OPEN SOFTWARE RADIO PLATFORM FOR NEW GENERATIONS OF MOBILE COMMUNICATION SYSTEMS

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

Institut Eurécom, 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

A major concern of today's and near future mobile communication systems is represented by the need of providing universal seamless connection to the users. Software Defined Radio (SDR) systems lend themselves to handle several different standards and types of service, and this motivates the already large and still increasing interest in this research area. 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. 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. 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 functionality. In this paper we address the major issues related to the design of a real-time SDR system. We also describe the general architecture, the signal processing techniques adopted to implement the transmitter and receiver SDR front-end, the current platform set-up, and we provide DSP performance measurements. Finally, we consider 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 poses the problem of providing universal seamless connection to customers with different service requirements or simply needing to access different wireless networks supporting different standards. Software defined radio (SDR) terminals able to reconfigure themselves to handle several different standards and different services represent a solution to this problem (notice that SDR is a very broad term involving several levels in the protocol stack, see e.g. [1], [2], [3] and references therein). Also motivated by the intensive world wide activity around the third generation mobile communication systems, 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 platform is characterized by the following major features:

- Flexibility, achievable by a software driven system

- Duplex communication
- Multiple antennas transmit and receive signal processing

Flexibility remains a key word for a software defined system. In our case it serves on several purposes. For instance, the receiver can be properly configured (and calibrated) to perform propagation channel measurements and transmitter characterization, while the whole system can be programmed to implement different signal processing algorithms for both single user and multi-user systems under different operating conditions. *Duplex communication* is also necessary to allow higher layer protocol services, and to analyze more complex system aspects, such as multi-access and power control, and optimize down-link signal processing from up-link measurements. The platform will allow *multiple antennas signal processing*, or more generally, spatio-temporal signal processing (also known as array processing). This is a very promising ensemble of techniques to significantly increase the capacity of wireless communication systems due to the possibility of jointly exploiting both the spatial and the temporal dimension.

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 a 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 requiring any substantial new design for the essential hardware and software components. The BTS and the MT are based on a 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 is comprising of

- A reconfigurable 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 DSP's and workstations
- Data management software (data routing, framing, synchronization)
- Signal processing software (digital transceiver algorithms, multiple-access protocols, error coding/decoding)

These elements may be replicated in a parallel fashion to implement a multiple-antenna systems (both at the BTS and MT). Figure 1 shows the current hardware setup:
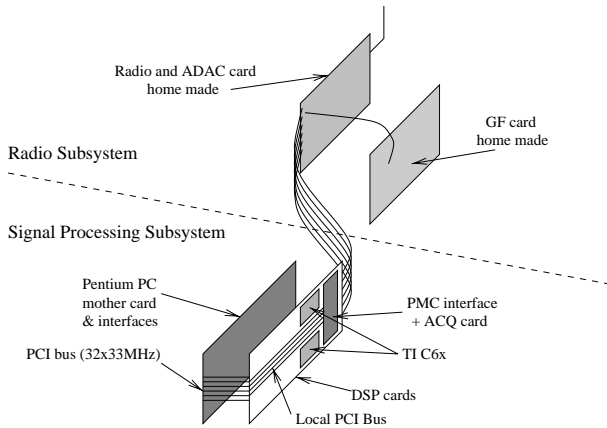

Figure 1: Mobile Terminal architecture

- A radio card capable to handle a 5 MHz bandwidth radio signals
- An A/D, D/A conversion card (ADAC)
- A data acquisition card (ACQ) 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 of 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 (ACQ) with 8 ADAC's.
- 8 DSP cards (the 2-processors DSP cards might be replaced by 4-processors DSP cards as shown in figure 2)

It is important to notice that the same architecture can be
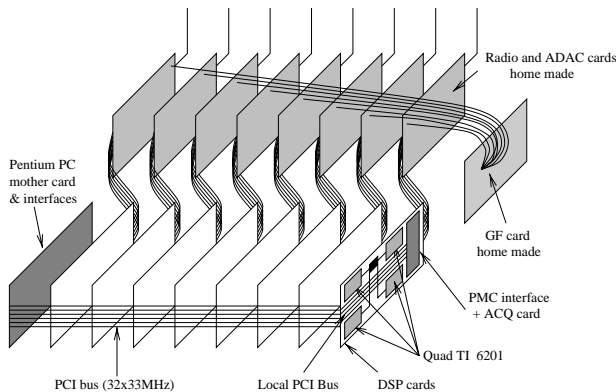

Figure 2: BTS structure

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 pre-detection 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 DSP's. Once the application and the corresponding operating mode are defined the dedicated software can be down-loaded and the platform completely reconfigured.

