EURASIP Journal on Applied Signal Processing 2005:16, 2664–2672

A Real-Time GPP Software-Defined Radio Testbed for the Physical Layer of Wireless Standards

R. Schiphorst
The Signals and Systems Group, Department of Electrical Engineering, Faculty of EEMCS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
Email: r.schiphorst@utwente.nl

F. W. Hoeksema
The Signals and Systems Group, Department of Electrical Engineering, Faculty of EEMCS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
Email: f.w.hoeksema@utwente.nl

C. H. Slump
The Signals and Systems Group, Department of Electrical Engineering, Faculty of EEMCS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
Email: c.h.slump@utwente.nl

Received 30 January 2004; Revised 23 September 2004

We present our contribution to the general-purpose-processor-(GPP)-based radio. We describe a baseband software-defined radio testbed for the physical layer of wireless LAN standards. All physical layer functions have been successfully mapped on a Pentium 4 processor that performs these functions in real time. The testbed consists of a transmitter PC with a DAC board and a receiver PC with an ADC board. In our project, we have implemented two different types of standards on this testbed, 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 limitation in our testbed is the maximal channel bandwidth of 20 MHz and of course the processing capabilities of the used PC. The transmitter functions require at most 714 M cycles per second and the receiver functions need 1225 M cycles per second on a Pentium 4 processor. In addition, baseband experiments have been carried out successfully.

Keywords and phrases: software-defined radio, testbed, baseband, physical layer, HiperLAN/2, Bluetooth.

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; see Table 1. 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 this device would consume and other physical limitations [2, 3]. It is therefore a challenge to design a system that preserves most of the properties of the ideal software radio while being realizable with current-day technology. Such a system is called a software-defined radio (SDR).

Software-defined radio has both advantages for consumers and manufactures because current products support only a fixed number of standards. Figure 1 shows the lifetime of products and wireless standards. One can see that products support a fixed number of standards and in time new standards emerge and old ones disappear, making a product eventually obsolete.

Software-defined radios on the other hand will enable consumers to upgrade their radio with new functionality, for example, required by new standards, just by software updates, without the need for new hardware. Moreover, manufacturers can upgrade or improve functionality of consumer-owned products and SDR could result in shorter development time, cheaper production due to higher volumes.

Table 1: Overview of wireless standards [16].

Standard	Frequency band	Modulation type
CT2	864/868 MHz	GFSK
CT2+	944/948 MHz	GFSK
DECT	1880–1900 MHz	GFSK
PHS	1893–1920 MHz	DQPSK
IEEE 802.15.4	2402–2480 MHZ (North America) 2412–2472 MHz (Europe) 2483 MHz (Japan)	GFSK
Bluetooth	2402–2480 MHz (North America and Europe) 2447–2473 MHz (Spain) 2448–2482 MHz (France) 2473–2495 MHz (Japan)	GFSK
HomeRF	2402–2480 MHz (North America and Europe) 2447–2473 MHz (Spain) 2448–2482 MHz (France) 2473–2495 MHz (Japan)	GFSK
IEEE 802.11a	5150–5250 MHz (USA) 5250–5350 MHz (USA) 5725–5825 MHz (USA)	OFDM: 2/4/16/64 QAM
IEEE 802.11b	2410–2462 MHz (North America) 2412–2472 MHz (Europe) 2483 MHz (Japan)	GFSK/DBPSK/DQPSK/QPSK
HiperLAN/2	5150–5250 MHz (USA) 5250–5350 MHz (USA) 5725–5825 MHz (USA) 5150–5350 MHz (Europe) 5470–5725 MHz (Europe) 5725–5875 MHz (Europe) 5150–5250 MHz (Japan)	OFDM: 2/4/16/64 QAM
IS-54/IS-136	824–849 MHz 869–894 MHz 1850–1910 MHz 1930–1990 MHz	CDMA: $\pi/4$ DQPSK
IS-95	824–849 MHz 869–894 MHz 1850–1910 MHz 1930–1990 MHz 1920–1980 MHz (Asia only) 2110–2170 MHz (Asia only)	CDMA: QPSK/OQPSK
IMT-2000/UMTS	1920–1980 MHz 2110–2170 MHz 1900–1920 MHz 2010–2025 MHz	CDMA: QPSK

Table 1: Continued.

