A Bluetooth Low-Energy Capture And Analysis Tool Using Software-Defined Radio

by

Christopher D. Kilgour

B.Eng., University Of Victoria, 1992

Project Submitted In Partial Fulfillment of the
Requirements for the Degree of
Master of Engineering

in the
School of Engineering Science
Faculty of Applied Science

Approval

Name:	**Christopher D. Kilgour**
Degree:	**Master of Engineering Science**
Title of Project:	***A Bluetooth Low-Energy Capture And Analysis Tool Using Software-Defined Radio***
Examining Committee:	**Chair:** Dr. Jie Liang Professor of Engineering Science, Graduate Chair

Dr. Rodney Vaughan
Senior Supervisor
Professor of Engineering Science

Dr. Shawn Stapleton
Supervisor
Professor of Engineering Science

Date Presented/Approved: April 19, 2013

Abstract

Wireless protocol analysis is a useful tool for researchers, engineers, and network security professionals. Exhaustive BTLE sniffing – the full capture and analysis of Bluetooth Low-Energy radio transmissions – has been out of reach for individuals to apply to research, engineering, and security analysis tasks. Discovering and following an arbitrary Bluetooth frequency-hopping pattern with a cheap narrow-band receiver is a complex undertaking with little chance of success. Further, the high-end test equipment capable of wide-band processing is prohibitively expensive to small organizations and individuals. This project includes an analysis of the problem, and a description of a wide-band Bluetooth sniffing implementation with open-source software and software-defined radio (SDR) techniques. A number of real-world signal captures are collected and processed, and the recovered Bluetooth data is presented. The resulting implementation is also published under an open-source license.

Keywords: Bluetooth; Low-Energy; Open-source software, Software-defined radio

Table of Contents

List of Tables

List of Figures

List of Acronyms

AA	Access Address
ADC	Analog-to-Digital Converter
BER	Bit Error Rate
BR	Basic-Rate (Bluetooth)
BT	Bandwidth-Time (product)
BTLE	Bluetooth Low-Energy
CORDIC	Coordinate Rotation Digital Computer
CRC	Cyclical Redundancy Check
DAC	Digital-to-Analog Converter
DDC	Digital Down-Converter
DUC	Digital Up-Converter
EUI	Extended Unique Identifier
FEC	Forward Error Correction
FPGA	Field-Programmable Gate Array
GMSK	Gaussian Minimum-Shift Keying
HCI	Host Controller Interface (Bluetooth)
HEC	Header Error Control
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific, Medical (frequency band)
LAP	Lower-Address Part
LE	Low-Energy (Bluetooth)
LFSR	Linear Feedback Shift Register
OUI	Organizationally Unique Identifier
PC	Personal Computer
PDU	Protocol Data Unit
RF	Radio Frequency
SAW	Surface Acoustic Wave
SDR	Software-Defined Radio
SFU	Simon Fraser University
SNR	Signal-to-Noise Ratio
UAP	Upper Address Part

USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VITA	VMEbus International Trade Association
XOR	eXclusive OR

1. Introduction

Wireless protocol analysis is a useful tool for researchers, engineers, and network security professionals. Capturing and analyzing Bluetooth traffic is more complicated and expensive than with other short-range wireless data networking technologies, but demand for cheap capture tools is increasing. The proliferation of smartphones has provided software developers with access to a large installed base of Bluetooth-enabled computing platforms. Combined with recent updates to the Bluetooth standard, a new generation of Bluetooth-enabled devices is now hitting the market. The development and testing of a these peripheral devices and accompanying smartphone applications is hindered by inadequate or overly expensive test tools. Setting out to improve the situation, this project report describes an implementation an exhaustive Bluetooth Low Energy capture tool based on GNUradio software, and leveraging relatively inexpensive software-defined radio hardware.

For wireless data standards such as IEEE 802.11, the ability to monitor and capture arbitrary RF channels with various modulation and coding schemes is built into even the most inexpensive chipsets already found in a typical laptop computer. This has allowed individuals and small organizations to collect detailed information on the operation of their network, and for example, facilitated research into the security of 802.11 standards. Although Bluetooth standards pre-date those of 802.11, Bluetooth devices have never been able to monitor and capture Bluetooth arbitrary traffic. Some vendors do offer Bluetooth test products, but these are highly-specialized, geared towards device manufacturers, and are therefore expensive and out of reach of the typical small developer. With the advent of cheap computers, high-speed data conversion devices, and FPGAs, a relatively cheap signal-processing platform is available to individuals. The SDR application described in this report can be applied by those developing modern Bluetooth-enabled products.

2. Project Overview

Figure 2.1 shows the project blocks in their simplest form. An antenna connects to a Universal Software-Defined Radio Peripheral (USRP) which in turn connects to a commodity Personal Computer (PC) with gigabit ethernet.

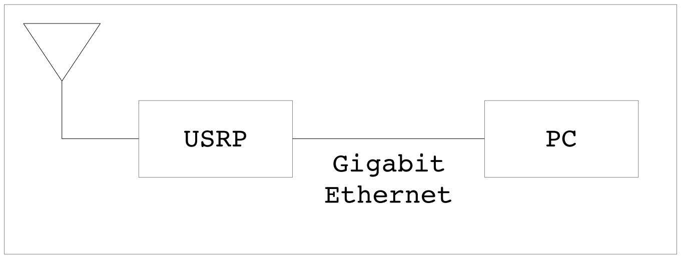


Figure 2.1: System Block Diagram (Simplified)

The project implementation includes three phases:

- partial band capture with all-software post-processing,
- intentionally-aliased widened capture with all-software post-processing, and
- full-band capture with FPGA-based real-time processing.

The USRP hardware used in the development of this project includes the model N210 base unit and the model RFX2400 transceiver. A description of this hardware is included in Appendix A.

The PC used in the development of this project is a commodity PC with a quad-core Intel i5-2400 CPU running at 3.1 GHz. All software development and operation was performed using freely-available Fedora Linux and GNUradio[1] [15] release 3.6.

[1] Although this project concentrates on capturing and analyzing Bluetooth Low-Energy, the project work complements and extends existing GNUradio project work for Basic-Rate Bluetooth.

2.1. Partial-Band Capture With All-Software Post-Processing

The first phase of the project concentrates on developing the software-defined Bluetooth receiver chain and demonstrating the proof-of-concept.

This first phase was performed with unmodified USRP hardware, and as such had inherent limitations. Although the N210 ADCs can operate at over 100 Msps, the sample bandwidth was limited to 40 Msps (complex samples) by both the throughput of the gigabit ethernet link and the anti-aliasing filter built into the RFX2400 transceiver[2]. Therefore, only a portion of the 2.4 GHz ISM band was available for use at this phase.

Further, the PC performance limited the GNUradio-based signal processing to non-real-time. Therefore, the development proceeded by using the USRP as a partial-band capture tool, then applying the baseband processing software to the captured sample data as a post-processing step. This phase had the advantage of supporting iterative software development, where incremental development of the signal processing software could be tested with a limited number of RF capture files known to contain Bluetooth transmissions.

2.2. Intentionally-Aliased Widened Capture With All-Software Post-Processing

The second phase of the project establishes an intentional-aliasing strategy to double the number of Bluetooth channels captured. This strategy folds the ISM band into approximately 40 MHz at the expense of a 3 dB SNR degradation.

The intentional-aliasing strategy requires hardware modifications. Specifically, the RFX2400 transceiver is modified for to widen the anti-aliasing filter cutoff from 20 MHz to 41 MHz. The modifications applied to the RFX2400 are provided in Appendix B, and the details of the intentional-aliasing strategy are provided later in this report.

[2] The RFX2400 is designed to be compatible with older equipment and therefore restricts the analog bandwidth of the received signal to 20 MHz. The purpose of the filter is to avoid passing unwanted frequencies beyond the Nyquist sampling limit of these slower ADCs.

2.3. Full-Band Capture With FPGA-Based Real-Time Processing

The final phase of the project includes an update to the USRP FPGA logic to provide real-time processing of the entire ISM band. The stock USRP logic, detailed in Appendix C, provides a straightforward bidirectional DDC and DUC service between baseband and the RF passband. The modified FPGA logic includes a multi-channel Bluetooth receiver bank based on the pure-software implementation developed in the earlier phases.

Implementing Bluetooth receivers in the FPGA eliminates the throughput and performance limitations exhibited by the earlier project phases. The USRP N210 quadrature ADCs can run at full speed and resolution. When operating at the intended 96 Msps, and with the bandwidth-widening modifications to the RFX2400, entire ISM band used by Bluetooth may be captured. Further, at a theoretical worst-case where all Bluetooth channels are constantly occupied, even the maximum demands of the gigabit ethernet is easily achievable on the hardware.

This final phase providing real-time Bluetooth capture is ongoing.

3. Bluetooth Concepts

Bluetooth is the trade name for a set of standards describing a short-range wireless technology commonly used for temporarily interconnecting battery-powered devices. In classic Bluetooth, interconnected nodes form a logical structure called a *piconet*.

Bluetooth standards have been published since 1994. Newer versions of the standard include backward compatibility with earlier versions. The backward-compatible subset of Bluetooth is know as Bluetooth Basic Rate (BR), or “Bluetooth Classic”.

This project concentrates on certain newer features introduced in Bluetooth version 4.0, first published in June 2010. Specifically, the interest is in Bluetooth Low Energy (LE), which has been recently re-branded as “Bluetooth Smart”.

In some ways Bluetooth LE may be seen as a subset of Bluetooth BR. Because of this relationship, the fact that both LE connections and BR piconets must coexist, and the natural inclination of this project to be enhanced to operate on Bluetooth signals generally, this section includes pertinent descriptions of both standards. Only the lowest layers of the Bluetooth protocol stack are considered in this project.

Table 3.1: Key Aspects of Basic-Rate and Low-Energy Bluetooth

Feature	Basic Rate	Low Energy	Notes
RF Channels	79	40	2 MHz spacing in LE
Modulation Index	0.25 to 0.35	0.45 to 0.55	LE has wider signal
Max Tx Power	+20 dBm (class 1) +4 dBm (class 2)	+10 dBm	LE has no class levels Regulatory limit
Rx Sensitivity (typ.)	-85dBm	-85 dBm	Pathloss 90dB @BR Pathloss 95dB @LE
Range (typ.)	30m	50m	

3.1. Radio Frequency

Both basic-rate and low-energy Bluetooth use frequency-hopping spread-spectrum techniques. As shown in Figure 3.1, Bluetooth divides the 2.4 GHz ISM band into 79 narrowband channels at 1 MHz spacing for BR. LE assigns 40 narrowband channels at 2 MHz spacing, coinciding with every other BR channel.

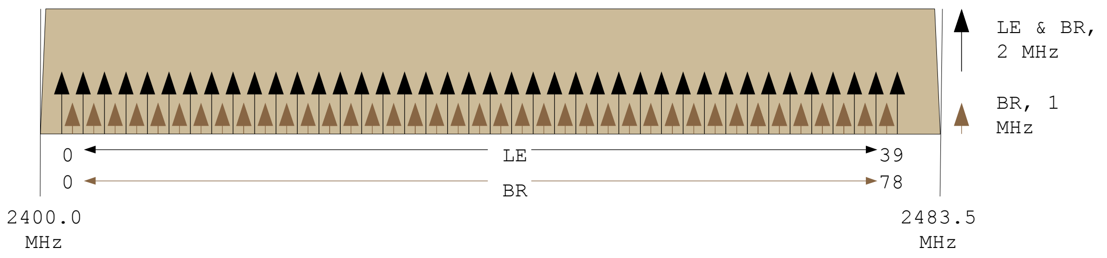


Figure 3.1: Bluetooth Radio Channels

3.1.1. Channel Hopping

Basic-rate Bluetooth hops at 800 Hz; each 1250-microsecond dwell is divided into two 625-microsecond timeslots. Each BR node maintains a synchronous timer, allowing time slots to be consistently enumerated within the *piconet*. Piconet masters begin transmissions on even slots – the first half of a channel dwell period, and slaves begin transmissions on odd slots – the second half of the channel dwell. The hop sequence always proceeds in a logical sense, however physical channel dwell is extended when transmitting packets are longer than one slot.

A given BR *piconet's* hop sequence is determined by certain parameters applied to a *selection kernel*. The parameters allow physically close *piconets* to coexist, minimizing collisions by following different hop sequences.

