libLTE

A modular open source library for LTE

Ismael Gomez, Paul Sutton,
Vuk Marojevic, Antoni Gelonch
libLTE Project: Started Jan-14

What is libLTE?

Library

Modular

Computationally-Efficient

No external dependencies or frameworks

Automatic code-generation tools for:

GNURadio, ALOE, Iris, SCA, OSSIE, Matlab-MEX, etc.
# Related Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>License</th>
<th>Framw.</th>
<th>Notes</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amarisoft</td>
<td>Proprietary</td>
<td>N/A</td>
<td>Full eNodeb + MME. 2-antenna, 20 Mhz in Quadcore. Requires Boost.</td>
<td>N210</td>
</tr>
<tr>
<td>openLTE</td>
<td>AGPLv3</td>
<td>GR</td>
<td>PHY + MAC + partial RRC. Not modular</td>
<td>osmoSDR</td>
</tr>
<tr>
<td>gr-lte</td>
<td>GPLv3</td>
<td>GR</td>
<td>Synch, PBCH and PCFICH. Requires Boost.</td>
<td>N/A</td>
</tr>
<tr>
<td>LTE Cell Scanner</td>
<td>AGPLv3</td>
<td>N/A</td>
<td>Synch and PBCH. Requires Boost &amp; IT++.</td>
<td>rtl-sdr</td>
</tr>
<tr>
<td>OSLD</td>
<td>LGPLv3</td>
<td>ALOE++</td>
<td>Single antenna partial Downlink. Distributed real-time and deterministic latency.</td>
<td>UHD</td>
</tr>
</tbody>
</table>
so... why we need another LTE sw?

Library:
To enable reuse of modules

Portability:
C++ or Boost unavailable in some embedded platforms

Framework:
Sometimes too much overhead
Compatibility between frameworks

Efficiency:
It’s not just about “implementing the standard”
Objectives

Modular & Flexible

Automatic Framework Code Generation

Efficient

Design

Low-level API
Total freedom for designer

High-level API
With some rules

Use acceleration libs if available
e.g. libfftw or VOLK
Example – Low-level API

```c
void mymodule_init(mymodule_t *h, int init_parameter);
void mymodule_free(mymodule_t *h);

int mymodule_getopt(mymodule_t *h);
void mymodule_setopt(mymodule_t *h, int value);
```
typedef struct {
    mymodule_hl obj;
    /* Initialization parameters */
    struct mymodule_init {
        int init_parameter;
    } init;
    /* Run-time input parameters */
    struct mymodule_input {
        int input_parameter;
    } ctrl_in;
    cf_t *input;
    int out_len;
    /* Run-time output parameters */
    struct mymodule_output {
        int output_parameter;
    } ctrl_out;
    cf_t *output[2]; // 2 out itf
    int *out_len[2];
} mymodule_hl;

int mymodule_initialize(mymodule_hl *h);
int mymodule_work(mymodule_hl *h);
int mymodule_stop(mymodule_hl *h);

pyclibrary
mod2xml.py
xml2aloе.py
xml2gr.py
...
How do we work?

TX C code

LTE

amarisoft

Model

MATLAB

RX C code

RX implementation

Check with live BS

TX implementation
Modular Structure

ue_mac.c

ue_phy.c

register.c  cell_search.c  connected.c

...  ...

phrach.c  ...  phbch.c  sync.c

...  phdsch.c

viterbi.c  scrambling.c  ratematching.c  pss.c  sss.c  sfo.c

crc.c  precoding.c  layer_mapping.c  ch_estimation.c
Directory Structure

```
/ 
lib/ 
  fec/ ← Modules organized in categories
  filter/
  io/
  modem/
  phch/ ← LTE PHCH processing. Uses other modules
  ratematching/
  scrambling/
  sync/
    pss.c
    sss.c ← Several modules in each category
    sfo.c
    sync.c
...
include/ 
  ...
  lte.h ← A single .h includes all modules
examples/ 
matlab/ 
scripts/ ← Framework code-generation tools
```
What we’ve done so far:

Real-time cell search. Find LTE BS & sync:

- **LTE synchronization:**
  - Primary Synchronization Signal (PSS). Transmitted every 5 ms. Need to correlate with 3 possible sequences
  - Secondary Synchronization Signal (SSS). Every 5 ms. Gives Cell ID information
  - Carrier Frequency Offset (CFO). Can be estimated from the Cyclic Prefix or PSS
  - Sampling Frequency Offset (SFO). Estimated from PSS index drift

- **Cell Search:**
  - Scans LTE band get RSSI
  - For each RSSI > threshold,
    - Search for PSS correlation peak
    - If found, track N frames and get CFO, SFO and cell id
LTE Challenges

✓ Sampling Frequency
   ✓ Not all HW has 3.84 Mhz clock.
   ✓ Resampling or change DFT size?
   ✓ libfftw good performance in arbitrary lengths

✓ Synchronization
   ✓ DFT-based correlation and narrowed tracking
   ✓ 1 MHz sampling is enough (PSS/SSS 945 Khz)
   ✓ 64-point FFT

✓ Decoding is very expensive!
   ✓ Turbo decoder. SIMD maybe required.

✓ Multiuser
   ✓ Synchronization & Detection also expensive.
## Roadmap

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Component</th>
<th>Status</th>
<th>Missing parts</th>
<th>MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.1</td>
<td>Cell Search</td>
<td>95 %</td>
<td>Final opt. tuning</td>
<td>0.25</td>
</tr>
<tr>
<td>0.0.2</td>
<td>eNodeB PSS + Broadcast</td>
<td>80 %</td>
<td>Fix RM and test</td>
<td>0.5</td>
</tr>
<tr>
<td>0.0.3</td>
<td>PDCCH (eNodeB + UE)</td>
<td>70 %</td>
<td>Blind decoder and test</td>
<td>0.75</td>
</tr>
<tr>
<td>0.0.4</td>
<td>PDSCH (eNodeB + UE)</td>
<td>70 %</td>
<td>Fix RM and test. Different DCI types. HARQ.</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>Downlink Integration &amp; Testing</td>
<td>0 %</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>0.1.1</td>
<td>PRACH</td>
<td>0 %</td>
<td>-</td>
<td>0.75</td>
</tr>
<tr>
<td>0.1.2</td>
<td>PUSCH</td>
<td>50 %</td>
<td>Most of the components are same to DL</td>
<td>0.5</td>
</tr>
<tr>
<td>0.1.3</td>
<td>CQI + Control</td>
<td>0 %</td>
<td>-</td>
<td>0.75</td>
</tr>
<tr>
<td>0.1.4</td>
<td>Multiuser synch + detection</td>
<td>0 %</td>
<td>-</td>
<td>1.5</td>
</tr>
<tr>
<td>0.2</td>
<td>Uplink Integration &amp; Testing</td>
<td>0 %</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>0.2.1</td>
<td>MIMO DL &amp; UL</td>
<td>0 %</td>
<td>-</td>
<td>1.5</td>
</tr>
<tr>
<td>0.2.2</td>
<td>SIMD Turbo Decoder</td>
<td>0 %</td>
<td>-</td>
<td>1.5</td>
</tr>
<tr>
<td>0.3</td>
<td>MAC Layer eNodeB + UE</td>
<td>0 %</td>
<td>-</td>
<td>1.5</td>
</tr>
<tr>
<td>1.0</td>
<td>Full eNodeB + UE Integration &amp; Testing</td>
<td>0 %</td>
<td>-</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Conclusions

- libLTE is a new open-source project.
  - Simple library of LTE modules

- We want to:
  - Enable easy reuse of modules
  - Avoid having to use frameworks or libraries
  - But exploit their potential if available.

- Call for contributions:
  - Matlab models or C implementations are welcome!
Thanks!

github.com/ismagom/liblte