Standard	Frequency band	Modulation type
GSM	824–849 MHz	GMSK
	869–894 MHz	
	880–915 MHz	
	925–960 MHz	
	1710–1785 MHz	
	1805–1880 MHz	
	1850–1910 MHz	
	1930–1990 MHz	
PDC	810–826 MHz	$\pi/4$ DQPSK
	940–956 MHz	
	1429–1441 MHz	
	1453–1465 MHz	
	1477–1489 MHz	
	1501–1513 MHz	

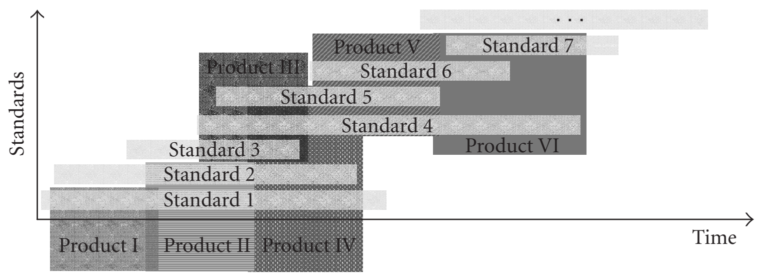

Figure 1: Standards support by products in time.

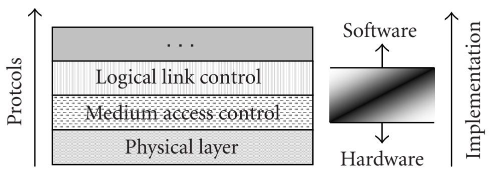

Figure 2: Mapping of protocols in current designs on hardware/ software.

However the downside of SDR will be power consumption as dedicated designs are more power efficient which is very important for mobile applications.

2. SOFTWARE-DEFINED RADIO

Figure 2 depicts the mapping in current radio designs of the different OSI[1] layers on software/hardware. The physical layer is generally implemented in hardware and higher layers are often software based with the logical link control (LLC) and medium access control (MAC) layer as a transition area. In our SDR project [4], we research whether the lowest layer, the physical layer, of wireless standards can be implemented in software running on a general-purpose processor 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 which differs from the views in [1, 5], that is, flexible, universal, radio systems at each layer of the OSI model from which manufacturers, network operators, and consumers can benefit. Our interpretation of SDR is more focussed on the physical layer, an implementation technology, invisible for consumers. Moreover we want to investigate if we can use existing processing capabilities (e.g., a notebook's CPU) for digital signal processing purposes, thereby possibly prolonging the lifetime of a device. This saves hardware and Moore's law will lower in time the computational load as a percentage of the computational capacity.

A flexible, all-standard, radio will always consume more energy than a dedicated radio; thus the first application for a flexible radio; will be an application where power consumption is less an issue, an example being a flexible radio in a

[1]Open system interconnection protocol model (OSI).

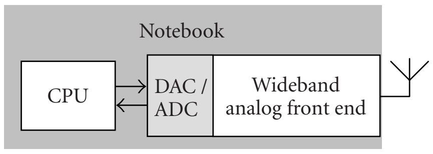

Figure 3: Project scope.

notebook. This application for SDR has three advantages. First, we can use the processing capabilities of the general-purpose processor for digital signal processing purposes. Second, in comparison to SDR for mobile phones, our demonstrator can consume much more power (in the order of 1 W). Third, a notebook is very suited for demonstration purposes.

