A Software Radio Testbed for UMTS TDD Systems

Christian Bonnet, Giuseppe Caire, Alain Enout, Pierre A. Humblet, Giuseppe Montalbano, Alessandro Nordio, and Dominique Nussbaum[1]

Institut Eurécom,[2] B.P. 193, 06904 Sophia–Antipolis CEDEX, France
Tel: +33 4 93 00 26 08, Fax: +33 4 93 00 26 27
E-mail: firstname.name@eurecom.fr

Abstract *– Today's and near future mobile communication systems need to provide universal seamless connection to the users between different standards and types of service. This requires a significant degree of flexibility that can be provided by Software Defined Radio (SDR) terminals. Eurécom and EPFL have started a joint project whose objective is to study and implement a real-time SDR communication platform to validate advanced algorithms for wireless communications. The platform is a real-time PC based system that can handle wide-band radio resources. It provides hardware, DSP software, and link-level software functionalities. Due to various practical design issues, the platform implements the essential physical layer features of the time-division duplex (TDD) mode of the UMTS standard proposal (air-interface and signal processing), although frequency-division duplex (FDD) mode and even other standards could also be implemented. In this paper we address the major general issues inherent to the design of a real-time SDR system. Then we focus on our SDR platform by describing the general architecture, the signal processing techniques adopted to implement the transmitter and receiver SDR front-end, the current set-up, and DSP performance measurements. Finally, we address the future perspectives for the platform evolution.*

1. Introduction and Motivation

The presence of several different wireless communication standards and the wide variety of services provided by mobile communication operators pose the problem of providing universal seamless connection to users. Software defined radio (SDR) terminals able to reconfigure represent a solution to this problem. SDR is a very broad term involving several levels in the protocol stack, (see e.g. [2]–[4] and references therein). Motivated by the intensive activity on software radio technologies (see e.g. [1]), Eurécom and EPFL (École Polytechnique Fédérale de Lausanne) have started a joint project with the objective of designing and implementing a real-time software radio communication platform to validate advanced mobile communication signal processing algorithms. The following major features characterize the platform:

- *Flexibility*, achievable by a software driven system.
- *Duplex communication*, necessary to allow higher layer protocol services, and to analyze more complex system aspects, such as multi-access and power control, and optimize downlink signal processing from uplink measurements.
- *Multiple antennas* transmit and receive signal processing.

The UMTS/TDD standard proposal has been considered for a first implementation because of several reasons. For instance, contrary to the FDD mode the TDD mode only needs a single band for both uplink and downlink. Under certain conditions TDD allows reciprocity between the uplink and downlink channels. This feature turns to be very useful to perform effective downlink transmit signal processing exploiting uplink channel measurements, in particular when employing multiple antennas. The UMTS/TDD proposal also envisages the use of short periodic spreading codes (contrary to the FDD mode) allowing the design and implementation of sophisticated signal processing algorithms (e.g. multiuser detection).
In the sequel we first provide a description of the general platform architecture. Then we consider end-to-end SDR signal processing solutions for efficient DSP algorithm implementation, followed by a description of the current platform set-up and detailed DSP performance measurements. Finally we address the near future perspectives for the platform evolution.

2. Platform Architecture

A single antenna architecture for both the Mobile Terminal (MT) and the Base Transceiver Station (BTS) has been chosen for a first implementation. This architecture can be easily upgraded without substantial re-design of

[1] The authors appear in alphabetical order.
[2] Eurécom's research is partially supported by its industrial partners: Ascom, Cégétel, France Télécom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

