A Coding Technique for the Spectral Shaping of Ultra-Wideband Time-Hopping Signals

Joe I. Jamp, *Student Member, IEEE*, and Lawrence E. Larson, *Fellow, IEEE*

***Abstract*—We introduce spectral-shaping coding techniques for interference mitigation, which is based on look-ahead block inversion applied to ultra-wideband time-hopped signals, by changing both the time offsets and the polarity of each pulse. Implementation complexity is reduced through the use of the Viterbi algorithm, and simulation results and design guidelines are presented. We calculate the effect of timing jitter on the performance and conclude that jitter effects are negligible for rms jitter values of less than 10 ps in most situations.**

***Index Terms*—Block codes, interference mitigation, pulse position modulation (PPM), spectral shaping, ultra-wideband (UWB).**

I. Introduction

An ULTRA-WIDEBAND (UWB) system is defined as a system with bandwidth that either exceeds 20% of the center frequency or has a 10-dB bandwidth of greater than 500 MHz [1]. These systems are allowed to operate in a frequency spectrum allocated by the FCC, which is from 3.1 to 10.6 GHz [1]. As a result, these systems are intended to overlay the existing narrowband services such as 802.11a and cordless telephones operating in the Unlicensed National Information Infrastructure bands at 5.2 and 5.8 GHz, as well as radar and other communication applications. By spectrally shaping the signals at the transmitter, UWB systems can create less interference to narrowband systems and, likewise, may not be adversely affected from the interference by these same systems. This reduction in the transmitted interference is in addition to the power limitation imposed by the FCC, which limits the spectral density of UWB systems to -41.25 dBm/MHz [1].

Several modulation techniques have been proposed for the UWB systems, such as pulse position modulation (PPM) [2], pulse amplitude modulation (PAM), binary phase-shift keying (BPSK), direct sequence spread spectrum [3], and orthogonal frequency division multiplexing [4]. With PPM, time-hopping (TH) sequences are used to avoid collisions in multiple access environments. PPM systems typically use the very narrow "Gaussian" pulses [2].

In this paper, we consider the spectral-shaping approaches for the two UWB modulation techniques: TH-PPM [2] and TH-BPSK. Both of these modulation techniques create a transmitted signal spectrum that is very broad, but it is difficult to filter the resulting waveform over a narrow frequency band to minimize interference. This is due to the fact that practical filters in the microwaveband are both highly dispersive and lossy. Several techniques to shape the Gaussian pulse spectrum away from sensitive bands have been proposed, including a duo-binary pulse [5] and a Manchester monopulse [6]. Both of these techniques are sensitive to small variations in the shape of the pulse and so have limited practical value. Pulse shapers designed for the FCC spectral mask have also been proposed [7], [8]. The pulse shaper in [7] requires a very high-speed digital-to-analog converter, limiting its practicality. The pulse shaper in [8] is very sensitive to timing jitter, and its spectral-shaping performance is similar to that of a digital finite-impulse response filter, so it is difficult or expensive to use it to shape the signal's spectrum away from the narrowband interference due to the large bandwidth of UWB signals. In addition, an approach using wavelet decomposition of a pulse is described in [9] but requires a complex pulse generation circuitry for each wavelet subcarrier, which may also limit its practicality.

Fig. 1. Binary block code that is showing a block of N_{b} bits with the flag bit F denoting the polarity of the associated block of bits.

We propose to spectrally shape the PPM waveform away from sensitive bands using a variation on the look-ahead block inversion algorithm (LABI) [10] by changing the time interval between pulses and/or the polarity of the pulses. In Section II, we will review LABI and discuss some results and improvements to the algorithm. Section III will describe the variations of LABI as applied to TH-BPSK and TH-PPM systems, and some results will be discussed. The effect of timing jitter on LABI for PPM performance will be discussed in Section IV.

II. Look-Ahead Block Inversion

LABI is a binary code where blocks of digital data are selectively inverted in order to shape the resulting frequency spectrum; as shown in Fig. 1, a trailing flag bit designates whether or not the block is inverted [10]. The algorithm is shown in Fig. 2. Given a desired shape of the spectrum $S_{\mathrm{desired}}(\omega)$ (see Fig. 3), the complementary filter is defined [10] as

$$H_{\mathrm{c}}(\omega) = \sqrt{S_{\mathrm{m}} - S_{\mathrm{desired}}(\omega)} \tag{1}$$

Manuscript received March 16, 2005; revised January 25, 2006 and June 7, 2006. This work was supported in part by the UCSD Center for Wireless Communications and its Member Companies and in part by the UC Discovery Grant of the State of California. The review of this paper was coordinated by Prof. R. Qiu.

The authors are with the Electrical and Computer Engineering Department, University of California at San Diego, La Jolla, CA 92037 USA (e-mail: jjamp@ece.ucsd.edu; larson@ece.ucsd.edu).

Digital Object Identifier 10.1109/TVT.2007.897230

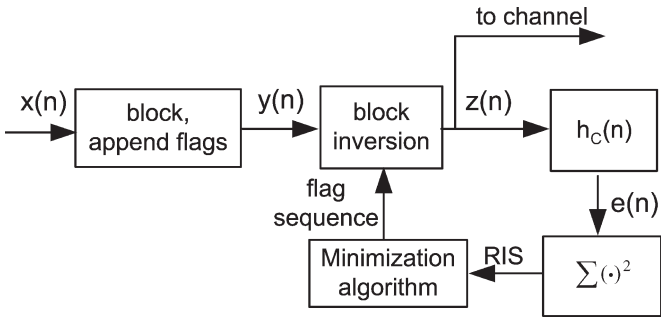

Fig. 2. LABI block diagram at the transmitter.

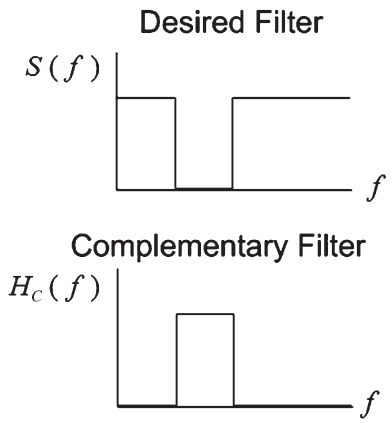

Fig. 3. Desired shape of the spectrum and the associated complementary filter.

where S_{m} is the maximum value of $S_{\mathrm{desired}}(\omega)$. We define $h_{\mathrm{c}}(n)$ as the inverse Fourier transform of $H_{\mathrm{c}}(\omega)$, which spans $N_{\mathrm{f}} = 2L$ blocks of N_{b} bits per block for a total $N_{\mathrm{c}} = N_{\mathrm{f}}N_{\mathrm{b}}$ bits. In general, $h_{\mathrm{c}}(n)$ is noncausal and centered around $n = 0$. Next, we define the flag padded sequence as $d_{\mathrm{B}}(n)$, which is given by

$$d_{\mathrm{B}}(iN_{\mathrm{b}}+n) = \begin{cases} F(i)d(iN_{\mathrm{b}}+n), & n = 0,1,\ldots,N_{\mathrm{b}}-2 \\ F(i), & n = N_{\mathrm{b}}-1 \end{cases} \tag{2}$$