Table 1 gives an overview of important wireless standards together with the used frequency bands and modulation type. It seems that each standard can be seen as a family of standards, an example being GSM. Thus the number of existing standards that manufacturers have to support is even larger than one would initially expect. However, there are also similarities among them; the used frequency bands are between 0.8 and 6 GHz with dominant frequency bands around the 0.8 GHz, 2 GHz, 2.4 GHz, and 5 GHz. In addition, three types of transceivers are used, phase modulation, OFDM (orthogonal frequency-division multiplexing), and CDMA (code-division multiple access) transceivers.

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 also defined the scope of the project: the physical layer excluding error-correction encoding/decoding. Recent literature [6] indicates, however, that especially error-correction decoding (Viterbi algorithm) requires most of the computational power in the lower layers of a system. Figure 3 summarizes the design goal of our project, a notebook with a wideband RF front end with 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 phase-modulation standard (Bluetooth) with an OFDM standard (HiperLAN/2). Table 2 shows some characteristics of the physical layer of both standards. HiperLAN/2 is a high-speed wireless LAN (WLAN) standard using OFDM. Its physical layer is very similar to the 802.11a standard. Bluetooth on the other hand is a low-cost, low-speed standard, designed for replacing fixed cables. Bluetooth uses continuous-phase modulation, Gaussian frequency shift keying (GFSK) which is also used by other standards such as HomeRF and DECT.

This paper discusses only the digital baseband part of the project. More information about the total project can be found in [7] or at the project's website [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. SDR BASEBAND TESTBED

In the first phase of the project, we built two separate receivers [8] in order to gain knowledge. After the first phase, we concluded that a real-time software implementation of the physical layer functions of the transmitter and receiver on a Pentium 4 processor was possible. Therefore we started in the second phase of the project with a real-time software implementation of the Bluetooth and HiperLAN/2 receiver and transmitter.

Although we show that a real-time software implementation of the receiver (and transmitter) functionality is possible using the notebook's processor, it requires, besides processing power, a real-time operating system. *Traditional* operating systems such as Windows or Linux are non-real time; for example, the latency of the system is undefined and can be up to 100 milliseconds for Linux [9]. So it is possible that our receiver *program* misses a buffer and data is lost. However, special patches can be applied to the Linux kernel for example, which reduces this maximal latency to about 5 microseconds [9].[2] In our testbed, we use large sample buffers of 100 milliseconds to avoid the influence of the operating system but additional research is needed to find the maximal allowable latency which is probably determined by the MAC layer. Furthermore, we have to investigate if this value can be achieved in our testbed. So at the moment, our testbed can only be used for continuous transmission of MAC bursts.

3.1. Functional architecture

Figure 4 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 Mbp. The BPSK symbols are filtered by a Gaussian lowpass filter and the filtered output is connected to VCO that translates the amplitude variation into frequency variations. At the receiver side, the first step is to select the wanted Bluetooth channel and suppress all others which is performed both digitally and by the analog front end. This is achieved by mixing the wanted channel to baseband and applying a lowpass 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 bit decision.

On the other hand, Figure 5 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 BPSK, QPSK, 16-QAM, or 64-QAM symbols, depending on the used mode. 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 a prefix, which results in

[2]A HiperLAN/2 MAC frame has a Duration of 2 milliseconds and the shortest Bluetooth MAC frame is 0.625-millisecond long.

Table 2: Some physical layer characteristics of Bluetooth and HiperLAN/2.

Parameter	Bluetooth	HiperLAN/2
Band	2.4–2.48 GHz	5.15–5.725 GHz
Channel spacing	1 MHz	20 MHz
Modulation	GFSK	OFDM: BPSK/QPSK/16 QAM/64 QAM
Nominal bit rate (no FEC)	172.8–723.2 kbps	12–72 Mbps

