SOFTWARE IMPLEMENTATION OF THE IEEE 802.11A/P PHYSICAL LAYER

Teo Cupaiuolo, Daniele Lo Iacono, Massimiliano Siti and Marco Odoni
Advanced System Technology
STMicroelectronics Italy
{teo.cupaiuolo,daniele.loiacono,massimiliano.siti,marco.odoni}@st.com

ABSTRACT

Software defined physical layer modems represent one of the main trends in communications and computing emerged in recent years. This is due on one hand to the need of supporting the requirements of modern communications systems in terms of seamless integration between different wireless technologies and multimedia convergence. On the other hand, programmable platforms are also beneficial as they allow consolidating methodologies, shortening development time and costs, extending products life-time. The drawback is that the complexity and power overhead of a pure computing fabric compared to a dedicated hardware can still represent a cost product developers are not willing to pay. This is particularly true for mobile terminals or in general for battery-powered devices. To become a concrete opportunity, baseband platforms should provide almost the same performance of custom designs while maintaining a certain degree of programmability. A good trade-off is represented by specific architectures integrating a proper mix of fine-grain general-purpose instructions and dedicated coarse-grain instructions wrapping custom hardware modules.

This paper presents a software implementation of a dual-mode IEEE 802.11a/p receiver on the Block Processing Engine (BPE), a proprietary template platform specifically designed for baseband processing. The combination of a novel extended instruction set with multi-thread processing support allows satisfying the most demanding requirements of the 802.11a and 802.11p standards.

1. INTRODUCTION

The automotive segment is now entering into the wireless arena with applications requiring reliable high data rate communications and including among the others vehicles safety, traffic control, remote tolling. To address vehicular environments, the IEEE recently introduced a new amendment to the 802.11 set of standards for wireless local area networks (WLAN) [1], namely IEEE 802.11p [2] for Wireless Access in Vehicular Environment (WAVE). Like several other standards developed for wideband digital communication, standards [1] are based on Orthogonal Frequency-Division Multiplexing (OFDM) modulation: among the others, a significant advantage of OFDM over single-carrier schemes is its ability to cope with severe channel multi-path conditions without the need for complex equalization filters.

The physical (PHY) layer [2] is mostly derived from the popular standard for indoor WLAN, IEEE 802.11a [3], with slight modifications required by the vehicular environment. The differences are designed to sustain robust vehicle-to-vehicle (V2V) communications as well as between vehicles and infrastructures located roadside (V2I). With respect to the indoor scenario, V2V and V2I require faster access time (<50 ms), increased range, robustness, reliability, security and indeed mobility. Being used also for critical safety applications, it will operate on the Dedicated Short Range Communications (DSRC) licensed spectrum at 5.9 GHz, exclusively reserved free-of-charge by the US FCC to vehicular communications.

These modifications come at the cost of a different and more complex digital baseband processing. Even with the recent proliferation of top performing programmable platforms, the traditional approach of implementing the PHY layer in hardware is still considered by many the best approach to fully satisfy the crescent needs of modern communications: area footprint and power consumption are a growing concern especially when targeting embedded portable devices. On the other hand, the process of developing an Application Specific Integrated Circuit (ASIC) is recognized as being long and costly, and can seriously compromise the ability to track a continuously evolving market, as typical of modern telecommunication systems. From this perspective, Software Defined Radio (SDR) represents a good opportunity strongly supported by research community: in its widest meaning, it aims to entirely replace the operations previously performed in hardware by purely software programmable resources.

Several advanced techniques have been introduced in the last years in order to fill the gap between the software and the hardware approach: the former has the undoubted advantage of reducing the development time, but it comes at cost of lower data rate, higher power consumption and area

TABLE I
THE 802 11A AND 11P STANDARDS
(MANDATORY VERSION)

