Title
Interference mitigation techniques for ultra-wideband systems

Permalink
https://escholarship.org/uc/item/61p531vt

Author
Jamp, Joe Izu

Publication Date
2007

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA, SAN DIEGO

Interference Mitigation Techniques for Ultra-Wideband Systems

A Dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Electrical and Computer Engineering (Communication Theory and Systems)

by

Joe Izu Jamp

Committee in charge:

Professor Lawrence Larson, Chair
Professor William Griswold
Professor Andrew Kummel
Professor Laurence Milstein
Professor Paul Yu

2007
Copyright
Joe Izu Jamp, 2007
All rights reserved.
This dissertation of Joe Izu Jamp is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2007
To my parents, brother, and grandparents.
# Table of Contents

Signature Page........................................................................................................ iii  
Dedication ................................................................................................................ iv  
Table of Contents ................................................................................................... v  
List of Figures......................................................................................................... vii  
List of Tables ......................................................................................................... xiii  
Acknowledgements ................................................................................................. xiv  
Vita, Publications, and Fields of Study ................................................................. xvi  
Abstract of the Dissertation ................................................................................... xviii  

## 1 Introduction............................................................................................................ 1  
  1.1 Motivation for Spectral Shaping Ultra-Wideband Signals ............................. 1  
  1.2 Wireless Technology Overview ................................................................. 3  
  1.3 Ultra-Wideband Overview ........................................................................... 6  
    1.3.1 Introduction to UWB ............................................................... 6  
    1.3.2 Impulse Radio .......................................................................... 9  
    1.3.3 Direct Sequence Spread Spectrum ............................................ 12  
    1.3.4 Orthogonal Frequency Division Multiplexing .............................. 15  
    1.3.5 Multi-band OFDM .................................................................... 18  
  1.4 Previous Spectral Shaping Techniques ....................................................... 22  
  1.5 Dissertation Organization .......................................................................... 25  

## 2 Look Ahead Block Inversion for Spectral Shaping UWB IR Signals .... 27  
  2.1 Look Ahead Block Inversion ................................................................. 28  
  2.2 Hardware Optimization of LABI ............................................................ 37  
  2.3 LABI for UWB PPM and IR-BPSK ....................................................... 45  
    2.3.1 LABI Time Offset ..................................................................... 45  
    2.3.2 Spectral Shaping with LABI TO ............................................... 47  
    2.3.3 LABI Pulse Inversion ............................................................... 52
List of Figures

Figure 1.1: Data rates and operational range of various wireless systems. .......... 5
Figure 1.2: Frequency usage of various wireless systems. ............................. 6
Figure 1.3: FCC power spectral density emissions limit [6]. ............................ 7
Figure 1.4: Proposed European UWB spectral mask emissions limit [8]............. 8
Figure 1.5: Time domain waveform of a UWB Gaussian monopulse for
\( \tau=1.12 \times 10^{-10} \) ................................................................................................. 10
Figure 1.6: Spectrum of a UWB Gaussian monopulse for \( \tau=1.12 \times 10^{-10} \) .......... 11
Figure 1.7: UWB Impulse Radio Modulation. (a) PPM, (b) BPSK or BPAM... 12
Figure 1.8: UWB DSSS Modulation, where (a) is the original symbol waveform,
and (b) is the spread waveform. The associated spectrums are also shown. 13
Figure 1.9: A decomposed OFDM signal, where a sinc function is centered at
each subcarrier. The grey arrow shows the side-lobes of the sinc from
neighboring subcarriers. Note that the signal from one subcarrier is zero
when the other subcarriers are at their maximum value. ......................... 17
Figure 1.10: The MB-OFDM bands and their grouping [14]. ...................... 18
Figure 1.11: Frequency hopping plan between the bands for Band Group #1... 19
Figure 1.12: Data path for an MB-OFDM system........................................ 19
Figure 1.13: Convolutional Encoder of rate \( R=1/3 \) with generator polynomials of
133_8 (A), 165_8 (B), 171_8 (C). ................................................................................ 20
Figure 1.14: MB-OFDM notch by turning off subcarriers. The notch is located at
around DC (0Hz), and is approximately -8dB in power.......................... 23
Figure 1.15: MB-OFDM notch by turning off subcarriers, zoomed into the notch.
The notch is located at around DC (0Hz), and is approximately -8dB in
power............................................................................................................. 24
Figure 2.1: Binary block code, showing a block of \( N_b \) bits with the flag bit \( F \)
denoting the polarity of the associated block of bits............................ 28
Figure 2.2: LABI block diagram at the transmitter................................. 29
Figure 2.3: Desired shape of the spectrum, and the associated complementary filter.......................................................................................................................... 29

Figure 2.4: Sample Viterbi trellis for a trellis state vector of length two. .......... 31

Figure 2.5: LABI spectrum for various complementary filter block spans. N_b=3.
Longer block spans improve performance..................................................... 32

Figure 2.6: LABI spectrum for various block lengths for N_f=11. Shorter block
lengths improve performance. ...................................................................... 33

Figure 2.7: Asynchronous cross-correlation diagram, showing how the bits in
block i of user l and blocks j and j+1 of user m line up, where the flag bits are
offset by a bits............................................................................................... 36

Figure 2.8: Maximum cross-correlation values, normalized to the Kasami
sequence period, when the period of the sequence fits into a LABI block of
length N_b. The flag bits were chosen to achieve the worst case cross-
correlation. .................................................................................................... 36

Figure 2.9: Monte-Carlo simulation of cross-correlation values for randomly
selected Kasami sequences, normalized to the period of the sequence. The
flag bit insertions cause an increase in the cross-correlation........................ 37

Figure 2.10: Trellis for N_f=4, with the state transition branches showing which
flag bits are used to calculate the RIS (branch metric). '0' and '1' represent
bit values of +1 and -1. ................................................................................. 39

Figure 2.11: The flag used in two RIS calculations. The white blocks show
which computations are the same, since the flag bits are the same. The black
blocks are where the flag bits differ. It can be shown that F'=-F................. 40

Figure 2.12: Complexity cost functions plotted versus block length for N_f=11... 44

Figure 2.13: Complexity cost functions plotted versus filter block span for N_b=4.
....................................................................................................................... 44

Figure 2.14: LABI TO spectrum with a 10MHz wide notch at 3GHz, N_b=2,
N_f=10.......................................................................................................................... 47
Figure 2.15: $|S(\omega_0)_{\text{norm}}|$ as a function of $N$, and compares the analysis (dotted lines) versus the simulation results (solid lines) for various block lengths $N_b$.

Smaller $N_b$ for a given $N$ minimizes $|S(\omega_0)_{\text{norm}}|$ better................................ 51

Figure 2.16: UWB IR-BPSK signal spectrally shaped by LABI PI. $N_b=2, N_f=10$.
....................................................................................................................... 54

Figure 2.17: LABI PI spectrum with a 50MHz wide notch at 3GHz. ............... 56

Figure 2.18: Two ways of cascading LABI PI and LABI TO to allow for spectral shaping of UWB IR signals by changing both the time offsets and pulse polarities........................................................ ......................... 57

Figure 2.19: The notch depth of the four LABI variants as complexity increases (filter block span increases). For TOPI and PITO, the second block in the cascade is fixed at $N_f=6$............................................................ 59

Figure 2.20: The spectrum at the notch for the LABI variants compared with the optimal choice in both pulse polarity and time position......................... 60

Figure 2.21: Increase in notch power vs. rms timing jitter. ......................... 64

Figure 3.1: Block diagram of LABI for OFDM. ............................................. 67

Figure 3.2: Data grouping for LABI for OFDM. The QPSK data bits are denoted as $d(n)$. Subcarriers 3 and 4 are nulled out, subcarriers 1-2 and 5-6 are coded, and subcarriers 0 and 7 are uncoded. The block length $N_b$ consists of multiple OFDM symbols. $N_f=4$. ................................................................. 67

Figure 3.3: Simulated notch power for $N_b=1$ as a function of the number of coded carriers $N_f$. ................................................................. 73

Figure 3.4: LABI coded OFDM spectrum (solid) compared with the uncoded spectrum (dotted). An improvement of 20dB can be seen for $N_b=2$, $N_f=10$, $K=128$. ................................................................. 74

Figure 3.5: Notch power improvements over uncoded notch power for selected block lengths ($N_b$) and coded carriers $N_f$. Two subcarriers are nulled........... 75
Figure 3.6: Effective overhead as a function of the notch depth for the coded and unencoded spectrums. .......................................................................................... 76

Figure 3.7: Notch power as a function of quantization resolution for various block lengths. $N_f=10$. ........................................................................................................ 78

Figure 3.8: The spectrum of quantized LABI coded OFDM at various resolutions. $N_h=4$, $N_f=10$. ................................................................................................................ 78

Figure 4.1: Block diagram of the real-time Spectral Encoded UWB transmitter. (a) The unencoded UWB input signal waveform and spectrum, with a narrowband interferer at 3.6GHz. (b) Spectral Encoded spreading sequence of chip bandwidth 66.67MHz. The chips at the interference frequencies are nulled out. (c) The Spectral Encoded output waveform (time-spread) and spectrum........................................................................................................ 84

Figure 4.2: Block diagram of the real-time Spectral Encoded UWB system. ..... 85

Figure 4.3: Operation of the Real-Time Spectral Encoded Signal, where the signals shown are: (a) the input signal centered about $T_s/2$, (b) the Fourier Transform, (c) after multiplying by the spreading sequence $H_{PN}$, and (d) the transmitted signal $s(t)$........................................................................................................ 89

Figure 4.4: The spillover power in the neighboring chip’s frequencies as a function of $T_s$ caused by the time windowing of the Fourier Transform on a single chip of the spreading sequence. The power decreases as $1/T_s$ due to the contribution from the main-lobe of the $sinc$ function. ............................ 93

Figure 4.5: Notch power as a function of the symbol duration $T_s$. The signal bandwidth is 1GHz, and two of the 15 chips of the spreading sequence are zeroed out to create the notch. ....................................................... 93

Figure 4.6: The spillover power in the adjacent chip as a function of the chip bandwidth for various $T_s$. The arrows point to the optimal bandwidth for each $T_s$. ............................................................................................................ 95

Figure 4.7: Block diagram of the Spectral Encoded UWB Transmitter with the DSP (Pre-calculate waveform block), memory, DAC, and antenna............. 98
Figure 4.8: Notch power versus DAC resolution. Also shown are the lower bounds from the DAC resolution SNR and the boundary for $T_s=35\text{ns}$.

Figure 5.1: Block diagram of the Spectral Encoded Transmitter with the DSP (Pre-calculate waveform block), memory, DAC, and antenna.

Figure 5.2: Circuit diagram of the 5-bit DAC. The three LSBs are binary weighted, and the two MSBs are equally weighted.

Figure 5.3: Glitches due to switching in a DAC. (1) is the LSB switch glitch, and (2) is the MSB switching glitch (which is 4 times the LSB).

Figure 5.4: Test chip block diagram.

Figure 5.5: The block diagram of the 5-bit counter.

Figure 5.6: The logic block diagram for bit-converter. Bits R3 and R4 are converted into a thermometer code.

Figure 5.7: Circuit schematic of an inverter, which can also be used as a buffer.

Figure 5.8: The circuit schematic of a NAND gate. All transistors are minimum sized.

Figure 5.9: The circuit schematic of a NOR gate. All transistors are minimum sized.

Figure 5.10: The circuit schematic of an XNOR gate.

Figure 5.11: Circuit schematic of the D-flip-flop, where D denotes the input and Q is the output.

Figure 5.12: Cadence simulated DAC output signal at a conversion rate of 10GS/s.

Figure 5.13: DAC/Counter microphotograph.

Figure 5.14: Probe testing setup block diagram. The arrows show where the probes are used.

Figure 5.15: Tech bench setup. The input sinusoidal clock generator is off to the right (not shown).

Figure 5.16: Measured DAC output at $f_{CLK}=2\text{GHz}$. 
Figure 5.17: Measured DAC output at $f_{CLK}=5$GHz and $f_{CLK}=10$GHz. 
Figure 5.18: Graphical example of INL and DNL calculation. The dotted line represents the ideal DAC output value at the sample times, and the solid line is the response of the DAC.
Figure 5.19: Dynamic INL and DNL for the DAC operating at $f_{CLK}=5$GHz.
Figure 5.20: Figure of merit (FOM) of the DAC (marked by x) compared with previously reported DACs in this frequency range (marked by circles).
List of Tables

Table 1.1: Data Rates and the corresponding PHY parameters........................... 19
Table 4.1: The memory size and notch power relationship for various Ts and resolutions, assuming the conversion rate is fixed at 10Gs/s.......................... 101
Acknowledgements

I would like express my sincere gratitude to Prof. Larry Larson for his invaluable support and guidance for my research over these past years. His continuous support and suggestions are indispensable in making this research and dissertation possible.

I would also like to thank Prof. Larry Milstein for his support. The occasional short meetings have provided valuable insights into various aspects of digital communications and coding. In addition, I would like to thank Dr. Junxiong Deng for the valuable advice and help in the DAC circuit tapeout.

I would like to thank Prof. Paul Yu, Prof. Andy Kummel, and Prof. Bill Griswold for being on the committee, and for their input in this dissertation.

I would like to acknowledge my colleagues at UCSD for their help and suggestions, especially Dr. John Fairbanks, Dr. Claudio da Silva, Mr. Rahul Kodkani, Mr. Himanshu Khatri, Mr. Yiping Han, Mr. Marcus Pan, and Mr. Mohammad Farazian for their knowledge and help in my research. Lastly, I would like to acknowledge the support of the Center for Wireless Communications at UCSD and its member companies, and the University of California Discovery Grant.

Chapter 2, in part, is a reprint of the material as it appears in “A Coding Technique for the Spectral Shaping of Ultra-Wideband Time Hopping Signals,” IEEE Transactions on Vehicular Technologies, July 2007, and “A Coding

Chapter 3, in part, has material submitted for publication in “A Coding Technique for Spectral Shaping UWB OFDM Signals,” *IEEE Transactions on Wireless Communications.* The authors are Joe Jamp and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.

Chapter 4, in part, is a reprint of the material as it appears in “Hardware Considerations for Spectral Encoded UWB Transmitters,” *2006 IEEE International Conference on Ultra-wideband,* Sept. 2006. The authors are Joe Jamp and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.

Chapter 5, in part, is a reprint of material as it appears in “A 10GS/s 5-bit Ultra-Low Power DAC for Spectral Encoded Ultra-Wideband Transmitters,” *2007 IEEE RFIC Symposium,* June 2007. The authors are Joe Jamp, Junxiong Deng, and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.
Vita, Publications, and Fields of Study

Vita

2001 Bachelor of Science, Electrical Engineering and Computer Science
University of California, Berkeley

2004 Master of Science, Electrical and Computer Engineering
University of California, San Diego

2004 Internship, Hewlett-Packard Laboratories
Palo Alto, California

2002-2007 Research Assistant, Electrical and Computer Engineering
University of California, San Diego

2007 Doctor of Philosophy, Electrical and Computer Engineering,
University of California, San Diego

Publications and Presentations.


ABSTRACT OF THE DISSERTATION

Interference Mitigation Techniques for Ultra-wideband Systems

by

Joe Izu Jamp

Doctor of Philosophy in Electrical and Computer Engineering (Communication Theory and Systems)

University of California, San Diego, 2007

Professor Lawrence Larson, Chair

Due to the large frequency band in which UWB systems may operate, interference mitigation techniques are necessary to reduce the interference caused by UWB systems to narrowband systems operating in the same frequencies. Based on LABI, coding techniques to spectrally shape a UWB signal’s spectrum are presented, which insert in-band notches into UWB IR and OFDM based
systems. They allow for the arbitrary shaping of the UWB spectrum without significant changes to the hardware.

For UWB IR signals, notches can be created by modulating the timing intervals and pulse polarities. By adapting LABI to change one of these aspects of a UWB IR signal, two techniques called LABI TO and LABI PI are presented, and can create notches of -10dB and -18dB respectively. Due to the time hopping nature, the performance of these algorithms may be sensitive to timing jitter, with a rms jitter of 40ps increasing the notch power by 7dB. The effects of timing jitter can be ignored if the rms jitter is below 10ps.

Improvements to these algorithms can be used to reduce the computational complexity of these algorithms without reducing the spectral shaping performance. By examining the calculations of the trellis branch metric of the Viterbi algorithm used in LABI, a Gray Code scheme is developed, and can reduce the complexity by approximately the complementary filter block span. An additional decrease in complexity can be achieved by jointly choosing the pulse polarities and timing intervals. The cascaded techniques LABI PITO and TOPI can lower complexity by implementing less complex Viterbi algorithms without reducing performance.

For UWB OFDM based systems, LABI is again applied to code the data in the vicinity of a notch created by nulling out subcarriers. In-band notches of -28dB can be created, and is an improvement of 20dB over an uncoded spectrum.

