OpenAirInterface and ExpressMIMO2

Mobile Communications Dept.

Presenter: R. Knopp, Eurecom

Acropolis Summer School on Cognitive Radio

July 12th , 2013
London, England

Overview of the tutorial

- Objectives of OpenAirInterface for Experimental Research in Wireless Networks
- ExpressMIMO2 Hardware
- Efficient x86 building-blocks for LTE / 802.11 Modems
- Real-Time Processing
- Experimental Application Scenarios

www.openairinterface.org

- **Provides open-source (hardware and software) wireless technology platforms**
  - target innovation in air-interface technologies through experimentation
- **We rely on the help of**
  - Publicly-funded research initiatives (ANR,ICT,CELTIC)
  - Direct contracts with industrial partners
  - Widespread collaboration with a network of partners using open-source development and tools
    - LINUX/RTAI based SW development for PCs
    - LEON3/GRLIB-based HW and SW development for FPGA targets
    - LINUX networking environment
  - Experimental Licenses from ARCEP (French Regulator) for medium-power outdoor network deployments
    - 1.9 GHz TDD, 5 MHz channel bandwidth
    - 2.6 GHz FDD (two channels), 20 MHz channel bandwidth
    - 800 MHz FDD (two channels) : 10 MHz channel bandwidth
    - 3.5 GHz TDD expected for future experimentation

OpenAirInterface Development Areas

Cognitive Technologies
Wideband RF, Agile Spectrum Management, Interference Management and Control, Distributed/Collaborative techniques, Spectrum Sensing, Cognitive and Flexible Radio Architectures, Ambient Networking

OPENAIR3 : Wireless Networking
All-IP, Mobility Management, 802.21, Cellular/Mesh Routing Protocols, Mesh Topology Management, Multimodal Radio Resource Management

OPENAIR2: Medium-Access Protocols
Cellular topologies, single-frequency resource allocation, cross-layer wideband scheduling, Mesh topologies, distributed resource control

OPENAIR1: Baseband/PHY
Advanced PHY (LTE), Propagation Measurement and Modelling, Sensing and Localization Techniques, PHY Modeling Tools

OPENAIR0: Wireless Embedded System Design
Agile RF design, Reconfigurable High-end Transceiver Architectures, FPGA prototyping, Simulation Methodologies, Software development tools, low-power chip design

Collaborative Web Tools

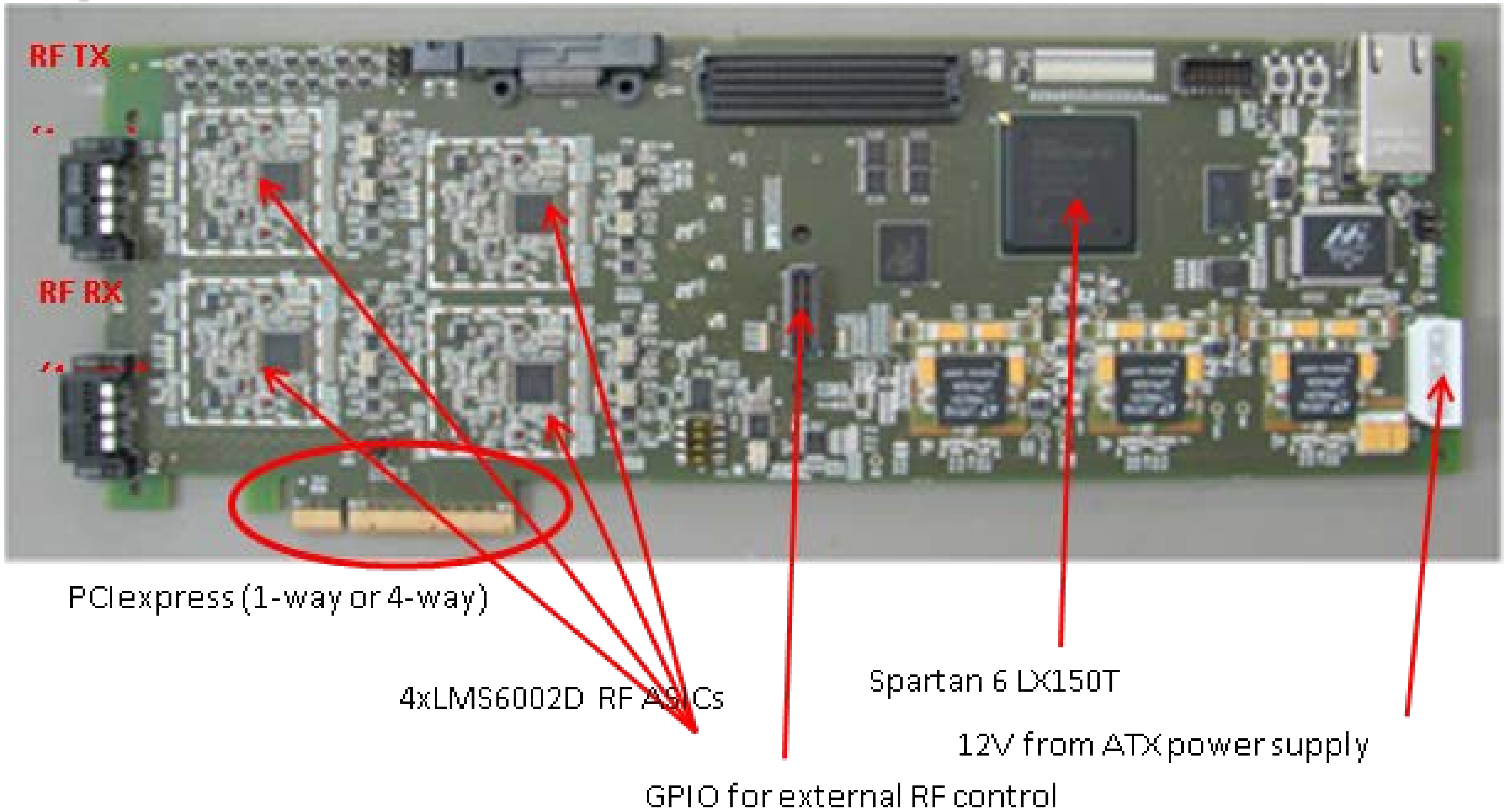
- **OpenAirInterface SVN Repositories**
  - All development is available through www.openairinterface.org's SVN repository (openair4G) containing
    - OPENAIR0 (open-source real-time HW/SW)
    - OPENAIR1 (open-source real-time and offline SW)
    - OPENAIR2 (open-source real-time and offline SW)
    - OPENAIR3 (open-source Linux SW suite for cellular and MESH networks)
    - TARGETS : different top-level target designs (emulator, RTAI, etc.)
  - Partners can access and contribute to our development
- **OpenAirInterface TWIKI**
  - A TWIKI site for quick access by partners to our development via a collaborative HOW-TO
- **Forum**
  - external support services (not currently used effectively)
- **Mailing list**
  - openair4G-devel@eurecom.fr

ExpressMIMO2 Data Acquisition

Data Acquisition

- **ExpressMIMO2 makes use of memory-mapped DMA transfers on a PCIe link**
  - ExpressMIMO2 firmware automatically
    - triggers transfers of signals to/from PC memory
    - Provides synchronization (via interrupts and time stamps)
- **PCIe is a serial-bus protocol**
  - V1.0 provides 2.5 Gbit/s bi-directional (case of Spartan6 on ExpressMIMO2), V2.0 5 Gbit/s, V3.0 8 Gbit/s, and V4.0 16 Gbit/s per lane (up to 32 lanes)
  - Transaction based protocol
    - Memory Transactions are packet based between Endpoint (e.g. ExpressMIMO2) and Root Port (e.g. Motherboard I/O Chipset)
    - Can make use of real-time interrupts for RTOS.