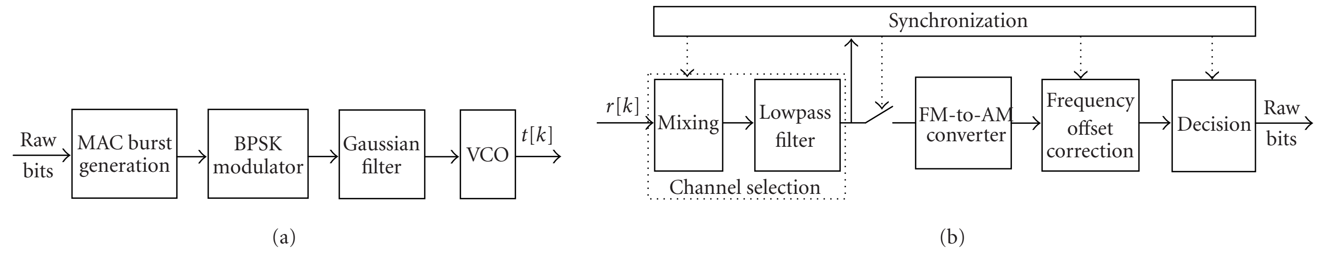

Figure 4: Functional architecture of the Bluetooth (a) transmitter and (b) receiver (functional layer excluding error-correction encoding/decoding).

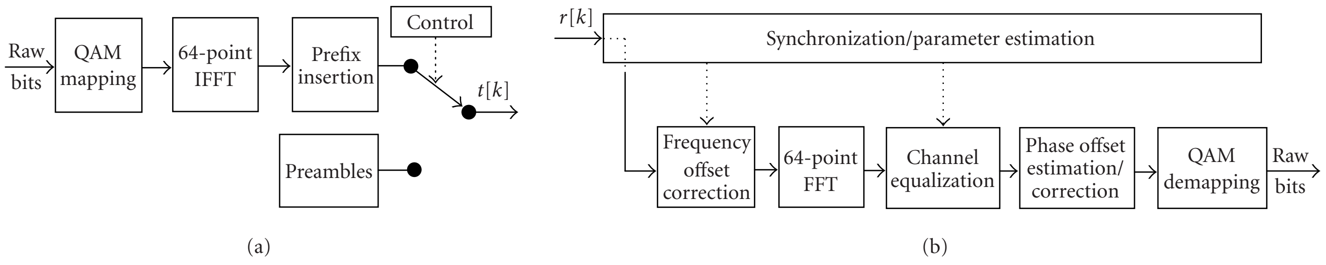

Figure 5: Functional architecture of the HiperLAN/2 (a) transmitter and (b) receiver (functional layer excluding error-correction encoding/decoding).

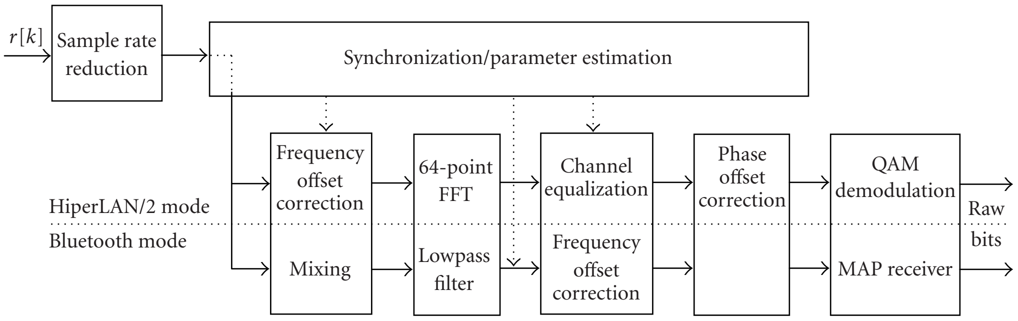

Figure 6: Functional architecture of the Bluetooth-enabled HiperLAN/2 receiver (functional layer excluding error-correction encoding/decoding).

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 an MAC burst. If it is 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 is QAM symbols which have to be demapped into raw bits.