Next, two hardware implementations of a Spectral Encoded system are discussed: SAW-based and monolithic. The former may be challenging to
implement due to the high time-bandwidth product required of the SAW filters, while the latter provides an integrated, low-cost solution. A target system is designed using the monolithic solution, and a 5-bit 10GS/s DAC is required. This DAC is implemented using a 0.18µm SiGe BiCMOS process, and consumes 10.2mW with the DNL and INL being within ±0.5LSB and ±1LSB respectively.
1 Introduction

1.1 Motivation for Spectral Shaping Ultra-Wideband Signals

With the proliferation of wireless consumer electronics and services, interference between the wireless systems is an important issue. A typical home may have many wireless devices running at the same time, such as cellular and cordless phones, Bluetooth headsets, and 802.11 wireless local area networks (WLANs). All these devices must not interfere with each other, and effective methods to mitigate interference from one system to another are being researched.

In the United States, the Federal Communication Commission (FCC) manages and regulates the wireless frequency spectrum. Traditionally, the FCC allocates bands of frequencies for various services, including cellular phones, GPS, AM and FM radio, over-the-air television broadcasts [1]. These bands are typically non-overlapping, with each band assigned to a specific service. Some of these bands have been auctioned off, such as the PCS cellular bands, with licenses costing billions of dollars [2].

The FCC has also allocated various frequency bands that are license-exempt. These “unlicensed” bands, such as those around 2.4GHz and 5GHz, can be used to design and operate wireless systems without the need to obtain
expensive licenses, which reduces the cost of development. Many wireless applications have flourished in these unlicensed bands, such as cordless telephones, Bluetooth [3] headsets, 802.11b/g WLANs [4], and WiMAX (Worldwide Interoperability for Microwave Access) [5] all operate in the 2.4GHz frequency band, often operating concurrently and in the vicinity of each other.

Compared to the traditional method of having a single application per frequency band, having multiple applications operating in the same frequency band is a more efficient use of the spectrum. However, having many wireless applications operating concurrently in the same frequency band can lead to interference issues. This is analogous to having a large room full of people talking; it can be difficult to ignore the noise and concentrate on a conversation. To filter out interference from other systems, various techniques must be used in wireless systems.

Techniques for mitigating interference to and from other wireless systems are especially important for Ultra-Wideband (UWB) systems. In 2002, the FCC allocated 7.5GHz of bandwidth for UWB use, from 3.1GHz-10.6GHz [6]. Within this frequency band, there were already many existing applications, such as the 5.8GHz unlicensed bands where 802.11a WLAN and cordless telephones operate, various radars, and WiMAX in the 3.7GHz band. Since UWB systems also operate in these frequencies, they have the potential to cause interference to these systems, and have interference caused by these same systems. Therefore, techniques must be used in UWB systems to mitigate this interference. In
particular, techniques that shape the UWB signal’s power spectrum will be discussed.

1.2 Wireless Technology Overview

Wireless technologies can be broadly categorized into three categories:

1. Wide Area Networks (WAN): Wireless WANs consists of technologies that can operate in ranges of several miles to thousands of kilometers, with typical data rates of a several Mbps, and up to 70 Mbps for WiMAX [5]. To operate at these distances, these systems operate at high transmit power. Examples of these technologies include Satellite TV, GPS, WiMAX, and cellular phone technologies (WCDMA, GSM, EVDO and UMTS).

2. Local Area Networks (LAN): The most commonly deployed Wireless LAN technology is commonly called WiFi, also known as the IEEE standard 802.11a/b/g [4]. WLANs have an operating range of approximately 150 feet (50m), and data rates of 54Mbps for 802.11a and g. Currently, a draft of the 802.11n standard is being finalized, which increases the data rate to 540Mbps. WLANs can consume high amounts of DC power, because of their potential operation over a relatively long distance.
3. Personal Area Networks (PAN): Wireless PAN technologies operate within 30 feet (10m), and include infrared devices (TV remote controls), Bluetooth (handsfree headsets), Zigbee [7] (sensor networks), and UWB (wireless USB). Data rates vary from a few kbps for Zigbee, to 3Mbps for Bluetooth 2.0, and up to 400Mbps for UWB. Due to the short distances in which WPAN systems operate, the power consumption is typically very low, especially relative to WLAN systems.

For some of the wireless technologies described, Figure 1.1 summarizes their data rates and operational ranges. As seen in Figure 1.1, UWB systems operate at high data rates over short distances. Next, the bandwidth used by each of these technologies can be summarized in Figure 1.2, where the frequency usage is shown. Within the large UWB band of frequencies, there are many “narrowband” systems. It is in these overlapping bands where interference can become an issue. If these systems are operating concurrently, they can interfere with each other.
Figure 1.1: Data rates and operational range of various wireless systems.
1.3 Ultra-Wideband Overview

1.3.1 Introduction to UWB

As defined by the FCC [1], a UWB signal is a signal whose fractional bandwidth, defined as the ratio of the signal bandwidth to the center frequency, is greater than 25%, or has an instantaneous bandwidth of greater than 500MHz. As mentioned above, UWB systems are allowed to operate in the band between 3.1GHz and 10.6GHz. The transmitted power is limited by FCC’s Part 15 ruling, which limits the average transmitted power to -41.25dBm/MHz. Figure 1.3
shows the FCC spectral mask limits, which also includes more rigid requirements, such as a -76dBm/MHz limit at 1.5GHz for GPS.

![FCC power spectral density emissions limit](image)

**Figure 1.3: FCC power spectral density emissions limit [6].**

Other countries are developing UWB systems, with spectral masks similar to that of the FCC. For example, the proposed European spectral mask [8] is shown in Figure 1.4, and has a similar shape in the 3.1-10.6GHz range, but has different out of band requirements. While the US spectral mask shown in Figure 1.3 has specific and constant limits for various out-of-band frequencies, such as the GPS band at 1.5GHz, the European mask specifies a decaying limit which can be written as $-51.3 + 87\log(f/3.1)$ for frequencies between 0-3.1GHz, and
\[-51.3 - 87 \log(f/10.6)\] for the frequencies higher than 10.6GHz, where the term \(f\) is in units of GHz.

![Graph](image)

**Figure 1.4:** Proposed European UWB spectral mask emissions limit [8].

Due to different requirements that may arise from operation in different regions of the world, and to limit interference to other systems that may share frequency spectrum, UWB systems that can perform arbitrary spectral shaping are key to global adoption by lowering the cost of the hardware. A single hardware solution with software customization of the spectral shape is less expensive to produce than a custom hardware solution for each region.
While the FCC has defined operational frequencies and power limitations for UWB transmitted signals, they do not specify how the data is transmitted. Next, several modulation techniques for UWB are introduced.

### 1.3.2 Impulse Radio

A UWB impulse radio (IR) signal \[9\] consists of a train of short duration pulses, which can be written as

\[
s_{IR}(t) = \sum_{n} a_n p(t - t_{pn}(n) - t_d d_{n} - nT_f)
\]

(1.1)

where \(a_n\) is the amplitude of the pulses, \(t_{pn}\) is the pseudo-random time hopping sequence used to randomize the time signal, \(t_d\) is the additional time offset specified by \(d_{n}\), and \(T_f\) is the frame rate, such that there will be one pulse every \(T_f\) seconds. In general, \(p(t)\) can be the time domain response of any arbitrary pulse. For this dissertation, we will model the pulse \(p(t)\) as a Gaussian monopulse, whose response is given by

\[
p(t) = \frac{1}{\eta \tau} e^{\left(\frac{t^2}{\tau^2}\right)}
\]

(1.2)

where \(\tau\) is a scaling factor that determines the pulse’s duration, and \(\eta\) is a normalization factor defined as
which integrates over the entire pulse duration $T_p$ and normalizes (1.2) to have unity energy. Figure 1.5 and 6 shows the time domain and frequency spectrum of a Gaussian monopulse for $\tau=1.12 \times 10^{-10}$ seconds. These pulses are used because they provide better multipath and bit error rate (BER) performance amongst the other pulses [10].

\[
\eta = \sqrt{\int_{-T_p/2}^{T_p/2} \frac{t}{\tau} e^{-\left(\frac{t}{\tau}\right)^2} \, dt}
\]

Figure 1.5: Time domain waveform of a UWB Gaussian monopulse for $\tau=1.12 \times 10^{-10}$. 
Data bits can be modulated onto a UWB IR signal either by modulating the time-offsets $d_n$, or the amplitude $a_n$. The former is commonly known as Pulse Position Modulation (PPM) [9] while the latter is known as BPSK or BPAM. Figure 1.7 shows an example how bits “0” and “1” would be transmitted in a PPM (Figure 1.7(a)) or BPSK (Figure 1.7(b)) system.
1.3.3 Direct Sequence Spread Spectrum

Direct sequence spread spectrum (DSSS) [11] is a modulation scheme that is commonly known as Code Division Multiple Access (CDMA). This modulation technique takes a waveform, such as the one in Figure 1.8(a), and multiplies it by a spreading sequence waveform. This spreading sequence is a set of +1 and -1 values, and the corresponding waveform consists of “chips,” which are rectangular waveforms with a +1 or -1 amplitude with a duration of \( t_{\text{chip}} \). An example of a spreading sequence waveform is shown in Figure 1.8(b), where the spreading sequence consists of seven “chips” valued at [+1 -1 -1 +1 -1 +1 -1]. The resulting spectrum is “spread” in the frequency domain, but contains the same energy as the unspread waveform. The resulting bandwidth is roughly...
inversely proportional to $t_{\text{chip}}$. Since the minimum UWB bandwidth is 500MHz, the maximum chip duration would be approximately 2ns.

![Figure 1.8: UWB DSSS Modulation, where (a) is the original symbol waveform, and (b) is the spread waveform. The associated spectrums are also shown.](image)

The spreading sequences used are a special type of sequences called maximal length sequences, or $m$-sequences [12]. Their autocorrelation function, given by

$$\phi(m) = \frac{1}{N_{\text{chips}}} \sum_{n=0}^{N_{\text{chips}}-1} t_{pn}(n)t_{pn}^*(n+m),$$

(1.4)
can take on the following values

$$\phi(m) = \begin{cases} 
1 & m = 0 \\
-1/N_{\text{chips}} & 1 \leq m \leq N_{\text{chips}} - 1
\end{cases}$$

(1.5)
where \( m \) is the offset from zero, and \( N_{\text{chips}} \) is the length of the sequence. When \( m=0 \), the value is the highest, enabling detection when a symbol starts, since for any other offset, the value is small compared to the non-offset value. Also, the cross-correlation properties of these codes are important, since each user maybe assigned a different code. Hence, the cross-correlations should also be as low as possible and certain codes such as Gold [12] and Kasami [12] codes have low cross-correlation properties.

A BPSK modulated DSSS signal can be written as

\[
s_{\text{DSSS}}(t) = \sum_{n} d(n) PN(t - nT)
\]

where \( d(n) \) is the data, \( T \) is the symbol duration, and \( PN(t) \) is the spreading sequence waveform which can be written as

\[
PN(t) = \sum_{k=0}^{N_{\text{chips}}-1} p_{\text{r,a}}(t - kT_{\text{chip}} - T_{\text{chip}} / 2)
\]

where \( N_{\text{chips}} \) is the number of chips in the PN sequence, and \( T_{\text{chip}} \) is the duration of a chip. \( p_{\text{r,a}}(t) \) is the rectangle function, and it and its Fourier transform \( P_{\text{r,a}}(\omega) \) can be written as

\[
p_{\text{r,a}}(t) = \begin{cases} 1 & -a/2 \leq t \leq a/2 \smallskip \rule{0pt}{3em} 0 & \text{otherwise} \end{cases}
\]
\[ P_{r,d}(\omega) = \int_{-\infty}^{\infty} p_{r,d}(t) e^{j\omega t} dt \]
\[ = \int_{-\infty}^{\infty} e^{-j\omega t} dt \]
\[ = \frac{e^{-j\omega t} - e^{j\omega t}}{-j\omega} \].
\[ = \sin(a\omega) \over a\omega \]  \hspace{1cm} (1.9)

(1.9) is also commonly known as the \textit{sinc} function. Thus, the Fourier transform of (1.6) can be written as

\[ S_{DSSS}(\omega) = \sum_{n} d(n) \sum_{k} P_{r,\text{chip}}/2(\omega) \]
\[ = \sum_{n} \sum_{k} d(n) \frac{\sin(\omega T_{\text{chip}}/2)}{\omega T_{\text{chip}}/2} e^{-j\omega(kT_{\text{chip}} + T_{\text{chip}}/2)}. \] \hspace{1cm} (1.10)

### 1.3.4 Orthogonal Frequency Division Multiplexing

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation technique used in 802.11a/g Wireless LAN and WiMAX systems. The advantages of OFDM are its efficient use of bandwidth, and its robustness to multipath. An OFDM signal is composed of many sinusoidal carriers, and its baseband representation for a single OFDM symbol can be written as

\[ x_{\text{OFDM}}(t) = \sum_{k=0}^{K-1} X(k)e^{j2\pi nk/K} \] \hspace{1cm} (1.11)

where \( X(k) \) is the data modulated onto subcarrier \( k \), \( e^{j2\pi nk/K} \) is the baseband representation of a sinusoidal wave with frequency \( 2\pi k/K \), and is valid for
$0 \leq t \leq T_s$ where $T_s$ is the symbol duration. Note that $1/T_s$ specifies the subcarrier spacing. To obtain the spectrum of $x_{OFDM}(t)$, it can be seen that (1.11) is time limited to a duration of $T_s$. Thus, one can rewrite (1.11) as

$$x_{OFDM}(t) = p_{r,T_s}(t - T_s / 2) \sum_{k=0}^{K-1} X(k) e^{j2\pi k/ K}$$  \hspace{1cm} (1.12)

Therefore, the Fourier transform of $x_{OFDM}(t)$ can be written as

$$X_{OFDM}(\omega) = P_{r,T_s}(\omega)e^{-j\omega T_s / 2} * \mathcal{F} \left\{ \sum_{k=0}^{K-1} X(k) e^{j2\pi k/ K} \right\}$$  \hspace{1cm} (1.13)

where * denotes the convolution operation and $\mathcal{F}$ denotes the Fourier transform operation. The Fourier transform of $e^{j2\pi k/ K}$ is $\delta(\omega - 2\pi k / K)$, where $\delta(x)$ is the Dirac Delta function defined as

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (1.14)

Therefore

$$\mathcal{F} \left\{ \sum_{k=0}^{K-1} X(k) e^{j2\pi k/ K} \right\} = \sum_{k=0}^{K-1} X(k) \delta(\omega - 2\pi k / K)$$  \hspace{1cm} (1.15)

and the (1.13) can be written as

$$X_{OFDM}(\omega) = \sum_{k=0}^{K-1} X(k) P_{r,T_s}(\omega - 2\pi k / K)e^{-j(\omega - 2\pi k / K)T_s / 2}$$  \hspace{1cm} (1.16)
Equation (1.16) describes the OFDM symbol spectrum, and is composed of the sum of $\text{sinc}$ functions, each centered at a subcarrier’s frequency. This can be seen in Figure 1.9, which shows the decomposition of the OFDM symbol into the $\text{sinc}$-like contribution from each of the subcarriers. It can be seen that the peak of each of the $\text{sinc}$ functions is located at a subcarrier frequency, and at that frequency, the magnitude of the $\text{sinc}$ functions from all the other subcarriers are zero. This aspect describes the orthogonality in an OFDM system. However, the frequencies between the subcarriers are not zero due to the $\text{sinc}$ roll off.

![Figure 1.9: A decomposed OFDM signal, where a $\text{sinc}$ function is centered at each subcarrier. The grey arrow shows the side-lobes of the $\text{sinc}$ from neighboring subcarriers. Note that the signal from one subcarrier is zero when the other subcarriers are at their maximum value.](image-url)
1.3.5 Multi-band OFDM

The UWB version of OFDM is known as Multi-band OFDM (MB-OFDM) [13][14], and the full specification can be found in document ECMA-368 [14], where there are 128 subcarriers spanning a total of 528MHz of bandwidth per symbol. Only 100 of the 128 subcarriers are used to transmit data, with the rest being allocated for other purposes, such as synchronization. To take advantage of the large UWB spectrum, the 3.1-10.6GHz frequency band is divided into 528MHz bands, as shown in Figure 1.10. Frequency hopping is employed between the bands, and a different band is used every 312.5ns, shown in Figure 1.11 for Band Group #1. Also, the frequency hopping scheme can be changed depending on the operational region, which helps with interference issues.

![Figure 1.10: The MB-OFDM bands and their grouping [14].](image-url)
MB-OFDM supports many data rates, and is listed in Table 1.1. The path the data takes from bits to OFDM symbol is shown in Figure 1.12, and each block will be described briefly.

