A SOFTWARE DEFINED RADIO FRONT-END IMPLEMENTATION

S. Abendroth[1,2], *R. Heuze*[1,3], *S. Weiss*[1]

[1] Dept. Electronics & Computer Science, University of Southampton, UK
[2] University of Karlsruhe, Karlsruhe, Germany
[3] Ecole Supérieure d'Ingénieurs en Electrotechnique et Electronique, Paris, France
s.weiss@ecs.soton.ac.uk

Abstract. We report on the implementation of a software defined radio (SDR) based on state-of-the-art digital signal processors (DSPs). 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. We comment on the SDR building blocks, and the software implemented on this test-bed including blind synchronisation, equalisation, and carrier recovery for differentially encoded quadrature amplitude modulation. The functionality has been tested by audio transmission. We conclude with an analysis of capabilities and limitations of the implemented SDR structure.

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 world. Similarly, 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].

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 downloading appropriate software code, a different functionality can be adopted by the communications system.

In this paper we report on the implementation of a software radio test-bed with an SDR transmitter and receiver. Both functions are implemented on C6711 digital signal processors performing the baseband operations. The conversion between baseband and an intermediate frequency (IF) is achieved by separate, fast digital transmit and receive processors as suggested in [5, 6]. 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.

We first discuss the hardware employed in realising the SDR transmitter and receiver in Sec. 2. Software implementations, particularly for the receiver functions such as blind adaptive carrier recovery, synchronisation, and equalisation, are outlined in Sec. 3. Sec. 4 highlights the testing and running of a 96 kbit/s data link across the SDR, while in Sec. 5 conclusions are drawn.

2 HARDWARE PLATFORM

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.

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. Finally, the second MCBSP on the baseband board is connected to a low cost on-board analogue-to-digital converter (ADC) acquiring a 12 bit single-channel audio data stream at a sampling rate of 8 kHz. In our setup, this analogue input is currently used to provide audio data for transmission as shown in Fig. 1.

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/sec (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/sec 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/sec for real signals or 32.5 MS/sec 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, another synchronisation process is required to extract the transmitted data words. Finally as shown in Fig. 1, our implementation sends these 12 bit data words to the

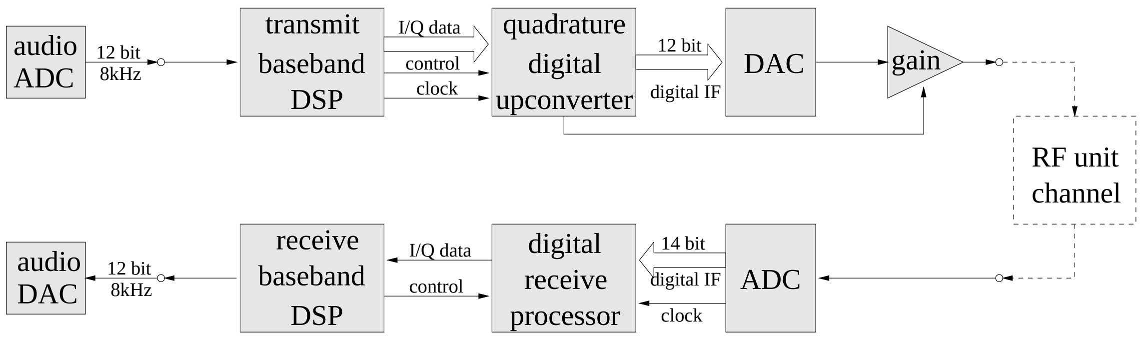

Fig. 1: Block diagram of the implemented software defined radio test-bed.

on-board DAC via the second McBSP to output an analogue audio waveform.

3 IMPLEMENTATIONS

After setting up the software controlling the different devices as well as communication between the baseband DSPs and the transmit and receive processors, the SDR platform outlined in Sec. 2 is reasonably flexible and can be programmed in C for any functionality within the capabilities of the test-bed. While in principle the bandwidth of the DTP and DRP would permit the implementation of W-CDMA with a required bandwidth of 5 MHz, using the McBSP1 in the receiver DSP and the baseband processing set limitations. Hence, in the following we describe the implementation of a data link of lower bandwidth such as found in GSM using the previously defined hardware platform.

3.1 Modulation and Demodulation

In the baseband DSP of the transmitter, the 12 bit audio samples were converted into a bit stream, which in turn was subjected to a Gray encoded differential QPSK (D-QPSK) scheme. Differential encoding was selected to enable blind synchronisation and equalisation without phase ambiguity in the receiver. The D-QPSK symbols are upsampled by a factor of four, and pulse shaping with a root raised cosine filter is applied prior to transferring the data to the DTP.