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 in 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

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

For Direct-Sequence CDMA system [4], using a spreading gain N, $a[k] = b[\lfloor k/N \rfloor]c[k]$, where $b[m]$ is the m-th modulation symbol (QPSK in the UMTS case) and $c[k]$ is the k-th chip and N is the spreading gain (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 baseband 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

$$y(t) = \mathrm{Re}\{x(t)\exp(\mathrm{j}2\pi f_{\mathrm{IF}}t)\} \qquad (2)$$

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 [4]. 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 bandpass filtering the output of a single D/A converter. This solution (for the receiver front-end) is described in [5] and [6]. However, since intermediate frequencies usually range between tens of MHz up to 100 MHz, 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. We may state that with the current DSP technology this approach is infeasible on a SDR system.

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), does not require explicit multiplication by the carrier signal, and only needs a single D/A converter and 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

$$f_s = \frac{f_{\mathrm{IF}}}{\ell \pm 1/4} \qquad \text{for a positive integer } \ell \qquad (3)$$

Then, we generate the discrete-time real signal

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

In this way, the periodic spectrum of $x'[n]$ has a replica centered at f_{IF} (see figure 3). 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 sampling rate must satisfy also the condition $f_s \geq 2W$.
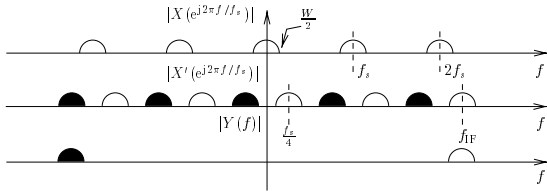

Figure 3: Spectrum of $x[n]$, $x'[n]$, and $y(t)$ with the integer $\ell = 2$ and the sign $(+)$ chosen in (3)

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 $f_d \gg f_{\mathrm{IF}}$, and up-sampling $x'[n]$ by the factor $L_{\mathrm{D/A}}$. In our setting we chose $L_d = 8$. 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 require to be strongly amplified to be transmitted giving rise to significant non-linear distortion. As an alternative, one may think of pre-compensating the linear distortion 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 is addressed in [7].

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, it is sampled by an A/D converter at a rate f_s (see figure 4). 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
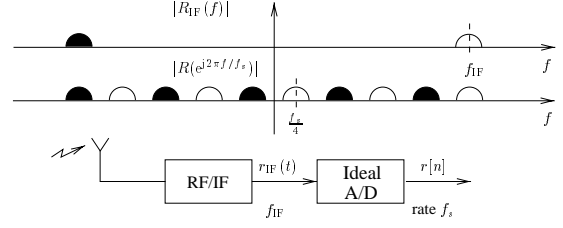

Figure 4: Receiver front-end

from f_{IF} and f_s at the transmitter, for the sake of simplicity we use the same notation. Notice that here in order to avoid signal re-sampling 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 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.

3.3. Slot-Timing Acquisition

For the acquisition of the slot-timing we superpose to the transmitted data a primary synchronization sequence (see [8]) 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 magnitude square 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 residual synchronization errors are accounted for and compensated by the channel estimation algorithm.

3.3.1. 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 $\mathbf{a} = [a[0], a[1], \ldots, a[M-1]]^T$ of length M chips, adding a cyclic extension of Q chips. This solution allows joint least-square 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 256 chips) according to [8]. This technique was proposed and described in [9] and [10]. Under these assumptions we can write the received signal sampled at frequency $f_s = N_c f_c = 4f_c$ during the M chips spanned by the base sequence as

$$\mathbf{w} = \bar{\mathbf{A}}\mathbf{g} + \boldsymbol{\nu}$$

where $\mathbf{w} = [w[0], w[1], \ldots, w[MN_c - 1]]^T$ is the received signal, $\mathbf{g} = [\mathbf{g}_1^T, \ldots, \mathbf{g}_u^T, \ldots, \mathbf{g}_U^T]^T$ is a vector containing the channel impulse responses of the U users, $\mathbf{g}_u = [g_u[0], g_u[1], \ldots, g_u[QN_c - 1]]^T$ is the u-th user's channel filter vector and $\boldsymbol{\nu}$ is a vector of interference plus noise samples,

assumed white. The $MN_c \times MN_c$ matrix $\bar{\mathbf{A}}$ is defined as $\bar{\mathbf{A}} = \mathbf{A} \otimes \mathbf{I}_{N_c}$ where $(\otimes)$ denotes the Kronecker product and $\mathbf{A}$ is a circulant matrix containing all the possible cyclic shifts (by columns) of the base sequence $\mathbf{a}$. The matrix $\bar{\mathbf{A}}$ is also circulant and it is unitary similar [11] to the diagonal matrix $\mathrm{diag}(\bar{\boldsymbol{\alpha}})$, where

$$\bar{\boldsymbol{\alpha}} = \underbrace{[\boldsymbol{\alpha}^T, \ldots, \boldsymbol{\alpha}^T]^T}_{N_c \text{ times}}$$

and where $\boldsymbol{\alpha}$ is the discrete Fourier transform of $\mathbf{a}$. After some algebra, it is possible to show that the Least Squares estimation of the overall channel impulse response g is given by

$$\hat{g} \quad = \quad \mathrm{IDFT}\left\{\frac{\mathrm{DFT}\,\{w\}}{\bar{\boldsymbol{\alpha}}}\right\} \tag{5}$$

where DFT and IDFT denote direct and inverse discrete Fourier transforms and the ratio of two vectors should be interpreted as the element-by-element division.

