A LTE Receiver Framework Implementation in GNU Radio

Johannes Demel, Sebastian Koslowski, Friedrich K. Jondral
Karlsruhe Institute of Technology (KIT)
{johannes.demel, sebastian.koslowski, friedrich.jondral}@kit.edu

***Abstract*—We present an open source LTE receiver framework. Using GNU Radio's block-based signal processing capabilities, various LTE baseband specific functionality has been implemented in dedicated easily reconfigurable blocks. These can be used to decode and analyze arbitrary channels in the LTE downlink signal. As an example we decode the Master Information Block (MIB) transmitted on the Broadcast Channel (BCH). Our work is focused on performance measurements in order to identify critical processing operations on a General Purpose Processor (GPP). By optimizing critical components, e.g. channel estimation, synchronization, we can improve the overall system performance and therefore the system's real-time capabilities.**

***Index Terms*—LTE, GPP, GNU Radio, Performance**

I. Introduction

Over the last decades digital radio systems evolved from GSM with Time Division Multiple Access (TDMA) to UMTS with Code Division Multiple Access (CDMA) to Long Term Evolution (LTE) which uses Orthogonal Frequency Division Multiple Access (OFDMA). Goals for LTE development include higher data rates and lower latencies. Along with higher bandwidth modes some new techniques were introduced to reach these goals. For example Multiple Input Multiple Output (MIMO) is employed to achieve higher diversity or higher data rates.

The Long Term Evolution (LTE) standard [1] defines a multi-mode air interface based on Orthogonal Frequency Division Multiplex (OFDM) within its downlink structure. LTE requires high performance hardware to compute all its signal processing. This need is either met by Field Programmable Gate Arrays (FPGAs), by Application-specific integrated Curcuits (ASICs) or by optimized software for GPPs.

Modern radio communication systems rely on the ability to adopt to new environments. One approach to achieve more flexiblity is to utilize Software Defined Radio (SDR) techniques [2]. GNU Radio is a flexible open source SDR framework for GPPs, which provides a flexible and extendable signal processing infrastructure to easily adopt and add new capabilities. It also includes a rich signal processing library which is split into blocks that can be reused and regrouped as needed. Signal processing functions are implemented using C++ to hit high performance needs. A GNU Radio application consists of a flowgraph made up of several blocks and its connections between them. The setup and construction of the flowgraph is done using Python to facilitate easy usability. The GNU Radio Companion (GRC) further eases the development of a flowgraph by providing a graphical interface to GNU Radio.

In the following we describe our LTE receiver framework in GNU Radio. The focus of our work is on the implementation and performance of the physical layer receiver components. As an example we show the extraction of the MIB which is always broadcasted by a LTE base station (BS) to convey its basic configuration.

The remainder of this paper is structured as follows: In Section II the LTE downlink air interface and an outline of the receiver algorithms is described. Its implementation in GNU Radio is presented in Section III. The results of our performance measurements and its outcome for future optimization is evaluated in Section IV.

II. LTE Downlink Air Interface

LTE transmissions from a base station to a mobile terminal are frame-based multi-carrier signals. The transmitted symbols are organized in a scheme referred to as the OFDM time-frequency grid. A Resource Element (RE) is one modulation symbol on one subcarrier. REs are grouped into Resource Blocks (RBs) which are composed of twelve consecutive subcarriers and six or seven OFDM symbols depending on the Cyclic Prefix (CP) mode, normal or extended, respectively. The system bandwidth is defined as a multiple of RBs in frequency domain and may range from six to one hundred RBs as indicated in Tab. I.

RBs	bandwidth	subcarriers
6	1.25 MHz	72
15	3 MHz	180
25	5 MHz	300
50	10 MHz	600
75	15 MHz	900
100	20 MHz	1200

TABLE I
LTE bandwidth configuration options

In the time domain a transmission is organized in frames, see Fig. 1. Each frame consists of ten subframes, which include two slots with each six or seven OFDM symbols (OFDM symbols) grouped together in one slot. The number of symbols per slot varies according to the CP mode (and therefore its length). Assuming normal CP-length (N_{CP}) there are seven OFDM symbols in one slot and 14 OFDM symbols in

one subframe. A frame hence consists of 140 OFDM symbols. The transmitted frames are numbered by the System Frame Number (SFN) which is a cyclic counter from zero to 1023 and thus is represented as a ten bit binary value.

