Functional and Performance Analysis of CalRadio 1 platform

Riccardo Manfrin, Andrea Zanella and Michele Zorzi
Department of Information Engineering – University of Padova, Italy
Consorzio Ferrara Ricerche
E-mail: {rmanfrin, zanella, zorzi}@dei.unipd.it

***Abstract*—**
CalRadio 1 is an open 802.11b-compatible development platform, designed and developed at UCSD with the aim of providing the research community with an open and fully reprogrammable board for experimental purposes. In this work we describe the hardware and software architecture of the board and we provide an accurate analysis of the limiting performance achieved by CalRadio 1 in comparison with commercial 802.11b wireless interfaces. The analysis offers a clear vision of the real potential and limitations of the CalRadio 1 board, pointing out the aspects of major concern for prospective developers.

***Index Terms*—CalRadio 1, DSP, ARM, MAC 802.11b**

I. Introduction

In the last few years the literature has been teeming with papers that propose novel algorithms and protocols for wireless networks. In most of these papers, performance analysis is obtained by means of mathematical models or, more often, computer simulations. Although the importance of these methods of analysis is undeniable, it is also recognized that simulation and mathematical models cannot fully capture the very complex and often unpredictable interactions that characterize a real system, nor include the practical constraints that have to be dealt with when implementing any novel idea in the real world. As a consequence, the results obtained by using these classical tools of analysis cannot be blindly trusted, since they might significantly differ from what observed when the system is deployed in a real environment. The scientific community has recognized the necessity of supporting "ethereal" analysis with "earthly" observations that can be obtained by means of experimental campaigns on real-world testbeds. Unfortunately, most commercial devices do not provide access to low-level functionalities and components of the board, thus preventing any modification of the software and firmware modules that control the device.

For these reasons, in the last years many actors have been designing and commercializing a number of open wireless communication platforms that permit fast implementation and testing of new algorithms and protocols. Examples of such platforms include the Universal Software Radio Peripheral (USRP) [1] and the related software GNURadio [2], Wireless Open-Access Research Platform (WARP) [3], FlexRadio [4] and CalRadio 1 [5], [6].

In this paper, we focus on the CalRadio 1 platform, an 802.11b [7] compliant, standalone board that can potentially provide the above mentioned functionalities for fast development and testing of communication algorithms and protocols.

CalRadio 1 is equipped with a 802.11b PHY layer in hardware. The advantage brought by the board is that the MAC layer is run on a DSP and can therefore be reprogrammed in standard ANSI C.

This work was partially supported by the EU through the ARAGORN project (FP7 ICT-216856) as part of the Seventh Framework Programme ICT-2007.1.1 "The network of the future"

In this work, we provide a functional and performance analysis of the board features, with the intent of giving to prospective developers a clear vision of the actual potential and limitations of the board. To this end, we start by describing the hardware and software architecture of CalRadio 1 in Sec. II. Then, in Sec. III we present and comment the results of a number of experiments that have been designed to evaluate the limiting performance of the device. The obtained results are then discussed in Sec. IV.

II. CalRadio 1 architecture

This section is devoted to the detailed description of the hardware architecture and of the software design of CalRadio 1, with the aim of providing the technical and practical information that we consider of interest for prospective developers.

A. Hardware

The CalRadio 1 platform consists of four main components, namely an ARM (ARMv7TDMI) processor, a Digital Signal Processor (TI 5471 DSP) [8], a Baseband (BB) processor (Intersil HFA3863) [9] and an RF transceiver (MAX28281) [10]. ARM and DSP are integrated in a single dual-core chip (TMS320VC5471) produced by Texas Instruments. With respect to the ISO/OSI reference protocol stack, the ARM processor supports the "higher layer" functionalities, from the network layer up, the DSP carries out the data link layer (DLL) functionalities, including Medium Access Control (MAC), whereas the physical layer (PHY) functionalities are implemented by the Baseband processor and the RF transceiver, as schematically represented in Fig. 1.

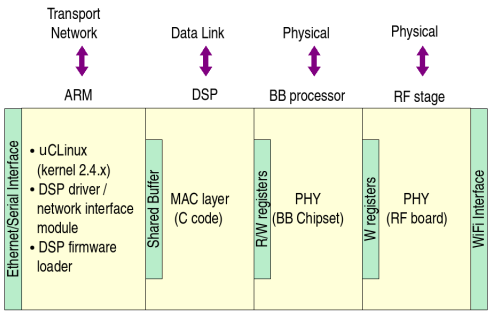

Figure 1. Hardware/software architecture

1) Modules interconnection: The different parts are connected to one another, in a hardware chain.