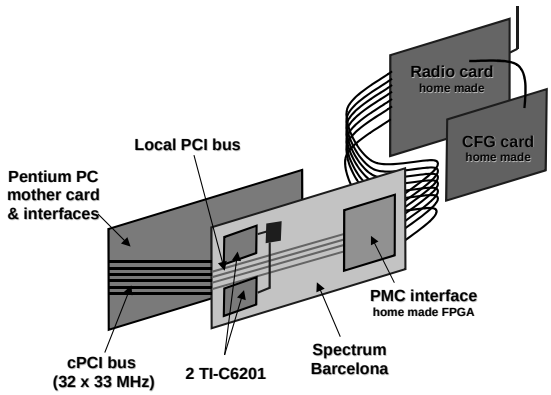

Figure 1: Current hardware setup

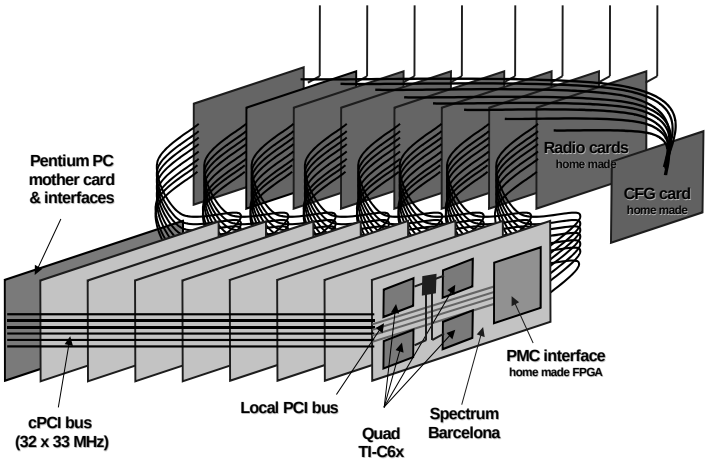

Figure 2: Target hardware architecture for the BTS

the essential hardware and software components. The BTS and the MT are based on similar hardware. The hardware is highly partitioned in order to maintain the maximum modularity and flexibility, allowing the use of different standard cards and components. The platform consists of two main subsystems: a signal processing subsystem and a radio subsystem. The signal processing subsystem comprises:

- A configurable data acquisition system based on Field Programmable Gate Array (FPGA) and PCI bus technology
- A PCI bus-based DSP system employing a combination of embedded DSPs and workstations
- Data management software (data routing, framing, synchronization)
- Signal processing software (digital transceiver algorithms, multiple-access protocols, and error coding/decoding).

These elements may be replicated in a parallel fashion to implement a multi-antenna system (both at the BTS and MT).

Figure 1 shows the current hardware setup:

- A radio card capable to handle a 5 MHz bandwidth radio signals
- An A/D, D/A conversion card (ADAC)
- A data acquisition card that is connected as a *mezzanine* (PMC) via a local PCI bus
- A 2-processors DSP card based on Texas Instruments TMS320C6x (in the sequel denoted as TI C6x) technology supporting signal processing at chip level
- A Pentium PC card (main CPU) that supports the processing at symbol level and the higher layer protocols

The target architecture for the BTS (figure 2) will include the following elements:

- 8 radio frequency cards
- A clock and frequency generation card (GF card)
- 8 data acquisition cards with 8 ADACs
- 8 DSP cards. Four-processors DSP cards as shown in figure 2 might replace the 2-processors DSP cards.

It is important to notice that the same architecture can be adopted to develop different real-time and non-real-time applications by properly defining different operating modes and developing the related software. The operating modes will include real-time signal processing for UMTS like applications, real-time processing for narrow-band signals (e.g. GSM, EDGE), non-real-time off-line processing to test high complexity algorithms or collecting measured data, and hardware simulation. Software plays an essential role at each stage of the digital processing chain: at the acquisition level via the use of programmable FPGAs, and at signal processing level via the use of DSPs. With the target system architecture one should be able to define the application and the corresponding operating mode so that the dedicated software can be downloaded and the platform properly configured.

3. End-To-End Signal Processing

This section gives an overview of the signal processing algorithms which have been designed and coded on the DSP to implement in real-time the essential UMTS/TDD physical layer. We start the analysis considering the transmitted data flow (for example a video stream) already coded and mapped onto the QPSK or BPSK alphabet.