	802.11a [3]	**802.11p [2]**
data rate	6–54Mbps	3–27Mbps
OFDM symbol duration	4µs	8µs
nb. of subcarriers	64 (48 data, 4 pilots, 12 virtual)	
bandwidth	20 MHz	5, 10 MHz
modulations orders	BPSK, QPSK, 16-QAM, 64-QAM	

requirement. One of the most widely used technique to effectively increase the throughput is the parallel processing enabled by Single Instruction Multiple Data (SIMD), which takes advantage of the high degree of data parallelism usually found in most telecommunication algorithms. To be effective, SIMD machines adopt parallel data paths of 128 bit or more. Another family is the Very Long Instruction Words (VLIW), wherein different types of functional units work in parallel. To further increase the performance, recent vector processors typically combine both of them [4][5]. Another recently emerged technique is the Single-Instruction Multiple-Task (SIMT), introduced by Coresonic to solve some of the major SIMD/VLIW drawbacks, like control overhead and improved memory utilization [6].

By today, various articles can be found about software implementation of the PHY of most common OFDM based standards, like [7][8][9], just to mention some of them. All these share the common approach of proposing a solution having some parts in software and some others in hardware. Typically software is used where there is some need for flexibility as can be the case of multi-standard support or real-time PHY reconfiguration; another case is represented by scalability requirements, like variable FFT sizes. Intensive data processing (as FIR filtering, FFT computation and Viterbi decoding) is usually left in hardware. Despite the commonalities with the 11a, indeed few works have been published about the 11p PHY: it can be recalled that [10] and [11] both developed a prototyping board reporting some implementation results, but these are still entirely hardware based.

This paper deals with a software implementation of a dual mode 802.11a/p receiver over the BPE, a mixed-grain template architecture designed for telecommunications and specifically customized for OFDM systems. We will show that a well balanced set of fine and coarse grain instructions allows to efficiently map the entire digital baseband receiver on a reconfigurable core. The distinctive flexibility of vectors processing of the BPE allows dealing with very different algorithms, ranging from filtering and synchronization down to the bit processing of de-puncturing and de-interleaving. To reduce the computational load while introducing no significant penalty in the programmability,

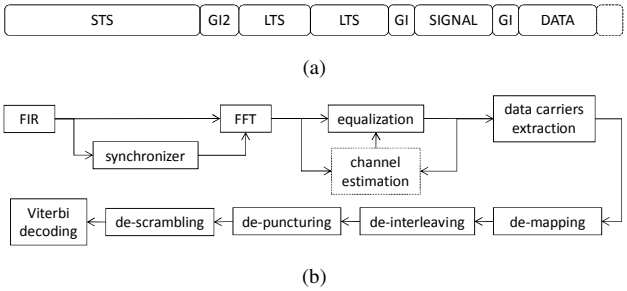

Fig. 1. (a) IEEE WLAN 802.11a/p frame structure and (b) receiver block diagram

Viterbi decoder has been implemented in hardware. Further, a new introduced multi-thread support allows executing several functions in parallel, thus effectively reducing the processing time compared to the single-thread version: the outcome is a software PHY that satisfies the most demanding data rate of 54 Mbit/s and 27 Mbit/s required by [3] and [2], respectively.

The paper is organized as follows: Sec. 2 recaps the typical algorithms involved in receivers complying with [2][3]; Sec. 3 gives an overview of the BPE processor; Sec. 4 details the algorithms mapping and profiling over the BPE; lastly, Sec. 5 concludes this paper.

2. THE IEEE 802.11A/P RECEIVER

2.1. Overview

Fig. 1a shows the structure of the WLAN frame [2][3], while the block diagram of the dual-mode receiver is shown in Fig. 1b. Referring to the frame structure depicted in Fig. 1a, the Short and Long Training Sequence (STS and LTS respectively) are known preambles used to perform time synchronization and channel estimation (CE). Details on both processes are provided later in this section.

The frame payload is composed by a first OFDM symbol (SIGNAL) holding, among the others, modulation parameters needed to demodulate the subsequent symbols (DATA): being crucial for the correct reception of the frame, the SIGNAL field is BPSK modulated and coded with maximum protection. Each symbol includes a guard interval (GI) of 16 samples to eliminate inter-symbol interference (ISI). Every OFDM symbol consists of 64 subcarriers composed by 48 data carriers (N_{SD}), 4 pilot tones (N_{SP}) and 12 null carriers (switched off to comply with spectrum emission mask). The OFDM symbol duration is 4 µs in case of 20 MHz bandwidth and 8 µs for 10 MHz.