ExpressMIMO2 Architecture

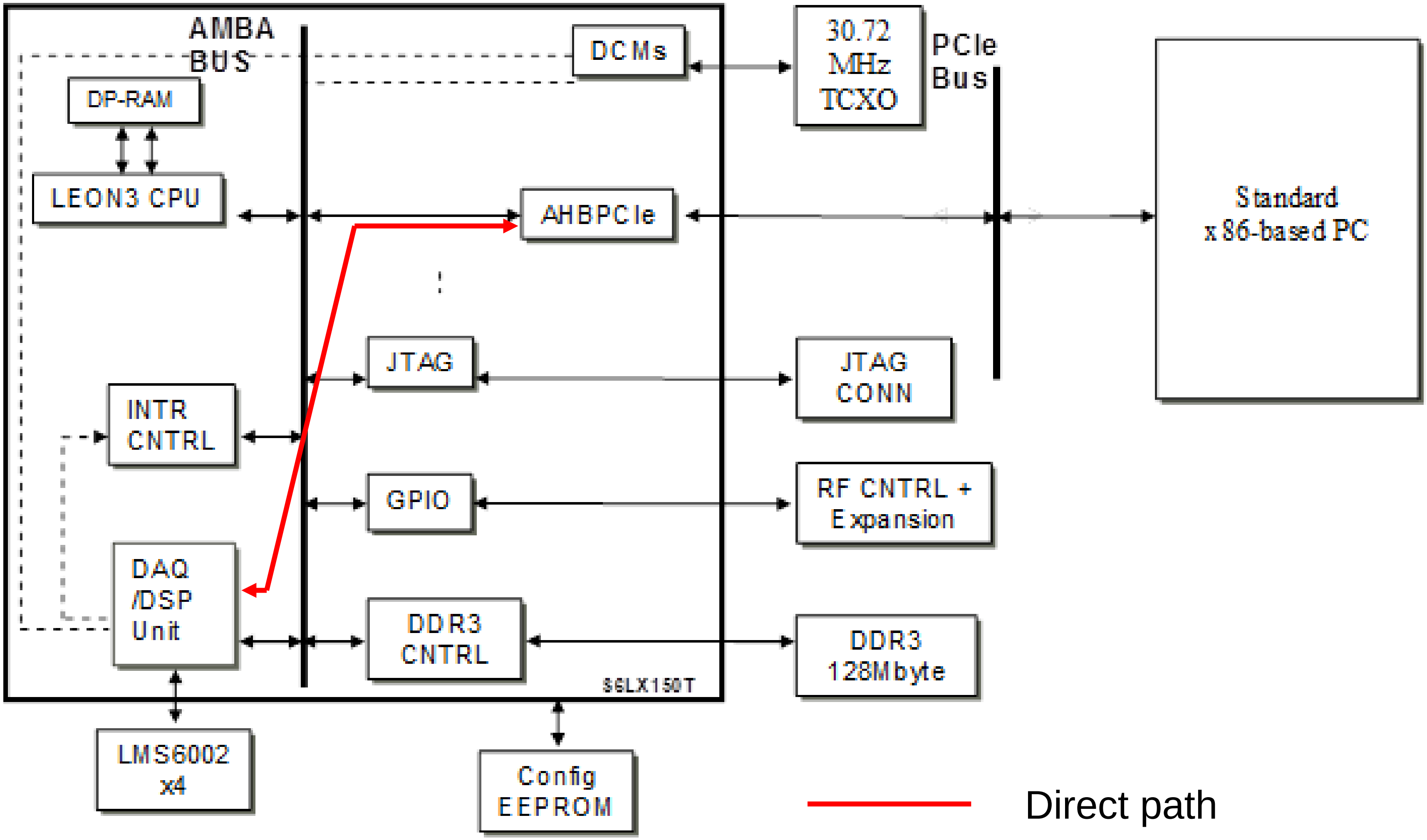


- **Real-time RF acquisition card**
  - 4 RF front-ends with A/D, D/A conversion
  - 1-way PCIe

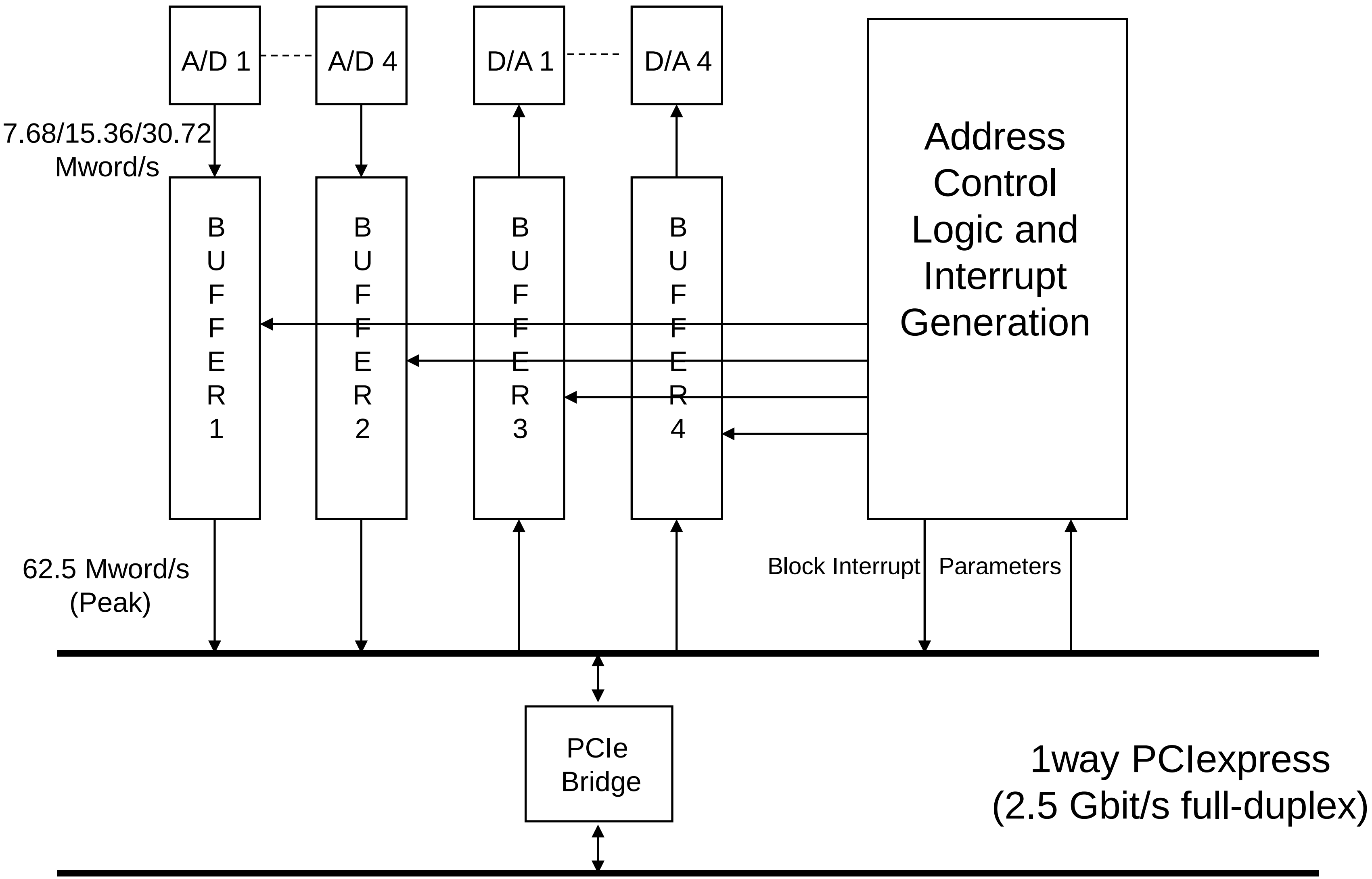
ExpressMIMO2

- **Spartan-6 FPGA**
  - Provides PCIe interface (1-way using Xilinx PCIe endpoint, 2.5 Gbps (full-duplex), 200 Mbytes/s real throughput in each direction (approx 40 Msamples/s for 32-bit complex samples)
  - Provides micro-controller (LEON3) for C-language configuration of RF chipsets and acquisition state-machine
- **Lime MicroSystems LMS6002 RF front-ends**
  - Integrated multi-band RF (300MHz – 3.8 GHz) and 12-bit A/D, D/A
  - High-linearity

