On Future Aeronautical Communications Standards: a Real-Time, Fully-Software AeroMACS waveform implementation based on the SCA-compliant OSSIE/USRP2 platform

Mario Di Dio, Daniele Bolognesi, Massimiliano Francone and Marco Luise
University of Pisa - Dip. Ingegneria dell'Informazione - Via Caruso, 16 - 56122 Pisa, Italy
{mario.didio, marco.luise}@iet.unipi.it; {bolognesi.daniele, francone.massimiliano}@gmail.com;

***Abstract*—This paper describes the architecture, implementation and computational results of a proof-of-concept, real-time, fully-software SCA-compliant AeroMACS waveform (PHY layer) based on ETTUS Research's Universal Software Radio Peripheral v2.0 (USRP2) platform. According to Software Defined Radio (SDR) paradigm, C++ software modules were developed from scratch by the authors in order to perform all the signal processing functions described in the AeroMACS standard. Software modules were integrated in a SDR development framework (OSSIE) compliant to the Software Communications Architecture (SCA) guidelines. The resulting waveform enjoys the portability and interoperability features which are characteristics of the SCA-compliant waveforms. The paper details the platform, the implementation technique and the computational performances of the waveform. The paper also outlines the design choices allowing real-time performance on an off-the-shelf personal computer and introduces the enabler technology of the project, the memory acceleration (MA) technique.**

***Index Terms*—Aeronautical Communications, AeroMACS, WiMax, OSSIE, Software Communications Architecture, Software Defined Radio, USRP, Baseband Processing.**

I. Introduction and motivation

THE air transportation market is expected to double by the 2025 and the current air traffic systems will not be capable to satisfy this growth. In particular, new security requirements are requested, for example, to efficiently move people and cargo. Focusing on communications issues, it is possible to identify some critical aspects to improve: pilots situation awareness, Airline Operation Center (AOC) data traffic capacity, passengers and cabin communications systems. The solution for these critical aspects is the convergence of protocols and interfaces towards a new open system that is the result of the collection of different communications technologies tailored for a specific operational setting. In this scenario, the key-concepts are the flexibility and the interoperability among the legacy communications systems and the new high-capacity communications standards. The challenge of this approach is to increase the capacity, security and efficiency of the aeronautical communications with no or small increase of the complexity and cost of the on-board equipments.

This has been also the objective of SANDRA (Seamless Aeronautical Networking though integration of Data links Radios and Antennas) [1], a research project financed by the European Commission and involving some academic research centres and some of the major players of the European telecommunications market. The final scope of the SANDRA project has been the integration at different levels, from antenna to the network layer, of several communications standards (analogue VHF, VDL2, B-GAN, AeroMACS) on a reconfigurable Integrated Modular Radio. The core of this project is the Software Defined Radio (SDR) approach in which reliable communications and interoperability among different standards, on the same reconfigurable hardware platform, can be easily provided by the Software Communications Architecture (SCA).

As known, SCA [2] is a non-proprietary, open architecture framework, created to solve the aris-

This work was in part supported by the European Commission in the framework of the FP7 project SANDRA (contract n. 233679)

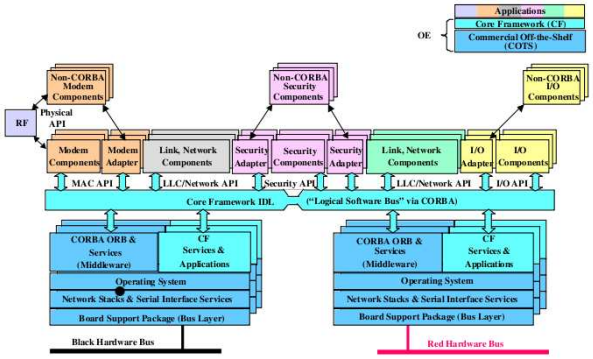

Fig. 1. Software architecture of a SCA-compliant waveform.

