srsUE:
A high-performance software radio LTE UE

Ismael Gomez, Paul Sutton

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

31/01/2016

SRS

Agenda

- Introduction to srsUE
- Architecture
- Optimizations
- Summary

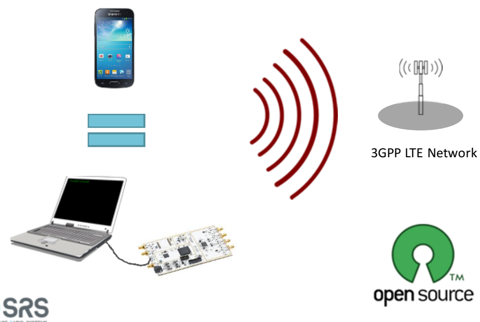
SRS

Agenda

- **Introduction to srsUE**
- Architecture
- Optimizations
- Summary

SRS

srsUE is an open source software radio LTE UE



SRS

Features

- AGPL License
- All bandwidths (up to 20 MHz) supported
- Transmission modes: 1 and 2
- Highly optimized SIMD code: +60 Mbps DL @ 20 MHz
- softUSIM supporting Milenage and XOR authentication
- Detailed log system with per-layer log levels and hex dumps
- MAC layer Wireshark packet capture
- Virtual network interface (*tun/tap* device)
- **Tested with Amarisoft eNodeB/EPC**

srsUE 1.1 now supports:

- **B210, X300 and bladeRF**
- **RRC Reconnection**
- **Paging**
- **PUCCH format 2A**

TODO lists

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

This is how srsUE looks like

```
Searching for cell...
Found CELL ID: 1 CP: Normal  , CFO: 0.1 KHz.
Trying to decode MIB...
 - Cell ID:         1
 - Nof ports:       1
 - CP:              Normal
 - PRB:             50
 - PHICH Length:    Normal
 - PHICH Resources: 1
 - SFN:             0
MIB received BW=10 MHz
Setting Sampling frequency 11.52 MHz
SIB1 received, CellID=1,  PLMN Id: MCC 1 MNC 1
SIB2 received
Random Access Transmission: seq=2, ra-rnti=5
Random Access Complete.     c-rnti=63, ta=1
RRC Connected
Network attach succesful.   IP: 192.168.3.2
RRC Connection released.
Random Access Transmission: seq=1, ra-rnti=5
Random Access Complete.     c-rnti=66, ta=1
RRC Connected
```

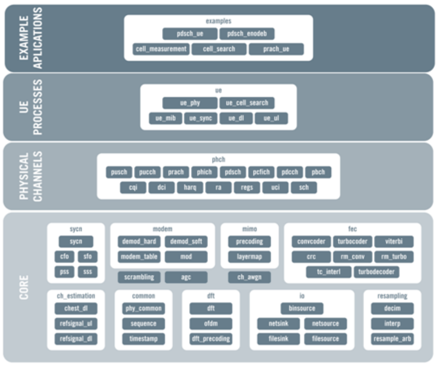
Set sampling 1.96 MHz

1. **Search for Cell (PSS, sync time)**
2. **Get Cell ID**
3. **Decode MIB**
4. **Reconfigure sampling**
5. **Receive required SIBs**
6. **Start RA procedure**
7. **Adjust Time Advance**
8. **Start *tun/tap* device with IP**
9. **eNodeB releases RRC connection**
10. **When UE-initiated IP session or paging received, start RA again**

Agenda

- Introduction to srsUE
- **Architecture**
- Optimizations
- Summary

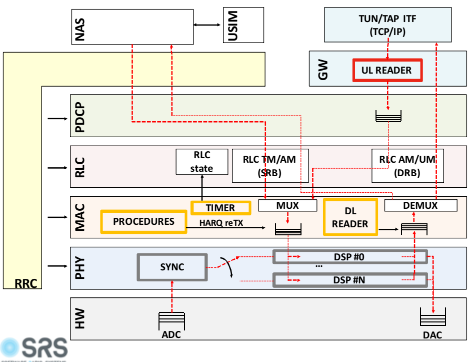
PHY DSP functions implemented in external srsLTE library



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

System Design



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	N+1	N+2	N+3	N+4

a)
N: **Reception of DL grant through PDCCH**
N: **Decoding of PDSCH**
N+4: **Transmission of ACK/NACK HARQ through PUCCH**