Although the functional architecture of both standards is very different, we have successfully integrated the Bluetooth receiver functionality into the HiperLAN/2 receiver [10] (Figure 6) by using a (simplified) maximum a posteriori

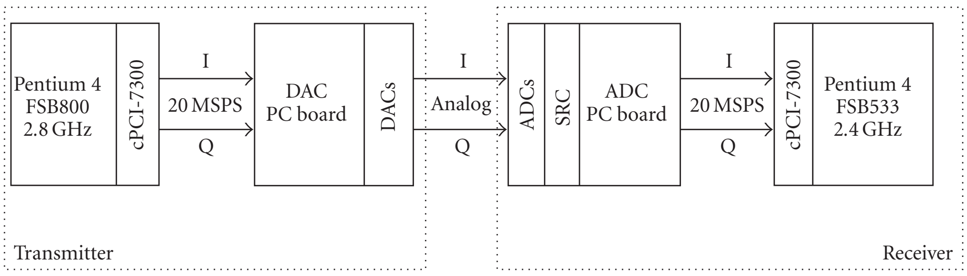

Figure 7: Component architecture of the SDR testbed.

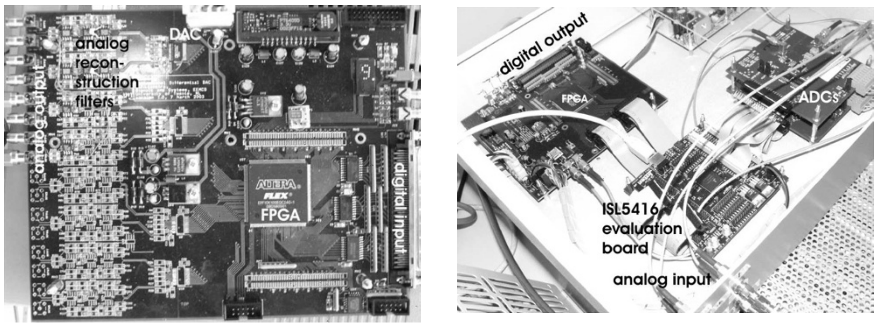

Figure 8: (a) DAC board and (b) ADC board.

probability (MAP) receiver which is a more advanced Bluetooth demodulation algorithm [11]. In this testbed, however, we did not implement this receiver (yet) but used instead a conventional receiver such as the one depicted in Figure 4.

3.2. Testbed setup

Figure 7 shows the component architecture of our SDR testbed. The testbed consists of four components; a transmitter PC, a DAC PC board (Figure 8a), a receiver PC, and an ADC PC board (Figure 8b).

The transmitter PC continuously generates HiperLAN/2 or Bluetooth MAC bursts which are sent in real time 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. The ADC board samples the analog signal with 80 MSPS and the onboard Intersil ISL5416 programmable down-converter decimates the digital signal into a complex 20-MSPS signal in HiperLAN/2 mode and into a 5-MSPS signal (including mixing 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 in real time.

At this moment in time, the analog signal of the DAC board is directly connected to the input of the ADC board but in the near future we will conduct RF experiments, in which the analog transmitter signal is upconverted to the 2.4 or 5 GHz frequency band. The RF signal will then be connected to the analog SDR front end [7] whose output signal is fed to the ADC board.

4. MEASUREMENTS IN THE TESTBED

4.1. 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.

4.1.1. Bluetooth user scenario

The Bluetooth symbol duration is 1 microsecond and data is transmitted in time slots with a duration of 625 microseconds [12]. 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.

4.1.2. HiperLAN/2 user scenario

A HiperLAN/2 MAC frame consists of 5 parts and has a maximal duration of 2 milliseconds [13]. We assume that all parts

Table 3: Computational load of the Bluetooth receiver and transmitter functions mapped on a Pentium 4 processor.

TX function	M operations/s	M cycles/s	RX function	M operations/s	M cycles/s
MAC burst generation	1	1	Integer-to-float conversion	20	88
GFSK modulation	2100	440	FM-to-AM conversion	75	261
Float-to-integer conversion	80	320	Synchronization	1	27
			Frequency offset correction	1	Not implemented
			Bit decision	3	33
Total	2181	714	Total	100	437

Table 4: Computational load of the HiperLAN/2 receiver and transmitter functions mapped on a Pentium 4 processor for 64-QAM mode.