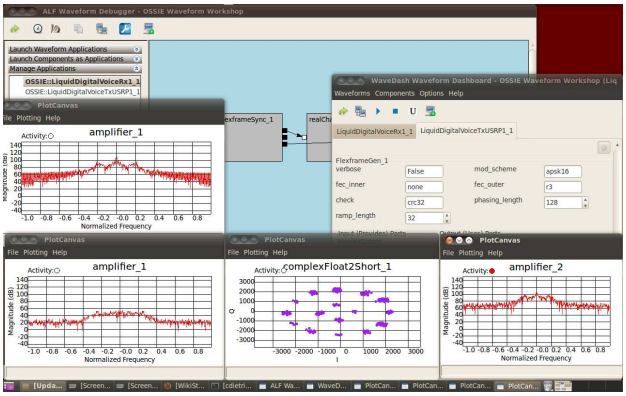

Fig. 2. OSSIE SDR development framework.

ing issues, especially in the military field, of the development of a programmable, modular, multi-mode radio. The SCA provides complete interoperability among the software defined radios (SDRs) implementing different radio communications standards (that are called waveforms in the SCA parlance), and high portability among heterogeneous hardware platforms (GPP, DSP, FPGA, etc). SCA is not a system specification, as it is intended to be implementation-independent, but rather a set of design constraints (Fig. 1). To sum up, the main advantages of this approach are: the greater portability of source/object code; the full interoperability among the SCA-compliant SDRs and the easy upgrade to further standard evolution.

This work can be considered as the joint result of the analysis of AeroMACS waveform, the WiMAX IEEE 802.16e standard [3] for ATS/AOC communications studied in the SANDRA project, and the SDR background of the DSPCoLa, University of Pisa, in implementing fully-software SCA-compliant waveforms [4]. The paper describes the implementation of a real-time, fully-software SCA-compliant AeroMACS waveform (PHY layer) [5]. As indicated in the SESAR/FCI recommendations, AeroMACS will provide the airport connectivity in the near future. Our implementation is based on the well-known Universal Software Radio Peripheral v2.0 (USRP2) hardware [6]. Specifically, the USRP2 provides the digital to analogue conversion and baseband-to-RF conversion at the transmitting side and the RF-to-baseband conversion including analogue to digital conversion at the receiving side. All baseband functions, except for the FFT block which is based on a standard routine [7], are implemented through an efficient C++ code that was developed from scratch by the authors.

II. The OSSIE/USRP2 SDR platform

This section describes the SW/HW platform on which the AeroMACS waveform was implemented. In particular, both the transmitter and receiver signal processing blocks were developed on an open-source SW/HW platform that we will shortly describe.

A. SW Section

Developed C++ software modules were integrated in a open-source SDR framework, called OSSIE [8]. It was developed at Wireless@Virginia Tech and represents a SW environment for implementing SCA-compliant waveforms (Fig. 2). The software package includes:

- an SDR core framework based on SCA;
- the Waveform Workshop, a set of tools for rapid development of SDR components and waveforms applications;
- an evolving library of pre-built components and waveforms.

This SDR framework is conceived to give to the developer a valid development environment and hide the complexity related to the implementation of the SCA hierarchical software architecture (Fig. 1). Unlike other SDR frameworks like GNURadio, signal processing blocks are written in pure C++ while the interconnection among such blocks is provided, as defined in the SCA standard [2], by means of CORBA (Common Object Request Broker Architecture) interfaces and *ports* communication. Communication from/to the USRP2 is obtained by using the SW interfaces of a module called *USRP_Commander* that includes the *libusrp2* external C++ library. During this work, we introduced also some ad-hoc modifications to the OSSIE primitives that manage communications from/to the USRP2 in order to fully exploit the functionalities of the selected HW platform.

B. HW section

The peripheral used to transmit/acquire the signal in this project is the well-known USRP2 system, which was developed by Matt Ettus [6]. The USRP2 is a board with open design and drivers. It consists of:

- 4 ADC at 100 MSamples/s with a resolution of 14 bit;
- 4 DAC at 400 MSamples/s with a resolution of 16 bit;
- Digital downconverters/upconverters with programmable decimation/interpolation rates;
- 1 Gigabit Ethernet interface;
- 2 Gbps high-speed serial interface for expansion;
- extension sockets to connect a wide variety of RF daughterboards;

Daughterboards are responsible for RF up/downconversion. In order to cover as many frequency bands as possible, several daughterboards were developed: receivers, transmitters or transceivers do exist which collectively cover the RF spectrum from DC to about 5 GHz. The RF front-end used in this project is the transceiver daughterboard XCVR2450 which operates from 4.9 MHz to 5.2 GHz. Complex-valued I-Q samples are sent over a Gigabit Ethernet interface (interleaved real/imaginary parts, both represented as a signed *short int* on 2 bytes). The maximum sampling frequency sustained by the Gigabit Ethernet interface is 25 MSamples/ s corresponding to 800 Mb/s over the interface, with which a full spectral window of 25 MHz can be processed with no loss.