Bluetooth devices – both basic-rate and low-energy – may choose to operate on the full channel set, or when narrow-band interference is identified, a subset of channels. For LE, the RF channels are also reinterpreted in terms of *channel index*, which are assigned as shown in Table 3.2. A connected LE device follows a hop sequence defined by its connection parameters, including the *channel map* and the *hop increment*.

After each packet exchange, the next *channel index* is selected by adding the hop increment *mod 37*. If the resulting channel is not used according to the assigned channel map, then the RF channel is selected from the channel map *mod N*, where *N* is number of channels in the map. The RF channel advances every connection interval, which is a timing parameter provided when a connection is established.

Table 3.2: Bluetooth Low Energy Channel Indices

Channel k	**Frequency (MHz)**	**Channel Index**	**Purpose**
0	2402	37	Advertising
1 to 11	2404 to 2424	0 to 10	Data
12	2426	38	Advertising
13 to 38	2428 to 2478	11 to 36	Data
39	2480	39	Advertising

3.2. Modulation

Both basic-rate and low-energy Bluetooth utilize gaussian minimum-shift keying (GMSK). GMSK is a form of continuous-phase modulation where a gaussian pulse shape is applied to the modulator with each data symbol. In Bluetooth, the symbol alphabet is M=2, so each symbol corresponds to either a positive-going or a negative-going gaussian pulse. The pulses appear as frequency deviations applied to the carrier.

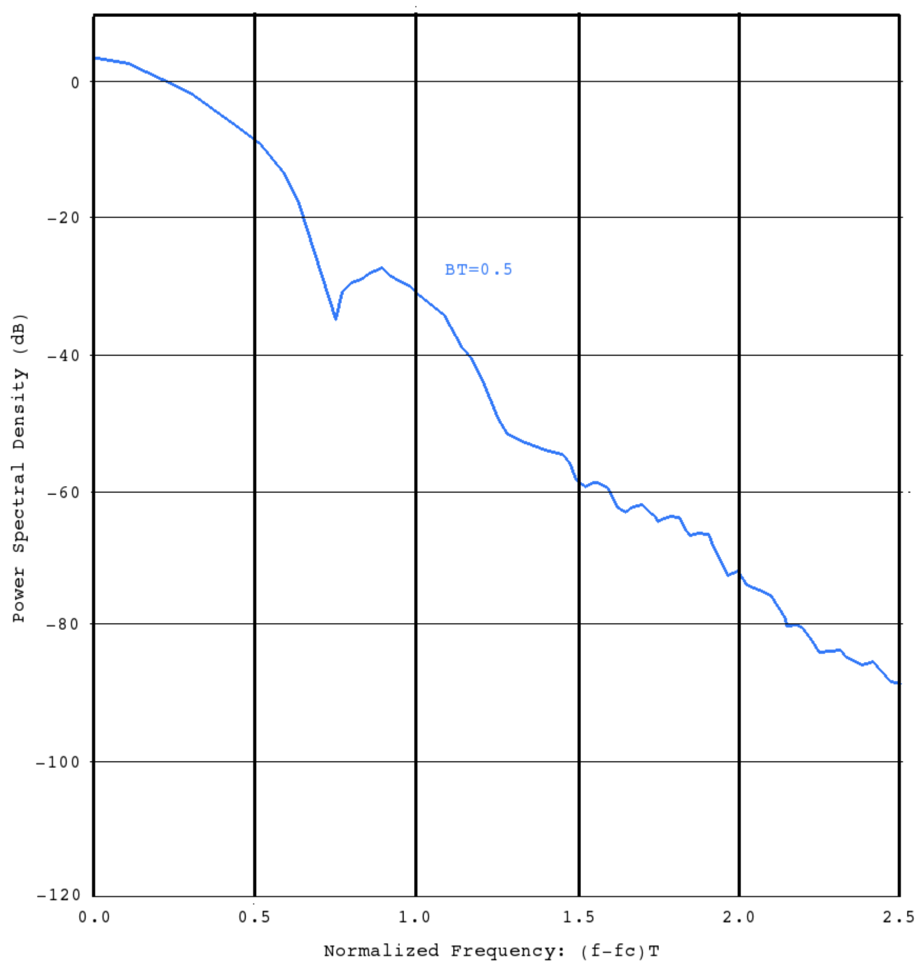


Figure 3.2: Power Spectral Density Of GMSK with BT=0.5, h=0.5

Although both Bluetooth standards discussed in this report use GMSK, the modulation index varies between BR and LE. For LE, the nominal modulation index is 0.5, resulting in a peak frequency deviation ±250kHz for the 1Mbps data stream. For BR, the nominal modulation index is 0.3, resulting in a peak frequency deviation of ±150kHz. That gaussian pulse shape is the same for both BR and LE; it is determined by the so-called bandwidth-time product (BT). The nominal BT value for Bluetooth is BT=0.5.

The power-spectral density of the LE baseband signal, shown in Figure 3.2, reveals that about 99% of the power within 650 kHz of the carrier [3]. Conversely, the BR spectrum is narrower with 99% of the signal power with 430 kHz [14].

3.3. Bitstreams

Bluetooth packet formats determine the structure necessary to decode and analyze packets from a captured symbol stream. Only well-formed symbol streams can be considered valid Bluetooth packets.

In general, Bluetooth bitstream processing occurs as shown in Figure 3.3. Prior to transmission as a symbol stream, portions of a Bluetooth packet are augmented (sometimes optionally) with error-detection codes, encryption, whitening, and in the case of BR also FEC encoding[3]. The opposite activities are performed at reception.

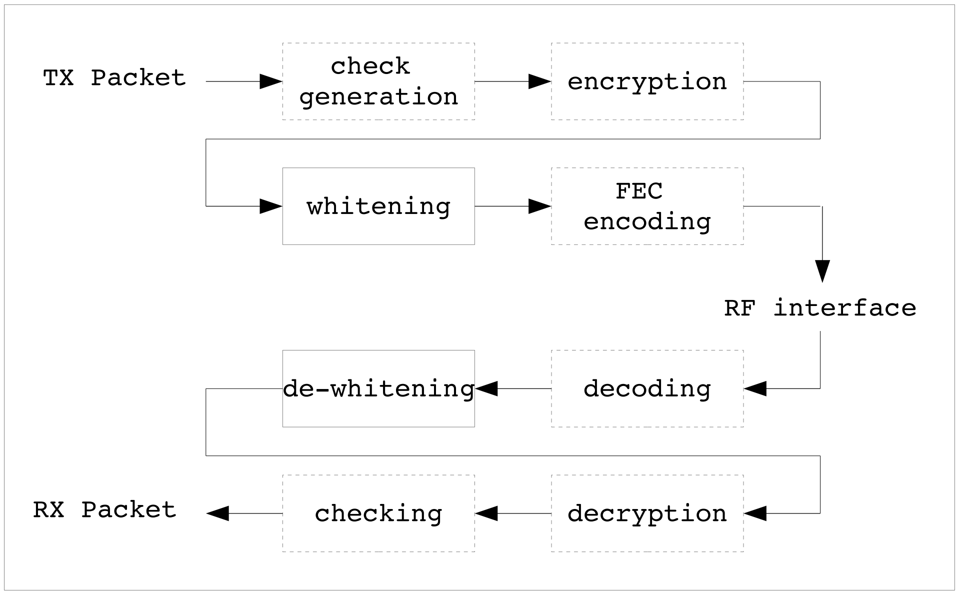


Figure 3.3: Bluetooth Bitstream Processing

One other common aspect of both BR and LE Bluetooth devices is addressing. Each Bluetooth device is assigned a 48-bit Extended Unique Identifier (EUI) by the manufacturer. The upper 24 bits of EUIs are called the Orgnaizationally Unique Identifier (OUI) and assigned to each manufacturer by the IEEE. .The lower 24-bit portion is uniquely assigned by the owner of the OUI assigned to the device.

[3] For forward error correction, basic-rate Bluetooth uses a simple R=1/3 repetition code.

3.3.1. *Whitening*

Similar to techniques common to several digital wireless standards, Bluetooth attempts to balance the the transmitted symbols so the average symbol is zero (minimizing the DC bias). This is generally accomplished with whitening, which applies to packet headers and payloads. Whitening occurs prior to encoding and transmission.

The whitening bit stream is generated with an LFSR and XORed with the packet bits. For BR, the whitening LFSR is initialized with a value derived from the master Bluetooth clock. With LE, the LFSR is initialized with a value derived from the RF channel used for transmission. The LFSR generator polynomial is the same for both standards.

3.3.2. *Basic Rate Packets*

The Bluetooth basic-rate packet format is shown in Figure 3.4. The smallest possible BR packet has only an access code.

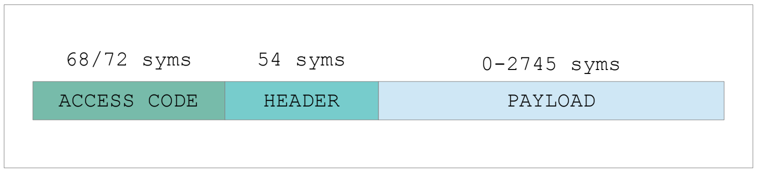


Figure 3.4: Bluetooth Basic Rate Packet Format

Reception is primarily determined by the access code, which has the structure shown in Figure 3.5. There is no data check, encryption, or whitening applied to the BR access code. The preamble and barker portions of the access code are of a fixed pattern determined by the LSB of the sync word and the MSB of the LAP, respectively. The trailer is also of a fixed pattern, but is not present on the shortest BR packets.

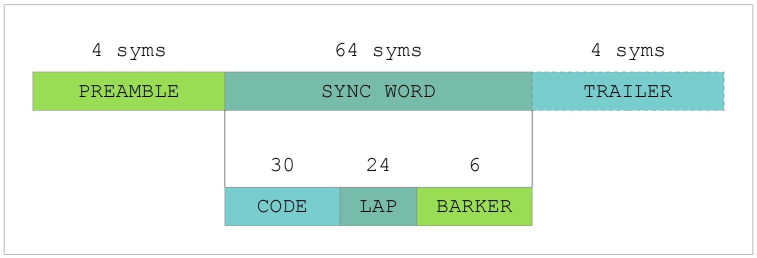


Figure 3.5: Bluetooth Basic Rate Access Code

Basic-rate Bluetooth constructs the sync word from the lower 24-bits of the destination device EUI which is termed the Lower Address Portion (LAP). LAPs are typically assigned sequentially by a device manufacturer and therefore are not inherently free of DC bias. Further, for power-efficiency reasons, a particular device will want to avoid processing Bluetooth packets not addressed to its LAP, so detection in a Bluetooth device involves correlating the received access code against its own LAP. To assist this process and reduce DC bias in the access code, a 30-bit code block is prepended to the LAP and XORed with a PN-sequence. The resulting access code provides low DC bias, good autocorrelation properties, and guarantees a minimum Hamming distance of 14 between different LAPs.

If a Bluetooth receiver accepts the access code, the remainder of the packet is also processed,. with the header and payload handled separately.

The packet header is first decoded from 54 symbols to 18 bits (it was encoded with a simple R=1/3 repetition code). The resulting bits are de-whitened. Part of the 18-bit header is an 8-bit Header Error Check (HEC) field. The HEC is generated with an LSFR applied to the 10 header bits, but initialized with a value dependent on the state of the transmitting device.

The BR packet payload is not discussed in detail within this report. The content of the payload is highly variable and contains fields specific to the particular transaction being received. The main factor considered in this project is the 4-bit packet type code from the header, which determines how many 625-symbol slots are occupied by the entire packet including the payload. A receiver must recover payload symbols across all occupied slots, and can be guaranteed that for the given *piconet*, no new packet will start until the next free slot.

3.3.3. Low Energy Packets

The Bluetooth Low-Energy packet format is shown in Figure 3.6. The same packet format is used on all channels, but the primitives supported in the PDU field differ for transmissions on the advertising RF channels versus data RF channels. Whitening applies to the PDU and CRC fields.

The first 56 symbols of the LE packet hold particular significance in this project. The preamble and certain sub-fields of the PDU header have a fixed structure and therefore form the basis of decoder match rules. The preamble is an alternating sequence of zeroes and ones, with the last preamble symbol the opposite of the first access address symbol. For packets on advertising channels, four (de-whitened) header symbols are always zero, and for packets on data channels, six (de-whitened) header symbols are always zero.

The access address is also important to packet analysis since it identifies the LE connection to which the packet belongs. On advertising channels, the AA is a fixed value and therefore may be used by a matching packet decoder.

