LOCAL DECODING OF WALSH CODES TO REDUCE CDMA DESPREADING COMPUTATION

Albert M. Chan, Jon Feldman, and Raghu Madyastha (Vanu, Inc., Cambridge, MA, USA, {chanal,jonfeld,raghu}@vanu.com); Piotr Indyk and David Karger (Computer Science & Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA, {indyk,karger}@mit.edu)

ABSTRACT

In traditional hardware implementations of the CDMA standard IS-95 [1], signals received at the base station are despread at a rate of 1.2288 Megachips/sec prior to Walsh decoding. However, in a software implementation where low computational complexity is critical, despreading at such a high rate imposes a strain on computational resources. In this paper, we take advantage of the flexibility afforded by software implementations and develop three classes of Walsh decoding algorithms that do not require full despreading of incoming signals at the base station. Two proposed classes of algorithms exploit the fact that Walsh codes are locally decodable codes, which have the surprising property that any bit of the message can be recovered (with some probability) by examining only a small number of symbols of the codeword. We also describe a third class of algorithms based on code puncturing. All these algorithms enable trading off computation for performance or, from another perspective, they enable the system to dynamically adapt the computational requirements of the despreader and subsequent Walsh decoder to changing channel conditions such that a target bit-error rate (BER) is maintained. These algorithms are applicable to other CDMA-based systems that use Walsh codes for orthogonal modulation, and the third class is also applicable to CDMA-based systems such as UMTS (3G WCDMA) that use codes for channelization.

1. INTRODUCTION

The design of receiver algorithms for code division multiple access (CDMA) cellular phone signals is challenging because CDMA signals must be processed at high rates compared to other cellular phone standards. In the IS-95 reverse link (mobile to base station communication in the second-generation CDMA standard [1]), chip pulses are transmitted at a rate of 1.2288 Megachips/sec. In the receiver despreading process, each chip is multiplied by a bit of the user's long code, where the long code is computed by applying a mask to a linear feedback shift register (LFSR). After the despreading process, the resulting sequence of bits must be further processed to determine the Walsh codewords transmitted by the mobile. The computation rate for long code computation, receiver despreading and Walsh decoding is thus roughly some multiple of 1.2288 MHz. Since a CDMA signal may propagate via many paths from mobile to base station, the base station receiver typically repeats the despreading and Walsh decoding processes for a number of received copies of the signal (usually four), thus scaling the computation rate by another factor. Furthermore, for a base station, the despreading and Walsh decoding processes for multiple paths must be performed for *each* mobile in the system, scaling the computation rate by yet another one to two orders of magnitude. Taking all this into consideration, these processes utilize a significant proportion of the available computational resources on a multi-GHz processor.

In this paper, we introduce classes of techniques to reduce the computational requirements of processing IS-95 reverse link signals at the base station. These techniques are also applicable to other advanced CDMA-based receivers that use Walsh codes for orthogonal modulation, and the code puncturing technique can also be modified to a partial correlation technique for other CDMA-based receivers that use codes for channelization. In particular, we develop methods that process only a fraction of the despread signal to decode Walsh codewords. In turn, only a fraction of the received signal needs to be despread, and only a fraction of the long code needs to be computed.

The remainder of this paper is organized as follows. We review the reverse link of IS-95 in Section 2 and Walsh codes, used for orthogonal modulation in IS-95, in Section 3. Three classes of Walsh decoding algorithms that use only a fraction of the despread signal are introduced in Section 4. The first is local decoding—a well known concept in the computational complexity and cryptography communities [2], but here we present the novel application of local decoding to a noncoherent receiver. The second algorithm class is generalized local decoding, which is a novel and elegant generalization of local decoding. The third is punctured decoding, which follows from the observation that the generator matrix of a Walsh code contains all distinct columns. We analyze the performance of these algorithms

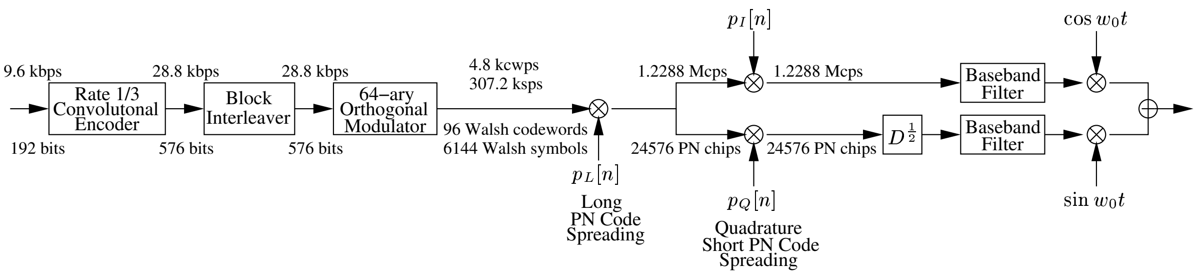

Figure 1: IS-95 reverse link physical layer at mobile.