The clocks in the transmitter and receiver circuits are governed by the baseband DSPs. As after demodulation a clock mismatch will lead to a carrier frequency offset, a time-varying rotation of the received D-QPSK symbols and a slight difference in the transmitted and received data rates can result. In order to mitigate these problems, the receive baseband DSP requires synchronisation and carrier offset recovery [7, 8]. This synchronisation is performed together with timing synchronisation and equalisation, which will be outlined in the next section.

3.2 Synchronisation and Equalisation

Amongst a variety of algorithms for carrier and timing offset compensation, such as phase-locked-loops and early-late-gate methods [7], adaptive filtering was chosen as a convenient way of solving the different synchronisation issues. We here apply a fractionally spaced adaptive equaliser, whereby the input is oversampled at twice the symbol rate to achieve timing greater resolution over standard symbol-spaced equalisers [9].

To recover the D-QPSK symbols, different adaptive schemes can be invoked. Standard adaptive algorithm can be employed if a training sequence is available. For our SDR test-bed, a blind synchronisation and equalisation scheme was preferred, whereby the transmitted data is unknown and recovery is

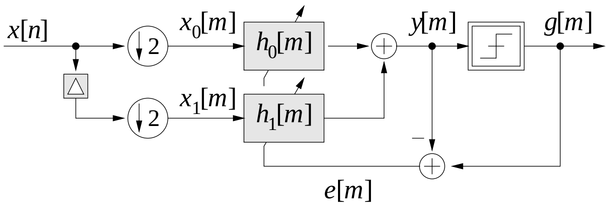

Fig. 2: Fractionally spaced adaptive equaliser in polyphase realisation with decision-directed updating.

based on the knowledge of the transmitted symbol alphabet only. A well behaved class of adaptive equalisers belong to the family of constant modulus algorithms (CMA) [10]. The CMA however leaves the phase of the received symbols ambiguous. Therefore, as specifically D-QPSK modulation had been selected for data encoding, a decision-directed scheme without phase-ambiguity was selected, as shown in Fig. 2.

With the oversampled received data $x[n]$, both the received data available to the equaliser at sampling period m as well as the equaliser coefficients $h[m]$ can be denoted in vector notation as

$$\boldsymbol{x}_m = \begin{bmatrix} \boldsymbol{x}_{0,m} \\ \boldsymbol{x}_{1,m} \end{bmatrix} , \qquad \boldsymbol{h}_m = \begin{bmatrix} \boldsymbol{h}_{0,m} \\ \boldsymbol{h}_{1,m} \end{bmatrix} , \tag{1}$$

whereby

$$\boldsymbol{x}_{i,m}^{\mathrm{T}} = \big[x_i[m] \;\; x_i[m-1] \;\; \cdots \;\; x_i[m-L+1]\big] \tag{2}$$
$$\boldsymbol{h}_{i,m}^{\mathrm{T}} = \big[h_{i,m}[0] \;\; h_{i,m}[1] \;\; \cdots \;\; h_{i,m}[L-1]\big] , \tag{3}$$

with $i = \{0, 1\}$. The output $y[m]$ of the equaliser is fed into a QPSK decision device with output $g[m]$. The difference between the input and output of this slicer therefore defines the error of the equaliser,

$$e[m] = g[m] - y[m] = g[m] - \boldsymbol{h}_m^{\mathrm{H}} \cdot \boldsymbol{x}_m \ . \tag{4}$$

In our scheme, the decision of the correct slicer output is based on the closest vicinity of a QPSK constellation point $s_{l,k} = \big((-1)^l + j(-1)^k\big)/\sqrt(2)$, with $j = \sqrt{-1}$ and $l, k \in \{0; 1\}$, to the received symbol $y[m]$,

$$g[m] = \arg\min_{s_{l,k}} |y[m] - s_{l,k}| \ . \tag{5}$$

With a suitable error minimisation criterion, for example a decision-directed normalised least mean squares (NLMS) adaptive algorithm

$$\boldsymbol{h}_{m+1} = \boldsymbol{h}_m + \tilde{\mu} \cdot \frac{\boldsymbol{x}_m}{\|\boldsymbol{x}_m\|^2} \cdot e^*[m] , \tag{6}$$

the filter coefficients in the equaliser can be automatically adjusted [9, 11].

