good level optimization was unable to achieve with this type of design. One of the possible solution to this problem is to identify these performance critical points and accelerate them via FPGA. The design required some modification in such a way where it can get maximum benefit of hardware accelerators. The initialization function was using some native functions from matlab which were required be implemented in C++ counterpart with same functionalities. To form a native C++ simulation environment,the channel design required some modification. The interface to the parts of the channel implementing different functionalities was changed to work best with the C++ data structures. Different functions translated from matlab script into C++ environment with their respective functionalities are listed below;

- **Generic CDL configuration**

  This is initial step and setting parameters for the model. This function gets environment conditions as input. It configures the channel parameters according to that specific environment conditions and selected scenarios.

- **Initialization function:**

This function performs the channel parameter initialization for channel from environment conditions. The input environment conditions are specified by a top interface which also works as a test-bench. The top interface mimics the external environment conditions. This function then initialized the channel parameters according to that conditions.

- **Channel execution:** This is the counter part of matlab script responsible for channel execution for different slots. It starts from the state set by initialization function and then computes impulse response of the channel. This response is then used to set the channel parameters accordingly. When the time correlation changes, the channel need to recalculate the response. This execution continues for each slot in each time sample.

- **Deletion:** This part is related to cleanup after the link has been terminated. When I certain slot is no more required, the resources assigned to it needs to be freed. This part of the model releases the memory occupied by structures of the slot that we want to delete.

5.2.1 Performance critical points identification

To optimize certain model, it is required to identify performance critical points first. By analysing the model execution in co-simulated and also isolated environment, two main critical points were identified.

- TDL coefficients computation

- Coefficients application to channel

After the identification of these points, the next step is acceleration and optimization of these operations.

5.2.2 FPGA Target Platform

The FPGA platform used for profiling of this application is Kintex UltraScale KU115 form Xilinx. These high performance FPGAs based on monolithic and stacked silicon interconnect (SSI) technology. Some of the key highlighting features of these devices are;

Name	**Number**
System Logic Cells	1,451,100
CLB Flip-Flops	1,326,720
CLB LUTs	663,360
Maximum Distributed RAM (Mb)	18.3
Block RAM Blocks	2,160
Block RAM (Mb)	75.9
DSP Slices	5,520

5.2.3 TDL coefficients computation

The part consists of a combination of nested for loops which computes coefficients for filters inside the channel model. These channel needs to be recalculated each sample time. To accelerate this part of design, the computation function is converted into an OpenCL kernel. The input data arrays are transferred into high speed global memory of FPGA device and the computation is performed in parallel using real hardware resources.

CPU Implementation

The pure CPU implementation consists of a structure of nested ***for*** loops. The inner most performs the computation part and updates the coefficients value in a data structure.

Average CPU execution time: *60.7ms*

5.2.4 FPGA implementation

The coefficient function was converted into an OpenCL kernel. As the initial design was written keeping in mind only functionality, the direct translation was very un-optimized.

Basic FPGA implementation without optimization

This part of the kernel contains *5 nested loops*. In this implementation, the top 4 loops were flattened into 1 loop excluding the inner most loop. None of these loops were pipelined. The following results were obtained for this implementation.

Parameter	**Value**
Clock	3.753*ns*
Latency(min)	6001345*cc*
Latency(max)	23724225*cc*
Pipelined	No

Comparing this execution profile with the CPU version, we get the following results;

- **Average kernel execution time:** 55.7*ms*
- **Speed-up :** 1.088*times*

Implementation with all loops flattened

All the loops in kernel were flattened. The synthesis result we got for that kernel are reported below;

Parameter	**Value**
Clock	4.48*ns*
Latency(min)	6862465*cc*
Latency(max)	16348033*cc*
Iteration latency	4662 ~11106
Pipelined	No

Comparing this execution profile with the CPU version, we get the following results;

- **Execution time:** 30.7*ms*
- **Speed-up :** 1.97*times*

Even the very unoptimized version of kernel performed better than a pure CPU centric implementation.

Implementation with all loops flattened and pipelined

In this implementation, all the loops were flatted into one loop. The design was pipelined by removing any dependencies. The following table shows results for this implementation.

Parameter	Value
Clock	3.035*ns*
Latency(min)	147571*cc*
Latency(max)	147571*cc*
Iteration latency	375
Initiation Interval	5
Pipelined	Yes

Comparing this execution profile with the CPU version, we get the following results;

- **Execution time:** 0.447*ms*
- **Speed-up :** 135.5*times*

At this stage, we have a very optimized version of kernel. At this stages, all the arrays reside in global memory. This design can be even more optimized by limiting the number of global memory accesses.