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

$$e(iN_{\mathrm{b}}+j) = \sum_{k=-L}^{L-1}\sum_{m=0}^{N_{\mathrm{b}}-1} d_{\mathrm{B}}\left((i-k)N_{\mathrm{b}}+m\right)h_{\mathrm{c}}(kN_{\mathrm{b}}+j-m) \tag{3}$$

where i and k are the block indexes, and j and m are the indexes 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

$$\mathrm{RIS}(i) = \sum_{j=0}^{N_{\mathrm{b}}-1} e^2(iN_{\mathrm{b}}+j). \tag{4}$$

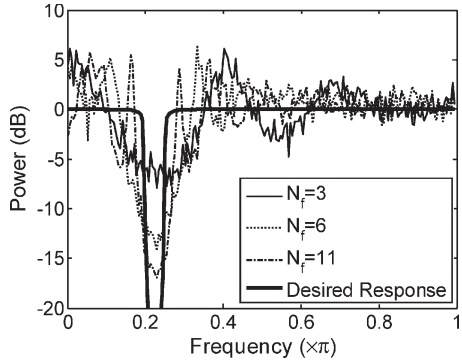

Fig. 4. LABI spectrum for various complementary filter block spans. $N_{\mathrm{b}} = 3$. Longer filter block spans improve performance.

The total RIS up to block r is given by

$$J(r) = \sum_{i=-\infty}^{r} \mathrm{RIS}(i). \tag{5}$$

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 $S_{\mathrm{desired}}(\omega)$ if the flag bits are chosen to minimize the RIS.

Since each flag bit will affect the 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 [11] to determine the flag-bit polarities. The state vector of the Viterbi trellis is defined as

$$\mathbf{sv}(i) = [F(i-L+1),\ldots,F(i),\ldots,F(i+L-1)] \tag{6}$$

with the state transition specifying the final flag bit. The RIS is used as the branch metric. Since we are using the Viterbi algorithm, one might want to map this trellis into a finite state machine, but doing so can be difficult. For each trellis state, the two incoming branches' J values are compared, and the path with the lower J is chosen as the survivor. Since the state transition specifying the final flag bit does not by itself determine the next state—rather the next state is choosing between the two possible histories—the trellis is the clearest representation, and that is what we will use for this paper.

Due to the complexity of the algorithm, closed-form analytical results for the resulting spectrum are difficult to develop. We rely on simulation for results. Figs. 4 and 5 show examples of the spectral shaping that can be performed with LABI, and it is shown 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.

From a practical standpoint, there appears to be an infinite delay through LABI, since minimizing $J(\infty)$ implies the use of

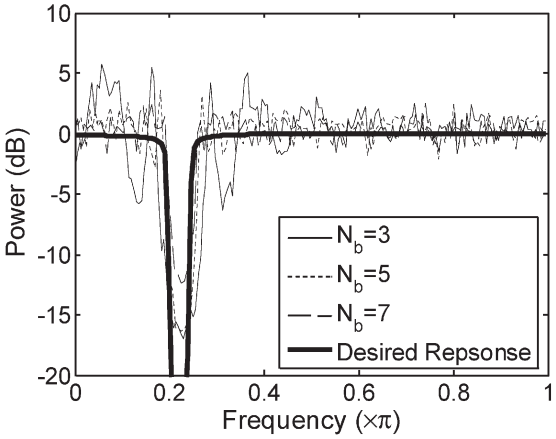

Fig. 5. LABI spectrum for various block lengths. $N_{\rm f} = 11$. Shorter block lengths improve performance.

LABI across an infinite sequence. With the Viterbi algorithm, it has been found that 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 (6) or $5N_{\rm f}$ blocks of bits for a total of $5N_{\rm f}N_{\rm b}$ 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 the 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, so any error correcting codes would detect a large number of incorrect 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 blocks of bits. Then, the algorithm will invert the selected blocks to achieve the desired spectral shape. With this case, we will examine the asynchronous cross-correlation between a desired user l and another user m, where both user's 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 $N_{\rm b}$. 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 Fig. 6, 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

$$R_{l,m}(i,j,a) = \sum_{n=0}^{N_{\rm b}-1-a} d_i^{(l)}(n) d_j^{(m)}(n+a) + \sum_{n=N_{\rm b}-a}^{N_{\rm b}-1} d_i^{(l)}(n) d_{j+1}^{(m)}(n-N_{\rm b}+a) \quad (7)$$

where $d_i^{(l)}(n)$ is the data bit at block i for user l and is defined from (2) as

$$d_i^{(l)}(n) = \begin{cases} F^{(l)}(i) d^{(l)}(iN_{\rm b}+n), & n = 0,1,\ldots,N_{\rm b}-2 \\ F^{(l)}(i), & n = N_{\rm b}-1 \end{cases}. \quad (8)$$

We expand the first term in (7) and obtain

$$F^{(m)}(j) d_i^{(l)}(N_{\rm b}-1-a) + \sum_{n=0}^{N_{\rm b}-2-a} d_i^{(l)}(n) d_j^{(m)}(n+a). \quad (9)$$

The first term in (9) is a direct contribution from the flag bit $F^{(m)}(j)$, and the second term is a partial cross-correlation. The second term in (7) can similarly be expanded, and we can obtain the contribution to the cross-correlation from $F^{(l)}(i)$. Both of these terms can add to the cross-correlation. As shown in Fig. 7, 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.

The second case is when the period of the m-sequence is larger than the block length. In this case, on applying LABI to the m-sequence, it is divided into blocks, with a flag bit inserted for each block. Fig. 8 shows a Monte Carlo simulation for the randomly chosen Kasami sequences of period 1023. The cross-correlation can increase to 16% of the period for low values of $N_{\rm b}$, 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, as we will see in Section II, with longer block lengths, LABI may not be able to achieve the desired spectral shaping.

A. Hardware Optimization of LABI

The performance of LABI is limited by the RIS calculation [10]. 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 (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 (3) as

$$e(iN_{\rm b}+j) = \sum_{k=-L}^{L-1} e_{\rm b}(i-k,j) \quad (10)$$

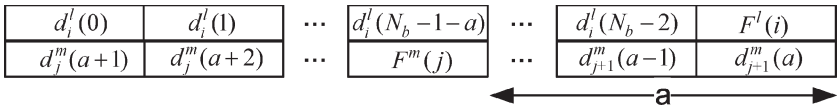

Fig. 6. Asynchronous cross-correlation diagram showing how the bits in block i of user l and blocks j and $j+1$ of user m line up, where the flag bits are offset by a bits.

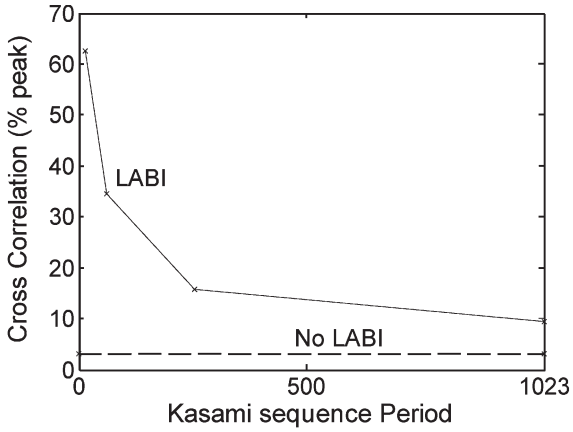

Fig. 7. Maximum cross-correlation values, which are normalized to the Kasami-sequence period, when the period of the sequence fits into a LABI block of length N_b. The flag bits were chosen to achieve the worst-case cross-correlation.

