OULUN AMMATTIKORKEAKOULU

Pauli Nikula

RF FRONT-END CUSTOMIZATION TOOL FOR 5G MILLIMETER-WAVE RADIO

RF FRONT-END CUSTOMIZATION TOOL FOR 5G MILLIMETER-WAVE RADIO

Pauli Nikula
Bachelor's Thesis
Autumn 2019
Information Technology
Oulu University of Applied Sciences

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutusohjelma, laite ja tuotesuunnittelun suuntautumisvaihtoehto

---

Tekijä: Pauli Nikula
Opinnäytetyön nimi: RF Front-End Customization Tool for 5G Millimeter-Wave Radio
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Syksy 2019 Sivumäärä: 52

Opinnäytetyön aiheena oli PC:llä toimivan graafisen suunnittelutyökalun toteuttaminen. Työkalua käytetään 5G FR2-teknologiaa käyttävien mobiililaitteiden radiotaajuuskomponenttien (RF) konfiguraatio- ja ohjausdatan generoimiseen. Työn toimeksiantaja oli MediaTek Inc. Opinnäytetyössä tutustuttiin myös yleisesti mobiililaitteen radiotekniikkaan, RF-komponenttien ohjausdatan kustomoinnin käsitteeseen sekä graafisen PC-työkalun luomiseen.

RF-komponenttien konfiguraation kustomointityökalu toteutettiin Java-ohjelmointikielellä, käyttäen Java Swing-grafiikkakomponentteja sekä erillistä kirjastoa jolla toteutettiin puukaavio-tyyppinen piirtoeditori. Suuri osa työtä oli myös selvittää uuden FR2-radion spesifikaatioita ja millaisia vaatimuksia se loi kustomointityökalulle.

Opinnäytetyön tuloksena syntyi PC:llä toimiva graafinen työkalu, jolla pystytään piirtämään mobiililaitteen FR2-radion komponenttikonfiguraatio. Työkalussa on myös koodigeneraattoriosa, joka generoi piirroksesta kooditiedostot mitä voidaan käyttää mobiililaitteen RF-ohjausohjelmiston myöhemmässä käännöksessä.

Asiasanat: RF, FR2, 5G, radio, mobiiliverkkoteknologia, RF-komponentti, kustomointityökalu, ohjausdata, komponenttikonfiguraatio

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Option of Device and Product Design

Author: Pauli Nikula
Title of thesis: RF Front-End Customization Tool for 5G Millimeter-Wave Radio
Supervisor: Kari Jyrkkä
Term and year when the thesis was submitted: Autumn 2019 Number of pages: 52

The subject of this bachelor's thesis was creating a graphical design tool for customizing the RF component layout and RF component configuration data generation of 5G FR2 mobile equipment. Part of the subject was also to explain the grown complexity of mobile phone RF components, what RF component customization means, why it is needed and what it takes to create a graphical-interface tool for a PC. The thesis was commissioned by MediaTek.

The customization tool was created with the Java programming language, using Java Swing graphics and a separate graph drawing library. A major part of the work was studying the RF system specification and having discussions with related engineering teams to establish the requirements of the customization tool.

As a result a new graphical design tool was created. It consists of a drawing editor part and a code generator part. The tool can work in a close relationship with the mobile phone RF control software in order to reduce the efforts in adapting the software to the mobile phone's RF front-end circuitry.

Keywords: 5G, FR2, RF component customization, graphical design tool, software adaptation

PREFACE

I had a great opportunity to write this thesis to finish my Bachelor's Degree at MediaTek Oulu site during the summer of 2019. I had been working as a trainee at Oulu site for quite some time already and the experience has been very interesting and useful. I have really enjoyed working with the latest technology, hardware and programming. This trainee experience and the writing of this thesis has been a pinnacle for me that I dreamed of when I first started building electronics and programming as a hobby.

I want to give special thanks for my manager Asko Ruotsalainen who has given me this opportunity, good advice and time flexibility to write this thesis during other work projects. I want to thank my thesis instructor and supervisor Kari Jyrkkä for his advice and guidance during the thesis process. I also want to thank lecturer Kaija Posio for instructing with the grammar of the thesis.

Finally I would also like to thank the MediaTek RFSW team members for providing feedback and advice for my thesis work and also for the great working atmosphere that I have been able to enjoy while working here.

Oulu, 14.8.2019
Pauli Nikula

CONTENTS

ABBREVIATIONS

Bandwidth	A measure of how much of a radio channel is used by a radio transmitter device
BWP	Bandwidth Part Adaptation, a 5G specification
CC	Component Carrier, a main data carrier signal
FFT	Fast Frequency Transform, a signal processing algorithm used in 4G/5G
FR2	Frequency range 2, specified in 5G
HPUE	High-Power User Equipment
IFIC	Intermediate-Frequency IC, a RF mixing and control IC used in mmW RF
Java Class	A basic code unit in Java programming
LNA	Low Noise Amplifier, used in RF receiver circuits
MIMO	Multiple Input Multiple Output, radio technology
mmW	Millimeter-wave radio technology
OFDMA	Orthogonal Frequency Division Multiple Access, channel sharing technique
PA	Power Amplifier, used in RF transmitter circuits
PS	Phase Shifter, used in mmW RF circuits
RF	Radio frequency, electrical signal type
RX	receiver, or receiver-side
RFFE	Radio frequency front-end, RF component section of radio device
SoC	System-on-a-chip, the main control chip in a mobile device
Struct	A data type in C programming language which is a structure of basic data types
TDMA	Time Division Multiple Access, channel sharing technique
TX	transmitter, or transmitter-side
UE	User Equipment, the mobile user device, most commonly the mobile phone
WCDMA	Wideband Code Division Multiple Access, channel sharing technique
5G	5th generation mobile network

1 INTRODUCTION

Public mobile networks and their underlying technology have been evolving for over 30 years. As of 2019, the mobile radio technology has seen already the 5th wave of new specifications. Naturally the evolving technology can be seen in the growing complexity of the devices. One example of this is the UE (User Equipment) device's radio. The RF circuitry of the UE has gradually changed to be more and more complex. The RF components require software control but the same control software does not work for all mobile phones. There are differences in the configuration of the RF circuitry, depending on the phone model and region where the phone is going to be used.

MediaTek designs cellular RF systems, and one part of the system is controlling an external RF component circuitry. The control software uses special configuration files to read the details of the RF component configuration. These configuration files have previously been created manually and it requires a lot of error-prone work. Every time a small change is made to the configuration, a manual modification is needed to the configuration files. To overcome this problem, a common RF component customization framework was created by MediaTek engineers. The customization enables a flexible configuration of the modem control software according to the RF components. One part of this framework is a PC based design tool. The tool can be used to auto-generate software setting files according to the drawing of the RF components that the customization designer has made. This greatly reduces the effort in deploying the modem and RF configuration to a new mobile phone design. For supporting the 5G FR2 mobile, a new customization tool was needed, which is the subject of this thesis. The RF front-end components and their layout are significantly different than in previous 3G and 4G devices, which poses a challenge for the customization tool. Part of the subject is to also explain the roots of RF component circuitry in different mobile generations and where the need for a customization tool originates.

2 EVOLUTION OF PHONE RF FRONT-END

In this chapter the evolution of mobile network technologies from the RF circuitry aspect is examined to establish understanding of the RF Front-End (RFFE) layout customization.

2.1 Cellular modem

At the heart of the radio signaling in a mobile device is a modem. It is a processor which can control the RF front-end components and process the radio signal in order to turn it into useful data for the mobile device operation. Nowadays, the modem is an integral subsystem of the main system chip found in smartphones. There are numerous mobile technology manufacturers that develop their own modem subsystem, including MediaTek. During the generations of mobile networks, the modem has evolved and it becomes more complicated each year.

2.2 RF Front-End

The RF front-end (RFFE) is a circuitry in a mobile device handset which exists between an RF transceiver IC and antennas. The RF front-end is comprised mostly of modular components such as power amplifiers (PAs), low noise amplifiers (LNAs), switches, duplexers, filters, and other passive devices. (1.)

It could be said that the modem chip is the brain which can process the data that is sent out from the mobile phone and received. It also manages all synchronization and control data between the mobile and cellular network. The modem calculates and decides what to send and when. The RF front-end components can be considered mostly passive components that the modem controls and uses to convert the digital data into electromagnetic waves so that the wireless communication can happen. In the MediaTek case, MediaTek is the company that provides the modem chip for a phone manufacturer. The manufacturer can use the front-end components selected by MediaTek but they also have an option to choose their own components (FIGURE 1.).

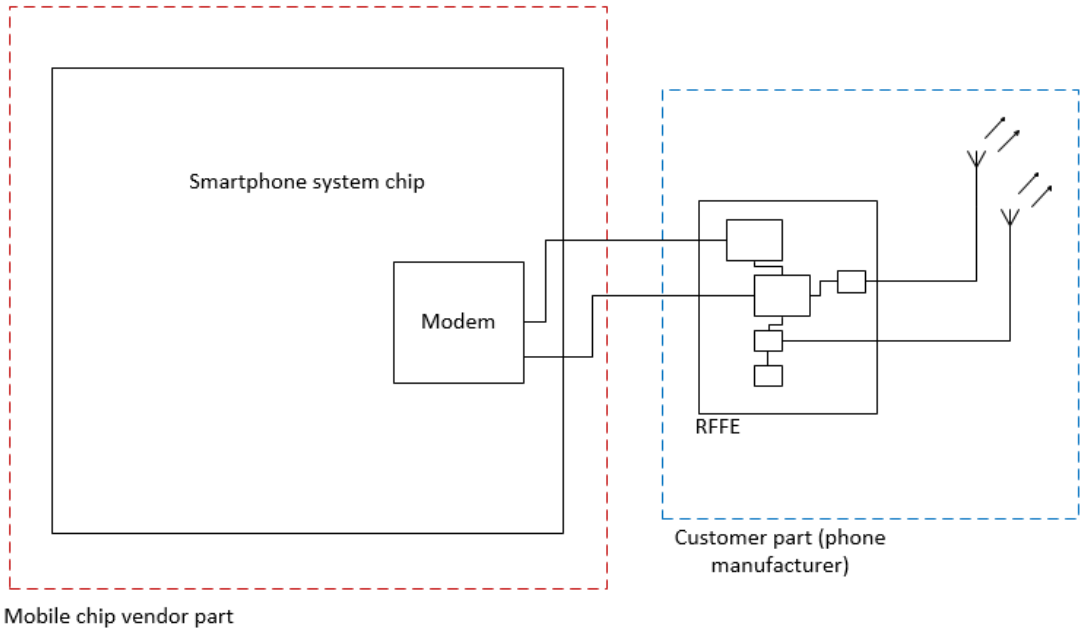


FIGURE 1. An overview of mobile device radio parts

2.3 Basic RF component types

In order to send or receive radio waves, a mobile device needs some basic building blocks (FIGURE 2). The first is the source of a modulated data signal. The data signal is also called a “baseband signal”. Modulation means that a bit sequence is transferred to a sinusoidal wave by making a change to the wave at certain points of time. This results in a signal which can carry the ones and zeros and the modem outputs this signal. (2.) For transmitting it to the air, the next component is an RF mixer. It is used to mix the baseband signal to the carrier wave or to an intermediate-frequency wave (depending on the technology) which is then mixed to the carrier wave. The baseband signal frequency is in the order of 1-100 MHz depending on the technology used but the carrier wave operates at the Gigahertz range. After the mixer component, the baseband signal is “carried” by the carrier wave, again suggested by the name.