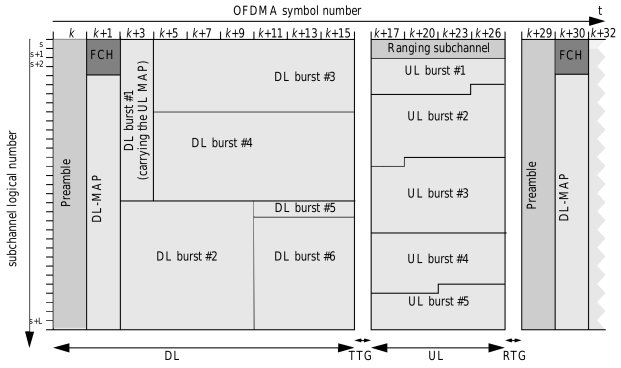

Fig. 3. AeroMACS OFDMA frame in TDD mode.

III. The AeroMACS standard

The AeroMACS standard [5] is based on the IEEE 802.16 - 2009 standard [3], selecting from this standard the parameters suitable for ATC and AOC communication in the airport surface environment. As

Parameter	*Required value*
Center frequency	5.091-5.150 GHz
System profile	OFDMA
Duplexing mode	TDD
Transmission Bandwidth	5 MHz
FFT size	512
Sampling factor	28/25
Sampling frequency	5.6 MSamples/s
Frame length	5 ms
OFDM symbol duration without guard interval	91.4 us
Guard interval	11.42 us
OFDM symbol duration with guard interval	102.84 us
Subcarrier spacing	10.94 kHz
# symbols per frame	51
Net useful bit-rate	12.25 Mb/s

TABLE I
AeroMACS waveform system parameters

IEEE 802.16, AeroMACS is based on a orthogonal frequency-division multiple-access (OFDMA) system employing a Time Division Duplexing (TDD) protocol. There are two possible transmission mode: 5 MHz and 10 MHz. In this work we implemented the 5 MHz mode. The most important system parameters of the implemented waveform are listed in the following table.

Fig.3 shows the time-frequency logical structure of an OFDMA AeroMACS frame in TDD mode. Each frame contains several logical data region belonging to the different users associated to the cell. So it is necessary to define a hierarchical grouping of the subcarriers in order to identify the resources allocated to the users.

The active subcarriers (carrying data or pilot) are divided into physical clusters containing 14 adjacent subcarriers over 2 consecutive symbols. In each symbol of this cluster 12 subcarriers are allocated for data transmission, while the remaining 2 are used as pilots. The physical clusters are renumbered into logical clusters and allocated to the active users. The subchannel, that is the minimum frequency-time resource unit, is composed by two clusters for a total of 48 subcarriers. Clusters allocated to a specific user are typically not adjacent to each other in the frequency domain and the spaces between them are not regular. For this reason, it is necessary to perform channel estimation on a cluster-by-cluster basis. In fact, channel knowledge on a given cluster does not provide any information about the channel realization over the other

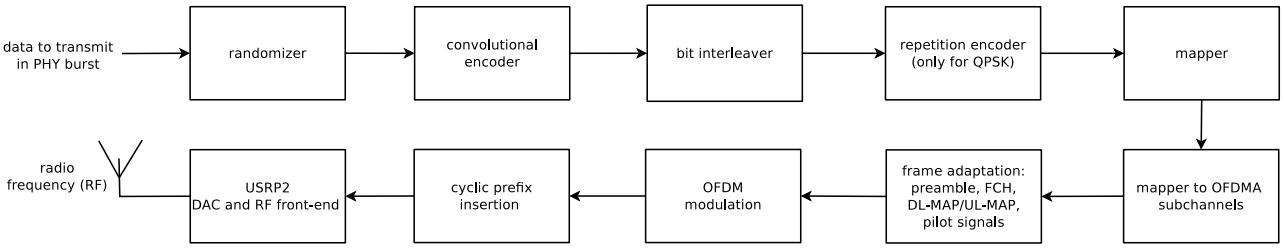

Fig. 4. Block scheme of the implemented transmitter signal processing blocks.

clusters assigned to the same user because they are normally located in different frequency positions.