ExpressMIMO2 Embedded System



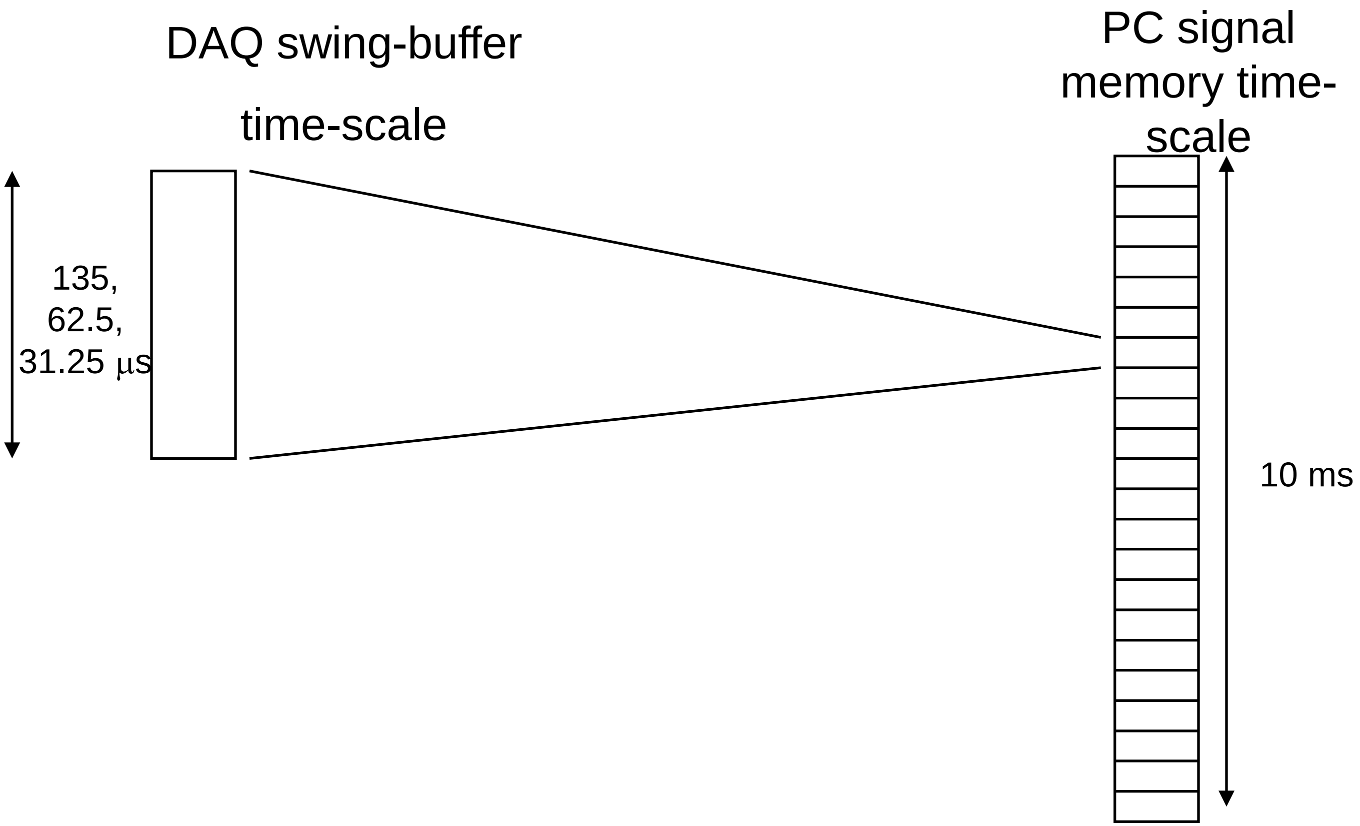
ExpressMIMO2 A/D, D/A PCIe interface



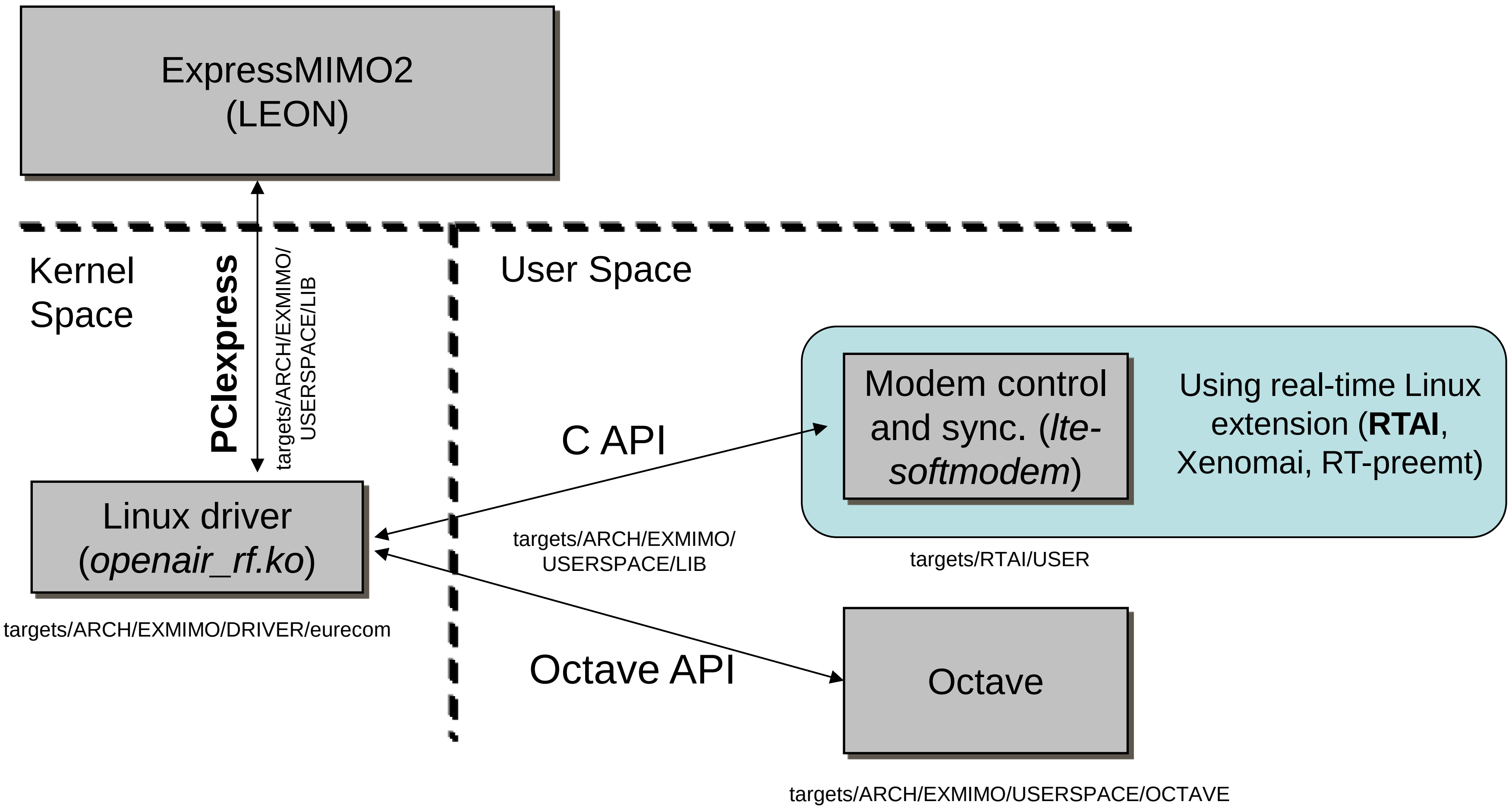
Services provided via PCIe

- **Command Registers**
  - Accessible via PCIe memory read/write from PC
  - Accessible via APB I/O read/write from LEON3
  - Functions
    - Trigger AHB interrupt (PCIe write)
    - Trigger PCIe legacy interrupt (APB write)
    - Trigger AHB ↔ PCIe DMA
    - Trigger DAQ ↔ PCIe DMA
  - LEON3 action register (PCIe write)
    - combined with AHB interrupt, provides a mechanism to trigger macro operations in LEON3 firmware (e.g. configure RF, start acquisition, etc.)
  - PCI descriptor for LEON3 to retrieve data structures via DMA (PCIe write)
  - DMA Address pointers (APB write)
  - DMA length/direction (APB write)
- **Status Register**
  - Readback of PCIe DMA error / status (APB)
  - Readback of interrupt status (PCIe or APB)
- **Boot firmware**
  - Allows PC driver to download firmware into LEON memory system via DMA and then execute