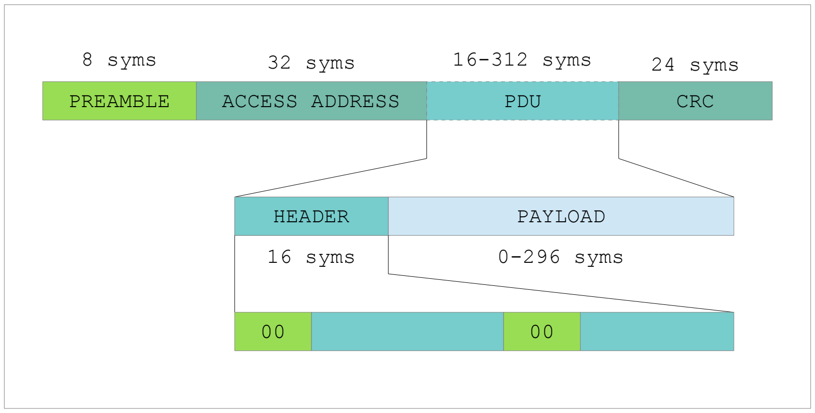


Figure 3.6: Bluetooth Low Energy Packet Format

In contrast to the Bluetooth basic-rate packets, there is no forward error correction applied to any LE packet fields. Consequently, the CRC field is useful for a binary determination that one or more decoded symbols are in error, but there are no mechanisms available to correct those errors at the receiver. According to the Bluetooth standards, packets with bad CRC are used to make higher-level decisions about re-transmission and connection quality, including the ultimate determination of connection loss.

3.4. Connections

3.4.1. Basic Rate Piconets

A full description of Bluetooth Basic-Rate *piconets* is beyond the scope of this report. However, a brief overview of BR *piconets* is given here to provide context for the environment in which LE capture must operate..

A BR device will function as the master of several slaves, which together form a *piconet*. The master provides the clock and frequency hopping pattern for all slaves. Although all slaves follow the master's hop pattern, Bluetooth considers a physical link to exist between each slave and the master node. There is no direct communication between *piconet* slaves.

The BR *basic piconet channel* is characterized by the hop sequence, which in turn is determined by the access code in use. The *piconet* access code is derived from the master EUI. When both slave and masters communicate on the same RF channel during a single dwell, the channel is known as the *adapted piconet channel*. Transmissions on these channels are directed to *access codes* specific to the *piconet*. Different access codes, other than the predefined set mentioned below, represent different *piconets*.

In order to support new BR devices forming new *piconets* or connecting to existing ones, RF channels sometimes function as an *inquiry scan channel* or a *page scan channel*. The access codes used on these channels are fixed by the Bluetooth specification, and dictate hop patterns (and longer dwell times) that differ from the active *piconets*. All Bluetooth devices know these fixed access codes and hop patterns, and given the relaxed timing, can find and connect to other devices within a short, bounded time period.

3.4.2. Low Energy Link Layer States

The scope of legal Bluetooth low-energy transmissions are determined by the state of the devices involved. Since we are interested in capturing all Bluetooth low-energy transmissions and determining the higher-layer structure of the exchanges, it is necessary to classify the potential transmissions into the LE device context. The first level of classification is provided by the LE link-layer states which are shown in Figure 3.7.

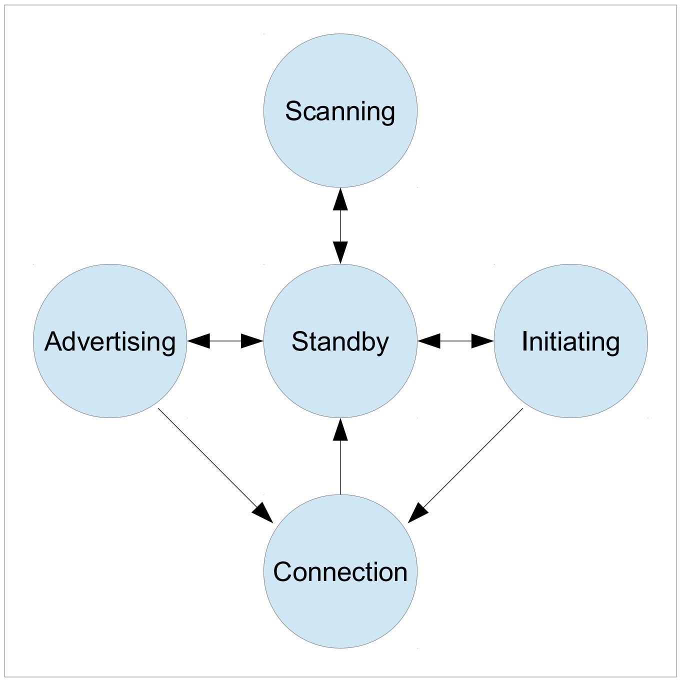


Figure 3.7: Bluetooth Low-Energy Link Layer State Diagram

No transmissions occur in the *Standby* state, so we restrict our attention to the *Scanning*, *Advertising*, *Initiating*, and *Connection* states. The *Scanning*, *Advertising*, and *Initiating* states produce transmissions that are restricted to the RF advertising channels, and contain a Protocol Data Unit (PDU) with an advertising-specific format. Transmissions by devices in the *Connection* state occur on data RF channels, and contain a PDU with a different, data-specific format.

The Access Address (AA) field, common to all transmissions, is also dictated by link-layer state. For advertising channel transmissions, the AA is a fixed value of 8e89bed6h. For data channel transmissions, the AA is a value uniquely determined when the devices enter the *Connection* state. Consequently, a capture of arbitrary LE traffic will not have access to AA values in use, but instead have to learn them.

3.5. Complementing Standard Diagnostics

The Bluetooth standards group have formally defined a mandatory interface between Bluetooth basic-rate and low-energy adapter chips and the host system: the so-called Host-Controller Interface (HCI). The HCI defines commands that cover transactions between the baseband and link-layer entities held within the adapter chip and the host layers. The HCI traffic will therefore be either directly reflected by activity on the RF channels, or reflective of semantic details related to those transmissions.

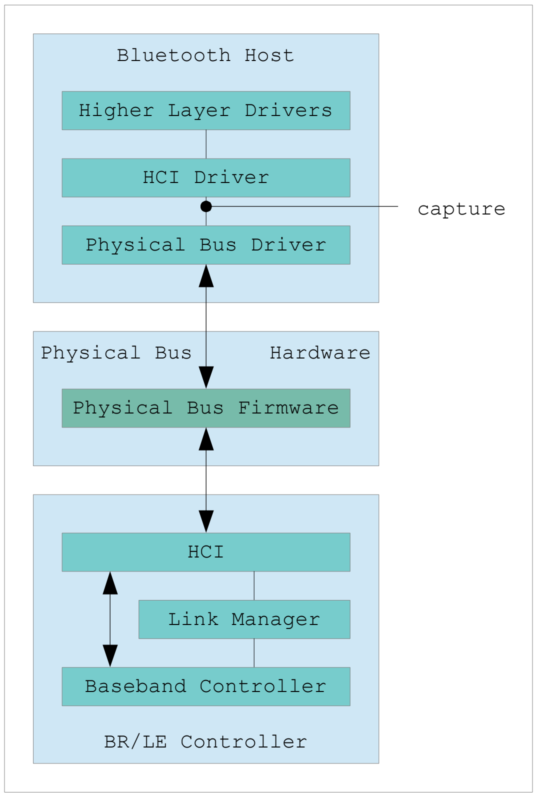


Figure 3.8: Bluetooth Host Controller Interface Capture Diagram

Figure 3.8 is a diagram adapted from the Bluetooth specifications, and shows the flow of data between a host system and an adapter chipset. Operating systems allow the traffic across the HCI to be captured and presented for analysis. A popular tool for visualizing the Bluetooth HCI data flows is Wireshark [18].

One desired feature of the present project is to provide data captured at RF in a format such that is may be viewed and interleaved alongside HCI captures. As such, the lower-level analysis of captured Bluetooth traffic is accomplished by the developments made in this project, with more sophisticated higher-level analysis (e.g. security analysis) applied by the end user, leveraging existing tools like Wireshark.

4. Capture and Analysis Techniques

The project described in this report applies wide-band capture techniques and performs analysis of the captured Bluetooth low-energy packets. For the purposes of research, product development, and security analysis, the most interesting LE packet exchanges occur on data channels when devices are in the *Connection* state. Because a capture and analysis session may commence with LE devices in arbitrary states, we must deal with the obvious situation where the captured traffic covers devices which are already connected.

In a fully-controlled test environment, it is possible to begin the Bluetooth packet capture prior to device connection. Specifically, this provides the opportunity capture the CONNECT_REQ packet exchange for a particular LE device, and parse the packet to determine values for connection information, including:

- the access-address (AA),
- the cyclical redundancy check initializer (CRCInit),
- the transmit window size (WinSize),
- the transmit window offset (WinOffset),
- the connection interval (Interval),
- the connection slave latency (Latency),
- the connection supervisor timeout (Timeout),
- the channel map (ChM), and
- the hop increment (Hop).

Whenever a CONNECT_REQ is parsed, the connection information is used for ongoing traffic analysis. However, when only data packets are available, certain techniques can be used to determine the parameters. The determination of these parameters is described in this section.

It is noted that the wide-band approach with its exhaustive capture does not *require* any connection parameters beyond the access address so long as RF conditions are good. However, when RF conditions are poor, missed packets cannot be predicted without knowledge of the Interval, Hop, and ChM parameters. The present project is intended to include the capture of valid packets, invalid packets to a particular tolerance, and a recognition of missing packets.

4.1. Narrow-band Versus Wide-band Capture

This project features exhaustive, wide-band RF capture. However, a narrow-band Bluetooth capture strategy is also possible. Inexpensive narrow-band techniques use a commodity 2.4 GHz transceiver chipset to capture one Bluetooth channel at a time. While a commodity USB Bluetooth adapter has the hardware capable of implementing a narrow-band Bluetooth capture tool, such an implementation is hindered by the limitations of the mandatory HCI. Nonetheless, custom hardware based on highly-integrated radio chips are possible: an example of a product using the narrow-band technique for Bluetooth capture is the Ubertooth [16].

Narrow-band gear like the Ubertooth is capable of following either a single BR *piconet*, or a single LE connection, but not both. The capture technique requires a learning phase prior to the commencement of frequency hopping. Learning occurs by camping on a single channel, collecting sufficient information to determine the BR *piconet* or LE connection parameters, then following the learned hop sequence along with the Bluetooth devices under analysis.

The narrow-band approach is not used in this project because of the described limitations. Wide-band techniques do not have a mandatory learning phase, and can capture and analyze multiple BR piconets and LE connections simultaneously.

4.2. Qualified Packet Decoding

Bluetooth low-energy packet decoding relies on correlating the recovered channel symbols against its known format. In this project, 56 LE symbols are inspected, of which a subset are used for the matching correlation.

Wide-band techniques have the advantage of processing all Bluetooth channels simultaneously, but have the disadvantage of accepting arbitrary access addresses. Only the LE preamble and a small proportion of the packet header is available with a predictable structure, so the decoder operates on only 15 of each 56 symbols recovered, leading to significant rates of non-Bluetooth signals being decoded as candidate Bluetooth LE packets. Two strategies are employed to mitigate these false-positive LE decodes: squelch and access-address classification.

4.2.1. Squelch

One strategy to reduce the false decodes is squelch action applied to each baseband channel. Two forms of squelch are used in this project: power-squelch and SNR-like squelch.

The power squelch is simply an absolute power threshold that qualifies the packet decoder. If the estimated in-band power for the channel does not exceed a programmable threshold, for example -80 dBm, then the packet decoder is disabled. Because local conditions predominate the ISM band noise floor, a power-squelch threshold must be manually adjusted. This manual adjustment can be difficult to determine and update over time, making the value of power-squelch limited in the exhaustive capture situation.

This project also includes an SNR-like squelch operation. The SNR of each baseband channel is estimated by computing the energy within the main baseband signal and at a narrow band-pass at specified offset from the carrier. The band-pass filter is located based on the spectral shape of the Bluetooth baseband of Figure 3.2. The SNR estimate thus provides a measure of how “Bluetooth-like” the channel looks. The packet decoder for a particular channel is disabled until the SNR estimate exceeds a threshold, say 15 dB[4]. When accepting arbitrary access addresses, the SNR-like squelch operation at 15dB reduces false-positive LE decodes by about 45 percent[5].

[4] The target BER for Bluetooth Low-Energy is 10^{-3} or better (LE is uncoded). A practical SNR-like squelch threshold may be selected based on Figure 3 of [7].