ARM and DSP are connected by means of a memory-mapped shared buffer residing on the DSP that is used for exchanging both data and control information. In order to avoid possible race conditions the buffer is spin-locked before accessing it. Reading and writing of the shared memory involves the Arithmetic-Logic Unit (ALU) of the DSP that, therefore,

cannot perform other operations in parallel. This characteristic also represents a potential bottleneck for the performance of the CalRadio 1 board. In fact, data reading/writing on the shared buffer may take a relatively long time that contributes to the overall packet delivery delay of the board.

Conversely, data transfer between DSP and Baseband processor is managed through Direct Memory Access (DMA) channels. Therefore, packet transfers from and to the Baseband do not engage the DSP, which can meanwhile perform other operations, such as rate selection, error checking and so on.

Communications between Baseband processor and RF transceiver are operated by means of dedicated channels and registers and occur without involving the DSP.

Besides the WiFi interface, CalRadio 1 offers a 10/100 Mbit/s Ethernet port for fast file transfers, and a serial connection, that is commonly used to control the device, and for code debugging purposes.

2) DSP functionalities: The feature that distinguishes CalRadio 1 from common commercial WiFi cards is the ability to access and reprogram the DLL and MAC protocols, which are run on the DSP. Moreover, the DSP controls the Baseband processor and the RF transceiver setup, thus permitting to directly play with most of the PHY settings also in runtime.

The DSP software can be written in ANSI-C language and compiled on an external PC. The generated binary code can be transferred to the ARM via standard File Transfer Protocol. However, the DSP only provides 72 kwords (of 16 bits each) of memory for storing the DLL/MAC software.

The DSP is also equipped with a single hardware timer, which is used to provide the absolute system time reference and for debugging purposes. The timer is clocked by a relatively fast (100 MHz) quartz with clock period $T_{CLK} = 10$ ns.

It must be noted that the timer should never expire, in order to maintain time consistency in tasks scheduling. This aspect may represent a limitation of the board, since it forces the developer to consider timer maintenance in software design.

3) Baseband and RF transceiver functionalities: As mentioned, the PHY functionalities are realized by the Baseband processor and the RF transceiver. In particular, the Baseband processor controls the IEEE 802.11 Clear Channel Assessment (CCA) signal according to the idle or busy status of the RF channel. Moreover it updates the Received Signal Strength Indication (RSSI), which describes the power level of each well-formed IEEE 802.11 received signal. The low level functionalities of the PHY layer, such as selection of the RF channel or RF signal modulation are carried out by the RF transceiver. Among others, a remarkable feature offered by the module is the ability to change the carrier frequency from 2.4 GHz to 2.499 GHz with a resolution of 1 MHz, thus permitting a very fast and fine spanning of the ISM band.

B. Software

CalRadio 1 is shipped with a minimal software development kit, which includes μCLinux [11] kernel, the toolchains for ARM and DSP, and a basic implementation of the Medium Access Control (MAC) protocol that permits bidirectional communication with legacy IEEE 802.11b cards in ad hoc mode. Fig. 2 provides a sketch of the software architecture.

1) Main Loop (ML): The implementation of the MAC layer on the DSP is based on an endless Main Loop (ML). Fig. 3 shows the flow of operations performed by the DSP in the ML. At the beginning of the ML, the DSP fetches any new data that the ARM has written onto the shared buffer. Data can consist of either commands for the DSP or data packets.

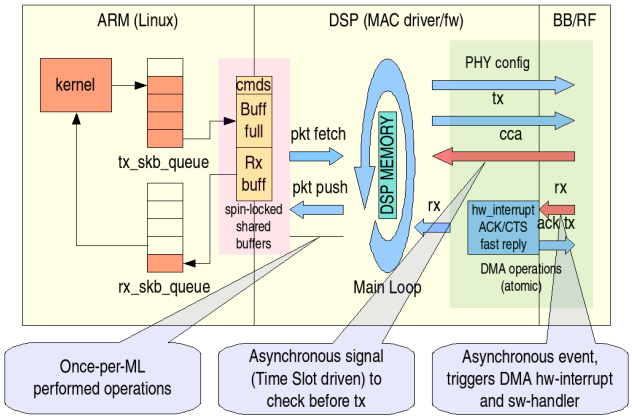

Figure 2. MAC software implementation

Once completed the transmission phase, the DSP checks whether its reception buffer has new packets coming from the RF transceiver. Any new packet is hence copied into the shared buffer, from which it will be read by the ARM. After this, the ML starts anew. In order to permit asynchronous data transfer between ARM and DSP, different segments of the shared buffer are allocated to ARM-to-DSP and DSP-to-ARM data exchange.