Pipelined implementation with BRAM

To limit the costly accesses to global memory, some of the arrays were moved to BRAM. This resulted to even further improvement of results. The following table shows results for this implementation.

Parameter	Value
Clock	2.971*ns*
Latency	88863*cc*
Iteration latency	376
Initiation Interval	3
Pipelined	Yes

Comparing this execution profile with the CPU version, we get the following results;

- **Execution time:** 0.263*ms*
- **Speed-up :** 230*times*

5.2.5 Channel application

This is the part of the model where the coefficients computed earlier are applied to channel parameters. It also consists of *nested for loops* structure which can be best optimized in OpenCL kernel. This kernel is mainly responsible for applying the channel parameters calculated in previous steps. It then computes the final output response.

To optimize this part, the same methodology was adopted as in case of coefficient computation kernel. Arrays with small size was copied into BRAM for fast access. Some of these arrays were even partitioned according to access pattern and are then mapped on registers. Some of the data structures were too large to fit on BRAM of the device considered here. To overcome this, the access pattern was modified for that data structures. Then only a part of it was moved to BRAM which is being used by the kernel. In this way, we reduce number of global memory accesses.

Chapter 6

Conclusion

6.1 Summary

The work of this presented thesis is to design a hardware accelerator for the channel model simulator in order to reduce the execution time and increase the throughput of the design. Fast and accurate simulators are required to model the exact behaviour of the system in real- world scenario. This Hardware accelerator is being designed for .

6.2 Results and future work

During this work, as explained in Chapter 5, different optimizations were performed for to evaluate the latency and execution time.

The design makes use of complex number operations were are not supported yet in OpenCL. To overcome this limitation, customized data structures and functions were implemented to perform arithmetic operations on complex numbers.

Another optimization performed was to reduced access requests to global memory. Some of the arrays were copied into BRAM which reduced the costly accesses to global memory and hence enhance performance.

Performance can further be improved by increasing the number of read ports and partitioning the arrays. An even further improvement can be mapping of small arrays into registers.

All these operations and optimization were performed for *double* data type. Performance can even further be improved by using fixed point operations were higher precision is not required.

Bibliography

[1] Sdaccel compilation flow and execution model. [Online]. Available: https://www.xilinx.com/html_docs/xilinx2019_1/sdaccel_doc/uqp1519743299633.html

[2] S. Hilton, "Machine-to-machine device connections: worldwide forecast 2010–2020," *Analysys Mason Report*, 2010.

[3] H. Jung, "Cisco visual networking index: global mobile data traffic forecast update 2010–2015," Technical Report, Cisco Systems Inc. 2011. Available online: https://www . . . , Tech. Rep., 2011.

[4] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[5] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The next generation wireless access technology*. Academic Press, 2018.

[6] "Tr 138 901 - v14.0.0 - 5g; study on channel model for frequencies from 0.5 to 100 ghz (3gpp tr 38.901 version 14.0.0 release 14)," https://www.etsi.org/deliver/etsi_tr/138900_138999/138901/14.00.00_60/tr_138901v140000p.pdf, (Accessed on 09/17/2019).

[7] Vivado design suite user guide: Getting started (ug910). [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug910-vivado-getting-started.pdf

[8] Sdaccel development environment. [Online]. Available: https://www.xilinx.com/

products/design-tools/software-zone/sdaccel.html

[9] "Intel® arria® 10 device overview," https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/a10_overview.pdf, (Accessed on 09/16/2019).

[10] "Intel arria 10 core fabric and general purpose i/os handbook," https://www.intel.com/content/www/us/en/programmable/documentation/sam1403483633377.html#sam1403481243928, (Accessed on 09/16/2019).

[11] https://www.khronos.org/registry/opencl/specs/opencl-2.1.pdf. [Online]. Available: https://www.khronos.org/registry/OpenCL/specs/opencl-2.1.pdf

[12] Opencl overview - the khronos group inc. [Online]. Available: https://www.khronos.org/opencl/

[13] J. van de Loosdrecht, "Accelerating sequential computer vision algorithms using commodity parallel hardware," Ph.D. dissertation, 09 2013.

[14] J. Medbo, K. Börner, K. Haneda, V. Hovinen, T. Imai, J. Järvelainen, T. Jämsä, A. Karttunen, K. Kusume, J. Kyröläinen, P. Kyösti, J. Meinilä, V. Nurmela, L. Raschkowski, A. Roivainen, and J. Ylitalo, "Channel modelling for the fifth generation mobile communications," in *The 8th European Conference on Antennas and Propagation (EuCAP 2014)*, April 2014, pp. 219–223.