A Software Defined Radio Testbed Implementation

S. Weiss[1], A. Shligersky[1], S. Abendroth[1], J. Reeve[1], L. Moreau[1], T.E. Dodgson[2] and D. Babb[2]
[1] School of Electronics & Computer Science, University of Southampton, UK
[2] Samsung Electronics Research Institute, Samsung Ltd. (UK), Staines, UK

Email: {s.weiss,jsr,lavm}@ecs.soton.ac.uk, {tdodgson,dbabb}@seri.co.uk

Abstract

We report on the implementation of a software defined radio (SDR) based on state-of-the-art digital signal processors (DSPs), which are linked serially to PCs. While time critical operations are executed on specialised transmit and receive processors with a fixed block structure, the baseband processing is performed on highly flexible DSPs. The latter run several concurrent functionalities, such as the reception of data over a serial link, the assembly of symbols and frames for transmission, as well as receiver functions such as a fractionally spaced equaliser for synchronisation and mitigation of potentially dispersive channels under a DSP/BIOS system. The testbed system is capable of transmitting packets of data over the SDR link.

1. Introduction

The design of highly flexible digital communication systems has become an area of considerable interest. Particularly for mobile wireless transmission via hand-held devices, size and cost competitiveness usually set limitations when trying to implement systems compatible with multiple standards that exist throughout the, upcoming standards will overlap with existing ones for a significant interim period, such as for example for the transition from GSM to UMTS [1, 2]. Similarly. reconfiguring a device near-instantaneously is desirable if the required throughput over a wireless link varies between such extremes as voice transmission or down-loading a large image.

This has motivated the concept of a software defined radio (SDR), whereby the digital-to-analogue and analogue-to-digital conversion are performed as close as possible to the radio frequency. The aim of extending the digital domain is to implement modulation, demodulation, channel coding and other required processing tasks in software [3, 4]. Therefore, users, service providers, and manufacturers become more independent of the realisation of one specific data transmission standard, since by down-loading appropriate software code, a different functionality can be adopted by the communications system.

In this paper we report on the implementation of a software defined radio (SDR) test-bed comprising a transmitter and receiver, which is implemented on digital signal processors (DSP) performing the baseband operations. Similar testbeds have been reported, based on FPGAs in the upconversion and downconversion stages to an intermediate frequency (IF) [3, 4, 5]. Here, the conversion between baseband and IF is achieved by separate, fast, and programmable digital transmit and receive processors as suggested in [6, 7, 8], which offer a flexible approach to implement various modulation and data rate modes [9]. The transmitted and received IF signals have a resolution of 12 and 14 bits respectively, and can be sampled at a maximum of 65 MHz. The aim of our testbed is to provide a wireless link between two PCs, which are connected to the baseband DSPs via RS232 connections.

In the following, we give a brief overview over the hardware and software structure of our testbed in Secs. 2 and 3, and discuss its capabilities and limitations in Sec. 4. Conclusions and an outlook to future extensions are presented in Sec. 5.

2. Hardware Structure

The general setup of the implemented SDR transceiver is shown in Fig. 1. Two floating point DSPs operate as baseband units in both the transmitter and the receiver, while dedicated circuitry in form of a digital transmit processor (DTP) and a digital receive processor (DRP) are employed for the conversion and processing stages between baseband and IF. In the following, we comment first on the transmit function in Sec. 2.1 and thereafter on the receiver hardware in Sec. 2.2. In order to potentially include the testbed into a network, Sec. 2.3 addresses the data link between the baseband DSPs and PCs.