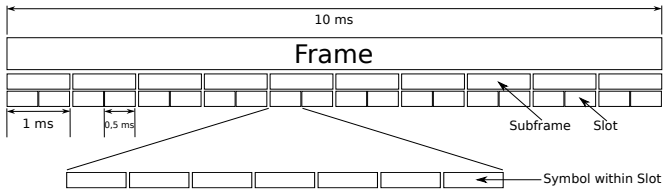

Fig. 1. LTE downlink frame structure

Together with the Reference symbols (RSs) for channel estimation and synchronization symbols, six channels are multiplexed on the time-frequency grid: the transport channels Physical Broadcast Channel (PBCH) which carries the essential system information, Physical Downlink Shared Channel (PDSCH) which carries the downlink data and Physical Multicast Channel (PMCH) for multicast purposes and the control information channels Physical Hybrid-ARQ Indicator Channel (PHICH) which provides an acknowledgement process and Physical Downlink Control Channel (PDCCH) and Physical Control Format Indicator Channel (PCFICH) which carry downlink control information.

A. Synchronization

For digital radio signal reception timing and frequency information of a transmitting station must be recovered in order to decode its signal. The synchronization used in our work was first described by [3]. Timing recovery is divided into several steps which include estimation of symbol timing along with Primary Synchronization Symbol (PSS) and Secondary Synchronization Symbol (SSS) detection. Whereas frequency recovery is divided into Integer Frequency Offset (IFO) and Fractional Frequency Offset (FFO) detection. PSS and SSS are not solely used for synchronization but also carry information on the Cell ID (N_{ID}). The N_{ID} is an identifier of the transmitting BS and has to be extracted to allow descrambling and the localization of RSs within each frame.

The first step in the synchronization process is symbol clock recovery exploiting the signal redundancy introduced by the CP. Symbol start detection is achieved by a sliding window correlation with a length of N_{CP} and a fixed lag of N_{FFT}:

$$\gamma(n) = \sum_{m=n}^{n+N_{\text{CP}}-1} r(m)\, r^*(m - N_{\text{FFT}}) \tag{1}$$

A subsequent magnitude peak detection within a search window of one OFDM symbol and averaging over the results yields a coarse estimate of the symbol timing.

In order to recover frame start there are two synchronization symbols embedded in the transmitted signal. First, the PSS which is sent every half frame is detected by correlation. The PSS is chosen amongst one of three Zadoff-Chu sequences with length-63 according to the Cell ID Number (N^2_{ID}) and enables half frame timing recovery. On the time-frequency grid it is located around the DC-carrier spanning over 6 RBs. Its 32nd value is set to zero because it is placed on the DC-carrier.

Besides half frame timing recovery using the PSS, IFO detection may be accomplished by a cross correlation with the received signal and all possible sent Zadoff-Chu sequences. FFO detection and compensation is accomplished afterwards by using the phase of the magnitude peak detected from (1).

Frame synchronization is finally achieved by a detection of the SSS. The SSS is also sent every half frame and consists of two interleaved m-sequences of length-31 according to the Cell Identity Group (N^1_{ID}). These sequences are interleaved differently depending on the SSS position within a frame and thus allow frame timing recovery.

After the synchronization is completed, the receiver has compensated frequency and timing offsets. Furthermore, the N_{ID} can be calculated from the separately extracted value N^1_{ID} and N^2_{ID}:

$$N_{\text{ID}} = 3 * N^1_{\text{ID}} + N^2_{\text{ID}} \tag{2}$$

B. Channel estimation and equalization

A transmission over the air is influenced by several types of distortion such as multipath scattering, fading and phase shifts. Especially a mobile communication system is therefore required to perform equalization on the received data which, in case of OFDM, can be achieved in the frequency domain.