IV. The AeroMACS waveform implementation

A. Transmitter signal processing blocks

Fig.4 depicts the constituent signal processing functions on the transmitter side. All of them, except for the FFT block which is based on an external standard routine [7], were implemented in order to optimize the usage of the computational resources and to reach real-time performance. The implemented chain is described as follows:

- the *randomizer* removes time domain correlation in the binary data to transmit by performing a XOR operation with a pseudo-random binary sequence (PRBS) generated via a generator polynomial and a feedback shift register (FSR);
- the *convolutional encoder* performs convolutional encoding with rate $1/2$, constraint length $K = 7$ and generator polynomial $g_1 = 177$ and $g_2 = 133$. The shift registers are initialized with the last seven bits of a Forward Error Correction (FEC) block, so realizing a FEC block-independent encoding. This technique is named *tail biting*;
- the *bit interleaver* prevents bits from original bitstream from being always associated with the same OFDM subcarriers that can offer, in general, a low signal-to-noise ratio (SNR);
- the *repetition encoder* block performs a repetition encoding only if the used constellation is the QPSK, as in the case of the FCH reference signal;
- the *mapper* block groups encoded interleaved bits and maps them into the constellation symbols (BPSK, QPSK, 16-QAM and 64-QAM);
- the *mapper to OFDMA subchannels* arranges the symbols in the OFDMA subchannels according to the allocation scheme fixed by the MAC layer (Fig.3);
- the *frame adaptation* block inserts in the OFDMA burst the reference signals (boosted pilot subcarriers, preamble, ...) necessary to synchronization steps (time/frequency) at the receiving side;
- the *OFDM modulation* block inserts the 92 virtual subcarriers and performs an IFFT operation with *N = 512* points;
- the *cyclic prefix insertion* block inserts the guard interval in order to annul the time dispersion of the channel and avoid the Inter-Symbol Interference (ISI) and the Inter-Block Interference (IBI);
- the *USRP2* performs the digital-to-analogue conversion by interpolating the digital complex samples and then shifts the analogue baseband I/Q signal to the radio-frequency channel defined in the AeroMACS standard (5091-5150 MHz).

B. Receiver signal processing blocks

In this section we describe the constituent blocks of the receiving section that pairs with the transmitter signal processing functions described in IV-A. As shown in Fig.5, the receiver chain can be divided in two sub-chains: the *synchronization and channel estimation/equalization* chain and the *channel decoding* subsystem.

The two subchains can be described as follows:

- the *USRP2* provides to the system the RF front-end/down-conversion to baseband and the analogue-to-digital conversion. The resulting signal is a stream of complex interleaved short samples;
- the *coarse timing offset estimation and correction* is performed by exploiting the characteristic frequency shape of the preamble. In fact, the preamble is transmitted by using *N/3* equispaced subcarriers while the remaining ones are forced to be null, so that two segments of the resulting signal in the time-domain with length *N/3* are highly-correlated. After the estimation is possible to identify and align to the start of the PHY

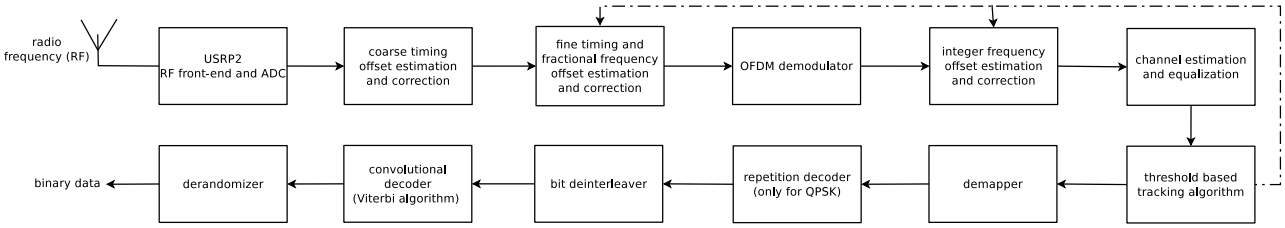

Fig. 5. Block scheme of the implemented receiver signal processing blocks.