This approach can be applied to both base-band and pass-band signals. In our case since the received signal is real and pass-band, its spectrum shows both the left and right-side replicas. As it is shown in figure 5 for $N_c = 4$, the left-side replica occupies the first and the second quarter of the DFT spectrum, while the right-side replica occupies the third and the fourth one.

The receiver can also use the a priori information that the signal bandwidth is limited to W. Then it can limit the computation to the range $[f_s/4 - W/2, f_s/4 + W/2]$, setting to 0 the rest of the channel estimates spectrum. Notice that this operation in the frequency domain corresponds to a low-pass filtering in the time-domain, moreover it reduces the computational cost since only a part of the MN_c products (by the element-wise inverses of $\bar{\boldsymbol{\alpha}}$ in (5)) needs to be computed. Finally the IDFT produces the estimated pass-band channels complex responses.
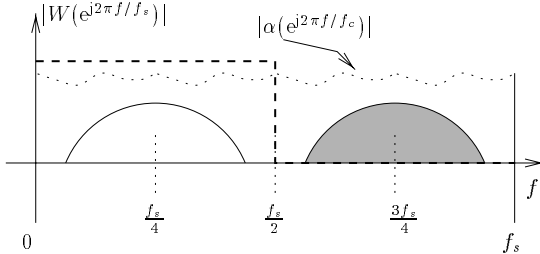

Figure 5: Channel estimation: pass-band

3.3.2. Matched Filter Synthesis and Data Detection

Given the channel estimates, we are then interested in synthesizing a Matched Filter (MF) matched to the channel (including also the pulse shaping filter)-spreading sequence cascade associated with the user of interest. The overall discrete-time matched channel-spreading sequence cascade is given by

$$f[k] = \sum_{i=0}^{N_c N - 1} s[i]g[k - i] \tag{6}$$

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

$$
\begin{aligned}
\hat{b}[k] &= (-\mathrm{j})^{NN_c k} \sum_m r[m]f^*[m + NN_c k] \\
&= \sum_m r[m]f^*[m + NN_c k]
\end{aligned} \tag{7}
$$

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

3.3.3. Carrier Synchronization and Decoding

The carrier synchronization is done at symbol rate with a classical decision directed algorithm [4]. 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 platform described in this paper has been validated by the transmission and the reception of of two user's real-time flows in an indoor environment. Two H263 video streams are transmitted in parallel and decoded in real time. For this we use the following parameters:

- 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
- RF band: 5 MHz at 2.1 GHz

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 performance are evaluated in terms of DSP clock cycles. The algorithms have been implemented on a TI-TMS320C6201 DSP. 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 TX 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 with roll-off factor equal to 0.22, truncated over a symmetric 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 $308 \, \mu s$ per slot.

The receiver front-end processing includes the following operations:

Primary synchronization code correlation. A routine is designed 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 $422\ \mu s$ 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 performed 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 ($38\ \mu s$ per slot) and returns the slot-timing estimate.

Joint channel estimation. Up to 3 users' channels each one with a 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 least-square (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 $78\ \mu s$).

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 $5.5\ \mu s$ per slot.

Matched filter synthesis The pass-band channel response of 64 chips is convolved 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 ($25\ \mu s$) 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 built, the pass-band signal at the output of the DAQ 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 ($220\ \mu s$) 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 sys-

Operation	μs per slot
Spreading and Scrambling	133
Pulse-shaping, oversampling and modulation	308
Total	441

Table 1: Transmitter: time required by DSP routines for 3 users

Operation	μs per slot
Joint channel estimation	78.0
Channel analyzer	16.5
Matched filter synthesis	25.0
Matched filtering and data detection	660.0
Total	779.5

Table 2: Receiver: time required by DSP routines for 3 users

tems based upon 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. multi user detection and iterative decoding), and higher layer protocol stacks (e.g. MAC layer). We also envision to improve 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.

7. REFERENCES

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

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

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

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

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

[6] 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.

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

[8] 3GPP-TSG-RAN-WG1, "TS-25.2xx series," tech. rep., January 2000.

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

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

[11] G. H. Golub and C. F. V. Loan, *Matrix Computation*. The John Hopkins University Press, 1996.

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