srsUE:
A high-performance software radio LTE UE
Ismael Gomez, Paul Sutton
Software Radio Systems, Ltd
http://www.softwareradiosystems.com
31/01/2016

Agenda
• Introduction to srsUE
• Architecture
• Optimizations
• Summary

srsUE is an open source software radio LTE UE
3GPP LTE Network
open source
**Features**

- AGPL License
- All bandwidths (up to 20 MHz) supported
- Transmission modes: 1 and 2
- Highly-optimized SIMO code: 160 Mbps DL @ 20 MHz
- softSIM supporting Mirasage and XDR authentication
- Detailed log system with per-layer log levels and hex dumps
- MAC layer Wireshark packet capture
- Virtual network interface (vnet/top device)
- Tested with Amtelsoft nNodeB/TPC

---

**TODO lists**

- The following features are currently not supported:
  - MIMO, transmission modes 3 and above
  - Semi-persistent Scheduling (SPS)
  - Discontinuous Reception (D Rx)
  - Aperiodic CQI reports
  - Closed-loop power control (DPC)

---

**This is how srsUE looks like**

![Diagram showing how srsUE looks]

**Agenda**

- Introduction to srsUE
- Architecture
- Optimizations
- Summary
Class/Layer Design

- srsUE is written in C++, srsLTE in C
- Each layer (PHY, MAC, RLC, PDCP, GW) is implemented in a single class
  - Some complex layers use auxiliary sub-classes
- Each layer provides a separate clean C++ pure virtual interface to any other class that make use of it (e.g. passing messages/data between layers)
- Threads only for performance or priority management reasons

Processing Latency Constraint in LTE

In LTE the basic time unit is 1 ms = 1 subframe

The processing latency constraint or critical time is given by:

\[
N \quad N+1 \quad N+2 \quad N+3 \quad N+4
\]

a) \(N\): Reception of DL grant through PDCCH
b) \(N\): Decoding of PDSCH
c) \(N+4\): Transmission of ACK/NACK HARQ through PUCCH
In LTE the basic time unit is 1 ms = 1 subframe

The processing latency constraint or critical time is given by:

\[
\begin{array}{cccccc}
N & N+1 & N+2 & N+3 & N+4 \\
\end{array}
\]

b) N: Reception of UL grant or NACK HARQ
N+4: (re-)Transmission of PUSCH

\[a+b\] N: Reception of DL grant + UL grant or NACK HARQ
N: Decoding of PDSCH
N+4: (re-)Transmission of PUSCH + ACK/NACK HARQ

In this case, we have 4 ms to receive samples from ADC, decode PDSCH, encode PUSCH and transmit samples to ADC in time.

### Some Considerations on Threading

- The maximum useful pipeline depth is 3 stages (3 threads).
- Dividing uplink and downlink in two threads is also inefficient because uplink thread has to wait for downlink thread (i.e. there is no parallelization gain)
- If more cores are available, we may divide each DSP thread and process multiple streams or codeblocks in parallel.

- Breakdown of the 4 ms deadline:
  - 1.0 ms for RX buffering
  - 0.5 ms for USRP -> Host transport
  - 2.0 ms left for processing
  - 0.5 ms for Host -> USRP transport
Further DSP optimizations

- FFT currently using libfftw and non-power of 2 sizes, allowing to reduce sampling rates, e.g.
  - 10 MHz BW: FFT 768 samples, 11.52 Msamples/s
  - 20 MHz BW: FFT 1536 samples, 23.04 Msamples/s

- This constrains us to use 32-bit complex float for transport and FFT processing.

- Yet to find a good LGPL/AGPL integer FFT library...

Turbo Decoder

- Yes... the Turbo Decoder is the most demanding component:

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Percentage of CPU @ 75 MHz</th>
<th>Percentage of CPU @ 90 MHz</th>
<th>Percentage of CPU @ 180 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo decoder (1 iteration)</td>
<td>78.14%</td>
<td>66.21%</td>
<td>50.89%</td>
</tr>
<tr>
<td>OFDM symbol stuffing</td>
<td>6.08%</td>
<td>11.76%</td>
<td>33.33%</td>
</tr>
<tr>
<td>Resource Element de-mapping</td>
<td>4.92%</td>
<td>9.31%</td>
<td>25.26%</td>
</tr>
<tr>
<td>Rate recovery</td>
<td>4.49%</td>
<td>5.64%</td>
<td>8.94%</td>
</tr>
<tr>
<td>CRC checksum</td>
<td>2.92%</td>
<td>2.23%</td>
<td>0.92%</td>
</tr>
<tr>
<td>Soft demodulation</td>
<td>1.76%</td>
<td>2.11%</td>
<td>3.38%</td>
</tr>
<tr>
<td>Equalization</td>
<td>1.16%</td>
<td>1.84%</td>
<td>4.46%</td>
</tr>
<tr>
<td>Others</td>
<td>1.53%</td>
<td>2.96%</td>
<td>55.12%</td>
</tr>
<tr>
<td>Total Execution Time</td>
<td>75.2%</td>
<td>48.1%</td>
<td>17.1%</td>
</tr>
</tbody>
</table>
Turbo Decoder

- Again, we used 16-bit integer arithmetic and 128-bit SSE4 instructions to compute all the trellis (8 states) in one shot
- With 8-bit arithmetic could do 2 Codeblocks in parallel
- With AVX2 could do 2 or 4 codeblocks in parallel

Performance Over the Air

- Amansoft eNodeB with N210
- srsUE with B210
- 1.5m separation between UE and eNodeB
- UDP test, continuous transmit (all subframes)

<table>
<thead>
<tr>
<th></th>
<th>Dual-Core 3.0 GHz</th>
<th>Quad-Core 3.6 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of DSP Threads</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10 MHz</td>
<td>30-35 Mbps</td>
<td>30-35 Mbps</td>
</tr>
<tr>
<td>20 MHz</td>
<td>35-40 Mbps</td>
<td>50-60 Mbps</td>
</tr>
</tbody>
</table>

Performance Over the Air

- Some experiences pushing the performance...
Agenda

• Introduction to srsUE
• Architecture
• Optimizations
• Summary

Summary

• srsUE is a high-performance and stable open-source LTE UE
• Huge care was put on a clean and efficient architecture for classes and threads
• Modular code with minimum inter-module dependencies. Easy to evolve to 5G
• Instrumentation facilities: logs, real-time metrics, Wireshark captures, etc.
• Currently supporting UHD and bladeRF drivers.

Thank you for your attention!

http://github.com/srsLTE

WE'RE HIRING!

Software Radio Systems, Ltd
careers@softwareradiosystems.com
http://www.softwareradiosystems.com