data burst;

- the *fine timing and fractional frequency offset estimation and correction* is based on the algorithm described in [9]. This algorithm operates in the time domain and implements a Maximum Likelihood (ML) open-loop timing and fractional (i.e a fraction of the intercarrier spacing) frequency offset estimation by exploiting the inner redundancy contained in OFDM cyclic prefix. In our work we adopted a modified version of this algorithm that uses averaged realizations of the log-likelihood function. After the estimation is possible to align to the first useful OFDM symbol and to correct in the time-domain the fractional frequency offset;
- the *OFDM demodulator* removes the cyclic prefix, performs a FFT operation with 512 points, acting as a matched filter for the OFDM modulation, and removes the virtual subcarriers;
- the *integer frequency offset estimation and correction* block jointly estimates the integer, i.e. multiple of intercarrier spacing, frequency offset and identify the training preamble. The algorithm maximizes a correlation function which depends on the frequency offset and the training preamble;
- the *channel estimation and equalization* block estimates channel frequency response by interpolating information carried by boosted pilot subcarriers and equalizes data subcarriers using the calculated channel profile and the Zero-Forcing (ZF) technique;
- the *threshold based tracking algorithm* is a finite-state machine that controls the renewal of timing and frequency offset estimations. It monitors the power level of the central subcarrier (DC) that, as stated in the standard, should be zero, and if the power goes over a threshold orders the renewal of the timing and frequency offset estimations;
- the *demapper* demodulates the received symbol constellation (BPSK, QPSK, 16-QAM or 64-QAM) into an encoded bit stream according to the used constellation;
- the *repetition decoder* block is enabled only for QPSK modulation and provides the decoded bitstream based on the classical majority decision-maker;
- the *bit deinterleaver* performs a bit-level operation in order to recover the right bit order after shuffling introduced by the bit interleaver on the TX side;
- the *convolutional decoder* performs the Viterbi decoding algorithm, the most famous method to decode a convolutional code. It is the computationally-heaviest block within the receiver chain having to work at a bitrate of 12.5 Mbps (measured after decoding);
- the *derandomizer* descrambles data stream by applying a XOR operation with a PRBS sequence.

V. Computational results

At the time of writing the implemented waveform is being tested and validated at DSPCoLa, University of Pisa, Italy. As stated in section IV-A, all the signal processing is performed by an off-the-shelf PC equipped with an Intel Core i7 2670QM processor (2.2 GHz), Ubuntu 11.10 Operating System (OS) and OSSIE 0.8.2.

As stated in section III, AeroMACS signal has a baud rate of 5.6 MSamples/s on a 5 MHz channel bandwidth, cyclic prefix length of 1/8, 64-QAM constellation for the user data, 1/2 convolutional coding rate, all yielding a useful bitrate of 12.5 Mbps. AeroMACS waveform proved able to correctly modulate and demodulate the signal in real-time with a maximum throughput of ≈16 Mbps at the transmission side and ≈13 Mbps at the reception side and absorbing less than two of the eight (virtual) processors available on the i7-2670QM (25% of total computational power). It is clear that, the use of a multithreaded software architecture, ie. the parallel execution of the diverse functional blocks, can increase

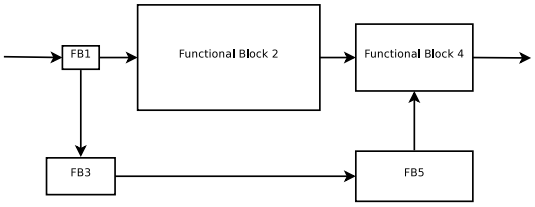

Fig. 6. Computational cost representation of the functional blocks.

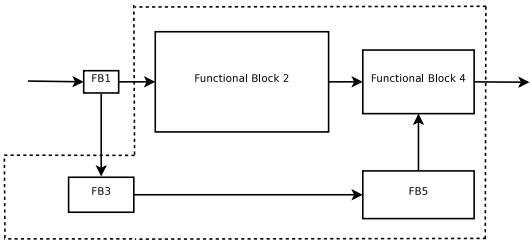

Fig. 7. Table boundary in RTAR procedure.

less than linearly the total throughput and equally distribute the load among the available cores. Nevertheless, the threads synchronization on a General Purpose Processor (GPP) can result in a more complicated and abstract software architecture. For this reason, in order to match the real-time constraints as fast as possible, we first profiled the processing blocks and then we implemented the waveform with the minimum number of threads.

