Power Matters: Automatic Gain Control for a Software Defined Radio IEEE 802.11a/g/p Receiver

Bastian Bloessl, Christoph Sommer and Falko Dressler
Distributed Embedded Systems Group, University of Paderborn, Germany
{bloessl,sommer,dressler}@ccs-labs.org

***Abstract*—Software Defined Radios (SDRs) have become a fundamental building block for research on wireless networks. Yet, using this platform for experiments in the field is hindered by many practical difficulties, an important one being the need for Automatic Gain Control (AGC). We demonstrate a way of implementing an AGC algorithm directly in the FPGA where we are able to meet tough timing constraints. This allows using SDRs not just for lab experiments but also for measurement campaigns or deployments in the field. With extending our GNU Radio-based Open Source stack for IEEE 802.11a/g/p WLAN with AGC, we provide an SDR platform that can be integrated into existing WiFi networks just as well as into future vehicular networks.**

I. Introduction

Software Defined Radios (SDRs) have proven to be invaluable tools to develop proof-of-concept prototypes of new communication technologies as well as to gain deeper insights into existing technologies [1]. One particularly interesting class of SDRs is realized by driving a lightweight radio frontend from software running on a host computer (as opposed to a fully integrated FPGA solution): Such an SDR allows for rapid prototyping and easy debugging in the lab as well as quick reconfigurability in the field. One of the most popular examples is the Ettus USRP platform running GNU Radio [2].

Aside from experiments in the lab, SDRs are also considered for field tests, be it for realizing visions of true cognitive radios or for future proofing products with long life-cycles such as cars in vehicular networks [3]. However, the use of SDRs for performing experiments in real life conditions is hindered by many difficulties, ranging from tight protocol timing constraints in the medium access layer to having to deal with changing input power levels.

One particularly complex example is IEEE 802.11 WLAN and application stacks building on it, e.g., the IEEE 1609 Wireless Access in Vehicular Environment (WAVE) stack for vehicular networking which builds on IEEE 802.11p. In earlier work, we presented a GNU Radio-based IEEE 802.11a/g/p receiver [4] that was fitted for operation with an Ettus USRP N210. The software stack was shown to be interoperable with commercial WiFi cards as well as IEEE 802.11p prototypes and is released under an Open Source license.[1] We were even able to investigate the feasibility of a fully software-based solution and standard compliant broadcast transmissions with marginal modifications of the FPGA [5].

[1]http://www.wime-project.net

Yet, one problem remains unaddressed for deployment and experimentation in the field: Decoding frames reaching the receiver with a wide range of power levels requires it to flexibly adapt its input amplifier, setting the *receive gain* to an optimal value for each incoming frame. If the receive gain is set too high, clipping would render the frame undecodable. If it is set too low, the A/D converter's resolution would not be used efficiently, resulting in high relative quantization noise. Setting a fixed gain is fine for lab environments with relatively static receive powers but infeasible for measurements in the field. In off-the-shelf hardware this process is handled by a dedicated component, the receiver's *Automatic Gain Control (AGC)*.

Since optimizing the receive gain is a prerequisite for decoding the frame, AGC has a very tight time window for operation. In IEEE 802.11 WLAN, the window starts from detecting an incoming frame and must not be longer than the short training sequence. This poses a problem for SDRs based on GNU Radio running on a host computer, where the round trip time between SDR and the PC is in the order of ms, while the short training sequence lasts only 8 µs for IEEE 802.11a/g and 16 µs for IEEE 802.11p.

The solution we are proposing is to move AGC into the small FPGA on the radio frontend. Any such solution will always be specific to the communication stack in question, as demands are very different for, e.g., WLAN and LTE. We chose IEEE 802.11a/g/p as the target technology to augment our aforementioned existing Open Source software stack. Being able to correctly tune the input amplifier even in dynamic scenarios turns our SDR solution into a viable measurement equipment for outdoor experiments.

II. Implementation of AGC

We implement AGC on the FPGA of the Ettus N210 using Xilinx ISE version 12.3.[2] The N210 consists of an FPGA board for digital signal processing of the baseband signal that can be equipped with various RF frontends for operation on different frequencies. To implement AGC we have to extend the FPGA to directly control the amplifiers of the frontend, rendering the implementation frontend specific. Like in [5], we use the XCVR2450, allowing for operation on the 2.4 GHz and 5.9 GHz band.

The XCVR2450 uses a MAX2829 transceiver chip from Maxim Integrated that supports two different gain modes.

[2]We experienced problems with corrupt bitstreams on more recent versions.

During normal operation, the gain is set via SPI. Furthermore, the transceiver can be configured to set the gain directly via pins, avoiding any additional delay.

Concerning signal processing, there are two possible locations where we can hook in AGC functionality. Either directly after the A/D converter or after the signal is channelized and downsampled. Implementation after the A/D converters minimizes delay but bears additional problems: Since the RF frontend can only tune to fixed frequencies, a shift to the carrier frequency is done in digital domain. Therefore, the signal will not be centered on the carrier frequency directly after the A/D converter, making it harder to exploit the autocorrelation pattern of the preamble for frame detection.