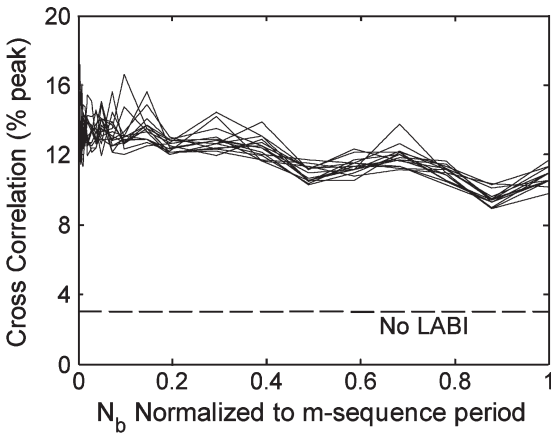

Fig. 8. Monte Carlo simulation of cross-correlation values for the randomly selected Kasami sequences, which are normalized to the period of the sequence. The flag-bit insertions cause an increase in the cross-correlation.

where $e_\mathrm{b}(i-k,j)$ is the contribution from block $i-k$ and its corresponding flag bit $F(i-k)$, and is defined as

$$e_\mathrm{b}(i-k,j)=\sum_{m=0}^{N_\mathrm{b}-1} d_\mathrm{B}\left((i-k)N_\mathrm{b}+m\right)h_\mathrm{c}(kN_\mathrm{b}+j-m). \tag{11}$$

From (10) and (11), each filter output calculation requires N_f flag bits, $N_\mathrm{f}-1$ of which are specified by the Viterbi trellis state vector (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 combination of flag bits are represented in the branch metric calculations. Fig. 9 shows a trellis for $N_\mathrm{f}=4$.

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 Fig. 9. 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 (Fig. 10). For the block with the differing flag bit, it can be seen from (10) that they have the same magnitude but the 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.

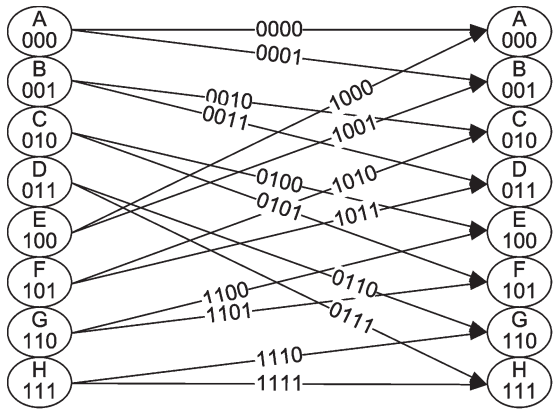

Fig. 9. Trellis for $N_\mathrm{f}=4$, with the state transition branches showing which flag bits are used to calculate the RIS (branch metric). "0" and "1" represent bit values of $+1$ and -1.

Fig. 10. 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. This way, we can guarantee 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,

we shall use a metric defined in [10], which is called the "computational load per data bit." 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, which is normalized by the number of data bits per block. It is defined as

$$C = \frac{C_{\text{filterout}} C_{\text{RIS}}}{\text{data bits per block}} \times \frac{\#\text{trellis state transitions per stage}}{2} \quad (12)$$

where $C_{\text{filterout}}$ is the number of multiply–accumulate operations performed to calculate the filter output, and C_{RIS} is the number of multiply–accumulate operations required to evaluate (4). The one-half comes from the quadratic nature of the RIS in (4), since the RIS value is the same even if all the flag bits are inverted.

From [10], the most computationally complex method is the explicit convolution, and its computational load per data bit is given by

$$C = \frac{N_{\text{f}} N_{\text{b}}^2}{N_{\text{b}} - 1} 2^{N_{\text{f}}-1}. \quad (13)$$

For the Gray Code optimized method, since given that one of the states has been calculated, each RIS calculation essentially involves calculating (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

$$C = \frac{(N_{\text{b}}+1)N_{\text{b}}}{N_{\text{b}} - 1}\left(2^{N_{\text{f}}-1} - 1\right) + \frac{N_{\text{f}} N_{\text{b}}^2}{N_{\text{b}} - 1}. \quad (14)$$

The second term in (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 (13), (14) is roughly lower by a factor of N_{f}, which is a significant improvement.

The authors of [10] also describe a table lookup method which stores the magnitude of $e_{\text{b}}(i-k, j)$ for each possible block $i-k$ into a table. With this method, to calculate the filter output in (10), we simply perform a table lookup for block $i-k$, then add or subtract the result depending on the flag bit $F(i-k)$. If we assume a table lookup and an addition/subtraction operation is equivalent to a multiply–accumulate operation, (10) requires $C_{\text{filterout}} = N_{\text{f}}$ table lookups, and the resulting computational load per input data bit is

$$C = \frac{N_{\text{f}} N_{\text{b}}}{N_{\text{b}} - 1} 2^{N_{\text{f}}-1}. \quad (15)$$

If we do a similar table lookup scheme with our Gray Code optimized version, as opposed to calculating (11) explicitly for each RIS calculation, we use a table lookup for the block's contribution. This would involve one table lookup and one multiply–accumulate operation, and the computational load per data bit is

$$C = \frac{2N_{\text{b}}}{N_{\text{b}} - 1}\left(2^{N_{\text{f}}-1} - 1\right) + \frac{N_{\text{f}} N_{\text{b}}}{N_{\text{b}} - 1}. \quad (16)$$

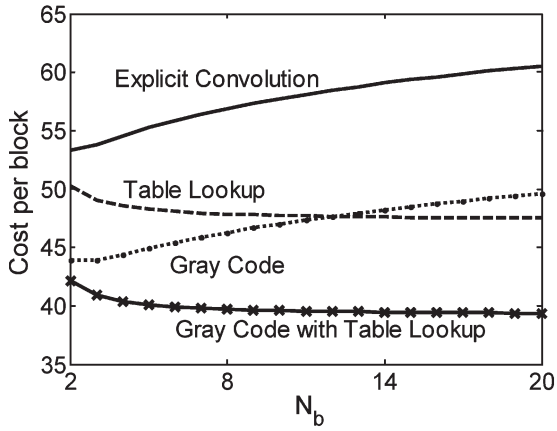

Fig. 11. Complexity cost functions plotted versus block length for $N_{\text{f}} = 11$.

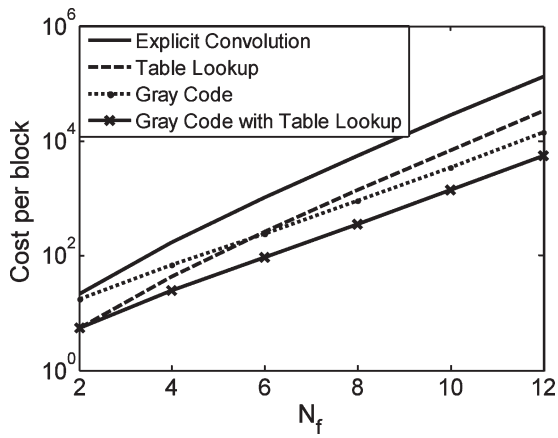

Fig. 12. Complexity cost functions plotted versus filter block span for $N_{\text{b}} = 4$.