**Table 1.1: Data Rates and the corresponding PHY parameters.**

<table>
<thead>
<tr>
<th>Data Rate (Mbps)</th>
<th>Coded Bits / OFDM Symbol ($N_{CBPS}$)</th>
<th>Tone Interleaver Block Size ($N_{Tim}$)</th>
<th>Time Spreading Factor ($N_{TDS}$)</th>
<th>Cyclic Interleaver Shift ($N_{cyc}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>53.3</td>
<td>100</td>
<td>10</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>80</td>
<td>100</td>
<td>10</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>106.7</td>
<td>200</td>
<td>20</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>160</td>
<td>200</td>
<td>20</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>20</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>320</td>
<td>200</td>
<td>20</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>400</td>
<td>200</td>
<td>20</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>480</td>
<td>200</td>
<td>20</td>
<td>1</td>
<td>33</td>
</tr>
</tbody>
</table>

**Figure 1.12: Data path for an MB-OFDM system.**
The first block is the convolutional encoder. A convolutional code is used to increase the data’s resilience to error due to noise and other interference. The encoder used in MB-OFDM uses a rate $R=1/3$ code with generator polynomials $g_0 = 133_8$, $g_1 = 165_8$, and $g_2 = 171_8$, as shown in Figure 1.13. To achieve a certain data rate, a different coding rate may be used, such as $R = 1/3, 1/2, 5/8, 3/4$. These coding rates are derived from the original rate of $R=1/3$ by the use of “puncturing.” Puncturing removes some of the coded bits, which reduces the number of transmitted bits and increases the coding rate.

![Figure 1.13: Convolutional Encoder of rate $R=1/3$ with generator polynomials of $133_8$ (A), $165_8$ (B), $171_8$ (C).](image)

The next block is the bit interleaver. Bit interleaving is used to provide robustness against burst errors, and Table 1.1 lists some of these parameters. Bit interleaving is done in three stages:

1. Symbol Interleaving: The data bits are permuted across six OFDM symbols to exploit frequency diversity from the frequency hopping.

Let $d_{\text{orig}}[n]$ and $d_{\text{si}}[n]$ be the original and after symbol interleaving data sequences, where $n = 0, 1, \ldots, N_{\text{CBPS}} - 1$. An equation for this operation can be written as
\[ d_{SI}[n] = d_{\text{orig}} \left[ \frac{j}{N_{\text{CBPS}}} + \frac{6}{N_{\text{TDS}}} \mod(j, N_{\text{CBPS}}) \right] \] (1.17)

2. Intra-symbol tone interleaving: The data bits within each symbol are permuted between the subcarriers to guard against narrowband interferers.

\[ d_{ISTI}[n] = d_{SI} \left[ \frac{j}{N_{\text{Tot}}} + 10 \mod(j, N_{\text{Tot}}) \right] \] (1.18)

3. Intra-symbol cyclic shifts: The data bits within a symbol are cyclically shifted \( N_{\text{cyc}} \) bits listed in Table 1.1.

\[ d_{BI}[n] = d_{ISTI} \left[ \frac{n}{N_{\text{CBPS}}} N_{\text{CBPS}} + \mod \left( i + \frac{n}{N_{\text{CBPS}}} \right) N_{\text{cyc}}, N_{\text{CBPS}} \right] \] (1.19)

Here, \( d_{BI}[n] \) is the final bit interleaved sequence.

The next block is the QPSK modulator, which takes pairs of bits and converts them into QPSK symbols. The QPSK symbols are then passed to the OFDM modulator, which takes every group of 100 QPSK symbols and modulates them on the 100 data subcarriers as in (1.12), where \( K=100 \).
1.4 Previous Spectral Shaping Techniques

Due to the potential of UWB systems to interfere with other systems, many spectral shaping techniques have been researched. In this section, a few of these techniques will be described.

A commonly used method to perform spectral shaping is to pass the signal through a filter, such as a Surface Acoustic Wave (SAW) filter or a digital finite impulse response (FIR) filter. These are commonly used to attenuate the power in the frequencies outside of the operational band. However, for in-band spectral shaping, the filter requirements may be impractical and potentially expensive, as a high order filter or a complex FIR filter would need to be designed. Furthermore, for analog and SAW filters, once designed and implemented, their frequency characteristics are fixed and cannot be tuned to filter out different frequencies.

Pulse shapers designed to shape a UWB IR signal have been published. A Duo-binary pulse [15] and Manchester monopulse [10] have been described, but their spectral shaping ability is sensitive to variations in the pulse shape. Also, UWB pulse shapers have been described in [16] and [17] that shape the pulse specifically for the FCC spectral mask, such that the signal does not emit power in out-of-band frequencies above the spectral mask limits. However, they do not provide spectral shaping for in-band notching.
To spectrally shape OFDM based systems, a simple method is to turn off specific subcarriers at desired notch frequencies. However, as mentioned in Section 1.3.4, the sinc contribution from neighboring subcarriers will cause the power in the notch to be non-zero, as shown in Figure 1.14 where a notch around DC (0 Hz) has approximately -8dB in power. Improvements to notch power can be obtained by using other methods, such as by cancellation tones described in [18] and [19] where nearby subcarriers are used to reduce the residual power in a notch. However, both these techniques reserve subcarriers that cannot be used to transmit data.

Figure 1.14: MB-OFDM notch by turning off subcarriers. The notch is located at around DC (0Hz), and is approximately -8dB in power.
Another method to spectrally shape UWB signals is Spectral Encoding [20]. Spectral Encoding can be seen as a frequency domain counterpart to DSSS, where the signal’s spectrum is multiplied by a spreading sequence, which results in a spreading of the time domain waveform. In the frequencies where a notch is desired, the corresponding chips are nulled out. This technique will be described in more detail in Chapter 4.

Lastly, coding techniques can be used to perform spectral shaping. Specifically, line codes or block codes can be used on the data to reduce the signal power in a desired notch. As an example, Look Ahead Block Inversion (LABI) [21] is binary block code that is used to shape a digital signal’s spectrum by inverting selective blocks of bits. We will extend LABI and use it to perform
spectral shaping on UWB signals, specifically for IR and OFDM modulation techniques.

1.5 Dissertation Organization

This dissertation will be organized as follows:

In Chapter 2, LABI, the binary block code used to arbitrarily shape a digital data’s spectrum, will be modified and applied to UWB IR signals to spectrally shape them. A low complexity implementation is described, and results and improvements to the techniques will be discussed. Lastly, this chapter will present the spectral shaping performance in the presence of timing jitter.

In Chapter 3, a coding technique based on LABI is applied to UWB MB-OFDM signals. The setup of the algorithm and results on in-band spectral shaping will be presented.

Chapter 4 will describe hardware implementation tradeoffs of a Spectral Encoded Transmitter. First, the Spectral Encoding technique will be briefly reviewed. Then, the real-time Spectral Encoded UWB transmitter will be described, along with two possible implementations: SAW based, and an integrated solution. System parameters choices and tradeoffs will be discussed for both implementations.
Finally, using the system parameters obtained in Chapter 4, Chapter 5 will describe the design choices, implementation, and experimental results of a 10GS/s 5-bit digital to analog converter (DAC).
2 Look Ahead Block Inversion for Spectral Shaping UWB IR Signals

Various interference mitigation techniques to spectrally shape a UWB IR signal’s spectrum away from an in-band narrowband system’s frequencies have been reported. These techniques generally involve some type of pulse shaping to meet spectral specifications, such as the FCC spectral mask [16][17], or generating a pulse with specific spectral characteristics [15][10]. These techniques do not depend on the transmitted data, but may require specialized hardware to generate the specific pulses.

In this chapter, a spectral shaping technique based off of a binary block code called Look Ahead Block Inversion (LABI) [21] is described. This technique codes the transmitted data such that the signal’s spectrum is shaped away from the desired frequencies by changing either the timing interval between pulses and/or the polarity of the pulses.

This chapter will be organized as follows. First, Section 2.1 will present a review of LABI, followed by a low complexity optimization for LABI for simpler hardware implementation in Section 2.2. Then, in Section 2.3, LABI will be applied to UWB IR signals, specifically the PPM and IR-BPSK modulations.
Simulation results will be presented, and improvements to the techniques will be discussed. Lastly, the effects of timing jitter on the algorithms will be discussed.

### 2.1 Look Ahead Block Inversion

LABI is a binary code where blocks of data bits are selectively inverted in order to shape the resulting frequency spectrum, as shown in Figure 2.1. A trailing flag bit designates whether or not the block is inverted [21], and a block diagram of the algorithm is shown in Figure 2.2. Given a desired shape of the spectrum $S_{\text{desired}}(\omega)$, the complementary filter [10] is defined as

$$H_c(\omega) = \sqrt{S_m - S_{\text{desired}}(\omega)} \quad (2.1)$$

where $S_m$ is the maximum value of $S_{\text{desired}}(\omega)$. We define $h_c(n)$ as the inverse Fourier Transform of $H_c(\omega)$, which spans $N_f = 2L$ blocks of $N_b$ bits per block for a total $N_c = N_f N_b$ bits. In general, $h_c(n)$ is non-causal and centered around $n = 0$.

---

**Figure 2.1:** Binary block code, showing a block of $N_b$ bits with the flag bit $F$ denoting the polarity of the associated block of bits.
Next we define the flag padded sequence as \( d_b(n) \), given by

\[
d_b(iN_b + n) = \begin{cases} 
F_id(iN_b + n) & n = 0, 1, ..., N_b - 2 \\
F_i & n = N_b - 1
\end{cases}
\]  

(2.2)
where \( F_i \) denotes the polarity of the flag bit at block \( i \) and \( d(n) \) is the original data sequence. Next, the data is passed through the complementary filter, and the filter output is

\[
e(iN_b + j) = \sum_{k=-L}^{L-1} \sum_{m=0}^{N_b-1} d_b ((i-k)N_b + m) h_i (kN_b + j - m)
\]

(2.3)

where \( i \) and \( k \) are block indices, and \( i \) and \( m \) index within a block. Next, we define a measure of the output power of the complementary filter as the Running Interference Sum (RIS) at block \( i \) as

\[
RIS(i) = \sum_{j=0}^{N_b-1} e^2 (iN_b + j)
\]

(2.4)

The total RIS up to block \( r \) is given by

\[
J(r) = \sum_{i=-\infty}^{\infty} RIS(i)
\]

(2.5)

Since the RIS is a measure of the power of the complementary filter output, by minimizing the total RIS up to the current block, the output power of the filter will be minimized. Thus, the shape of the digital data’s spectrum will resemble the desired shape \( S_{\text{desired}}(\omega) \) if the flag bits are chosen to minimize the RIS.

Since each flag bit will affect past and future values of the RIS, we want to minimize \( J(\infty) \) by choosing the appropriate flag bit polarities. This involves a comprehensive search, and we use the Viterbi algorithm [24] to determine the flag bit polarities. The state vector of the Viterbi trellis is defined as
with the state transition specifying the final flag bit. The RIS is used as the branch metric. As an example, Figure 2.4 shows the Viterbi trellis with a trellis state vector of length two. The values shown in states A-D are the flag bit values, where a “0” denotes -1 and a “1” denotes +1. Each state transition (denoted by an arrow) has a RIS branch metric where the flag bits used in calculating the RIS is shown on the arrow.

Due to the complexity of the algorithm, closed form analytical results for the resulting spectrum are difficult to develop. Instead, simulation results are presented. Figure 2.5 and Figure 2.6 show examples of the spectral shaping that can be performed with LABI, and it can be seen that smaller block lengths as well as longer filter block spans have better spectral shaping performance. For a given complementary filter length $N_c$, choosing smaller $N_b$ will increase $N_f$ which leads to better performance. Thus, LABI can be tuned by choosing an appropriate complementary filter with the desired spectral specifications, then choosing the
largest value of $N_b$ such that the desired spectral shape is achieved with minimal overhead.

![LABI spectrum for various complementary filter block spans. $N_b=3$. Longer block spans improve performance.](image)

Figure 2.5: LABI spectrum for various complementary filter block spans. $N_b=3$. Longer block spans improve performance.
Figure 2.6: LABI spectrum for various block lengths for $N_f=11$. Shorter block lengths improve performance.

From a practical standpoint, there appears to be an infinite delay through LABI, since minimizing $J(\infty)$ implies the use of LABI across an infinite sequence. With the Viterbi Algorithm, it has been found that the surviving sequences past approximately five times the constraint length all contain identical bits [12]. Therefore, the delay through LABI can be as little as five times the size of the state vector defined in (2.6), or $5N_f$ blocks of bits for a total of $5N_fN_b$ bits, without degrading the performance of the algorithm.

The receiver needs to detect another bit before choosing whether or not to invert the block. However, an error in the flag bit detection would cause all bits in the block to be decoded incorrectly, leading to a higher bit error rate (BER). However, assuming the channel BER is relatively low, such as $10^{-3}$ or $10^{-4}$, we
can assume that if there is an error, there is only one bit error per block, which is either a data bit or a flag bit. If the error is a data bit, after inverting and removing the flag bit from the data sequence, one can rely on error correcting codes [12] to detect and correct the error. If, however, the flag bit is incorrect, then an entire block will be decoded incorrectly. However, LABI does not provide error correction, and thus would need to depend on error correcting codes to correct \( N_b - 1 \) bits, such as convolutional codes with a Hamming distance greater than \( 2(N_b - 1) \) bits.

For multiple user environments, we examine the effects of LABI on the orthogonality of maximal length orthogonal codes (\( m \)-sequences) such as Kasami or Gold Sequences [12]. Specifically, we will examine the effect on the cross-correlation, since \( m \)-sequences are designed to have low cross-correlation properties. If we view the \( m \)-sequence as the data bits to be spectrally shaped by LABI, LABI will insert a periodic flag bit into the data bit sequence to create a sequence of a block of bits. Then, the algorithm will invert selected blocks to achieve the desired spectral shape. In this case, the asynchronous cross-correlation between a desired user \( l \) and another user \( m \), where both users’ data bits have been shaped by LABI and both users are using different Kasami sequences of the same period.

There are two cases that need to be examined. The first case is when the entire \( m \)-sequence period fits entirely within a LABI block of length \( N_b \). Since
we are examining an asynchronous cross-correlation, the flag bits may not line up,
and we will assume that they are offset in time by a bits. From Figure 2.7, which
shows how the bits line up for a cross-correlation between block \( i \) of user \( l \) and
blocks \( j \) and \( j+1 \) of user \( m \), we can write the cross correlation as

\[
R_{l,m}(i, j, a) = \sum_{n=0}^{N_b-1-a} d_i^{(l)}(n)d_j^{(m)}(n + a) + \sum_{n=N_b-a}^{N_b-1} d_i^{(l)}(n)d_{j+1}^{(m)}(n - N_b + a) \quad (2.7)
\]

where \( d_i^{(l)}(n) \) is the data bit at block \( i \) for user \( l \), and is defined from (2.2) as

\[
d^{(l)}(iN_b + n) = \begin{cases} 
F_i^{(l)}d^{(l)}(iN_b + n) & n = 0, 1, ..., N_b - 2 \\
F_i^{(l)} & n = N_b - 1
\end{cases} \quad (2.8)
\]

We expand the first term in (2.7), and obtain

\[
F_j^{(m)}d_i^{(l)}(N_b - 1 - a) + \sum_{n=0}^{N_b-2-a} d_i^{(l)}(n)d_j^{(m)}(n + a) \quad (2.9)
\]

The first term in (2.9) is the direct contribution from the flag bit \( F_j^{(m)} \), and the
second term is the partial cross correlation. The second term in (2.7) can be
similarly expanded, and we can obtain the contribution to the cross-correlation
from \( F_i^{(l)} \). Both of these terms can add to the cross-correlation. As shown in
Figure 2.8, where the flag bits were chosen to achieve the worst case cross-
correlation, the impact on the cross-correlation is greater when the period of the
\( m \)-sequence is shorter.
Figure 2.7: Asynchronous cross-correlation diagram, showing how the bits in block $i$ of user $l$ and blocks $j$ and $j+1$ of user $m$ line up, where the flag bits are offset by $a$ bits.

Figure 2.8: Maximum cross-correlation values, normalized to the Kasami sequence period, when the period of the sequence fits into a LABI block of length $N_b$. The flag bits were chosen to achieve the worst case cross-correlation.

The second case is when the period of the $m$-sequence is larger than the block length. In this case, by applying LABI to the $m$-sequence, it is divided into blocks, with a flag bit inserted for each block. Figure 2.9 shows a Monte Carlo simulation for randomly chosen Kasami sequences of period 1023. The cross-correlation can increase to 16% of the period for low values of $N_b$, which is much higher than the Welch bound of 3% for Kasami sequences [12]. Note that
as the block length increases, the cross-correlation generally decreases. As a result, to keep the cross-correlation low, longer block lengths up to the period length of the $m$-sequence are recommended. However, with longer block lengths, as will be discussed later, LABI may not be able to achieve the desired spectral shaping.