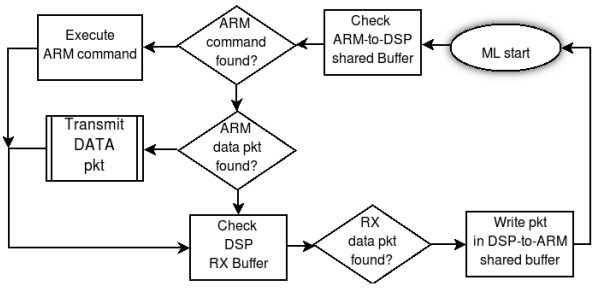

Figure 3. Main Loop operations flow

2) Packet Reception routine: All the above operations are executed in sequential order, once per ML cycle. However, asynchronous events may generate hardware interrupts that freeze the execution of the ML in order to let the DSP execute appropriate interrupt routines. In particular, when the RF transceiver locks a signal with a valid preamble, ML sequential code execution is blocked to invoke the reception interrupt handler. In the reception routine, the incoming data packet is demodulated by the RF transceiver and Baseband processor and copied into the DSP reception buffer through the DMA channels. These operations do not involve the DSP, which can use the packet reception time to set up in advance some critical operations that need to be performed immediately after the completion of the packet reception within strict time constraints. For instance, the IEEE 802.11b standard requires that, after a packet has been correctly received, the node transmits an ACK frame to the sender within a Short Inter-Frame Space (SIFS) of $10\mu s$, before the ACK timeout expires on the other host, as shown in Fig. 4. In this period of time the DSP needs to perform a number of operations, such as executing the data consistency check on the received packet, checking the destination address of the packet, creating the acknowledgement frame (ACK), when required, setting the transmission rate to the basic rate of 1 Mbit/s, and so on. It is then clear that the reception routine has to be carefully designed and optimized in order to allows the DSP to perform all these tasks while respecting the time

boundaries dictated by the standard.

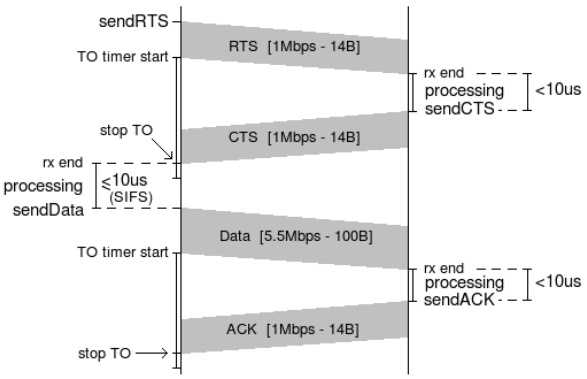

Figure 4. Transmission handshake

An asynchronous event, such as a new packet reception, can interrupt the execution of the ML in any instant. However, the DSP can act on an Interrupt Mask Register and an Interrupt Flag Register to selectively enable/disable the different types of interrupt in order to avoid recursive routine calls and/or interruption of critical operations that have to be kept atomic.

III. Performance Assessment

In this section we present the results of some basic experiments to compare the CalRadio 1 performance with respect to commercial cards and for identifying opportunities and drawbacks of the board as an open platform for experimental research. More specifically, we are interested on the following timing and throughput performance figures:

- minimum time required for: changing carrier frequency (RF channel), varying PHY rate, reading CCA register;
- maximum transfer rate from: ARM to DSP, DSP to PHY, ARM to PHY and application to application.

A. Timing performance figures

Timing measurements were obtained by using the DSP hardware timer (with time unit set to $\delta = 100\,ns$), which offers enough precision without exposing the timer to frequent maintenance procedures.

1) ARM to DSP packet fetching time: The first important measurement regards the time required for a packet to be fetched by the DSP when found on the shared buffer. Fig. 5 reports the time taken by the DSP to fetch a packet from the shared buffer, when varying the packet size. The measurements have been performed by modifying the ML in order to avoid any other operation and disabling all the interrupts that could block the execution of the code. As can be seen, the fetching time grows linearly with the packet size with a slope of 16 Mbit/s, which is of the same order of magnitude of the maximum transmission speed provided by the 802.11b PHY layer (11 Mbit/s). Therefore, the delay undertaken by a packet from the instant it has been written by the ARM to the shared buffer to the instant it is passed to the transmission procedure is comparable to the time required to transmit the packet through the wireless interface at the maximum admissible rate.

