Designing an Automatic Gain Control for the bladeRF

BY ROBERT GHILDUTA, NUAND LLC
Introduction

• Owner of Nuand, LLC
• Interests are software, hardware, and RF engineering
• Background in DSP (telecom, and acoustics)
• Previously: 6 years of professional experience building enterprise networking equipment
  • Worked on first wave1 and wave2 enterprise 802.11ac APs
• Long time information security researcher
Nuand is known for the bladeRF, a low cost USB3.0 software defined radio product line that is used by thousands of engineers, hobbyists, and researchers.

- Launched crowd-funding campaign for the original bladeRF in 2013
  - Revamped product line with bladeRF 2.0 micro
- Decades of combined engineering experience in Software Defined Radio, enterprise networking equipment, and defense
- Focusing on low-cost Software Defined Radios and applications
  - Support for GNURadio, gr-osmosdr, YateBTS, SDR#, SDRAngel, SRSLTE
  - Lightweight C library and API, command line interface
  - OS support: Linux, Windows, macOS, FreeBSD
The design of bladeRF 1

- OrCad for schematics
  - 300 components
- Cadence for layout
  - 8 layer, 5/5 mil PCB
  - FR-4 TG
- Simulations done with HSPICE
  - 2.5 field solver
Architecture of bladeRF 1

- Cypress FX3 USB 3.0 transceiver
  - Embedded 200MHz ARM7
- Altera Cyclone 4
- LimeMicro LMS6002D
  - “All in one” RF-to-bits MMIC
  - 300MHz – 3.8GHz RF range
  - Full duplex 12 bit 40MSPS quadrature sampling
- Si5338 synthesizer
  - SiLab’s top of the line synthesizer
- Development board
  - GPIO, LEDs, all JTAG connector populated
bladeRF 1 block diagram
LMS6002D
Incoming signals

- Realistically you will see things between -110dBm to -25dBm.
  - That’s a 85dB range!

- Problem....
  - Analog to Digital Converter (ADC) has a limited sensitivity range
    - Each bit is ~6.02dB
    - 12 bits is ~72.24dB
  - So a 12bit ADC does not have the ability to simultaneously listen to a -25dBm and -100dBm signal
  - This is known as static range
So how do we have a frontend that can listen to a -110dBm signal as well as a -25dBm signal?

They cannot occur simultaneously

So how about packets?

Change gains on a per packet basis by adjusting the RF frontend’s gains

- LNA1 (0 to 6dB)
- RXVGA1 (5dB to 30dB)
- RXVGA2 (0dB to 30dB)

This is known as dynamic range

Total dynamic range of bladeRF 1’s RF gain stages is 66dB

Total dynamic range of bladeRF 1’s sensitivity is 66dB+72dB=138dB
Create a framework where user code (waveform) interacts with a device-independent AGC interface

- Contains IIR, decisions to increase gain, etc

- Device Driver contain gain strategy tables, and all necessary logic to change gain
Motivations

- AGC is tightly coupled to the waveform being developed
- Unified AGC helps more easily port waveforms between devices
- Reusable RF Front End Drivers
- Unify behavior of AGC
  - Digital Interface
    - Request hold
    - Signal valid
  - Performance
    - Settling times
What order should the gains be adjusted in?

- Combat noise figure
  - Amplifiers amplify and add noise
  - Quickly separate signal from noise floor
- Friis formulas for noise basically says:
  - It’s best to first turn up gain (and noise) of the LNA
### Recommended gain tables

