A Split Architecture for Random Access MAC for SDR Platforms

Paolo Di Francesco*, Séamas McGettrick*, Uchenna K. Anyanwu‡, J. Colman O'Sullivan*,
Allen B. MacKenzie‡ and Luiz A. DaSilva*‡

*CTVR / The Telecommunications Research Centre, Trinity College Dublin, Ireland
‡Wireless @ Virginia Tech, Blacksburg, Virginia, USA
Email: pdifranc@tcd.ie, smcgettr@tcd.ie, uchevt@vt.edu, cosull13@tcd.ie, mackenab@vt.edu, dasilval@tcd.ie

***Abstract*—Implementation of carrier-sensing-based medium access control (MAC) protocols on inexpensive reconfigurable radio platforms has proven challenging due to long and unpredictable delays associated with both signal processing on a general purpose processor (GPP) and the interface between the RF front-end and the GPP. This paper describes the development and implementation of a split-functionality architecture for a contention-based carrier-sensing MAC, in which some of the functions reside on an FPGA (field programmable gate array) and others reside in the GPP. We provide an FPGA-based implementation of a carrier sensing block and develop two versions of a CSMA MAC protocol based upon this block. We experimentally test the performance of the resulting protocols in a multihop environment in terms of end-to-end throughput and required frame retransmissions. We cross-validate these results with a network simulator with modules modified to reflect the mean and variance of delays measured in components of the real software-defined radio system.**

I. Introduction

The term Cognitive Radio (CR) applies to a wide variety of systems, ranging from transceivers that use simple techniques to gather information about the wireless environment and have basic decision-making capabilities, to systems that have multi-sensory features and are capable of sophisticated analysis, learning, and decision-making. Cognitive radios can be viewed as radio nodes that are aware of the context in which they are operating and can reconfigure themselves to best fit the current conditions in the medium.

A cognitive radio should be able to sense activity in the channel in which it is operating and reconfigure itself based on the results. To achieve this reconfiguration, cognitive radios are often built on software-defined radios (SDRs). Flexible SDR platforms have been incorporated into a wide range of wireless communications technology, spanning from satellite communications to sensor networks. They allow the reconfiguration of waveforms, frequency, and modulation schemes in order to improve communication performance. Traditional SDR platforms are capable of performing most, if not all, of their signal processing tasks on a general purpose processor (GPP). The hardware radio front-end employed usually performs only minimal tasks. The combination of minimal radio hardware and appropriate software packages (e.g. GNU Radio, Iris, Sora) offers great opportunities for researchers to carry out Cognitive Radio experiments at a relatively low cost.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 258301 (CREW project).

This research also received support from the Science Foundation Ireland under Grant no. 10/IN.1/I3007 and from a Bradley Fellowship from Virginia Techs Bradley Department of Electrical and Computer Engineering and made possible by an endowment from the Harry Lynde Bradley Foundation.

Having cognitive radio nodes reconfigurable via software makes it possible to analyze the radio environment and to adjust the system parameters to a particular operational situation quickly and without redesigning hardware. However, the flexibility achieved with the software introduces high delays due to the nature of the off-the-shelf computer processing the radio waveforms. To date, SDR and CR experiments have primarily focused on environments where a single link is established or, at most, two competing links coexist in an interference channel. Higher layer issues have been mostly neglected or analyzed using unrealistic assumptions due to the limitations introduced by the aforementioned delays.

Enabling networking experimentation on affordable SDR platforms presents a challenge at the Medium Access Control (MAC) layer, where the issues in implementing protocols with reasonable performance have been well summarized in the literature [1]–[3], and were alluded to in our earlier work [4].

In this paper we propose a new architecture for a CSMA MAC, with MAC functionality split between an FPGA (field programmable gate array) and the GPP. This approach provides an improved Rx/Tx turnaround response time for MAC implementations. We carry out time-critical functions in the FPGA, while non-time-critical functions remain implemented in software running on a GPP [3]. This work makes the following contributions:

- We propose an architecture for a CSMA MAC on an RF front-end with limited computing capabilities.
- We provide a split-functionality implementation of a random access MAC where some of the functions reside on an FPGA and others reside in the GPP.
- We describe a carrier-sense block in an FPGA, closer to the RF front-end. While we test this functionality within a set of radios running Iris, the implementation of carrier sensing on the FPGA is agnostic to the choice of software platforms.
- We cross-validate all results between prototype-based experimentation and network simulations using OMNeT++.