[5] In a one-second sample run, no-squelch operation produced 369 false-positive decodes, and with a 15-dB SNR squelch threshold, only 202 false-positive decodes.

4.2.2. Access Address Filtering

A Bluetooth low-energy device typically includes a receiver with a programmable white-list of access addresses[6]. This white-list is updated through the normal course of operation, including the transition into the *Connection* state. The purpose of the white-list is to improve battery life by limiting the packet decoder to processing only white-listed packets. The present project, being a permanently-powered capture tool, has no requirement for extensive battery life.

The packet decoder used in this project has similarities to actual LE devices. Because the LE access address for advertising channels is a fixed-constant value, the decoder used in this project can operate on 44 symbols of each 56 used in the packet matching operation. The same applies for any known AA values used on data channels. This provides such a powerful decoder restriction, it becomes necessary to relax the packet match operation to accept small but non-zero Hamming distances so that marginally-impaired packets are still passed by the decoder for analysis.

Only 15 of 56 symbols are used for packet matching with arbitrary access addresses on data channels, so these are handled differently. Although AA values used on data channels are randomly-generated, they are subject to certain restrictions which allow some packets to be rejected due to illegal access address. An exhaustive analysis demonstrates that of the 2^{32} possible AA's, approximately one third are not legal and may be discarded when received[7], or at least contribute to the matching metric calculation. Filtering out radically malformed access addresses reduces the amount of work required to learn valid access addresses for already-connected LE devices. However, despite the reduction of possible access addresses, the number of AA candidates is still vast.

When CONNECT_REQ packets are parsed for particular LE connections, or when the AA is otherwise restricted by the user explicitly providing values, this project allows packet decoding to be limited to those packets matching the desired access

6 For example, one vendor's LE receiver chip provides registers for programming up to sixteen Access Addresses. The receiver will only activate and decode packets where the AA matches one of the register values. The registers form a white-list.

7 An exhaustive software analysis reveals that of 4294967296 possible 32-bit Access Address values, 2900629660 (67.5%) are acceptable and 1394337636 (32.5%) are invalid.

address, and those with small 56-symbol Hamming distances. However, in the general case, the LE packet capture engine must accept all valid access addresses.

4.3. Devices Already Connected

While it is true that a wide-band capture strategy provide exhaustive coverage, inferring the participants in LE connections still requires decision logic to extract the connection parameters. A number of the steps presented below may be employed by the wide-band analysis in this project, or equally applied to the narrow-band techniques used by others.

As Bluetooth Low-Energy packets are recovered from the decoder, each access address encountered is recorded along with a timestamp, indicating a candidate *transmit window* for the connection. Only the most-recent 32 seconds of candidate *transmit windows* are considered, since by the published standards, LE connections are considered lost after this maximum time.

4.3.1. Learning Access Addresses

If an LE access address is seen at least three times with the 32 second inspection window, it may be considered an active connection. Whenever an AA has not been seen for 32 seconds, the connection associated with that AA is flushed from active consideration.

4.3.2. Timing Interval, Hop Increment and Channel Maps

If the full channel map is assumed, determining the LE connection Interval and Hop parameters are straightforward. The timing interval between subsequent candidate *transmit windows* - packets bursts spaced at least 7.5 ms apart, but with the same access address – determine the *Interval* parameter to the nearest multiple of 1.25 ms. The *Increment* parameter is the difference between two subsequent channel indices modulo 37.

Because packets may be lost due to the RF conditions present at the capture device, a sequence of at least three packet bursts with equal spacing in time and frequency hop is necessary to determine the connection timing interval and hop increment with a full channel complement.

In the general case, the full channel map cannot be assumed. Therefore, a full determination of the channel map is accomplished by inspecting 37 consecutive connection *transmit windows*. The RF channels encountered within the 37 consecutive windows form the channel map. This channel-map confirmation can require a significant capture time, in the order of minutes, since an LE hop *Interval* can be as large as 4.0 seconds.

4.3.3. Cyclical Redundancy Check Parameters

The CRC portion of each recovered packet may be used to determine whether the packet contains bit errors. This is only possible if the *CRCInit* parameter is known, since it is used to initialize the LFSR applied to the CRC operation.

The CRC initializer can be determined by reversing the LFSR normally applied to a PDU. For each packet captured in the candidate *transmit windows*, a candidate *CRCInit* parameter is generated. Since any arbitrary captured packet may or may not contain bit errors, confidence in the CRC reversal must be built up by repetitive determination of the same *CRCInit* parameter. In the present project, when the same candidate parameter is seen at least four times, it is taken to be correct *CRCInit* value.

It should be noted that the CRC matching operation is strictly an analysis feature of this project, used to identify impaired packets rather than reject them.

5. All-Software Multi-Channel Bluetooth Receiver

The first phase of the present project consists of a capture of a portion of the ISM band to a file, with post-processing using the GNUradio framework. The implementation is published on the internet [17] under open-source license.

5.1. Sample Acquisition

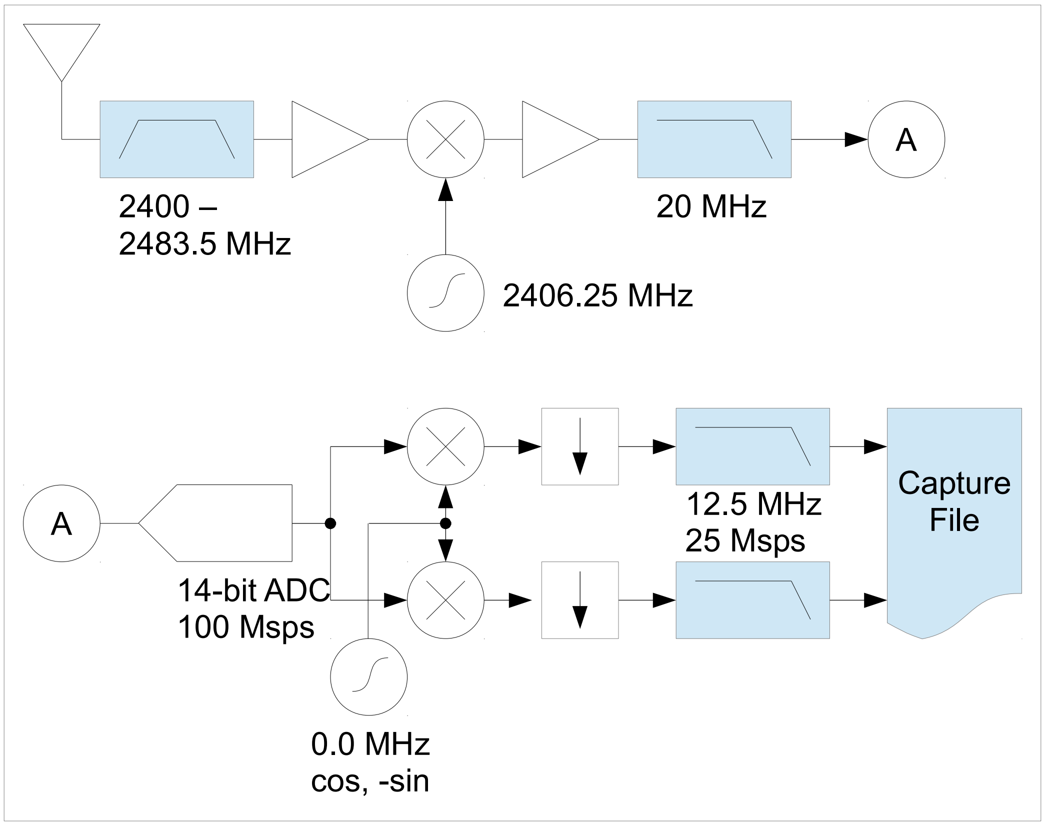


Figure 5.1: Sample Acquisition Block Diagram (Unmodified RFX2400)

The signal flow applied by the USRP hardware during a capture session is shown in Figure 5.1. The RFX2400 module covers the analog portion from the antenna to point A, and the USRP N210 main-board covers processing from point A to the sample capture file.

The RFX2400 module includes a SAW filter, limiting the receiver spectrum to the 2.4 GHz ISM band. RF is mixed with a frequency-agile quadrature local oscillator, selecting the centre of the pass-band. The quadrature mixer includes an anti-aliasing low-pass elliptical filter.

From point A, the USRP samples the quadrature analog signal to 14-bit digital samples at 100 Msps. A digital down-converter implemented in the USRP FPGA provides a decimated stream at a selectable rate. For the examples provided in this report, a 25 Msps capture is used. The capture stream is transferred across the USRP's gigabit ethernet port to a PC that records the sample series to a capture file.

5.2. Packet Recovery

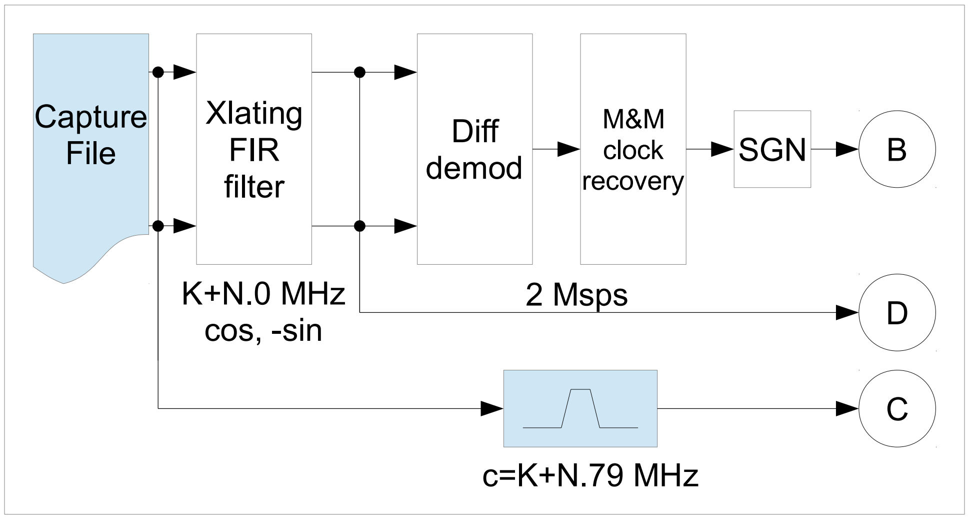


Figure 5.2: Bluetooth Symbol Recovery and Off-Channel Bandpass Filter

This section provides an overview of the Bluetooth packet recovery performed at the first stage of this project. Detailed descriptions of the various signal-processing sub-blocks are provided in further subsections. The implementation is accomplished by applying existing modules from the GNUradio toolkit and new modules developed explicitly for this project.

Bluetooth packets are recovered with a software post-processing stage applied to the capture file. The signal processing flow is described in two major parts. Figure 5.2 shows symbol recovery for the Bluetooth channels represented by the capture file, and Figure 5.3 shows the symbol decoding and squelch operations.

Bluetooth packet recovery proceeds for all RF channels covered by the capture file. A frequency-translating, decimating FIR filter provides a 2 Msps sample stream for the channel with a 3 dB bandwidth of 500 kHz, shown at point D. At the same time, a FIR band-pass filter provides a sample stream that is offset by 790 kHz from the channel centre, shown at point C. The on-channel samples are then converted to symbols (phase differences) with a differential demodulator. A Mueller & Müller clock-recovery block aligns the demodulated symbols to the centre, and a final hard-decision block provides recovered symbols at point B.

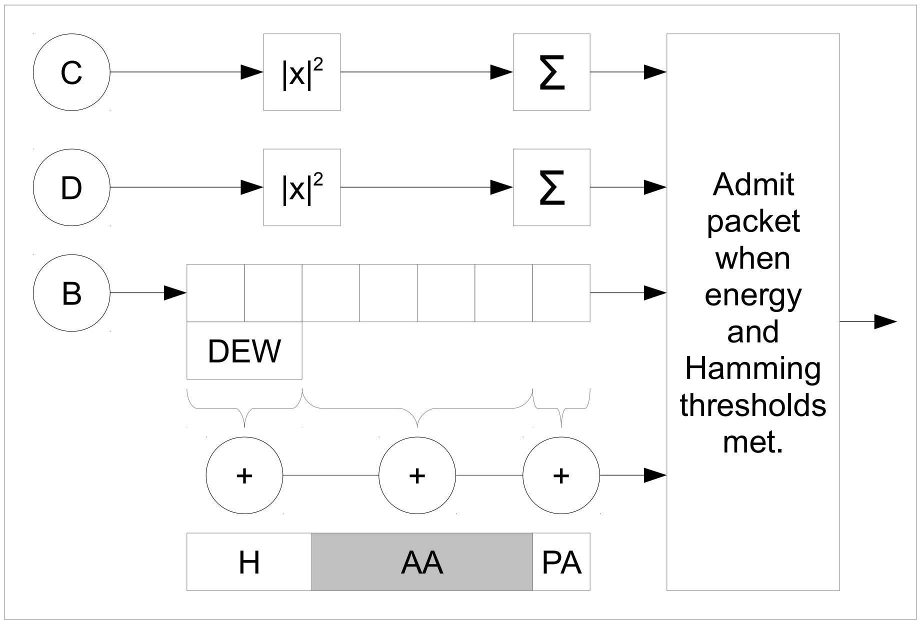


Figure 5.3: Bluetooth Low-Energy Squelch and Symbol Decode

The data recovered at points B, C, and D are processed to determine whether a valid Bluetooth Low-energy packet is present. The packet-presence qualification includes both a signal-energy squelch component and sequence-matching component.