3.1. Transmitter Front-End

Let $a[k]$ denote a sequence of chips and let $\psi(t)$ denote the pulse-shaping filter, band-limited over $[-W/2, W/2]$ and T_c the chip interval. The corresponding continuous-time complex base-band equivalent linearly modulated signal is given by

$$x(t) = \sum_k a[k]\psi(t - kT_c) \tag{1}$$

For DS-CDMA systems [13], using a spreading factor N, $a[k] = b\lfloor k/N \rfloor c[k]$ where $b[m]$ is the mth modulation symbol (QPSK in the UMTS case) and $c[k]$ is the kth chip (this generalizes trivially to systems with several spreading layers, like IS-95). Notice that (1) can also represent the sum of the chip sequences associated with different users. In general, for digital transmitters, the signal $x(t)$ is the output of a D/A converter, which takes as input the discrete-time signal $x[n] = x(n/f_s)$ with sampling frequency $f_s \geq W$. In classical I-Q modulators, the continuous base-band components $\mathrm{Re}\{x(t)\}$ and $\mathrm{Im}\{x(t)\}$ are generated by low-pass filtering the output of two separate D/A converters, and the IF signal

$$y(t) = \mathrm{Re}\left\{x(t)\exp(-\mathrm{j}2\pi f / f_{\mathrm{IF}})\right\} \tag{2}$$

is then produced by mixing $\mathrm{Re}\{x(t)\}$ and $\mathrm{Im}\{x(t)\}$ with IF carrier signals in-phase and quadrature and by summing the modulated real signals [13]. This approach requires two D/A converters, two low-pass filters, two analog mixers and one adder. Thus it looks quite costly from the hardware point of view. Another approach consists of producing a sampled version of the IF modulated signal $y(t)$ by using a sampling rate f_s greater than $2f_{\mathrm{IF}}$. The continuous-time signal is obtained by band-pass filtering the output of a single D/A converter. This solution is described in [6] and [7] for the receiver front-end. However, since intermediate frequencies usually range between tens of MHz up to 100MHz, this approach is extremely computationally intensive, as it requires the generation of signal samples and the multiplication by the carrier signal at extremely large sampling frequency. In the following we propose an efficient transmitter front-end architecture that allows a working sampling frequency of the order of the baseband signal bandwidth (and not of the order of the IF carrier). It does not require explicit multiplication by the carrier signal, and only needs a single D/A converter and an analog filter centered at f_{IF}.

3.1.1. IF-Sampling and Up-Conversion

We choose the sampling rate f_s according to the expression

$$f_s = \frac{f_{\mathrm{IF}}}{\ell + 1/4} \tag{3}$$

for a positive integer ℓ. Then, we generate the discrete-time real signal

$$x'[n] = \mathrm{Re}\left\{x[n]\mathrm{e}^{\mathrm{j}2\pi(f_{\mathrm{IF}}/f_s)n}\right\} = \mathrm{Re}\left\{\mathrm{j}^{\pm n}x[n]\right\} \tag{4}$$

In this way, the periodic spectrum of $x'[n]$ has a replica centered at f_{IF}. After ideal D/A conversion, a pass-band filter centered at f_{IF} removes the other replicas, generating the desired IF modulated signal. The discrete-time modulation by $f_s/4$ and taking the real part of the modulated signal as expressed in (4) requires an almost negligible computational cost since the whole processing reduces to changing alternately the signs of $x[n]$. In order to avoid aliasing when taking the real, the condition $f_s \geq 2W$ must also be satisfied.

3.1.2. D/A Conversion

In the above description we assumed an ideal D/A converter with flat frequency response. Actual D/A converters exhibit a low-pass frequency response (approximately) of the form $\mathrm{sinc}(f / f_s)$ that does not extend to IF. A way to extend the D/A converter response so as to reduce the attenuation at IF consists of clocking the converter at rate $f_d = L_{\mathrm{D/A}} f_s$, where $L_{\mathrm{D/A}}$ is a positive integer such that $f_d >> f_{\mathrm{IF}}$, and up-sampling $x'[n]$ by the factor $L_{\mathrm{D/A}}$. In our setting we chose $L_{\mathrm{D/A}} = 8$. Notice that this approach has the undesired effect of reducing the average signal energy per sample by a factor $L_{\mathrm{D/A}}$. Hence for high $L_{\mathrm{D/A}}$ the IF analog signal can be very weak after D/A conversion, therefore it may need high amplification to be transmitted, giving rise to significant non-linear distortion. As an alternative, one may think of pre-compensating the linear response of the D/A converter by introducing a pass-band FIR filter between the up-sampler and the D/A converter. The filter must be designed in order to enhance the spectrum replica at IF while attenuating the other replicas. In this way the analog signal at IF would require lower amplification gains in the IF-RF conversion stage reducing the signal distortion due to the non-linearities of the power amplifiers. A low-complexity implementation of this technique is addressed in [8].