This paper is structured as follows. In section II, we

CROWNCOM 2013, July 08-10, Washington DC, United States
Copyright © 2013 ICST
DOI 10.4108/icst.crowncom.2013.252050

describe the proposed architecture, whose functionality is split between an FPGA and a GPP. In section III, we illustrate the implementation of the different functions. Section IV presents the tests and results of our CSMA implementations. Finally, section V summarizes our main conclusions and discusses areas for future work.

II. Random Access MAC Protocols for SDR Platforms

There are a number of considerations to take into account when designing a MAC protocol for cognitive radio nodes on SDR. A designer must first consider the needs of the functions at the physical (PHY) and MAC layers, and the limitations of the available hardware platforms. With this information the designer can make more informed decisions about how to implement the radio protocol stack. In this section we do this by first looking at the requirements of the radio functions to implement an effective MAC. We then discuss the limitations of the available platforms. Finally, we propose our solution to these limitations and describe the proposed radio.

A. Radio Considerations

In this paper we are particularly interested in contention-based MAC protocols. These protocols often use a carrier-sense mechanism, which is a fundamental part of most wireless networking stacks (e.g. in wireless LAN and sensor networks), to detect other transmissions. The implementation of a CSMA MAC radio node can be roughly split into six main functions. These are clear-channel assessment (CCA), backoff, mod/demodulation, frame recognition, retransmission, and ACKs transmission. Each of these functions has different requirements with regards to latency, computational complexity, flexibility, and order of execution. These requirements are summarized in Table I.

From Table I we observe that the CCA requires low latency. Thus the channel assessment is best placed as close to the radio front end as possible to ensure its low latency. The backoff can be tightly coupled to the carrier sense unit since it requires data from the carrier sensing unit to function.

TABLE I
Radio function requirements for a CSMA MAC protocol

Function	Latency	Complexity	Flexibility	Coupled
CCA	Low	Low	Low	-
Backoff	High	Low	Low	Carrier Sense
Mod/Demod	Low	High	High	-
Frame Recognition	Low	Low	High	Mod/Demod
Re-Tx	High	Low	Low	Mod/Demod
ACKs	Low	Low	Low	Mod/Demod

Likewise we can determine that the mod/demodulation should also be placed as close as possible to the radio front-end to ensure low latency. The remaining three modules cannot be implemented before the mod/demodulation unit and as such should remain tightly coupled to that module.

The mod/demodulation module has one further consideration that should be taken into account. The mod/demodulation has high computation complexity which means it might be suited to hardware implementation. However, the module also requires a large amount of flexibility to allow it to be used in many different configurations, which is not conducive to hardware implementation. A tradeoff will need to be made between the flexibility of the module and the throughput the unit will be able to achieve. The result of this tradeoff is largely determined by the hardware platform used.

B. Platform Considerations

SDR platforms used by research groups for cognitive radio experimentations usually rely on inexpensive radio front-ends connected to a PC, where most of the radio chain actually runs in software in a GPP. This configuration allows the radio node to be inexpensive and extremely flexible. This low cost and high flexibility enables large and complex cognitive radio experiments. The USRP [5] is the most popular example of a minimal RF front-end that relies on a PC for baseband signal processing.

There may be significant and variable latency between the processing elements on which signal processing occurs and the physical radio front-end. This leads to unpredictable performance of the MAC protocol. It particularly affects the turnaround time, which results from the combination of MAC layer frame processing time, non-negligible latencies associated with sending samples from the front-end to the GPP and from the GPP to the RF front-end, and the time taken to mod/demodulate frames. It also leads to difficulty in predictably scheduling frames for transmission by the SDR. Furthermore, it results in a *blind-spot* [1] in which assessments of the channel state may be stale due to signal processing delay, and hence useless. This makes implementation of a reliable CCA mechanism difficult.

C. Proposed Solution

In the previous two sections we have looked at the requirements of a CSMA MAC radio node and the limitations of platforms that use a minimal RF front-end. The large latency introduced by the use of a minimal RF front-end makes it difficult to achieve the low latency requirements of the carrier sense module in the radio node. To circumvent this problem we propose and implement a split-functionality hardware-software architecture for MAC protocols, judiciously placing signal processing functions over multiple computing platforms, such as an FPGA and a GPP. Many radio front-ends like the USRP contain an FPGA which is used to route data and for some basic DSP operations. These FPGAs often have unused logic available and we have used this logic to place components adjacent to the radio front-end. Although much of this discussion will be focused on the USRP radio front-end, the experience gained can be applied to any minimal RF front-end with an integrated FPGA.