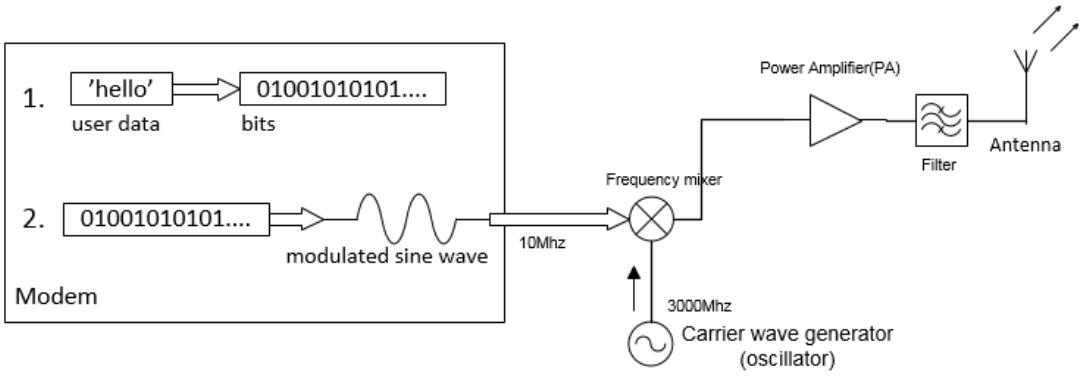


FIGURE 2. A simplified diagram of modem and other basic RF components in a digital radio transmitter

At this point the signal is almost ready for transmitting but it needs to be amplified. The reason is that the baseband signal and the mixer output signal are never very strong. A separate Power Amplifier (PA) component is used to amplify the signal before it is sent to the antenna. (3.)
For receiving, the signal direction is reversed. After the antenna picks up the signal from air, it needs to be filtered and amplified. This is done by an LNA component (Low-Noise Amplifier). After the LNA, there can be additional amplifying stages and then the signal is input again to an RF mixer. This time the mixer works the other way round and extracts the low-frequency baseband signal from the carrier wave. Then the baseband signal is input to the modem chip where it is further processed to extract the data for use.

As the signal goes from the modem chip, through the components, to the antenna, it forms a signal path. Here 3 basic types of components are introduced, the RF mixer, the PA and the LNA. It sounds very simple but it becomes more complicated when the requirement to support multiple frequency bands is introduced. All frequency bands simply cannot be supported by just one of these components. For cost efficiency reasons, RF front-end components are designed to operate in a certain frequency range which does not cover all of the bands. Thus, different bands need to use a different set of these basic components. This introduces the need of path switching components which are just called switches. The MIMO and multi-carrier technology will cause the need to have more sets of these components for each antenna path. In case of 4x4 MIMO, where there are 4 modulated baseband signals transmitted and received through 4 different antennas, it basically quadruples the number of these basic components.

2.4 1G

From the first generation mobile standards perhaps the most notable is the NMT network. It was opened in the beginning of 1980 in Norway, Sweden and Finland. The other major technology belonging to the first generation, was AMPS in the Americas (4.). NMT uses a simple analog FM modulation technique for transmitting voice, which is essentially the same method as in FM radios. It means that the speaker's voice is carried "as-is" in the carrier radio wave, without converting it into some digital keying format (FIGURE 3).

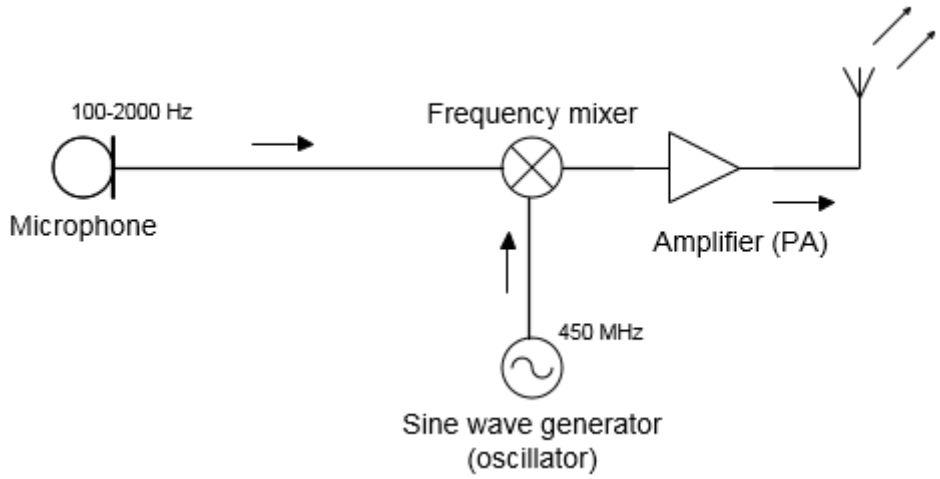


FIGURE 3. A simplified FM transmitter circuit of NMT-450 phone

NMT supported two operating frequencies, 450 and 900 MHz. From the circuitry aspect, an FM modulator or demodulator can be made from simple analog components. The speaker's voice can be input directly to the circuit from the microphone. One phone only supported the 450 or 900 MHz band at a time thus the NMT RF circuitry did not need to support mode changes.

2.5 2G

In the 1990's the GSM network was created and it quickly became the standard mobile network around the world. GSM started the 2G era and it is the most known standard under 2G. In GSM, the calls were transmitted digitally, meaning that the voice was converted to digital bit sequences which were transmitted in the radio wave. GSM used a combination of frequency division (FDMA) and time division (TDMA) channel sharing techniques, and it enabled multiple users to share the same frequency channel at the same time. At first GSM worked only on the 900 MHz band but later

a 1800 MHz “high-band” was introduced. When the first phones were designed to support both bands, it was the first time the RF circuitry of the phone started to require more complex dynamic capabilities. In total, there are 4 major frequency bands for the GSM network use: GSM900 and 1800 that are used mainly in Europe and Asia, and GSM850 and 1900 that are used in the Americas.

Another major 2G mobile network system is the cdmaOne which is used in the US. Contrary to GSM, it uses the CDMA radio access technology which is described in more detail in chapter 2.6.

FIGURE 4 illustrates the added complexity of the RF front-end in case of a dual-band phone of the GSM era. The RF signal can take a different path depending on the currently selected operating band. The PA components require software control to switch them on and off. The modem processor chip must control the components but the component control interfaces can vary between manufacturers.

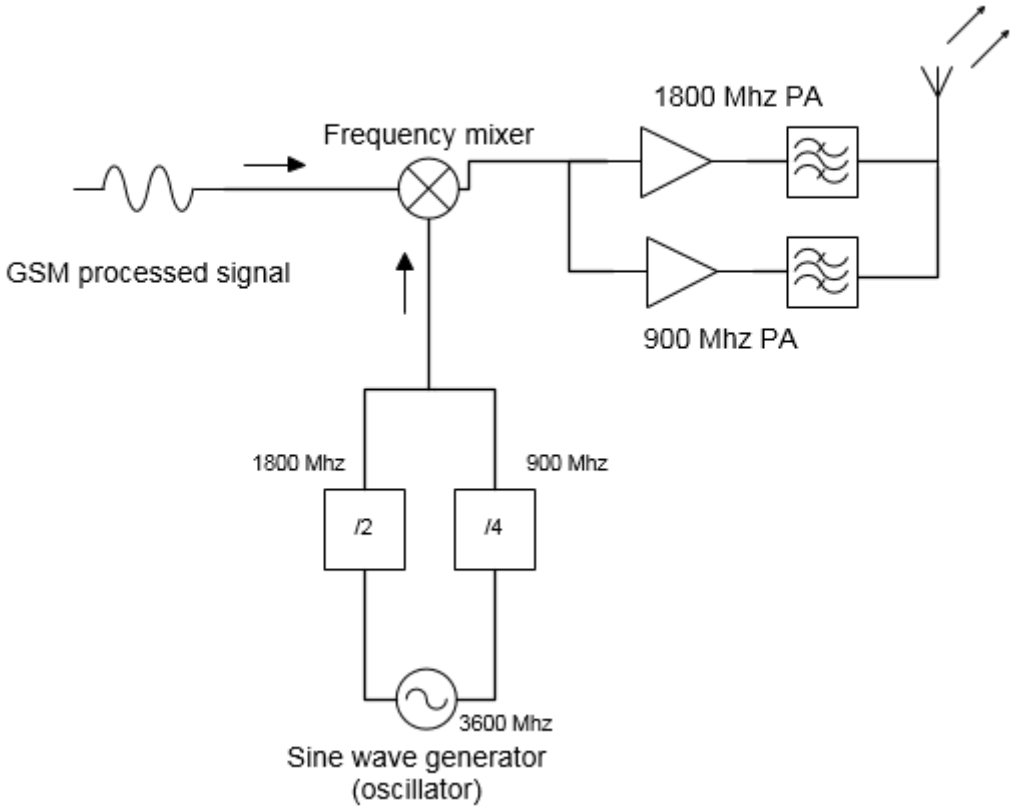


FIGURE 4. A simplified Dual-Band GSM phone transmitting circuit

2.6 3G

2G networks satisfied the needs for voice calls but it became a reality that mobile phones can also be used for much more than just voice communications and simple text messages. People wanted to access the Internet with their mobile phones, and for this purpose the data rate of GSM was simply not enough. New radio technologies were developed in purpose of achieving much higher bit rates, and this now became the 3G era in the 2000's.

CDMA and WCDMA (5.) became new channel sharing techniques in 3G systems. This was much different from the 2G TDMA. In CDMA, each user is given own special code, which is used to spread the user data signal into a larger-bandwidth signal which resembles noise. Now this spread signal is sent in the radio channel. The mobile phones are effectively shouting at the same channel each with their own language. There were different variations of the CDMA system used in different areas: WCDMA was mainly used in Europe, TDS-CDMA in China and CDMA2000 in the USA.

As 3G networks were still not going to replace 2G networks, separate frequency bands were allocated for the 3G network usage. This meant again more demand for the RF components to support different scenarios. A city area could have a 3G network but going out of the city there might be only 2G available. Like in 2G, different regions had deployed different 3G bands, so the RF front-end capabilities were different between phones sold into different parts of the world. At some point a new concept was born, which is the RF front-end configuration version. Phones started to have a version of RF front-end for Europe, North America and China.

In pursue of higher speeds, new signal modulation technologies were deployed. QPSK and 8-PSK, which were used in GSM, gave way to 16-QAM and 64-QAM. It meant totally new challenges for the digital signal processing area but actually not much for the RF components. The amplifiers and filters do not care for the modulation but it meant higher linearity requirements for the components. In the case of 2G and 3G coexistence, however, the PA component is an exception. The GSM transmission uses much more power than 3G, so the PA is separate for GSM radio access. The use of multiple carriers was also introduced during the 3G era (6.). It means that instead of just one

main carrier wave at 2.4GHz, for example, another carrier wave is used at a slightly different frequency which could really boost the data rate. For the RF front-end circuitry, this meant the addition of another full RF signal path to support the second carrier.

2.7 4G

In the 4G era, there were major changes for the radio access methods. Channel sharing techniques were developed and a technique called OFDM was put into use. In this technique, the data signal wave is constructed from multiple smaller data signals which are called subcarriers. The combination of these subcarriers is called a component carrier (CC). The OFDM is made possible by fast frequency transform (FFT) computation by the powerful signal processing hardware of today's mobile phones. OFDM again improved the data rate. It improves spectral efficiency, which basically means that more information can be transmitted with the same radio bandwidth. By using OFDM, more data symbols can be crammed to a carrier wave.

4G introduced again some new bands. (7.) Some of the bands operate close to or even above 3 GHz frequencies. Now the phone RF front-end circuits had to support 4G, 3G and 2G. The RF front-end was divided into 3 band "regions": Low-Band, Mid-Band and High-Band. Each of these regions usually require own RF components. Some manufacturers introduced Multi-Mode RF components, which combined components for different bands into same package. This helped to make the ever-increasing number of components in the circuit board smaller but the multi-mode components still required control data customization for supporting different front-end configuration versions. The multi-carrier technology introduced in 3G was developed further in 4G. In 4G it is possible to use even more than 2 carriers at once, and there are different types of this carrier aggregation (CA) (8.). Differences of total bandwidth caused by different CA schemes are also impacting the RF front-end, as different types of filters or component filter settings are needed.

In 4G the MIMO technology is used extensively. MIMO means Multiple Input Multiple Output. It is a technique of exploiting multipath propagation to send and receive more than one modulated data signal over the same radio channel (9.). This can effectively multiply the data rate because the signal streams can be distinguished from each other by exploiting the signal differences that are caused by the different propagation paths that the signal experience. This meant a new demand for the RF Front-End of mobile devices. New signal paths had to be supported that are activated at the same time (FIGURE 5).