3.2. Receiver Front-End

3.2.1. IF-Sampling and Down-Conversion

Once the RF signal incoming from the antenna has been down-converted to IF, an A/D converter samples it at a rate f_s. Let $r_{\mathrm{IF}}(t)$ denote the received IF analog signal at the input of the A/D converter. By choosing $f_s \geq 2W$ according to (3), because of the periodicity of the discrete-time signal spectrum, the resulting real sampled signal $r[n] = r_{\mathrm{IF}}(n / f_s)$ is pass-band with a spectrum replica centered at $f_s / 4$. We shall remark that although f_{IF} and f_s at the receiver can be different from f_{IF} and f_s at the transmitter for simplicity we use the same notation. Notice that here in order to avoid signal resampling we suppose the rate f_s to be a multiple integer of the chip rate (i.e. $f_s = N_c f_c$ where in our implementation we set $N_c = 4$). A detailed discussion on resampling techniques for general SDR is provided in [8] and references therein. A base-band version of the received signal can be obtained by multiplying $r[n]$ by $(-\mathrm{j})^n$ followed by low pass filtering. We shall show that both channel estimation and the data detection process can also be performed in pass-band with the same complexity and avoiding explicit demodulation.

3.2.2 Slot-Timing Acquisition

For the acquisition of the slot-timing we superpose to the transmitted data a primary synchronization sequence (see [9]) at the beginning of every slot. The signal at the output of the A/D converter is filtered by a correlator matched to the primary synchronization sequence. The square magnitude of the filtered signal is averaged with an exponential window over several noise realizations (i.e., over several slots) and over the observation window. Then the slot-timing is detected as the time instant, with a resolution of one chip period, corresponding to the maximum of the averaged square output of the correlator. We remark that there is no need for higher resolution since the channel estimation algorithm compensates the residual synchronization errors.

3.2.3. Channel Estimation

Here we consider the training-sequence based multiuser channel estimation procedure for block-synchronous CDMA described in the UMTS/TDD standard proposal. In this scheme users are roughly synchronized to a common time-reference and transmit their training sequences at the same time (user timing errors are included as effect of the channel and taken automatically into account by the estimation procedure). The maximum allowed channel length (including possible timing errors) is Q chips and the training sequence sent by each user is built from a common base sequence of M chips, by adding a cyclic extension of Q chips. This solution allows a joint least-square (LS) estimation of all users' channels if $M \geq QU$, where U is the number of interfering users. In our set-up we assume $M = 192$ and $Q = 64$ (corresponding to a user's training sequence of length 192+64=256 chips) according to [9]. This technique was proposed and described in [10] and [11].
Due to the cyclic structure of the users' training sequences it suffices one DFT and one inverse DFT to have an estimate of all users' channels. Moreover this approach can be applied to both base-band and pass-band signals (see [8] and references therein for details).

3.2.4. Matched Filter Synthesis and Data Detection

Given the channel estimates, each user's symbol Matched Filter (MF) can be computed as the filter matched to the channel (including also the pulse shaping filter)-spreading sequence cascade associated with the user of in-

terest. The overall discrete-time channel-spreading sequence cascade is given by $f[k] = \sum_{i=0}^{N_c N-1} s[i]g[k-i]$ where $s[i]$ denotes the up-sampled version of the chip rate spreading sequence $s[n]$, defined as $s[i] = s[n]$ if $i = 4n, s[i] = 0$ otherwise and $g[i]$ denotes the channel estimate associated with the user of interest. The data symbols are detected by filtering the received signal with the MF, and sampling the filter output with the right timing at symbol rate. In this way the demodulation by $f_s / 4$ is automatically achieved and the symbol rate sequence is base-band. Hence the MF output at symbol rate can be written as

$$\hat{b}[k] = (-\mathrm{j})^{NN_c k} \sum_m r[m] f^*[m + NN_c k] = \sum_m r[m] f^*[m + NN_c k] \tag{5}$$

where the last equality holds in our setting with $N_c = 4$.