A plot of these four "cost functions" is shown in Figs. 11 and 12, and they show that the optimized versions of LABI have roughly N_{f} lower computational complexity than the original LABI algorithm, except when $N_{\text{f}} = 2$ where the computational complexities are roughly the same as the nonoptimized versions. When $N_{\text{b}} < N_{\text{f}}$, both optimized methods have lower complexities than even the table lookup method mentioned in [10], although 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 that 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.

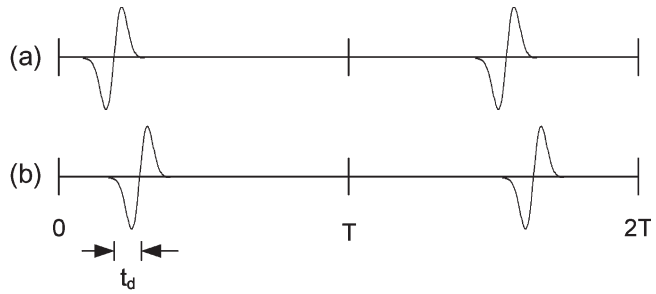

Fig. 13. UWB PPM signal. The pulse's position is determined by the data bit for (a) "0" bit and (b) "1" bit.

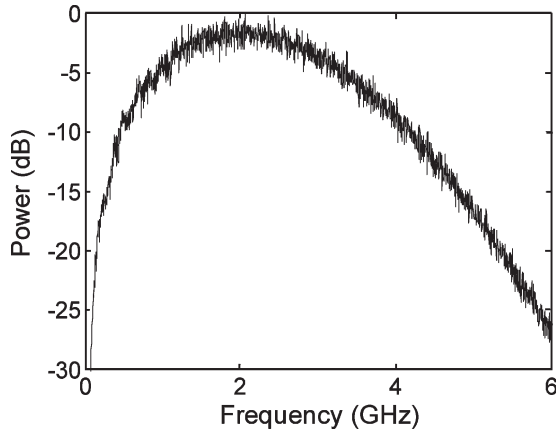

Fig. 14. UWB PPM signal spectrum with a Gaussian monopulse with a center frequency of 2 GHz.

III. LABI FOR UWB TH-PPM AND TH-BPSK

A. UWB Time-Hopped Signals

A UWB TH-PPM signal (Fig. 13) can be written as [2]

$$s(t) = \sum_n p\left(t - nT - t_{\rm pn}(n) - t_{\rm d} d(n)\right) \tag{17}$$

where T is the frame length, $t_{\rm pn}(n)$ is the pseudorandom time offset for pulse n, $t_{\rm d}$ is the time offset depending on the bit specified by the data bits $d(n)$. We model the pulse $p(t)$ as a Gaussian monopulse (Fig. 14) [2], but, in general, it can be of any arbitrary shape, including those specified in [5]–[8]. For this paper, we normalize the pulse power to unity. Depending on the symbol rate of the UWB system, each bit can modulate multiple pulses. For simplicity purposes, we use one pulse per data bit.

While TH-PPM changes the time intervals between pulses, all the pulse polarities are the same. As an alternative, data can be transmitted by modulating the polarity of the pulse, i.e., binary PAM. The TH term $t_{\rm pn}(n)$ is retained for the pseudorandom aspect. The resulting signal, which is called TH-BPSK, can be written as

$$s(t) = \sum_n d(n) p\left(t - nT - t_{\rm pn}(n)\right). \tag{18}$$

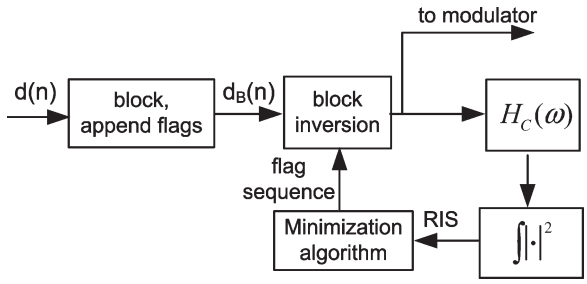

Fig. 15. LABI for UWB PPM block diagram at the transmitter.

B. LABI Time Offset

LABI depends on the pulses having positive and negative amplitudes as determined by the data bits. The TH-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 (Fig. 15). First, we rewrite (17) as

$$s(t) = \sum_i \sum_{n=0}^{N_{\rm b}-1} p\left(t - \tau_1(i,n)\right) \tag{19}$$

where $\tau_1(i,n) = (iN_{\rm b}+n)T - t_{\rm pn}(iN_{\rm b}+n) - t_{\rm d} d_{\rm B}(iN_{\rm b}+n)$, and $d_{\rm B}(\cdot)$ is described in (2). Next, we define $h_{\rm c}(t)$ as the inverse Fourier transform of $H_{\rm c}(\omega)$, and it spans $N_{\rm f} = 2L$ blocks of $N_{\rm b}$ bits per block for a total $N_{\rm c} = N_{\rm f}N_{\rm b}$ bits. Hence, the filter has an impulse response duration of $N_{\rm c}T$, where T is the bit rate. In general, $h_{\rm c}(t)$ is noncausal, and we center it around $t = 0$. Then, we filter the signal $s(t)$ defined in (19) through $h_{\rm c}(t)$, and we redefine the RIS for block i as

$$\begin{aligned}{\rm RIS}(i) &= \int_{-\infty}^{\infty} \left| s(t) * h_{\rm c}(t) \right|^2 dt \\ &= \int_{-\infty}^{\infty} \left| \left[\sum_{k=-L}^{L-1} \sum_{n=0}^{N_{\rm b}-1} p\left(t - \tau_1(i-k,n)\right) \right] * h_{\rm c}(t) \right|^2 dt \end{aligned} \tag{20}$$

where $*$ denotes the convolution operation. Using Parseval's theorem, we can equivalently write (20) as

$$\begin{aligned}{\rm RIS}(i) = \int_{-\infty}^{\infty} &\left|P(\omega)\right|^2 \left|H_{\rm c}(\omega)\right|^2 \\ &\times \left| \sum_{k=-L}^{L-1} \sum_{n=0}^{N_{\rm b}-1} e^{-j\omega\tau_1(i-k,n)} \right|^2 d\omega \end{aligned} \tag{21}$$

where $P(\omega)$ is the Fourier transform of $p(t)$. We can obtain a simpler expression for (21) if we assume that within a band of interest, the spectrum of the pulse $P(\omega)$ and the filter $H(\omega)$

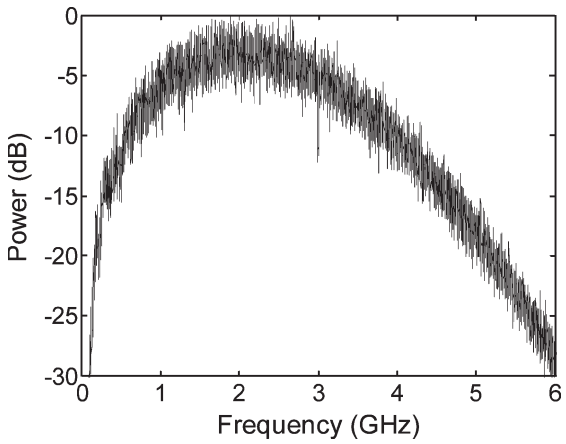

Fig. 16. LABI TO spectrum with a 10-MHz wide notch at 3 GHz.