Since the performance of the carrier sensing CCA is crucial to the MAC we should place it adjacent to the radio hardware. Moreover, the radio hardware must be able to store the transmitting frames in a buffer, and, based on the CCA result, quickly deactivate the carrier-sensing and transmit the frame

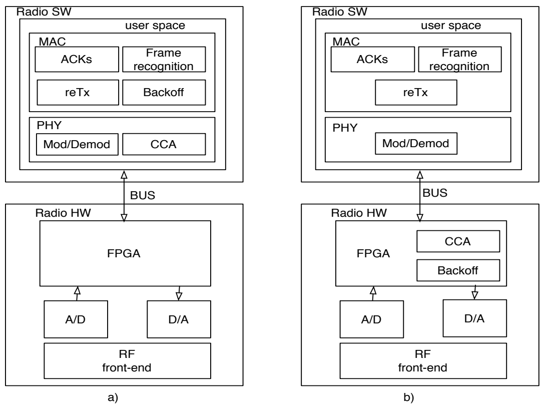

Fig. 1. a) Full Software architecture for a typical Radio Hardware (FPGA) + Radio Software (host user space) configuration. b) Split Functionality architecture proposed.

without waiting for further communications from the host. Without this storing system, a CCA in hardware would lead to little gain in terms of delay reduction.

Another important function in contention-based MACs is the backoff. This function is used to schedule a transmission on a random basis when multiple nodes try to access the medium simultaneously. It attempts to avoid two or more nodes accessing the channel at the same time when the channel is sensed free. Since the backoff requires communication with the CCA it should be also implemented on the FPGA. In this work we have not implemented the backoff on the FPGA; however, this module is currently under development.

An argument could be made to move the mod/demodulation modules to the FPGA since they exhibit high computational complexity and the overall throughput of the system is limited by these units. We opted to keep these units in software, as FPGA space is limited and we did not wish to be confined to a limited set of mod/demodulation schemes.

Other functions, such as frame recognition, retransmission, and ACKs transmission remain on the GPP. These functions require fully demodulated incoming frames, and since we have opted to leave the demodulation on the GPP, the natural place for these functions is to remain on the host.

In Figure 1 we depict how the proposed split functionality architecture would look in a typical FPGA+GPP configuration.

III. Implementation

In this section, we present the system designs for both the MAC implementation on the GPP and the carrier sense on the FPGA. We have selected the USRP E100 as our hardware platform based on available FPGA space, programmability, ease of use, and cost. However, this design can be easily extended to other platforms using the USRP + GPP configuration (e.g. USRP N210+PC).

We have used Iris [6] as the software platform for our proof-of-concept implementation. It interfaces with Ettus USRP RF front-end hardware, to allow for affordable experimentation. It should be pointed out, however, that the on-FPGA element is agnostic of the choice of attached software platform. It would

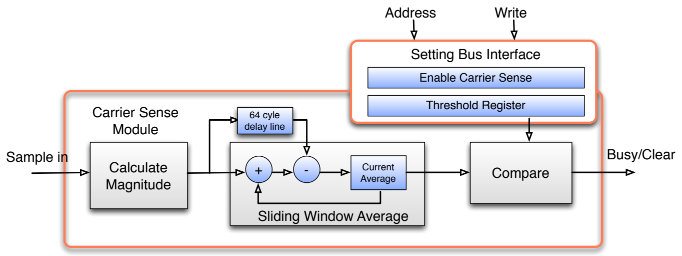

Fig. 2. Block Diagram of the proposed carrier sense module on E100 USRP

be usable, for instance, in a GNU Radio environment, as it preserves all UHD (USRP Hardware Driver) semantics.

A. USRP E100 Overview

The USRP E100 is an embedded standalone software-defined radio platform. The radio front-end is connected directly to the FPGA and all data transmitted from the GPP or received by the radio front-end must pass through the FPGA. The FPGA is therefore ideally placed to implement latency-sensitive processes, like sensing whether a channel is occupied and controlling transmissions to avoid collisions. To implement this system it is necessary to integrate customized FPGA blocks into the existing FPGA software design.