3.2.5. Carrier Synchronization and Decoding

The carrier synchronization is done at symbol rate with a classical decision directed algorithm [13]. The algorithm then takes a decision on the symbols and recovers the data flow (in our example a video stream).

4. Validation of the Existing Platform

The transmission and the reception of two user's real-time data streams (two H263 video streams) in an indoor environment have validated the platform described here. For this we use the following parameters:

- Chip rate $f_c = 3.6864$Mc/s and sampling rate $f_s = 4 \times 3.6864 = 14.7456$Ms/s
- Spreading factor $N = 16$
- Peak bit rate per user equal to 397.44 kbps
- Symmetric TDD slot arrangement (transmission occurs every 2 slots).
- Two synchronous users per slot
- IF frequency $f_{\mathrm{IF}} = 70$MHz
- RF band: 5MHz at 2.1GHz

5. End-To-End Processing DSP Performance

In this section we provide performance figures for the transmitter and receiver front-end algorithms previously described with the above set-up. The performances are evaluated in terms of DSP clock cycles. The algorithms have been implemented on a Texas Instuments-TMS320C6201 DSP (1600 Mips). The code is hand-optimized in parallel assembly [12]. The *transmitter front-end* processing includes the following operations:

Spreading and Scrambling. The spreading and scrambling routine takes as input the user information bit streams. Then it spreads, scrambles and eventually maps it to QPSK symbols. User's symbols are then amplified in order to assign the corresponding power and then summed up together. The routine also creates the slot structure filling each transmit slot with the midamble, the primary synchronization sequence and the user's symbols. This process takes for each slot a fixed amount of 7700 DSP cycles plus 6300 cycles per user corresponding to $(38.5 + 31.5 \times U)\ \mu$s.

Pulse-shaping, oversampling by 4 and modulation by $f_s / 4$. All these operations are implemented in a single assembler routine. The routine up-samples by a factor 4 the chip sequence previously generated, then passes it through a root-raised cosine filter $f[k]$ with roll-off factor equal to 0.22, truncated over a symmetric rectangular window of 12 chip periods, and modulates the output of the filter by $f_s / 4$, taking only the real part of the modulated signal. The oversampled version of the pulse-shaping filter is implemented as a polyphase filter bank with 4 phases. This processing requires 6 DSP cycles per output sample. Processing an entire slot of 2560 chips requires $6 \times 2560 \times 4 = 61440$ cycles to which one must add the cycles required by the prolog and the epilog needed for the routine pipeline [12]. Finally about 61500 cycles are needed to process a slot of 2560 chips corresponding to about 308 μs per slot.

The *receiver front-end* processing includes the following operations:

Primary synchronization code correlation. A routine is de-signed to compute the real correlation between the primary code and the samples (from the ACQ card) for the initial synchronization. This convolution is done at 4 times the chip rate and exploits the hierarchical properties of the primary synchronization code. The primary code only contains –1 and +1 and all users share the same code. Hence, the convolution is performed by using ADD and SUB instructions [12]. The routine also takes the square magnitude of the output. The routine loop kernel takes 33 cycles to produce an output sample at chip rate (about 422 μs per slot). To acquire the slot-timing we also perform both noise and temporal averaging. Noise averaging is done by accumulating several slots and by averaging with an exponential window with a properly chosen forgetting factor. Temporal averaging is per-

formed over a time interval equal to two slots by an exponential window with a forgetting factor. A single routine performs all these operations in 3 cycles per output sample at chip rate (38 μs per slot) and returns the slot-timing estimate.