![Figure 2.9: Monte-Carlo simulation of cross-correlation values for randomly selected Kasami sequences, normalized to the period of the sequence. The flag bit insertions cause an increase in the cross-correlation.](image)

### 2.2 Hardware Optimization of LABI

The performance of LABI is limited by the RIS calculations [21]. Due to its use as the trellis branch metric, it is important to minimize its computational complexity. In this section, we will reduce the computational complexity of the
RIS calculation by reducing the redundant calculations. First, we examine how the RIS is calculated. Since the RIS from (2.4) is computed by squaring and summing the filter output, by reducing the computations required for the filter output, we reduce the complexity of the RIS calculations. We rewrite the filter output (2.3) as

\[
e(iN_b + j) = \sum_{k=-L}^{L} e_b(i-k, j)
\]  

(2.10)

where \( e(i-k, j) \) is the contribution from block \( i-k \) and its corresponding flag bit \( F_{i-k} \), and is defined as

\[
e_b(i-k, j) = \sum_{m=0}^{N_p-1} d_b((i-k)N_b + m)h_c(kN_b + j - m)
\]  

(2.11)

From (2.10) and (2.11), each filter output calculation requires \( N_f \) flag bits, \( N_f - 1 \) of which are specified by the Viterbi trellis state vector (2.6), with the final flag bit specified by the state transition. Hence, there are \( 2^{N_f} \) trellis branches for each stage of the trellis, each requiring a RIS calculation. This also means all possible combinations of the flag bits are represented in the branch metric calculations. Figure 2.10 shows a trellis for \( N_f = 4 \).
Next, we note that there are pairs of branches where the set of flag bits used in calculating the filter output in each branch only differ by one flag bit, such as with the two branches coming out of state A in Figure 2.10. Except for the contribution by the block with the differing flag bit, the contributions from the other blocks are the same since the flag bits for those blocks are all the same (Figure 2.11). For the block with the differing flag bit, it can be seen from (2.10) that they have the same magnitude but opposite polarity. Then, the difference in the filter output between the two branches is just twice the magnitude of the contribution from this block. Therefore knowledge of the filter output for one of
the trellis branches makes it quite simple to calculate the filter output for another branch, when it only has one differing flag bit.

![Diagram](attachment:trellis_diagram.png)

**Figure 2.11: The flag used in two RIS calculations.** The white blocks show which computations are the same, since the flag bits are the same. The black blocks are where the flag bits differ. It can be shown that $F'=-F$.

Since at each trellis stage, all possible flag bit combinations are used to calculate the RIS for all the branches, one can order these sets of flag bits in a Gray Code manner. Gray Coding guarantees that between each successive set, there is only one flag bit that is different. Therefore, after calculating the RIS for the first branch, it is simple to calculate the RIS for the second branch, then the third, and so on until all the branch metrics have been computed. This reduces the computational complexity significantly.

Compared to performing an explicit convolution for each RIS calculation, this method requires fewer computations since an explicit convolution is not performed for each RIS; only one block’s contribution needs to be calculated. To assess the improvement achieved by using this Gray Code optimization, the “computational load per data bit,” originally defined in [21], will be used. This metric represents the computational complexity to code each data bit, and is the total number of multiply-accumulate operations required to calculate the RIS for
all the branches at each trellis stage, normalized by the number of data bits per block. It is defined as

\[ C = \frac{C_{\text{filterout}} C_{\text{RIS}}}{\text{data bits per block}} \frac{\# \text{trellis state transitions per stage}}{2} \]  

(2.12)

where \( C_{\text{filterout}} \) is the number of multiply-accumulate operations performed to calculate the filter output, \( C_{\text{RIS}} \) is the number of multiply-accumulate operations to evaluate (2.4). The factor of one-half comes from the quadratic nature of the RIS in (2.4), since the RIS value is the same even if all the flag bits are inverted.

From [21], the most computationally complex method is the explicit convolution, and its computational load per data bit is given by

\[ C = \frac{N_f N_b^2}{N_b - 1} 2^{N_f - 1} \]  

(2.13)

For the Gray code optimized method, since given that one of the states has been calculated, each RIS calculation essentially involves calculating (2.11) for the differing block (\( N_b \) multiply-accumulate operations), doubling and subtracting from the known filter output (one multiply-accumulate operation). Therefore, the computational load per data bit is

\[ C = \frac{(N_b + 1)N_b}{N_b - 1} (2^{N_f - 1} - 1) + \frac{N_f N_b^2}{N_b - 1} \]  

(2.14)
The second term in (2.14) is the computational load required to calculate the “initial” state, and the first term is the computational load to compute the rest. Compared to (2.13), (2.14) is approximately lower by a factor of $N_f$, a significant improvement.

The authors of [21] also describe a table lookup method which stores the magnitude of $e_b(i-k, j)$ for each possible block $i-k$ into a table. With this method, to calculate the filter output in (2.10), we simply perform a table lookup for block $i-k$, then add or subtract the result depending on the flag bit $F_{i-k}$. Assuming a table lookup and an addition/subtraction operation is equivalent to a multiply-accumulate operation, (2.10) requires $C_{\text{filterout}} = N_f$ table lookups, and the resulting computational load per data bit is

$$C = \frac{N_fN_b}{N_b-1}2^{N_f-1}$$  \hspace{1cm} (2.15)

If a similar table lookup scheme is optimized using the Gray Coded optimization, as opposed to calculating (2.11) explicitly for each RIS calculation, a table lookup operation is performed to obtain the block’s contribution. This involves one table lookup and one multiply-accumulate operation. The resulting load per data bit is

$$C = \frac{(N_b+1)N_b}{N_b-1}(2^{N_f-1} - 1) + \frac{N_fN_b^2}{N_b-1}$$ \hspace{1cm} (2.16)
A plot of these four “cost functions” are shown in Figure 2.12 and Figure 2.13, and they show that the optimized versions of LABI has roughly $N_f$ lower computational complexity than the original LABI algorithm, except when $N_f = 2$ where the computational complexities are roughly the same as the non-optimized versions. When $N_b < N_f$, both optimized methods have lower complexities than even the non-optimized table lookup method, though the lowest complexity is when both table lookup and the Gray code optimization are used. Although the optimized versions reduce the number of computations required, it increases the latency by roughly the same factor, since each RIS value depends on knowing a previous one. Assuming all computations can be completed at a given bit rate, the Gray Code optimized versions can lead to simpler hardware design due to the reduction in the number of multiply-accumulate operational blocks.
Figure 2.12: Complexity cost functions plotted versus block length for $N_f=11$.

Figure 2.13: Complexity cost functions plotted versus filter block span for $N_b=4$. 

2.3 LABI for UWB PPM and IR-BPSK

2.3.1 LABI Time Offset

The spectral shaping achieved by LABI depends on the pulses having positive and negative amplitudes as determined by the data bits. By contrast, a PPM signal’s pulse polarities are the same regardless of bit value; the data bits modulate the time position of the pulses. As a result, we must modify LABI to account for the time position modulation. This is done by changing the calculation of the RIS metric, with the rest of the algorithm being left unchanged.

First, we rewrite the UWB IR Equation (1.1) for PPM as

\[ s_{\text{PPM}}(t) = \sum_{i} \sum_{n=0}^{N_b-1} p(t - \tau_i(i, n)) \]  

(2.17)

where \( \tau_i(i, n) = t_{\text{on}}(iN_b + n) - t_{\text{off}}d_b(iN_b + n) - (iN_b + n)T_f \) and \( d_b(\bullet) \) is described in (2.2). Next, we define \( h_c(t) \) as the inverse Fourier Transform of \( H_c(\omega) \), and it spans \( N_f = 2L \) blocks of a total of \( N_c = N_fN_b \). Hence, the filter has an impulse response duration of \( N_cT_f \), where \( T_f \) is the bit rate. In general, \( h_c(t) \) is non-causal and we center it around \( t = 0 \). Then, we filter the signal (2.17) through \( h_c(t) \), and we redefine the RIS for block \( i \) as

\[ \text{RIS}(i) = \int_{-\infty}^{\infty} \left| s_{\text{PPM}}(t) * h_c(t) \right|^2 \, dt \]

= \int_{-\infty}^{\infty} \left[ \sum_{k=-L}^{L} \sum_{n=0}^{N_b-1} p(t - \tau_i(i-k, n)) \right] * h_c(t) \, dt \]  

(2.18)
where * denotes the convolution operation. Using Parseval’s Theorem, (2.18) can be rewritten as

$$RIS(i) = \int \left| P(\omega) \right|^2 \left| H_c(\omega) \right|^2 \left| \sum_{k=L}^{L-1} \sum_{n=0}^{N_k-1} e^{-j\omega \tau_i(i-k,n)} \right|^2 d\omega$$

(2.19)

where $P(\omega)$ is the Fourier Transform of $p(t)$. A simpler expression for (2.19) can be obtained if the assumption is made that within a band of interest, $P(\omega)$ and $H_c(\omega)$ are relatively constant. Then for a single notch, (2.19) can be simplified as

$$RIS(i) = \sum_{k=-L}^{L-1} \sum_{n=0}^{N_k-1} \sum_{p=0}^{N_k-1} \frac{\sin \omega_U \left( \tau_i(i-k,n) - \tau_i(i-q,p) \right)}{\tau_i(i-k,n) - \tau_i(i-q,p)}$$

(2.20)

$$- \frac{\sin \omega_L \left( \tau_i(i-k,n) - \tau_i(i-q,p) \right)}{\tau_i(i-k,n) - \tau_i(i-q,p)}$$

where $\omega_U$ and $\omega_L$ are the upper and lower frequencies of the band of interest. If there are multiple bands of interest, (2.20) can be calculated for each band, with the results summed to determine the RIS at block $i$. With this definition of the RIS, as shown in Figure 2.14, spectral shaping can be achieved for PPM signals using this LABI variant, which is called LABI Time Offset (TO).
2.3.2 Spectral Shaping with LABI TO

To obtain some understanding of how LABI TO performs spectral shaping, consider a simple case with a notch at frequency $\omega_b$ and control over a total of $N$ data bits. The Fourier Transform of (2.17) can be written as

$$S(\omega) = P(\omega) \sum_{n=0}^{N-1} e^{-j\omega(nT_f + \tau_{in}(n))} e^{-j\omega d_n}$$  \hspace{1cm} (2.21)

For small values of $\omega d$, we can rewrite (2.21) as

$$S(\omega) = P(\omega) [V - U]$$  \hspace{1cm} (2.22)

where
\[ V = \sum_{n=0}^{N-1} e^{-j\omega(nT_f + t_{pn}(n))} \]  \hspace{1cm} (2.23)

\[ U = j\omega_d \sum_{n} d_n e^{-j\omega(nT_f + t_{pn}(n))} \]  \hspace{1cm} (2.24)

With the desired notch at \( \omega_0 \), the goal is to choose \( d_n \) such that \( U = V \), which will minimize \( |S(\omega_0)| \). The magnitude of \( V \) will depend on the exact time hopping sequence used, and for our analysis we assume that \( \omega_0(nT_f + t_{pn}(n)) \) is a i.i.d. uniform random process from \([0, 2\pi] \). We need to determine

\[ E(|V|) = E\left(\sum_{n=0}^{N-1} \cos(\omega(nT_f + t_{pn}(n))) - j\sum_{n=0}^{N-1} \sin(\omega(nT_f + t_{pn}(n)))\right) \]  \hspace{1cm} (2.25)

where \( E(\cdot) \) is the expectation operator. For sufficiently large values of \( N \), \( V \) can be considered a sum of independent random variables, and it can be shown that the two summations are uncorrelated zero mean Gaussian random variables with variances given by

\[ E\left(\sum_{n=0}^{N-1} \cos(\omega(nT_f + t_{pn}(n)))\right)^2 = E\left(\sum_{n=0}^{N-1} \sin(\omega(nT_f + t_{pn}(n)))\right)^2 = \frac{N}{2} \]  \hspace{1cm} (2.26)

Hence, \( |V| \) is a Rayleigh random variable, whose mean is given by

\[ E(|V|) = \sqrt{\frac{N\pi}{4}} \]  \hspace{1cm} (2.27)

In the limit of \( N_b = 1 \) (a block length is equal to a single bit) and small \( \omega_d \), the optimum LABI TO algorithm will invert all the \( U \) vectors that point in
the $V$ direction. Let us denote the new data bits as $d_i'$ and the sum of this new set of vectors as $U'$. Now, the phases for these vectors are uniform $[\pi, 2\pi]$ (if $V$ is rotated to $\pi / 2$). In this case

$$E(|U'|) = E\left(\alpha_d \sum_{n=0}^{N-1} e^{-j\alpha(nT_f + \phi_m(n))} d_n\right) = \alpha_d \frac{2}{\pi} N$$  \hspace{1cm} (2.28)

Then, the normalized magnitude of $|S(\omega_b)|$ is given by

$$|S(\omega_b)| = \frac{E(|V - jU'|)}{E(|V|)} = 1 - \alpha_b l_d \frac{4}{\sqrt{\pi^2}} \sqrt{N}$$  \hspace{1cm} (2.29)

For a more realistic situation, consider the case when $N_b > 1$ and $N = N_f N_b$. Now, $U$ can be rewritten as

$$U = \omega_b l_d \sum_{i=0}^{N_b - 1} F_i u_i$$  \hspace{1cm} (2.30)

where

$$u_i = \sum_{m=0}^{N_b - 1} d_{iN_b + m} e^{-j\alpha_b ((iN_b + m)T_f + \phi_m (iN_b + m))} = a_i e^{-j\theta_i}$$  \hspace{1cm} (2.31)

$$E(a_i) = E\left(\sum_{m=0}^{N_b - 1} d_{iN_b + m} e^{-j\alpha_b ((iN_b + m)T_f + \phi_m (iN_b + m))}\right)$$  \hspace{1cm} (2.32)

and where $F_i$ denotes the flag bit for block $i$, and $d_n$ in this case is the flag bit padded data sequence where $d_{(i+1)N_b - 1} = F_i$. The quantity $u_i$ can be treated as a single vector with magnitude $a_i$ and phase $\theta_i$ as determined by the summation.
If we assume $d_n$ is a i.i.d. binary random process where for each $n$, $d_n$ can take values of $\{+1, -1\}$ with equal probability, then $u_i$ is a sum of independent random variables.

For large values of $N_b$, similar to the analysis above for $V$, it can be shown that $E(a_i) = \sqrt{N_b\pi}/4$. The goal, as above, is to invert all the vectors $u_i$ that point in the same direction as $V$. If we assume that $\theta_i$ is uniform from $[0, 2\pi]$, then when all the vectors are inverted, let $\theta'_i$ be defined as uniform from $[\pi, 2\pi]$ (again if $V$ is rotated to $\pi/2$). Thus,

$$E[|U|] = \omega_d f_d \sum_{i=0}^{N_f-1} E[a_i \sin \theta_i]$$

$$= \omega_d f_d \frac{2}{\pi} \sum_{i=0}^{N_f-1} E[a_i]$$

$$= \omega_d f_d \frac{2}{\pi} N_f E[a_i]$$

$$= \omega_d f_d \frac{1}{\sqrt{\pi}} N_f \sqrt{N_b}$$

With these results, it can be determined at the normalized spectrum at $\omega_0$ can be written as

$$|S(\omega_0)_{\text{Norm}}| \approx \frac{\sqrt{N_f N_b \pi}}{4} - \omega_d f_d \frac{N_f \sqrt{N_b}}{\sqrt{\pi}} = 1 - \omega_d f_d \frac{2}{\pi} \sqrt{\frac{N}{N_b}}$$

(2.34)
Therefore, for a given $N$, longer block lengths may not be able to achieve the desired spectral shaping. This result matches the simulation results shown in Figure 2.15 quite well. Note that for better spectral shaping with LABI, smaller blocks lengths would be used, and is the same conclusion as with LABI TO. Even though we used the Gaussian approximation for large $N_b$, as Figure 2.15 shows, the results match well for small block lengths.

![Figure 2.15: \( |S(\omega_b)_{\text{norm}}| \) as a function of $N$, and compares the analysis (dotted lines) versus the simulation results (solid lines) for various block lengths $N_b$. Smaller $N_b$ for a given $N$ minimizes $|S(\omega_b)_{\text{norm}}|$ better.](image)

There are some limitations to the above analysis. First, it will only hold for very narrowband notches, as a single frequency is considered. If the notch is too wide, then inverting the $U$ vectors may no longer be sufficient, and
consequently the solution to the problem is a full search to determine the flag bits that minimize the magnitude or power across the entire notch bandwidth. Next, the inversion of the vectors only applies when \( |U| \leq |V| \). At the point when inverting another vector would cause \( |U| \geq |V| \), then a full search would need to be used to determine which vectors (and hence data bits) would need to be inverted such that \( |U| = |V| \) to minimize \( |S(\omega_0)_{\text{Norm}}| \). Note that for LABI (the original algorithm) and LABI PI (described in the next subsection), since there are no \( U \) vectors, the choice becomes choosing the vectors to let \( V \) self-cancel. Lastly, the above analysis assumes that the complementary filter is of length \( N \) bits, and assumes \( N \) is large. While the behavior of LABI TO is similar for small values of \( N \), because it does choose the \( U \) vectors to point as best as possible in the \(-V\) direction, the predicted values of the amplitudes do not match the analysis, due to the use of the Central Limit Theorem in the analysis.