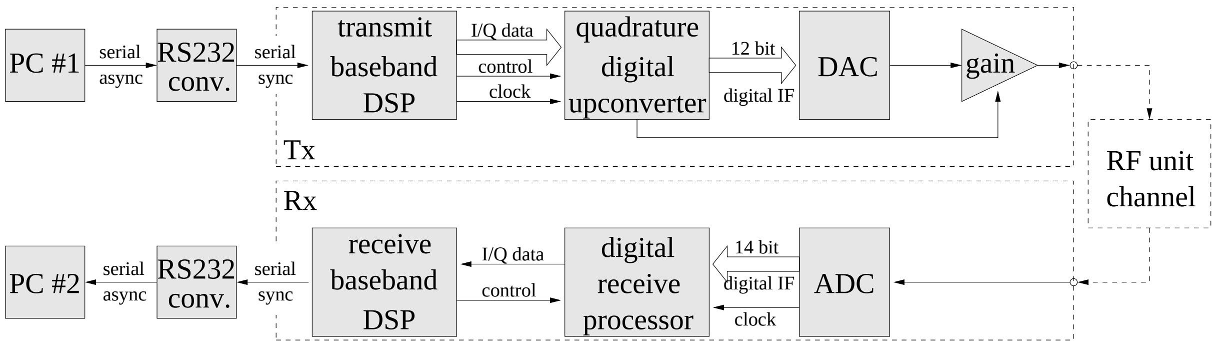

Fig. 1: SDR block diagram.

2.1 Transmitter Circuit

The transmitter hardware encompasses a Texas Instruments C6711 floating point DSP for baseband processing. Based on a suitable input signal, this processor provides a stream of complex valued symbols for transmission, which are then passed to a DTP, for which an Analog Devices AD9856 has been utilised. The selected DTP has an on-board digital-to-analogue converter,(DAC) to finally transform the digital data into an analogue signal at IF. As indicated in Fig. 1, the circuit is completed by an amplifier with variable gain, which can be controlled digitally.

Our selected baseband DSP is suitable for processing tasks that are highly demanding in terms of required computational complexity. and can therefore encompass tasks such as source and channel coding. The C6711 DSP used here, evaluating 900 million floating point operations per second at a clock rate of 150 MHz, resides on a low-cost DSP starter kit (DSK) board, which permits access to various ports and interfaces of the DSP. For the communication between DSP and DTP, both the DSP's data bus and one of its two multichannel buffered serial ports (McBSP) have been utilised. Inphase and quadrature components of the complex valued symbols are presented to the DTP interleaved and in a parallel fashion over the DSP's expansion memory interface, whereby 12 bits parallel data and a control line to enable transmission are occupied. The initialisation of the DTP and also its control during transmission are serviced via an McBSP by the DSP over a number of control lines.

The baseband data is transfered to the DTP via custom build latches, with inphase and quadrature data components interleaved. The DTP is placed on an evaluation board comprising an AD9856 quadrature digital up-converter. Within this up-converter, it is possible to download different sets of coefficients for transmit filtering and choose different oversampling ratios in the interpolation stages. At the output of the interpolation stage, a quadrature amplitude modulation is placed, where the inphase and quadrature signal components are multiplied by sine and cosine waveforms. At a maximum input bandwidth of 25 MS/s (considering both inphase and quadrature components), the maximum sampling rate at the output of the DTP is 160 MHz, which is limited by a 65 MHz lowpass filter and the rate of the on-board digital-to-analogue converter (DAC). This DAC has a resolution of 12 bits, and is followed by a programmable gain amplifier (AD8320) for power control. This gain amplifier can be regulated through the transmit processor, which in turn is controlled via the DSP's serial port.

As indicated in Fig. 1, the analogue IF signal could now be further modulated up to radio frequency and be transmitted via an antenna. A matching analogue circuit could receive the transmit antenna signal, and down-convert it to IF. In our test-bed, this hardware stage is currently omitted, and the IF signal from the DTP is passed straight back into a digital receive processor (DRP).

2.2 Receiver Circuit

On the receiver side, the analogue IF signal is fed to an AD6644ST analogue to digital converter (ADC) sampling at a maximum of 80 MS/s with 14 bit resolution. As shown in Fig. 1, this data is passed to the DRP over a parallel port. Here, this dedicated processor is an AD6620, which has a maximum input bandwidth of 65 MS/s for real signals or 32.5 MS/s for complex valued data. Over two stages of cascaded-integrator-comb (CIC) filters with definable decimation ratios, finally a programmable receive filter of order 255 can be applied to the data in the down-conversion process.