The FPGA uses the VITA radio transport (VRT) protocol to communicate with the ARM processor. VITA was developed to provide interoperability between diverse SDR components by defining a transport protocol to convey digitized signal data and receiver settings [7].

Our carrier sense implementation in hardware needed to be tightly coupled with the VITA hardware in the FPGA to access RX samples before they are sent to the host and to control the TX stream's access to the channel. Both functions are required by an FPGA carrier sense implementation. Therefore it was necessary to tap into the in-phase and quadrature (I/Q) samples at the VITA control block in the receive path. Moreover, the rate at which these samples are produced at this point in the chain is equal to the rate requested by the user, i.e down-conversion has already taken place, which is useful for debugging receiver issues. The I/Q samples are sent to the carrier sense block, which in turn produces a signal indicating whether a carrier is present.

B. Carrier Sense MAC Design

The Carrier Sense block in the FPGA relies on a simple energy-detection strategy. The block computes the average signal power for a set of samples and then compares this power with a programmable threshold value. If the received signal strength is greater than the threshold, then a signal indicating the carrier is present is sent to the VITA TX controller. The state machine in the VITA TX controller has been modified to ensure that the carrier present signal is low before sending frames to the radio front-end. In this way, our carrier sense module ensures the channel is free before transmitting. The details of carrier sense block implementation will be discussed in the next section.

TABLE II
CARRIER SENSE MODULE UTILIZATION IN XILINX FPGA (W/O CS IN PARENTHESES)

Logic Utilization	Total	Available	Utilization
Total No. Slice Registers	16,245 (15,007)	33,280	48% (45%)
No. of occupied Slices	13,464 (12,068)	16,640	80% (72%)
Total No. of 4 input LUTs	22,461 (20,901)	33,280	67% (62%)

1) Carrier Sense module: A block diagram of the carrier sense module as implemented on the E100 FPGA is shown in Figure 2. The data path can be split into four separate processes. These processes are the control registers, the magnitude calculation, the sliding window average, and the threshold compare. The control registers are used to enable/disable the carrier sense unit and to set the threshold value for the carrier sense unit. These registers are connected to the UHD settings bus and can be written to from software through the UHD library. When the carrier sense module implemented is disabled, the FPGA image works exactly like the pre-installed FPGA image on the USRP, and thus the carrier sense module can permanently remain on the FPGA without interfering with non-carrier sensing experiments.

The remaining three hardware components implement the sliding window carrier sensing module. Data is collected from the receive chain and the I/Q values are fed to the magnitude calculation hardware. The magnitude value is added to the current average and simultaneously placed at the top of a delay queue. The data on the bottom of the delay queue is subtracted from the current average as this value is exiting the sliding window. Once these two operations are complete the current value register is updated with the new sliding window average. In this implementation the sliding window averages over 64 baseband samples. This number is currently fixed at build time but it can be easily changed to suit a particular application.

Finally the average of the samples is compared with a programmable threshold value and the Busy/Clear signal is set accordingly. If the current average is above the threshold, then the data path asserts the Busy/Clear signal, which instructs the TX path in the FPGA to not transmit until the channel is free. If the current average is below the threshold, then the TX path is free to send frames.

Table II is a summary of the FPGA resource utilization with both the basic UHD modules supplied with the USRP and the proposed carrier sense module. The table also shows, in parentheses, the FPGA resource utilization of only the UHD hardware as supplied on the E100. 72% of the total slices, which consist of registers and look-up tables, are used by the carrier sense and the supplied modules, which means there is still considerable space available in the FPGA fabric to extend our carrier sense module or provide other functions.

Since we implemented the carrier sense module on the FPGA, the carrier sense mechanism works independently of our MAC implementation in software. So, when frames are sent from the MAC layer in Iris, the MAC has no control over what is done in the FPGA. There are several advantages associated with implementing carrier sense in the FPGA. Firstly, the FPGA-based carrier sense module can be used with

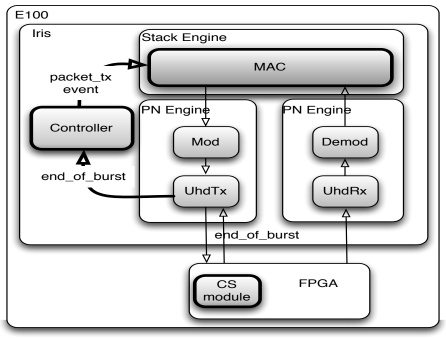

Fig. 3. Iris node - software implementation to allow carrier sensing feedback to the MAC.

any software defined radio, as it does not require any specialized software modules. Secondly, the carrier blind spot [1] is significantly decreased, since the latency from sensing to sending frames is greatly reduced, as compared as to a full software implementation of CSMA. This decreases the chance of collisions with other nodes' transmissions.