using the simulation of additive white Gaussian noise (AWGN) channels. The different classes of techniques are characterized by different tradeoffs between computational complexity and bit-error rate (BER). Section 5 explores a realistic wireless scenario in which base stations encounter reverse link channels with fluctuating instantaneous SNR. We present Walsh decoders based on local decoding, generalized local decoding, and punctured decoding that adapt the amount of computation according to the instantaneous SNR, so that a target BER is achieved with minimal computation. Conclusions are presented in Section 6.

2. IS-95 REVERSE LINK

This section describes physical layer processing for the reverse link of IS-95 [1]. For simplicity, we focus on the 9.6 kbps data rate of Rate Set I on the reverse traffic channel, but the concepts are similar for other vocoder rates and for the reverse access channel.

Figure 1 shows the physical layer processing of the reverse traffic channel at the mobile. The traffic channel bits for Rate Set I are encoded using a rate-1/3 convolutional encoder followed by block interleaving. After interleaving, the coded bits, at 28.8 kbps, are mapped six bits at a time into one of 64 possible Walsh codewords of length 64. This modulation provides additional coding gain and simplifies noncoherent detection of the data at the base station. The resulting ± 1 symbol stream, now at a rate of 28.8 x 64/6 = 307.2 ksps, is spread by a long code generated based on a mask assigned to the specific user. Specifically, each symbol of every Walsh codeword is spread onto four chips of the long pseudo-noise (PN) code, resulting in a spread spectrum signal with rate 1.2288 Megachips/sec. After long PN code spreading, the chips are fed into the quadrature spreading and modulation structure, which includes multiplication of the long code spreading output by short in-phase and quadrature-phase PN codes, a delay of the resulting quadrature-phase signal by half a PN chip, baseband filtering, and quadrature modulation.

Unlike the forward link of IS-95, the reverse link does not utilize a pilot channel and thus signal processing at the base station is noncoherent. The term “noncoherent” means that the random phase associated with the received carrier is unknown, so detection is done by observing the envelope of the received signal. Under the assumption that proper timing synchronization has been achieved, the optimal noncoherent receiver structure for one path of the IS-95 reverse link [3] is depicted in Fig. 2. It is sufficient for our purposes to note that correlations of each of the 64 possible Walsh codewords with two signals derived from the received signal are combined into a set of nonnegative squared correlation values. The 64 squared correlation values indicate the likelihood of each of the 64 possible Walsh codewords. Taking into account the interleaver and the six bits that each Walsh codeword represents, the set of 64 correlation values can be converted into soft information to be fed into a convolutional decoder [4].

3. WALSH CODES

The Walsh code of length $n = 2^k$ is a set of perfectly orthogonal codewords that can be defined and generated by the rows of a $2^k \times 2^k$ Hadamard matrix [5]. Starting with a 1×1 matrix $\mathbf{H}_1 = [0]$, higher-order Hadamard matrices can be generated by the following recursion:

$$\mathbf{H}_{2^k} = \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & \overline{\mathbf{H}}_{2^{k-1}} \end{bmatrix}.$$

For example, $\mathbf{H}_4$ can be recursively generated as

$$\mathbf{H}_1 = [0], \mathbf{H}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{H}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

so that the Walsh codewords of length 4 in bipolar (± 1) form are $\mathbf{w}_0 = [+1,+1,+1,+1]$, $\mathbf{w}_1 = [+1,-1,+1,-1]$, $\mathbf{w}_2 = [+1,+1,-1,-1]$, and $\mathbf{w}_3 = [+1,-1,-1,+1]$. Note that the codewords are mutually orthogonal; that is, the dot product of any pair of codewords is zero.