From the DRP, the inphase and quadrature data is then serially transmitted to the second C6711 DSP hosting the baseband receiver functions. Although the selected DRP also allows a parallel interface, the employed DSK board permits only to write to the McBSP but not the data bus on the expansion memory interface. Through the McBSP, the DSP can also control the parameters on the DRP, including the decimation ratios as well as the selection of the CICs and the receive filter.

In the baseband receiver DSP, the oversampled data is fed into a synchronisation and equalisation stage, which will be described in Sec. 3.. Thereafter, the complex data symbols are translated into a bit-stream. From this bit stream, finally another synchronisation process is required to extract the transmitted data words.

2.3 Baseband DSP to PC Link

In an earlier version of this testbed, on-board DAC and ADCs were used to function as source and sink of an audio signal to be transmitted via the testbed. In the version presented here, this data mode has been replaced by a serial link to PC via RS232 with the aim of embedding the testbed into a communications network [10].

The serial link is operated across the 2nd McBSP in both the Tx and Rx DSP. Since the PC serial port can by conveniently operated by a universal asynchronous receiver/transmitter (UART) protocol, the synchronous McBSP needs to be appropriately interfaced. Using methods in [11] and dedicated hardware to convert RS232 signalling levels to and from CMOS, the McBSP only needs to be driven by an internal clock and be provided with a suitable frame synchronisation signal, which can be derived from the transmitted data. Given such a frame synchronisation, in order to obtain an uncorrupted bit stream, unfortunately the signal needs to be oversampled in the McBSP. In our case, we have used a factor of 16 [10], which creates considerable redundancy and an interrupt rate higher than the bit rate thus burdening the DSP.

3. SDR Software

The heart of the SDR testbed are the C6711 DSPs, which run a TI DSP/BIOS real-time system and control the interfaces in addition to the specific Tx and Rx functions. DSP/BIOS provides a convenient multi-tasking capability through events, software- as well as hardware interrupts, which operate at various levels of priority. Other functionalities, such as configuring the DTP and DRP can be performed by either dedicated host PC software or through writing specific control words to an interface. The serial connection on the PC side is established by standard JAVA libraries. Thus, we concentrate on the baseband DSP software, and below separately describe software implementations for the transmitter and receiver baseband DSPs.

3.1 Baseband Transmitter

The Tx DSP is required to receive serial data packets from the PC, and form packets of symbols, which are embedded into a flexible frame in order to permit synchronisation in the Rx DSP. The data is oversampled and pulse shaped before passing it on to the upconverter. Thus, the aims of the baseband transmitter DSP implemented in software comprise:

(1) Initialisation of interfaces and interrupts.

(2) Initialisation of upconverter. This is done via McBSP0.

(3) Reception of data from the PC. this is performed over McBSP1, a the serial link described in Sec. 2.3, and transfers data via enhanced direct memory access (EDMA) whenever McBSP1 issues a hardware interrupt due to a full receive data register;

(4) Formation of data frames. In regular intervals, a frame synchronisation byte is inserted into the bit stream in order to later be able to perform word synchronisation in the receiver.

*(5) Mapping from bytes to symbols.*The bit stream is converted into complex quadrature amplitude modulated (QAM) symbols, whereby here a QPSK mode is employed, although higher modulation modes are possible [12].

(6) Pulse shaping. The symbol stream is moderately oversampled by a factor of 2, which is required as a minimum by the subsequent upconverter hardware, and filtered by a root raised cosine filter.

(7) Passing I/Q samples to upconverter. The oversampled and pulse shaped signal values are passed over the parallel port of the PC to the upconverter, alternating real and imaginary values. Synchronisation and clock signal need to be provided by the DSP to the upconverter.