The fractionally spaced equaliser described above is capable of carrier offset recovery by following a dynamic optimum solution,

$$\boldsymbol{h}_{\mathrm{opt},m} = \boldsymbol{c}_{\mathrm{inv}} \cdot e^{j(\Omega_{\mathrm{off}} \cdot m + \vartheta_{\mathrm{off}})} , \tag{7}$$

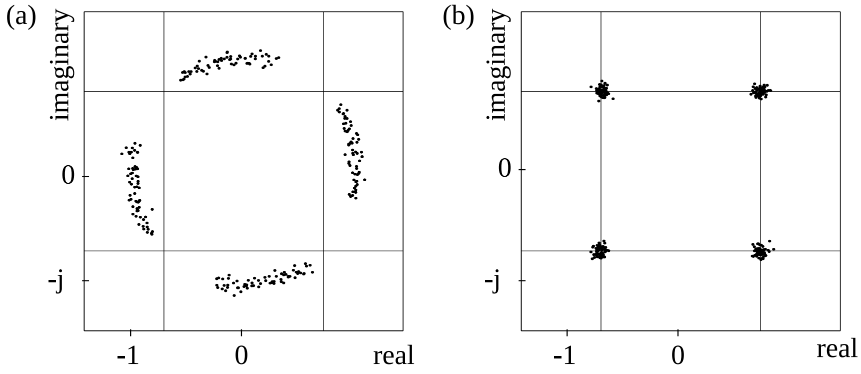

Fig. 3: Received constellation pattern (a) without and (b) with carrier offset recovery by the adaptive equaliser.

whereby $\boldsymbol{c}_{\rm inv}$ is the linear time-invariant channel inverse regularised by the channel noise, and $\Omega_{\rm off}$ and $\vartheta_{\rm off}$ are the carrier normalised angular frequency and phase offsets, respectively. Timing recovery is achieved by $\boldsymbol{c}_{\rm inv}$ modeling a fractional delay filter, whereby the equaliser benefits from an increased resolution due to fractional spacing. Therefore, as long as the initial starting value for $\boldsymbol{h}_m$ is reasonably close to the true solution, and the carrier frequency offset $\Omega_{\rm off}$ is within limits, the adaptive algorithm in (6) can converge to and track the optimum solution for the equalisation problem.

4 OPERATION AND AUDIO TRANSMISSION EXAMPLE

The operation of the SDR can be verified by accessing all memory locations and registers on the DSPs via the software tools provided for the C6711. An example for the operation of the carrier recovery and synchronisation is given in Fig. 3(a) and (b), where the received constellation over a short time interval shows the rotation of the symbols which in Fig. 3(b) has been correctly compensated to retrieve the data.

Therefore with the correct operation of the overall SDR hardware and software as outlined in Fig. 1, a 96 kbit/s transceiver for narrowband speech / audio can be performed. This data rate agrees with the 12 bit word length and 8 kHz sampling rate of the baseband unit on-board ADC/DACs. The data was modulated to and demodulated from D-QPSK symbols in both baseband DSPs and four times oversampled. Adjustments were made in the DTP and DRP such that an IF of 5 MHz was achieved. In the receive baseband DSP, all synchronisation task were performed blindly and adaptively.

5 DISCUSSION AND CONCLUSIONS

A software radio design has been implemented based on state-of-the-art DSPs. The time critical operations of the digital front-ends were delegated to specialized processors with a fixed but programmable block structure, while the baseband functionality is fully programmable on floating point DSPs. The cricuit and the application to an audio signal transmission at 96 kbps has been outlined.

The limitations of the structure are currently the use of the serial port for communication between the digital receive processor and the baseband receive DSP, as well as the speed of the baseband DSPs. These limitations manifest themselves in a restriction to approximately 256 kbit/s bandwidth for transmitted D-QPSK data. While GSM or similar standards can be well implemented within such limits, for example W-CDMA has a too high bandwidth requirement for our specific SDR realisation.

The flexibility of the presented structure has to be based on variation of D-QPSK transmission, or potentially the transmit and receive filtering, or the channel coding. Some of our current work focuses on implementing a differential 16-QAM data transmission, whereby blind synchronisation schemes have to be more elaborate as in the simpler D-QPSK case. The ultimate aim is to apply adaptive QAM, whereby the modulation level is adjusted to hostility of the channel, i.e. high level QAM schemes with a high data throughput are call on whenever the channel quality permits this.

6 ACKNOWLEDGEMENT

We would like to thank Prof. L. Hanzo for encouragement and advice, and M. Schlosser and J. Stefanov for their contribution in form of an early version of the presented testbed.

7 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] 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.

[6] M. Schlosser, "Software radio development," Diploma thesis, University of Southampton, Southampton, UK, May 2000.

[7] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, Kluwer Academic Publishers, Boston, 2nd edition, 1994.

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

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

[10] 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.

[11] 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.