Given these drawbacks, we implemented AGC based on the channelized samples and, thus, operate on the very same data that is also streamed to the PC. The general AGC procedure is to detect a frame, estimate its power level, and adjust the gain to bring it as close as possible to a desired reference power level. We determined the reference level empirically with a comprehensive set of Packet Delivery Ratio (PDR) measurements with different input power levels and RX gains.

Frame detection is implemented based on two mechanisms. First, we use autocorrelation to exploit the cyclic pattern of the short training sequence. Second, we trigger frame detection if the input power level exceeds a threshold where it overdrives the A/D converters and thus distorts the cyclic pattern.

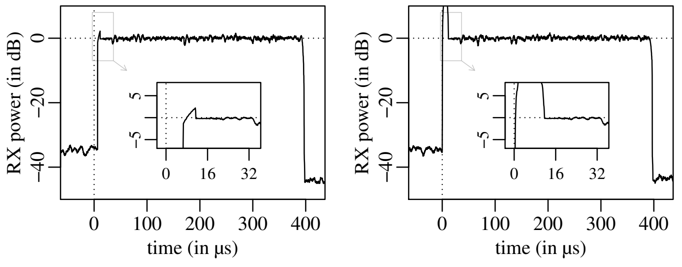
The gain of the MAX2829 transceiver can be distributed across two amplifiers, a Low Noise Amplifier (LNA) close to the antenna and a Variable Gain Amplifier (VGA) at a later signal processing stage. While the LNA provides a coarse resolution with only three steps for gain values 0 dB, 15 dB, and 30 dB, the VGA allows for finer resolution, covering 62 dB in steps of 2 dB. Both stages combined provide a gain range of 92 dB. When no frame is detected the AGC will switch to an intermediate level of 46 dB.

III. Evaluation

To highlight the need for AGC and to show its performance improvements over fixed gain configurations we used a setup with two WiFi receivers. One placed in close proximity of the SDR sending with high power (15 dBm) and, several meters away, another one sending a lower power signal of −15 dBm, i.e., the difference was greater than 30 dBm.

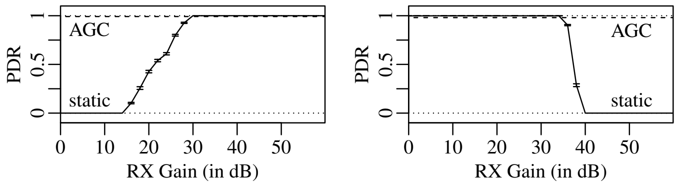
In a first experiment, we plotted the signal power in time domain to measure the delay that the AGC needs to reconfigure the gain. This delay is very interesting since the group delay of the digital signal processing chain in the FPGA is unknown.

The results are plotted in Figure 1a for the low power transmitter and in Figure 1b for the high power transmitter. Two observations are interesting in this context: First, we see that both signals are tuned to the very same power level, demonstrating basic functionality of the AGC, i.e., the low power signal is amplified while the high power signal gets attenuated. Second, the time to detect the frame, reconfigure the gain, and stabilize after a transient phase is shorter that the short training sequence of the WiFi frame. Hence, the actual frame is not corrupted by changing power levels, proving the feasibility of our approach.



(a) low TX power (b) high TX power

Figure 1. AGC delay when (a) increasing or (b) decreasing gain.



(a) low TX power (b) high TX power

Figure 2. Packet Delivery Ratio when using fixed RX gain settings or AGC.

In a second test, we highlight the need for AGC and show possible performance improvements. We use the same setup and measure the PDR for BSPK½-modulated frames with a payload of 128 Byte. For both configurations, we vary the receive gain to trigger the expected behavior. Figure 2a depicts the PDR of the low power transmitter. We see that for low receive gains, the frames cannot be decoded since the power level is very low, resulting in a high relative quantization noise. The exact opposite happens for high powered signals (cf. Figure 2b): With low receive gains, the frames can be decoded without problems, but for higher gains the A/D converters overdrive, resulting in clipping noise. For both configurations, we also measured the PDR with AGC enabled and were able to receive close to all frames.

A simple live demo of the AGC decoding frames of a WiFi card that varies its TX power level can be found on YouTube.[3]

References

[1] K. R. Chowdhury and T. Melodia, "Platforms and Testbeds for Experimental Evaluation of Cognitive Ad Hoc Networks," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 96–104, September 2010.

[2] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, no. 122, June 2004. [Online]. Available: http://www.linuxjournal.com/article/7319

[3] O. Altintas, M. Nishibori, T. Oshida, C. Yoshimura, Y. Fujii, K. Nishida, Y. Ihara, M. Saito, K. Tsukamoto, M. Tsuru, Y. Oie, R. Vuyyuru, A. Al Abbasi, M. Ohtake, M. Ohta, T. Fujii, S. Chen, S. Pagadarai, and A. M. Wyglinski, "Demonstration of Vehicle to Vehicle Communications over TV White Space," in *IEEE VTC2011-Fall*. San Francisco, CA: IEEE, September 2011, pp. 1–3.

[4] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio," in *ACM SIGCOMM SRIF 2013*. Hong Kong, China: ACM, August 2013, pp. 9–16.

[5] B. Bloessl, A. Puschmann, C. Sommer, and F. Dressler, "Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture," in *ACM WiNTECH 2014*. Maui, HI: ACM, September 2014, pp. 57–63.

[3]https://www.youtube.com/watch?v=fHyiXIpfw2I