are relatively constant. Then, for a single notch, (21) can be simplified as

$$\mathrm{RIS}(i)=\sum_{k=-L}^{L-1}\sum_{n=0}^{N_\mathrm{b}-1}\sum_{q=-L}^{L-1}\sum_{p=0}^{N_\mathrm{b}-1}\frac{\sin\omega_\mathrm{U}\left(\tau_1(i-k,n)-\tau_1(i-q,p)\right)}{\tau_1(i-k,n)-\tau_1(i-q,p)}$$

$$-\frac{\sin\omega_\mathrm{L}\left(\tau_1(i-k,n)-\tau_1(i-q,p)\right)}{\tau_1(i-k,n)-\tau_1(i-q,p)} \tag{22}$$

where ω_U and ω_L are the upper and lower frequencies of the band of interest. If there are multiple bands of interest, we would calculate (22) for each band and sum the results to determine RIS at block i. With this definition of RIS, as shown in Fig. 16, spectral shaping can be achieved for TH-PPM signals using this LABI variant, which we call LABI TO.

To obtain some intuition on how LABI TO performs spectral shaping, let us consider a simple case with a notch at frequency ω_0 and control over N data bits. The Fourier transform of (17) can be written as

$$S(\omega)=P(\omega)\sum_{n=0}^{N-1}e^{-j\omega(nT+t_\mathrm{pn}(n))}e^{-j\omega t_\mathrm{d}d(n)}. \tag{23}$$

For small values of ωt_d, we can rewrite (23) as

$$S(\omega)\approx P(\omega)[V-U] \tag{24}$$

where

$$V=\sum_{n=0}^{N-1}e^{-j\omega(nT+t_\mathrm{pn}(n))} \tag{25}$$

$$U=j\omega t_\mathrm{d}\sum_{n=0}^{N-1}d(n)e^{-j\omega(nT+t_\mathrm{pn}(n))}. \tag{26}$$

With the desired notch at ω_0, the goal is to choose $d(n)$ so that $U\approx V$ such that $|S(\omega_0)|$ is minimized. The magnitude of V will depend on the exact TH sequence used, and for our analysis, we assume that $\omega_0(nT+t_\mathrm{pn}(n))$ is an independent identically distributed (i.i.d.) uniform random process from $[0,2\pi]$. We need to determine

$$E\left(|V|\right)=E\left(\left|\sum_{n=0}^{N-1}\cos\left(\omega(nT+t_\mathrm{pn}(n))\right)\right.\right.$$
$$\left.\left.-\sum_{n=0}^{N-1}j\sin\left(\omega(nT+t_\mathrm{pn}(n))\right)\right|\right) \tag{27}$$

where $E(\cdot)$ is the expectation operator. For sufficiently large 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

$$E\left(\sum_{n=0}^{N-1}\cos\left(\omega(nT+t_\mathrm{pn}(n))\right)\right)^2$$
$$=E\left(\sum_{n=0}^{N-1}\sin\left(\omega(nT+t_\mathrm{pn}(n))\right)\right)^2$$
$$=\frac{N}{2}. \tag{28}$$

Hence, $|V|$ is a Rayleigh random variable, whose mean is given by

$$E\left(|V|\right)=\sqrt{N\pi/4}. \tag{29}$$

In the limit of $N_\mathrm{b}=1$ (a block length equal to a single bit) and small ωt_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 $d'(n)$ 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

$$E\left(|U'|\right)=E\left(\left|\omega t_\mathrm{d}\sum_{n=0}^{N-1}e^{-j\omega(nT+t_\mathrm{pn}(n))}d'(n)\right|\right)$$
$$=\omega t_\mathrm{d}\frac{2}{\pi}N. \tag{30}$$

Hence, the normalized magnitude of $|S(\omega_0)|$ is given by

$$|S(\omega_0)_\mathrm{norm}|=\frac{E\left(|V-jU'|\right)}{E\left(|V|\right)}\approx 1-\omega t_\mathrm{d}\frac{4}{\sqrt{\pi^3}}\sqrt{N}. \tag{31}$$

For a more realistic situation, we consider the case when $N_\mathrm{b}>1$, and $N=N_\mathrm{f}N_\mathrm{b}$. Now, we can rewrite U as

$$U=\omega t_\mathrm{d}\sum_{i=0}^{N_\mathrm{f}-1}F(i)u_i \tag{32}$$

where

$$u_i=\sum_{m=0}^{N_\mathrm{b}-1}d(iN_\mathrm{b}+m)e^{-j\omega((iN_\mathrm{b}+m)T+t_\mathrm{pn}(iN_\mathrm{b}+m))}$$
$$=a_ie^{-j\theta_i}. \tag{33}$$

and

$$E(a_i) = E\left(\left|\sum_{m=0}^{N_\mathrm{b}-1} d(iN_\mathrm{b}+m)e^{-j\omega((iN_\mathrm{b}+m)T+t_\mathrm{pn}(iN_\mathrm{b}+m))}\right|\right) \tag{34}$$

and where $F(i)$ denotes the flag bit for block i, and $d(\cdot)$, in this case, is the flag-bit padded data sequence where $d((i+1)N_\mathrm{b}-1) = F(i)$. The quantity u_i can be treated as a single vector with magnitude a_i and phase θ_i as determined by the summation. If we assume $d(n)$ is an 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 aforementioned analysis for V, it can be shown that $E(a_i) = \sqrt{N_\mathrm{b}\pi/4}$. The goal, as aforementioned, is to invert all the vectors u_i that point in the V direction. If we assume that θ_i is uniform from $[0, 2\pi]$, then when we invert all the vectors that point in the V direction, we define θ_i' as uniform from $[\pi, 2\pi]$ (if V is rotated to $\pi/2$), and then obtain

$$\begin{aligned} E\left[|U|\right] &= \omega t_\mathrm{d} \sum_{i=0}^{N_\mathrm{f}-1} E[a_i \sin\theta_i] \\ &= \omega t_\mathrm{d} \frac{2}{\pi} \sum_{i=0}^{N_\mathrm{f}-1} E[a_i] \\ &= \omega t_\mathrm{d} \frac{2}{\pi} N_\mathrm{f} E[a_i] \\ &= \omega t_\mathrm{d} \frac{1}{\sqrt{\pi}} N_\mathrm{f} \sqrt{N_\mathrm{b}} \end{aligned} \tag{35}$$

and

$$\begin{aligned} |S(\omega_0)_\mathrm{norm}| &\approx \frac{\sqrt{\frac{N_\mathrm{f}N_\mathrm{b}\pi}{4}} - \omega t_\mathrm{d}\frac{1}{\sqrt{\pi}} N_\mathrm{f}\sqrt{N_\mathrm{b}}}{\sqrt{\frac{N_\mathrm{f}N_\mathrm{b}\pi}{4}}} \\ &\approx 1 - \omega t_\mathrm{d} \frac{2}{\pi}\sqrt{\frac{N}{N_\mathrm{b}}}. \end{aligned} \tag{36}$$

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 Fig. 17 quite well. Note that for better spectral shaping with LABI, we, generally, would use smaller block lengths to provide better spectral shaping, which is the same result for LABI TO. Even though we used the Gaussian approximation for large N_b, as Fig. 17 shows, the results match well for small block lengths.