2.3.3 LABI Pulse Inversion

For UWB IR-BPSK systems, even though the pulse polarities can be changed in a similar manner to the original LABI algorithm, LABI must be adapted to account for the pseudo-random time hopping nature. This is because LABI assumes that the time interval between each pulse is the same. Integrating the time hopping nature into LABI, in a manner similar to what is done with
LABI TO, is most easily done by changing the calculations of the RIS metric. Let us rewrite (1.1) as

\[ s(t) = \sum_{i} \sum_{n=0}^{N_b-1} d_b(iN_b + n) p(t - \tau_2(i, n)) \quad (2.35) \]

where \( \tau_2(i, n) = (iN_b + n)T_f - t_{pn}(iN_b + n) \) and is similar to \( \tau_1(\cdot) \) used in (2.17) without the data modulated time offset term. Using a similar method as described in the previous subsection for LABI TO, the RIS for block \( i \) can be written as

\[ RIS(i) = \int | P(\omega) |^2 | H_\tau(\omega) |^2 \left| \sum_{k=-L}^{L-1} \sum_{n=0}^{N_b-1} d_b((i - k)N_b + n)e^{-j\omega \tau_2(i-k,n)} \right|^2 d\omega \quad (2.36) \]

Then for a single notch, (2.36) can be simplified as

\[ RIS(i) = \sum_{k=-L}^{L-1} \sum_{n=0}^{N_b-1} \sum_{q=-L}^{L-1} \sum_{p=0}^{N_b-1} d_b((i - k)N_b + n)d_b((i - q)N_b + p) \left\{ \frac{\sin \omega_c (\tau_2(i-k,n) - \tau_2(i-q,p))}{\tau_2(i-k,n) - \tau_2(i-q,p)} - \frac{\sin \omega_c (\tau_2(i-k,n) - \tau_2(i-q,p))}{\tau_2(i-k,n) - \tau_2(i-q,p)} \right\} \quad (2.37) \]

By using this definition of the RIS as the Viterbi trellis branch metric, spectral shaping for IR-BPSK signals can be achieved using LABI. Figure 2.16 shows a simulated spectrum shaped by this variant of LABI, which is called LABI Pulse Inversion (PI).
2.3.4 Results and Discussion

Due to the complexity of the algorithm, analytical expressions to predict the spectrum are difficult to derive, except for the limiting cases described previously. So simulations are relied on for results. As shown in Figure 2.14 and Figure 2.16, spectral shaping can be achieved with LABI TO and LABI PI. As described in the analysis for LABI TO, smaller block lengths and longer block spans of the filter provide better performance. The trade offs are increased overhead (for smaller block lengths) and increased computational complexity (for longer complimentary filter block spans), respectively. Also, for a given block length and complementary filter block span, the narrower the notch is, the deeper it is, which can be seen from comparing Figure 2.16 and Figure 2.17. Therefore,
these algorithms are better suited for notches of small fractional bandwidths. Lastly, there is a delay (trace-back depth) of approximately five times the length of the trellis state vector of the Viterbi algorithm, and has been verified through simulations as roughly $5N_b N_f$ bits.

One advantage of these algorithms is that they allow the use of any arbitrary pulse waveform, including the pulse shapers like those described in [16]-[17] and the wavelet based waveform in [18]. For example, using the pulse waveform described in [17] with this algorithm, one can both provide a signal spectrum that both complies with the FCC UWB spectral mask, and provides the necessary spectral notches for the various in-band narrowband systems. In addition, these algorithms are insensitive to multipath effects. If a linear, slowly varying multipath channel is assumed, the spectrum of the channel multiplies with the transmitted signal spectrum. The result is a spectrum with less interference in the band of interest. In addition, since both algorithms assume that the time-hopping sequence is provided, special time-hopping codes that are designed to produce spectral shaping can be used [25]. Lastly, one might be concerned with the effects of LABI on the orthogonality of UWB Time hopping sequences. Since only the data bits are changed, which in turn modulate a very small time offset $t_d$, the effects should be minimal, and similar to the effects of having any data modulated using PPM.
One item to note when using this algorithm is the bandwidth of the narrowband system with which we would like to avoid interfering. Depending on the width of the bandpass filter of the narrowband system, our algorithm may not reduce the interference that the narrowband system receives. If the bandwidth of the filter is wider than the notch, then the interference mitigation provided by our algorithm is degraded significantly.

2.3.5 Improvements to LABI PI and LABI TO

Although both LABI PI and LABI TO can spectrally shape UWB IR signals, both methods only vary one aspect of the UWB IR signal to achieve spectral shaping: the time intervals between pulses, or the polarity of the pulses.
A combination of pulse polarities and time offsets can be chosen jointly to achieve the desired spectral shape. One way to achieve this combination is to cascade one LABI variant after the other. We call the cascaded algorithm LABI PITO or LABI TOPI, depending on the ordering of the techniques (Figure 2.18).

![Figure 2.18: Two ways of cascading LABI PI and LABI TO to allow for spectral shaping of UWB IR signals by changing both the time offsets and pulse polarities.](image)

First, LABI TOPI is described, as shown in Figure 2.18(b). Since the LABI TO block is first in the cascade, it takes the input data bits $d(n)$ and creates the flag padded data sequence with the appropriate blocks that are inverted in polarity. This determines the time offsets $\tau_t(n)$ in the output signal. Then the resulting signal is passed to LABI PI block to determine the optimum pulse polarities. The LABI TOPI output signal can then be described as

$$s(t) = \sum_n a_b(n) p(t - \tau_t(n))$$

(2.38)

where $a_b(n)$ can take on values of +1 or -1. Similarly, we can write the output signal of the LABI PITO case as
\[ s(t) = \sum_{n} d_y(n) p(t - \tau_z(n) - a(n)t_d) \]  

(2.39)

where \( a(n) \) can take on values of +1 or -1. In both cases, the block lengths and filter block spans can be independently chosen for both blocks in the cascade. For the results discussed in this dissertation, we assume the parameters for the second block in the cascade to be \( N_b = 2 \) and \( N_f = 6 \), and vary the parameters of the first block.

Figure 2.19 compares the performance of LABI TO, PI, TOPI, and PITO. Performance improves as \( N_f \) increases, which results in an increase in complexity with diminishing returns. It also shows that the cascaded versions perform better than the non-cascaded versions, especially for small \( N_f \). Note that LABI PI outperforms LABI TOPI for \( N_f > 6 \), which is due to choosing \( N_f = 6 \) for the PI block in the cascade. The crossover point would be at a larger value of \( N_f \) if a larger value of \( N_f \) for the PI block in the LABI TOPI cascade is chosen, since LABI TOPI’s performance would increase. From Figure 2.19, to achieve the same spectral shaping performance, one can either use a non-cascaded variant (LABI PI or TO) with a high order of complexity, or a cascaded variant with each component having lower complexity. Since LABI’s performance is limited to the complexity of the Viterbi algorithm, if we choose the number of Viterbi trellis states required as the measure of complexity for comparison purposes, the latter maybe easier to implement since it requires \( 2^{N_f, m} + 2^{N_f, p} \) trellis states, which is
equal to the sum of the trellis states from each block in the cascade, as opposed to the former that requires $2^{N_f}$ trellis states. As an example from Figure 2.19, one can achieve a -14dB notch with either a low complexity LABI PITO ($2^4 + 2^6 = 80$ states) or a high complexity LABI PI ($2^{10} = 1024$ states). In addition, an advantage of cascading is that while the first system in the cascade needs to append a flag bit to every block of data bits, the second acts independent of the data bits. Even though the computational complexity, and consequently performance, is increased, the data bit overhead remains the same.

Figure 2.19: The notch depth of the four LABI variants as complexity increases (filter block span increases). For TOPI and PITO, the second block in the cascade is fixed at $N_f=6$. 
While cascading improves spectral shaping performance for a given level of complexity, one would expect even better performance by combining the two and simultaneously choosing both the set of pulse polarities and the set of time offsets (denoted as LABI BOTH in Figure 2.20). Figure 2.20 compares the spectrum of LABI BOTH with all the other LABI variants; it has better performance than either of the cascaded versions. However, the increase in performance comes at a cost of increasing the complexity, with the Viterbi trellis state vector size increasing to $N_{f,pi} + N_{f,to}$.

![Figure 2.20: The spectrum at the notch for the LABI variants compared with the optimal choice in both pulse polarity and time position.](image)
2.4 Effects of Timing Jitter on LABI TO

Since the LABI variants for UWB IR systems rely on precise timing between the pulses, timing jitter will affect the performance of the algorithm. Hence, we would like to predict its effect on the resulting spectrum. In this section, we will focus on the effects of timing jitter on the spectral notch. First, we include the timing jitter term and write the signal and its Fourier transform as

\[ s(t) = \sum a_n p(t - \gamma_n - \delta_n) \quad (2.40) \]

\[ S(\omega) = P(\omega) \sum a_n e^{-j\omega (\gamma_n + \delta_n)} \quad (2.41) \]

where \( \delta_n \) is the jitter, modeled as a zero mean Gaussian random process with variance \( \sigma^2 \), and \( \gamma_n = t_{pn}(n) + t_d(n) + nT \). In a similar manner to Section 2.3.4, we assume that \(|P(\omega)|\) is relatively constant within a notch, so we ignore its variation. Therefore the energy in the notch as a function of frequency can be written as

\[ E_{jitter}(\omega) = |S(\omega)|^2 = \sum_m \sum_n a_m a_n \cos(\omega(\gamma_m - \gamma_n)) \cos(\omega(\delta_m - \delta_n)) + a_m a_n \sin(\omega(\gamma_m - \gamma_n)) \sin(\omega(\delta_m - \delta_n)) \quad (2.42) \]

From (2.42), we can write an expression for the average effect of jitter as
\[
E\left[ E_{\text{jitter}}(\omega) \right] = \sum_{m} a_m a_n \cos(\omega(\gamma_m - \gamma_n)) E\left[ \cos(\omega(\delta_m - \delta_n)) \right] \\
+ a_m a_n \sin(\omega(\gamma_m - \gamma_n)) E\left[ \sin(\omega(\delta_m - \delta_n)) \right]
\] (2.43)

If we use the identity \( \cos(x) = (e^{ix} + e^{-ix})/2 \), then it can be seen that

\[
E\left[ \cos(\omega(\delta_m - \delta_n)) \right] = E\left[ \frac{e^{j\omega(\delta_m - \delta_n)} + e^{-j\omega(\delta_m - \delta_n)}}{2} \right]
\] (2.44)

Thus, for \( m \neq n \), let \( \Delta = \delta_m - \delta_n \), and thus \( \Delta \) is also a Gaussian random variable with variance \( 2\sigma^2 \). Therefore, one can write

\[
E\left[ e^{j\Delta} \right] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}2\sigma} e^{j\omega\Delta} e^{\frac{1}{4\sigma^2}(\Delta^2)} d\Delta
\]

\[
= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}2\sigma} e^{\frac{1}{4\sigma^2}(\Delta^2)} d\Delta
\]

\[
= e^{\frac{2\sigma^2}{\omega}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}2\sigma} e^{\frac{1}{4\sigma^2}(\Delta^2)} d\Delta
\] (2.45)

Because \( \Delta' = \delta_n - \delta_m \) is also a Gaussian random variable with variance \( 2\sigma^2 \), (2.45) is also valid for the second term in (2.44). Therefore, it follows that

\[
E\left[ \cos(\omega(\delta_m - \delta_n)) \right] = e^{\frac{2\sigma^2}{\omega^2}}
\] (2.46)

Similarly, it can be shown that \( E\left[ \sin(\omega(\delta_m - \delta_n)) \right] = 0 \). Therefore, (2.43) becomes
\[
E\left[ E_{\text{jitter}}(\omega) \right] = \sum_{n} 1 + e^{\sigma^{2} \omega^{2}} \sum_{n,m \neq n, m} \cos(\omega(\gamma_{m} - \gamma_{n})) \quad (2.47)
\]

Similarly, we can write the energy in the notch without jitter as

\[
E_{\text{nojitter}}(\omega) = \sum_{n} 1 + \sum_{n,m \neq n, m} \cos(\omega(\gamma_{m} - \gamma_{n})) \quad (2.48)
\]

Next, we take the ratio of (2.47) to (2.48), and if we ignore the first term from both equations since they are both much smaller than the second term, we obtain

\[
\frac{E\left[ E_{\text{jitter}}(\omega) \right]}{E_{\text{nojitter}}(\omega)} \approx e^{\sigma^{2} \omega^{2}} \quad (2.49)
\]

We can equivalently write the ratio of the power with jitter to the power without jitter, and rewrite (2.49) as

\[
E\left[ P_{\text{jitter}} dB(\omega) \right] - P_{\text{nojitter}} dB(\omega) \approx 10\omega^{2} \sigma^{2} \log_{10}(e) \quad (2.50)
\]

(2.50) predicts that the effect of jitter is to “fill-in” the notch, and is a function of the frequency and jitter variance. As Figure 2.21 shows, the analysis matches the simulation results well, with jitter standard deviations of 10ps and 40ps degrading the notch depth by about 1dB and 7dB, respectively. Hence, the algorithm is sensitive to timing jitter. However, UWB timing circuits with a rms jitter of less than 10ps have been reported [26], so the performance degradation is negligible under these circumstances.
Figure 2.21: Increase in notch power vs. rms timing jitter.

2.5 Summary

A coding technique based on LABI is presented to spectrally shape UWB IR signals away from in-band narrowband system frequencies for the purposes of interference mitigation. To achieve spectral shaping, LABI PI varies the pulse polarities while LABI TO varies the timing interval between pulses. It is shown that LABI PI can produce a 10MHz wide notch with a depth of 18dB at an arbitrary frequency. While these algorithms are generally not useful for spectrally shaping the UWB signal to satisfy the FCC spectral mask, it can be used with any pulse shaper to further enhance the spectral characteristics of the signal’s spectrum, including pulses that fit within the FCC spectral mask. Cascaded variants of LABI PI and TO – called LABI PITO and LABI TOPI – can provide
even better spectral shaping performance without increasing data overhead, at a cost of increased complexity. One advantage of the cascaded system is the use of two lower complexity Viterbi algorithm implementations to achieve the same notch depth, which is easier to implement. Lastly, the effect of timing jitter on the performance of the algorithm is examined. It is determined that the algorithm is sensitive to timing jitter, with a degradation of 7dB to the notch depth when the rms timing jitter is 40ps, while the degradation is negligible for timing jitter of less than 10ps.

3 LABI for UWB OFDM

In Chapter 2, a binary block code called LABI is applied to UWB IR signals to insert in-band spectral notches into the signal’s spectrum. For OFDM based UWB systems, such as MB-OFDM systems, interference mitigation techniques for in-band spectral notching have been reported, such as the cancellation carriers in [27]. However, the cancellation carriers cannot be used to transmit data. This chapter will focus on spectrally shaping the OFDM spectrum by changing the data modulated on each of the OFDM subcarriers. Specifically, a coding algorithm also based on LABI will be used to spectrally shape UWB MB-OFDM based systems for in-band interference mitigation purposes.

This chapter will be organized as follows. First, in Section 3.1, the coding algorithm, which is called LABI for OFDM, is described. Then, the simulation results and spectral shaping performance will be presented in Section 3.2.

3.1 LABI for OFDM

While all data for LABI PI or LABI TO is coded for spectral shaping purposes, for OFDM-based systems, only the data modulated on the subcarriers in the vicinity of the desired notch will be coded, with the remaining subcarriers left
uncoded to minimize overhead. A simplified block diagram of the algorithm is shown in Figure 3.1. A coding example is shown in Figure 3.2.

![Block diagram of LABI for OFDM](image)

**Figure 3.1:** Block diagram of LABI for OFDM.

![Data grouping for LABI for OFDM](image)

**Figure 3.2:** Data grouping for LABI for OFDM. The QPSK data bits are denoted as \( d(n) \). Subcarriers 3 and 4 are nulled out, subcarriers 1-2 and 5-6 are coded, and subcarriers 0 and 7 are uncoded. The block length \( N_b \) consists of multiple OFDM symbols. \( N_f = 4 \).