PC Memory View



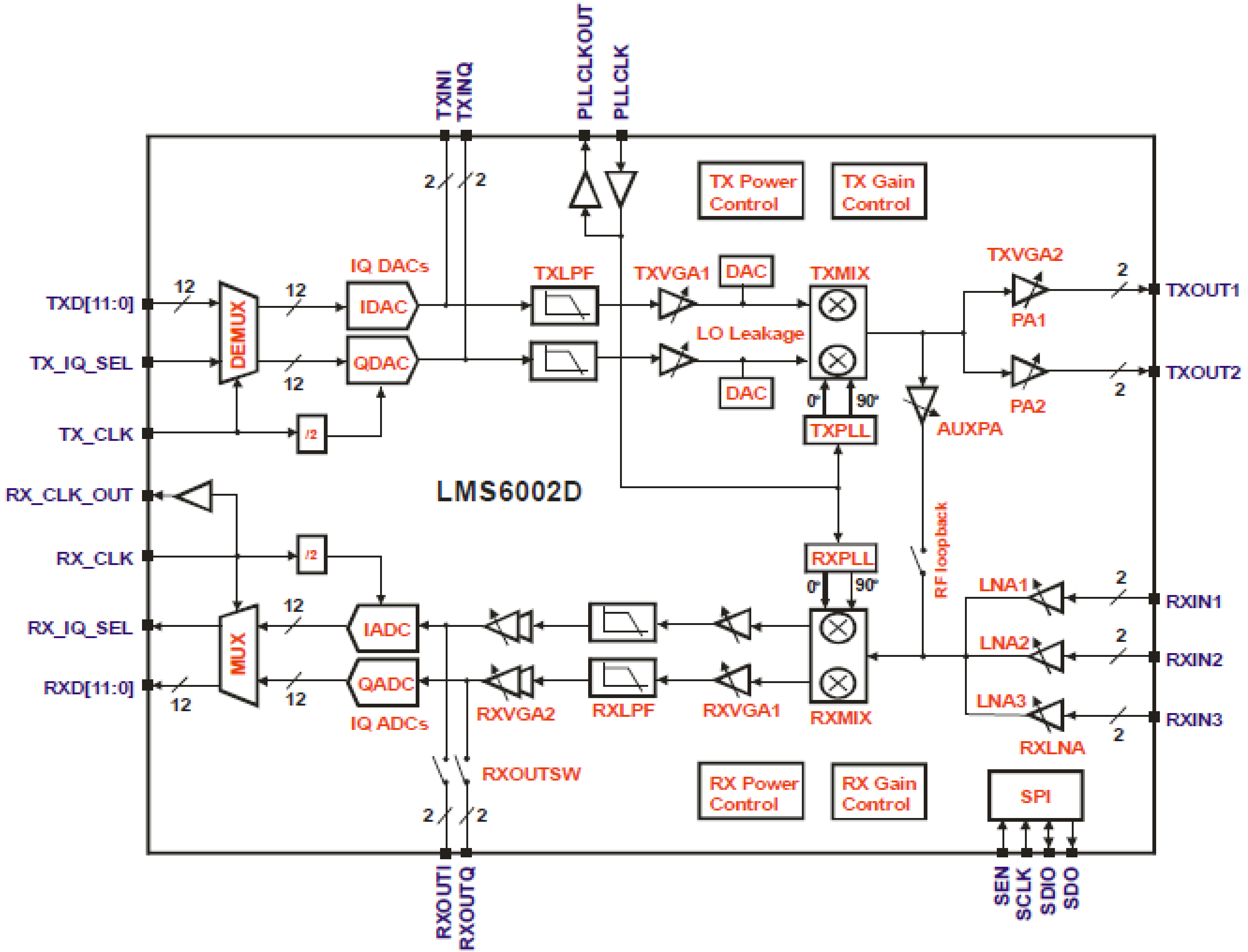
ExpressMIMO2 software architecture



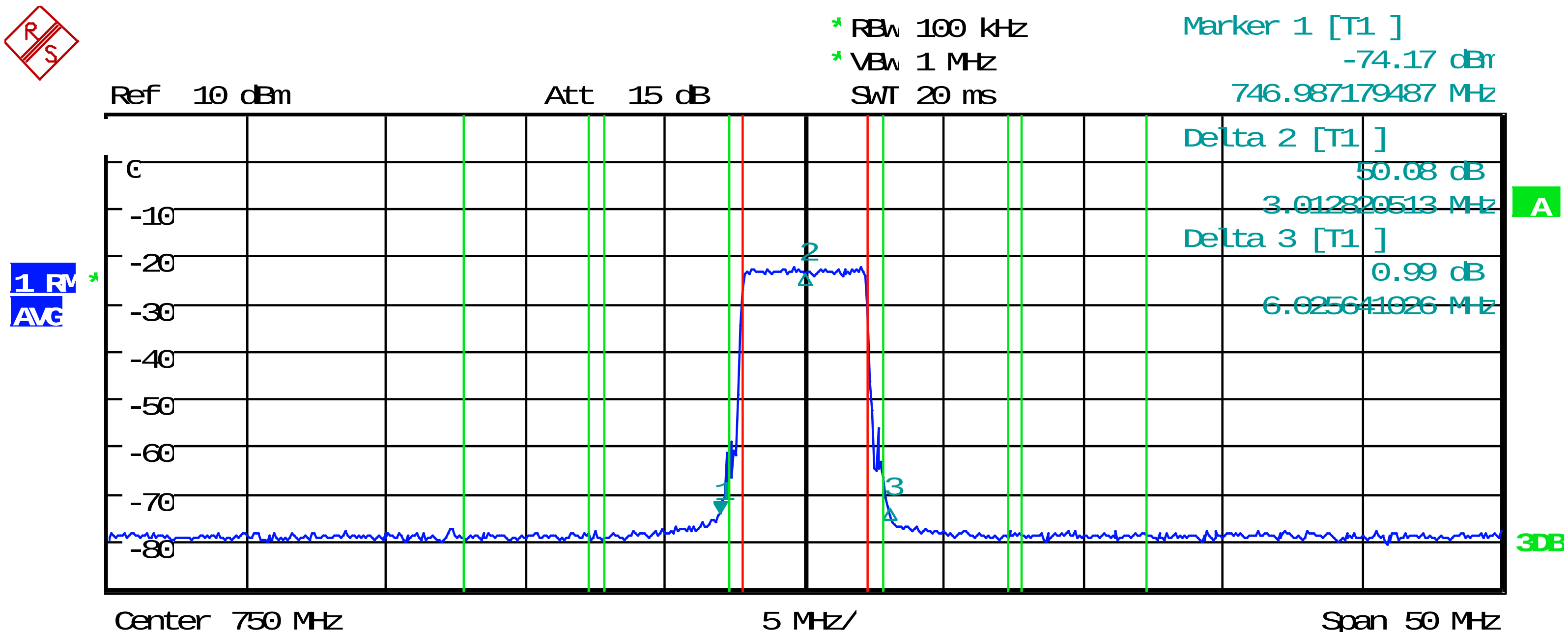
Lime RF

- **LMS6002 provides a very high-performance RF solution**
  - 350MHz – 3.8 GHz tuning frequency
  - TDD or FDD operation
  - SPI programming interface
    - Frequency control
    - Auto-calibration procedures
  - -5 dBm transmit level
  - 120 dB RX gain
  - Approximately 10 dB noise figure (without external LNA)
  - Programmable bandwidth baseband filters (1.5 – 28 MHz)
  - Integrated A/D and D/A converters (12-bit)
  - Feedback paths
- **Potential DSP enhancements off-chip via Spartan6**
  - RX IQ imbalance correction

Lime LMS6002D

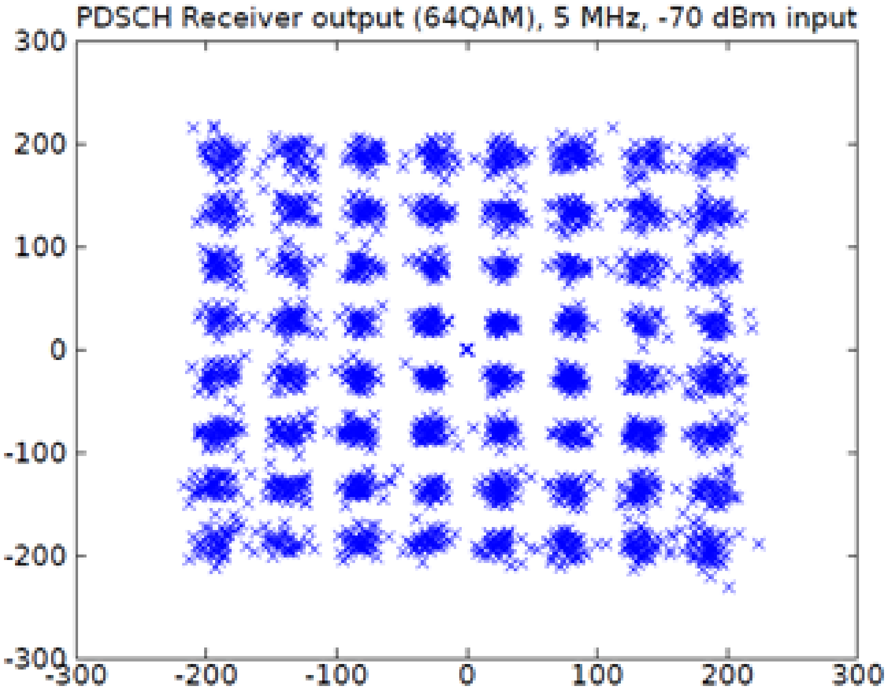


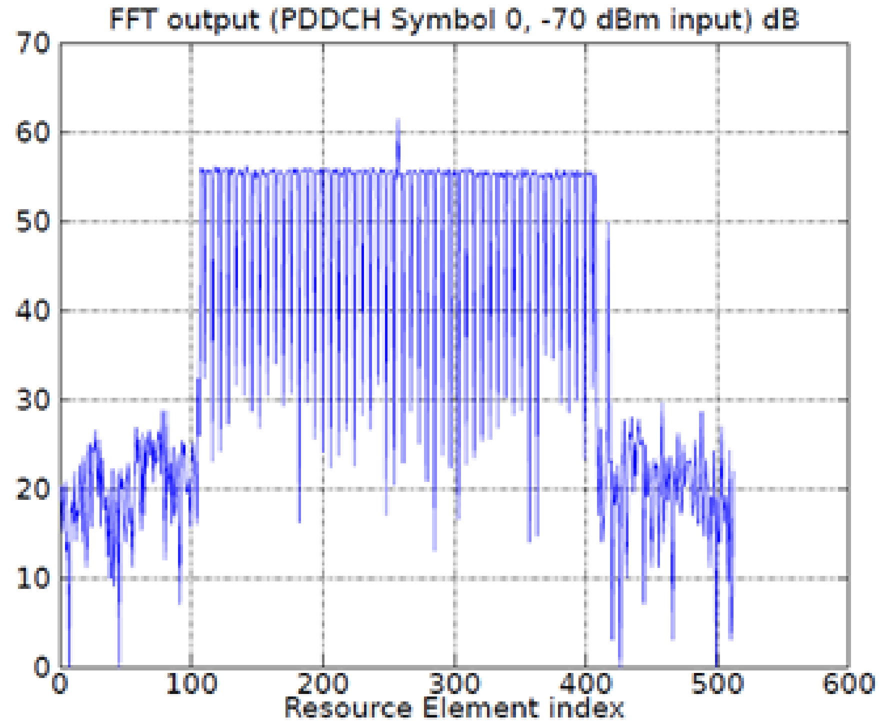
ExpressMIMO2 + LMS6002 (TX, 750 MHz)



Tx Channel			**E-UTRA/LTE Square**
Bandwidth	4.5 MHz	Power	-7.29 dBm
Adjacent Channel			
Bandwidth	4.5 MHz	Lower	-51.70 dB
Spacing	5 MHz	Upper	-52.18 dB
Alternate Channel			
Bandwidth	4.5 MHz	Lower	-55.56 dB
Spacing	10 MHz	Upper	-55.43 dB

ExpressMIMO2 + LMS6002 (RX, 2.6 GHz)





RTOS issues

- **Low-latency radio applications for PHY (e.g. 802.11x,LTE) should run under an RTOS**
  - eCos/MutexH for generic GNU environment
  - RTAI for x86
  - VXWorks ($$$)
- **Example: RTAI / RT-PREEMPT kernel can achieve worst-case latencies below 30μs on a loaded-PC. More than good enough for LTE, but not 802.11x because of MAC timing.**
- **Should make use of POSIX multithreading for SMP**
  - Rich open-source tool chains for such environments (Linux, BSD, etc.)
  - Simple to simulate on GNU-based systems for validation in user-space
  - Allow each radio instance to use multiple threads on common HW

Issues with Linux

- **Scheduler latency**
  - Kernel is not pre-emptible
  - Overhead in disabling/enabling interrupts
- **Mainstream kernel solutions**
  - Kernel preemption (RT-PREEMPT) – mainstream until 2.6.32 (patches afterwards)
  - Latency reduction (soft-RT kernels)
- **Patches / dual-OS solution**
  - ADEOS + RTAI/Xenomai

RTAI/Xenomai

- **Real-time nano-kernel tightly integrated with Linux kernel**
  - Linux runs as a low-priority thread under RTAI/Xenomai and and both share same memory space
  - Both RTAI and Xenomai make use of ADEOS which is a hardware abstraction framework which allows several OS to share HW resources
    - Low-level Scheduler allows Linux kernel to run in background
  - Both RTAI and Xenomai provide user-space real-time functionality in addition to kernel-only applications
    - Provide various API flavours to resemble classical OS (e.g. Posix, VXWorks, RTDM, etc.)