2) DSP to PHY packet transfer time: Even though the DSP is virtually not involved in DSP to PHY memory transfers, it is actually blocked to prevent critical errors that would occur, for instance, in case PHY settings are modified when transmission is still ongoing. Therefore, the data transfer from DSP to PHY represents another potential source of performance loss. In Fig. 6 we report the DMA transfer speed for various

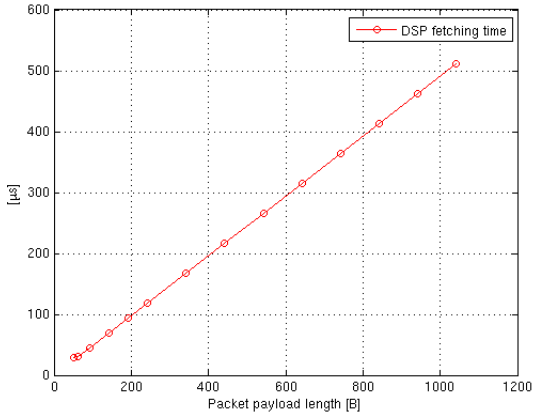

Figure 5. DSP packet fetching speed (from shared buffer)

packet sizes, when the PHY rate was set to 1 Mbps. As can be observed, the DMA transfer rate is almost independent of the packet size and slightly lower than the PHY rate. This small deviation from the reference rate can be ascribed to the fact that the time measurement actually includes some operations (such as timer maintenance) that cannot be avoided and that introduce unpredictable extra delay in the measured latency, thus lowering the measured transfer speed.

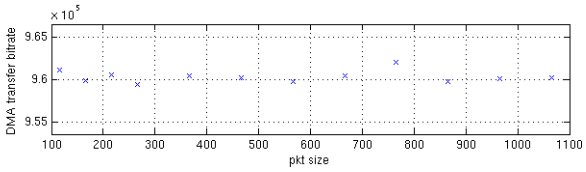

Figure 6. DMA transfer speed (PHY trasmission @ 1 Mbps)

3) Carrier-shifting time: Spectrum agility, dynamic spectrum access, spectrum-sniffing are just some examples of functionalities that require fast channel switching. Therefore, one of the aims of our analysis was to determine the time required by the CalRadio 1 card to change frequency channel. A carrier shifting operation involves the RF stage registers, which only allow write operations. Since the register cannot be read, it is not possible to directly measure the time taken by the RF-transceiver to actuate the command. To overcome this drawback, we set up an experiment in two stages. In the first stage, the same packet was continuously and repeatedly transmitted on a single frequency many times. In order to avoid any source of randomness in the measure, the DSP code was modified in order to bypass all the DLL and MAC procedures, such as backoff mechanism, physical and virtual carrier sense, retransmission procedure and so on. The receiver was programmed to keep track of the reception time of the different packets, which reflected only the latency due to the transfer of the data packet from the DSP to the Baseband, the time to transmit the data on-air, and the very limited processing delay required to manage the basic reception operations. In the second stage of the experiment, before each packet transmission, the sender switched the carrier frequency to a different channel and then immediately back to the original channel. In this way, all the packets were still transmitted on the same channel (thus permitting to keep the receiver on the same frequency), but each transmission was delayed by the time required to change the frequency twice. The average time difference between the two sets of results, reported in Fig. 7, gives the time required to perform two frequency shifting operations that amounts to approximately

$14\,\mu s$. Therefore, the time for a single carrier shifting operation is about $7\,\mu s$.

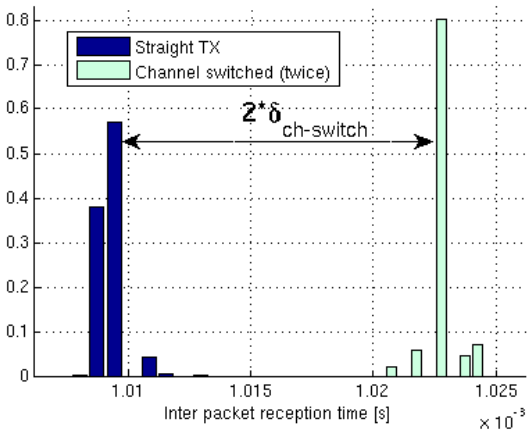

Figure 7. Time difference introduced by carrier shifting operations

Carrier shifting operations need to be kept atomic in order to avoid abnormal behavior, a requirement that introduces a small additional delay (that we accounted for in the obtained result) in order to disable/enable the proper interrupts.