There are some limitations to the aforementioned analysis. First, the aforementioned analysis will also only hold for very narrowband notches, as we consider a single frequency. 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 we would need to resort to a full search to determine

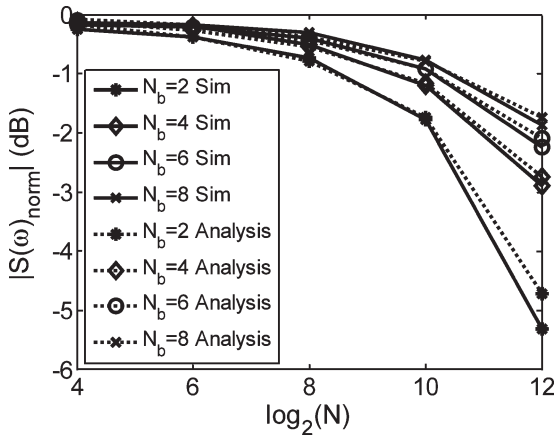

Fig. 17. $|S(\omega_0)_\mathrm{norm}|$ as a function of N and compares (dotted lines) the analysis versus (solid lines) the simulation results or various block lengths N_b. Smaller N_b for a given N minimizes $|S(\omega_0)|$ better.

which vectors (or data bits) would need to be inverted such that $|U| \approx |V|$ to minimize $|S(\omega_0)|$. Note that for LABI and LABI pulse inversion (PI), which is described next, since there are no U vectors, the choice becomes choosing the vectors (or data bits) to let V self-cancel. Lastly, the above analysis assumes that the complementary filter is of length $N_\mathrm{c} = N$ bits, and it assumes N is large. While the behavior of LABI TO is similar for small values of N, since it does choose the U vectors to point as best as possible in the $-V$ direction, the predicted values of the amplitudes no longer match the analysis, due to the use of the central limit theorem in the analysis.

C. LABI Pulse Inversion

For UWB TH-BPSK systems, we also need to make a modification to LABI, since LABI assumes that the time interval between each pulse is the same. Due to the TH nature, the time interval between pulses varies, so we integrate this aspect into the algorithm in a manner similar to LABI TO. Let us rewrite (18) as

$$s(t) = \sum_i \sum_{n=0}^{N_\mathrm{b}-1} d_\mathrm{B}(iN_\mathrm{b}+n)p\left(t-\tau_2(i,n)\right) \tag{37}$$

where $\tau_2(i,n) = (iN_\mathrm{b}+n)T - t_\mathrm{pn}(iN_\mathrm{b}+n)$ and is the same as $\tau_1(\cdot)$ without the data modulated time offset. Using a similar method as described previously for LABI time offset, we can write the RIS for block i as

$$\begin{aligned} \mathrm{RIS}(i) = \int_{-\infty}^{\infty} &|P(\omega)|^2 |H_\mathrm{c}(\omega)|^2 \\ &\cdot \left|\sum_{k=-L}^{L-1}\sum_{n=0}^{N_\mathrm{b}-1} d_\mathrm{B}\left((i-k)N_\mathrm{b}+n\right) e^{-j\omega\tau_2(i-k,n)}\right|^2 d\omega. \end{aligned} \tag{38}$$

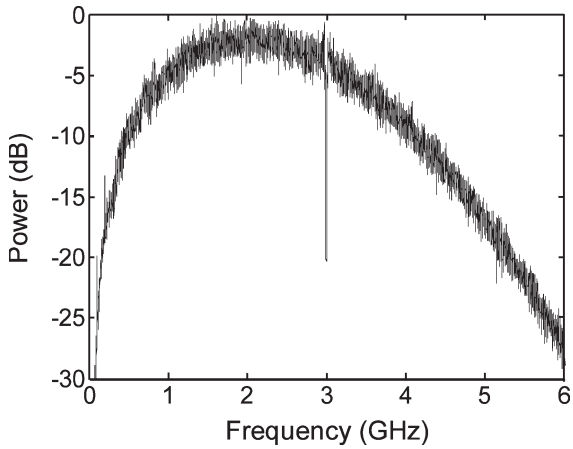

Fig. 18. UWB TH-BPSK signal spectrally shaped by the LABI PI.

Then, for a single notch, (38) can be simplified as

$$\begin{aligned}\mathrm{RIS}_s(i) &= \sum_{k=-L}^{L-1}\sum_{n=0}^{N_\mathrm{b}-1}\sum_{q=-L}^{L-1}\sum_{p=0}^{N_\mathrm{b}-1} d_\mathrm{B}\left((i-k)N_\mathrm{b}+n\right) \\ &\quad \times d_\mathrm{B}\left((i-q)N_\mathrm{b}+p\right) \\ &\quad \cdot \left(\frac{\sin\omega_\mathrm{U}\left(\tau_2(i-k,n)-\tau_2(i-q,p)\right)}{\tau_2(i-k,n)-\tau_2(i-q,p)}\right. \\ &\quad \left. - \frac{\sin\omega_\mathrm{L}\left(\tau_2(i-k,n)-\tau_2(i-q,p)\right)}{\tau_2(i-k,n)-\tau_2(i-q,p)}\right). \quad (39)\end{aligned}$$

With this definition of RIS, we can now achieve spectral shaping for TH-BPSK signals with this LABI variation. Fig. 18 shows a simulated spectrum shaped by LABI PI.

D. 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, we rely on simulations to obtain results. As shown in Figs. 16 and 18, one can achieve spectral shaping 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 tradeoffs are increased overhead (for smaller block lengths) and increased computational complexity (for longer complementary filter block spans). In addition, for a given block length and complementary filter block span, the narrower the notch is, the deeper it is, which we can see by comparing Figs. 18 and 19. Therefore, our algorithm is better suited for the notches of small fractional bandwidths. Lastly, there is a delay (trace-back depth) of approximately 5 times the length of the trellis state vector of the Viterbi algorithm and has been verified through simulation as roughly $5N_\mathrm{c}$ 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 [7] and [8], and the wavelet-based waveform in [9]. For example, using the pulse waveform described in [8] with this algorithm, one can both provide a signal spectrum

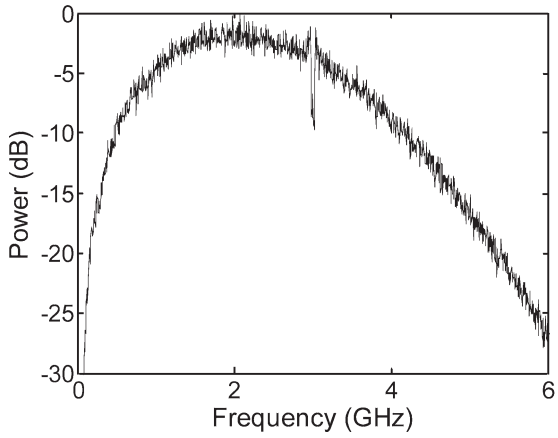

Fig. 19. LABI PI spectrum with a 50-MHz wide notch at 3 GHz.