5.18 What is recommended gain table for the receiver?

#### Receiver gain table versus modulated input signal level 3GPP Band 1 (1950MHz)

<table>
<thead>
<tr>
<th>Min. Signal (dBm)</th>
<th>Max. Signal (dBm)</th>
<th>SNR(dB) Min.</th>
<th>SNR(dB) Max.</th>
<th>Antenna Switch</th>
<th>LNA Gain</th>
<th>RxVGA1 gain</th>
<th>RxVGA2 gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>-117 **</td>
<td>-85</td>
<td>-17.4</td>
<td>14.6</td>
<td>-</td>
<td>Max.</td>
<td>Max.</td>
<td>Max.</td>
</tr>
<tr>
<td>-84 *</td>
<td>-53</td>
<td>15.6</td>
<td>46.6</td>
<td>-</td>
<td>Max.</td>
<td>Max.</td>
<td>Variable</td>
</tr>
<tr>
<td>-52</td>
<td>-28</td>
<td>47.6</td>
<td>71</td>
<td>-</td>
<td>Max.</td>
<td>Variable</td>
<td>Min.</td>
</tr>
<tr>
<td>-27</td>
<td>-22</td>
<td>69.46</td>
<td>74.46</td>
<td>-</td>
<td>Mid (Max - 6 dB)</td>
<td>Min.</td>
<td>Min.</td>
</tr>
<tr>
<td>-21</td>
<td>-13</td>
<td>67.98</td>
<td>75.98</td>
<td>-</td>
<td>Bypassed</td>
<td>Min.</td>
<td>Min.</td>
</tr>
<tr>
<td>-12</td>
<td>4</td>
<td>60.5</td>
<td>76.48</td>
<td>Switched</td>
<td>Bypassed</td>
<td>Min.</td>
<td>Min.</td>
</tr>
</tbody>
</table>

#### Receiver gain table versus modulated input signal level for Band 5 (840 MHz)

<table>
<thead>
<tr>
<th>Min. Signal (dBm)</th>
<th>Max. Signal (dBm)</th>
<th>SNR(dB) Min.</th>
<th>SNR(dB) Max.</th>
<th>Antenna Switch</th>
<th>LNA Gain</th>
<th>RxVGA1 gain</th>
<th>RxVGA2 gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>-84 *</td>
<td>-53</td>
<td>15.5</td>
<td>46.5</td>
<td>-</td>
<td>Max.</td>
<td>Max.</td>
<td>Variable</td>
</tr>
<tr>
<td>-52</td>
<td>-39</td>
<td>47.5</td>
<td>54.75</td>
<td>-</td>
<td>Max.</td>
<td>Variable</td>
<td>Min.</td>
</tr>
<tr>
<td>-38</td>
<td>-32</td>
<td>51.25</td>
<td>57.25</td>
<td>-</td>
<td>Mid (Max - 6 dB)</td>
<td>Min.</td>
<td>Min.</td>
</tr>
<tr>
<td>-31</td>
<td>-13</td>
<td>51.5</td>
<td>69.5</td>
<td>-</td>
<td>Bypassed</td>
<td>Min.</td>
<td>Min.</td>
</tr>
<tr>
<td>-12</td>
<td>4</td>
<td>51.5</td>
<td>69.5</td>
<td>Switched</td>
<td>Bypassed</td>
<td>Min.</td>
<td>Min.</td>
</tr>
</tbody>
</table>
6 different gains will take far too long to settle
- Each gain that is changed is one separate SPI transaction
- I wanted a fast AGC

Solution:
- 3 gain settings, High/Mid/Low
- At most only 2 variables change (33% faster than changing 3 variables!)

<table>
<thead>
<tr>
<th>High gain: -82dBm - -52dBm</th>
<th>Mid gain: -52dBm - -30dBm</th>
<th>Low gain: -30dBm - -17dBm</th>
</tr>
</thead>
<tbody>
<tr>
<td>lna_gain</td>
<td>lna_gain</td>
<td>lna_gain</td>
</tr>
<tr>
<td>=&gt; 6 dB (Max)</td>
<td>=&gt; 3 dB (Mid)</td>
<td>=&gt; 3 dB (Mid)</td>
</tr>
<tr>
<td>rxvga1_gain =&gt; 30 dB</td>
<td>rxvga1_gain =&gt; 30 dB</td>
<td>rxvga1_gain =&gt; 12 dB</td>
</tr>
<tr>
<td>rxvga2_gain =&gt; 15 dB</td>
<td>rxvga2_gain =&gt; 0 dB</td>
<td>rxvga2_gain =&gt; 0 dB</td>
</tr>
</tbody>
</table>
Now you know the gains – how do you make this automatic?