The squelch component computes both on-channel and 790 kHz off-channel energy and can be used to qualify the signal with a simple SNR-like measure. When the on-channel energy exceeds an absolute threshold, and the ratio of the on-channel and valley energy exceeds a configurable threshold, then the signal spectrum is considered distinct enough from wide-band noise to be a candidate Bluetooth signal.

At the same time, the recovered symbol stream is compared against fixed portions of the LE packet format. All LE packets must start with an 8-symbol preamble (PA) which consists of alternating 1's and 0's. For symbol streams recovered from advertising channels, the 32-symbol access address (AA) is also fixed and therefore used in the decode decision. The 16-symbol de-whitened PDU header is also inspected and compared with likely candidates. The decode stage results in a matching metric, primarily a Hamming distance, computed for the candidate packet. If this metric exceeds a programmable threshold, then the symbol stream is interpreted as a complete packet, and admitted to the recovery log.

5.2.1. Frequency-Translating, Decimating FIR Filter

A frequency-translating, decimating FIR filter is applied to the sample stream for every channel of interest covered by the sampling spectrum. All processing proceeds with floating-point complex values such that both input and output streams apply in quadrature. The project implementation uses an existing component from the GNUradio toolkit.

The FIR filter is low-pass with a 3 dB cutoff of 500 kHz and transition width of 300 kHz, modelled with a Hann window[8], yielding 355 taps. The decimation rate is selected to yield a 2 Msps output, based on the input sampling rate. Consequently, 2 Msps is the lowest input sampling rate supported by this project.

5.2.2. Bandpass Noise Filter

A second frequency-translating, decimating FIR filter is applied to the input sample stream to feed the SNR-based squelch mechanism. The FIR filter is low-pass with a 3 dB cutoff of 22500 Hz and a transition bandwidth of 10000 Hz, modelled with a Hann window, yielding 7751 taps. The filter applies to a pseudo-carrier that is 790 kHz

[8] Rather than through any formal process, the Hann window was selected due to its convenience in the GNUradio framework.

offset from the on-channel carrier. The filter provides the same decimation rate as the on-channel filter, resulting in a 2 Msps complex sample stream.

5.2.3. Differential Demodulator

Symbols are recovered as phase shifts determined by a differential demodulator operating in the manner shown in Figure 5.4. Refer to [5] for an explanation of the theoretical basis of operation. The project implementation modifies an existing component from the GNUradio toolkit.

Complex input samples are processed in quadrature. The resulting recovered soft symbols are floating-point values. There is no decimation performed, so the output symbol rate is 2 Msps.

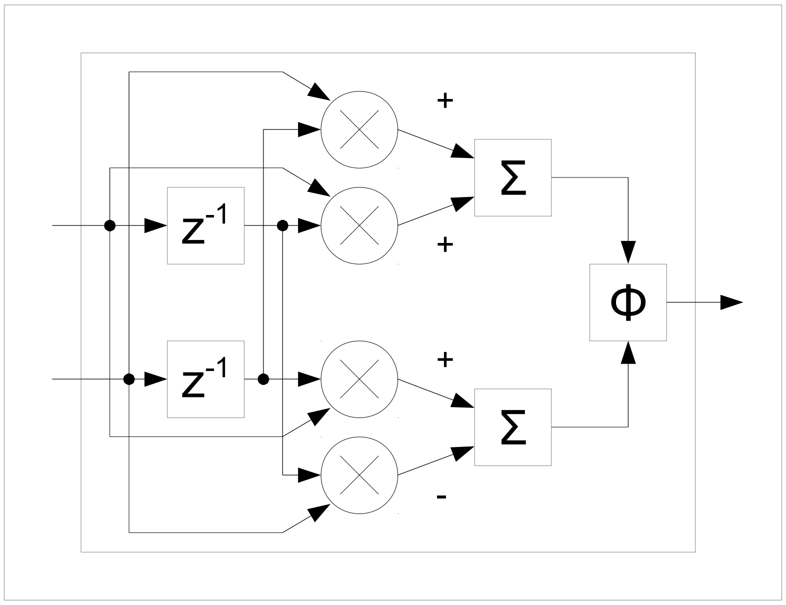


Figure 5.4: Differential Demodulator Block Diagram

5.2.4. Mueller & Müller Clock Recovery

Clock recovery proceeds with the strategy proposed by Mueller and Müller, and described in [12]. This process determines and tracks a fractional offset between samples that represents the peak excursion of the (interpolated) recovered symbols. The project implementation modifies an existing component from the GNUradio toolkit.

The interpolation is performed by an 8-tap FIR filter that allows the shape of the recovered symbol pulses to be approximated. The interpolated value is taken to be the soft symbol at the clock recovery point. The output of this operation decimates by a factor of two, resulting in a 1 Msps symbol rate. The floating-point soft symbols are then simply applied to a hard-decision mapper to yield a recovered hard symbol stream at 1 Msps. This corresponds to point B of Figure 5.2.

5.2.5. Squelch Operation

Squelch is applied as described in the previous chapter. The implementation was developed explicitly for this project.

The energy in both the on-channel samples and off-channel samples are determined by summing the the magnitude-squared for all samples in the decode window. If the on-channel energy exceeds a given threshold, and the ratio between on-channel and off-channel energies exceeds its threshold, then the squelch asserts and allows the decoder to operate.

5.2.6. Symbol Decoding

Symbol decoding proceeds as described in the previous chapter. The implementation was developed explicitly for this project.

Decoding proceeds as qualified by the squelch operation mentioned above. A 56-symbol candidate decode window is processed each time the decoder operates. The first 40 symbols are processed as received, and the next 16 symbols, representing the PDU header, are de-whitened before further processing.

The 56-symbol window is always compared against the fixed portions of the LE packet, specifically the preamble and fixed-zero portions of the PDU header. A Hamming distance calculation is performed determining the base matching metric for all candidate packets.

The matching metric is then increased by a value representative of the access address match. For advertising channels, the Hamming distance to the fixed access address is added. For data channels, the matching metric is increased by the number of offenses to the AA limitations. Data channel AA offenses are:

- the AA more than six consecutive zeroes or ones,
- the AA matches the advertising channel or one bit variations thereof,
- all four bytes of the AA are the same,
- the AA contains more than 24 bit transisitions, or
- the AA has fewer than two or more than six transitions in the most significant six bits.

Decoded packets are admitted to the output when the matching metric is less than a threshold. In this project implementation, any matching metric less than 3 allows the decoder to proceed and recover the entire packet. The remaining packet symbols are also de-whitened prior to admission.

5.3. Example Results

A number of real-world captures are used to verify the correct operation of the project implementation. Three commercial Bluetooth low-energy devices were used to generate the radio transmissions necessary for testing. The devices used are:

- Apple iPad Mini, operates as LE scanner and initiator
- Polar H7 Heart Monitor, operates as LE advertiser
- Texas Instruments CCTAG, operates as LE advertiser

Two sample runs are presented here. The captures associated with these sample runs were used to validate the signal processing blocks described above.

Each run consists of a one-second window of RF samples collected at 25 Msps, centred at 2406.25 MHz. The USRP ships with software that provides for generic RF streaming to a capture file. The sample captures were accomplished with the following command invocation

```
$ uhd_rx_cfile --gain=60 –samp-rate=25000000 –freq=2406250000
  --nsamples=25000000
```

During each capture, iPad was instructed to establish a connection with one of the advertising devices. The results for each capture are listed in Table 5.1.

Table 5.1: Bluetooth Low-Energy Packet Capture And Recovery Examples

Device	**False-positive Packet Count**	**Recovered Packet Count**	**Reported SNR (dB)**
Polar H7	37	27	22.3
TI CCTAG	33	5	34.7

An example of a recovered packet is:

```
snr=26.6, BTLE index=37, AA=8e89bed6, PDUType=4, TxAdd=0, RxAdd=0,
Length=23
AdvA=e60700d02200
  (char) ScanRspData= . . P o l a r   H 7   0 0 0 7 E 6
  (byte) ScanRspData=1009506f6c61722048372030303037453 6
```

An example of a false-positive packet is:

```
snr=15.6, BTLE index=01, AA=16aca53f, LLID=2, NESN=1, SN=1, MD=0,
Length=24
```

6. Project Extensions

This preceding project phase detailed in this report has demonstrated workable wide-band capture and post-processing strategy for Bluetooth channels. There are two primary weaknesses to the previous approach: limited capture bandwidth and limited base-band processing capacity. This section describes two extensions to the project to address these problems.

6.1. Intentional Aliasing Extension

The capture bandwidth of the N210 and RFX2400 hardware used in this project is limited by analog bandwidth of the RFX2400 (20MHz), and the bandwidth of the digital down-converter employed in the N210. The filtering bandwidth of the DDC is related to the output sampling rate.

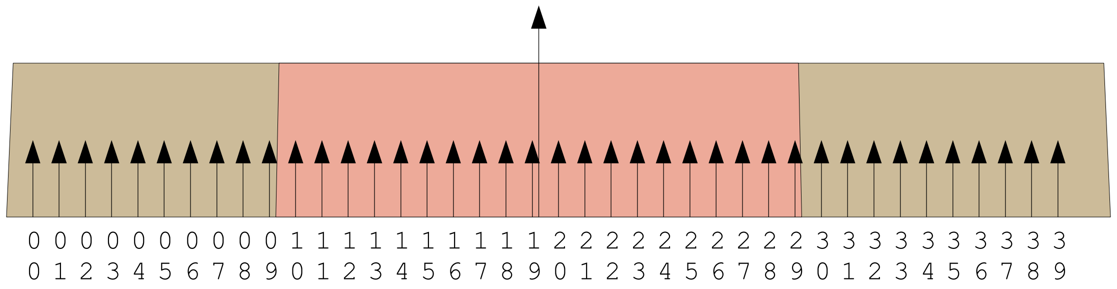


Figure 6.1: BTLE Capture with Unmodified N210 and RFX2400

As a minor extension to the project, an intentional aliasing strategy is added to allow the entire ISM band to be captured. The aliasing is based on a delivered sample rate of 40 Msps. If the receiver is tuned to mid-band at 2040.25 MHz, the analog bandwidth of the RFX2400 dominates, resulting in a capture of 2020.25 to 2060.25 MHz. This is only about half the ISM band of interest, with LE channels captured as shown in Figure 6.1

Table 6.1: Bluetooth Low-Energy Channels Under Intentional Aliasing

Channel k	Offset From Baseband Centre (kHz)
0 to 9	-750 - k*2000
10 to 29	-38250 + k*2000
30 to 39	79250 - k*2000

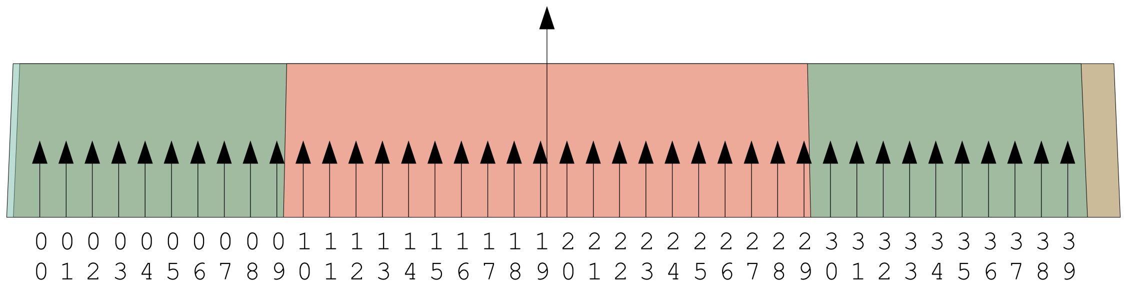


Figure 6.2: ISM Band Capture with Unmodified N210 and Modified RFX2400

When the RFX2400 receiver bandwidth is extended to 41 MHz, for example with the procedure detailed in Appendix B, the pass-band of the project hardware is extended as indicated in Figure 6.2. The analog pass-band is indicated in green, but prevented from appearing in the capture by the filtering action of the digital down-converter of the N210. With the DDC filter removed, a simple modification to the N210 FPGA, the captured base-band is organized as shown in Figure 6.3. The corresponding LE channels are located as listed in Table 6.1 The 250 kHz offset at band centre allows the aliased channels to be cleanly interleaved such that they appear with 1 MHz spacing (except near the pass-band centre. where the channels are 500 kHz apart).