RT-PREEMPT Kernels (borrowed from rt.wiki.kernel.org)

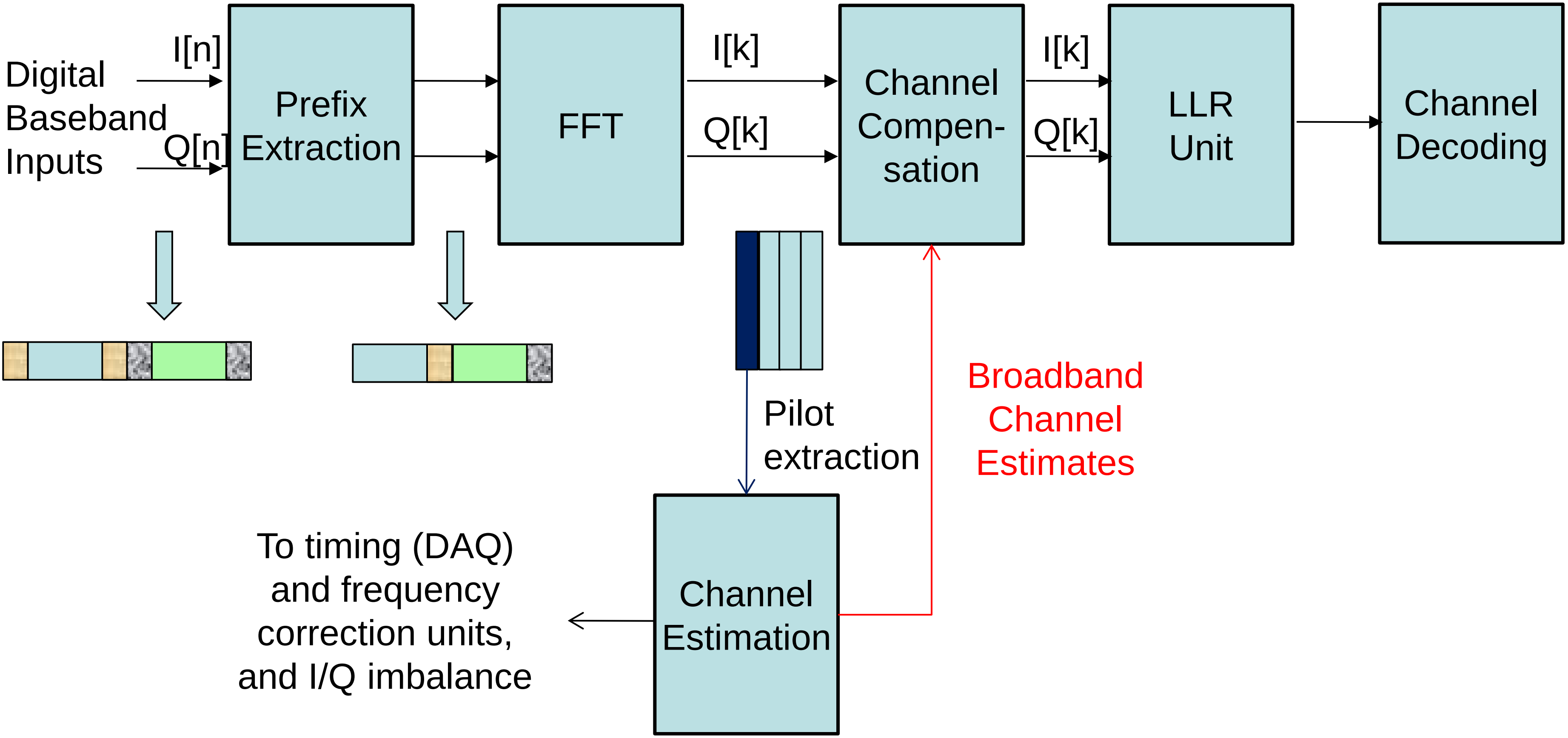
- **The RT-Preempt patch converts Linux into a fully preemptible kernel. The magic is done with:**
  - Making in-kernel locking-primitives (using spinlocks) preemptible though reimplementation with rtmutexes.
  - Critical sections protected by i.e. spinlock_t and rwlock_t are now preemptible. The creation of non-preemptible sections (in kernel) is still possible with raw_spinlock_t (same APIs like spinlock_t).
  - Implementing priority inheritance for in-kernel spinlocks and semaphores. For more information on priority inversion and priority inheritance please consult Introduction to Priority Inversion.
  - Converting interrupt handlers into preemptible kernel threads: The RT-Preempt patch treats soft interrupt handlers in kernel thread context, which is represented by a task_struct like a common user space process. However it is also possible to register an IRQ in kernel context.
  - Converting the old Linux timer API into separate infrastructures for high resolution kernel timers plus one for timeouts, leading to user space POSIX timers with high resolution.
- **In our experience, RT-PREEMPT provides slightly lower-performance w.r.t. RTAI, but the other advantages greatly outweigh this penalty.**

x86 Building Blocks for LTE/802.11 MODEMS

x86 Baseband DSP

- **One approach for DSP is to use SIMD engines on general-purpose CPUs (e.g. x86) for front-end and channel decoding**
  - This was started (first?) in the SpectrumWare project in the late1990s at MIT for GSM soft-basestations and resulted in a company called VANU inc. (still exists!)
  - A couple of years later (1999) we developped Wireless3G4Free.com (precurser to OpenAirInterface.org) which did the same for TD-SCDMA
  - OpenAirInterface (2007) provided OFDMA => OpenAir4G (2011)
- **Key elements**
  - Real-time extensions to Linux OS (later)
  - Real-time data acquisition to PC
  - SIMD optimized integer DSP (64-bit MMX → 128-bit SSE2/3/4 → 256-bit AVX2)

Generic OFDM Receiver



Key linear processing operations which can be vectorized easily with SIMD

- Inner-products (synchronization, projection operations)
- Componentwise products (compensation/equalization, channel estimation)
- FFT/IFFTs (OFDM TX/RX)
- Peak search in vector (synchronization)
- Vector addition, subtraction, energy, absolute value

Coding with SIMD

- **SIMD = Single Instruction Multiple Data**
  - Operate on vectors of samples with parallel instructions
- **Good for linear processing in front-end and trellis-processing (turbo/Viterbi decoding)**