- **Starting configuration**
  - Gains should be turned up high
    - It’s better to hear a signal and clip than to not hear a signal and miss it
  - Gains should decrease as a “hot” signal comes in
    - 12-bit ADC means you’ll have a -2047 to 2048 range on IQ samples
    - Decrease gain when within 3dB of hitting the top value (2048)

- **Problems**
  - Gain stages have group delay
    - You need to wait a little bit
  - Signal has a high PAPR or is very noisy
    - IIR
      - Solution: Hysteresis
  - Signal is very hot
    - Shortcut settling time
Tested using one of these
FPGA Development

- Altera Cyclone IV (40 or 115 kLE)
  - Interfaces with:
    - LMS6002D Transceiver
    - Si5338 Clock Controller
  - Timestamp support
  - (Manual) IQ Balance
    - Automated in GNU-Radio

- bladeRF FPGA support written in VHDL

- **Quartus II 16.0 Web Edition** ($0)
  - Can use SignalTap if you enabled TalkBack
HDL logic

- SPI Bus sharing (bus arbiter)
  - LMS controller only has one set of SPI controller lines
    - Commands can come in from NiOS (libusb)
    - AGC should have priority
  - Priority round-robin implementation
Arbiter
Transceiver imperfections

- AGC’s additional RF problem
  - Frequency specific DC offset
    - Every time the gain is changed the DC IQ cal changes

- IQ imbalance and DC offset are an analog phenomenon in Zero IF transceivers
  - Most transceivers have the ability to use analog correction methods
    - Opamps and programmable chargepumps to subtract out DC offset

- DC Offset
  - Expected in zero IF architectures
  - Analog values not “resting” at exactly 0
  - Symptoms
    - Y offset in time domain
    - “Large” spike in frequency domain
    - Uncentered constellation
  - Huge problem for AGC because power = sqrt(I^2+Q^2)
- IQ imbalance
  - IQ amplitude differences and phase offsets
  - Most noticeable as “mirror image” at ±$F_{\text{sig}}$
  - HW/SW compensation
    - gr-iqbal

- IQ imbalance is gain setting agnostic but DC offset is not
  - The closer the gain stage is to the ADC the more of an effect it has

- libbladeRF
  - Calculates settings of analog DC cal stages
    - Too impractical to change these values when changing gain
    - Easiest solution is to keep analog compensators the same and just subtract out DC offset
      - Calculated during new DC cal
Solution?

- DC lookup table mux
- Calibrate DC offsets at max gain value across band at 100MHz increments
  - Calibrate DC offset opamps
  - Then adjust gains to mid and low values and observe DC mean error
  - Save DC correction values and mean error values for high, mid, low gain settings
- At runtime libbladeRF loads appropriate high, mid, low gain settings
  - These are exported in HDL
  - The AGC tells a mux its setting
  - The mux then selects the appropriate mean error value
  - The correction block then removes the DC offset
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_dec_req</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_inc_req</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_rst_req</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_high</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_mid</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>gain_low</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc</td>
<td>reset</td>
</tr>
<tr>
<td>lms6002d:U_lms6002d</td>
<td>rx_reset</td>
</tr>
<tr>
<td>synchronizer:U_agc_en</td>
<td>sync</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>gpio_export[12]</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>current.gain_state:HIGH_GAIN_STATE</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>current.gain_state:LOW_GAIN_STATE</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>current.gain_state:MID_GAIN_STATE</td>
</tr>
<tr>
<td>bladerf_agc_lms_drv:U_agc_lms</td>
<td>current.gain_state:UNSET_GAIN_STATE</td>
</tr>
<tr>
<td>Group</td>
<td>00000h</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>nios_system:U_nios_system</td>
<td>arbiter:arbiter_0</td>
</tr>
<tr>
<td>* fpda_dc_i_correction[0.15]</td>
<td>2</td>
</tr>
<tr>
<td>* fpda_dc_q_correction[0.15]</td>
<td>21</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_i_max_export[0.15]</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_q_max_export[0.15]</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_i_mid_export[0.15]</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_q_mid_export[0.15]</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_i_min_export[0.15]</td>
</tr>
<tr>
<td>* nios_system:U_nios_system</td>
<td>agc_dc_q_min_export[0.15]</td>
</tr>
<tr>
<td>Group</td>
<td></td>
</tr>
</tbody>
</table>
bladeRF 2.0 micro

- Improved RF, power, and form factor performance over original bladeRF
- Characteristics and capabilities
  - 2x2 MIMO, 61.44MSPS sampling rate, 56MHZ IBW
  - 47MHz to 6GHz frequency range, shielding and spur mitigation for high SFDR performance
  - Cyclone V FPGA – xA4 (49KLE) and xA9 (with largest-in-class 301KLE FPGA)
- Bias-Tee Modules – Powered directly by the bladeRF 2.0 micro
  - Wideband Low-Noise Amplifiers and Power Amplifiers, and (soon) Filters