TABLE I lists the main features of the two standards. Receivers able to demodulate both waveforms can share most of the processing, the main differences being filtering and CE functions. First of all, the larger delay spread makes the length of the 11a GI inadequate to prevent inter-symbol

interference (ISI). To face this problem while maintaining the same OFDM symbol structure (and hence same PHY processing), GI duration has been doubled by simply halving the channel bandwidth from 20 to 10 MHz. Another effect of the vehicular channel is the Doppler shift which must be compensated at receiver side with a more sophisticated estimate of the channel.

For both systems the processing performed at the receiver side can be divided in four phases: 1) *coarse synchronization*, i.e. packet detection and coarse carrier frequency offset (CFO) estimation based on the STS; 2) *fine synchronization*, i.e. frame synchronization and fine CFO estimation based on LTS; LTS is also used for the CE; 3) afterwards, processing of the signal field allows to identify the Modulation and Coding Scheme (MCS) chosen at the transmission side, i.e. the modulation order and the code rate associated to the next incoming data and other parameters, like the packet length; 4) lastly, the data field is demodulated to generate the bit stream to be propagated to upper layers.

2.2. Receiver Blocks

In this paragraph we shortly recap the typical algorithms performed at the receiver to enable the demodulation of the OFDM symbols, i.e. synchronization and CE.

2.2.1. The Short and Long Training Sequence

The STS is used to detect the presence of a frame, i.e. to perform the coarse timing synchronization. The algorithm takes advantage of the STS autocorrelation properties [12].

The coarse time synchronization can be performed by computing the M_n timing metric of the n-th sample:

$$M_n = \left|P_n\right|^2 \Big/ Q_n^2 \,, \tag{1}$$

where P_n is the autocorrelation function of $L = 16$ received samples r_n and Q_n their energy:

$$P_n = \sum_{k=0}^{L-1} r_{n+k+L} \cdot r_{n+k+L}^{*} \,, \tag{2}$$

$$Q_n = \sum_{k=0}^{L-1} \left|r_{n+k+L}\right|^2 \,. \tag{3}$$

Given the periodic structure of the STS, the timing metric takes a plateau form that begins with the first STS symbol and gradually decreases with the subsequent symbols: the frame is detected if M_n is between a (chosen) lower and an upper threshold for a given number of consecutive samples.

Then, CFO compensation takes place, using the following phase error coarse estimation:

$$\phi = \angle P_n \big/ N_{SP} \,, \tag{4}$$

where the phase of the autocorrelation function is computed over a variable number of samples (N_{SP}) depending on the estimated noise variance value.

After frame detection and CFO estimation and compensation, frame synchronization takes place in order to find the frame start. It is based on the cross-correlation between the incoming signal and a local replica of the LTS. At this time, coarse CFO estimation is refined using the autocorrelation properties of the LTS, carried (as for the STS processing) over a variable number of samples (N_{LP}). After CFO compensation, the GI of each OFDM symbol is removed and the 64 samples are transformed from the time domain to the frequency domain by 64-points FFT.

The next samples belonging to the LTS are used to compute the CE required during the processing of the OFDM symbols for equalization and soft-output de-mapping of the received bits. Afterwards, the GI2 (32 samples) is removed and the frequency domain per-tone CE H_m (for the m-th OFDM subcarrier) is computed. The basic CE corresponds to compute the mean of the two equalized received LTS $Y_{LTS1,m}$, $Y_{LTS2,m}$:

$$H_m = \frac{1}{2} \frac{Y_{LTS1,m} + Y_{LTS2,m}}{L_m} \,, \tag{5}$$

where the known LTS is denoted as L_m.

2.2.2. Signal and Data Field Processing

The signal and data fields are processed by the same algorithms, wherein the signal samples are BPSK modulated and define the MCS of the subsequent data samples. For every incoming OFDM symbol, the GI is removed and a group of 64 samples are Fourier transformed and equalized: using the previously computed CE, the symbols are (soft) de-mapped, de-interleaved and de-punctured and finally de-coded by a Viterbi decoder to produce the bit stream to be sent to the upper layers.