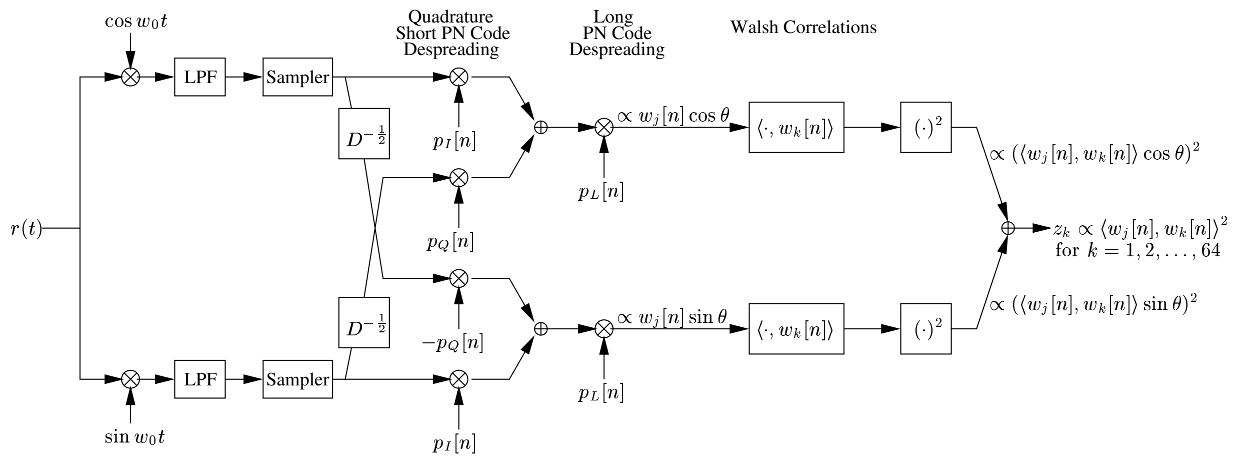
$\cos w_0 t$
LPF
Sampler
$D^{-\frac{1}{2}}$
Quadrature Short PN Code Despreading
Long PN Code Despreading
Walsh Correlations
$p_I[n]$
$p_Q[n]$
$-p_Q[n]$
$p_L[n]$
$\propto w_j[n]\cos\theta$
$\propto w_j[n]\sin\theta$
$\langle \cdot, w_k[n]\rangle$
$(\cdot)^2$
$\propto (\langle w_j[n], w_k[n]\rangle \cos\theta)^2$
$\propto (\langle w_j[n], w_k[n]\rangle \sin\theta)^2$
$z_k \propto \langle w_j[n], w_k[n]\rangle^2$ for $k = 1, 2, \ldots, 64$
$r(t)$
$\sin w_0 t$

Figure 2: Optimal noncoherent receiver structure for one IS-95 signal path.

Because of the special structure inherent in Walsh codes, it is well known that the correlations of a sequence with the 2^k Walsh codewords can be efficiently computed in $O(n \log n)$ operations using the fast Hadamard transform (FHT) [5]. The FHT has the same butterfly structure as the fast Fourier transform (FFT) [6], but the coefficients for the FHT are ± 1 rather than complex exponentials. Thus, the correlations in the optimal noncoherent receiver described in Section 2 can be performed by two FHTs of length 64.

4. SUBCODE-BASED WALSH DECODING

Traditionally, Walsh codes are decoded by optimal Walsh decoding with fixed computational complexity. In this section we introduce algorithms that efficiently trade off computational complexity and BER based on the mathematical structures underlying Walsh codes. These algorithms are suboptimal but effective and form the basis of the dynamic adaptation algorithms of Section 5.

A generator matrix for a binary linear code C is a matrix **G** whose rows form a basis for C. Thus, the codewords in binary form can be generated by computing **c=xG** for all $1\times k$ binary row vectors **x**.

The generator matrix for the (64,6) Walsh code used in IS-95 is a 6×64 matrix whose columns are the 64 distinct 6×1 binary vectors; i.e.,

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & & 1 \end{bmatrix}.$$

The special structure of this generator matrix makes it possible to efficiently decode the Walsh codeword without examining the entire codeword. In this section, we describe three such techniques.