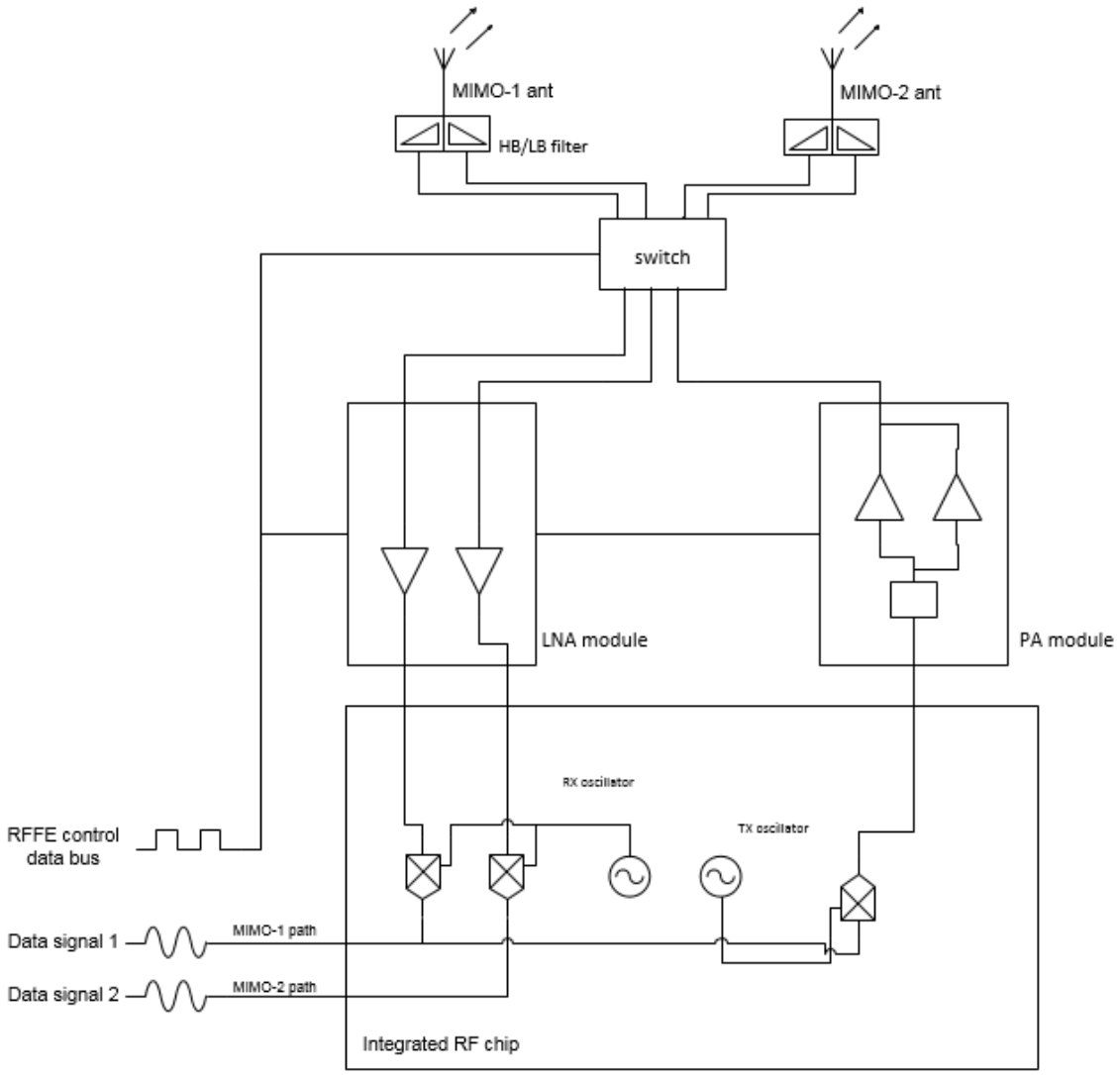


FIGURE 5. An illustration of multiple antenna paths (2X2 MIMO) and an integrated RF chip in 4G

At first the MIMO scheme began in 3G with an addition of a second antenna. The advantages of this design became clear, so next was the addition of two more MIMO antennas. These configurations became known as 2X2 or 4X4 MIMO configurations. This is only used in a receiver (RX) part. A TX side MIMO has been specified in 4G but there are no known implementations. In 5G, the TX MIMO is going to be implemented.

For simplifying the control of RF front-end components, a new standard called MIPI RFFE was created. The MIPI RF Front-End Control Interface is a dedicated control interface for the RF front-end subsystem. It enhances the control of the complex RF subsystem environment, which has very high performance requirements and can include up to 20 components such as power amplifiers,

antenna tuners, filters and switches. The interface can be used for the full range of RF components to simplify product design and to facilitate interoperability of components supplied by different vendors. (10.) Basically, the MIPI interface is just a typical serial bus with a fast clock speed. But the key aspect of MIPI was to unify the control interface of front-end components between manufacturers.

2.8 Summary

In this chapter the origins of an RF front-end layout customization tool were explained. The modem control software must be adapted to different front-end configurations with different components for 2G, 3G and 4G bands and MIMO paths (FIGURE 6.). A graphical tool is well suited for drawing the front-end layout as a graph and it is then used to generate control data for the modem software. The tool provides a centralized method to maintain different RF front-end configuration versions.

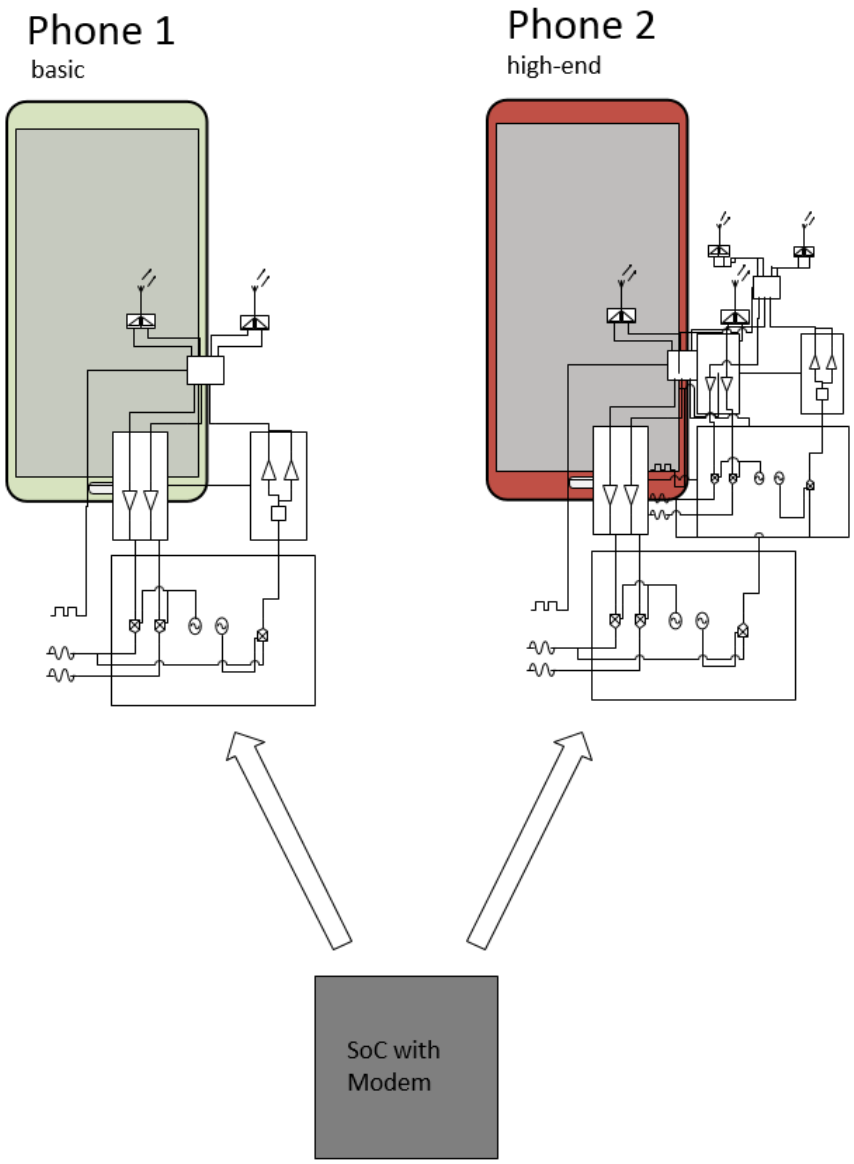


FIGURE 6. An illustration of the challenge to adapt the modem software to control a different RF front-end

For 4G and previous generation RF front-end layouts, a tool like this had been created by MediaTek engineers. The same idea and existing software was used to start creating a new tool for the 5G FR2 purpose.

3 5G AND FREQUENCY RANGE 2

The 5G rollout can be divided into 2 parts, FR1 and FR2. They mean the frequency range that the mobile device uses for its carrier radio wave. FR1 is for frequencies below 6 GHz which overlaps and extends 4G LTE frequencies, operating from 450 MHz to 6000 MHz. Bands are numbered from 1 to 255. This is commonly referred to as New Radio (NR) or sub-6GHz (11.). FR2 is the millimeter-wave, for which is specified a range of 24250-52600 MHz (12.) (FIGURE 7).

FR1 specification could be seen as a logical step further from 4G/LTE networks with a number of advancements. The bandwidths have increased. Now, the maximum bandwidth for FR1 can be 100 MHz, which is a huge improvement over 20 MHz in LTE. There are new enhancements, such as High-Power UE (HPUE) (13.) and Bandwidth Part Adaptation (BWP) (14.). Some new frequency bands have been introduced, going above 3 GHz. There are also specified different operation modes for 5G which are Stand-Alone (SA) and Non Stand-Alone (NSA). The NSA is an operating mode for using a 5G carrier for the data transfer but it relies on 4G infrastructure. SA would be the operating mode for a 5G-only network where the UEs and base stations are operating according to 5G specifications.
FR2, or the millimeter-wave (mmW in short), is a new topic in many aspects. The major difference is the carrier frequency, which is risen to levels not seen before in public mobile network technologies. The mmW frequencies have been previously used in satellites for atmosphere measurement and radars.

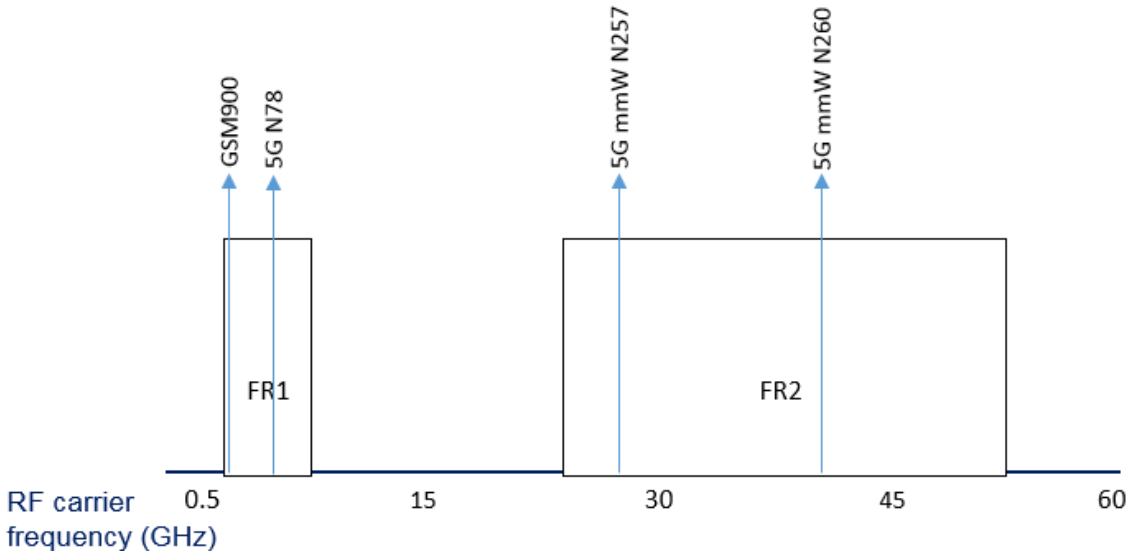


FIGURE 7. A Comparison between mmW and legacy frequencies. Arrows show the frequency of different bands.

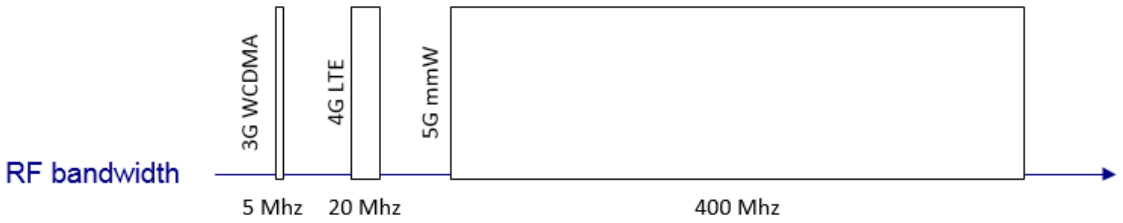


FIGURE 8. A Comparison between mmW and legacy bandwidths (to scale).