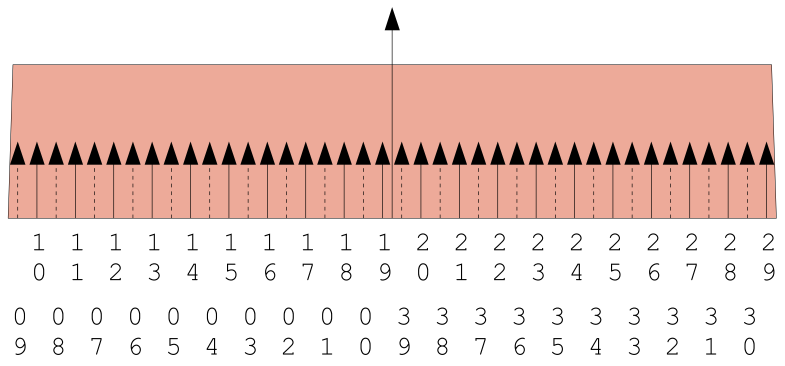


Figure 6.3: ISM Band Capture WIth Intentional Aliasing

6.1.1. *Problems With Intentional Aliasing*

Although the intentional-aliasing strategy does provide access to the full ISM band with the available hardware, it does have some negative side-effects. Because the noise power is aliased along with the desired Bluetooth channels, the capture noise level doubles compared to the non-aliased operation, so SNR is degraded by 3 dB. Also, interference from narrow-band sources that might have led Bluetooth devices to operate with a reduced channel map, can be reflected into the occupied channel range and interfere with packet reception and recovery.

Assuming a benign ISM environment, the capture of a single LE connection suffers no particular problems with the intentional aliasing approach, because with the frequency-hopping nature of the transmissions, only one narrow-band channel is in use at any time. However, the capture of multiple LE connections can suffer because the guard band between LE channels is reduced by the aliased channels. Of course, transmissions on aliased channels can only cause interference with simultaneous transmissions on "neighbour" channels in the main pass-band, where the notion of a neighbour differs from the straightforward due the aliasing operation.

While the interference due to aliasing when capturing multiple connections is difficult to quantify, it is clearly a statistical process related to the hop patterns and transmission duty cycle of the devices in use. Whatever the nature of this degradation, due to the interleaving nature of the aliasing approach, the degradation is not directly an increase in *collisions*. If two or more LE connections have aligned *transmit windows*, they will collide by transmitting on the same RF channel. Aliased channels are not aligned with the main pass-band channels, so rather than directly colliding, these "near miss" transmissions can still appear with adequate SNR for demodulation, perhaps with tighter channel filters . An interesting follow-on sub-project might be to model and verify the collision-like degradation imposed by the intentional-aliasing approach.

6.1.2. *Basic-Rate Extension*

It is noted that Bluetooth basic-rate channels are also reflected in this intentional-aliasing strategy. Where the target LE channels appear every 1 MHz in Figure 6.3, the BR channels appear on a 500 kHz raster, both aligned-with and between the LE channels. The rationale for capturing BR traffic with the intentionally-aliased approach extend naturally from the LE, but because a BR transmission occupies more than half the bandwidth of an LE transmission, it would suffer even more from the described interference degradation.

6.2. Real-Time Extension

The previous project phases described in this report provide for non-real-time operation of the Bluetooth capture and analysis tool. This section describes a project extension where the main signal processing blocks proven by the all-software implementation are included in a modified N210 FPGA load, and provide a real-time packet stream, for all Bluetooth channels simultaneously. This project extension also leverages and the analog pass-band extension provided in the previous section and detailed in Appendix B.

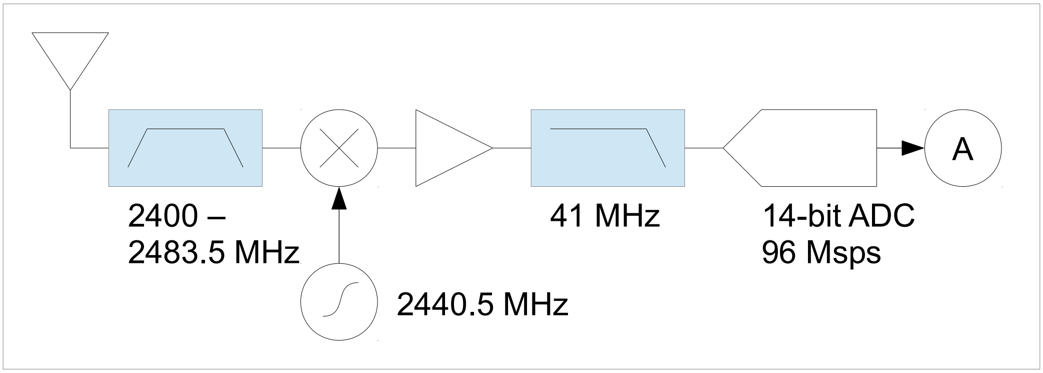


Figure 6.4: Full ISM Band RF Receiver and Digital Conversion Block Diagram

As shown in Figure 6.4, the modified RFX2400 is tuned to 2440.5 MHz, near the centre of the ISM band. With the analog pass-band of the down-converting mixer extended to 41 MHz, the entire ISM band is presented to the N210 receive path. The N210 is configured to sample the base-band analog signal at 96 Msps in quadrature. This provides a receiver chain that captures the entire ISM band occupied by Bluetooth without any need for the aliasing strategies described previously.

For Bluetooth low-energy, a bank of 40 GMSK demodulators as shown in Figure 6.5 are developed and introduced into the FPGA processing chain. The blocks making up the demodulation chain directly parallel the upstream blocks demonstrated in the all-software implementation of this project, but translated into dedicated logic in the FPGA. Indeed, as these VHDL blocks are developed, they are tested against the same sample captures used in the all-software implementation. The resulting recovered packets may be compared directly as a way of validating the VHDL implementation prior to performing real-time operation in the N210 hardware.

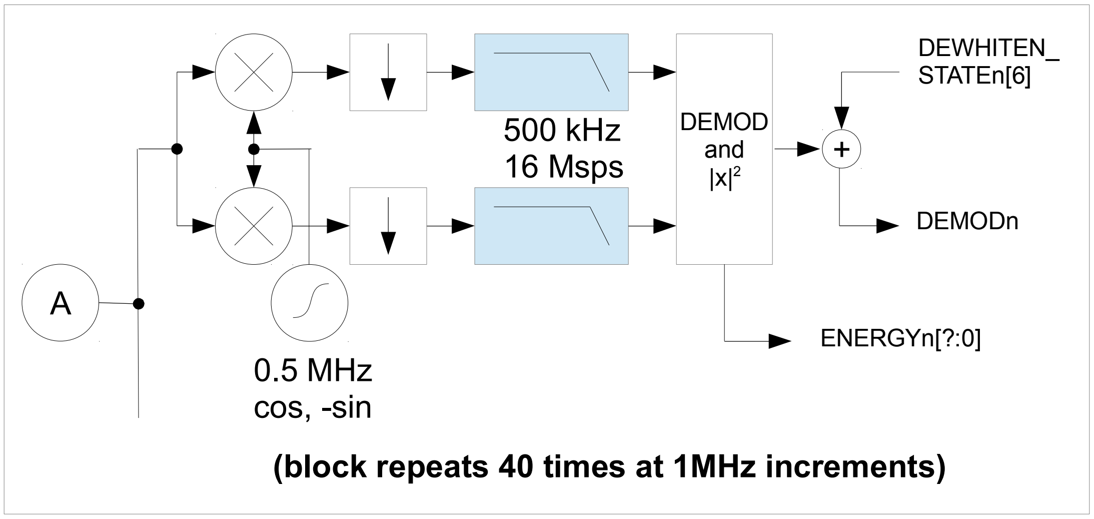


Figure 6.5: Full ISM Band Bluetooth Low-Energy Demodulators

The standard logic blocks provided by the N210, detailed in Appendix C, occupy approximately 30% of the Xilinx XC3SD3400A FPGA resources. It is expected that the remaining 70% of the FPGA resources are adequate for the logic implementation described in this section. As open-source hardware, all the Verilog sources to the standard FPGA logic are available from the manufacturer. However, the Xilinx toolchain required for targeting this high-end FPGA demands a price of $2500, which is more than the cost of the hardware used in this project. The cost of the Xilinx tools are a significant impediment to this project phase, however HDL simulation techniques may be freely used for validating the approach.

6.2.1. Frequency-Translating FIR Filter

A CORDIC-based digital down-converter is utilized in the standard N210 FPGA logic, and this module is re-used, in modified form for the frequency-translating FIR filter of Figure 6.5. The implementation includes a by-eight decimator, making the output of each of these blocks a 16 Msps complex sample stream for each Bluetooth channel.

6.2.2. Differential Demodulator, Clock Recovery, Hard Symbols

The differential demodulator described in the all-software implementation is implemented directly in logic with fixed-point word sizes instead of floating-point.

The Mueller and Müller clock recovery block described in the all-software implementation is implemented directly in logic with fixed-point word sizes instead of floating-point. The output of the clock-recovery clock, as 1 Msps, is converted to hard symbols with a simple sign threshold.

6.2.3. On-Channel Energies

Similar to the all-software implementation, the energy for each channel is accumulated with a history reflective of the 56-symbol decoder window, as magnitude-squared values.

All 40 channel energies are compared, and the minimal value is taken as the noise level of the entire ISM band. This allows the SNR-based squelch operation to proceed without requiring any extra filtering steps.

6.2.4. Decoder and De-Whitening

A single matching decoder block described in the all-software implementation is implemented in logic. The decoder compares a 56-symbol window against the fixed portions of the packet format and computes Hamming distances for them. A further Hamming distance calculation is provided in parallel for the fixed access address used on advertising channels. Also, counts are generated for offensive patterns found in access addresses on data channels. The measures are then added, and compared to a matching threshold to determine whether the packet decode should apply or not. The decoder then allows the entire packet to pass, via the de-whitener, out to the packet router shown in Appendix C. The Bluetooth LE packet length is determined from the appropriate field in the de-whitened header.

Since this decoder and de-whitener block runs once per recovered symbol for each channel, the implementation can be shared among all demodulator chains. Each demodulator block holds state variables including:

- a flag as to whether a packet is actively being decoded,
- the remaining number of symbols in the packet, and
- the de-whitener's LFSR state.

If the channel is being actively decoded, then the de-whitened symbols are shifted into a per-channel holding buffer. Otherwise, as symbols are recovered, the packet-matcher is used to determine whether a new packet decode should begin. Candidates that do not match are ignored, and the process repeats with the next symbol.

When a Bluetooth packet is fully collected, it is passed on to the packet router for forwarding to the PC host over gigabit ethernet. Packets are delivered with the industry-standard VITA format. The VITA format includes a higher-resolution timestamp already supported by the N210 logic. Fields within the new VITA ethernet packets indicate the recovered data is demodulated Bluetooth rather than raw quadrature samples.

The demands placed on the gigabit ethernet link in this latter case is a small fraction of those required for the high-speed capture used in previous phases of this project. At worst case, all 40 Bluetooth LE channels are constantly decoding packets, resulting in a 40 Mbps payload stream, compared to 40 Msps complex sample stream. Even with overhead this is only a small fraction of the gigabit ethernet link capacity.

7. Conclusions

The project presented in this report implements an open-source Bluetooth low-energy capture tool on inexpensive software-defined radio hardware. Bluetooth capture and analysis is an important tool for engineers and researchers involved in Bluetooth produced development and security. The moderate cost of the hardware – less than $2000 – allows individuals and small companies to perform active Bluetooth capture and analysis of traffic at a fraction of the cost of commercial products.