4.1 Local Decoding

The special structure of Walsh codes permits an estimate of one of the six input bits by observing only two of the 64 symbols of the received Walsh codeword, via a process known as *local decoding*, which is well known in the computational complexity and cryptography communities [2]. To understand this technique, we label the symbols of the codeword as $\mathbf{c} = [c_0, c_1, c_2, \ldots, c_{63}]$, where the subscripts are the decimal values of the corresponding binary column vectors in **G**. Suppose we wish to decode the zeroth bit of $\mathbf{x} = [x_0, x_1, x_2, x_3, x_4, x_5]$ in the absence of receiver noise. We can obtain the value of x_0 by modulo-2 adding any pair of codeword components c_i and c_j for which the binary representations of i and j (and the corresponding columns in **G**) differ only in the zeroth bit position:

$$\begin{aligned} c_i \oplus c_j &= \left(\sum_{s=0}^{5} x_s G_{si}\right) \oplus \left(\sum_{t=0}^{5} x_t G_{tj}\right) \\ &= \sum_{s=0}^{5} x_s \left(G_{si} \oplus G_{sj}\right) \\ &= x_0 \left(G_{0i} \oplus \overline{G_{0i}}\right) \oplus \sum_{s=1}^{5} x_s \left(G_{si} \oplus G_{si}\right) \\ &= x_0 . \end{aligned}$$

For example, c_2 and c_{34} form one of the 32 possible pairs that can be used to decode x_0. In a practical situation in which Walsh codewords are represented in bipolar (± 1) format as opposed to binary format, the modulo-2 addition is replaced by normal multiplication of c_i and c_j.

The signal-to-noise ratios (SNRs) encountered in practice are usually too low to reliably decode a bit by examining only one pair of codeword symbols. Instead, the sum of products of up to 32 bipolar pairs can be quantized to ± 1 to decode one bit in a "soft voting" procedure.

Our use of local decoding in a noncoherent system is novel. The resulting receiver structure is the same as in Fig. 2 except that the two Walsh correlators are replaced by two local decoders that sum the products of the same pairs, and the subsequent squarers are eliminated since the multiplication of symbol pairs already squares $\cos\theta$ and $\sin\theta$. The soft votes from the two local decoders are then added together to eliminate any dependency on θ, and the sum is quantized to ± 1 to decode one bit.

The simulated AWGN BER for local decoding of the IS-95 reverse link as a function of both the number of pairs used to decode each bit and the SNR per information bit is depicted in Fig. 3. (For the purposes of computing SNR per information bit in this paper, "information bit" shall refer to bits at the input of the Walsh encoder and not the bits at the input of the convolutional encoder.) On the leftmost part of the graph is the baseline curve of optimal noncoherent decoding. The successive curves to the right are generated by locally decoding each input bit using 32, 16, 8, 4, 2 and 1 pair(s) of codeword symbols. Although local decoding using 32 pairs of codeword symbols makes use of the entire Walsh codeword, its performance is nearly 3 dB worse than optimal decoding because the relationships between symbols in different pairs are not fully exploited. Each reduction of the number of local decoding pairs by a factor of two causes an additional loss of 2 dB.

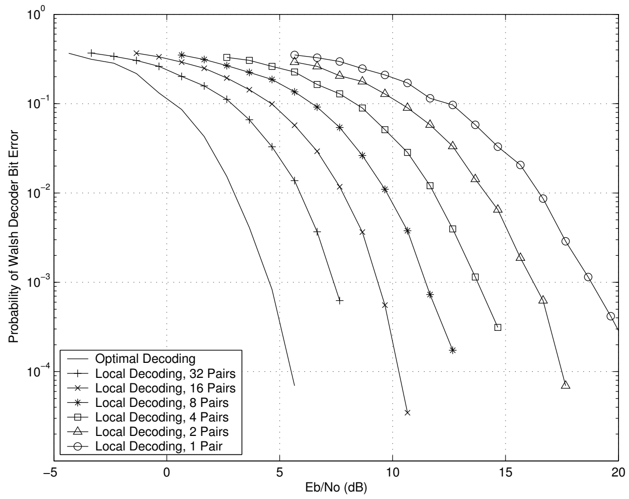

Figure 3: BER of 2-point local Walsh decoding for AWGN channels as a function of SNR per information bit.

4.2. Generalized Local Decoding

To decode two input bits using local decoding, one pair of symbols in the Walsh codeword can decode one bit and another pair can decode the other bit. While local decoding can decode bits using few computational operations and examining potentially few codeword symbols, one might expect to get a lower BER by *jointly* decoding both input bits when constrained to examine only 4 of the 64 symbols of a received Walsh codeword.