The maximum mmW component carrier bandwidth can be as high as 400 Mhz. This is 20 times larger than the LTE specified maximum bandwidth for a single component carrier. The mmW connection would provide an extremely fast multi-gigabit connection speed and support a large amount of users. It would work mainly indoors and with short distances because of the nature of the waves in the mmW range. These waves cannot sufficiently penetrate objects or walls, so the connection is limited to a line-of-sight connection only.

3.1 UE Analog beamforming

A significant new feature of the FR2 is the use of analog beamforming in the UE (user equipment) side. Analog beamforming has been used previously in the base station antennas but not in the mobile devices. This means that the over-the-air radio connection of the mobile equipment and base station will be a “beam”, like the light beam of a spotlight (FIGURE 9). This beam can be steered and formed to be narrow. This can enhance the multi-user capabilities of the network by dedicating one beam for one user, thus reducing interference between users.

The beamforming introduces new demands for the RF front-end hardware. Instead of a single antenna, an array of multiple antennas is going to be used. The phase and gain factors of each signal path, through each antenna, are adjusted to create the beam. The RF signal to the antenna array is provided by a beamforming driver IC (FIGURE 10.) which will include a phase shifting component (PS) for each antenna in the array. By controlling the phase shifters, the beam forming and steering is realized. The phase shifters are something not seen before in a typical mobile device. The beamforming IC also includes the typical PA and LNA amplifiers for transmit and receive direction signal paths.

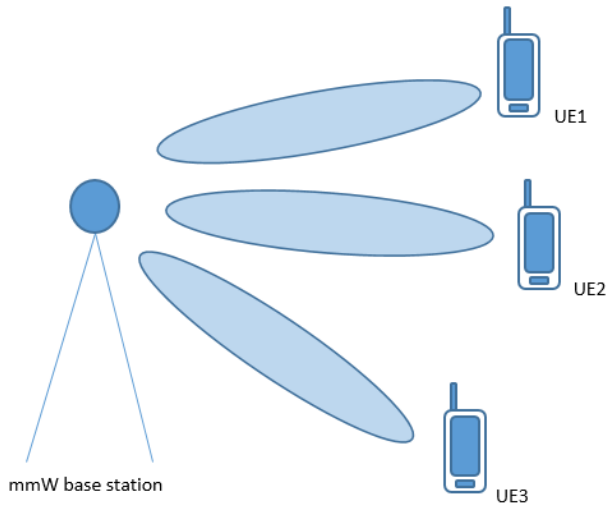


FIGURE 9. The beam connection between an mmW base station and UEs

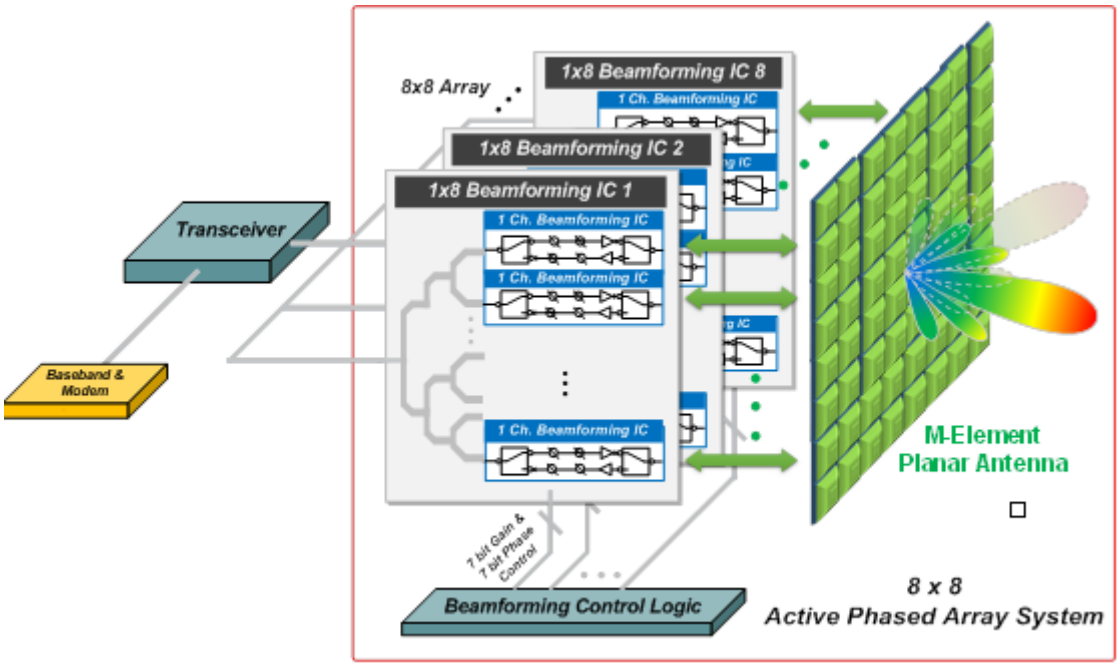


FIGURE 10. An illustration of mmW beamforming hardware (15.)

3.2 Antennas with different polarization

More than one antenna polarization type is used in mmW device antennas, which is something not seen before in typical mobile devices. Usually, mobile radio devices use only antennas with a single polarization type.

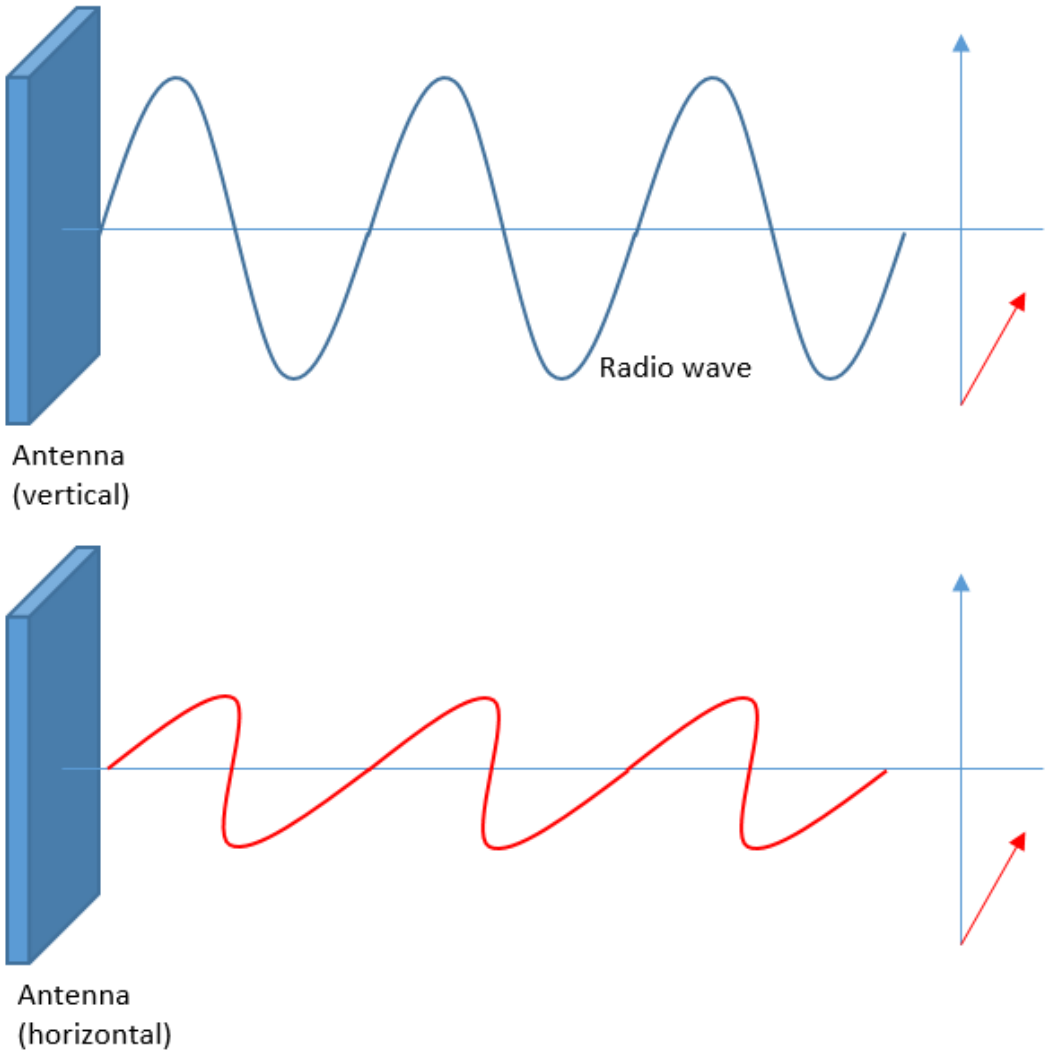


FIGURE 11. An illustration of radio waves with linear horizontal and vertical polarizations

A linear polarized antenna radiates in one geometric plane containing the direction of propagation (FIGURE 11.). Antennas can also be circular polarized. In a circular polarized antenna, the plane of polarization rotates, making one complete revolution during one period of the wave. The circular polarization can be left-hand (LHC) or right-hand (RHC) depending on the direction of the plane rotation. An antenna is said to be vertically polarized (a linear type) when its electric field is perpendicular to the Earth's surface. (16.)

Horizontal and vertical polarization types are used in mmW antenna panels. The circular polarization type is not used. Polarization is used to achieve the MIMO capability in the mmW radio technology.

In mmW, the usual principle of multipath propagation cannot be used for creating MIMO signal streams because of the inability of millimeter-wavelength waves to penetrate objects and their short range. Instead, the polarization is used. One signal stream is sent with a vertical polarization and one with horizontal polarization. At the receiver side, the respective polarized antennas can receive the streams.

3.3 Summary

As the FR2 bands operate at a very high 25-55 GHz range, the devices using this technology will have a very limited range and only a line-of-sight connection is possible. Because of this, the mmW technology introduces antenna arrays, beamforming, and MIMO implementation through antenna polarization. These technologies will have a direct impact to the RF front-end of an mmW mobile device. For the beamforming capability there is a new type of RF component, the beamforming IC. The antenna section of RF front-end is completely different and it will consist of antenna panels with numerous antennas of different polarization. This adds a large amount of complexity and control requirements to the RF front-end.

4 REQUIREMENTS FOR THE NEW CUSTOMIZATION TOOL

In this chapter the requirements for the mmW RF front-end customization tool are reviewed.

4.1 Customizing beamforming IC configuration

The FIGURE 12. below shows a functional diagram of a simple beamforming IC device. It consists of the modulated signal input, an oscillator to mix the signal to the mmW frequency (28GHz for example) and an array of phase shifters and amplifiers which comprise a single RF port which ends to an antenna. An mmW mobile device can contain multiple of these beamforming ICs. The reason is that as the device is moved and rotated in the user's hand, the beam might become blocked from one corner of the device, so the beam needs to be switched to a better one. A beamforming IC positioned on the other side of the device might provide a better connection in this situation (FIGURE 13.). This became the key aspect in the mmW device RF front-end customization.