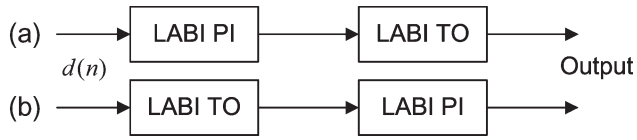

Fig. 20. Two ways of cascading LABI PI and LABI TO to allow for spectral shaping of UWB TH signals by changing the time offsets and the pulse polarities.

that both complies with the FCC UWB spectral mask and provides the necessary spectral notches for the various in-band narrowband systems. In addition, our algorithm is insensitive to multipath effects. If we assume a linear slowly varying multipath channel, the spectrum of the channel multiplies with the transmitted signal spectrum. The result is a spectrum with less interference in the narrowband of interest. In addition, since both the aforementioned algorithms assume that the TH sequence is provided, one can also use TH codes that are designed to produce spectral shaping [13]. Lastly, one might be concerned with the effects of LABI on the orthogonality of UWB TH sequences. Since we only change the data bits, 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 TH-PPM.

E. Improvements to LABI PI and LABI TO

Although both LABI PI and LABI TO can spectrally shape UWB TH signals, both methods only vary one aspect of the UWB TH 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 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 (Fig. 21).

First, we describe LABI TOPI, as shown in Fig. 20(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 appropriate blocks that are inverted in polarity. This determines the time offsets $\tau_1(n)$ in the output signal. Then, the resulting signal is passed to LABI PI to determine the optimum pulse

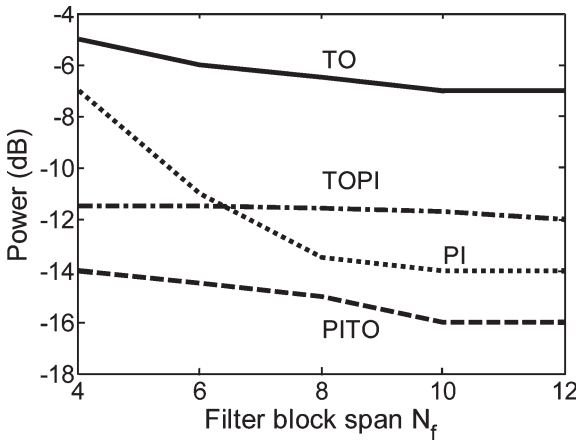

Fig. 21. Notch depth of the four LABI variants as complexity increases (filter block span increases). For TOPI and PITO, the second block in the cascade is fixed at $N_{\mathrm{f}} = 6$.

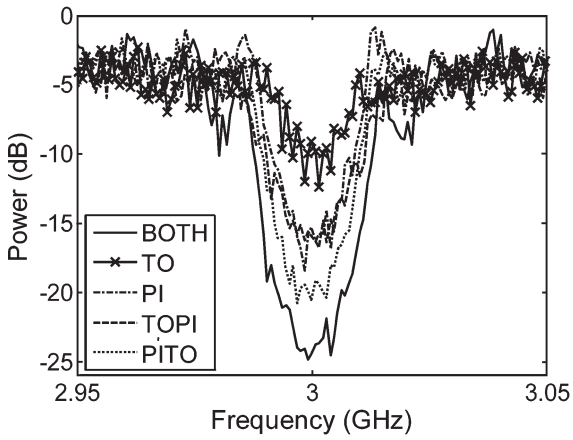

Fig. 22. Spectrum at the notch for the LABI variants compared with the optimal choice in both pulse polarity and time position.

polarities. The output signal can then be described as

$$s(t) = \sum_n a_{\mathrm{B}}(n) p\left(t - \tau_1(n)\right) \tag{40}$$

where $a_{\mathrm{B}}(n)$ can take on values $+1$ or -1. Similarly, we can write the output signal of the LABI PITO case as

$$s(t) = \sum_n d_{\mathrm{B}}(n) p\left(t - \tau_2(n) - a(n) t_{\mathrm{d}}\right) \tag{41}$$

where $a(n)$ can take on values $+1$ 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 paper, we assume the parameters for the second block in the cascade to be $N_{\mathrm{b}} = 2$ and $N_{\mathrm{f}} = 6$, and vary the parameters of the first block.

Fig. 21 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 noncascaded versions, particularly for small N_{f}. Note that LABI PI outperforms LABI TOPI for $N_{\mathrm{f}} > 6$, which is due to choosing $N_{\mathrm{f}} = 6$ for the PI block in the cascade. The crossover point would be at a larger value of N_{f} if we chose a larger N_{f} for the PI block in the LABI TOPI cascade, since LABI TOPI's performance would increase. From Fig. 21, to achieve the same spectral-shaping performance, one can either use a noncascaded 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 may be easier to implement since it requires $2^{N_{\mathrm{f,TO}}} + 2^{N_{\mathrm{f,PI}}}$ 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 $2^{N_{\mathrm{f}}}$ trellis states. As an example from Fig. 21, one can achieve a -14-dB 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.

While cascading improves the 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 Fig. 22). Fig. 22 compares the spectrum of LABI BOTH with all the LABI variants, and 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 $N_{\mathrm{f,PI}} + N_{\mathrm{f,TO}}$.

IV. Effects of Timing Jitter on LABI TO

Since the LABI variants for UWB TH systems rely on precise timing between 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

$$s(t) = \sum_n a_n p(t - \gamma_n - \delta_n) \tag{42}$$

$$S(\omega) = P(\omega) \sum_n a_n e^{-j\omega(\gamma_n + \delta_n)} \tag{43}$$

where δ_n is the jitter, which is modeled as a zero mean Gaussian random process with variance σ^2, and $\gamma_n = t_{\mathrm{pn}}(n) + t_{\mathrm{d}} d(n) + nT$. In a similar manner to Section III, 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

$$\begin{aligned} E_{\mathrm{jitter}}(\omega) &= |S(\omega)|^2 \\ &= \sum_m \sum_n a_n a_m \cos\omega(\gamma_m - \gamma_n) \cos\omega(\delta_m - \delta_n) \\ &\quad + a_n a_m \sin\omega(\gamma_m - \gamma_n) \sin\omega(\delta_m - \delta_n). \end{aligned} \tag{44}$$

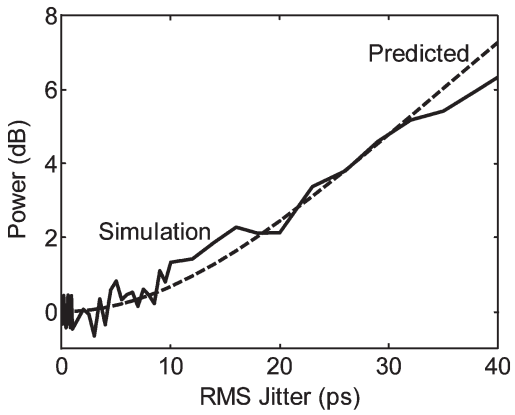

Fig. 23. Increase in notch power versus rms timing jitter.