TX function	M operations/s	M cycles/s	RX function	M operations/s	M cycles/s
QAM mapping	38	85	Integer-to-float conversion	80	351
IFFT	230	128	Synchronization and parameter estimation	4	60
Float-to-integer conversion	80	215	Frequency offset correction	39	120
			FFT	230	318
			Channel equalization	39	79
			Phase offset detection and correction	40	127
			64-QAM demapping	77	204
Total	348	500	Total	509	1225

have equal duration and that we have to demodulate 2 parts (one common and one user part).

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

4.2.1. *Software*

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 because floating-point operations are as fast as fixed-point operations on a Pentium 4 processor. Moreover, we used the open-source FFTW library [14] for computing the inverse FFT and FFT. 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. It was observed that these conversions take a long time to compute and therefore special SSE instructions [15] are used for acceleration.

4.2.2. *Time measurement method*

Time measurements were performed on a Pentium 4 processor at 2.8 GHz by counting the number of cycles for each function. A Pentium 4 processor is a very complex design and therefore the number of cycles needed for computing a particular function is influenced by many parameters such as cache misses, memory alignment, and so forth. It is for that reason that we used average values in these time measurements. The number of cycles required for the whole receiver or transmitter function (total values) is measured separately and not determined by summing up all individual components.

4.2.3. *Results*

Table 3 lists, for each function of the Bluetooth transmitter and receiver, the number of required operations (multiplications, additions, etc.) and how much cycles this function needs on a Pentium 4 processor. Especially the GFSK modulation, conversion to fixed-point numbers of the Bluetooth transmitter, and FM to AM conversion of the receiver require most of the cycles. In the GFSK modulation function, a 60-tap Gaussian filter is used that requires 1000 million 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 optimalization reduces the amount of computations significantly.

Table 4 shows the number of required operations and cycles for each function of the transmitter and receiver for HiperLAN/2.

Computational intensive parts are the conversion to floating-point precision, FFT and 64-QAM demapping in the receiver, and conversion to fixed-point numbers in the transmitter. Moreover, the HiperLAN/2 transmitter requires less computational power than the Bluetooth transmitter, although more bits are transmitted by the first one. The HiperLAN/2 receiver requires on the other hand more cycles per second than the Bluetooth receiver, but the latter one operates also at a much lower sample rate.

4.2.4. *Experiments*

Baseband experiments have been performed with the setup in Figure 7. In HiperLAN/2 mode, successful transmission and reception of continuously transmitted MAC bursts is achieved. For Bluetooth mode, however, baseband experiments still have to be carried out but we do not expect problems as HiperLAN/2 is more demanding.

5. CONCLUSIONS

This paper describes a software-defined radio testbed for wireless LAN standards. The physical layer of the HiperLAN/2 standard has been implemented in software running real time on a normal PC and baseband experiments have verified the system. However, literature [6] shows that for HiperLAN/2, one of the most demanding parts is the FEC coding and FEC decoding (e.g., Viterbi algorithm) which we did not implement in the project. Additional research has to be carried out if this holds also for a Pentium 4 implementation and whether this limits a GPP-based software-defined radio. Moreover, further research focusses on increasing functionality of our testbed, such as implementation of other standards and performing RF experiments.

ACKNOWLEDGMENTS

We thank our colleagues from the IC-Design Group for their work on the analog part of the front end and for interesting discussions. Moreover, we thank Henny Kuipers and Geert-Jan Laanstra for building the DAC and ADC boards. This research is supported by the PROGram for Research on Embedded Systems & Software (PROGRESS) of the Dutch Organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs, and the Technology Foundation STW.

REFERENCES

[1] J. Mitola III, *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, John Wiley & Sons, New York, NY, USA, 2000.

[2] R. H. Walden, "Performance trends for analog to digital converters," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 96–101, 1999.