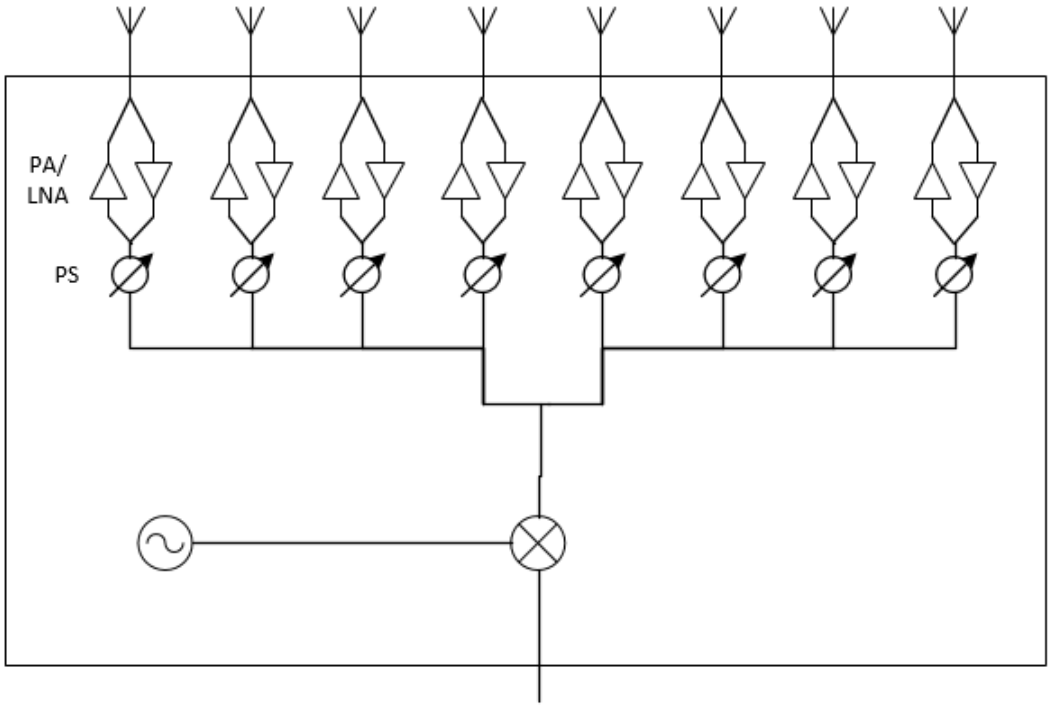


FIGURE 12. A simplified beamforming IC with 8 antennas

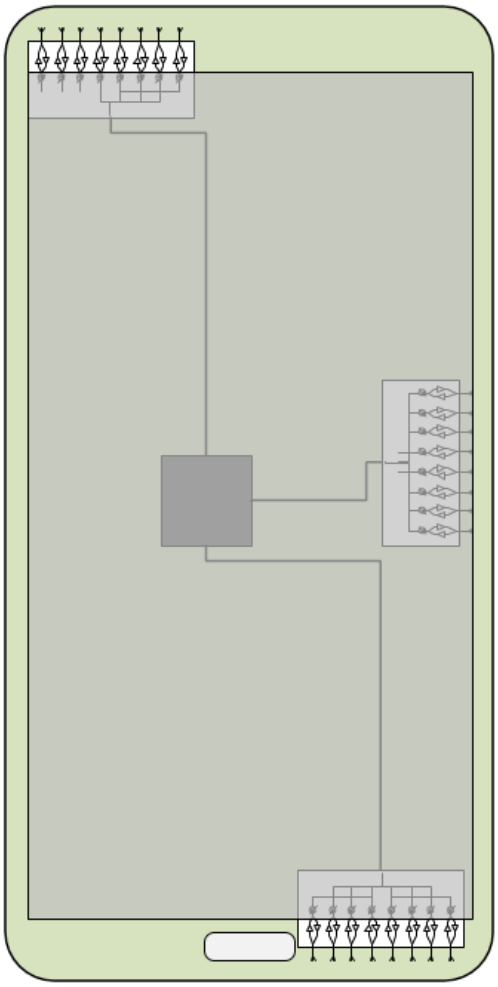


FIGURE 13. Beamforming ICs in different positions in a mobile phone

Depending on the cost and size class of the device, the phone manufacturer may want to use a different number of mmW chips. An mmW modem chipset is not only used on a mobile device but it could also be used in other products, such as data dongle, tablet and laptops.

There could be many different types of these beamforming ICs. They could be arranged in groups to provide even more controllable antennas for beamforming. The customization tool needs to provide support for abstracting this beamforming IC with different connection ports (FIGURE 14). The user will select these beamforming devices and draw their connections to the modem chip in the GUI.

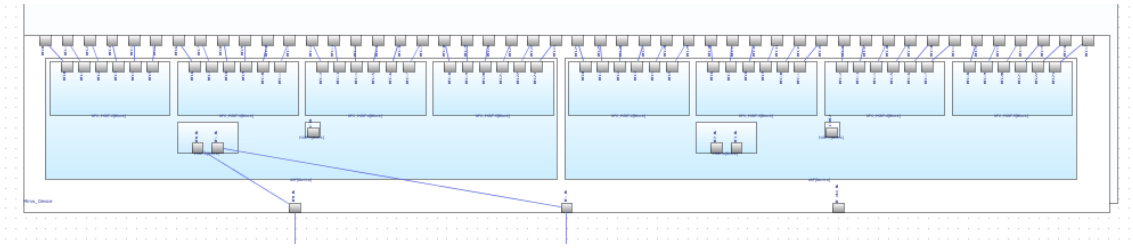


FIGURE 14. A beamforming device shown in the tool GUI

The beamforming device has generally 2 main interfaces: antenna ports and control ports. Some beamforming devices might only contain the amplifier and phase shift parts, so they must be connected to an RF mixer part which is a separate device. Some beamforming devices might contain the RF mixer part internally. This also posed a requirement for the tool that is to be able to integrate different types of devices into a same package.

4.2 Customizing mmW antenna panel

The mmW antenna panel is the other key item that is needed to be customized in the tool. An mmW antenna panel consists of multiple antennas which can have different polarization or construction type (a dipole antenna or a patch antenna). Each of the beamforming IC RF ports get connected to an antenna and the polarization and type information needs to be conveyed for the modem control software.

For the antenna panel, the customization tool needs to provide options for:

- Number of antennas in an antenna panel
- Arrangement of antennas (coordinates)
- Individual antenna polarization
- Individual antenna type (patch or dipole)

In the GUI, the antenna panel is shown as a rectangular component with the individual antennas as smaller rectangles inside it (FIGURE 15). The panel has a row of interface pins which are connected to the antennas. The pin names describe the polarization and type information.

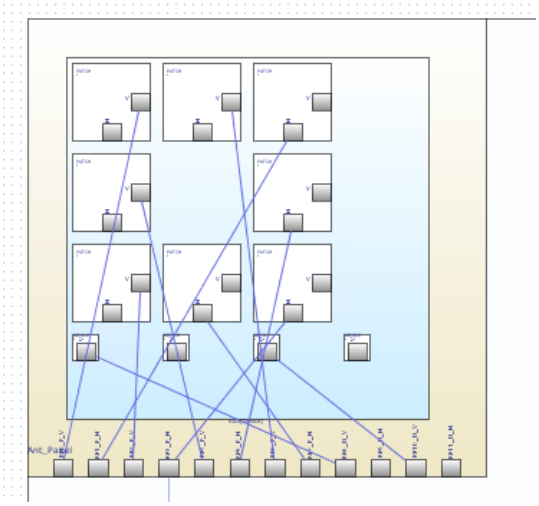


FIGURE 15. An antenna panel abstracted in the tool GUI

4.3 Customization tool requirement table

A customization requirement table (Table 1.) was formulated after much research, lengthy discussions with related engineering teams and many design iterations. Some of the requirement sources are from tool user's perspective and some are from the perspective of the mmW RF system design.

TABLE 1. mmW customization tool requirements

Item	Details	Source
Create a component abstraction layer for loading components	How the components are expressed in the configuration input file. Reused existing software with modifications.	
Model antenna panel		RF specification
Model patch type antenna		RF specification
Model dipole type antenna		RF specification
Model vertical and horizontal polarization for each antenna type		RF specification

Model connector pins for components		Component abstraction
Model component connection		Component abstraction
Connection rule check for some components		RF specification
Model integrated components	Combine components inside a larger module	RF specification
Model beamforming IC		RF specification
Save the project file	JGraph library function	Tool user
Load the project file	JGraph library function	Tool user
Undo action	JGraph library function	Tool user
Save part of drawing as a reusable block		Tool user
Create methods to parse the component graph and connections		Code generator implementation
Generate a C data structure of RF devices		RF SW specification
Generate a C data structure of antenna panels and connection information		RF SW specification

5 CREATING THE TOOL

In this chapter the structure of the created tool software and the tools used to create it are explained. Some key software functions, which were developed, are described in detail in chapters 6 and 7.

5.1 Platform

The GUI was developed with the Java programming language. A Similar GUI tool had been developed earlier at Mediatek for 4G and legacy RF front-end customization, using Java, so it was logical to start the mmW tool development with Java, too. The original reason for using Java in customization tools was to support multiple PC platforms (Windows, Linux, Mac) as the tool is released to customers for use.

5.2 IDE

Netbeans Java IDE was used for writing the code, running and debugging (17.). Netbeans is free software and provides rich features for an easy Java development.

5.3 Tool software overview

The GUI interface was created with Java's native Swing library. It provides the visible GUI components, such as the main frame, panels, buttons, menus and text labels. The Netbeans IDE contains an interactive tool which can be used to place the GUI components and create the layout as wanted. Then, it generates event handlers to the code to which the interaction logic can be written.

The code can roughly be divided into 4 parts:

- Graph library and utility function
- Configuration input file parser
- RF component abstraction layer
- Code Generator

The figure 16 below illustrates the basic Java classes that form the tool code.

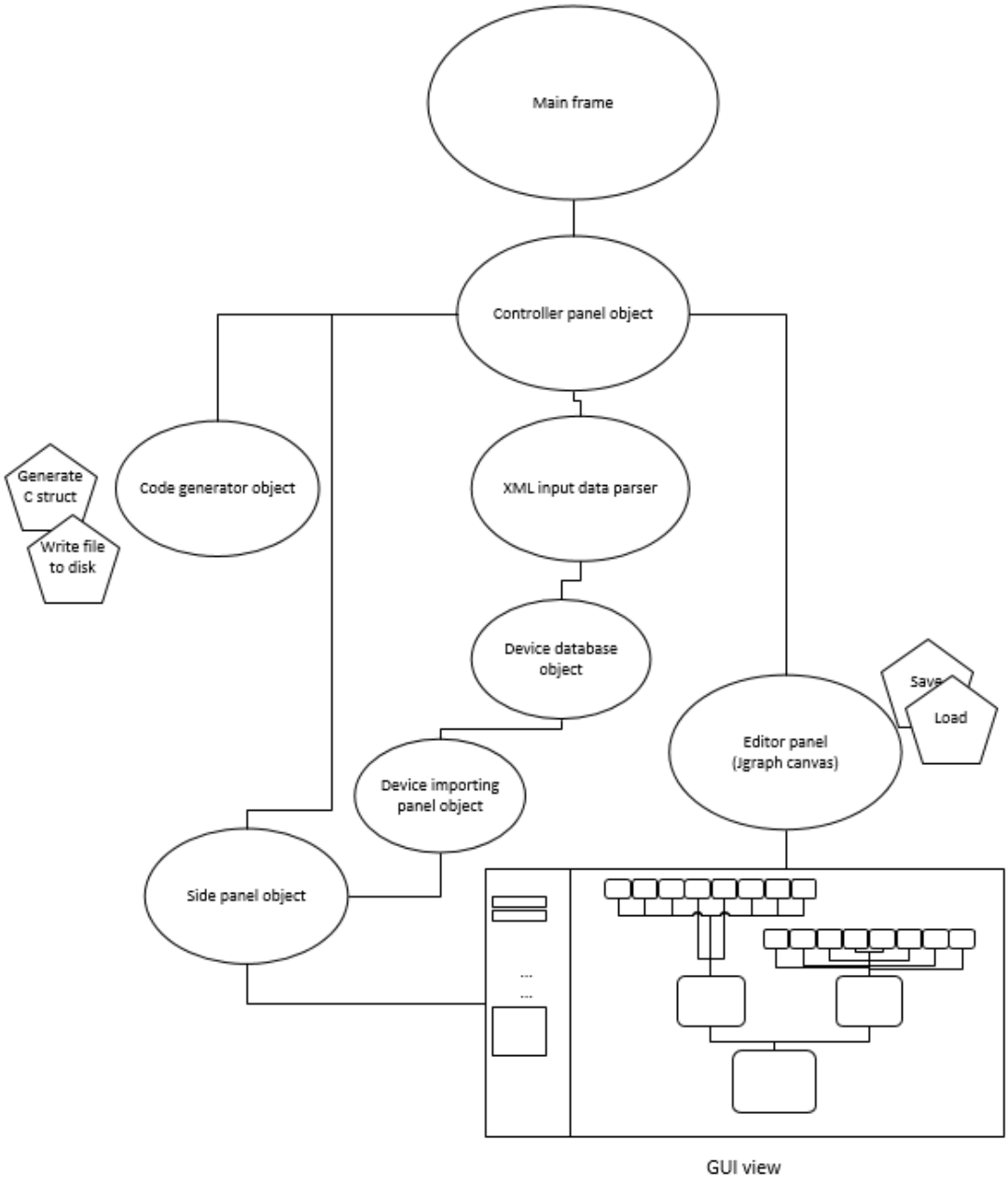


FIGURE 16. A diagram of tool main classes

The graph library provides a GUI "canvas" where the graph can be drawn. The graph consists of cells and wires that connect the cells. The cells represent the RF components (FIGURE 17). At first it was attempted to create this graph engine from scratch using the Java Swing components as a base. However, a better solution was found, which is the JGraph library (18.). The JGraph library provides a complete support for this kind of graph creation. It includes support for copy, paste and undo functions. It also includes support for saving and loading the graph file, which was a huge

time-saver, although many difficult bugs had to be solved before it started working. The JGraph seemed very old software. As detailed documentation was lacking, much of the problem solving had to be made by trial and error.