Unfortunately at this step, it is very difficult to validate the implemented waveform with commercial devices because we implemented only the PHY layer without any MAC primitives. For this reason, we are planning to implement basic MAC primitives implementing some basic operations like the network identification and association.

VI. Memory acceleration within AeroMACS waveform

Reaching real-time performance described in Sec.V could not be possible without an extensive usage of a SDR implementation technique developed by the authors and called Memory Acceleration (MA).

The rationale behind this technique is the observation that, on machines equipped with GPPs, memory resources (RAM) are usually abundant, cheap, and not power-hungry if compared to computing cores. MA belongs to the broader class of optimizations known in computer science literature under the collective denomination of space/time trade-off. According to this approach, a classic SDR system is re-arrange in a memory-aware fashion by dividing it into several component functional sub-blocks which are then conveniently re-aggregated and implemented through memory tables. The instruments to perform this redesign are two algorithmic procedures called respectively the Algorithm Segmentation (AS) and the Recursive Table Aggregation Rule (RTAR).

In order to briefly explain the usage of these techniques, imagine to divide a whole waveform in several functional blocks. This 0-step of the AS procedure may be the direct translation of the signal processing blocks described as atomic, ie. independent, in the waveform standard. On the contrary, the aim of AS is providing a decomposition of each functional block $f_n(...)$, formerly assumed atomic, into a chain of inner sub-blocks $f_{n,p}(...)$ whose end-to-end behaviour is equivalent to the original function $f_n(...)$. AS is not an algorithm redesign and, as a consequence, it does not change the overall computational cost of the segmented algorithm. If we called W_n the computational cost of the *n-th* functional block, we can give a visual representation a SDR signal processing chain in which the size of each functional block $f_n(...)$ is proportional to its computational cost W_n (Fig.6). At each functional block is associated a weight coefficient W_{Mm} representing the total computational cost of memory management for table $t_m(...)$, i.e. the cost of memory address calculation and the memory access latency if that block was implemented using the table $t_m(...)$.

Recursive Table Aggregation Rule (RTAR) tries to find the optimum trade-off between granularity and aggregated blocks memory implementation by means of iterative segmentation and re-aggregation operations. In fact, if we consider that a memory implementation of a functional block requires at least a look-up action to provide the pre-computed output, it is necessary to minimize W_{Mm} aggregating as many functional blocks $f_n(...)$ as possible. On the contrary, we have to take in account the limitation of the memory resources. Furthermore, we have to preserve the cache-friendliness of the tables, i.e. the memory-contiguity of information that is used contiguously in time. A key-concept of RTAR is the table boundary (TB), that is the closed line delimiting the subsystem we intend to memory-accelerate at a certain iteration of RTAR (Fig.7). The connection arrows crossing the TB provide the interfaces from and towards the remainder of the system. A block is called *peripheral* if all of its input or all of its output connections cross the TB. RTAR works as a finite state machine (FSM) in which functional blocks are recursively released and