An initial all-software implementation based on the GNUradio framework is presented. The implementation is described in this report and has been published under an open-source license on the internet. The presented implementation has demonstrated that BTLE traffic can be recovered by capturing the RF traffic from a number of commercial low-energy devices.

7.1. Ongoing Work

The initial implementation has bandwidth and performance limitations, so extensions are described that allow for full band capture and real-time processing. These extensions are based on enhancements to the foundation proven in the base implementation.

The first extension involves improving the capture bandwidth to the full ISM band, by making some simple hardware alterations and updating the FPGA logic. The second extension involves porting the most computationally-intensive signal-processing blocks from the GNUradio framework into the FPGA.

This work is ongoing.

References

[1] Kuchi, K.; Prabhu, V.K., "Power spectral density of GMSK modulation using matrix methods," *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE* , vol.1, no., pp.45,50 vol.1, 1999

[2] Aulin, T, Sundberg, C-E, "An easy way to calculate power spectra for digital FM", *Communications, Radar and Signal Processing, IEE Proceedings F*, Volume 130, Issue 6, 1983

[3] Murota, K.; Hirade, K., "GMSK Modulation for Digital Mobile Radio Telephony," *Communications, IEEE Transactions on* , vol.29, no.7, pp.1044,1050, Jul 1981

[4] Lampe, L.; Jain, M.; Schober, R., "Improved decoding for Bluetooth systems," *Communications, IEEE Transactions on* , vol.53, no.1, pp.1,4, Jan. 2005

[5] Simon, Marvin K.; Wang, C., "Differential Versus Limiter--Discriminator Detection of Narrow-Band FM," *Communications, IEEE Transactions on* , vol.31, no.11, pp.1227,1234, Nov 1983

[6] Bayaki, E.; Lampe, L.; Schober, R., "Performance Evaluation of Bluetooth Systems With LDI, Modified LDI, and NSD Receivers," *Vehicular Technology, IEEE Transactions on* , vol.57, no.1, pp.157,168, Jan. 2008

[7] Ibrahim, N..; Lampe, L.; Schober, R., "Bluetooth Receiver Design Based on Laurent's Decomposition," *Vehicular Technology, IEEE Transactions on* , vol.56, no.4, pp.1856,1862, July 2007

[8] Forney, G.D., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *Information Theory, IEEE Transactions on* , vol.18, no.3, pp.363,378, May 1972

[9] Trachtman, A.; Kalet, I.; Shamai, S., "Limiter-discriminator detection of continuous phase modulation (CPM) Tomlinson filtering," *Communications, IEEE Transactions on* , vol.42, no.234, pp.819,825, Feb/Mar/Apr 1994

[10] Proakis, J.G.; Salehi, M., "Digital Modulation Schemes," in *Digital Communications*, 5th ed, McGraw-Hill Higher Education, 2008, pp 95-148.

[11] Proakis, J.G.; Salehi, M., "Optimum Receivers For AWGN Channels," in *Digital Communications*, 5th ed, McGraw-Hill Higher Education, 2008, pp 160-265.

[12] Meyr, H. et al, "The Mueller and Müller (M&M) Synchronizer" *in Digital Communication Receivers*, John Wiley & Sons, 1998, pp 86-88

[13] Valenzuela, J.L.; Hernández, A.; Valdovinos, A., "Modeling Collision and Connection Statistics in Bluetooth Networks," *Wireless Communication Systems, 2005. 2nd International Symposium on* , vol., no., pp.570,574, 7-7 Sept. 2005

[14] Emira, A.E., “Bluetooth/WLAN Receiver Design Methodology and IC Implementations”, Ph.D dissertation, D.L. Coll. of Eng., Texas A&M University, 2003

[15] Braun, M. et al, *GNU Radio* [website]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki

[16] Ossman, M., Spill, D., *Project Ubertooth* [website]. Available: http://ubertooth.sourceforge.net/

[17] Kilgour, C, *gr-bluetooth-cdk project* [website]. Available: https://github.com/whiterocker/gr-bluetooth-cdk/tree/cdk

[18] Combs, G. et al., *Wireshark* [website]. Available: http://www.wireshark.org/

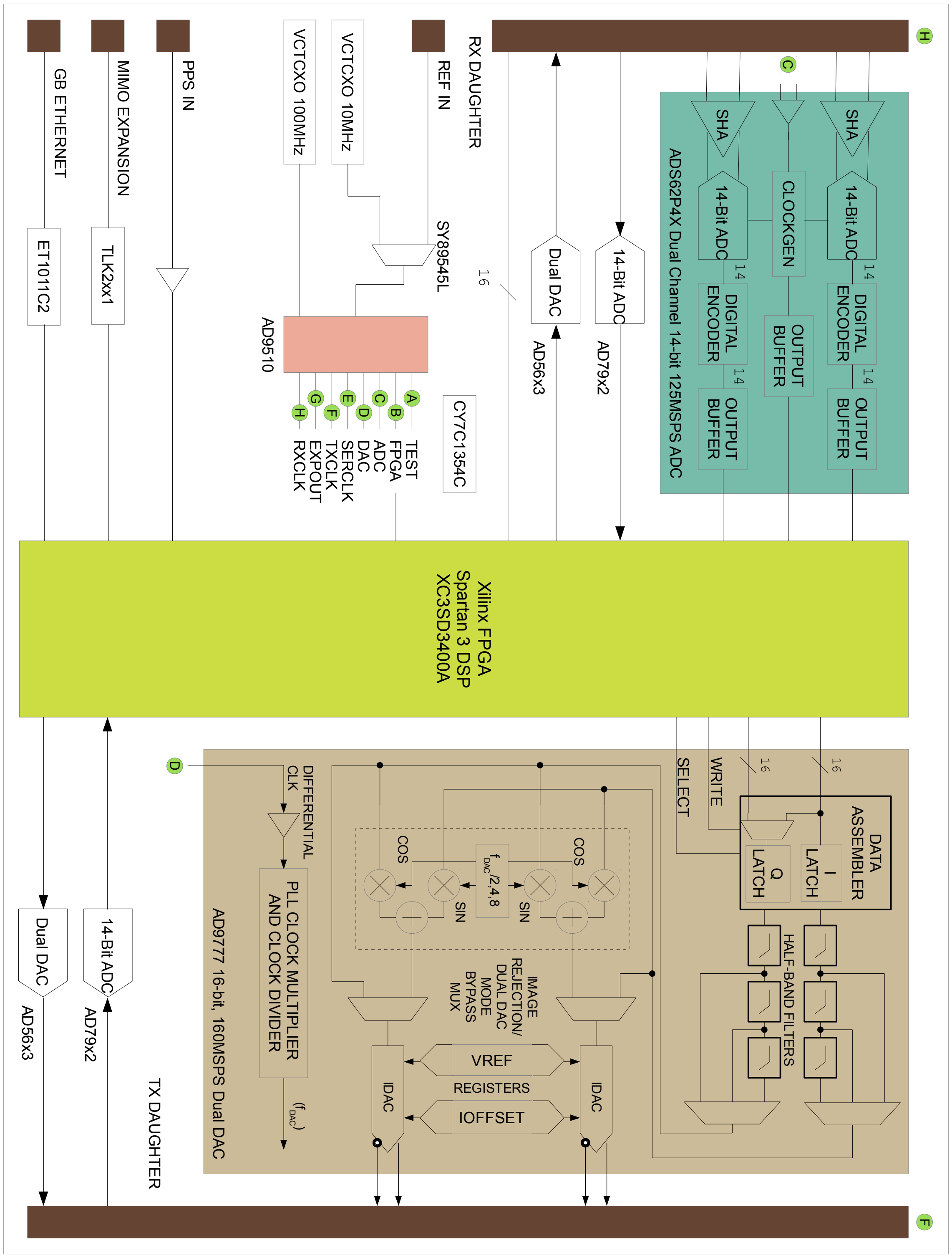
Appendices

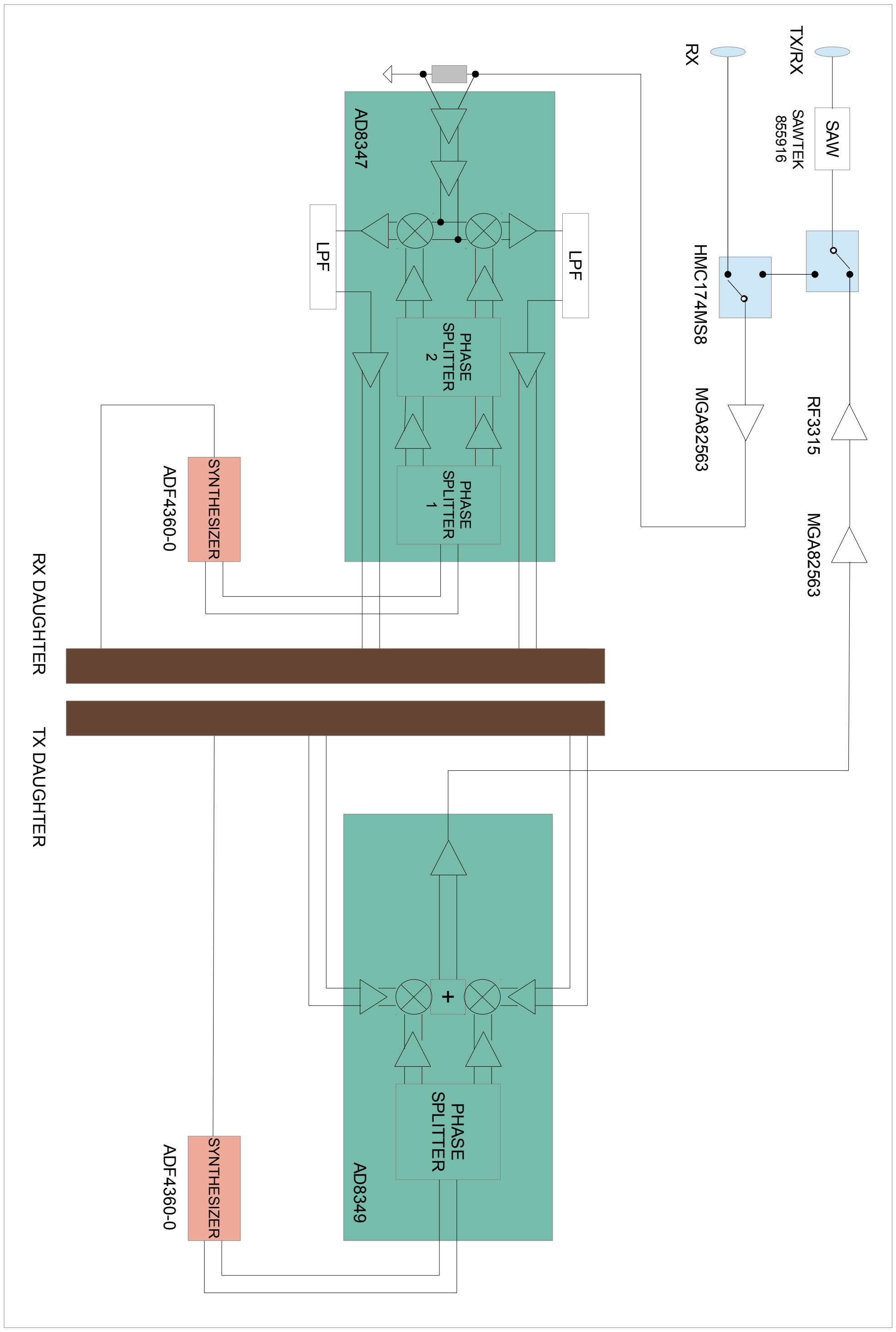
Appendix A. USRP N210 and RFX2400 Internals

The hardware used in this project is the Universal Software Radio Peripheral N210 from Ettus Research and pictured below. Populated into the N210 is an RFX2400 daughter card providing a transceiver covering the 2.4 GHz ISM band. Only the receive portion of this hardware is used in this project.

The following two block diagrams apply to the N210 and RFX2400 respectively. RF signals apparent at the antenna are down-converted in quadrature with dedicated hardware on the RFX2400 and presented to the N210 at baseband (zero-IF). The quadrature ADC on the N210 samples at 100 Msps which is fed into the FPGA. The main signal processing blocks of the N210 FPGA are described in Appendix C. The FPGA provides the resulting samples as an ethernet stream on the N210 gigabit ethernet port.

Both of these block diagrams were developed specifically for this project based on the schematics for the N210 and RFX2400. The schematics are freely available from the manufacturer.