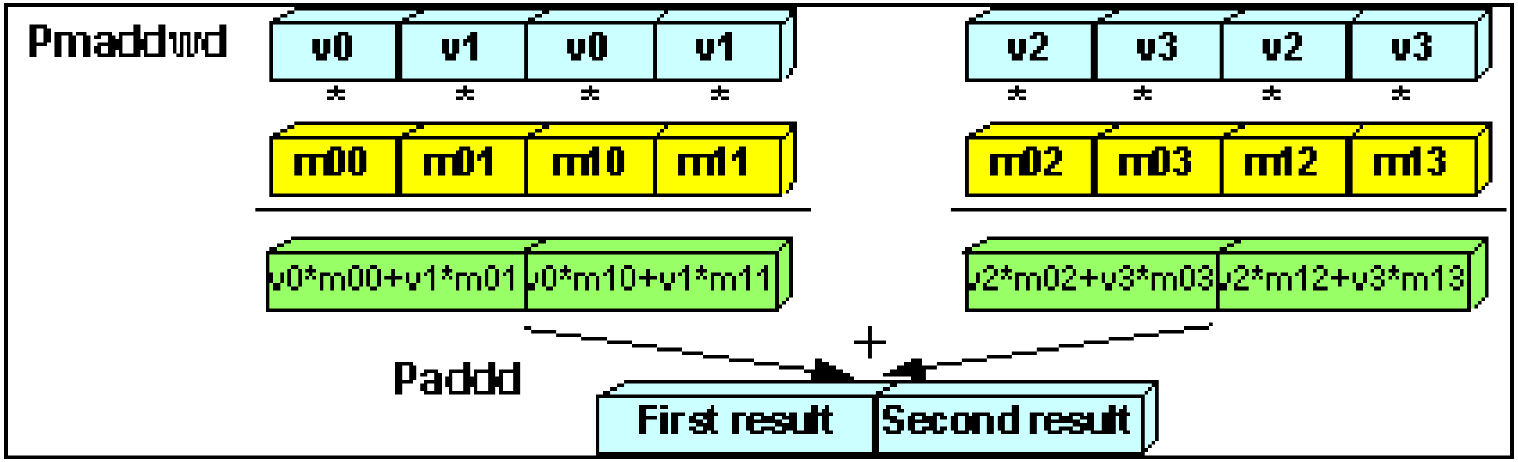
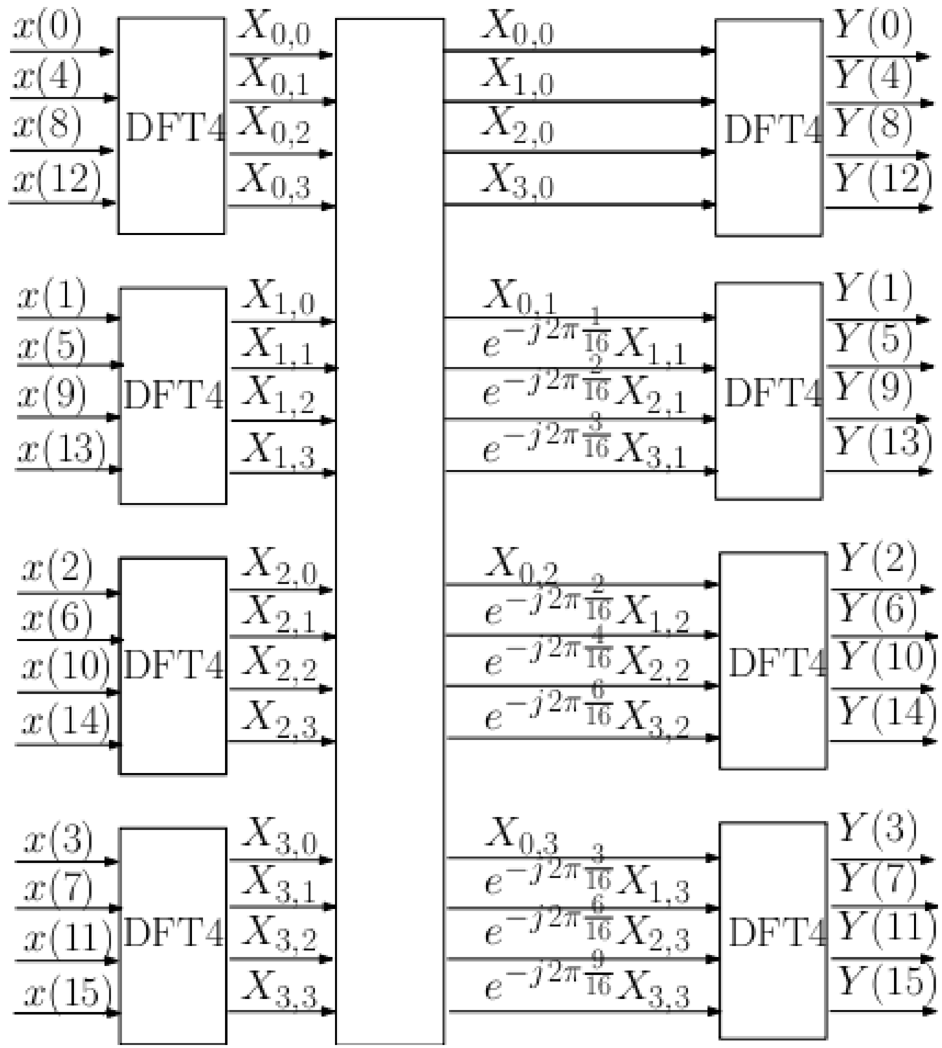


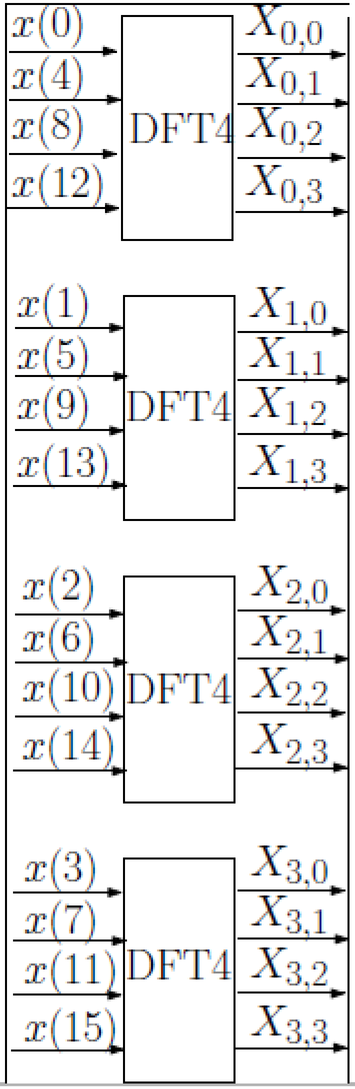
Figure 5. MMX Technology Matrix-Vector Multiplication

OAI Radix-4 DFT

- **Basic idea is to optimize**
  - the 16-point DFT with SIMD
  - the 16-bit complex radix-4 and radix-2 butterflies
  - Build up 64,128,256,512,1024,2048 from there
- **Two butterfly stages (radix 4), one with twiddles**
- **One permutation**



OAI DFT with SSE4



=

```
register __m128i x1_flip,x3_flip,x02t,x13t;
register __m128i xtmp0,xtmp1,xtmp2,xtmp3;

 // First stage : 4 Radix-4 butterflies without input twiddles
 x02t    = _mm_adds_epi16(x128[0],x128[2]);
 x13t    = _mm_adds_epi16(x128[1],x128[3]);
 xtmp0   = _mm_adds_epi16(x02t,x13t);
 xtmp2   = _mm_subs_epi16(x02t,x13t);

 // multiplication of x1 and x3 by sqrt(-1)
 x1_flip = _mm_sign_epi16(x128[1],*(__m128i*)conjugatedft);
 x1_flip = _mm_shufflelo_epi16(x1_flip,_MM_SHUFFLE(2,3,0,1));
 x1_flip = _mm_shufflehi_epi16(x1_flip,_MM_SHUFFLE(2,3,0,1));
 x3_flip = _mm_sign_epi16(x128[3],*(__m128i*)conjugatedft);
 x3_flip = _mm_shufflelo_epi16(x3_flip,_MM_SHUFFLE(2,3,0,1));
 x3_flip = _mm_shufflehi_epi16(x3_flip,_MM_SHUFFLE(2,3,0,1));

 x02t    = _mm_subs_epi16(x128[0],x128[2]);
 x13t    = _mm_subs_epi16(x1_flip,x3_flip);
 xtmp1   = _mm_adds_epi16(x02t,x13t);  // x0 + x1f - x2 - x3f
 xtmp3   = _mm_subs_epi16(x02t,x13t);  // x0 - x1f - x2 + x3f
```

Performance and Extensions

- **OAI Radix-4 DFTs provide essentially the same performance as FFTW**
  - Approx 300 cycles for 64-point DFT on a x86-64 core i5/i7/Xeon machine (gcc generated)
  - For larger DFTs, both are slightly less than the closed IPP libraries from Intel (based on SPIRAL code generator)
  - OAI doesn't make use of FFTW only because we use the DFT optimizations for teaching SIMD coding to our Engineering diploma students.
- **OAI also provides all the mixed-radix DFTs that are required for UE and eNB DSP**
  - $2^a 3^b 5^c$ for SC-FDMA (uplink transmission formats)
  - $2^N 3$ for random-access preamble (PRACH TX at UE and RX at eNB)

Turbo Decoding

- **The primary (by far) bottleneck in an HSPA/LTE receiver is the turbo decoder**
- **SIMD parallelization can be done in different ways**
  - OAI adopts a scalable SIMD *max-logmap* approach whereby code blocks (8-bit LLRs) are split in 16 (SSE4) or 32 (AVX2) equal size sub-blocks
  - integer x86 SIMD arithmetic is used to efficiently perform the forward-backward BCJR recursions (16 or 32-way parallel add, subtract, max) and extrinsic information computation
  - Care must also be taken when it comes to the interleaving and data reforming operations in a turbo decoder. These can also benefit from SIMD
- **x86-64 is much more efficient for Turbo decoding than legacy x86 because of the fact that the number of available registers is doubled**

OAI Modem Performance (eNB)

- **gcc 4.7.3, x86-64 (1.8 GHz Xeon E5-2650L), 20 MHz bandwidth (mcs 19 – 16QAM, transmission mode 1 - SISO)**

eNB RX function statistics (per 1ms subframe)

OFDM_demod time :202.992302 us (100 trials)
ULSCH demodulation time :347.516264 us (100 trials)
ULSCH Decoding time (39.23 Mbit/s, avg iter 2.000000) :1271.786873 us (100 trials, max 3032.161541)
|__ sub-block interleaving 18.597523 us (700 trials)
|__ demultiplexing 213.908853 us (100 trials)
|__ rate-matching 19.591607 us (700 trials)
|__ turbo_decoder(5632 bits) 103.893272 us (187021 cycles, 700 trials)
|__ init 16.228172 us (cycles/iter 14606.423571, 700 trials)
|__ alpha 5.040021 us (cycles/iter 18145.403571, 2800 trials)
|__ beta 5.173861 us (cycles/iter 18627.263571,2800 trials)
|__ gamma 1.609997 us (cycles/iter 5796.413571,2800 trials)
|__ ext 3.851109 us (cycles/iter 13865.007857,2800 trials)
|__ intl1 3.933486 us (cycles/iter 7080.792857,1400 trials)
|__ intl2+HD+CRC 7.454629 us (cycles/iter 13419.312857,700 trials)

OAI Modem Performance (eNB)