While step (1) and (2) are only performed when the DSP is booted — which can be done from an EPROM on the C6711 DSK board — the remaining functions (3) – (7) run concurrently and at different rates, exploiting the capabilities of DSP/BIOS.

3.2 Baseband Receiver

In the Rx DSP, the data is received at a 4 times oversampled rate to enable Rx filtering and subsequent timing synchronisation. The timing synchronisation is performed by a fractionally spaced equaliser, which can be adapted either blindly based on the knowledge of the transmitted symbol alphabet [9] or on training data embedded in the transmitted frames [13]. Additionally, compensation for carrier mismatch, and byte synchronisation are performed [14]. The Rx DSP can then store the retrieved packet of data and output it serially to a second PC via RS232. In detail, the required functionality encompasses:

(1) Initialisation of interfaces and interrupts.

(2) Reception of I/Q data from downconverter. The real and imaginary signal values are receiver from the downconverter via the McBSP1. These samples are oversampled by a factor of 4 compared to the symbol rate in order to permit sufficient resolution for timing synchronisation.

(3) Pulse shaping. The 4 times oversampled complex baseband signal is filtered by a root raised cosine filter matched to the one in the transmitter, together forming a Nyquist system.

(4) Fractionally spaced equaliser. At a two times oversampled rate, the received signal stream is passed into a fractionally spaced equaliser, which performs blind channel equalisation and can at the same time accomplish timing synchronisation. For updating of the blind equaliser, a constant modulus algorithm is used in combination with differential QAM symbol transmission [13].

(5) Byte synchronisation. After the retrieval of the bit synchronisation in (4), the detection of the frame word is utilised for byte synchronisation.

(6) Data transfer to host PC. The detected frames are transfered to memory and, after accumulation of a packet, passed on via EDMA to the McBSP0 to the PC on the receiver side.

Step *(1)* is performed at boot level, while, different from the transmitter, the downconverter needs to be initialised via a host PC and dedicated software since integration with the baseband DSP has not been completed yet. The remaining steps *(3)* – *(6)* again have to be operated concurrently and at different rates. Processing of both the transmitter and receiver functions are performed in interrupt service routines, whereby the processor falls into an idle mode if no interrupts need to be services, from which it returns if new data or symbols need to be processed.

4. Capabilities and Bottlenecks

The current SDR testbed, as depicted in an experimental setup in Fig. 2, can transmit at up to 256 kbit/s in differential QPSK mode between the two baseband DSPs, whereby the limiting factor is the serial connection between the DRP and the baseband receiver DSP. Higher data throughput is in principle possible by selecting multi-level QAM schemes for symbol encoding, although this would result in a system more sensitive to channel noise. Removing the bottle neck would require the possibility of a parallel data transfer between the DRP and the DSP, which is difficult due to limitations on the selected DRP.

In the transmission mode shown in Fig. 1, packets of data can be transmitted from PC to PC across the SDR testbed. However, with the RS232 link being limited to 115 kbit/s and the oversampling of the bit stream in the McBSP1, the bottleneck is now moved from the DRP-DSP link to the serial PC connections in transmitter and receiver.

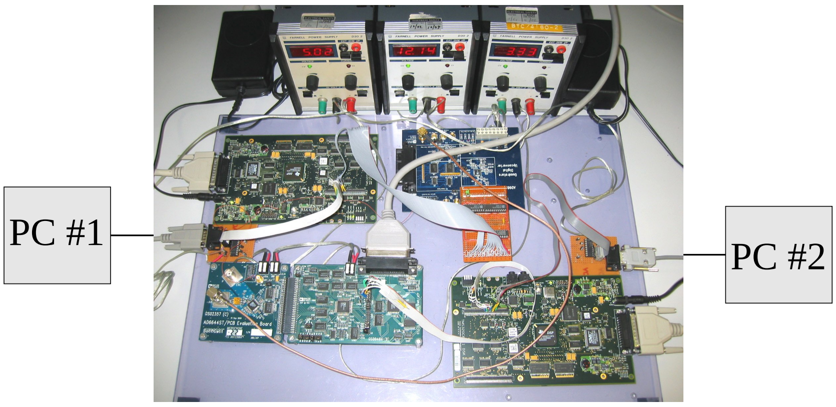

Fig. 2: Experimental setup of SDR testbed.