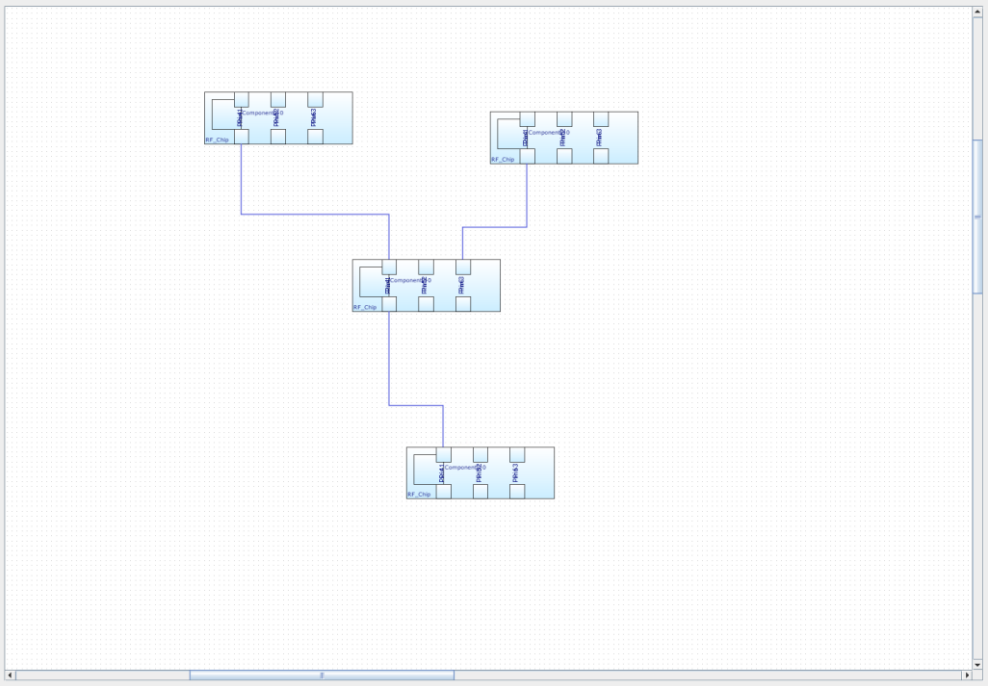


FIGURE 17. The JGraph drawing canvas with example components

The configuration input file parser is used to read the RF component data that is presented for the tool. In the data files there are presented the component type, shape in the GUI, number of pins, pin connection restrictions, internal details. The parser provides an interface from the configuration file to the GUI object. The parser was originally written at MediaTek for the existing customization tool, and a copy of it was utilized for this mmW tool with some modifications.

The RF component abstraction structure is a class ecosystem in the program code. It is used to represent each RF front-end component as a Java object. Each object stores data of the component, its connection pins and possible inner subcomponents. Working together with the JGraph library, it provides information about the connections that the component has to other components.

The code generator is the final part that is used in the customization flow. When the user has finished the drawing, they can press a button which starts the generation of custom data files. The

code generator parses the RF front-end component object structure, arranges different types of components into different lists, performs sorting and checking and finally generates the C-programming language compatible output that describes the RF front-end layout and control data of the components.

5.4 General program operation principle

A tool support team will design and provide the component descriptor files that can be used in the tool. Then, the customer will see a collection of these components and they can 'drag and drop' a component to the drawing area of the tool. Then, they can make connections in and out of the ports of the components. Different versions of these RF front-end configurations can be created and saved as a project.

5.5 Backbone in the GUI tool code

The tool has a couple of main classes that act as a backbone of the tool. They will store the drawing graph object, component palette object, reference to the input file parser and methods that are executed when a component or button is clicked. In a large software project like this, it becomes important to organize the classes well.

The drawing graph class is provided by the JGraph library and the "Editor panel" class uses it. All editing actions to the drawing are processed inside the graph class. It provides event handlers, which make it easy to implement the editing functions. Other actions that are not handled by the JGraph library, such as showing device information and starting the code generator, are handled by a main controller class which is the "Controller panel".

5.6 Class description

- Main frame
    - The visible frame window of the tool. It is a Java Swing-based object.
- Controller panel

    - A Java Swing-based panel object which acts as a master root class and contains the other objects (except the main frame)
- Configuration input file parser
    - Parses device description files and creates interface objects out of them
- Device database & Device import panel
    - Contains the mmW device objects and the devices can be dragged to the graph from the import panel
- Side panel
    - Acts as a container for displaying device information and control buttons
- Editor panel
    - Contains the JGraph main object and the drawing canvas which is displayed
- Code generator
    - Contains a main function to generate customization data output files. It works with other code generator related classes.
- Graph parser utility
    - Contains functions to parse the JGraph graph and to find devices from it

6 DEVICE ABSTRACTION

The customization tool required an interface to implement the RF front-end devices. This makes it possible to model the components as graphical objects in the GUI and model their properties that are essential in the generated output files. The device interface consist of a few layers. These are:

- Device combination layer
    - Combines multiple devices
- Device itself
- Device interface connector pin layer
    - Defines device input/output pins
- Device internal part layer
    - Defines functional parts inside device
- Part internal connector pin layer
    - Defines connector pins inside parts

6.1 Defining device data

Each of these layers can be initialized in a configuration file which acts as a device descriptor (FIGURE 18). Each layer has a different set of possible parameters. Devices and parts have names, types and internal data, e.g. the antenna type. Connector pins have an index number, name and some data parameters, such as input/output type and a data protocol type. The input file parser reads the descriptor file and creates corresponding Java objects that represent the device. This device abstraction system is used to create the required mmW RF devices in the tool, such as the beamforming IC and antenna panel.

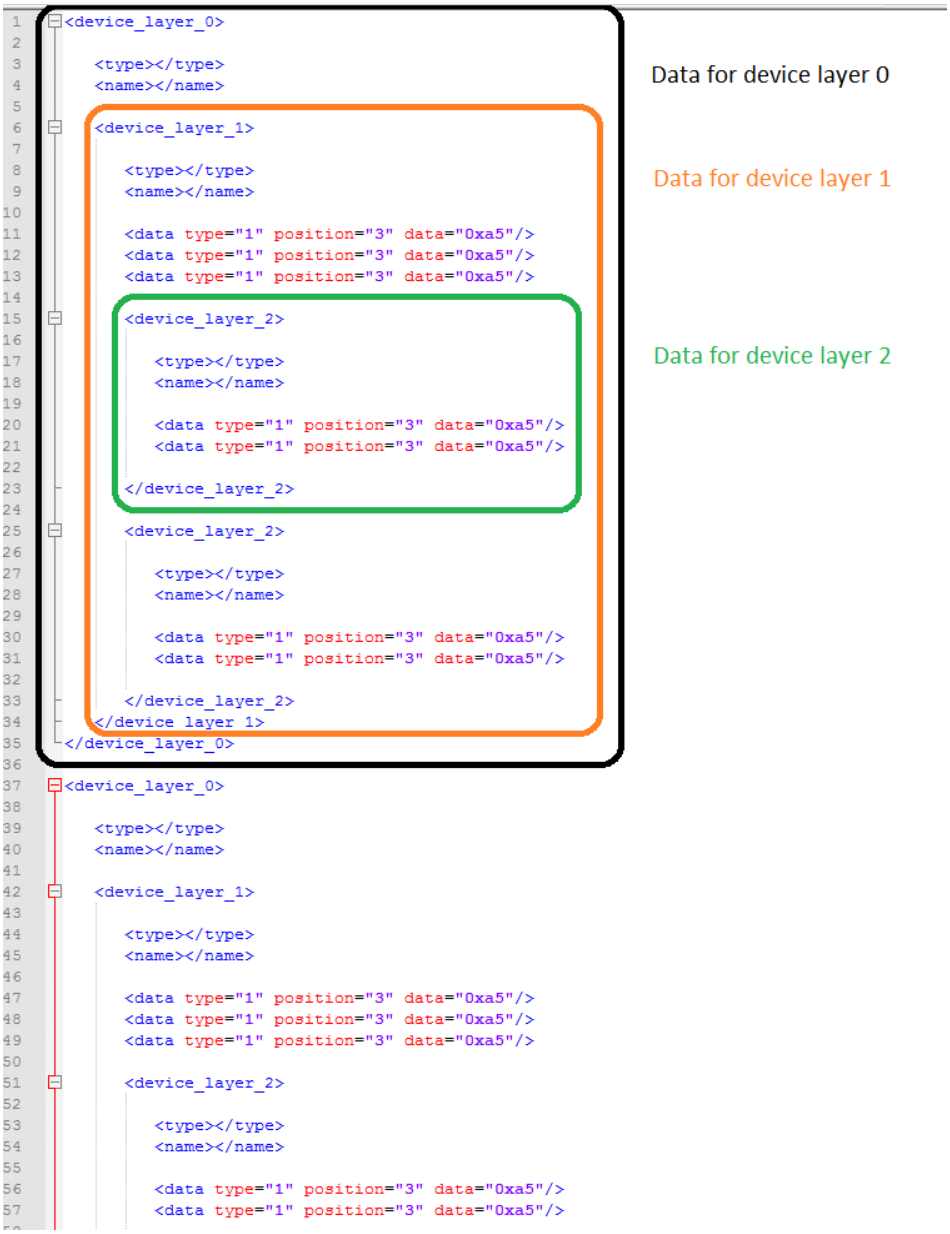


FIGURE 18. An example of a device descriptor file

6.2 Creating a visible device object

The device combination object is used when in creating the visible JGraph object (FIGURE 19) and also in code generator processing. These devices are then put into a database and they are visible

in a “device palette” panel in the GUI, where the user can pick a device and drag it to the drawing canvas. The device objects are attached to the JGraph cells. The JGraph cells form the RF front-end graph that the user builds, and the device object attached to a cell contains the device data. As the user modifies the devices or makes connections, the data inside a device object is updated and later used in the code generator phase.

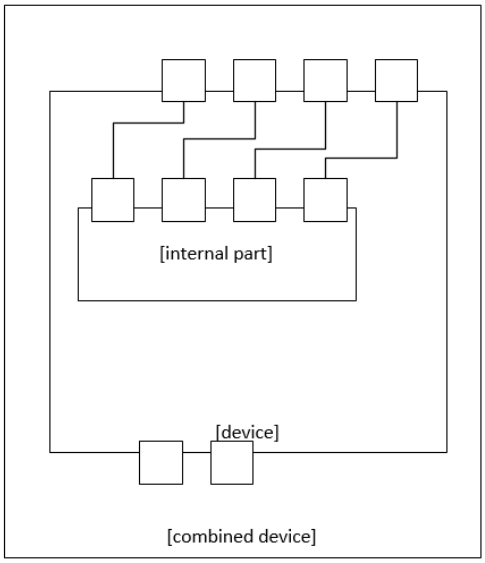


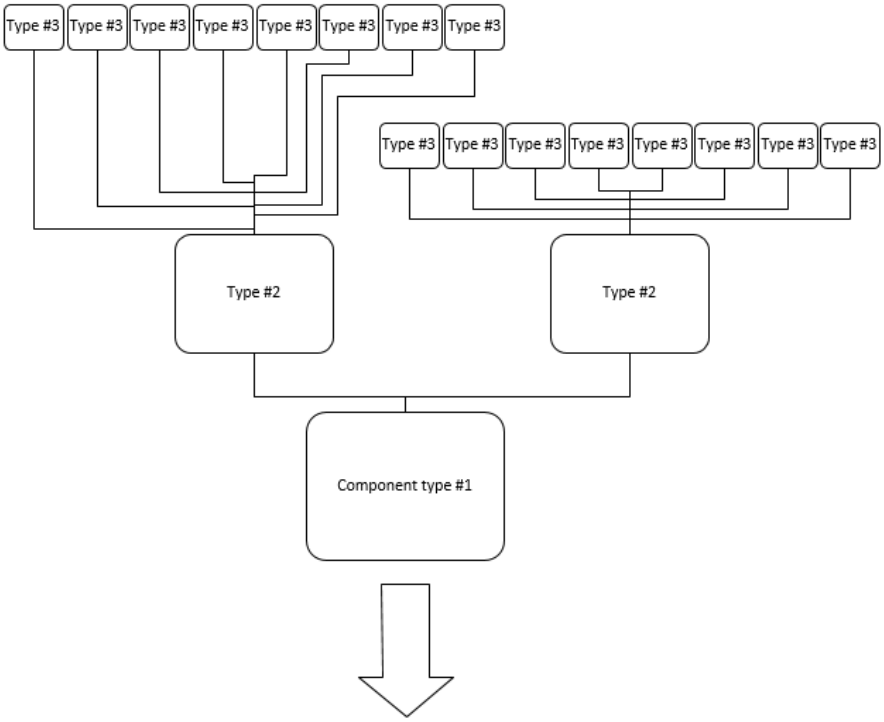
FIGURE 19. A resulting visible device object from the descriptor

7 CODE GENERATOR

The code generator is a part of the customization tool that is used in the final stage when the drawing has been completed. The code generator generates the output using the RF front-end drawing as an input. The output is a C-language file that needs to comply with the C syntax but also with a higher level *format* that is specified by the modem chip control software designers. The generator output file is called a custom file as it contains the customization data.