For the purpose of correct data reception, the frequency response of the radio channel has to be estimated using the scattered pilots, or Reference symbols, embedded in each frame. Magnitude and phase distortion between received and known RSs are calculated and then a linear interpolation is performed to get channel estimates for the RE carrying data. These operations are based on the assumption that the maximum multipath delay is within the CP duration and channel coherence between RSs in time and frequency domain.

Equalization is performed in the frequency-domain using a one-tap equalizer per subcarrier. To simplify the channel estimation process a zero-forcing equalization was chosen for now.

C. Precoding

To improve diversity or data rate, LTE supports 2x2 and 4x4 MIMO configurations using Space Time Block Codes (STBCs) – or Alamouti Codes – [4]. In case of 2x2 MIMO the symbols $x(n)$ are coded and assigned to the transmit antennas t_n according to (3).

$$\begin{array}{c} \\ t_0 \\ t_1 \end{array} \begin{array}{c} \begin{array}{cc} \text{time}_0 & \text{time}_1 \end{array} \\ \begin{pmatrix} x(n) & x(n+1) \\ -x(n+1)* & x(n)* \end{pmatrix} \end{array} \tag{3}$$

As described above, the transmission channels introduce independent frequency flat channel distortions h_n to a receiver antenna signal $r_k(n)$.

$$\begin{array}{rcccl} r_0(n) & = & h_0\, x(n) & - & h_1\, x^*(n+1) \\ r_1(n) & = & h_0\, x(n+1) & + & h_1\, x^*(n) \end{array} \tag{4}$$

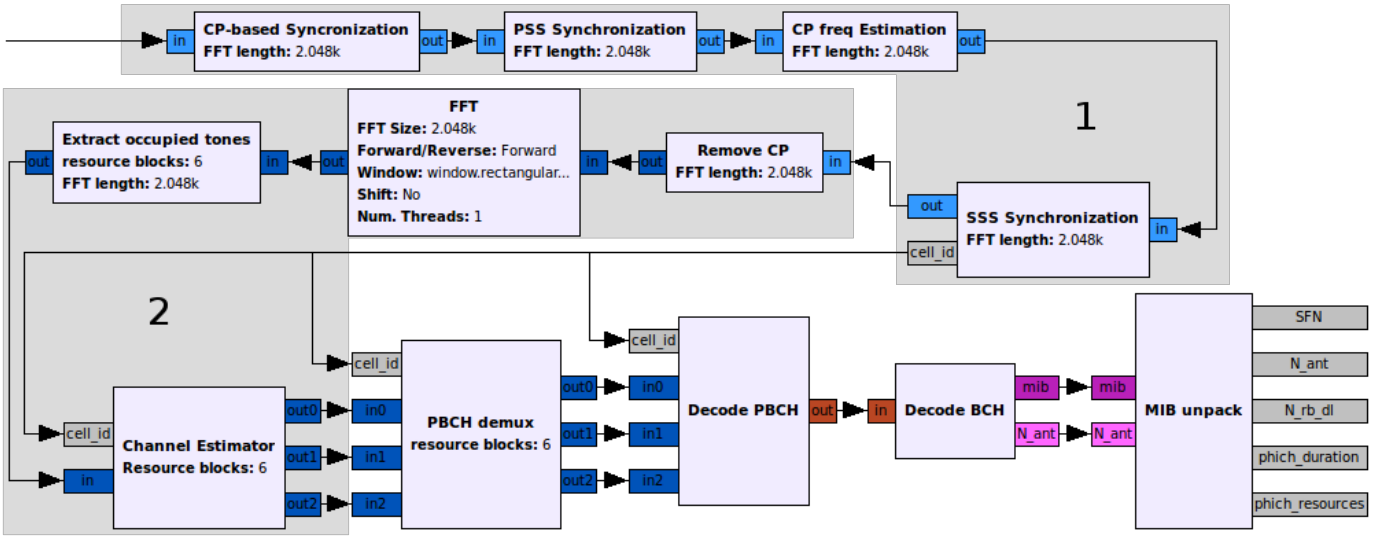

Fig. 2. GNU Radio flowgraph with Synchronization (1), OFDM receiver operation (2)

The transmitted symbols can be estimated by

$$\begin{aligned} \hat{x}(n) &= h_0^* r_0(n) + h_1 r_1^*(n) = 2x(n) \\ \hat{x}(n+1) &= h_0^* r_1(n) - h_1 r_0^*(n) = 2x(n+1) \end{aligned} \tag{5}$$

Assuming perfect equalization, the result indicates a 3 dB gain. For the LTE receiver presented here we are using one receive antenna for now.

