A GPP-BASED SOFTWARE-DEFINED RADIO FRONT-END FOR WLAN STANDARDS

[1]*Roel Schiphorst,* [1]*Fokke Hoeksema,* [2]*Vincent Arkesteijn,* [1]*Kees Slump,* [2]*Eric Klumperink and* [2]*Bram Nauta*

r.schiphorst@utwente.nl
[1] University of Twente, EEMCS department, Signals and Systems group,
P.O. Box 217, 7500 AE Enschede, The Netherlands
[2] University of Twente, EEMCS department, Integrated Circuit Design Group,
P.O. Box 217, 7500 AE Enschede, The Netherlands
http://www.sas.el.utwente.nl/home/SDR/

ABSTRACT

This paper presents a software-defined radio testbed for the physical layer of wireless LAN standards. All baseband physical layer functions have been successfully mapped on a Pentium 4 processor that performs these functions in real-time. This has been tested in combination with a CMOS integrated wideband analog front-end containing a low noise amplifier, downconversion mixers and filters.

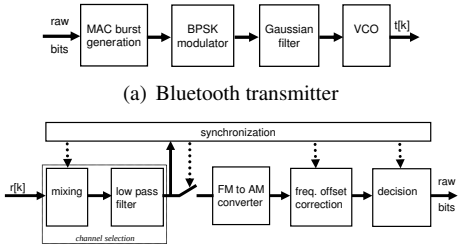
The testbed consists of both a transmitter and a receiver. The transmitter contains a transmitter PC with a DAC board, an Agilent E4438C generator for upconversion and an antenna. The receiver consists of an antenna, a wideband SDR analog front-end and a receiver PC with an ADC board.

On this testbed we have implemented two different types of standards, a continuous-phase-modulation based standard, Bluetooth and an OFDM based standard, HiperLAN/2. However, our testbed can easily be extended to other standards, because the only limitations in our testbed are the maximal channel bandwidth of 20 MHz, the dynamic range of the wideband SDR analog front-end and of course the processing capabilities of the used PC.

1. INTRODUCTION

New wireless communications standards do not replace old ones, instead the number of standards keeps on increasing and by now an abundance of standards already exists. Moreover there is no reason to assume that this trend will ever stop. Therefore the software-radio concept is emerging as a potential pragmatic solution: a software implementation of the user terminal able to dynamically adapt to the radio environment in which the terminal is located [1].

Because of the analog nature of the air interface, a software radio will always have an analog front end. In an ideal software radio, the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) are positioned directly after the antenna. Such an implementation is not feasible due to the power that such device would consume and other physical limitations [2]. It is therefore a challenge to design a system that preserves most properties of the ideal software radio while being realizable with current-day technology. Such a system is called a software-defined radio (SDR).



(a) Bluetooth transmitter

(b) Bluetooth receiver

Figure 1: Functional architecture of the Bluetooth transmitter and receiver

2. SDR PROJECT

In our SDR project we research whether the lowest layer, the physical layer of wireless standards can be implemented in software running on a General Purpose Processor (GPP) and estimate the costs of such an implementation with respect to power consumption and computational power requirements. So, we interpret SDR as an implementation technology, invisible for consumers.

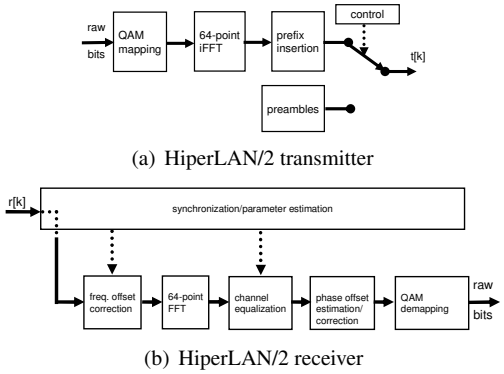
In our SDR project we decided not to focus on an all-standard radio but to start with a software-defined radio for wireless LAN standards first. The research is carried out by two chairs of the University of Twente: the IC-Design group which focusses on the RF part and the Signals and Systems group focussing on the baseband part. At the project's start we defined also the scope of project: the physical layer excluding error correction encoding / decoding. Our design goal is a notebook with a wideband RF frontend and a software implementation of the physical layer.