2) Software Implementation: Spreading the MAC between the FPGA and software implementation has some drawbacks, as the software MAC does not know whether a frame has been sent or whether the frame is just waiting in the TX buffer. The original MAC protocol used in this work and described in [4] requires knowledge of when a frame is sent across the network. In the software-only version of the protocol it could be assumed that the frame would be sent immediately. The time spent forwarding of the frame through the FPGA logic is relatively fixed due to the deterministic nature of FPGA signal processing. It is necessary for the MAC to estimate this time in order to start the retransmission counter; the frame is then retransmitted if no ACK is received by the MAC before the timer is up.

However, in the FPGA/software system the MAC unit can no longer assume that the frame is sent and so does not know when to start its transmission timeout timer. If the MAC starts the timer immediately, and the carrier sense unit stops the frame from being transmitted, the MAC unit might assume the frame has been lost and thus retransmit it. This would lead to unnecessary frames being sent on the channel and to a lower throughput. Therefore to implement the FPGA/software hybrid version of the protocol we made a number of changes to the software running on the GPP.

Figure 3 shows a block diagram of the software implemented on the USRP E100. Iris components used in the software-only version of the protocol, which do not interface to the FPGA, are shown in light grey. The modifications necessary to allow feedback from the FPGA carrier sense module are highlighted in bold. In order to give feedback to MAC functions implemented in software as to when a frame is sent we exploited the existing "frame sent" feedback message of UHD.

We created an Iris controller block that eavesdrops on these UHD status messages and reports back to the MAC whenever a frame is transmitted over the air. We changed the MAC

TABLE III
RF PARAMETERS

RF Parameter	Value
Frame size	600 B
Center Frequency	5.009 GHz
Bandwidth	200 KHz
FFT size	256

to utilise this information to start the retransmission timeout timer. Thus, the retransmission timeout timer is only initiated once the frame has been sent across the channel, instead of when it leaves the MAC layer. In this way, the software and hardware work in tandem to provide access to the channel.

IV. Test and results

In order to evaluate the impact of our carrier-sense module in a network of SDR nodes, we built a prototype of the carrier sense module and tested it in an experimental two-hop network, we then replicated the setup in a network simulator to cross-validate our findings.

A. Protocols

In the current work, we incorporate the proposed split functionality into two versions of a CSMA MAC protocol, one employing explicit acknowledgements and the other employing implicit acknowledgements, where this last one is particularly tailored for multihop wireless environments. For more details about the explicit and the implicit acknowledgments protocols, interested readers are referred to [4].

B. Carrier Sense Results

Our two-hop testbed network consisted of three USRP E100 nodes with Iris running on the ARM processor and the XCVR2450 daughterboard [5] as a RF front-end transceiver. The testbed uses a host machine running a local NTP server for synchronization and monitoring purposes.

Each experimental run involved 100 frames successfully transmitted from the source node to the destination node through one intermediate node. The source and the destination are within interference range of each other. In this setting, the channel contention is due to intra-flow interference created by the DATA/ACK frame exchange between nodes. Table III presents the parameters used for the radio experimental setup. It is worth noting how, even though the HW used would allow data rates up to 8 MSamples/sec [5], we made use of a data rate of 200 KSamples/sec in order to avoid the loss of samples from the FPGA to the OMAP due to overflow events [8].

It is understood that SDR operations and the wireless environment introduce a high degree of unpredictability on the system's behavior. For that reason, we repeated the experiments 50 times for each configuration and report the results in a box-and-whisker diagram (Figures 4 and 5). In order to assess the performance of the system, we looked at two metrics: the total number of retransmissions for a DATA frame, given by the sum of the retransmissions at the source node and the intermediate node, and the end-to-end throughput.