D. Modulation and Coding

LTE supports several modulation schemes depending on the link quality. All channels can use a Quadrature Phase Shift Keying (QPSK) constellation for modulation. PDSCH and PMCH optionally allow the usage of 16QAM or 64QAM instead.

For transport channels tail biting convolutional coding and turbo coding are utilized. While PBCH uses tail biting convolutional coding the PDSCH uses turbo coding. For the control information channels a tail biting convolutional code is employed together with a block or repetition code.

III. LTE Receiver Implementation

In this section we present our GNU Radio implementation of the LTE receiver algorithms outlined in the previous section. All source code of the receiver implementation is available under the terms of GPL on Github [5].

The input can either be a recorded, generated or live data stream. It is expected to be a complex base band signal sampled at a multiple (power of two) of the LTE subcarrier spacing (15 kHz). Depending on the number of RBs to be extracted, the signal bandwidth has to be chosen according to Tab. I.

The overall flowgraph structure is presented in Fig. 2. Five main sections can be identified in this flowgraph: Synchronization (1), the OFDM receiver operation (2) and the PBCH-, BCH- and MIB-decoding.

A. OFDM receiver

In the first section synchronization is achieved together with the extraction of Cell ID (N_{ID}). The synchronization information is propagated downstream from block to block by using stream tags, which provide meta information on individual samples in the stream. The parameter N_{ID} is not associated with any sample in particular and is therefore published using GNU Radio's message passing infrastructure, which provides asynchronous data transfer capabilities between blocks.

The signal processing itself starts with the block *CP-based Synchronization* as described in Sec. II-A. Then, the PSS detection is performed in the block *PSS Synchronization*, which is a hierarchical block comprised of several blocks. After an initial acquisition phase all of these blocks are set into a tracking mode which reduces computing complexity by only correlating with the detected PSS at its expected position. Note, that due to the possibility of different lengths of the CP within one LTE slot fine frequency compensation based on CP correlation is only possible after *PSS Synchronization*. To complete the timing recovery processing, the *SSS Synchronization* block detects the SSS, computes the N_{ID} and, afterwards, also switches to tracking mode.

Once the synchronization is completed, the complex valued data stream is routed to section 2, which performs an inverse OFDM operation. This includes CP removal, Fourier transformation, extraction of subcarriers of interest and channel estimation. After those operations are completed, the received data is available as described in a time-frequency grid together with the corresponding channel estimates. All physical downlink channels can now be extracted from the time-frequency grid with their channel estimate values.

The next two sections of the flowgraph in Fig. 2 handle PBCH- and BCH-decoding. This is described in subsections III-B and III-C. The final step here is the *MIB unpack* block, which decodes the received MIB data and makes it available on

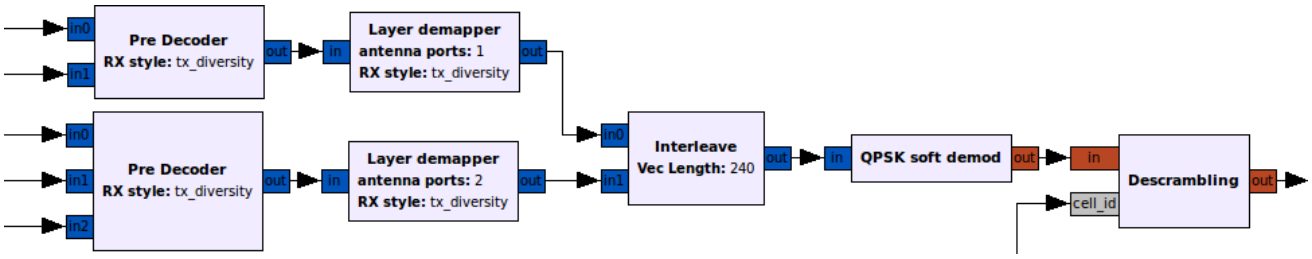

Fig. 3. Decode PBCH blocks

its message ports. It contains the number of RBs, the PHICH duration and resources and the SFN.