Wireless LAN standards use phase modulation or OFDM in the 2.4 GHz or 5 GHz frequency band, so we decided in our project to combine an instance of a continuous-phase modulation standard (Bluetooth [3]) with an OFDM standard (HiperLAN/2 [4]).
The rest of the paper is organized as follows: First the functional architecture of the physical layer of both standards is discussed, which is followed by a description of the testbed. This paper concludes by presenting real-world measurements done with the testbed.

3. TESTBED

3.1. Functional architecture

Figure 1 depicts the functional architecture of the Bluetooth transmitter and receiver. The first step in the transmitter is to embed the raw bits into MAC bursts which are then BPSK modulated at 1 Mbit/s. The BPSK symbols are filtered by a Gaussian low pass filter and the filtered output is connected to an VCO that translates the amplitude variation into frequency variations. At the receiver side, the first step is to select the wanted Bluetooth channel and suppressing all others which is performed both digitally and by the analog front-end. This is achieved by mixing the wanted channel to zero IF and applying a low-pass filter. The next step is to demodulate the FM signal into an AM signal by taking the derivative of the phase. Because a frequency offset introduces an offset in the AM signal, it has to be corrected before the bit decision.
Figure 2 depicts the HiperLAN/2 physical layer architecture which is very different from the Bluetooth architecture one. The HiperLAN/2 transmitter starts with mapping raw bits on QAM symbols (either BPSK, QPSK, 16-QAM or 64-QAM symbols). In the next step, the QAM symbols are mapped on data carriers and an OFDM symbol is constructed by adding pilot carriers, applying an inverse FFT and adding an prefix, which results in a 20 MSPS signal. MAC bursts are then created by adding special symbols, preambles, to the start of the MAC burst.
The HiperLAN/2 receiver starts by searching for the start of a MAC burst. If found, it estimates the frequency offset and channel parameters. After these steps the data OFDM symbols can be demodulated by first correcting the frequency offset, performing an FFT, correcting the channel and detecting and correcting the phase offset by using the pilot tones. The output are QAM symbols which have to be de-mapped into raw bits.
Although the functional architecture of both standards is very different, we have successfully integrated the functional architecture of the Bluetooth receiver into the HiperLAN/2 receiver [5] by using a (simplified) maximum a posteriori probability (MAP) receiver which is a more advanced Bluetooth demodulation algorithm [6]. In this testbed,



(a) HiperLAN/2 transmitter

(b) HiperLAN/2 receiver

Figure 2: Functional architecture of the HiperLAN/2 transmitter and receiver

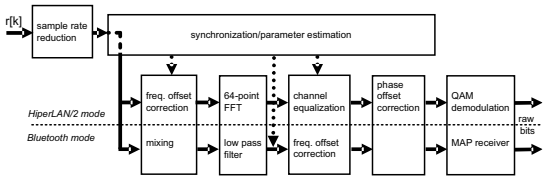


Figure 3: Functional architecture of the Bluetooth-enabled HiperLAN/2 receiver

however, we did not implement this receiver (yet) but used instead a conventional receiver such as depicted in Figure 1.

3.2. Testbed setup

Figure 4 shows the component architecture of our SDR testbed. The testbed consists of eight components; a transmitter PC, a DAC PC board, an Agilent E4438C generator, two antennas, a wideband SDR analog front-end, a receiver PC and an ADC PC board.
The transmitter PC continuously generates HiperLAN/2 or Bluetooth MAC bursts which are sent in real-time[1] to the DAC board at 20 MSPS by using an Adlink cPCI-7300 digital I/O PCI card. This DAC board converts the digital signal into a complex analog baseband signal which is upconverted to radio frequencies by the Agilent E4438C generator. This RF signal is transmitted through the air and received by the receiver antenna. The antenna output is connected to the wideband SDR analog front-end that converts the RF signal into a baseband signal. The ADC board samples this analog signal with 80 MSPS and the onboard Intersil ISL5416 programmable down converter (SRC) decimates the digital signal into a complex 20 MSPS signal in HiperLAN/2 mode and into a 5 MSPS

[1] We use a non-real time operating system, e.g. linux and therefore the transmitter and receiver programs use large buffers of 100 ms to avoid the influence of the operating system. Additional research is needed to find the maximal allowable latency of the system.