3. THE BLOCK PROCESSING ENGINE

In this section, we will recall only the main features of the BPE; a throughout description of the BPE can be found in [13][14]. Compared to the previous versions, the Instruction Set Architecture (ISA) has been extended withtrigonometric functions, coarse-grain FIR filtering, code generation and bit-level manipulation. From the core architecture perspective, in order to fully exploit data pipelining, it has been added the support for multi-thread function calls.

3.1. Vector Processing

The template architecture of the BPE is given in Fig. 2. The ISA implements two types of instruction: 1) basic scalar instructions (b-instructions) mainly devoted to flow control

and data access configuration; 2) dedicated vector instructions (d-instructions) performing intensive data processing. While b-instructions are locally executed, d-instructions are executed on the customizable dedicated unit (d-unit) bank. Depending on data dependencies and resources availability, units can be scheduled by the controller to run in parallel. Vectors are allocated on the dedicated memory (d-memory) bank, a set of static embedded memories allowing fast and parallel access to data.

The interconnection between d-memory bank and d-unit bank is guaranteed by the routing mesh, which is run-time configured by the controller on an instruction-by-instruction basis. To further optimize the data exchange between units, the routing mesh supports instruction pipelining through direct connection between units, thus avoiding the typical register pressure drawback of VLIW architectures: a group of pipelined d-instructions is called *macro* (instruction). Pipelined processing is the key enabler for high computational efficiency, since it allows propagating data from unit to unit without needing to store intermediate results for subsequent processing.

The controller fetches and schedules instructions one after another until one of them requires resources that have been already allocated, as can be the case for a memory or another unit. It then waits until the execution of the instructions using those resources has been completed. A side benefit of such policy is that b-instructions executed right after the scheduling of d-instructions do not cause additional delay. The latter consideration inherently suggests that maximum efficiency can be reached only using vectors long enough to absorb b-instruction execution.

3.2. The d-Instruction Set

The type of processing required by a generic PHY might be quite heterogeneous when moving down the receiver chain. The level of granularity changes according to the processing stage: whereas some blocks perform processing on a carrier by carrier basis, some others are characterized by bit-level granularity, which can cause a processing time penalty in a purely SIMD based processor. Furthermore, the type of operations performed changes significantly, ranging from typical signal processing, as can be the case of filtering, cross-correlation, equalization, to pure memory addresses computation as required for instance by the de-interleaver. Another interesting example might be the scrambler, which performs fast and with small complexity overhead only if supported by a dedicated unit (and related instruction) implementing a generic programmable linear feedback shift register (LFSR).

For the above reasons, the ISA has been extended with new instructions, like scrambling, Finite Impulse Response (FIR) filtering as well as more advanced operators like

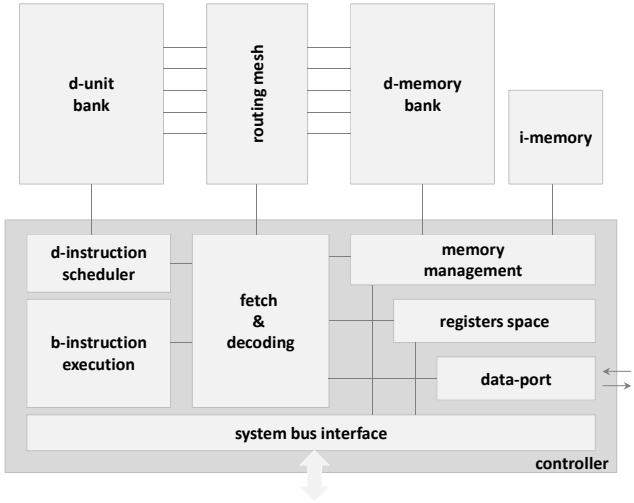

Fig. 2. The BPE template architecture

division, square root, hyperbolic and trigonometric functions all based on a low-latency fully pipelined CORDIC operator [15].

3.3. Multi-Thread Support

In [14] the technique of macro instructions pipelining has been introduced, wherein the macros themselves are defined as a group of pipelined d-instructions. Once the data dependencies of a given algorithm have been identified, macros can be connected to form a pipelined processing, where the pipeline stages are decoupled by memories implementing a ping-pong mechanism.