4) PHY Rate-switching time: While frequency shift operations are carried out by the RF-transceiver, a change of the PHY rate involves the Baseband processor. Clearly, also rate switching has to be performed in an atomic fashion. Our measurements were performed in two different ways; a first series of results were obtained by writing and subsequently reading the register responsible for PHY rate switching. As reading and writing operations on the Baseband processor are symmetric, the amount of time required for each of them is the same and we can simply divide the resulting latency by 2 to obtain our measure.

The second method aims at confirming that there are no additional latencies introduced by the Baseband, once the rate setup has been written to the appropriate register. Such trial measures the difference between two consecutive transmissions performed without changing rate, and two transmissions with a rate switch in between. The measured time required to perform a rate switch operation was confirmed by both tests to be $4.3\mu s$.

It is important to understand that a change in PHY rate could need to be invoked for every single packet transmission. An additional delay is required to setup the packet duration on the Baseband. The responsible function must calculate the symbol length through the packet length in bytes and the selected rate for the subsequent transmission. Once the length is calculated, the Baseband must be notified about it, to know the transmission duration. All these tasks generate an additional delay of $8.2\mu s$.

With reference to Fig. 4, it is clear that rate adaptation requires software design optimization, as the update of the rate and packet length values for transmitting the ACK might take longer than the ACK timeout.

5) CCA reading time: The last local measure we performed refers to the CCA reading time. In particular, we were interested in determining how much time is left after a CCA check to perform other operations within an IEEE 802.11 time slot. While most of the 802.11 information elements are accessable through the Baseband processor registers, the CCA is direcly mapped to a dedicated input port of the DSP. This allows the developers to simply treat such signal as a volatile readable internal register always ready to be checked. This not only allows to read the CCA anytime during code execution, but grants to such operation a very high speed and independence from lower layers hardware constraints. We estimated the required time to be around $0.2\,\mu s$, which permits to develop rather sophisticated channel sensing procedures and leaves time to perform many other operations during a time slot.

B. Throughput measurements

In this section we will evaluate CalRadio 1 throughput efficiency through a set of measurements. First we will compare the different PHY rates with the real sustainable throughputs achieved, hence extracting an efficiency metric. We will then compare such result with the performance of Atheros commercially available WiFi boards.

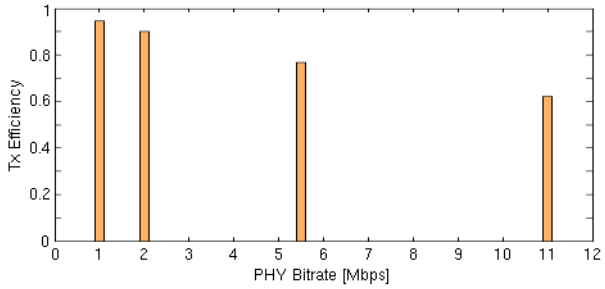

Figure 8. CalRadio transmission efficiency upper bound

In Fig. 8 we can observe a performance upper bound, derived from the results found in the above sections. The upper bound is based on the assuption that the device is only transmitting and we do not take into account the additional latencies introduced by MAC mechanisms such as backoff or acknowledgements. For the test, we set the packet size to 1000 bytes.

It can be seen that for higher PHY rates, the throughput efficiency decreases, as constant terms such as DSP fetching time become the major contributions to the overall transmission time. We can observe the same behavior in Fig. 9 for an ad hoc communication between two commercial WiFi interfaces (Atheros chipset).

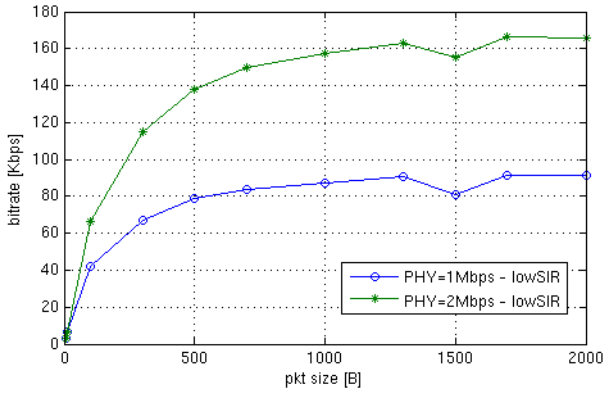

Figure 9. Atheros saturated tx performance

When switching the PHY rate from 1Mbps to 2Mbps, the throughput efficiency decreases. For the same reason, when increasing the packet length, the efficiency increases, as the idle time between two consecutive transmissions becomes negligible, compared to the transmission time.