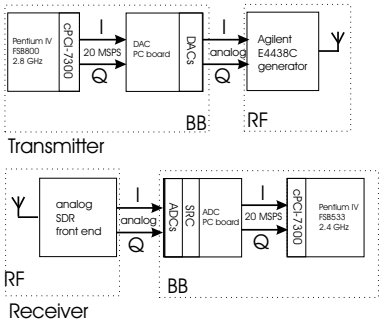


Figure 4: Component architecture of the SDR testbed

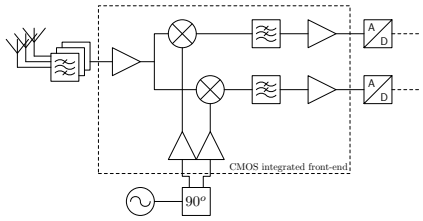


Figure 5: Analog front-end

signal (including mixing of the wanted channel to baseband) for Bluetooth. This signal is transported to the receiver PC by using another Adlink cPCI-7300 digital I/O PCI card. The receiver PC performs all demodulation functions and demodulates the MAC bursts real-time.
The wideband SDR analog front-end and the digital parts are discussed in more detail below.

4. WIDEBAND SDR ANALOG FRONT-END

This section discusses the analog part of the SDR front-end. A block schematic can be seen in Figure 5.
As discussed in [7], the front-end uses separate antennas and RF filters for the various bands. The rest of the receiver however, which should have a large bandwidth, is integrated. This way, the integrated circuit can still be used for a large number of applications, and only a new PCB with filters and possibly the antenna has to be designed when a receiver for a new frequency band is required.
The rest of the discussion in this section will focus on the part in Figure 5 inside the dashed rectangle. For this part, an integrated circuit has been designed using a standard 0.18μm CMOS process.

The circuit starts with a wideband low noise amplifier (LNA). This LNA employs noise cancelling [8], which is useful for obtaining a low noise figure over a wide bandwidth.
The LNA is followed by two mixers. Two local oscillator (LO) signals with a phase difference of 90 degrees are present. These implement quadrature downconversion, enabling both zero-IF and low-IF architectures.
The two mixers are both followed by a baseband amplifier combined with a low-pass filter with a cut-off frequency of around 10 MHz. This is more than enough for Bluetooth and sufficient for HiperLAN/2 signals.
The circuit's correct functioning has been verified, both stand-alone and in combination with the digital front-end. More details on the analog front-end can be found in [9].

5. DIGITAL FRONT-END

The source code of the Bluetooth and HiperLAN/2 transmitter and receiver is written in C and compiled with the Intel compiler 7.1 under Linux, using floating-point precision. As a DAC requires fixed-point numbers, the transmitter has to convert the floating-point numbers into fixed point. The receiver, on the other hand, receives fixed-point numbers from the ADCs, so it has to do the inverse process.

6. MEASUREMENTS IN THE TESTBED

6.1. User scenarios

For both standards, Bluetooth and HiperLAN/2, we derived a user scenario to estimate and measure the computational requirements, assuming continuous transmission. This scenario can be compared with a realistic scenario that includes the influences of the higher OSI layers on the physical layer.

6.1.1. Bluetooth user scenario

The Bluetooth symbol duration is 1 μs and data is transmitted in time slots with a duration of 625 μs [3]. For estimating computational requirements, we assume maximal transfer rate. In this mode, Bluetooth uses a packet which spans 5 time slots and 1 time slot is used for uplink communication.

6.1.2. HiperLAN/2 user scenario

A HiperLAN/2 MAC frame consists of 5 parts and has a maximal duration of 2 ms [4]. We assume that all parts have equal duration and that we have to demodulate 2 parts (one common and one user part).