The macro-pipeline technique performs well within the processing of a given algorithm: the combinatorial path from the data source to the data output can be chopped into smaller pipelines until the required throughput is reached. Instead, a complete digital receiver involves several algorithms whose processing time can be effectively reduced by executing several functions in parallel. It will be shown that this enhancement is a key enabler for high data rate software based implementation.

4. MAPPING OF THE 802.11A/P PHY ON THE BPE

4.1. Short and Long Preamble Processing

The most computationally intensive kernels during the STS and LTS processing are the synchronization and the CE algorithms.

4.1.1. Synchronization and CFO computation

Provided that synchronization happens quite before data demodulation, a function implementing the synchronization algorithm is allowed to use all the available resources.

For an efficient mapping, eq. (2) and (3) can be rewritten in an iterative way: for example, the autocorrelation function P_n can be rewritten in vector form, like

$$P_{n+1} = P_n + r^*_{n+L} \cdot r_{n+2L} - r^*_n \cdot r_{n+L}. \quad (6)$$

The phase error (eq. (4)) is computed using CORDIC dedicated instructions. Once the CFO estimate is available, blocks of incoming samples are rotated executing a vector instruction from the d-instruction set.

TABLE II lists the clock cycles required for the synchronization function: the value has to be interpreted as latency, since the synchronization algorithm is not deterministic in nature. When clocking the BPE at 250 MHz, this latency translates into the need of buffering 2 OFDM symbols.

4.1.2. TD-LS channel estimation

The channel is first estimated using LTS (see eq. (5)) and further improved through reduced rank Time Domain Least Square (TD-LS) CE [17]: the incoming data are converted to the time domain (via inverse FFT), smoothed by a pre-computed matrix (related to the LTS itself) and converted back to the frequency domain. The main computational load is due to the IFFT and FFT operations: these are discussed in the subsequent section.

4.2. DATA Field Processing

4.2.1. The Fast Fourier Transform

The Fourier Transform is computed based on the Fast Fourier Transform (FFT) which is an efficient reformulation of the original algorithm that considerably reduces the computational load. In order to achieve fast FFT computation, the BPE embeds radix-2 and radix-4 butterflies as well as address calculation functions allowing data re-ordering among stages (e.g. bit-reverse). TABLE II lists the processing time required by the FFT software implementation.

4.2.2. Soft de-mapping

The simplified soft-out demapper algorithm [16] has been adopted: it avoids using costly (in terms of cycles) conditional constructs, without noticeable performance degradation compared to the optimal.

4.2.3. Intra-vector permutations: data carriers extraction, de-interleaving, de-puncturing and de-scrambling

Down the receiver chain, after de-mapping and prior to decoding, some specific intra-vector permutations need to be carried out: such operations do not perform any computation, but rather concern a change of position of the data within the vector.

TABLE II
PHY ALGORITHMS PROFILING

standard	algorithm	cycles	time [µs] @ 250 MHz
802.11a/p	synchronization [a]	1536	6.14
@DATA FIELD (MCS-7)			
802.11a/p	filtering (FIR)	162	0.65
	FFT (radix-4)	200	0.80
	equalization	64	0.26
	carriers re-ordering	56	0.22
	de-mapping	348	1.39
	de-interleaving [b]	408	1.63
802.11p	HDD DA CE	96	0.38
	TD-LS CE [c]	759	3.04
Total 802.11a[d]		1432	5.7
Total 802.11p[d]		2150	8.6

[a] it includes frame detection, CFO estimation and compensation
[b] it includes de-puncturing and de-scrambling
[c] based on the radix-4 FFT
[d] including software overhead

When performing this type of processing, SIMD based architectures turn to be heavily underutilized thus incurring in a waste of resources and a considerable increase in the required processing time. Power overhead is another side effect of underutilizing the vector. In turn, the BPE has a flexible vector management support, as described in Sec. 3.1: a fast execution time is achieved as it can be seen from TABLE II.