We have developed an elegant generalization of local decoding to perform such joint decoding, which uses certain groups of four symbols in the Walsh codeword to jointly decode two input bits at low BERs. In local decoding, the two symbols c_i and c_j used to determine the bit x_t are chosen such that the binary representations of i and j differ only in the tth bit position. In what we call *generalized local decoding*, the four symbols c_g, c_h, c_i and c_j used to determine the pair of bits x_s and x_t are chosen such that the binary representations of g, h, i and j all differ now in both the sth and tth bit positions.

These four symbols are processed in a way that generalizes local decoding. In local decoding, x_t is determined by taking the product of codeword components c_i and c_j in bipolar format and quantizing to ± 1. This process is equivalent to performing a 2-point FHT on the two-component vector $[c_i, c_j]$ where $i < j$, and determining x_t to be the binary index of the maximum component of the FHT. That is, if the zeroth FHT component is largest then decode as 0, otherwise the first FHT component is largest and decode as 1. In generalized local decoding, a 4-point FHT is performed on the four-component vector $[c_g, c_h, c_i, c_j]$ where $g < h < i < j$, and x_s and x_t are decoded as the binary index of the maximum component of the FHT where $s < t$.

Generalized local decoding works also for groups of 8, 16, and 32 codeword symbols, for which 3, 4, and 5 bits are decoded respectively. Note that at one extreme, generalized local decoding for groups of 2 codeword symbols is simply local decoding, while at the other extreme generalized local decoding for the entire 64 symbols of the Walsh codeword is optimal Walsh decoding implemented by a 64-point FHT. All six input bits can be decoded using combinations of various-sized FHTs. For example, one could use three 4-point FHTs or two 8-point FHTs.

As in the previous section, multiple groups of codeword symbols can be processed and combined to decode input bits. In local decoding, up to 32 pairs of codeword symbols can be used to decode one input bit because there are 32 choices for the five fixed bits in the binary representation of the codeword symbol indices. In general, up to 2^{6-p} groups of 2^p codeword symbols can be used to decode p input bits

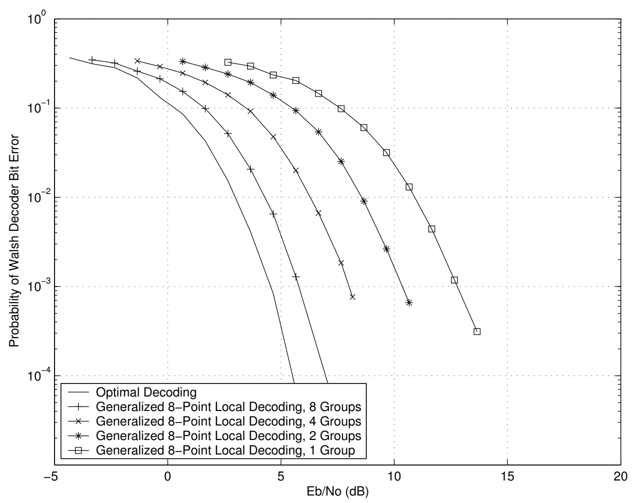

Figure 4: BER of generalized 8-point local Walsh decoding for AWGN channels as a function of SNR per information bit.

because there are 2^{6-p} choices for the fixed $6-p$ bits in the binary representation of the codeword symbol indices. Because the reverse link of IS-95 is noncoherent, the components of the 2^{6-p} FHTs are squared before being added component-wise. Figure 4 shows the simulated AWGN BER for generalized local decoding of the IS-95 reverse link as a function of both the number of 8-point FHTs used to decode three bits and the SNR. From Figs. 3 and 4, it is clear that if the number of codeword symbols examined to decode each input bit is fixed, better performance is obtained with higher-order FHTs.

4.3. Punctured Decoding

Thus far, we have been exploiting $(2^p, p)$ codes that are found within the (64,6) Walsh code—that is, we have been using a subset of the codeword symbols to decode a subset of the input bits.

In this section, we use a subset of the codeword symbols to decode *all* 6 input bits. Since the columns of the generator matrix **G** for the (64,6) Walsh code include all possible 6-bit binary vectors, any $(n,6)$ binary linear code with $n < 64$ and distinct generator matrix columns can be obtained by puncturing those symbols in the (64,6) Walsh code that correspond to unwanted columns in **G**.