6.2. Computational power requirements

We used the user scenarios of both standards for the implementation of the transmitter and receiver. This section presents the required computational power for each function that is mapped on the Pentium 4 processor and the

function	M operations/s [analyzed]	M cycles/s [measured]
Bluetooth		
TX function	2181	714
RX function	100	437
HiperLAN/2		
TX function	348	500
RX function	509	1225

Table 1: Computational load of the Bluetooth and HiperLAN/2 transceiver functions

number of cycles needed by the CPU for computing the function.

6.2.1. Time measurement method

Time measurements were performed on a Pentium 4 at 2.8 GHz by counting the number of cycles for each function. We used average values in the time measurements as a Pentium 4 is a very complex design.

6.2.2. Results

Table 1 shows the results for both transceivers. (More information can be found in [5].) The HiperLAN/2 receiver function is most demanding and has a load of $\approx 44\%$ on a Pentium 4 at 2.8 GHz whereas the Bluetooth receiver requires only $\approx 16\%$ on the same processor.
In the Bluetooth transmitter, the GFSK modulation function uses a 60-tap Gaussian filter which requires 1000 additions plus multiplications per second. In our implementation we replaced this filter by lookup tables as the output value of the filter depends on the last 4 BPSK symbols. This optimization reduces the amount of computations significantly.

6.2.3. Experiments

RF experiments have been performed with the setup of Figure 4 (without antennas) and a picture of the testbed is shown in Figure 6. In both Bluetooth and HiperLAN/2 mode, successful transmission and reception of continuously transmitted MAC bursts is achieved. In addition, some initial tests have been performed successfully with antennas in Bluetooth mode. More experiments will be conducted in the near future but we focus on verifying the built hardware first.

7. CONCLUSIONS

This paper describes a software-defined radio testbed for wireless LAN standards. The physical layer of both standards, Bluetooth and HiperLAN/2, has been implemented in software running real-time on a normal PC and experiments have verified the system. Our testbed can easily

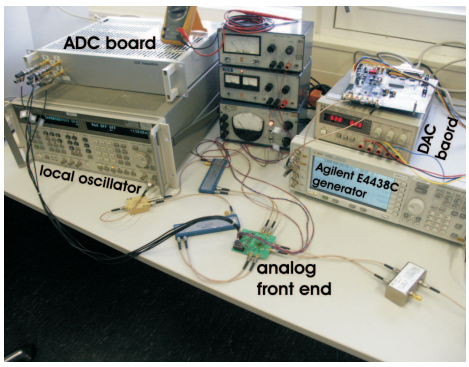


Figure 6: Experiment setup (both computers are not shown)

be extended to other standards, because the only limitations in our testbed are the maximal channel bandwidth of 20 MHz, the dynamic range of the wideband SDR analog front-end and the processing capabilities of the used PC.

8. REFERENCES

[1] J. Mitola, *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, Wiley, 2000.

[2] R.H. Walden, "Performance trends for analog-to-digital converters," *IEEE Communications Magazine*, pp. 96–101, February 1999.

[3] BluetoothSIG, "Specification of the bluetooth system - core," Technical Specification Version 1.1, 'Bluetooth SIG', February 2001.

[4] ETSI, "Broadband radio access networks (bran); hiperlan type 2; physical (phy) layer," Technical Specification ETSI TS 101 475 V1.2.2 (2001-02), ETSI, February 2001.

[5] R. Schiphorst, F.W. Hoeksema, and C.H. Slump, "Implementation alternatives for a flexible wireless lan transceiver," *Workshop on Software Radios (WSR2004), Karlsruhe (G)*, March 2004.

[6] R. Schiphorst, F.W Hoeksema, and C.H. Slump, "A (simplified) bluetooth maximum a posteriori probability (map) receiver," *Proceedings of IEEE SPAWC2003*, June 2003.

[7] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump, "A software-defined radio test-bed for wlan front ends," 3^{nd} *PROGRESS workshop on Embedded Systems and Software*, 2002.

[8] F. Bruccoleri, E.A.M. Klumperink, and B. Nauta, "Wideband CMOS low-noise amplifier exploiting thermal noise canceling," vol. 39, no. 2, pp. 275–282, Feb. 2004.

[9] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump, "A combined receiver front-end for bluetooth and hiperlan/2," 4^{nd} *PROGRESS workshop on Embedded Systems and Software*, 2003.