In Figures 4 and 5 we report the total number of retransmissions in the system and the end-to-end throughput, respectively. A few observations are noteworthy for the implementation results. First, despite the presence of the carrier-sensing, retransmissions can still take place due to either errors in correction of the carrier offset (rare) or timeout expiration in the MAC due to over-the-air collisions. Second, since fewer frames are required when using CSMA with *Implicit ACKs* compared to CSMA with *Explicit ACKs*, thus less contention for the medium, the former exhibits fewer retransmissions compared to the latter. Third, since the throughput is influenced by both retransmissions and the Round Trip Time (RTT), we observed that the end-to-end throughput in the *Explicit ACKs* with Carrier Sensing almost matches the throughput in the *Implicit ACKs* with Carrier Sensing. This is due to the time required by the source node to receive the corresponding ACK from the intermediate node. The ACK frame sent by the intermediate node in the Explicit strategy is short, requiring little time over the air, compared to an Implicit ACK frame, which has the dimension of a DATA frame. The consequence is that with the Explicit strategy, a new frame is introduced into the network earlier than with the Implicit strategy. As a result, the end-to-end throughput for the Explicit ACK case is very close the end-to-end throughput for the Implicit ACK despite showing more retransmissions.

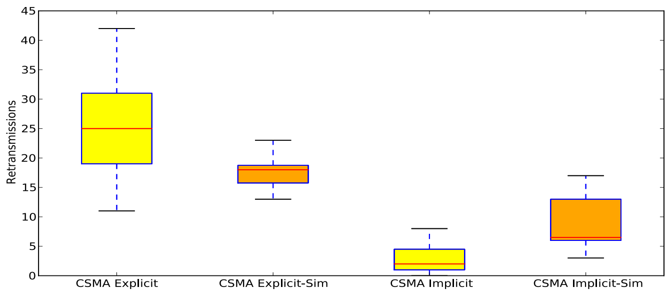

Fig. 4. Retransmissions. Implementation and simulations results are shown in yellow and orange respectively. For each box, the whiskers below and above show the minimum and maximum value, respectively. The bottom and the top of the box represent the 1st and 3rd Quartile respectively. The horizontal line inside the box represents the median. Outliers are excluded from calculation of min/max and are shown as small crosses. An outlier is defined as the value of an observation which is less than 1.5x1stQuartile or greater than 1.5x3rdQuartile.

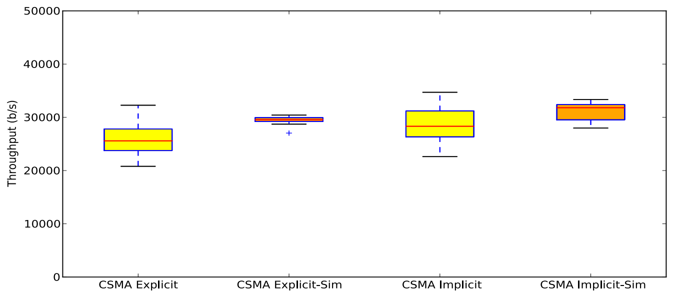

Fig. 5. Throughput. Implementation and simulations results are shown in yellow and orange respectively. See Figure 4 for details

C. Simulations

Simulations play an important role during protocol design and refinement phases. However, many simulation results are not believable because they are never validated against realworld or experimental scenarios. We sought here to create a validated simulation to reinforce our understanding of our experimental results and to enable us to test our protocols in situations unobtainable in the lab. The OMNeT++ network simulator was selected. We set up a two-hop network using OMNeT++'s standard libraries and modified existing modules in order to replicate our SDR implementation of CSMA, including the impact of modules running in the GPP and those running in the FPGA. It is well known that a conventional SDR node based on USRP suffers of *unpredictable turnaround* time [4]. Unfortunately, as we have experienced, these effects are more prominent in embedded systems.

In order to model the unpredictable turnaround time, to simulate an SDR node, we first characterized the time required to process the signal, and the time needed to pass data to each component. Then we modified the existing Network, MAC, and PHY layer modules in OMNeT++ to introduce the delays modeled. In Table IV we report the time spent in critical components in the receiver and transmitter chain, differentiating whether the entry refers to the time necessary to process a DATA frame or an ACK. We also calculated the time taken by the different engines to pass data among themselves. Several observations can be drawn from the results.