For a given $n < 64$, the set of all possible $(n,6)$ linear codes created by distinct puncturing patterns has a wide range of BERs. For example, some $(n,6)$ codes have a row of zeros in their generator matrices and thus have a minimum Hamming distance of zero, while others maximize the minimum distance. In noncoherent systems such as IS-95, the $(n,6)$ code that maximizes the minimum distance does not necessarily lead to the best BER, and can even lead to performance much worse than randomly choosing a $(n,6)$ code. Take for instance the (32,6) biorthogonal code, which is the (32,6) linear code with the maximum minimum distance. However, the complement of each codeword is also a codeword, so each codeword can be mistaken for its complement and vice-versa in a noncoherent system with unknown carrier phase. A more appropriate criterion for selecting a $(n,6)$ code for a noncoherent system is to maximize the minimum distance and simultaneously minimize the maximum distance. While tables of codes that maximize minimum distance are readily available and are constantly improved and expanded [7], we are not aware of similar tables that simultaneously consider both metrics.

Although the development of such tables would be an interesting area for future research, it is well known in coding theory that randomly generated codes are good codes with high probability, especially for long codeword lengths (large n) [8]. In light of this, we can randomly select a subset of n codeword symbols from the (64,6) Walsh code, thereby effectively obtaining an $(n,6)$ code that, with high probability, has good BER performance.

The maximum-likelihood decoder for the resulting $(n,6)$ code is the minimum-distance decoder which, in the case of bipolar codewords, is equivalent to taking the correlations of the 64 possible codewords of length n with the punctured received codeword, and selecting the codeword corresponding to the maximum. This set of 64 correlations can still be efficiently computed using an FHT on the received codeword with the $64-n$ unwanted codeword symbols set to zero. In a noncoherent system such as IS-95, the receiver structure is similar to Fig. 2, except that puncturing is prior to the Walsh correlations.

Figure 5 shows the simulated AWGN BER curves for randomly punctured codes and for codes with only minimum distance optimized, with parameters (60,6), (56,6), (48,6), (32,6), and (16,6). The fact that the biorthogonal (32,6) code has indistinguishable codewords in noncoherent systems is reflected by the poor BER when compared to the ensemble of (32,6) codes. For the other code lengths simulated, the performance of the code with maximum minimum distance is similar to the average performance of codes in the ensemble, and even slightly better in the high SNR regime where errors are perhaps mostly caused by codewords at the minimum distance.

5. ADAPTIVE SUBCODE-BASED WALSH DECODING

Like typical wireless channels, the IS-95 reverse link encounters large fluctuations in instantaneous SNR. The fluctuations due to path loss and shadowing are partially mitigated by open-loop and closed-loop power control, but the rapid SNR fluctuations due to multipath fading remain, especially at high mobile speeds. The techniques presented in the previous section provide a variety of operating points in the tradeoff plane between BER and computational complexity at a fixed SNR, so in theory if we knew the instantaneous SNR, we could choose the specific technique (and thus, operating point) at that time to achieve the target BER with as little computation as possible. However, this kind of information is not necessarily available or accurate at the base station.

To address this issue, we can test a series of suboptimal Walsh decoding algorithms with operating points that simultaneously move towards higher computational complexity and lower BERs until the target BER is achieved. Although we cannot know the true BER, we can get an indication by monitoring a reliability metric generated by the Walsh decoding algorithm until it exceeds a threshold. The suboptimal Walsh decoding techniques can be chosen such that only an incremental amount of computation is required for the next, slightly more complex technique. This idea of incrementally increasing computation until sufficient reliability is achieved is similar in spirit to decoding "rateless" codes [9,10].

We test our adaptive algorithms by simulating the IS-95 reverse link, assuming that the fluctuating SNR per information bit at the base station has a lognormal distribution with mean 4.7 dB and variance 1.5 dB. Each adaptive Walsh decoding technique at a particular stopping threshold corresponds to a point in the tradeoff plane between BER and computational complexity, averaged over many IS-95 frames experiencing independent fades.

Due to implementation-specific software optimizations, it is not very meaningful or realistic to count the total number of algorithmic operations for long code generation, despreading, and Walsh decoding as a measure of computational complexity. Rather, we take as a measure the average number of Walsh codeword symbols used to decode all six input bits, because the computation for long code generation, despreading, and Walsh decoding is roughly linear in this quantity.