B. PBCH Decoding

The PBCH is transmitted in slot two of 20 using six RBs. Its payload is the BCH encoded using a repetition code, scrambled and spread over four OFDM frames. However, the whole BCH is transmitted in each OFDM frame using a different, initially unknown, phase of the scrambling sequence. In addition, the antenna configuration of the BS is unknown at this point. Therefore we process all of the different configurations in parallel *Pre Decoder* and *Layer Demapper* blocks. The resulting streams are interleaved as all further processing is the same. Once a MIB is decoded successfully, the antenna configuration can be deduced.

The block *Decode PBCH* in Fig. 2 combines the receiver side signal processing blocks of the PBCH shown in Fig. 3 into one hierarchical block [1]. First, the inverse Precoding operation is performed as indicated in Sec. II-C with channel estimate values calculated by the *Channel Estimator* block. The output of the *Pre Decoder* block is arranged according to the employed layer mapping which is described in [1] and the block *Layer demapper* does the inverse operation.

The block *QPSK soft demod* does QPSK soft demodulation and therefore marks the transition from complex to real values. In contrast to hard symbol decision soft demodulation was chosen to increase decoding performance.

The *Descrambling* block receives the N_{ID} via message passing which triggers an internal refresh of the descrambling sequence. The PBCH consists of 16 repetitions of the same data unit scrambled with a Pseudo Random Noise (PRN) sequence. For decoding, the data of one OFDM frame is repeated four times, descrambled and split into 16 data units according to the initial size of those data units. The position within the descrambled sequence of the successfully decoded data unit determines the Least Significant Bits (LSBs) of the SFN.

C. BCH Decoding

After the PBCH decoding operations the data units are passed to the hierarchical *Decode BCH* block which performs BCH-decoding [6] shown in Fig. 4.

The *Rate unmatch* block deinterleaves a data unit and by this prepares the correct sample order for the block *BCH Viterbi*. This block is a hierarchical block which parameterizes the GNU Radio Viterbi block and provides the correct data format. The output of the *BCH Viterbi* block are data units which carry the MIB information including a Cyclic Redundancy Check (CRC).

The block *CRC Calculator* checks if the received checksum is correct for different LTE antenna configurations assumed above. Depending on the antenna configuration of the BS a different scrambling scheme is chosen for the checksum and thus enables the receiver to determine this antenna configuration.

IV. Performance Analysis

In order to identify critical signal processing operations we use the profiling tool Valgrind [7]. For the performance analysis a computer with an Intel Core i3-2330M with 4 GiB of RAM running Xubuntu Linux 12.04 is used.

In Fig. 5 the different performance measurements are shown for the first running prototype and the currrent, optimized implementation. As the bar chart represents relative values, it can be seen that the sections *Decode PBCH*, *Decode BCH* and *MIB* consume far less CPU power. The performance improvements in these sections have been achieved by refactoring and optimization. GNU Radio's SIMD-optimized VOLK library has been employed where appropriate. The optimization of the synchronization section is yet to be completed and therefore shows no improvement over the initial version.

The receiver implementation allows to independently set the input sampling rate and the number of RBs processed after the inverse OFDM operation. The input sampling rate for the flowgraph directly corresponds to the Fast Fourier Transformation (FFT)-length. Thus increasing the FFT-length is expected to increase the CPU load of synchronization algorithms because these operate at the full input sampling rate. Fig. 6 shows the Instruction Fetch results of all custom GNU Radio LTE blocks. It can be observed that a smaller FFT-length results in a relative increase of CPU load for all sections but synchronization.