For a final comparison, in Fig. 10 we report the efficiency results obtained for two Atheros commercial boards in ad hoc mode on the 13 available channels. In order to get accurate timings, disregarding any additional software introduced latencies, we chose to perform our measurements at the receiver, by

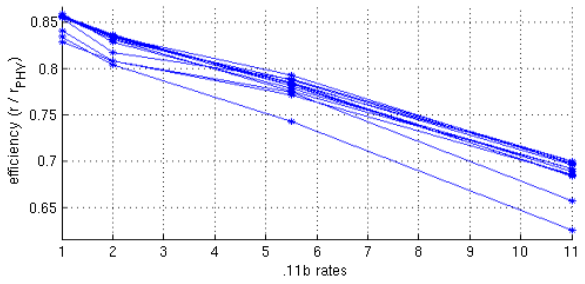

Figure 10. Atheros per channel throughput efficiency

sniffing the packets with a third Atheros interface in monitor mode. This gave us the required accuracy and independence from possible unpredictable packet queuing phenomena, inside the kernel buffers. In these tests, we chose to send unicast UDP packets in order to measure the efficiency with the various IEEE 802.11b rates. For a fair comparison that would not consider backoff and retransmissions, we measured the additional delay introduced when transmitting unicast packets instead of broadcast (PHY rate = 1Mbps, pkt length = 1000B) and removed this delay from the collected data.[1] Moreover, by means of artificially introduced sequence numbers, we considered only the timings between two packets successfully transmitted on the first try. The additional delay introduced by the unicast transmission is shown in Fig. 11. Its average value is $300\mu s$.

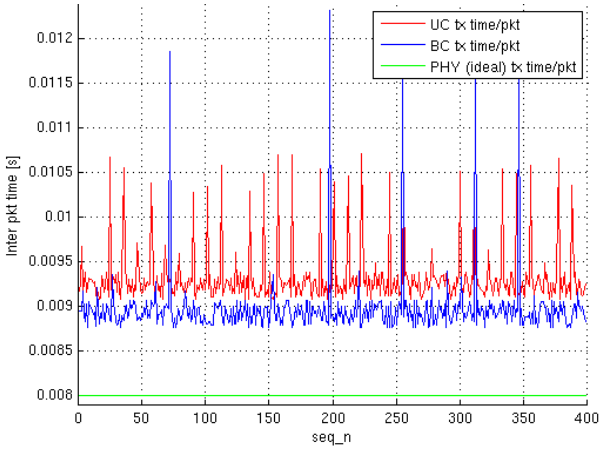

Figure 11. Atheros UC vs BC additional delay

By checking the sequence numbers, we managed to see that the majority of the packets were transmitted only once and correctly received. As a result, most of the delay is due to the ACK reply time, whose value fairly confirms the theoretical timings defined by the IEEE 802.11 standard. By subtracting this delay from the collected data, we managed to show the efficiency, based only on the additional time introduced by internal processing (and not by backoff/ACK timeout/retransmission mechanisms) and compare this with the CalRadio.

1) Measurement set-up: In order to figure out any possible bottleneck in CalRadio 1 architecture, we made our throughput measurements in two different fashions.

A first set of tests were performed decoupling the DSP from the ARM. This allowed the measurement of the pure transmission without the interference of the slower kernel processing at the ARM side. In order to bypass the ARM processing, we pushed a packet from the kernel onto the DSP. Here we transmitted the packet but did not notify the ARM of the end of the transmission. Moreover, we did not clean the shared buffer from which the packet is fetched by the DSP. Through this hack, we prevented the ARM from pushing new packets onto the DSP and kept finding a packet to transmit on the DSP for each new loop cycle. By disabling retransmissions and backoff mechanisms, we managed to see the maximum throughput available on the CalRadio 1, if the DSP did not have a MAC layer running on it and an ARM with a TCP/IP stack on top, controlling the flow of packets.

In the second set of measurements, we checked the ARM introduced delay by pushing packets through the entire TCP/IP stack in the kernel (we used UDP for the purpose).