The tradeoff between BER and average computational complexity for adaptive local decoding is shown in Fig. 6. In this algorithm, the number of pairs used to locally decode one of the six input bits is increased by increments of one until the magnitude of the sum of the pair products exceeds a threshold. Since there are 64 possible terminations to the algorithm, we call this a 64-stage algorithm. Each point was generated by assuming a specific stopping threshold, and assuming that the order in which the pairs are examined is chosen at random. The curve is traced out from left to right as the threshold increases from 0 to infinity; at low thresholds the algorithm typically stops after looking at few pairs, while at high thresholds the algorithm typically stops after looking at many pairs. The reason the tradeoff curve here is not a very good one is that even if few pairs are used to decode each of the six bits, most of the pairs for each of the six bits do not correspond to the same Walsh codeword symbols, and computational complexity becomes very high.

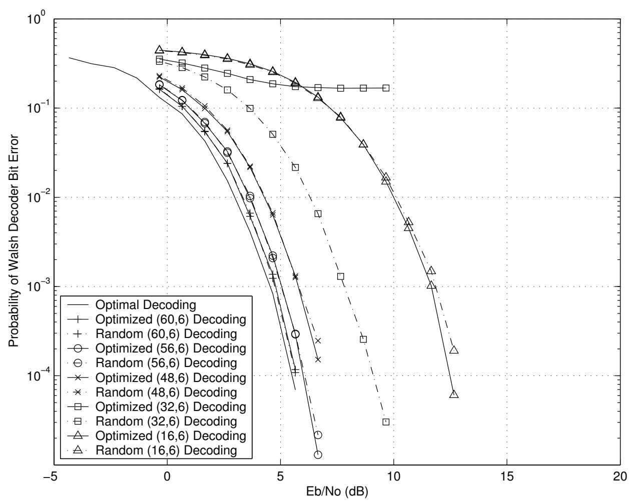

Figure 5: BER of punctured Walsh decoding for AWGN channels as a function of SNR per information bit.

Figure 6 also shows the tradeoff curve for an adaptive generalized local decoding algorithm using 8-point FHTs. This 8-stage algorithm increments the number of 8-point FHTs used to decode three of the six input bits until the metrics for the three bits all cross a threshold. When each additional 8-point FHT is used, the squared correlations are added to the sum of squared correlations of the 8-point FHTs already used. The metric for each bit, for the purposes algorithm termination, is the absolute difference between the maximum sum of squared correlations over all three-bit input patterns with a zero in that bit position and the maximum sum of squared correlations over all those patterns with a one in that bit position. This adaptive generalized local decoding algorithm creates a better tradeoff than adaptive local decoding.

The tradeoff curve for adaptive punctured decoding is also shown in Fig. 6, corresponding to a 64-stage algorithm in which up to all 64 Walsh codeword symbols are used to decode all six input bits. When an additional codeword symbol is used, the length of the 64 correlations is extended by that additional symbol. The metric for each bit, for the purposes of stopping the algorithm, is the absolute difference between the maximum squared correlation over all six-bit input patterns with a zero in that bit position and the maximum squared correlation over all those patterns with a one in that bit position. This 64-stage algorithm approaches the BER of optimal noncoherent decoding (the BER of the rightmost point on the curve) with lower computational complexity. Note that adaptive random punctured decoding has the better tradeoff at low BERs, while adaptive generalized 8-point local decoding has the better tradeoff at higher BERs.

In Fig. 7, we compare the BER and computational complexity of optimal noncoherent decoding specifically to adaptive random punctured decoding. While optimal decoding requires a fixed amount of computation regardless

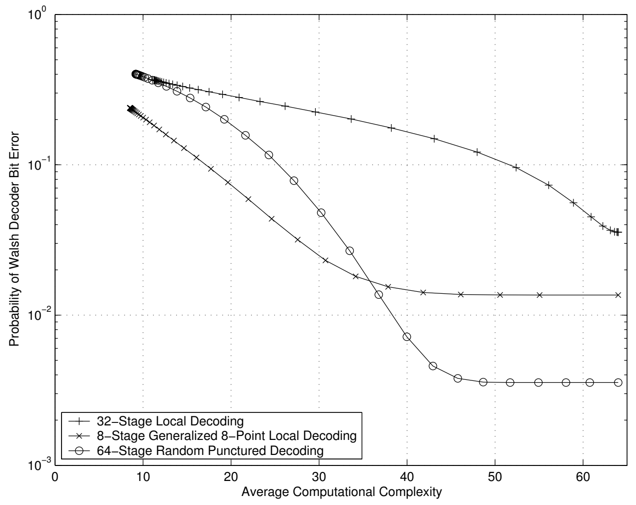

Figure 6: Tradeoff between BER and average computational complexity for the three classes of adaptive subcode-based Walsh decoding. The SNR per information bit is lognormally distributed with mean 4.7 dB and variance 1.5 dB.