7.1 Overview

Usually in the custom file the data is expressed as a series of 'struct' data types that are nested and put into different arrays. For example, one PA component could be expressed as one struct. The struct would contain control data for the PA, such as register addresses and data that is required to change the PA mode. Then, an array could contain these PA component structs, depending whether there are many of them. FIGURE 20 illustrates what the code generator does.




```
TYPE_1_STRUCT
{
        data
        data
        &TYPE_2_LIST
        &TYPE_2_LIST
}

TYPE_2_LIST[]
{
        TYPE_2_STRUCT
        {
                data
                data
                &TYPE_3_LIST
        },
        TYPE_2_STRUCT
        {
                data
                data
                &TYPE_3_LIST
        },
}
```

```
TYPE_3_LIST[]
{
        TYPE_3_STRUCT
        {
                data
                data
        },
        TYPE_3_STRUCT
        {
                data
                data
        },
        TYPE_3_STRUCT
        {
                data
                data
        },
        TYPE_3_STRUCT
        {
                data
                data
        },
        TYPE_3_STRUCT
        {
                data
                data
        },
        TYPE_3_STRUCT
        {
                data
                data
        },...
```

FIGURE 20. The code generator objective

The struct formats can become quite complex. Designing the generator software flow was started by examining an example output file and trying to find patterns. Pattern finding is important. It helps

to determine what kinds of loops and procedures are needed in the program code to generate it. Another important thing is to characterize the components. For mmW design, there were only a couple of component types, the antenna panel, the beamforming IC and an intermediate frequency IC (IFIC). But the design was still hard because the whole mmW device ecosystem is so new. There were many uncertainties about which parts are needed to customize and the requirements changed during the work. A difficult part was to invent a modular and logical component type definition that would work for representing different combinations of the mmW components.

The code generation required developing a few key software methods which are introduced in this chapter. These are:

- Java class for abstracting the C programming language's structure data type
- Algorithm to find certain type of components from a tree graph
- Algorithm to search through a connector wire in a tree graph

These are key functions that are used in higher level loops and functions to implement the customization data code generator.

7.2 Abstracting C struct type with Java class

An idea that was implemented to make the code generation easier and the tool code less complicated was to create a Java class for representing a C struct type and a C array.

A struct in the C programming language is a user-defined data type which is a collection of numbers or characters and own name can be given to it. A struct type can also contain inner struct types which create nested structs which are very common in programming. This is also used for the component customization data.

FIGURE 21 shows an example of a user-defined struct data type which is used in C code to express beamforming device information. Some of the fields might be filled with device data that is already specified in the device input files and some data is filled according to the customization tool user input.

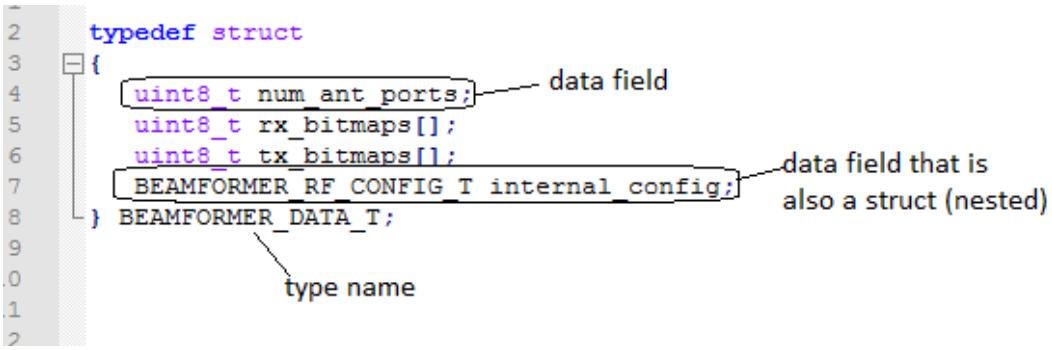


FIGURE 21. An example struct type for expressing beamformer device information

A Java class was created to implement a C struct type like this. It was given a name “CommonStruct”. The class has variables for the struct type name and instance name. It has a map for the fields where a data field can be stored. It has also methods to initialize new fields and finally print out the struct when it is complete.

The figure 22 shows how the C struct class is used in Java code to create a new struct. The struct type name and instance name are given as arguments to the object creation function. Then, the data fields of the struct are added with the “addField” method. “addField” takes the field type, field name and initialization value as a parameter. The initialization value can also be added later, which is done for the array field types (FIGURE 23). The nested struct “internal_config” is initialized with another “CommonStruct” object that has been created earlier.

```
// create new struct and init fields
CommonStruct beamformerInfoT = new CommonStruct("BEAMFORMER_DATA_T",
                                                "beamformer_data_0", true);
beamformerInfoT.addField("uint8_t", "num_ant_ports", "12");
beamformerInfoT.addField("uint8_t", "rx_bitmaps", null,
                                    CommonStruct.FIELD_TYPE_E.ARRAY_FIELD);
beamformerInfoT.addField("uint8_t", "tx_bitmaps", null,
                                    CommonStruct.FIELD_TYPE_E.ARRAY_FIELD);

// add inner struct field which was created earlier
beamformerInfoT.addInnerStructField("internal_config", internalConfigT);
```

FIGURE 22. Creating the example beamformer info struct in Java code using the "CommonStruct" class

```
// init the array fields by appending to them
beamformerInfoT.appendToArrayField("rx_bitmaps", "0b01010100");
beamformerInfoT.appendToArrayField("rx_bitmaps", "0b01110100");
beamformerInfoT.appendToArrayField("rx_bitmaps", "0b01010111");
beamformerInfoT.appendToArrayField("tx_bitmaps", "0b01010100");
beamformerInfoT.appendToArrayField("tx_bitmaps", "0b01110100");
beamformerInfoT.appendToArrayField("tx_bitmaps", "0b01010111");
```

FIGURE 23. Initializing array type fields with the method "appendToArrayField"

This makes it very easy to generate new structs in the code generator phase. These struct objects can be put to an array and a C struct array can then be easily printed out to the custom data file. In the FIGURE 24 the example beamformer struct has been printed to a file. It complies with the C syntax to initialize a “BEAMFORMER_DATA_T”-type struct variable whose name is “beamformer_data_0”.

```
368   BEAMFORMER_DATA_T beamformer_data_0 = {
369       .num_ant_ports = 12,
370       .rx_bitmaps = {
371           0b01010100,
372           0b01110100,
373           0b01010111,
374       }
375       .tx_bitmaps = {
376           0b01010100,
377           0b01110100,
378           0b01010111,
379       }
380       .internal_config = {
381           .rf_conf_data = 0x8afe,
382       },
383   }
```

FIGURE 24. After using the print method of CommonStruct, the struct is created to C code ready to use

The other method to generate these structs would be a “string format” method as shown in FIGURE 25. Advantage of this method is its initial simplicity, only a “template” of the generated struct is needed and then it is formatted with the data. But the template is hard and time-consuming to write because it requires adding the newline and indentation characters manually. Especially, for a larger nested struct the object-based “CommonStruct” provides much more advantages. The fields can be added flexibly, not having to add all at once. The object-based structs can be managed more easily and the code is easier to read.

```
String beamformerInfoT =
        "BEAMFORMER_DATA_T beamformer_data_0 = "
        + "{\n"
        + "    .num_ant_ports = %d;\n"
        + "    .rx_bitmaps = {\n"
        + "       %s,\n       %s,\n       %s\n    }\n"
        + "    .tx_bitmaps = {\n"
        + "       %s,\n       %s,\n       %s\n    }\n"
        + "    .internal_config = {\n"
        + "       .rf_conf_data = %s\n    }"
        + "\n}\n";

beamformerInfoT = String.format(beamformerInfoT, 12, "0b01010100",
                                "0b01110100", "0b01010111",
                                "0b01110100", "0b01110100",
                                "0b01010111", "0x8afe");
```

FIGURE 25. The string format method to generate a C struct with data

7.3 Algorithm to find components

The customization data format requires certain types of components to be put into a same list and gathering connections for only certain types of components at a time. This required devising an algorithm to find only certain types of components that are connected to a parent component (FIGURE 26).

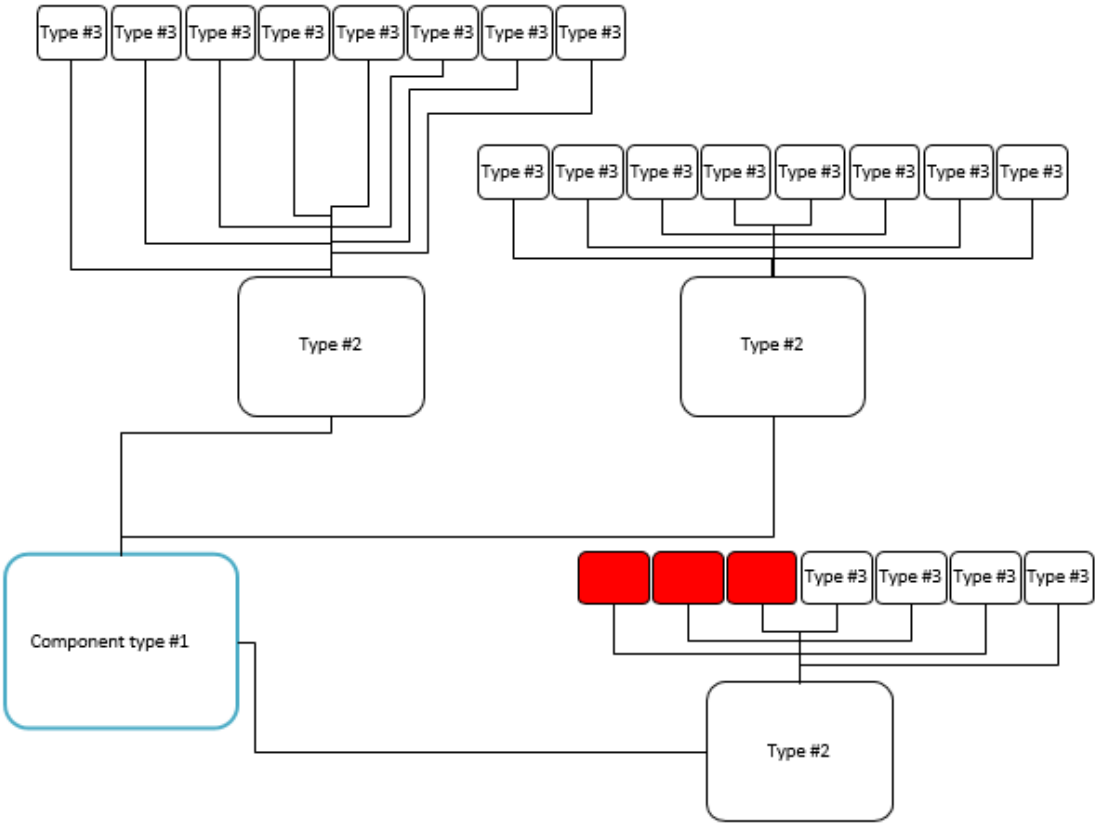


FIGURE 26. A function was needed to find only red components starting from the blue circled parent component

The search function takes as parameters the starting cell, a result list and a match criterion cell type (FIGURE 27). The function works by looking at the child cell list of the start cell. Then, the function calls itself and passes a child cell as the starting cell and the action is repeated. If the function finds a cell that is of a wanted type, it adds it to a list. The function looks at all cells and finally returns when all child cell lists have been looped through.

```java
/**
 * Get a cell by matching type.
 *
 * @param cellList
 * @param startCell the root cell to start search from
 * @param matchType the type of device to search
 */
public static void getCellsByType(ArrayList<mxCell> cellList,
                                  mxICell startCell,
                                  String matchType) {

    if (startCell.getChildCount() > 0) {

        // loop the child cells of start cell
        for (int i = 0; i < startCell.getChildCount(); i++) {
            // recursive call to process child cell the same way
            getCellsByType(cellList, startCell.getChildAt(i), matchType);
        }
    }

    if (startCell.getValue() != null){

        // get the "mmw device" object from the jGraph cell
        Device device = (Device) startCell.getValue();
        if (device.getType()!= null) {

            if (device.getType().equals(matchType)) {

                // add cell to result list
                cellList.add((mxCell) startCell);
            }
        }
    }
}
```

FIGURE 27. The cell search function

7.4 Algorithm to search a connector wire

Another type of algorithm was needed to find out which cell is connected to the other side of a connection wire (FIGURE 28).