OFDM_mod time	**:176.144838 us (100 trials)**
DLSCH modulation time	**:55.319101 us (100 trials)**
DLSCH scrambling time	**:22.194255 us (100 trials)**
DLSCH encoding time	**:79.016000 us (100 trials)**
|__ DLSCH turbo encoding time	**:27.376582 us (200 trials)**
|__ DLSCH rate-matching time	**:19.505191 us (200 trials)**
|__ DLSCH sub-block interleaving time	**:20.249680 us (200 trials)**

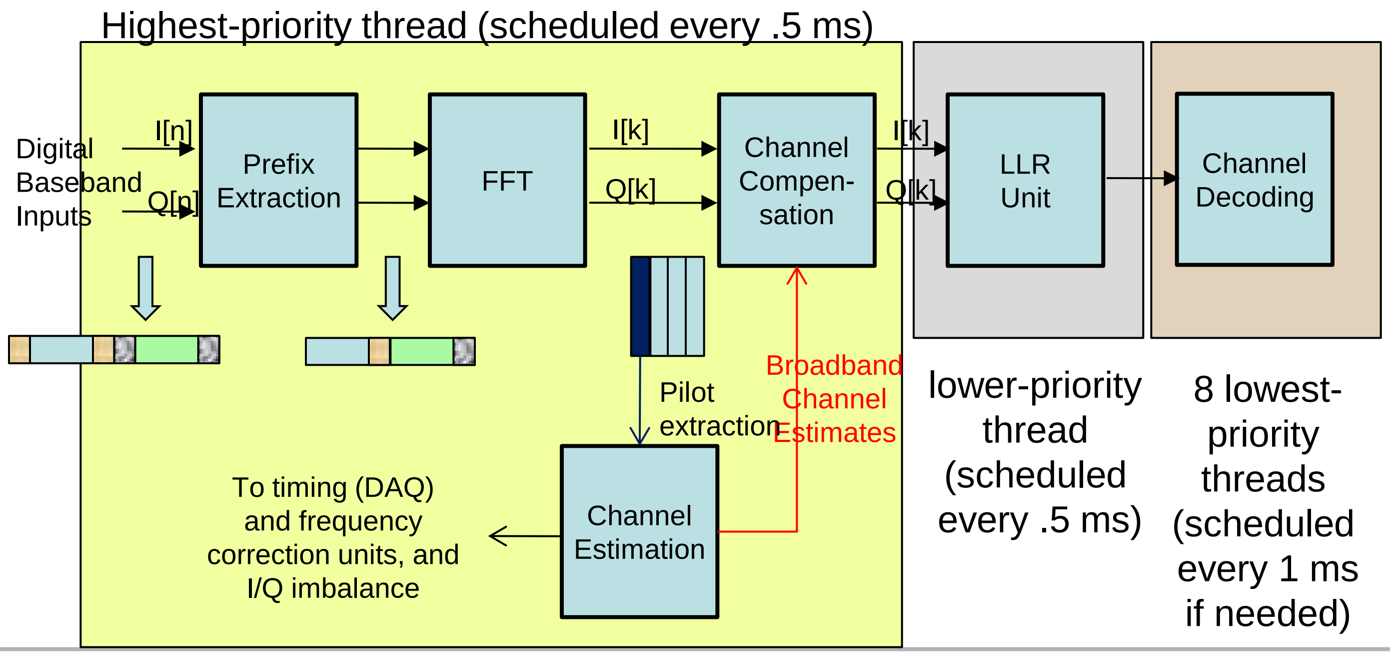
- **Summary (processing for 1ms subframe)**
  - RX : 1820 μs (< 2 cores)
  - TX : 330 μs (1/3 core)
- **On 3 GHz machine, < 2 cores for 20 MHz eNB**
- **On future AVX2 (256-bit SIMD), turbo decoding and FFT processing will be exactly twice as fast**
  - 1 core per eNB

802.11a/p MODEMS (OpenAirITS)

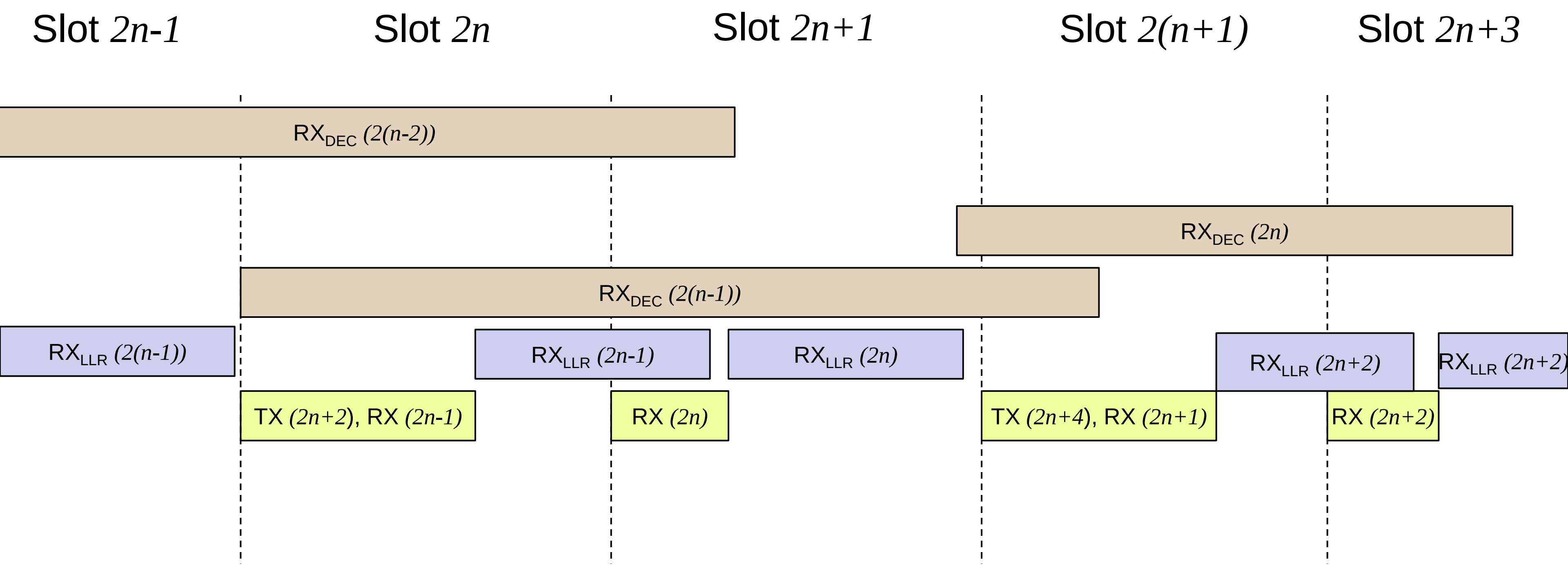
- **Similar (simpler) MODEMs using the same DSP tools are available for real-time 802.11a/p applications**
  - Integrated with standard mac80211 interfaces (Intel / atk) for Linux 802.11 networking
  - Main issue is that we do not have an acceptable method for ACK/NAK, RTS/CTS exchanges because of tight latency constraints

Real-time Soft-MODEMS

- **Map OAI DSP to HW using RTAI or RT-PREEMPT**
  - RT-PREEMT has benefit that it is compatible/transparent with standard Linux
    - Same application can be executed (with RT faults) on regular kernel
  - Map modem processing based on time-scales to multiple threads
    - Allows efficient use of available CPU cores
    - Allows for prioritization of processing