[3] V. J. Arkesteijn, E. A. M. Klumperink, and B. Nauta, "An analogue front-end architecture for software defined radio," in *Proc. 13th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC '02)*, Veldhoven, the Netherlands, November 2002.

[4] *The Bluetooth-HiperLAN/2 SDR receiver project website*, http://www.sas.el.utwente.nl/home/SDR/.

[5] W. Tuttlebee, *Software Defined Radio: Origins, Drivers and International Perspectives*, John Wiley & Sons, New York, NY, USA, 2002.

[6] K. Masselos, S. Blionas, and T. Rautio, "Reconfigurability requirements of wireless communication systems," in *Proc. IEEE Workshop on Heterogeneous Reconfigurable Systems-on-Chip (SOC '04)*, Hamburg, Germany, 2004.

[7] 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," in *Proc. 4th PROGRESS Workshop on Embedded Systems and Software*, Nieuwegein, the Netherlands, October 2003.

[8] 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," in *Proc. 3rd PROGRESS Workshop on Embedded Systems and Software*, Utrecht, the Netherlands, October 2002.

[9] I. Ripoll, P. Pisa, L. Abeni, et al., "Deliverable D1.1-RTOS analysis," http://www.mnis.fr/opensource/ocera/rtos/, 2002.

[10] R. Schiphorst, F. W. Hoeksema, and C. H. Slump, "A Bluetooth-enabled HiperLAN/2 receiver," in *Proc. IEEE 58th Vehicular Technology Conference (VTC '03)*, vol. 5, pp. 3443–3447, Orlando, Fla, USA, October 2003.

[11] R. Schiphorst, F. W. Hoeksema, and C. H. Slump, "A (simplified) Bluetooth maximum a posteriori probability (MAP) receiver," in *Proc. IEEE 4th Signal Processing Advances in Wireless Communications (SPAWC '03)*, pp. 160–164, Rome, Italy, June 2003.

[12] BluetoothSIG, *Specification of the Bluetooth System-Core*, Technical Specification Version 1.1, 'Bluetooth SIG', February 2001.

[13] ETSI, *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer*, Technical Specification ETSI TS 101 475 V1.2.2 (2001–2002), ETSI, February 2001.

[14] http://www.fftw.org/.

[15] Tommesani.com, MMX/SSE/SSE2 primer http://www.tommesani.com, 2003.

[16] http://www.semiconductors.philips.com/comms.

R. Schiphorst received his M.S. degree in electrical engineering from the University of Twente in 2000 for his research on software-defined radio. In September 2000, he joined the Signal and Systems Group of the University of Twente as a Ph.D. student to continue his research on software-defined radio. His research interests include the design and analysis of the digital part of software-defined radios. In September 2004, he finished his Ph.D. thesis and he is now working as a Research Scientist in the same chair.

F. W. Hoeksema studied electrical engineering at the University of Twente. After his graduation in 1987, he joined Philips Research Laboratories in Eindhoven and contributed to (pre JPEG) still image picture coding and CCD sensor modeling. In 1998, he joined the Department of Electrical Engineering at the University of Twente. In the TIOS/TSS Group, he worked in the field of image coding for ATM networks and at performance measurements for BISDN systems (ACTS INSIGNIA Project). In 2001, he switched to the Signals and Systems (SAS) Group to participate in projects that focus on digital signal processing for radio systems: the PROGRESS SDR Project, the Freeband AWGN Project, and recently the Freeband AAF Project. His current interest is in design and implementation of (digital) signal processing functions for radio systems and especially in parameter estimation for communication system purposes.

C. H. Slump received the M.S. degree in electrical engineering from Delft University of Technology, Delft, The Netherlands in 1979. In 1984, he obtained his Ph.D. degree in physics from the University of Groningen, The Netherlands. From 1983 to 1989, he was employed at Philips Medical Systems in Best as Head of a predevelopment group on medical image processing. In 1989, he joined the Signal and Systems Group of the University of Twente, Enschede, The Netherlands. His main research interest is in digital signal processing, including realization of algorithms in VLSI.