Let \( d_i(k) \) be the QPSK modulated data transmitted on subcarrier \( k \) for OFDM symbol \( l \), and \( K \) QPSK symbols are modulated per OFDM symbol. If a
subcarrier is to be coded, then a flag bit is inserted into that subcarrier’s data
stream every \( N_b \) data symbols, where \( N_b \) is the block length. This flag bit
describes the polarity of a block of \( N_b - 1 \) QPSK data symbols, which for a coded
block of data at subcarrier \( k \) is defined as \( d_{\text{block}}(k) \) can be expressed as

\[
d_{l,\text{block}}(k) = \begin{cases} 
F_k d_{l}(k) & l = 0, 1, ..., N_b - 2 \\
F_k & l = N_b - 1 
\end{cases}
\]  

(3.1)

where \( F_k \) is the flag bit that describes whether the block is inverted. Recall from
Chapter 1, (1.11) is the baseband representation of an OFDM symbol, and is
rewritten here as

\[
x_{\text{symbol}}(n) = \frac{1}{K} \sum_{k=0}^{K-1} d(k) e^{j 2 \pi n k / K}.
\]  

(3.2)

Thus, the OFDM symbol \( x(lK + n) \) can be written as

\[
x(lK + n) = \frac{1}{K} \sum_{k=0}^{K-1} d_{l}(k) e^{j 2 \pi n k / K}
\]  

(3.3)

where \( n = 0, 1, ..., K - 1 \) and \( l = 0, 1, ..., N_b - 1 \), and thus a block of OFDM symbols
consists of \( N_b \) OFDM symbols as defined by (3.3). Equation (3.3) can be
rewritten as the sum of the coded and uncoded components as

\[
x_{h}(lK + n) = x_{u}(lK + n) + x_{c}(lK + n)
\]  

(3.4)
where \( x_u(lK+n) \) and \( x_c(lK+n) \) are the uncoded and coded components, respectively, and are defined as

\[
x_u(lK+n) = \frac{1}{K} \sum_{k=0}^{K-1} d_u(k) e^{j2\pi nk/K} \quad (3.5)
\]

\[
x_c(lK+n) = \frac{1}{K} \sum_{k=0}^{K-1} d_{\text{block}}(k) e^{j2\pi nk/K} \quad (3.6)
\]

To examine the spectral characteristics of the resulting coded OFDM signal, the Fourier transform of (3.4) needs to be performed. Due to the time-limited duration of the OFDM symbol, as explained in Chapter 1, each subcarrier contributes a sinc-like spectrum where the nulls of the sinc function occur at the center frequencies of the other subcarriers. However, in the frequencies between the subcarriers, the sinc contributions to the spectrum are not zero. Thus, to examine the spectrum between the subcarriers, the signal will be upsampled by \( Q \) by repeating each data point \( Q \) times. Then, for a single symbol, the upsampled Fourier transform of (3.2) can be written as

\[
X(q) = \sum_{n=0}^{KQ-1} x_{\text{symbol}}(n) e^{-j2\pi nq/KQ}
\]

\[
= \frac{1}{K} \sum_{k=0}^{K-1} \sum_{n=0}^{Q-1} d(k) e^{j2\pi n(k-q)/K} \quad (3.7)
\]

where \( q \) indexes the frequency term. Similarly, the Fourier transform of a block of symbols can be written as
Decomposing (3.8) into the uncoded and coded components, we obtain

\[ X_b(q) = X_{b,u}(q) + X_{b,c}(q) \]  

(3.9)

where

\[ X_{b,u}(q) = \frac{1}{K} \sum_{k \text{ uncoded}} d_i(k) e^{j2\pi n(k-q)/(N_Q)} \]  

(3.10)

\[ X_{b,c}(q) = \frac{1}{K} \sum_{k \text{ coded}} \sum_{n=0}^{N_q-1} d_{i,\text{block}}(k) e^{j2\pi n(k-q)/(N_Q)} \]  

(3.11)

Assuming the desired notch is between subcarriers \( k_1 \) and \( k_2 \), all the subcarriers between \( k_1 \) and \( k_2 \) inclusive are nulled, or equivalently \( d_i(k) = 0 \) for \( k \in [k_1, k_2] \). Then, \( N_f \) subcarriers on either side of the notch are coded. Thus, for each block, there are \( N_f \) flag bits that need to be chosen to minimize the power in the notch, which is given by

\[ P_{\text{notch}} = \sum_{q \in [k_1, k_2]} |Y_b(q)|^2 = \sum_{q \in [k_1, k_2]} |Y_{b,u}(q) + Y_{b,c}(q)|^2 \]  

(3.12)

There are \( 2^{N_f} \) possible flag bit combinations, and a full search is required to determine the set that minimizes (3.12). Intuitively, this algorithm functions similarly to LABI TO and LABI PI, where the goal is to choose the flag bits such that the vector \( X_{b,c}(q) \) cancels out the vector \( X_{b,u}(q) \) as well as possible.
As an example, Figure 3.2 shows the setup of LABI for OFDM for $K = 7$, $k_1 = 3$, $k_2 = 4$, $N_f = 4$, $N_b = 3$, and the coded subcarriers are divided equally on both sides of the notch. The notch is located between subcarriers 3 and 4, and all subcarriers between them inclusive are nulled. The flag bit $F_i$ changes the polarity of the block of data modulated on subcarrier 1, consisting of \{d(1), d(7), F_1\}, with similar setups on subcarriers 2, 5, and 6.

To predict the spectral shaping performance of our coding technique, the OFDM spectrum will be approximated as a summation of sinc functions centered at the subcarriers’ center frequencies. Thus, a single OFDM symbol waveform with $K$ subcarriers can be written as

$$x(t) = p_{r,T}(t) \sum_{k=0}^{K-1} d(k) \cos(2\pi k/K + \theta)$$  \hspace{1cm} (3.13)

where $T_s$ is the symbol duration, $\theta$ is a random phase, $p_{r,a}(t)$ is the rectangle function defined in (1.8), and $d(k)$ is assumed to be a i.i.d. and can take on values of \{+1, +j, −1, −j\} with equal probability. Thus, the autocorrelation of function can be written as

$$R_x(t_1, t_2) = E[x(t_1)x^*(t_2)] = \frac{1}{2} \sum_{k=0}^{K-1} \sum_{k'=0}^{K-1} A(t_1, t_2) B(k, k') C(t_1, t_2, k, k')$$ \hspace{1cm} (3.14)

where $x^*$ is the complex conjugate of $x$, and

$$A(t_1, t_2) = p_{r,T}(t_1) p_{r,T}(t_2)$$ \hspace{1cm} (3.15)
\[
B(k,k') = E\left[ d(k)d^*(k') \right] = \delta_{k,k'} \quad (3.16)
\]
\[
C(t_1,t_2,k,k') = \cos(2\pi(t_1k-t_2k')/K) + \cos(2\pi(t_1k+k')/K+2\theta) \quad (3.17)
\]
where \( \delta \) is defined as
\[
\delta_{k,k'} = \begin{cases} 
1 & k = k' \\
0 & \text{otherwise}
\end{cases} \quad (3.18)
\]
Therefore, (3.14) can be simplified as
\[
R_s(t_1,t_2) = \frac{1}{2}A(t_1,t_2) \sum_{k=0}^{K-1} \cos(2\pi(t_1-t_2)k/K) \quad (3.19)
\]
It can be shown that (3.15) can be rewritten as
\[
A(\tau) = \begin{cases} 
1-|\tau/T_s| & |\tau|<T_s \\
0 & \text{otherwise}
\end{cases} \quad (3.20)
\]
where \( \tau = t_1 - t_2 \), and therefore
\[
R_s(\tau) = \frac{A(\tau)}{2} \sum_{k=0}^{K-1} \cos(2\pi\tau k/K) \quad (3.21)
\]
The power spectral density is given by the Fourier transform of the autocorrelation function [12], and can be written as
\[
PSD(\omega) = \frac{1}{2} \sum \text{sinc}^2 \left( (\omega-\pi k/K)T_s/2 \right) + \text{sinc}^2 \left( (\omega+\pi k/K)T_s/2 \right) \quad (3.22)
\]
Thus, (3.22) describes the uncoded spectrum, and with two subcarriers nulled, the predicted and simulated notch powers match well and show a -8dB notch power.
Analytical expressions for the power spectrum for a coded symbol are difficult to derive, due to the difficulty in deriving an expression for the correlation between a flag bit and the data on the subcarriers (i.e. $E[F_d^*d(k')]$). So simulations are performed to obtain the spectrum and notch power for this case, and the notch power as a function of the number of coded subcarriers for $N_b=1$ is shown in Figure 3.3, which is the best possible performance of this algorithm.

![Simulated notch power for $N_b=1$ as a function of the number of coded carriers $N_f$.](image)

**Figure 3.3:** Simulated notch power for $N_b=1$ as a function of the number of coded carriers $N_f$.

### 3.2 Results and Discussion

Simulations are performed to obtain results, and the spectrum and notch power are examined. For the coded spectrum shown in Figure 3.4, the system
parameters are $N_b = 2$, $N_f = 10$, $K = 128$, where two subcarriers are nulled. As shown in Figure 3.4, notches of -28dB are achievable, and is an improvement of 20dB over the uncoded spectrum. Next, Figure 3.5 shows the improvement of the notch power of the LABI coded spectrum over the uncoded spectrum for selected values of $N_b$ and $N_f$. In general, smaller block lengths ($N_b$) and more coded subcarriers ($N_f$) provide better spectral shaping performance. The tradeoffs are the increased overhead for both smaller values of $N_b$ and larger values of $N_f$, and the increased computational complexity for larger values of $N_f$.

![Figure 3.4: LABI coded OFDM spectrum (solid) compared with the uncoded spectrum (dotted). An improvement of 20dB can be seen for $N_b=2$, $N_f=10$, $K=128$.](image)
Figure 3.5: Notch power improvements over uncoded notch power for selected block lengths ($N_b$) and coded carriers $N_f$. Two subcarriers are nulled.

As a measure of the overhead used to obtain spectral shaping, we will use the ratio of the effective number of subcarriers used for spectral shaping purposes (coded and nulled) to the total number of subcarriers $K$. Because each of the coded subcarriers transmits $N_b - 1$ data symbols per block, those subcarriers can be considered to be $(N_b - 1)/N_b$ of an uncoded subcarrier. So the effective number of subcarriers is the ratio $(N_b - 1)/N_b$ multiplied by the number of coded subcarriers $N_f$. Thus, the effective overhead of this technique is defined as

$$
\text{Effective Overhead} = \frac{\frac{N_b - 1}{N_b} N_f + N_{\text{nulled}}}{K}
$$

(3.23)
where $N_{\text{null}}$ is the number of nulled subcarriers. Equation (3.23) also shows that smaller values of $N_b$ and larger values of $N_f$ result in increased overhead.

For the example shown in Figure 3.4, the effective overhead is equal to approximately 3.9% for a notch of -28dB. As a comparison, if instead subcarriers were nulled such that the power in the same notch frequencies is also -28dB, then it can be seen that from Figure 3.6, which shows the effective overhead as a function of the notch depth, that an effective overhead of 73.4% is required. Thus, our algorithm provides the desired notch power at a significant reduction in overhead.

![Figure 3.6: Effective overhead as a function of the notch depth for the coded and uncoded spectrums.](image)
Next, the effects of quantization on the spectral shaping performance are examined. The metric used is the notch power at different quantization resolutions. The simulation results for various quantization resolutions and block lengths $N_b$ are shown in Figure 3.7 for $N_f = 10$. For $N_b \geq 4$, it can be seen that the minimum resolution required to maintain the notch power is six bits, as resolutions greater than six bits only decreases the notch power by less than one dB. For $N_b = 2$, the minimum resolution is eight bits. Recall that due to the time windowing of an OFDM symbol, the spectrum of each subcarrier has a sinc-like spectrum, with the nulls of the sinc occurring at the center frequencies of the other subcarriers and thus contributing no power at all. So for a notch that spans the entire frequency range between two subcarriers’ frequencies, any quantization noise raises the noise floor, and as a result power is present at those frequencies. This is the reason for the small decrease in notch power as the resolution increases. The effects of quantization on the notch power can be seen in Figure 3.8, which shows the power spectrum at various quantization resolutions for $N_b = 4$ and $N_f = 10$ with two subcarriers located at 70MHz and 74MHz nulled.
Figure 3.7: Notch power as a function of quantization resolution for various block lengths. \( N_f = 10 \).

Figure 3.8: The spectrum of quantized LABI coded OFDM at various resolutions. \( N_b = 4, N_f = 10 \).
Finally, the effects of coding on the receiver will be briefly discussed. If an error occurs in decoding the symbol on one of the subcarriers, it is possible that the error is in detecting the flag bit. An error in the flag bit would create errors in decoding the $N_b - 1$ QPSK data symbols in that block. However, due to the bit interleaving in the MB-OFDM specification (Chapter 1 Section 1.3.5), which introduces time and frequency diversity, any bit errors would be spread among the bits at the output of the receiver’s de-interleaver, and thus should have a minimal effect on the bit error performance of the system.

### 3.3 Summary

In this chapter, a coding technique based on LABI is used to spectrally shape a UWB MB-OFDM signal away from in-band narrowband frequencies is described. Flag bits, which describe the polarity of a block of bits, are inserted to describe the polarity of a block of QPSK data modulated on selected subcarriers in the vicinity of a notch. The polarities of those flag bits are chosen such that the notch power is minimized. For example, notches of -28dB can be created by nulling two subcarriers and coding ten subcarriers. The effective overhead is 3.9%, and a notch can be created with minimal overhead. For this notch, it can be shown that to maintain the notch depth, the minimum quantization resolution is eight bits.
This chapter, in part, has material submitted for publication in “A Coding Technique for Spectral Shaping UWB OFDM Signals,” *IEEE Transactions on Wireless Communications*. The authors are Joe Jamp and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.
4 Hardware Considerations for a UWB Spectral Encoded System

In the previous chapters, coding techniques for spectrally shaping UWB IR and OFDM signals are presented. In this section, another spectral shaping technique will be described, called Spectral Encoding [20]. This technique has similarities with CDMA, and has the advantage of inserting spectral notches in a transmitted signal’s spectrum, as well as rejecting energy in the same notch frequencies at the receiver.

In this chapter, a real-time implementation of the Spectral Encoded transmitter will be described, and the system parameters will be discussed. First, we will briefly review the Spectral Encoding technique. Then, the real-time implementation will be described, along with two possible hardware implementations which will be compared. Their system parameters tradeoffs will be discussed and values will be chosen.

4.1 Spectral Encoded UWB

Spectral Encoding can be considered as a frequency domain counterpart to DS/CDMA. This technique multiplies the input signal in the frequency domain by a spreading sequence, as shown in Figure 4.1. The resulting signal is spread in
Similar to DS/CDMA, multiple access is achieved by assigning different spreading sequences to different users.

As reported in [20], the Spectral Encoded system can be extended to achieve spectral shaping to combat narrowband interference. As shown in Figure 4.1, a Spectral Encoded system can achieve narrowband interference suppression by nulling some of the spreading chips. The resulting signal is still spread in time, but has a spectral null at the narrowband interferer frequencies. Thus at the transmitter, a transmitted Spectral Encoded signal has less energy at the narrowband interferer frequencies. At the receiver, the de-spreading operation is performed by multiplying the received signal with the complex conjugate of the transmitter’s spreading sequence, which rejects energy from the narrowband interference. Hence, a main advantage of this technique is that it inherently provides interference mitigation at the transmitter and interference rejection at the receiver. This is in contrast with UWB systems based on DS/CDMA or PPM where notch filters or other forms of spectral shaping may be needed, such as those described in the previous chapters and in [10][16][17][23][25].

The block diagram of the Spectral Encoding transmitter is shown in Figure 4.1. It is composed of three blocks: the Fourier transform block, the Spectral Encoding block, and the inverse Fourier transform block. Here, the input signal is a second derivative of the Gaussian monopulse (Figure 4.1(a)), although in general, the input signal can be of any shape. Then, after performing the Fourier transform on the input signal, the spectrum is then multiplied by the spreading
sequence (Figure 4.1(b)). After performing an inverse Fourier transform on the encoded signal, the output signal, shown in Figure 4.1(c), is spread in time and contains the desired spectral notches.
Figure 4.1: Block diagram of the real-time Spectral Encoded UWB transmitter. (a) The unencoded UWB input signal waveform and spectrum, with a narrowband interferer at 3.6GHz. (b) Spectral Encoded spreading sequence of chip bandwidth 66.67MHz. The chips at the interference frequencies are nulled out. (c) The Spectral Encoded output waveform (time-spread) and spectrum.
4.2 Real-Time Implementation Overview

A possible implementation of a Spectral Encoded UWB transmitter is a SAW-based real-time implementation. The block diagram is shown in Figure 4.2, where the Fourier transform and inverse Fourier transform blocks are implemented using chirp transforms [30], where for the Fourier transform block the signal \( x(t) \) is multiplied by a chirp (linear frequency modulated) signal denoted by \( \cos(\omega_0 t - \beta t^2) \), and the resulting output is sent through a filter with an impulse response of \( u(t) \cos(\omega_0 t + \beta t^2) \), where \( u(t) \) is the unit step function. These chirp filters can be generated using SAW filters [20][30][31], or tap-delay line filters [30].

\[
\begin{align*}
\text{Fourier Transform} & \\
& \text{Mixer} \quad \cos(\omega_0 t + \beta t^2) \\
& \text{SAW Filter} \\
& \cos(\omega_0 t - \beta t^2) \\
& x(t) \\
& \text{Mixer} \\
& \text{Inverse Fourier Transform} & \\
& \text{Mixer} \quad \cos(\omega_0 t - \beta t^2) \\
& \text{SAW Filter} \\
& \cos(\omega_0 t + \beta t^2) \\
& X(\omega) \\
& H_{ps}(\omega) \\
& s(t)
\end{align*}
\]

Figure 4.2: Block diagram of the real-time Spectral Encoded UWB system.