TABLE IV
PROFILING COMPONENTS IN IRIS

Component	Frame	μ[ms]	Median[ms]	σ[ms]
Modulator	DATA	2.58	2.57	0.06
	ACK	0.45	0.45	0.03
UhdTx	DATA	0.34	0.33	0.03
	ACK	0.16	0.15	0.03
Demodulator	DATA	7.45	7.23	2.58
	ACK	6.53	7.21	2.43
Data Passing				
Stack$\rightarrow$PN	DATA	3.26	0.52	5.94
	ACK	3.75	0.23	8.43
PN$\rightarrow$Stack	DATA	23.16	10.18	31.15
	ACK	27.99	23.98	26.72

First, we observe that the receiver chain is the critical part in a SDR node. While the transmitter chain works only when the MAC sends down a frame, the receiver chain has to process samples coming constantly from the FPGA. Anytime the UHDRx fills its buffer with samples coming from the FPGA, it sends these samples to the next component, processed in turn in Iris. Second, the different frames' dimensions do not affect the processing time on the demodulator. The reason for that resides on equalization and carrier phase recovery functions, which are independent from the data size. Last, the most critical operation and the real source of unpredictability for the turnaround time (which also affects the RTT) is not in the demodulator, as we were expecting, but in the data being passed from the *PN Engine* to the *Stack Engine* during the receiving phase. The reasons for that stem from a combination of the current Iris framework and the thread scheduling in the operating system handling the USRP E100. The thread running the PN Engine in the receiver chain gains greater priority with respect to the one running the Stack Engine because it constantly has data to be analyzed. This leads to an unfair scheduling of the resources between the threads involved in Iris, where, even if the *Stack component* has data sitting in the buffer ready to be processed, it waits for the *PN Engine* to complete its operations.

We repeated the simulations 15 times for each MAC configuration. We discovered that the simulation and the implementation results were remarkably consistent as we show in Figures 4 and 5. As a result, our experimental findings have been cross-validated by the simulation results.

V. Conclusions & Future work

In this paper we have described a general architecture for CSMA MAC protocols on inexpensive reconfigurable radio platforms. We have designed this architecture using a split-functionality approach, moving the most delay sensitive functions on the FPGA. We have then combined this FPGA-based implementation with appropriate modifications to the GPP-based MAC functions of a software radio built with the Iris software platform and the USRP E100, though the FPGA-modifications themselves are software platform agnostic.

We performed mutihop experiments to assess the performance of our carrier-sensing MAC in a realistic environment, measuring the throughput and number of retransmissions in a two-hop linear network.

We also cross-validated our experimental results using a network simulator (OMNet++), which we modified to reflect the details of our MAC implementation as well as the component delay times seen in our experimental system.

We are currently enhancing our carrier-sensing MAC protocol implementing backoff functionality adjacent to the sensing mechanism on the FPGA and more sophisticated frames dependent logic. We also plan further tests in larger, more complex (and more realistic) network topologies.

References

[1] T. Schmid, O. Sekkat, and M. B. Srivastava, "An Experimental Stady of Network Performance Impact of Increased Latency in Software Defined Radios," in *Proc. of ACM workshop on Wireless network testbeds, experimental evaluation and characterization (WinTECH)*, 2007.

[2] A. Puschmann, M. A. Kalil, and A. Mitschele-Thiel, "A Flexible CSMA based MAC protocol for Software Defined Radios," *Journal of RF-Engineering and Telecommunications (Frequenz)*, 2012; **6(9-10)**.

[3] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC protocol Implementations on Software Defined Radios," *in Proc. of USENIX symposium on Networked systems design and implementation (NSDI)*, 2009.

[4] J. C. O' Sullivan, P. Di Francesco, U. K. Anyanwu, L. DaSilva, and A. MacKenzie, "Mutihop MAC implementations for affordable SDR hardware", in *IEEE New Frontiers in Dynamic Spectrum Access Networks (DySPAN),* 2011.

[5] "Ettus Research LLC." [Online]. Available: http://www.ettus.com/.

[6] P. D. Sutton, J. Lötze, H. Lahlou, S. A. Fahmy, K. E. Nolan, B. Özgül, T. W. Rondeau, J. Noguera, and L. E. Doyle, "Iris: An Architecture for Cognitive Radio Networking Testbeds", *IEEE Communications Magazine*, 2010; **48(9)**, pp. 114–122.

[7] R. Normoyle, and P. Mesibo, "The VITA radio transport as a framework for Software Definable Radio architectures," in *SDR 08 Technical Conference and Product Exposition*, 2008.

[8] P. Balister, 2011. *High Performance Interface between the OMAP3 and an FPGA*, [Online]. Available: http://elinux.org/images/7/7b/Omap3-fpga.pdf.