2) DSP-PHY throughput: In order to monitor the CalRadio 1 transmission timings we set an Atheros interface in monitor mode and parsed the timestamps from incoming packets (with the help of libpcap [12]). On the CalRadio we sent a unicast packet towards a fake node and disabled the retransmission mechanism. Once the packet was loaded onto the shared buffer, the DSP started to transmit it. To evaluate the pure transmission efficiency, we also disabled reception on the DSP. Still some random operation (such as timer maintenance) could interrupt the code flow. With a sufficient number of collected timestamps, we obtained the results shown in Fig. 12.

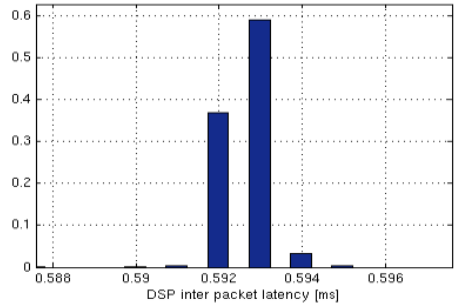

Figure 12. Distribution of the delay introduced by DSP processing for transmission

The resulting average delay introduced by the DSP processing, between the end of a packet transmission and the beginning of a new one (hence excluding PHY transmission time), is $593\mu s$. Assuming to transmit a 1000 bytes long packet, with R_{PHY} @ 1Mbps, we obtain the real efficiency of the CalRadio 1 with the following equation:

$$\eta_{\substack{@1Mbps \\ 1000B}} = \frac{T_{tx}}{T_{tx} + \overline{\Delta_{DSP}}} = \\ = \frac{L_{pr} + L_{PLCP} + L_{data}}{R_{PHY} \cdot (\frac{L_{pr}+L_{PLCP}+L_{data}}{R_{PHY}} + \overline{\Delta_{DSP}})} = \\ \simeq 0.9322;$$

where T_{tx} is the packet RF transmission time, L_{data} are the 1000 bytes of the packet sent, L_{pr} is the long preamble length (18 bytes), L_{PLCP} the PLCP header length (6 bytes) and $\overline{\Delta_{DSP}}$ is the average delay of $593\mu s$, introduced by the DSP between the end of a transmission and the beginning of a new one.

Again it must be noted that we are considering a non receiving device, hence not taking into account any acknowledge mechanism, just like we did by transmitting broadcast packets on the Atheros chipset. Thanks to this approach, we can fairly compare the chipsets in terms of efficiencies.

[1]We were forced to use unicast, as a broadcast transmission would not allow us to play with different rates

There is another consideration that needs to be made regarding the reception. When getting a packet from the RF stage, the DSP is stuck into a loop to check the DMA transfer until it is complete. Different approaches, possibly followed by other cards, could exploit these idle periods to perform other operations, hence allowing a higher efficiency.

Regarding the code and hardware implementation of the CalRadio 1, as mentioned above, we can see that the absence of a pipelined architecture represents one of the major elements that shape (and flatten) the efficiency upper bound. As for the transmission, we can assume that including the reception procedure would affect the transmission efficiency. This does not come from the additional RF reception time, which is something that affects every card, but again from the absence of a pipelined procedure to carry the packet out of the DSP, while it is being received by the RF.

3) ARM-PHY throughput: The last measurement we provide is the ARM introduced additional latency in the network traffic. In this case, every transmission is initiated by the ARM and then entrusted to the DSP. Before performing the measurement we disabled the virtual carrier sensing, retransmission mechanisms and the RF reception as well. This approach allows a comparison with the above results found for the DSP introduced lantency. The measurement results are shown in Fig. 13.

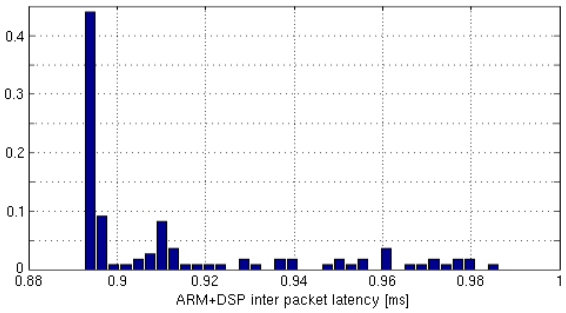

Figure 13. Distribution of the delay introduced by the ARM and the DSP processing for transmission