Consider a single symbol as the input to the Fourier transform block as \( x(t) \), where \( x(t) \) is an even signal centered about \( T_s / 2 \) and is time limited to \( t \in [0, T_s] \), where \( T_s \) is the transmitted symbol duration and \( x(t) \) has a duration smaller than \( T_s \) to allow for time-spreading. From Figure 4.2, after multiplying
$x(t)$ by a chirp signal and convolving by $u(t)\cos(\omega_0 t + \beta t^2)$, the output of the Fourier transform block can be written as

$$X(2\beta t) = \int_{0}^{T} x(\tau) \cos(\omega_a \tau - \beta \tau^2) \cos(\omega_0 (t - \tau) + \beta (t - \tau)^2)$$

$$= X_r(2\beta t) \cos(\omega_0 t + \beta t^2) - X_i(2\beta t) \sin(\omega_0 t + \beta t^2)$$

(4.1)

where $2\beta$ and $\omega_a$ are the slope and the starting frequency of the chirp, $X_r(2\beta t)$ and $X_i(2\beta t)$ are the real and imaginary parts of the Fourier transform, so $X(2\beta t)$ is the time domain representation of the Fourier transform of $x(t)$ where the frequency evolves with time as given by the relationship $\omega = 2\beta t$. $X_r(\omega)$ and $X_i(\omega)$ are given by

$$X_r(\omega) = \int_{0}^{T} x(\tau) \cos(\omega \tau) d\tau$$

(4.2)

$$X_i(\omega) = \int_{0}^{T} x(\tau) \sin(\omega \tau) d\tau.$$ 

(4.3)

$X(\omega)$ is only a valid Fourier transform during the time $x(t)$ is fully contained in the chirp filter labeled $\cos(\omega_0 t - \beta t^2)$. Assuming this filter is truncated to length $T_i$, then the Fourier transform is valid in the time interval $t \in [T_s, T_i]$. Correspondingly, the frequencies for which the Fourier transform is valid are $\omega \in [2\beta T_s, 2\beta T_i]$. Hence, $\beta$ and $T_i$ must be chosen such that $2\beta T_s = \omega_L$ and $2\beta T_i = \omega_U$, where $\omega_L$ and $\omega_U$ are the lower and upper frequencies of the desired Fourier transform. In a UWB context, possible values
could be 3.1GHz and 5GHz, or 3.1GHz and 10.6GHz for the entire UWB frequency range. Values for the other parameters will be discussed in the next section.

After the Fourier transform block, the signal is windowed in the “frequency domain” to only have energy where the Fourier transform is valid, which is in the interval $\omega \in [2\beta T_s, 2\beta T_i]$. Then it is multiplied by $H_{PN}(\omega)$ given by

$$H_{PN}(\omega) = \frac{8\beta}{\pi} \left[ PN_R(\omega) \cos(\omega T_i) + PN_I(\omega) \sin(\omega T_i) \right] \quad (4.4)$$

where $PN(\omega) = PN_R(\omega) + jPN_I(\omega)$ is the spreading sequence in the interval $\omega \in [2\beta T_s, 2\beta T_i]$, $PN_R(\omega)$ and $PN_I(\omega)$ consist of a sequence of $\pm 1$ (or zeros for notching), and the constant $8\beta / \pi$ is the normalization factor used to cancel out the constants that would otherwise appear in (4.5). After multiplication by $H_{PN}(\omega)$, the “spectrum” is passed through the inverse Fourier transform block, and the transmitted signal $s(t)$ is given by

$$s(t) = \int_{-\infty}^{\infty} X_R(\omega) PN_R(\omega) \cos(\omega(t - T_i)) - X_I(\omega) PN_I(\omega) \sin(\omega(t - T_i)) d\omega \quad (4.5)$$

where $\omega = 2\beta \tau$ in (4.5). Recall that the Fourier transform is only valid during the interval that the signal is fully contained in the convolution filter. Using similar arguments, the encoded signal $s(t)$ at the output of the inverse Fourier transform block is only valid while the “spectrum” is fully contained in the convolution
filter, and thus only valid within the interval $t \in [T_i, T_i + T_s]$, and must be truncated in time to this interval. Since the input to the system is located at $t \in [0, T_s]$, it must be shifted to $t \in [T_i, T_i + T_s]$ to obtain a valid inverse Fourier transform [30], and the $\cos(\omega T_i)$ in (4.4) provides this time shift. The resulting symbol length is $T_s$. Figure 4.3 shows the operation of this algorithm in the time domain, where the signals have been truncated to where they are valid signals. It also shows the various time delays through the Fourier transform and inverse Fourier transform blocks.
Figure 4.3: Operation of the Real-Time Spectral Encoded Signal, where the signals shown are: (a) the input signal centered about $T_s/2$, (b) the Fourier Transform, (c) after multiplying by the spreading sequence $H_{PN}$, and (d) the transmitted signal $s(t)$.

4.3 Hardware Considerations

In this section, two possible hardware implementations of the real-time Spectral Encoded system are discussed: a SAW based implementation, and a monolithic one. The choice of system parameters will be discussed, and conclusions will be given. As a comparison metric, the ratio of the power in the notch to the total power will be used, since the goal at the transmitter is to
spectrally shape the transmitted signal to contain as little power in the narrowband interference frequencies as possible.

4.3.1 SAW Based Implementation

From the previous section, there are three parameters – $\beta$, $T_s$, $T_i$ – but only two equations relating them: $2\beta T_s = \omega_k$ and $2\beta T_i = \omega_u$. Because the parameters are interdependent, the symbol duration $T_s$ will be examined first. Since $s(t)$ is a truncated version of the ideal spread-time signal, $T_s$ needs to be chosen such that it contains most of the time-spread energy. By allowing $T_s$ to approach infinity, $s(t)$ approaches the desired signal, since the effects of windowing on the Fourier transform and inverse Fourier transform diminishes with increasing $T_s$.

The SAW chirp filter length $T_i$ can be calculated as

$$T_i = T_s\omega_u / \omega_k. \quad (4.6)$$

Large values of $T_s$, and equivalently $T_i$, are a challenge since long delay SAW filters are both challenging to realize and cost more to fabricate as the time-bandwidth product grows [32]. Therefore, smaller values of $T_i$, and equivalently $T_s$, are desired. However, smaller values of $T_s$ can remove spectral characteristics, such as the spreading sequence chips and notches. To see the
effect of smaller values of $T_s$ on the spectrum, recall that the output of the inverse Fourier transform is given by (4.5) is only valid for $t \in [T_s, T_s + T_s]$, and is a windowed version of the ideal spread-time waveform. Conceptually, applying a rectangular window to the desired signal of length $T_s$ in the time domain causes the desired spectrum to be convolved with a sinc function. Therefore, the energy from neighboring chips is “spilled” into the notch. To determine the amount of energy spilled over by a single chip, let us define the chip as

$$S_{\text{chip}}(\omega) = \Pi_{W/2}(\omega - \omega_{0,\text{chip}})$$

(4.7) where $W$ is the bandwidth of the chip, $\omega_{0,\text{chip}} = (\omega_{U,\text{chip}} + \omega_{L,\text{chip}})/2$ is the center frequency of the chip, $\omega_{U,\text{chip}}$ and $\omega_{L,\text{chip}}$ are the upper and lower frequencies of the chip, and $\Pi_a(x)$ is the rectangular function defined as

$$\Pi_a(x) = \begin{cases} 1 & |x| \leq a \\ 0 & \text{otherwise} \end{cases}$$

(4.8)

The spectrum of a single chip due the time windowing is (4.7) convolved with the Fourier transform of (4.8), which is $\sin(\omega T_s / 2) / (\pi \omega T_s / 2)$, and is given by

$$S_{\text{chip, win}}(\omega) = S_{\text{chip}}(\omega) \ast \frac{\sin(\omega T_s / 2)}{\pi \omega T_s / 2} = \int_{\omega_{L,\text{chip}}}^{\omega_{U,\text{chip}}} \frac{\sin((\omega - \gamma)T_s / 2)}{\pi(\omega - \gamma)T_s / 2} d\gamma$$

(4.9)

where $\ast$ denotes the convolution operation.

Since notches are created from a zero-valued chip, the power outside the chip’s bandwidth is examined to see how much power is “spilled” into the desired
null. Specifically, we consider the power for frequencies \( \omega > \omega_{L,chip} \) in the adjacent chip bandwidth. The power in the desired null is therefore given by (from Parseval’s theorem)

\[
P_{null} = \frac{1}{T_s} \int_{\omega_k}^{\omega_l} |S_{chip,win}(\omega)|^2 \, d\omega
\]  

(4.10)

which integrates the power over the notch frequencies caused by the neighboring chip. Figure 4.4 shows the spilled power in the adjacent chip as a function of \( T_s \), with a chip bandwidth of 66.67MHz. As \( T_s \) increases, the power decreases at roughly \( 1/T_s \), primarily due to the main-lobe of the sinc function. Similar results can be obtained for the chips on the lower frequency side of the notch (\( \omega < \omega_{L,chip} \)).
Figure 4.4: The spillover power in the neighboring chip’s frequencies as a function of $T_s$, caused by the time windowing of the Fourier Transform on a single chip of the spreading sequence. The power decreases as $1/T_s$ due to the contribution from the main-lobe of the sinc function.

For a single nulled chip in the middle of the spreading sequence, the power spilled by the two adjacent chips is effectively doubled, and thus is roughly 3dB higher than what is shown in Figure 4.4. Hence, a lower bound on the notch power can be obtained from Figure 4.4, which is -23dB at 35ns (3dB above the results shown), and limits the smallest $T_s$ that can be used for a desired notch power level. The above analysis assumes the power spilled into the notch only comes from the adjacent chips, where in reality power is contributed to the notch from all chips, and depends on the location of the nulled chips in the spreading sequence. If it is assumed that the notch is located at the $k$th chip for $K$ chips, then the actual contribution to the notch from all the chips can be written as
\[ P_{\text{notch}} = \sum_{i=0}^{k-1} P_{\text{chip}}(i) + \sum_{i=k+1}^{E-1} P_{\text{chip}}(i) \]  
(4.11)

where \( P_{\text{chip}}(i) \) is the contribution from the \( i \)-th chip to the notch, and is similar to (4.10), with the only difference being the integrand of (4.9), which would be modified to be

\[
S_{\text{chip,win},i}(\omega) = \int_{\omega_{\text{chip},i}}^{\omega_{\text{chip},i+1}} \frac{\sin((\omega - \gamma)T_s/2)}{\pi(\omega - \gamma)T_s/2} d\gamma
\]  
(4.12)

where the index \( i \) denotes the contribution from the \( i \)-th chip, and the integral bounds change for different \( i \). Equation (4.11) is also plotted in Figure 4.4.

As an example, for the target system, the input to the system is the first derivative of the Gaussian monopulse. The analyzed signal bandwidth is 1GHz, and the spreading sequence consists of 15 chips, two of which are set to zero for spectral shaping purposes. Figure 4.5 shows the simulated notch power as a function of \( T_s \), and the notch power is slightly higher than the predicted one in (4.11). There are diminishing returns to the notch depth as \( T_s \) increases, since as described above, the power in the adjacent chip decreases in dB as \(-20\log T_s\). To obtain another 10dB reduction in the notch power, \( T_s \) would need to be approximately three times larger (e.g. a -20dB notch with \( T_s = 30\text{ns} \) would require roughly \( T_s = 100\text{ns} \) to obtain a -30dB notch).
Another important parameter is the spreading sequence. Longer spreading sequences are desired, since the length determines the number of simultaneous users. Each chip that is set to zero reduces the number of users, since the auto-correlation and cross-correlation properties of these sequences will change. This effect can be reasonably ignored for long spreading sequences, since the cross-correlation relative to the auto-correlation changes very little for long sequences. As an example, for a spreading sequence of length 1023, zeroing out ten bits can, in the worst case, reduce the auto-correlation by about 1%, and increase the cross-correlation by about 1%. Figure 4.6 shows the power in the adjacent chip versus chip bandwidth for various $T_s$, and the arrows show the point of minimum
bandwidth for each $T_s$ for a desired notch power of -28dB. For $T_s = 40$ns the bandwidth is approximately 34MHz.

\[
\text{Figure 4.6: The spillover power in the adjacent chip as a function of the chip bandwidth for various } T_s. \text{ The arrows point to the optimal bandwidth for each } T_s.
\]

Lastly, one must choose an appropriate value for $\omega_c$, the chirp carrier frequency, since the Fourier transform of the signal in (4.1) is located on that carrier frequency. From [30], $\omega_c$ must be chosen to be much higher than the bandwidth of the chirp. However, one would like to keep this value as small as possible for easier implementation, as higher operating frequencies for SAW filters are challenging to design [32]. Since the real-time implementation described above performs spectral encoding at baseband, from simulations, the requirement is to choose $\omega_c > 4\omega_U$. 
For the target system operating in the 3-5GHz frequencies with 1GHz bandwidth, using the results above we obtain the following parameters: $T_s = 35\text{ns}$, $T_i = 58.3\text{ns}$, $\omega_a = 20\text{GHz}$, and the spreading sequence length is 15, with a chip bandwidth of 66.67MHz. However implementation can be challenging, since currently reported SAW filters can only operate up to 10GHz in frequency [32] with bandwidths of up to 1.2GHz [34].

### 4.3.2 Monolithic Implementation

Since implementation of the real-time Spectral Encoded UWB transmitter with SAW filters is challenging, we also consider other methods, such as an integrated solution. An integrated solution has the advantage of being lower cost compared to using discrete off-chip SAW filters. In this integrated implementation, we assume that the Spectral Encoding spreading sequence is fixed for each user. Hence, the transmitter can be composed as shown in Figure 4.7, where a DSP pre-calculates the waveform that is then stored into a high speed memory. The memory feeds this waveform into the digital-to-analog converter (DAC), which in turn directly drives the antenna. Even though this type of system is no longer real-time, the same setup can be used.
First, the design of the DAC will be discussed. The two primary parameters are the conversion rate and the resolution. For the target system operating between 3GHz and 5GHz, the conversion rate requirement is at least 10GS/s. DACs running at these speeds are typically implemented with a current steering topology [34].

Next, the DAC resolution requirements are examined. From [36], assuming quantization noise is the primary source of noise, each bit of resolution provides roughly 6dB of signal-to-noise ratio (SNR). While a large number of bits provides a better representation of the signal, each additional bit also doubles the power consumption and doubles the memory size, which increases the overall power consumption of the system. Hence, to minimize power consumption, the goal is to choose the minimum number of bits that maintains the desired spectral characteristics. To determine the optimal number of bits, the spectral notch in the Spectral Encoded signal shown in Figure 4.1(c) is examined for various resolutions. Using system values obtained in the previous section, the notch
power is expected to be -20dB for a symbol length of $T_s = 35\text{ns}$ . Thus, to
determine the minimum resolution required, the signal shown in Figure 4.1(c) is
quantized, and the resulting spectral notch power is examined. As shown in
Figure 4.8, where the notch power is plotted as a function of the DAC resolution,
the minimum number of bits required to faithfully represent the signal is five,
which gives a theoretical SNR of 30dB.

![Notch power versus DAC resolution](image)

**Figure 4.8: Notch power versus DAC resolution. Also shown are the lower bounds from the DAC resolution SNR and the boundary for $T_s=35\text{ns}$.

Since DSPs running at 10GHz are challenging to implement, a high speed
memory driving the DAC will be considered, which has the advantage of using a
slower DSP to pre-compute the Spectral Encoded waveform, and can lead to a
lower power implementation. The memory storage requirement is determined by
the symbol length $T_s$, the conversion rate and the resolution, and is the product of all three terms. Even though the parameter values obtained previously were optimized to maintain the lowest notch power for the smallest SAW filter size, the same analysis can be used to obtain the smallest memory size. Using the values obtained previously, decreasing $T_s$ below 35ns, or reducing the resolution below 5-bits will reduce the notch performance. Thus, the choice of the memory size is limited by the performance requirements – the notch power. The relationship between the memory size and the notch power is summarized in Table 4.1, where the two parameters are varied at a fixed conversion rate of 10GS/s. Reducing the memory size, which could lead to easier implementation and lower power consumption, would require reducing $T_s$ or the DAC resolution, either of which would increase the notch power (from Figure 4.6 and Figure 4.8). However, it can be seen from Table 4.1 that choosing $T_s = 25$ns creates a -14dB notch, as does the 4-bit resolution. Therefore, an optimal choice in this case would be to choose both $T_s = 25$ns and 4-bits of resolution would yield a memory size of 1000 bits with a notch power of -14dB. However, to create a notch with -20dB, as shown in Table 4.1, would require $T_s = 35$ns and 5-bit resolution for a total memory size of 1750 bits.

From a practical standpoint, the integrated implementation seems to be more realizable with currently reported technology, as a 6-bit DAC using current-steering topology has been reported running at 22GS/s [36]. However, high speed
memory of size 1750 bits running at 10GHz are challenging to design, as the currently reported read speed for a 1024-bit memory is roughly 3GHz [37].

Table 4.1: The memory size and notch power relationship for various Ts and resolutions, assuming the conversion rate is fixed at 10Gs/s.