4.2.4. The Time Varying Channel Tracking

For a reliable communications in a mobility scenario, the time-varying channel has to be tracked, i.e. the initial CE based on LTS (Sec 4.1.2) must be updated for every OFDM symbol. This is especially true when the relative speed between two moving objects (as for two vehicles) becomes large and the Doppler effect has an impact on the system performance. In this work, the algorithm described in [18] has been implemented, specifically the TD-LS Hard Decision Directed (HDD) Data Aided (DA) CE, briefly referred to as TD-LS HDD CE.

During data processing, the CE is performed in a two steps process, as shown in Fig. 3: 1) data detection of the k-th received OFDM symbol using the CE corresponding to the (k–1)-th OFDM symbol; 2) the channel corresponding to the k-th OFDM symbol is estimated by using the newly detected symbol information based on HDD and followed by the TD-LS refinement.

The HDD CE is computed as follows: 1) frequency domain equalization of the m-th subcarrier of the k-th OFDM received symbol $Y_{k,m}$ with the available channel

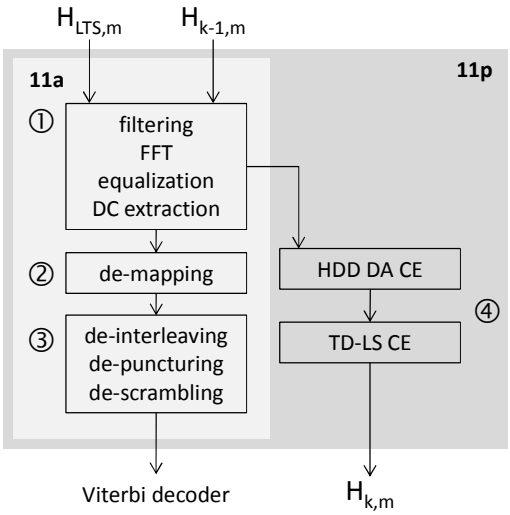

Fig. 3. DATA field processing (circled numbers refer to threads): ① filtering, FFT, equalization and data carriers extraction; ② de-mapping; ③ de-interleaving and de-puncturing; ④ channel estimation update

frequency response $H_{k-1,m}$ as $X_{k,m} = Y_{k,m}/H_{k-1,m}$; 2) hard detection of estimated data symbols $A_{k,m}$ to obtain the symbol sequence $\tilde{A}_k$: the hard detection implies hard de-mapping of the received symbol by associating the closest QAM symbol $A_{k,m}$ to the received equalized symbol $X_{k,m}$ (in terms of Euclidean distance); 3) estimation of H_k performed using as input the received symbol Y_k and the reconstructed $\tilde{A}_k$, used as an a-priori known sequence.

The CE is a computationally intensive task: a significant portion of time is spent on the FFTs and the filtering operation; refer to TABLE II for the overall processing time.

4.3. Single-Thread vs Multi-Thread Parallelism

As discussed in Sec. 3.3, the BPE versions [13][14] were based on a single-thread processor which implies that the overall processing time of the DATA field is given by the sum of the execution time of the single functions.

The first two rows of TABLE III list the clock cycles and the processing time required by a single thread to demodulate one DATA field, that is 5.7 µs for the 11a and 8.6 µs for the 11p PHY (including the overhead due to the software implementation, i.e. instructions scheduling and execution): both exceed their respectively OFDM symbol duration of 4 µs and 8 µs. The only way to increase the throughput is by taking advantage of the multi-thread support of the BPE.

Focusing first on [3], the idea is to pipeline the functions implementing the different stages of the processing, so as to demodulate different OFDM symbols in parallel: the parallelism degree is a combination of target data rate, single-stage pipe duration and availability of resources (memories and units). The multi-thread processing

TABLE III
DATA FIELD PROCESSING (MCS-7)

parallel threads	standard	clock cycles	time [µs] @ 250 MHz
1	802.11a	1432	5.7
1	802.11p	2150	8.6
3	802.11a	596	2.4
2	802.11p	1490	5.9

for the DATA payload is shown in Fig. 4a, wherein the algorithms are split among three parallel threads (①,② and ③) according to the diagram block of Fig. 3 (light-gray area): within each pipeline stage, the functions are processed serially, while the stages itself are executed in parallel. Once the overall pipeline has been filled, three OFDM symbols are elaborated concurrently, resulting in an overall elaboration time of 2.4 µs.