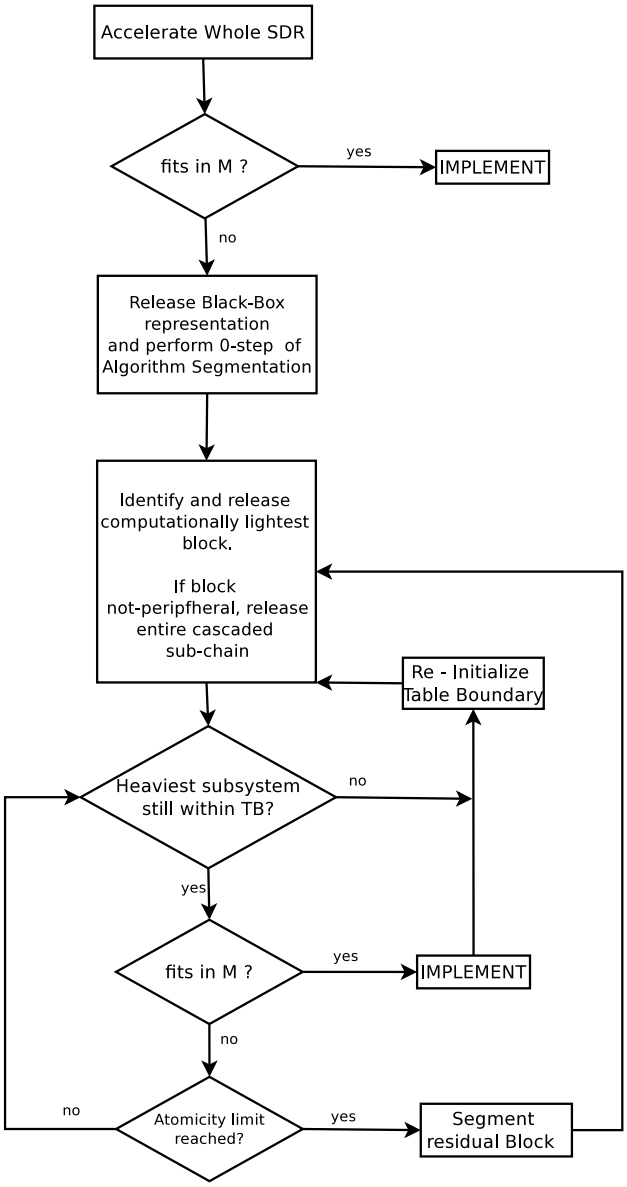

Fig. 8. FSM implementing RTAR.

included in the TB in order to maximize the memory-usage and minimize the overall computational cost releasing step by step only the lightest block within the TB. Fig.8 shows the FSM implementing the RTAR procedure. Both AS and RTAR procedures are extensively described in [10] together with the most significant performance results.

Within this work, MA was applied to the two computationally-heaviest functional blocks of the receiving chain: the timing and fractional frequency offset estimation algorithm described in [9] and the $K=7$ Viterbi decoder. At the time of writing, the acceleration factor for both the algorithms was greater than one order of magnitude wrt a previous MA-free but computationally optimized version of such algorithms. We believe that further improvements for our MA-based algorithms can be possible and will be the object of further research.

VII. Conclusions and perspectives

This paper describes the implementation and the computational performance of a real-time, fully-software, SCA-compliant AeroMACS waveform based on the USRP2/OSSIE platform.

Computational performance achieved by the AeroMACS waveform has to be intended intended as the proof-of-concept of feasibility of fully-software, GPP-based modulation/demodulation of even high throughput communication standards such as AeroMACS. Real-time performance was only possible via the extensive usage of the MA technology that we briefly described in section VI and in [10], and that we consider as the enabler technology of this work.

We believe that the realization of this SCA-compliant AeroMACS waveform with good computational efficiency can be considered as the first step towards the realization of a low-cost, high-efficiency, flexible and reconfigurable SDR node in which heterogeneous aeronautical communications standards can easily coexist.

References

[1] SANDRA project. http://www.sandra.aero
[2] SCA 2.2.2 *[Online]. Available: http://sca.jpeojtrs.mil/*
[3] WiMax 802.16e. *[Online]. Available:http://standards.ieee.org.*
[4] M. Di Dio, A. Morelli, M. Luise,"A Real-Time, Fully-Software VHF aeronautical tx/rx waveform based on the SCA-compliant OSSIE/USRP2 platform", in Proc. Karlsruhe Workshop on Software Radios (WSR), Karlsruhe, Germany, Mar. 2012
[5] AeroMACS. *[Online]. Available:http://www.eurocontrol.int.*
[6] Ettus Research LLC website, *[Online]. Available: http://www.ettus.com.*
[7] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT", in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, WA, May 1998, pp. 1381–1384.
[8] OSSIE, SCA-Based Open Source Software Defined Radio *[Online]. Available: http://ossie.wireless.vt.edu/*
[9] JJ. van de Beek, M. Sandell and P. O. Borjesson, "ML Estimation of Time and Frequency Offset in OFDM systems", *IEEE Transactions on signal processing* 45(7), 1800-1805 (1997).
[10] V. Pellegrini, M. Di Dio, L. Rose, M. Luise,"On the computation/memory trade-off in software defined radio", in Proc. IEEE Global Telecommunications Conference (GLOBECOM), Miami, FL, USA, Dec. 2010.