<table>
<thead>
<tr>
<th>$T_s$ (ns)</th>
<th>Resolution (bits)</th>
<th>Memory Size (bits)</th>
<th>Notch Power (db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>4</td>
<td>1000</td>
<td>-14</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>1050</td>
<td>-12</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>1250</td>
<td>-14</td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>1400</td>
<td>-14</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>1500</td>
<td>-18</td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>1750</td>
<td>-20</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>2000</td>
<td>-20</td>
</tr>
<tr>
<td>35</td>
<td>6</td>
<td>2100</td>
<td>-20</td>
</tr>
</tbody>
</table>

4.4 Summary

Two possible implementations of a Spectral Encoded UWB transmitter are discussed. The first is a real-time, fully analog implementation using off-chip SAW filters to perform the Fourier chirp transforms. The advantage of this system is its real-time performance. However, the primary challenge for this system is the design of the SAW chirp filters that have a large time-bandwidth product. The second implementation is an integrated one, where the Spectral Encoded waveform is pre-calculated, and stored in a high speed memory. This high speed memory drives the DAC, which may be directly connected to an antenna. This setup has the advantage of being low cost due to the integration of all components, and low power because there isn’t a high power consuming block between the DAC and antenna, such as a power amplifier. While high speed
DACs meeting the requirements of a Spectral Encoded transmitter have been reported, high speed memory of size 1750 bits running at 10GHz are challenging to build, and is a possible area for future research.

This chapter, in part is a reprint of the material as it appears in “Hardware Considerations for Spectral Encoded UWB Transmitters,” 2006 IEEE International Conference on Ultra-wideband, Sept. 2006. The authors are Joe Jamp and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.
5 Design of a 10GS/s 5-bit Digital-to-Analog Converter for UWB Spectral Encoded Transmitters

In Chapter 4, the system parameters of a Spectral Encoded transmitter are discussed. One of the systems described is the monolithic system, where all the components are integrated on chip to reduce cost and power consumption. This system includes a DSP to pre-calculate the Spectral Encoded waveform, a high speed memory in which the waveform is stored and subsequently fed to the DAC, and the DAC which is directly connected to an antenna. To test the performance of a Spectral Encoded system, the DAC will be designed as a first step towards the completed system. Devices, such as FPGAs, can be used to simulate the memory. In this chapter, the design, implementation, and experimental results of the DAC are presented and discussed.
5.1 Design of the DAC

From the results from Chapter 4, the required conversion rate for a system operating in the frequency band from 3-5GHz is at least 10GS/s. For a 20dB in-band notch at a symbol duration of $T_s = 35\text{ns}$, the required quantization resolution is five bits to maintain spectral notch characteristics. Thus, a 10GS/s 5-bit DAC will be designed.

A differential current steering architecture is used for this DAC, because this architecture is conducive to running at these high speeds [34]. The switching core for each bit is composed of bipolar transistors. The parasitic capacitance at the collectors of the bipolar transistors can limit the switching speed of the DAC, so minimum sized bipolar transistors are used for the LSB to minimize this capacitance. The DC bias current mirrors are implemented using MOSFETs to facilitate low Vdd operation for low power consumption. A circuit diagram of the DAC is shown in Figure 5.2.
Next, the choice of using a segmented (equally weighted) or binary weighted switches will be discussed. In a binary weighted setup, bit $N$ is $2^{N-1}$ larger than the LSB (least significant bit). Thus, the corresponding switches are also $2^{N-1}$ times larger than the LSB, and also consume $2^{N-1}$ times more current. An advantage of the binary weighted setup is its simplicity, since there is a direct mapping between the binary bit values stored in the memory and the represented voltage level. However, it is prone to large most significant bit (MSB) switching glitches, such as when the binary value changes from “011” to “100”, as shown in Figure 5.3 where circle (1) denotes the glitch for the LSB switching, and circle (2) denotes the glitch for a switch 4 times the LSB size. The glitch amplitude can exceed one LSB at higher resolutions.

A fully segmented setup, where all switches are of the same size and equal to one LSB, can reduce the MSB glitch energy. As an example, for a 3-bit DAC there are a total of eight switches all sized the same as the LSB. Creating one of

\[ \text{Figure 5.2: Circuit diagram of the 5-bit DAC. The three LSBs are binary weighted, and the two MSBs are equally weighted.} \]
the levels involves switching the corresponding number of cores, and as a result switching from level “011” to “100” only involves switching one more core, which results in smaller glitches. However, this type of setup requires the addition of a bit-converter, which converts the binary values from the DSP or memory into the thermometer code used in the segmented setup.

As a compromise between ease of implementation and minimizing glitches, a hybrid setup is used, where the two MSBs are converted into three segmented bits, and the three LSBs are binary weighted, as shown in Figure 5.2.

![Figure 5.2: Glitches due to switching in a DAC. (1) is the LSB switch glitch, and (2) is the MSB switching glitch (which is 4 times the LSB).]

To reduce power consumption of the overall transmitter, the DAC may be directly connected to an antenna to reduce the need for a power amplifier, and the
antenna connections are modeled as a 50\(\Omega\) load at the collectors of the DAC switching cores. Furthermore, smaller values of the Vdd and DC current are desired to reduce the overall power consumption of the DAC. Thus, the DAC will be designed to operate between Vdd=1.6-1.8V, with a total current consumption of 3.2mA, including the current mirror biasing circuit. The expected swing at the outputs in a 50\(\Omega\) load is 160mV peak-to-peak per leg with a total power dissipation of 5.74mW. From simulations, the instantaneous peak power to average power ratio is approximately 16.5dB. Therefore, \(V_{\text{rms}}=7.1\text{mV}\), and under the 100 differential load of the DAC, this corresponds to an average power consumption of 0.5\(\mu\)W (-32dBm), which is compatible with the FCC limit of \(-41.25\text{dBm/MHz}\) for a UWB signal with bandwidth of 500MHz.

To facilitate the testing of the DAC at high speeds, test circuitry is integrated on chip, and the overall block diagram is shown in Figure 5.4. The chip consists of five blocks: a 5-bit counter, a 5-bit to 6-bit converter, a flip-flop signal synchronization block, a buffer, and the DAC. First, the counter will be described.
Figure 5.5 shows a logic block diagram for the high-speed 5-bit counter, which counts from zero to 31 sequentially before resetting to zero. For this counter, the speed limiting factor will be the carry path from the LSB to the MSB. As shown in Figure 5.5, the longest carry bit path is from R1 to R4, and this delay must be carefully designed in the circuit layout such that the counter can operate at the required 10GHz rate.
Figure 5.5: The block diagram of the 5-bit counter.

The next block is the bit-converter, which converts the 5-bit binary counter output into the six bits that will control the DAC switching cores (Figure 5.6). Because the two MSBs (R3 and R4) are implemented as three segmented bits, they are converted into a thermometer code (Th3-Th5) with the logic shown in Figure 5.6. The three LSBs are passed through buffers to insert a delay that is roughly equal to those experienced by the MSBs.
Figure 5.6: The logic block diagram for bit-converter. Bits R3 and R4 are converted into a thermometer code.

All of the gates are implemented using standard current mode logic (CML). Figure 5.7 shows the circuit schematic of an inverter, where the bipolar transistors are minimum sized, and the output loads are set to 155Ω to achieve approximately 160mVpp swing at each leg for a total bias current of 1mA. Next, the NAND and NOR gate schematics are shown in Figure 5.8 and Figure 5.9, respectively. The design of these two gates very similar, because the differences between the two are that all the positive and negative inputs and outputs are swapped: as an example, outp on the NAND gate becomes outm on the NOR gate. For these gates, the total current is 2mA, and the output is expected to swing 200mVpp per leg. Lastly, the XNOR gate schematic is shown in Figure 5.10, which is also known in analog circuits as the Gilbert cell. The XNOR gate consumes 2mA of current and also has a 200mVpp per leg.
Figure 5.7: Circuit schematic of an inverter, which can also be used as a buffer.

Figure 5.8: The circuit schematic of a NAND gate. All transistors are minimum sized.
Figure 5.9: The circuit schematic of a NOR gate. All transistors are minimum sized.

Due to the delay through the long carry path and the delay through the bit-converter, the counter outputs R0-R4 and the corresponding bit-converter outputs
Th0-Th5 may be valid at different times. The flip-flop block synchronizes all six signals, and consists of six D-flip-flops, and is clocked with the same input clock as the counter. Lastly, a buffer is inserted between the flip-flop block and the DAC to ensure fast switching. Figure 5.12 shows the simulated DAC output, and the swing is expected to go from -160mV to +160mV over 3.2ns at a conversion rate of 10GS/s.

Figure 5.11: Circuit schematic of the D-flip-flop, where D denotes the input and Q is the output.
Figure 5.12: Cadence simulated DAC output signal at a conversion rate of 10GS/s.

5.2 Experimental Results

The chip was implemented in Jazz Semiconductor’s 0.18μm SiGe BiCMOS process [40], which has an $f_T$ (unity gain frequency) of 155GHz. A micrograph of the chip is shown in Figure 5.13, whose dimensions are 1.0x1.5mm. The input to the chip is a sinusoidal differential clock signal at the desired conversion rate. The DAC outputs are measured with a 20GS/s sampling oscilloscope. Probe testing is used to obtain results, with test setup block diagram shown in Figure 5.14, where the arrows denote the probes. A photograph of the test bench is shown in Figure 5.15, which shows the use of a sinusoidal clock generator (not shown), the probe testing station, the DC power supplies and multimeters, and the 20GS/s oscilloscope.
Figure 5.13: DAC/Counter microphotograph.

Figure 5.14: Probe testing setup block diagram. The arrows show where the probes are used.
Figure 5.15: Tech bench setup. The input sinusoidal clock generator is off to the right (not shown).

Figure 5.16 shows the measured DAC output at the input clock frequency of 2GHz (which corresponds to a conversion rate of 2GS/s), and Figure 5.17 shows the output at 5GHz and 10GHz.
Figure 5.16: Measured DAC output at $f_{CLK}=2GHz$.

Figure 5.17: Measured DAC output at $f_{CLK}=5GHz$ and $f_{CLK}=10GHz$. 
Next, the linearity results of the DAC will be presented. There are two
primary metrics that are used to characterize a DAC’s linearity performance: the
integral non-linearity (INL), and the differential non-linearity (DNL). The INL
describes how the resulting signal deviates from the ideal one at each sample
point, and can be mathematically expressed as

\[
\text{INL} = \left( V_D - V_{\text{zero}} \right) / V_{\text{LSB}} - D
\]

(5.1)

where \( V_D \) is the voltage represented by level \( D \), \( V_{\text{zero}} \) is the voltage level
corresponding to level with all the inputs equal to zero, \( V_{\text{LSB}} \) is the ideal voltage
step between two levels, and \( D = 0, 1, \ldots, 2^N - 1 \). A graphical example of the INL
calculation is shown in Figure 5.18. The DNL describes how the levels between
each step vary from an ideal step size, also shown in Figure 5.18, and can be
expressed mathematically as

\[
\text{DNL} = \left( V_{D+1} - V_D \right) / V_{\text{LSB}} - 1
\]

(5.2)

where \( D = 0, 1, \ldots, 2^N - 2 \). To maintain linearity, both the INL and DNL must be
within \( \pm 1 \) LSB (or \( V_{\text{LSB}} \). As shown in Figure 5.19, and the DNL is less than
\( \pm 0.6\text{LSB} \) and the INL is within \( \pm 1 \) LSB.
Figure 5.18: Graphical example of INL and DNL calculation. The dotted line represents the ideal DAC output value at the sample times, and the solid line is the response of the DAC.

Figure 5.19: Dynamic INL and DNL for the DAC operating at $f_{CLK} = 5$GHz.
The DC bias current and Vdd are tuned at various conversion rates to obtain the lowest operating power while maintaining linear performance. At 5GHz, Vdd is set at 1.7V and the DAC consumes 10.2mW. At 10GHz, it consumes 12mW. Note that while this power consumption is roughly double of the designed one, it is necessary to increase the current to obtain maintain linear performance. To compare with other DACs reported in literature, a figure-of-merit (FOM) is defined as

\[ \text{FOM} = \frac{f_{\text{CLK}}}{2^N P_{\text{DC}}} \]  

(5.3)

where \( f_{\text{CLK}} \) is the frequency, \( N \) is the resolution of the DAC in bits, and \( P_{\text{DC}} \) is the DC power consumption. Figure 5.20 shows the FOM for this DAC, and it shows that this DAC has one of the best FOMs of any reported DAC in this operational frequency range and resolution.
This work

Figure 5.20: Figure of merit (FOM) of the DAC (marked by x) compared with previously reported DACs in this frequency range (marked by circles).

5.3 Summary

In this chapter, a 10GS/s 5-bit DAC designed for use with a Spectral Encoded UWB transmitter is described. The design choices in choosing transistor sizes, and architectures are described. The DAC uses a hybrid architecture, where the two MSBs are segmented, and the three LSBs are binary weighted. Experimental results are measured using a probe testing setup. It can operate at conversion rates of over 10GS/s. At 5GS/s, the DNL is within 0.5LSB, and the INL is within 0.8LSB. The DAC consumes 10.2mW, and has one of the best
FOM compared with currently reported DACs in the same resolution and operational frequency range.

This chapter, in part, is a reprint of material as it appears in “A 10GS/s 5-bit Ultra-Low Power DAC for Spectral Encoded Ultra-Wideband Transmitters,” 2007 IEEE RFIC Symposium, June 2007. The authors are Joe Jamp, Junxiong Deng, and Lawrence Larson. The dissertation author is the primary author and investigator of this paper.
6 Conclusion

Interference between UWB systems and existing narrowband systems is an important issue for both types of systems to coexist. Due to the large bandwidth in which a UWB system may operate, there is potential for a UWB system to interfere with many narrowband systems. Effective methods to mitigate the interference caused by UWB systems are being researched. In this dissertation, coding techniques to spectrally shape a UWB signal’s spectrum are presented, and they insert in-band notches into UWB IR and MB-OFDM signals’ spectrum. Also, a hardware-based implementation of another interference mitigation technique called Spectral Encoding is presented. In this chapter, the results from the research presented in this dissertation are summarized.

In designing a technique to spectrally shape UWB IR and MB-OFDM signals, the primary goals are to allow for arbitrarily shape the spectrum, and to do so in a way that reduces changes in the hardware. By adapting LABI to account for the modulation scheme used in the UWB system, notches of varying depths can be created. For LABI TO, to which spectrally shapes a UWB PPM signal, the timing intervals between pulses are varied, and notch depths of -10dB can be created for a notch bandwidth of 10MHz. If instead, the pulse polarities are varied instead of the time position of the pulse, LABI PI can create notch depths of -18dB.
The notch depth in a UWB IR system can be sensitive to timing jitter, as a rms jitter of 40ps can increase the notch power by 7dB. If it is kept to below 10ps, then the timing jitter effects can be ignored.

Improvements to these algorithms can be used to provide the same spectral shaping performance at lower computational complexity. By reducing the computational complexity, hardware implementation can be made easier. LABI already uses the Viterbi algorithm to reduce the complexity to determine the appropriate flag bits to reduce the notch power. A further reduction in computational complexity is achieved by examining the computations required in the Viterbi algorithm, and by using a Gray Code scheme, complexity can be reduced by roughly the complementary filter block span $N_f$, which is a significant reduction. Another reduction involves looking at the UWB IR signal, and modulating both the timing intervals and the pulse polarities at the same time. The cascaded techniques LABI PITO and TOPI can be used to reduce the complexity by implementing less complex Viterbi algorithms without reducing performance.

LABI can be extended to spectrally shape MB-OFDM signals. In-band notches of -28dB can be created, which is a 20dB improvement over an uncoded spectrum.

Next, a hardware implementation of a real-time Spectral Encoding system is described. A Spectral Encoded system operates in a manner that resembles a
CDMA system. The spectrum is multiplied by the spreading sequence, which spreads the resulting signal in time, and notches can be inserted by nulling out some of the chips in the spreading sequence. For this system, two hardware implementations are described: a SAW based implementation, and a monolithic implementation. The former has the advantage of being real-time, but can be challenging to implement high time-bandwidth SAW filters, and can cost more due to the off-chip components. The integrated solution, which consists of a high speed memory driving a high speed DAC, can be lower cost compared to the SAW implementation.

A target system implemented with the monolithic solution is described, where the symbol length is 35ns, the memory size is 1750 bits, and the DAC is a 5-bit one operating at a conversion rate of 10GS/s.

To test the performance of the system, a high speed DAC is first implemented in Jazz Semiconductor’s 0.18μm SiGe BiCMOS process. The tradeoffs in designing the DAC is described, and a hybrid architecture is used to reduce glitch energy and maintain simplicity of implementation. The DAC output is measured by a sampling oscilloscope at various conversion rates, and it is shown that the DAC can operate at conversion rates of greater than 10GS/s. Linearity results are shown at 5GS/s, and the DNL and INL are within ±0.5LSB and ±1LSB, respectively. The DAC consumes 10.2mW of power, and has one of the best power consumption compared to currently reported DACs with similar resolution and conversion rates.
References