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	N+1	N+2	N+3	N+4

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

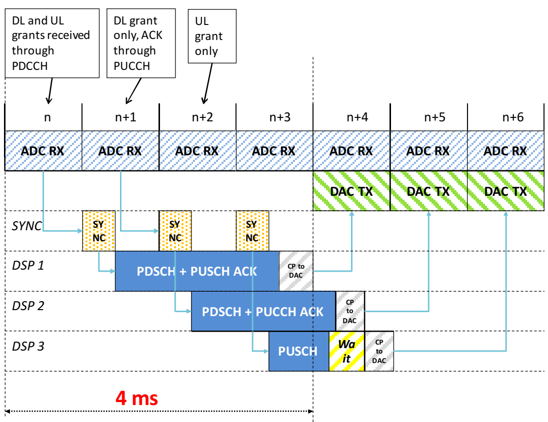
Processing Latency Constraint in LTE

BUT worse case happens in case of a union of a) and b), then:

N	N+1	N+2	N+3	N+4

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

Agenda

- Introduction to srsUE
- Architecture
- **Optimizations**
- Summary

SRS SOFTWARE RADIO SYSTEMS

Further DSP optimizations

2 essential tools: **SIMD** and **Memory**

- In srsLTE we use **VOLK** as much as we can
- Hand-written kernels using GCC intrinsics if unavailable
  - e.g. integer arithmetic
  - At compile time, choose generic/SSE4/AVX
- Initially target GPPs, so assume memory is almost free
- Use LUTs or pregenerate signals extensively:
  - Scrambling sequences
  - PUCCH signals for each subframe
  - DL RS and SRS for each subrame
  - CRC
  - Rate matching interleaver
  - …

SRS SOFTWARE RADIO SYSTEMS

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…

SRS SOFTWARE RADIO SYSTEMS

Turbo Decoder

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

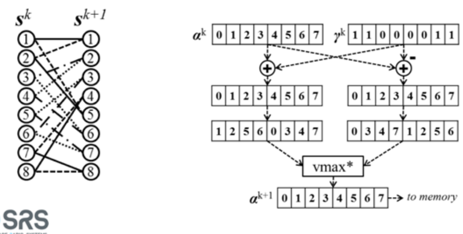
TABLE I
TOTAL PDSCH RECEIVER PROCESSING TIME AND BREAK-DOWN OF THE CPU UTILIZATION FOR 20 MHZ BANDWIDTH CONFIGURATION.

Module Name	Percentage of CPU 75 Mbps 64QAM	30 Mbps 16QAM	3.62 Mbps QPSK
Turbo decoder (1 iteration)	78.14 %	64.21 %	20.89 %
OFDM receive processing	6.08 %	11.70 %	33.33 %
Resource Element de-mapping	4.92 %	9.31 %	25.26 %
Rate recovery	4.49 %	5.64 %	8.34 %
CRC checksum	2.92 %	2.23 %	0.72 %
Soft demodulation	1.76 %	2.11 %	3.38 %
Equalization	0.16 %	1.84 %	4.98 %
Others	1.53 %	2.96 %	55.12 %
Total Execution Time	954 μs	488 μs	170 μs

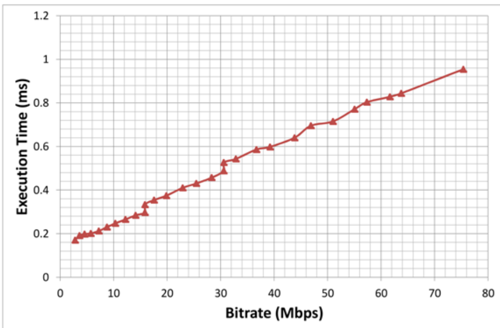
SRS SOFTWARE RADIO SYSTEMS

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



Execution time of PDSCH only @ 20 MHz



Performance Over the Air

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

	Dual-Core 3.0 GHz	Quad-Core 3.6 GHz
Number of DSP Threads	2	3
10 MHz	30-35 Mbps	30-35 Mbps
20 MHz	35-40 Mbps	50-60 Mbps

Performance Over the Air

- Some experiences pushing the performance

Agenda

- Introduction to srsUE
- Architecture
- Optimizations
- **Summary**

SRS

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.

SRS

Thank you for your attention!

http://github.com/srsLTE

WE'RE HIRING!

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

SRS