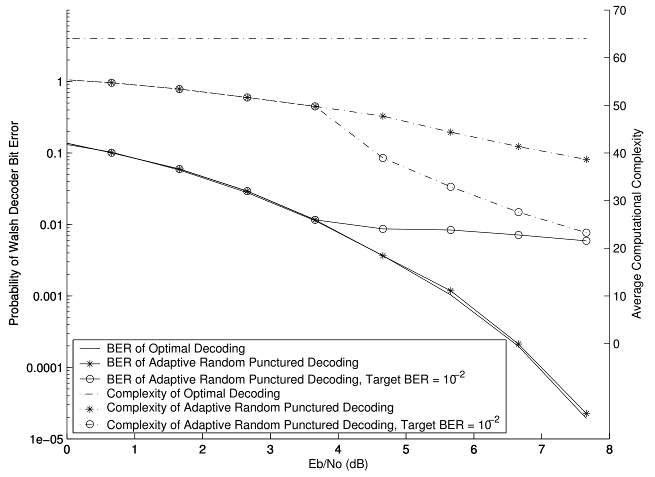

Figure 7: BER and average computational complexity of adaptive random punctured decoding as a function of average SNR per information bit. The variance of the SNR is 1.5 dB.

of SNR, similar BERs can be achieved by adaptive random punctured decoding with an amount of computation that decreases as SNR increases. If the goal is to achieve a target BER of 10^{-2}, the computation of adaptive random punctured decoding is even less.

Although we have used the same threshold at each stage of these multi-stage algorithms for convenience, there is no reason that different thresholds cannot be used at different stages. For a given algorithm, varying the thresholds at each stage generates different tradeoff curves. Thus, unlike Fig. 6, the set of points in the tradeoff plane achievable by a particular adaptive algorithm is a region rather than a curve. While it is difficult to analytically or numerically determine the optimal thresholds that achieve points that lie on the lower left border of the achievable region, it is possible to find distinct thresholds for each stage that generate improved tradeoff curves. This is an interesting area for future research.

Furthermore, adaptive algorithms with fewer stages may be more practical. For example, a 2-stage adaptive local decoding algorithm looks at a certain number of pairs, and then decides whether to examine the remaining pairs. If designed correctly, the reduction in the number of stages can simplify the algorithm and yet retain a similar tradeoff.

6. CONCLUSIONS

In this paper, we have developed several classes of algorithms that reduce the amount of computation required for Walsh decoding and, in turn, for despreading and long code generation. These algorithms are made possible by software radio's inherent flexibility, which allows us to monitor reliability metrics as successively larger portions of a Walsh codeword are examined and to detect the codeword when reliability is sufficient.

Software radio compels a paradigm shift in the design of new signal processing algorithms. This paper provides a flavor of how algorithms can be designed for software rather than traditional hardware.

7. REFERENCES

[1] *EIA/TIA/IS-95 Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System*, Telecommunications Industry Association, July 1993.

[2] L. Trevisan, "Some Applications of Coding Theory in Computational Complexity," *Quaderni di Matematica*, Vol. 13, pp. 347-424, 2004.

[3] J.S. Lee and L.E. Miller, *CDMA Systems Engineering Handbook*, Artech House, Boston, 1998.

[4] A.J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Addison-Wesley, Reading, MA, 1995.

[5] R.K. Yarlagadda and J.E. Hershey, *Hadamard Matrix Analysis and Synthesis, with Applications to Communications and Signal/Image Processing*, Kluwer Academic, Boston, 1997.

[6] A.V. Oppenheim and R.W. Schafer, with J.R. Buck, *Discrete-Time Signal Processing,* 2nd ed., Prentice Hall, Upper Saddle River, NJ, 1999.

[7] V.S. Pless and W.C. Huffman, Eds., *Handbook of Coding Theory*, Vol. 1, Elsevier Science, Amsterdam, 1998.

[8] S.J. MacMullan and O.M. Collins, "A Comparison of Known Codes, Random Codes, and the Best Codes," *IEEE Transactions on Information Theory*, Vol. 44, No. 7, pp 3009-3022, Nov. 1998.

[9] M. Luby, "LT-codes," in *Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp.271-280, 2002.

[10] A. Shokrollahi, "Raptor codes," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 36, 2004