Regarding [2], the (TD-LS HDD) CE is the main bottleneck: the channel tracking is a recursive algorithm, thus the DATA processing cannot be pipelined as for [3]. Still, as explained in Sec. 4.2.4, the CE works independently from the demodulation thread and thus it can be run in parallel along the detection path, as shown in Fig. 3 (dark grey area): the previous three demodulation threads (①,② and ③) are executed one after each other, but in parallel with the CE thread (④). The corresponding (2-stage) multi-thread processing is shown in Fig. 4b: the overall demodulation time is 5.9 µs, which is well below than the target 8 µs symbol duration complying with [2] (see TABLE III).

5. CONCLUSIONS

In this paper we investigated the feasibility of a software implementation of the IEEE 802.11a/p [2][3] PHY on the BPE processor. The two standards share most of the receiver processing, but target two different scenarios: [3] targets indoor wireless communications, whereas [2] deals with outdoor vehicle to vehicle communications.

By properly exploiting the instruction set and the features of the BPE (instructions pipelining combined with parallel execution and multi-thread support), software real-time implementation has been developed.

We have shown that a balanced mixture of fine- and coarse-grain instructions allows to cover efficiently all the algorithms of the receiver chain. All the most computationally intensive kernels, including synchronization and FFT are performed in software; the only block left in hardware is the Viterbi channel decoder.

The data rate required by [3] can be achieved by adopting a three stage multi-thread processing, due to the absence of data dependencies (i.e. feed-back) between the computational blocks.

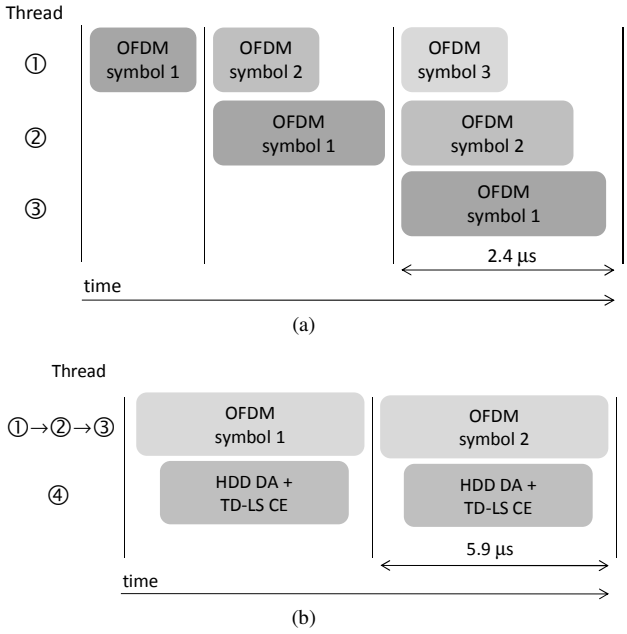

Fig. 4. Data field timing diagrams (MCS-7) based on multi-thread processing (timing not in scale): (a) 802.11a (b) 802.11p. The figure is labeled according to Fig. 3. Viterbi decoding is performed by a dedicated hardware design.

The requirements of [2] are satisfied by 2-stage multi-thread processing, but leaves little margin for optimization due to the need of recursively updating the channel tracking: the optional 20 MHz bandwidth mode, more challenging in terms of timing requirements, currently cannot be supported. It should be finally noted that a more advanced channel tracking algorithm, like Soft Decision Directed DA (based on Viterbi decoding) [19], would introduce an even higher latency, thus reducing the throughput.

In order to tackle these aspects, several further architectural enhancements are in place, like extension of the SIMD instruction width or the more future looking and challenging concept of cluster of processors.

6. REFERENCES

[1] IEEE Std 802.11 (ISO/IEC 8802-11: 1999): IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

[2] IEEE P802.11p-2010, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", Amendment 6: Wireless Access in Vehicular Environments, 2010.