From (44), we can write an expression for the average effect of jitter as

$$E\left[E_{\text{jitter}}(\omega)\right] = \sum_m \sum_n a_n a_m \cos\omega(\gamma_m - \gamma_n) E\left[\cos\omega(\delta_m - \delta_n)\right] + a_n a_m \sin\omega(\gamma_m - \gamma_n) E\left[\sin\omega(\delta_m - \delta_n)\right]. \quad (45)$$

It can be shown that $E[\cos\omega(\delta_m - \delta_n)] = e^{\omega^2 o^2}$ and $E[\sin\omega(\delta_m - \delta_n)] = 0$. Therefore, (45) becomes

$$E\left[E_{\text{jitter}}(\omega)\right] = \sum_n 1 + e^{\omega^2\sigma^2} \sum_{\substack{n,m \\ n\neq m}} \cos\omega(\gamma_m - \gamma_n). \quad (46)$$

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

$$E_{\text{nojitter}}(\omega) = \sum_n 1 + \sum_{\substack{n,m \\ n\neq m}} \cos\omega(\gamma_m - \gamma_n). \quad (47)$$

Next, we take the ratio of (46) to (47), and if we ignore the first term from both equations since they are both much smaller than the second term, we obtain

$$\frac{E\left[E_{\text{jitter}}(\omega)\right]}{E_{\text{nojitter}}(\omega)} \approx e^{\omega^2\sigma^2}. \quad (48)$$

We can equivalently write the ratio of the power with jitter to the power without jitter and rewrite (48) as

$$E\left[P_{\text{jitter}} d_{\text{B}}(\omega)\right] - P_{\text{nojitter}} d_{\text{B}}(\omega) \approx 10\omega^2\sigma^2 \log_{10}(e). \quad (49)$$

Equation (49) predicts that the effect of jitter is to "fill in" the notch and is a function of frequency and jitter variance. As shown in Fig. 23, the analysis matches the simulation results well, with the jitter standard deviations of 10 and 40 ps degrading the notch depth by about 1 and 7 dB, respectively. Hence, the algorithm is sensitive to timing jitter. However, UWB timing circuits with an rms jitter of less than 10 ps have been reported [14], so the performance degradation is negligible under these circumstances.

V. Conclusion

In this paper, we introduce LABI PI and LABI TO, which are algorithms that spectrally shape the UWB TH-BPSK and TH-PPM signals, respectively. To achieve spectral shaping, LABI PI varies the pulse polarities, while LABI TO varies the timing intervals between pulses. Both algorithms shape the UWB spectrum away from the in-band narrowband systems for the purposes of interference mitigation. For example, LABI PI can produce a 10-MHz wide notch with a depth of 18 dB at an arbitrary frequency. While it is not generally useful for spectrally shaping the UWB signal to 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—LABI PITO and LABI TOPI—can provide even better spectral-shaping performance without increasing data overhead, at a cost of increased complexity. One advantage with the cascaded systems is the use of two lower complexity Vitberi algorithm implementations to achieve the same notch depth, which is easier to implement. However, due to the TH nature of the signals examined in this paper, this algorithm is sensitive to timing jitter, with a degradation of 7 dB to the notch depth when the rms timing jitter is 40 ps, respectively.

References

[1] Federal Communications Commission, *Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems: First report and order*, Apr. 2002. ET-Docket 98-153.
[2] M. Z. Win and R. Scholtz, "Impulse radio: How it works," *IEEE Commun. Lett.*, vol. 2, no. 2, pp. 10–12, Jan. 1998.
[3] *UWB Forum*. [Online]. Available: http://www.uwbforum.org/
[4] *WiMedia Alliance—ECMA International Standards*. [Online]. Available: http://www.wimedia.org/en/resources/eis.asp
[5] K. Eshima, Y. Hase, S. Oomori, F. Takahashi, and R. Kohno, "Performance analysis of interference between UWB and SS signals," in *Proc. IEEE Symp. Spread Spectrum Tech. Appl.*, Sep. 2002, vol. 1, pp. 59–62.
[6] X. Chen and S. Kiaei, "Monocycle shapes for ultra-wideband system," in *Proc. ISCAS*, May 2002, vol. 1, pp. 597–600.
[7] B. Parr, B. Cho, K. Wallace, and Z. Ding, "A novel ultra-wideband pulse design algorithm," *IEEE Commun. Lett.*, vol. 7, no. 5, pp. 219–221, May 2003.
[8] X. Luo, L. Yang, and G. B. Giannakis, "Designing optimal pulse-shapers for ultra-wideband radios," *J. Commun. Netw.*, vol. 5, no. 4, pp. 344–353, Dec. 2003.
[9] K. Ohno, T. Ikebe, and T. Ikegami, "A proposal for an interference mitigation technique facilitating the coexistence of bi-phase UWB and other wideband systems," in *Proc. UWBST*, May 2004, pp. 50–54.
[10] J. K. Cavers and R. F. Marchetto, "A new coding technique for spectral shaping of data," *IEEE Trans. Commun.*, vol. 40, no. 9, pp. 1418–1422, Sep. 1992.
[11] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.
[12] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 2001.
[13] J. Bellorado, S. Ghassemzadeh, A. Kavcic, B. Tarokh, and V. Tarokh, "Time-hopping sequence design for narrowband interference suppression," in *Proc. VTC—Fall*, Sep. 2004, vol. 6, pp. 3925–3929.
[14] D. Rowe, B. Pollack, J. Pulver, W. Chon, P. Jett, L. Fullerton, and L. Larson, "A Si/SiGe HBT timing generator IC for high-bandwidth impulse radio applications," in *Proc. IEEE Cust. Integr. Circuits*, May 1999, pp. 221–224.
[15] R. Scholtz, "Multiple-access with time-hopping impulse modulation," in *Proc. MILCOM*, Oct. 1993, pp. 447–450.

Joe I. Jamp (S'03) received the B.S. degree in electrical engineering and computer science from the University of California at Berkeley, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at San Diego (UCSD), La Jolla, in 2003 and 2007, respectively.

He is currently with the Electrical and Computer Engineering Department, USCD.

Lawrence E. Larson (F'00) received the B.S. degree in electrical engineering and the M.Eng. degree from Cornell University, Ithaca, NY, in 1979 and 1980, respectively, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles, in 1986.

From 1980 to 1996, he was at Hughes Research Laboratories, Malibu, CA, where he directed the development of high-frequency microelectronics in GaAs, InP and Si/SiGe, and microelectromechanical-system (MEMS) technologies. He was with the faculty at the University of California at San Diego (UCSD), La Jolla, in 1996, where he is the inaugural holder of the Communications Industry Chair. During the 2000–2001 academic year, he was on leave at the IBM Research, San Diego, where he directed the development of radio frequency integrated circuits for 3G applications. From 2001 to 2006, he was the Director of the UCSD Center for Wireless Communications. During the 2004–2005 academic year, he was a Visiting Professor at TU Delft, Delft, The Netherlands. He has published over 200 papers, coauthored four books, and is the holder of 31 U.S. patents.

Dr. Larson was the recipient of the 1995 Hughes Electronics Sector Patent Award for his work on RF MEMs, corecipient of the 1996 Lawrence A. Hyland Patent Award of Hughes Electronics for his work on low-noise millimeterwave HEMTs, the 1999 IBM Microelectronics Excellence Award for his work in Si/SiGe HBT technology, and the 2003 Custom Integrated Circuit Conference Best Invited Paper Award.