Joint channel estimation. Up to 3 users' channels each one with duration of 64 chips, can be estimated with the current set-up. The training sequence (the midamble) period is 192 chips that, accounting for the oversampling factor of 4, corresponds to $192 \times 4 = 256 \times 3$ samples. Therefore a joint LS pass-band channel estimate can be obtained by one mixed radix FFT (radix 4 and radix 3) and one mixed radix inverse FFT (IFFT). The samples from the output of the ACQ card, corresponding to a real pass-band signal, are sent to the channel estimator that performs one real FFT. The LS channel estimate is produced in the FFT domain by multiplying the corresponding samples of the FFT of the received signal with the inverse of the FFT of the basic midamble period, which has been precomputed. A pass-band filtering is also performed to reject the image spectrum of the input real signal, by setting to zero the undesired samples. An IFFT produces a channel pass-band estimate. All the processing described above requires about 15600 cycles per slot (about 78 μs).

Channel analyzer. This routine analyzes the channel estimates, computing the channel energy, the channel length and the channel position that serve to the slot timing tracking operation. The routine also cleans the estimates from the round-off noise, clips the significant portion of each channel response. This process requires about 5.5 μs per slot.

Matched filter synthesis. The pass-band channel response of 64 chips is convoluted with each user's spreading sequence of 16 chips. This operation generates the user's symbol pass-band matched filter and can be performed in about 5000 cycles (25μs) per user per slot.

Pass-band matched filtering and data detection. Once both slot-timing and channel have been estimated, and the users' matched filters can be built. Then the pass-band signal at the output of the acquisition card is sent directly to each user's matched filter and the output is down-sampled at symbol rate (note that this operation automatically involves a demodulation to baseband avoiding the need of explicit demodulation). This processing requires 44000 cycles (220 μs) per user per slot.

6. Conclusion and Future Perspectives

In this paper we presented the major features of a real-time SDR platform implementing the essential physical layer of UMTS/TDD. This first prototype demonstrated the viability of SDR systems based on the DSP technology to provide universal seamless connection to wireless communication users. For the next platform upgrade we envision the implementation of a multiple antenna system, more sophisticated signal processing algorithms (e.g. multiuser detection and iterative decoding), and higher layers of the protocol stack (e.g. MAC layer). We also envision improving the design of the radio subsystem in terms of both flexibility and sensitivity. Along with these activities the platform will be opened to both academic and industrial partners to activate collaborations on specific research topics.

References

[1] "SDR Forum Current Technologies and Related Sites," http://www.mmitsforum.org/technology.html

[2] S. Srikanteswara, J. H. Reed, P. Athanas, and R. Boyle, "A soft radio architecture for reconfigurable platforms," *IEEE Communications Magazine*, February 2000.

[3] "Special issue on software radio," *IEEE JSAC*, vol. 4, April 1999.

[4] "Software radio," *IEEE Personal Communications*, vol. 4, August 1999.

[5] T. Hentschel, M. Henker, and G. Fettweis, "The digital front-end of software radio terminals," *IEEE Personal Comm.*, pp. 6–12, vol. 4, August 1999.

[6] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, pp. 26–38, May 1995.

[7] J. Razavilar, F. Rashid-Farrokhi, and K. J. R. Liu, "Software radio architecture with smart antennas: A tutorial on algorithms and complexity," *IEEE JSAC*, vol. 17, pp. 662– 676, April 1999.

[8] G. Caire, P. A. Humblet, G. Montalbano, and A. Nordio, "Transmission and reception front-end algorithms for software radio." submitted to IEEE JSAC Wireless Comm. Series, August 2000.

[9] 3GPP-TSG-RAN-WG1, "TS-25.2xx series," tech. rep., January 2000.

[10] B. Steiner and P. Jung, "Optimum and suboptimum channel estimation for the uplink of CDMA mobile radio systems with joint detection," *European Transactions on Comm.*, vol. 5, pp. 39–49, Jan.-Feb. 1994.

[11] G. Caire and U. Mitra, "Structured multiuser channel estimation for block-synchronous DS/CDMA." Submitted to IEEE Transactions on Communications., July 1999.

[12] Texas Instruments, *TMS320C62/C67x Programmer's Guide*, February 1998.

[13] J. G. Proakis, *Digital Communications*. NY: McGraw Hill, 2nd ed., 1989.