The average delay introduced when enabling the whole transmission chain from the TCP/IP stack on the ARM down to the transceiver is $909\mu s$. An important note is that, compared to the delay distribution found for the sole DSP transmission (Fig. 12), the presence of the ARM introduces a greater variance in the delay. As a result we have few packets affected by longer delays, which could represent a threat for time sensitive applications such as video/audio streaming. Comparing this result with the transmission time of a 1000 bytes long packet at 1Mbps we can find the efficiency

$$\eta_{\substack{@1Mbps \\ 1000B}} = 0.90;$$

As we can see the ARM processor introduces an additional delay between two adjacent transmitted packets. While this delay does not particularly affect the efficiency when considering slow PHY bitrates, it becomes more important for higher bitrates. This behavior is shown in Tab. I where we report the throughput efficiencies of the CalRadio 1 for the various 802.11b PHY rates. We derived these results from the efficiencies found at 1Mbps, considering that a difference in the used PHY rate only affects the average transmission time, while it does not modify the amount of required kernel and DSP processing.

PHY Rate	η_{DSP}	η_{ARM}
1Mbps	0.93	0.90
2Mbps	0.87	0.82
5.5Mbps	0.73	0.64
11Mbps	0.60	0.50

Table I
CALRADIO THROUGHPUT EFFICIENCIES

IV. CONCLUSION

In this paper we gave a brief overview of CalRadio 1, a flexible architecture equipped with a DSP controlled 802.11b interface for ready implementation and testing of MAC/PHY algorithms. Specifically, we analyzed the platform in terms of maximum throughput efficiency obtainable and measured its performance based on the time required for basic operations such as rate/channel switching or transmission setup. Through the results found the reader has the means to define the boundaries for the set of applications and algorithms that can or cannot be implemented on the CalRadio 1 platform. As an example, the measurement of the PHY rate switching time proved that it is possible to implement rate adaptation on the CalRadio 1 under the assumption of using optimized code and smart software design choices.

Other considerations can be made with regard to channel occupancy and selected PHY rate. Depending on the packet size and on the number of nodes competing for transmission, we can choose a slower PHY rate than the optimal one. This way we can match the CalRadio 1 DSP/ARM introduced delay with the time the channel shall be busy due to other transmitting nodes.

In the last part of this article we compared CalRadio 1 with standard commercially available 802.11b interfaces. The result showed that the CalRadio presents a comparable behavior to these commercial devices in terms of efficiency, while allowing a much greater degree of flexibility through a completely reconfigurable software MAC.

As we proved the feasibility of rate adaptation on the CalRadio, under the above mentioned conditions, our future work will focus on the modifications to the driver in order to support our algorithms. Specifically we will identify the critical operations in the code flow and will work on their optimization. The target of this work will be the developement of a module allowing direct plugin of any rate adaptation algorithm, to validate its effectiveness via experimental testing.

REFERENCES

[1] M. Ettus, *USRP User's and Developer's guide*.
[2] E. Research, "Universal Software Radio Peripheral - The Foundation for Complete Software Radio Systems Research."
[3] K. Amiri, Y. Sun, P. Murphy, Hunter, J. R. Cavallaro, and A. Sabharwal, "WARP, a Unified Wireless Network Testbed for Education and Research," in *IEEE International Conference on Microelectronic Systems Education (MSE '07)*, 2007.
[4] N. C. 3KNC, "FlexRadio Systems FLEX-5000A Software Defined Radio Transceiver," Oct. 2008.
[5] C. I. for Telecommunications and I. Technology, "Calradio website." [Online]. Available: http://calradio.calit2.net/calradio1.htm
[6] A. Jow, C. Schurgens, and D. Palmer, "CalRadio: A Portable, Flexible 802.11 Wireless Research Platform," in *1st international workshop on System evaluation for mobile platforms*, Jun. 2007.
[7] *ANSI/IEEE Std 802.11, 1999 Edition*, IEEE Std., Jun. 2003.
[8] T. Instruments, *TMS320VC5471 Fixed-Point Digital Signal Processor Data Manual*, Dec. 2002.
[9] Interil, *HFA3863 Direct Sequence Spread Spectrum Baseband Processor*, Apr. 2000.
[10] Maxim, *MAX2820/2821 2.4GHz 802.11b Zero-IF Transceivers*, Nov. 2003.
[11] D. J. Dionne and M. Durrant, "uCLinux - Embedded Linux/ Microcontroller Project." [Online]. Available: http://www.uclinux.org/
[12] V. Jacobson, C. Leres, and S. M. (LBNL), "tcpdump/libpcap." [Online]. Available: http://www.tcpdump.org/