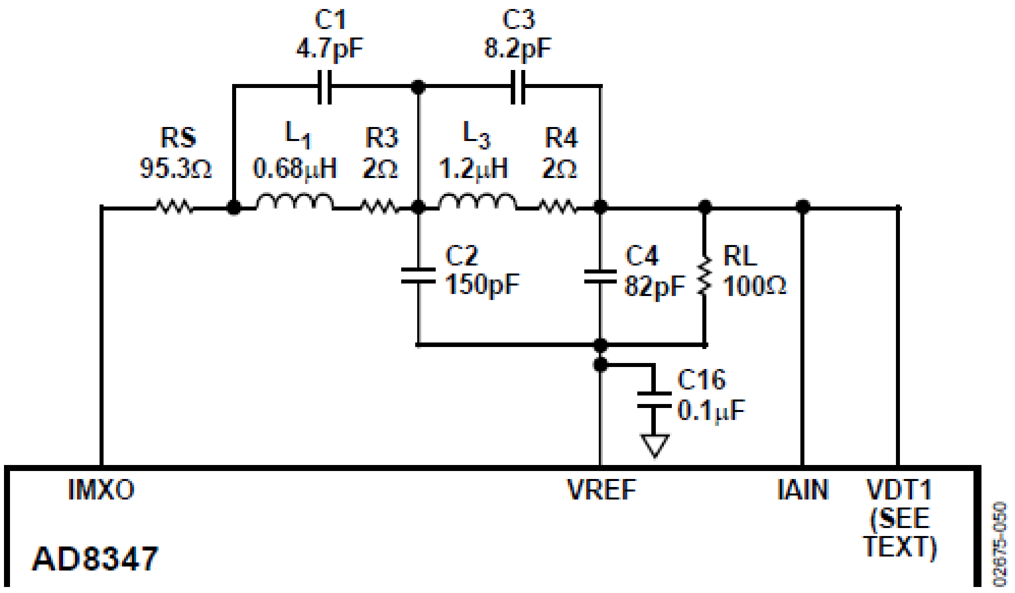

ADS62P4X Dual Channel 14-bit 125MSPS ADC
SHA
14-Bit ADC
DIGITAL ENCODER
OUTPUT BUFFER
CLOCKGEN
14
RX DAUGHTER
REF IN
SY89545L
VCTCXO 10MHz
VCTCXO 100MHz
AD9510
TEST
FPGA
ADC
DAC
SERCLK
TXCLK
EXPOUT
RXCLK
CY7C1354C
14-Bit ADC
AD79x2
Dual DAC
AD56x3
16
PPS IN
MIMO EXPANSION
TLK2xx1
GB ETHERNET
ET1011C2
Xilinx FPGA
Spartan 3 DSP
XC3SD3400A
SELECT
WRITE
DATA ASSEMBLER
I LATCH
Q LATCH
HALF-BAND FILTERS
DIFFERENTIAL CLK
PLL CLOCK MULTIPLIER AND CLOCK DIVIDER
COS
SIN
$f_{DAC}/2,4,8$
IMAGE REJECTION/ DUAL DAC MODE BYPASS MUX
VREF
REGISTERS
IOFFSET
IDAC
(f_{DAC})
AD9777 16-bit, 160MSPS Dual DAC
TX DAUGHTER
A
B
C
D
E
F
G
H


TX/RX
RX
SAW
SAWTEK 855916
HMC174MS8
RF3315
MGA82563
MGA82563
AD8347
LPF
LPF
PHASE SPLITTER 2
PHASE SPLITTER 1
SYNTHESIZER
ADF4360-0
RX DAUGHTER
TX DAUGHTER
+
PHASE SPLITTER
AD8349
SYNTHESIZER
ADF4360-0

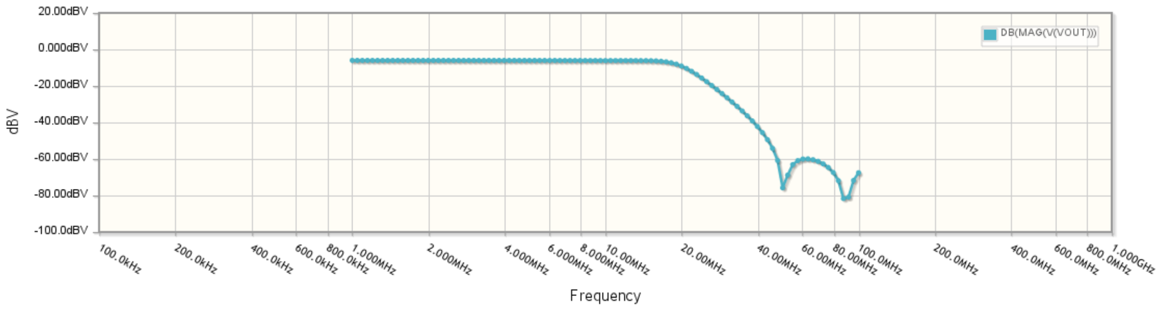
Appendix B RFX2400 Modifications

To support the intentional aliasing and real-time full-band operation provided in this project, the RFX2400 anti-aliasing filter is modified. The RFX2400 was designed to operate with earlier-generation SDR hardware employing lower-rate ADCs and therefore ships with a relatively low bandwidth of 20 MHz. The modified RFX2400 provides a bandwidth of 41 MHz.

The anti-aliasing filter implemented in the RFX2400 is an analog elliptical filter applied to the AD8347 Direct-Conversion Quadrature Demodulator. Two identical filters are implemented: one each for the in-phase and quadrature branches of the demodulator. Below is an extract from the circuit diagram showing the elliptical filter components.



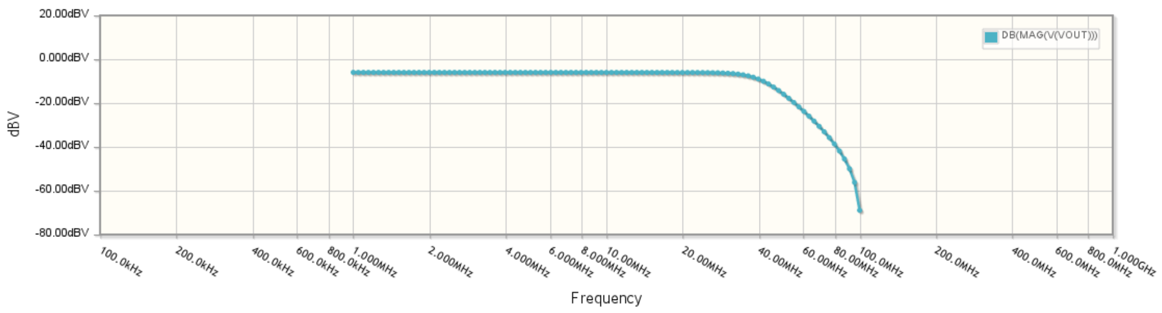
The frequency response of the stock elliptical filter was analyzed and found to match the published bandwidth of 20 MHz. This provided both a check for factory operation and a mechanism to validate the circuit modifications prior to application to the RFX2400 circuit board. The frequency response provided by the Spice analysis of the unmodified RFX2400 is shown below.



The modified RFX2400 replaces the following components:

- C1: from 4.7pF to 2pF
- C2 from 150pF to 68pF
- C3 from 8.2pF to 3.6pF
- C4 from 82pF to 30pF
- L1 from 680nH to 270nH
- L3 from 1200nH to 680nH

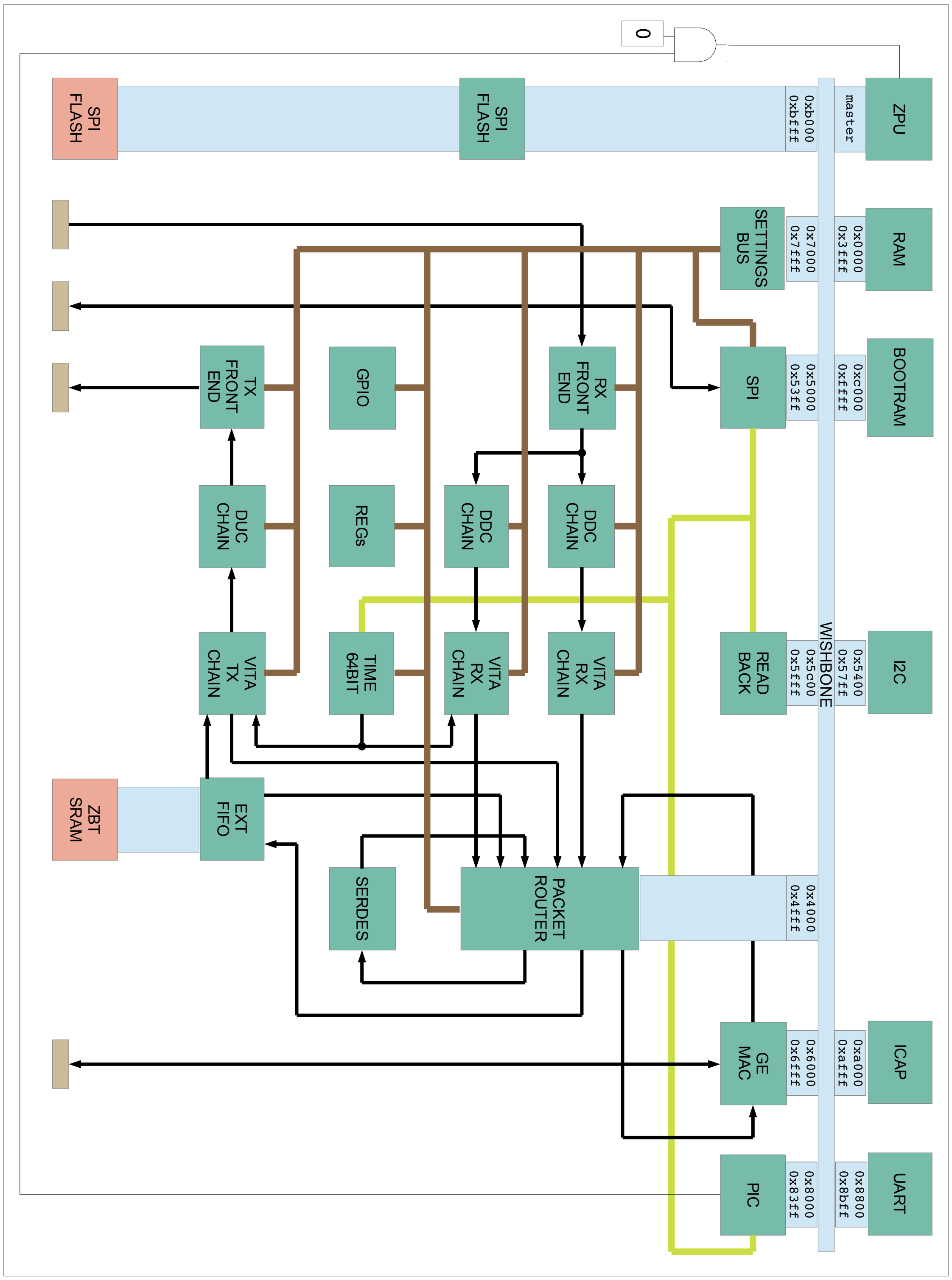
The resulting post-modification filter frequency response is shown below.



Appendix C. USRP Standard Internal Logic

The following figure is a block diagram showing the main functional blocks of the standard USRP N210 internal FPGA logic. This diagram was developed specifically for this report, based on a code inspection of the Verilog sources provided by Ettus Research.

The input/output connections are shown as tan rectangles near the bottom of the diagram. The receive chain is the primary subsystem used in this project, where ADC samples flow into the RX FRONT END block, through the quadrature DDC chains, formatted according to VITA standards, passed to a PACKET ROUTER, then out the GE MAC and finally the gigabit ethernet connector.



ZPU
master
RAM
0x0000 0x3fff
BOOTRAM
0xc000 0xffff
I2C
0x5400 0x57ff
ICAP
0xa000 0xafff
UART
0x8800 0x8bff
WISHBONE
0xb000 0xbfff
SPI FLASH
SETTINGS BUS
0x7000 0x7fff
SPI
0x5000 0x53ff
READ BACK
0x5c00 0x5fff
0x4000 0x4fff
GE MAC
0x6000 0x6fff
PIC
0x8000 0x83ff
0
SPI FLASH
RX FRONT END
DDC CHAIN
DDC CHAIN
VITA RX CHAIN
VITA RX CHAIN
TX FRONT END
DUC CHAIN
VITA TX CHAIN
GPIO
REGs
TIME 64BIT
EXT FIFO
ZBT SRAM
SERDES
PACKET ROUTER