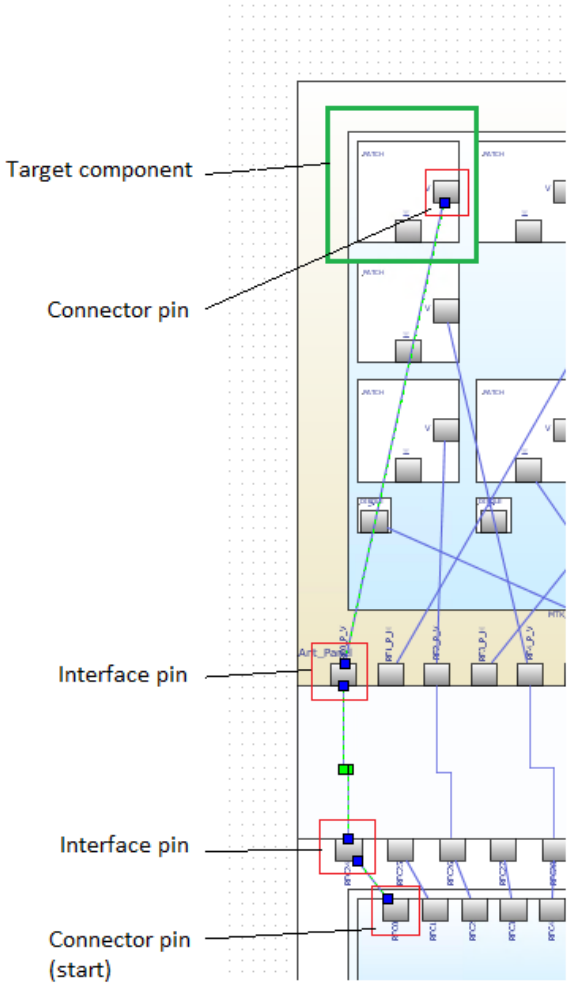


FIGURE 28. A function was needed to make the target component searching from the start pin

This requirement provided a challenge at first because of the “interface pins” on the connection route. It was not enough to just get the next connection pin from the start pin.

A recursive function was again devised to solve this problem (FIGURE 29). The function starts to look at the next connected pin from the start pin and calls itself passing the next pin as a parameter. For an interface pin, the function ignores the previous connector wire and looks at the next interface pin. This action is repeated until the function finds an “end” pin which has no further connection. Then, it returns the parent component of the connector pin.

```
public static mxCell getConnectedCell(mxCell pincell,
                                      mxCell ignoreEdge, Class<?> clazz) {

    if (pincell.getEdgeCount() == 0) return null;// check if nothing is connected to pincell

    // if ignore edge is not specified, do not accept multiple edges on block pin
    if (pincell.getEdgeCount() > 1 && ignoreEdge == null) {
        ErrorReporter.throwCodegenError();
        log.error("There cannot be more than 1 connection to block pin!");
        return null;
    }

    // if ignore edge is specified, there can't be more than 2 edges
    if (ignoreEdge != null && pincell.getEdgeCount() > 2) {
        ErrorReporter.throwCodegenError();
        log.error("There cannot be more than 2 connection to block pin in middle point pin!");
        return null;
    }

    mxCell connectedCell = null;
    mxCell edge = null;
    if (ignoreEdge == null) edge = (mxCell) pincell.getEdgeAt(0);
    else {
        //ignore edge is not null so the pin cell is a middle point pin with in and out connection
        // the in connection must be ignored
        for (int i = 0; i < pincell.getEdgeCount(); i++) {
            edge = (mxCell) pincell.getEdgeAt(i);
            if (!edge.equals(ignoreEdge)) {
                break; //there should be only 2 edges, so break when find one which is NOT the ignore edge
            }
        }
    }

    connectedCell = (mxCell) edge.getSource();
    if (connectedCell != null) {

        // if cell is external pin, call this function again
        // to find the device that is connected to external pin
        if (((Pin) connectedCell.getValue()).isExternalPin.equals("true")) {

            // if connected cell has more than 1 edge, then it is a middle point cell with further connect
            // call this function again and find further pin from external pin
            if (connectedCell.getEdgeCount() > 1) return getConnectedCell(connectedCell, edge, clazz);
            else return null; // the external pin was not connected to anything
        }
        else { //connected cell was not external pin so it must be something interesting

            // if specific class is given as function parameter, then check if the cell is of the class ty
            if(clazz != null){
                if (clazz.isInstance(connectedCell.getParent().getValue())) {
                    return connectedCell;
                }else return null;
            }
            else return connectedCell; //return the connected cell that was found
        }
    }
    return null;
}
```

FIGURE 29. The connected component search function

8 CONCLUSION

In this thesis work, the idea of an RF front-end configuration customization tool was introduced. The history of mobile phone RF front-end circuitry was analyzed and it was explained how it has gotten more complex during the years and generations. It was explained that the number of frequency bands and radio access methods grew, and different bands were used in different countries. The mobile phone RF front-end configuration developed a need for different versions and eventually a tool was needed for managing these versions. The history of the RF front-end was presented quite extensively.

New requirements for an RF front end required by the new FR2, or mmW, technology were analyzed. It was discussed how these new requirements would impact a new mmW RF front-end customization tool. The main requirements were well defined and presented.

The creation of this new tool and the software and methods behind it were introduced. Some of the algorithms used by the tool were explained. The code generator part could have been explained also at a higher level but it was left out because it was still a work-in-progress by the time of writing of the thesis. A comprehensive picture about the tool's user interface would have been a good addition but it was left out for confidentiality reasons. The project was started with little information about the coming requirements. For the first couple of months the plan was just to try to gather information about the mmW components and what is needed to customize them. The starting date was too early regarding the status of the mmW system design but on the other hand it provided more time to solve the purely Java and GUI-related problems. A lot of code was made which was later deleted and reworked. The tool was created as a stand-alone tool but the plan is to later integrate it into the existing customization tool which is already used for sub-6 GHz RF front-end customization.

One major concern with the tool development was the compatibility with the said sub-6 GHz customization tool. It meant that the code needed to follow similar design principles and provide a same look and feel to the user. Part of this was easy to achieve thanks to the JGraph library, which was also used in the existing tool. Also the input configuration file parser is based on the same code. The device abstraction system is also mostly the same but the device objects are handled differently. These differences will probably result in later efforts when the mmW tool needs to be

updated by someone else than the author. But generally problems like this are unavoidable in software development.

An unresolved problem was the performance of the JGraph drawing system and some details related to it. When creating large devices with many internal pins, the GUI responds incredibly slowly. Partly this problem is due to the inherent nature of Java. It is naturally slower than traditional compiled codes like C++. The author would have liked to solve this problem, possibly by trying a different drawing library, or continuing with the self-made library that was made in the beginning but due to the tool compatibility requirement, the JGraph system stayed. Also the JGraph had some annoying details, mainly the size and behavior of the graphical components that could not be altered or the information needed to alter them was not existing.

In the end it seemed that the number of customization requirements posed by new mmW RF front-end components is quite small but there would probably be some new items in the future, when the technology becomes more common and more manufacturers start to offer mmW RF components. Main functional requirements of the tool were completed but many details about the code generator output were still unspecified by the time this thesis was written. The tool is successful and it can support the new FR2 RF front-end customization requirements.

The project greatly increased the knowledge and skills of the author in many areas. These areas are RF hardware, object-oriented programming, Java programming, GUI program design and programming in general.

REFERENCES

1. Lam, W. 2017. The RF Front-End: Unsung Hero of the Premium Smartphone. Date of retrieval 28.5.2019. https://technology.ihs.com/593647/the-rf-front-end-unsung-hero-of-the-premium-smartphone
2. The Many Types of Radio Frequency Modulation. Date of retrieval 5.8.2019. https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-modulation/the-many-types-of-rf-modulation-radio-frequency/
3. Active Components in RF Circuits. Date of retrieval 5.8.2019. https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/rf-principles-components/active-components-in-rf-circuits/
4. Advanced Mobile Phone System. 2019. Wikipedia. Date of retrieval 13.8.2019. https://en.wikipedia.org/wiki/Advanced_Mobile_Phone_System
5. Wideband Code Division Multiple Access. 2019. 3GPP. Date of retrieval 13.8.2019. https://www.3gpp.org/technologies/keywords-acronyms/104-w-cdma
6. Johansson, K. – Bergman, J. – Gerstenberger, D. 2009. Multi-Carrier HSPA Evolution. Date of retrieval 7.8.2019. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.6719&rep=rep1&type=pdf
7. LTE frequency bands. http://niviuk.free.fr/lte_band.php. Date of retrieval 5.8.2019.
8. Wannstrom, J. 2013. Carrier Aggregation explained. 3GPP. Date of retrieval 7.8.2019. https://www.3gpp.org/technologies/keywords-acronyms/101-carrier-aggregation-explained
9. What is MIMO Wireless Technology. 2019. Electronics-notes.com. Date of retrieval 5.8.2019. https://www.electronics-notes.com/articles/antennas-propagation/mimo/what-is-mimo-multiple-input-multiple-output-wireless-technology.php
10. MIPI RFFE specification. MIPI alliance. Date of retrieval 14.6.2019. https://www.mipi.org/specifications/rf-front-end
11. 5G: What is Standalone (SA) vs Non-Standalone (NSA) Networks? 2018. MediaTek. Date of retrieval 7.8.2019. https://www.mediatek.com/blog/5g-what-is-standalone-sa-vs-non-standalone-nsa
12. 3GPP TS 38.104 V15.0.0 (2017-12). 3rd Generation Partnership Project. Technical Specification Group Radio Access Network; NR; Base Station (BS) radio transmission and reception (Release 15). Date of retrieval 13.8.2019. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3202

13. 5G NR Uplink Enhancements. White paper. 2018. MediaTek. Date of retrieval 7.8.2019. https://d86o2zu8ugzlg.cloudfront.net/mediatek-craft/documents/UL-Enhancements-White-Paper-PDFULEWPA4.pdf
14. Bandwidth Part Adaptation. White paper. 2018. MediaTek. Date of retrieval 7.8.2019. https://d86o2zu8ugzlg.cloudfront.net/mediatek-craft/documents/Bandwidth-Part-Adaptation-White-Paper-PDFBPAWPA4.pdf
15. Beamforming IC Figure. 2017. 2017 KAIST Tech Fair to Showcase Ten Cutting-Edge Technologies. KAIST. Date of retrieval 1.8.2019. http://www.kaist.edu/_prog/_board/?mode=V&no=69461&code=ed_news&site_dvs_cd=en&menu_dvs_cd=0601
16. Antenna polarization. 2019. Wikipedia. Date of retrieval 14.8.2019. https://en.wikipedia.org/wiki/Antenna_(radio)#Polarization
17. Netbeans IDE. 2019. Apache. Date of retrieval 2.8.2019. https://netbeans.org/
18. JGraph Java library. Date of retrieval 2.8.2019. https://github.com/jgraph/jgraphx