In Fig. 7 a comparison between different numbers of processed RBs is shown. The relative processing costs increase for

Fig. 4. Decode BCH blocks

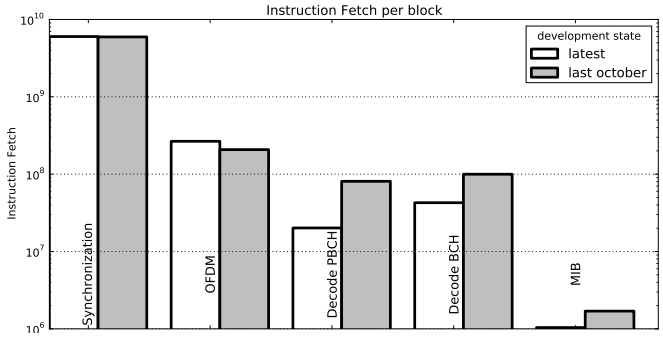

Fig. 5. Comparison: October 2012 vs. April 2013

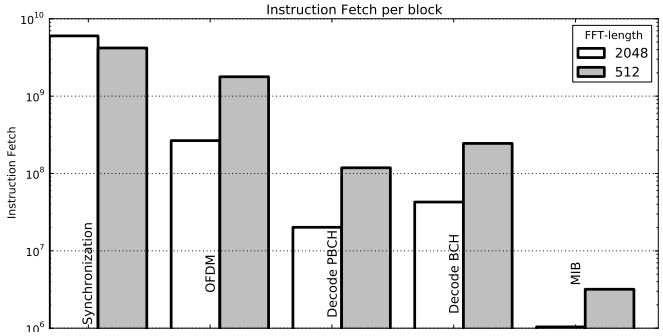

Fig. 6. Comparison: effect of different FFT-lengths

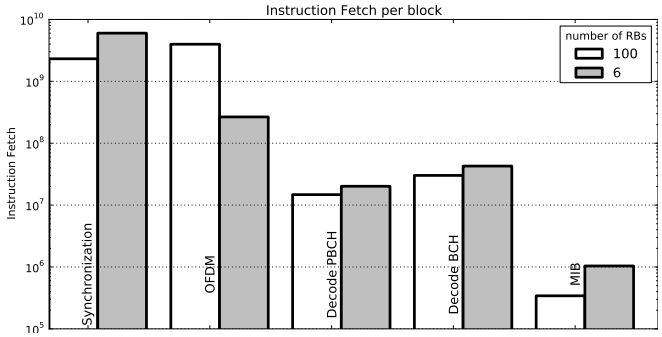

Fig. 7. Comparison: effect of different number of RBs

the OFDM section as it is the section which is configured by the number of RBs. For the synchronization section the relative CPU load increases with fewer RBs processed. However, the absolute CPU load remains the same. The processing costs shift towards the OFDM section for higher numbers of RBs processed. The main reason for that is the *Channel Estimator* block. Since we are only extracting the PBCH no additional CPU load is required in the other sections. If we extracted physical channels transported on a greater number of RBs the CPU load for their processing would increase.

V. Conclusion

In this paper we described a LTE receiver framework implementation in GNU Radio. We have outlined the algorithms necessary for synchronization, channel estimation and equalization as well as decoding arbitrary LTE physical channels. Our implementation uses advanced GNU Radio capabilities such as stream tags and message passing. Due to its modular design more functionality can easily be added by extending the flowgraph. Our future work will concentrate on further optimization, decoding of additional channels and the adoption of a partial event-based processing model which was recently introduced into GNU Radio.

References

[1] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation," 3rd Generation Partnership Project (3GPP), TS 36.211 v10.0.0, Jan. 2011.

[2] J. Mitola, "Software radios-survey, critical evaluation and future directions," *National Telesystems Conference*, May 1992.

[3] K. Manolakis, D. M. G. Estévez, V. Jungnickel, W. Xu, and C. Drewes, "A closed concept for synchronization and cell search in 3gpp lte systems," in *WCNC*, 2009, pp. 616–621.

[4] S. Alamouti, "A simple transmit diversity technique for wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 8, pp. 1451 –1458, oct 1998.

[5] gr-lte Repository. (2013, Apr.) https://github.com/kit-cel/gr-lte. accessed 11th April 2013.

[6] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), TS 36.212 v10.0.0, Jan. 2011.

[7] Valgrind Website. (2013, Apr.) valgrind.org. accessed 10th April 2013.