[3] IEEE 802.11a-1999 Standard, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band", 1999.

[4] Kees van Berkel, Frank Heinle, Patrick P. E. Meuwissen, Kees Moerman, and Matthias Weiss, "Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices," EURASIP Journal on Applied Signal Processing, vol. 2005, no. 16, pp. 2613-2625, 2005. doi:10.1155/ASP.2005.2613

[5] B. Bougard, B. De Sutter, S. Rabou, D. Novo, O. Allam, S. Dupont, and L. Van der Perre, "A coarse-grained array based baseband processor for 100Mbps+ software defined radio". In Proceedings of the conference on Design, automation and test in Europe (DATE '08). ACM, New York, NY, USA, 716-721.

[6] D. Liu, A. Nilsson, E. Tell, D. Wu, and J. Eilert, "Bridging dream and reality: programmable baseband processors for software-defined radio". Comm. Mag. 47, 9 (September 2009), 134-140.

[7] V. Ramadurai, S. Jinturkar, S. Agarwal, M. Moudgill, and J. Glossner, "Software Implementation of 802.11a blocks on Sandblaster DSP", Proceedings of Software Defined Radio Technical Forum (SDR Forum '06), Orlando Florida, November, 2006.

[8] S. Eberli, A. Burg, T. Bösch and W. Fichtner, "An IEEE 802.11a baseband receiver implementation on an Application Specific Processor", Proceedings of IEEE Midwest Symposium on circuit & Systems, Montreal, Quebec, Canada, pp. 1324-1327, Aug 2007.

[9] A. Nilsson, E. Tell, D. Liu, "An 11 mm^2, 70mW Fully Programmable Baseband Processor for Mobile WiMAX and DVB-T/H in 0.12 µm CMOS", IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, USA, 90-97, 2009.

[10] D. Carona, A. Serrador, P. Mar, R. Abreu, N. Ferreira, T. Meireles, J. Matos and J. Lopes, "A 802.11p prototype implementation," Intelligent Vehicles Symposium (IV), 2010 IEEE , vol., no., pp.1116-1121, 21-24 June 2010.

[11] H. Harada, R. Funada, K. Sato, K. Iigusa and K. Li, "Research and development on UHF band inter-vehicle communication systems," Intelligent Transport Systems Telecommunications, (ITST), 2009 9th International Conference on , vol., no., pp.279-284, 20-22 Oct. 2009.

[12] T.M. Schmidl and D.C. Cox, "Robust frequency and timing synchronization for OFDM", IEEE Transactions on Communications, vol. 45, pp. 1613-1621, Dec. 1997.

[13] D. Lo Iacono, J. Zory, E. Messina, N. Piazzese, G. Saia, and A. Bettinelli. 2006. "ASIP architecture for multi-standard wireless terminals". In Proceedings of the conference on Design, automation and test in Europe: Designers' forum (DATE '06). European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 118-123.

[14] T. Cupaiuolo and D. Lo Iacono, "Software Implementation of Near-ML Soft-Output MIMO Detection," Washington, DC, USA, 30 November - 3 December, 2010, Software Defined Radio Forum 2010 (SDR'10).

[15] J. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, 1959

[16] F. Tosato, P. Bisaglia, "Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2", in: Proceedings of IEEE International Conference on Commun., New York, April/May 2002, pp. 664-668.

[17] E. Dall'Anese, A. Assalini, and S. Pupolin, "On reduced rank channel estimation and prediction for OFDM-based systems," in Proc. Int. Symp. on Wireless Pers. Multimedia Commun., Jaipur, India, Dec. 2007.

[18] M. Siti, A. Assalini, E. Dall'Anese and S. Pupolin, "Low Complexity Decision-Directed Channel Estimation based on a Reliable-Symbol Selection Strategy for OFDM Systems" (2010) IEEE International Conference on Communications (ICC'09) - Workshop on Vehicular Connectivity - Cape Town, South Africa, 23-27 May 20.

[19] L. Jarbot, "Combined decoding and channel estimation of OFDM systems in mobile radio networks," in Proc. IEEE Vehicular Tech. Conf., vol. 3, May 1997, pp. 1601–1604.