Real-time SoftModems



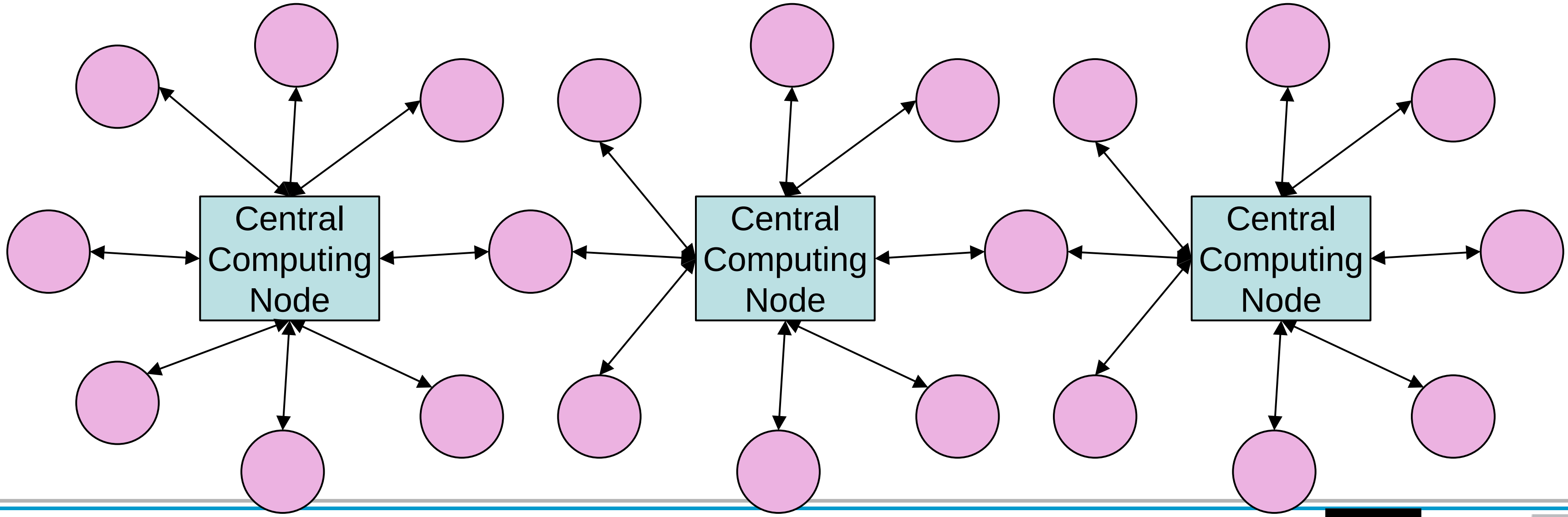
Some Applications

Experiments with real UEs

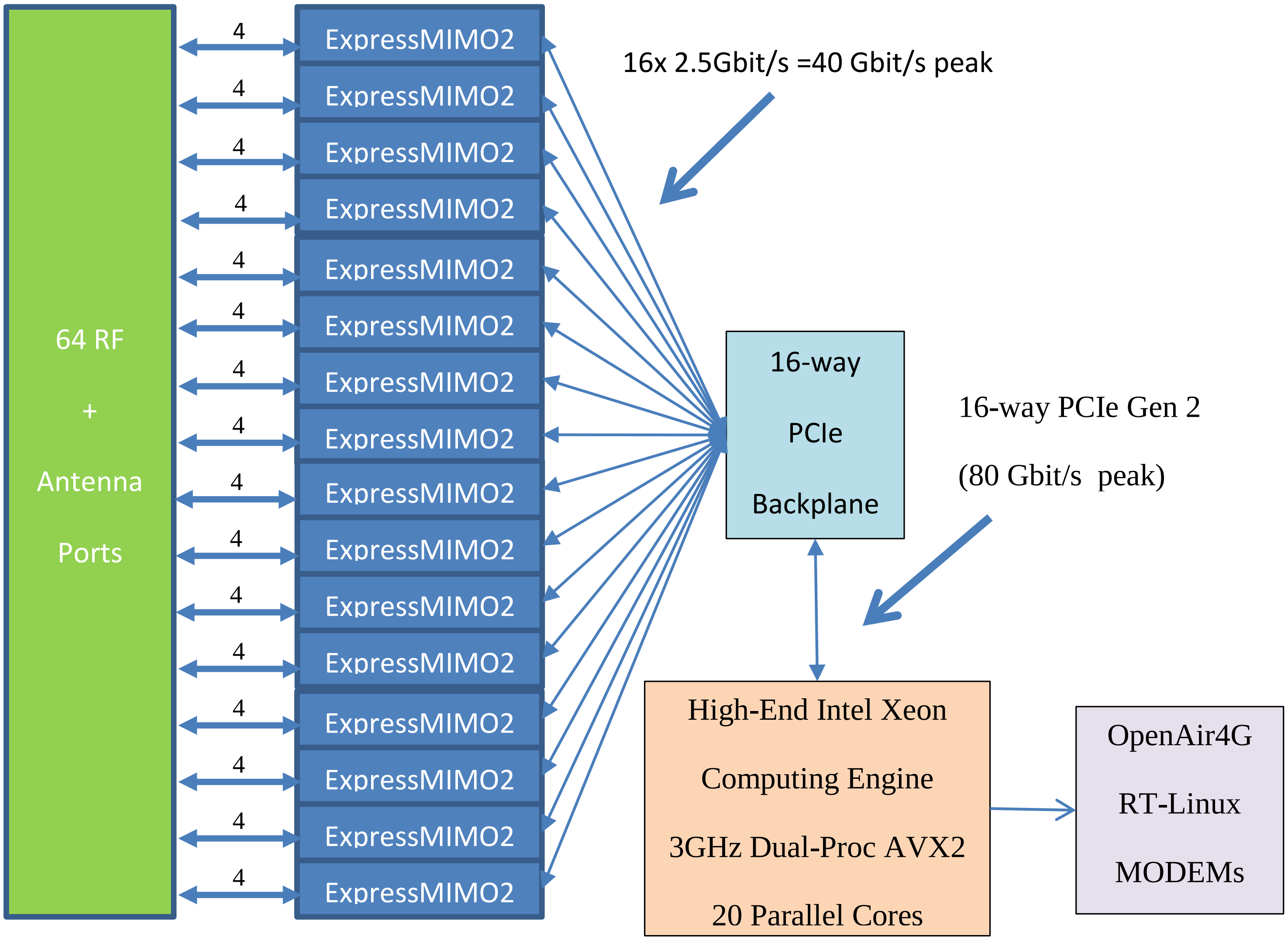
- **Interconnect OAI eNB implementation with ExpressMIMO2 to Multiple commercial UEs**
  - Using synthetic channel simulation in eNB TX can test
    - RRM policies (dynamic interference, etc.)
    - Handover performance
    - MIMO performance
    - Traffic scheduling policies
  - Provides a controllable and repeatable emulation environment

A look to the future for "Massive" SDR

- **Research in PHY networking is going "massive"**
  - Tens or hundreds of lower-power distributed antennas per basestation
    - next generation "Remote-Radio-Heads"
  - Large-scale collaboration between basestations at the signal level
    - "Light radios" each serving a small set of terminals, but collaborating heavily with their neighbouring basestations



Parallel use of ExpressMIMO2

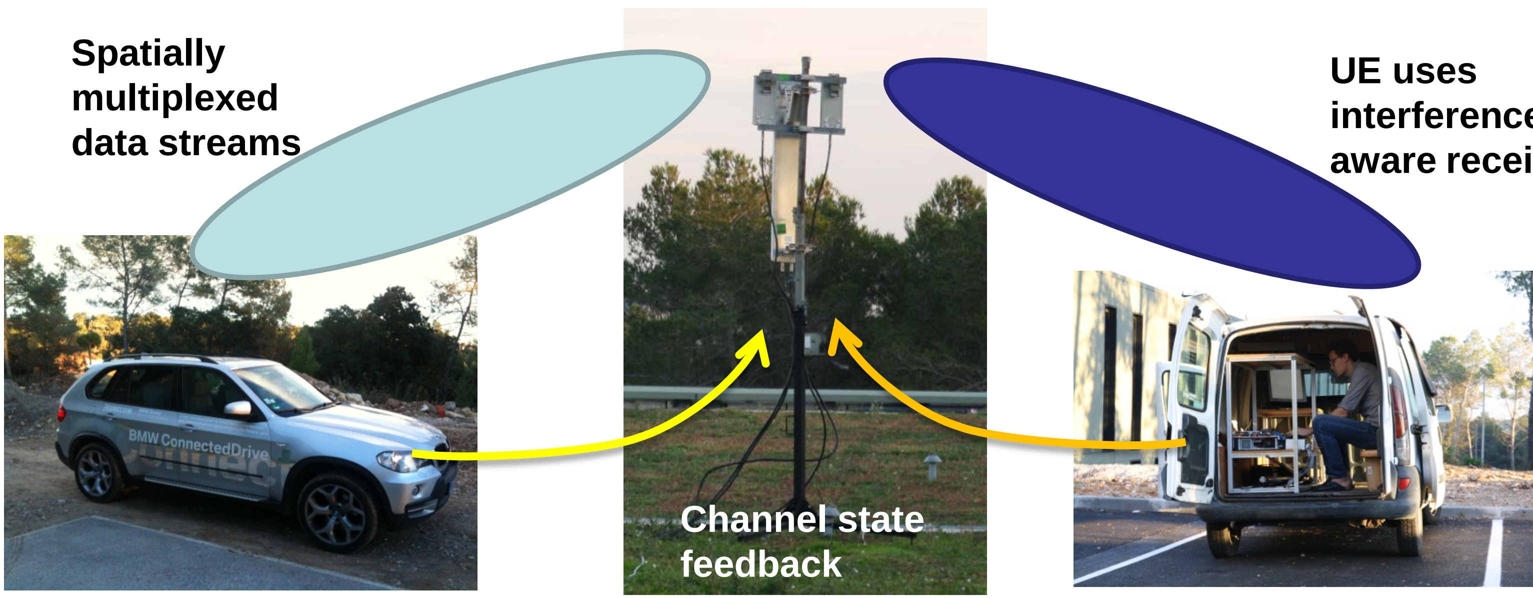


Notes

- **Imagine replacement of ExpressMIMO2 with PCIe-based CPRI interface interconnected with commercial RRH (Alcatel, ZTE, etc.)**
  - Large Chinese industrial project on “cloud-ran” (3CRAN)
    - Some partners (IBM, Agilent, Orange) evaluating OAI for this type of system
  - Alcatel-Lucent Bell Labs / EURECOM collaboration on OAI (CIPRI / 40Gethernet switch)
- **Largest XEON server**
  - 16 10-core processors on common motherboard (160 AVX2 cores!)

Outdoor Experimentation

- **SAMURAI (finished Dec 2012)**
  - Demonstrated benefits for interference aware receiver in an LTE MU-MIMO scenario (transmission mode 5)
  - Video streaming to two simultaneous users
  - Precoder selection based on real-time feedback



Concluding Remarks

- **Looking for collaboration on OAI**
  - EURECOM can arrange for refabrication of ExpressMIMO2
  - Several major industrial players are interested in different parts of the development (ALU Bell-Labs, IBM, Agilent, Orange, Thales)
  - Software can be used for rapid simulation
  - Several laboratory sessions can be made available for teaching purposes
    - Digital communications
    - Radio engineering
    - LTE MODEM design