5. Conclusions and Further Developments

We have presented an SDR testbed and highlighted its structure and functionality, as well as some of its capabilities and limitations. The DSP processor is controlling various concurrent functionalities through a real-time DSP/BIOS system, and is at present capable of transmitting up to 256 bit/s in QPSK mode. The devices are currently configured when the system is booted up, and in case of the DRP, through dedicated host PC software.

We aim to achieve reconfigurability of the testbed by first adding flexibility to the baseband DSPs, whereby adaptive modulation with channel SNR dependent modulation levels appears to be a suitable candidate. Ideally, a multiband approach could be accomplished if the DSPs could in turn reconfigure the DTP and DRP. While this is currently possible on the transmitter side, where the DSP initialises and controls the DTP, the DRP is configured through external software and not accessible to the baseband DSP in the present form of the SDR.

References

[1] "Special Issue on Software Radio," *IEEE Journal on Selected Areas of Communications*, vol. 6, no. 4, August 1999.

[2] F. Jondral, A. Wiesler, and R. Machauer, “A Software Defined Radio Structure for 2nd and 3rd Generation Mobile Communication Standards,” in *IEEE 6th International Symposium on Spread Spectrum Techniques and Applications*, Piscataway, NJ, September 2000, vol. 2, pp. 637–640.

[3] C. Bonnet, G. Caire, A. Enout, P. A. Humblet, A. Montalbano, and D. Nussbaum, “Open Software Radio Platform for New Generations of Mobile Communication Systems,” in *3rd European DSP Education and Research Conference*, Paris, September 2000.

[4] M. Jian, S. R. Bai, K. T. Heng, and W. H. Yung, “A Software Radio Development Platform PCP200 — Partnering ’C6x with Virtex FPGA,” in *3rd European DSP Education and Research Conference*, Paris, September 2000.

[5] Multiple Access Communications Ltd., “Aspe 1000 advanced signal processing engine,” datasheet, http://www.macltd.com/ASPE.htm, 2002.

[6] T. Hentschel, M. Henker, and G. Fettweis, “The Digital Front-End of Software Radio Terminals,” *IEEE Personal Communications*, vol. 6, no. 4, pp. 6–12, August 1999.

[7] M. Schlosser, “Software radio development,” Diploma thesis, University of Southampton, Southampton, UK, May 2000.

[8] S. Abendroth, “Development of a Digital Front-End for a Software Defined Radio,” Tripartite fifth year project report, University of Southampton, Southampton, UK, May 2001.

[9] S. Abendroth, R. Heuzé, and S. Weiss, “A Software Defined Radio Front-End Implementation,” in *Proceedings 2nd Karlruhe Workshop on Software Radios*, Karlsruhe, Germany, March 2002, pp. 57–61.

[10] A. Shligersky, “Real-time system approach for the sdr front-end,” Diploma thesis, University of Southampton, Southampton, UK, June 2003.

[11] T. Hiers, R. Ma, P. Malleth, and S. Chen, “Tms320c6000 mcbsp: Uart,” TI Application Report, SPRA633A, August 2001.

[12] R. Heuzé, “Software Synchronisation and Equalisation,” Tripartite fifth year project report, University of Southampton, Southampton, UK, May 2001.

[13] C. R. Johnson, P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, and R. A. Casas, “Blind Equalization Using the Constant Modulus Criterion: A Review,” *Proceedings of the IEEE*, vol. 86, no. 10, pp. 1927–1950, October 1998.

[14] U. Mengali and A. d’Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, New York, 1997.