Design and Implementation of a Low-Complexity Symbol Detector for Sparse Channels

Yanjie Peng, Xinming Huang, *Senior Member, IEEE*, Andrew G. Klein, *Member, IEEE*, and Kai Zhang

***Abstract*— In this paper, we present a low-complexity symbol detector for communication channels which have long spanning durations but a sparse multipath structure. Traditional maximum-likelihood sequence estimation using the Viterbi algorithm can provide optimal error performance for eliminating the multipath effect, but the hardware complexity grows exponentially with channel length and it is not practical for long sparse channels. We implement a near-optimal algorithm and its architecture by cascading an adaptive partial response equalizer (PRE) with an iterative belief propagation (BP) detector. A sparse channel is f rst equalized by a PRE to a target impulse response (TIR) with only a few nonzero coeff cients remaining. The residual intersymbol interference is then canceled by a BP detector whose complexity is solely dependent on the number of nonzero coeff cients in the TIR. Moreover, we present a pipeline high-throughput implementation of the detector for channel length 30 with quadrature phase-shift keying modulation. The detector can achieve a maximum throughput of 206 Mb/s with an estimated core area of 3.162 mm^2 using 90-nm technology node. At a target frequency of 515 MHz, the dynamic power is about 1.096 W.**

***Index Terms*— Belief propagation (BP), low complexity, partial response equalizer, sparse channel, symbol detection.**

I. Introduction

SPARSE channels are characterized as having long multipath delay spreads but with few nonzero coefficients. Such channels arise in a number of modern wireless communication applications. For instance, in underwater acoustic communications, the intersymbol interference (ISI) can span several hundreds of symbols but the multipath structure is usually very sparse [1]. In hilly terrain broadband wireless communications [2] and high-definition television (HDTV) [3], the delay profile also exhibits a sparse channel impulse response (CIR).

In these applications, ISI is one of the major impairments to reliable symbol detection. The optimal maximum-likelihood (ML) detector based on the Viterbi algorithm (VA) may be used to estimate the transmitted sequence in the presence of the ISI [4]. However, the computational complexity of the ML detector increases exponentially with the delay spread of the channel. Thus, the optimal ML detector is impractical for sparse channels whose delay spread is usually very long. Near-optimal detectors for sparse channels have been investigated by several researchers. The parallel trellis VA (PTVA) [5] reformulated the original single trellis into a set of independent trellises which could operate in parallel and have much less complexity. The PTVA requires that the sparse channel have equispaced coefficients, which, however, often cannot be satisfied in practice. Although a generalized PTVA is proposed in [5] to address general sparse channels, optimal performance can be guaranteed only if the CIR is well matched to an equispaced structure. A near-optimal detector based on a multitrellis VA (MVA) [6] was proposed that decomposes the trellis into multiple irregular sub-trellises by investigating the dependencies between the received symbols. It was shown that the complexity does not depend on the channel length but only on the number of nonzero coefficients. However, the trellis decomposition is not straightforward for a general sparse channel. Decision feedback sequence estimation (DFSE) [7], [8], which is a popular scheme for long ISI channels, can also be an alternative for sparse channels. However, it yields high performance only if the CIR is minimum-phase. The whole system could be unstable if a prefilter is used in front of the DFSE to transform the CIR to its minimum-phase equivalent. In a recent work, iterative belief propagation (BP) detectors have been proposed for ISI channels [9]–[11]. It has been shown in [9] that, for an uncoded system, the BP detector achieves near-optimal performance over frequency-selective ISI channels. Furthermore, the BP detector has a complexity which depends only on the number of nonzero channel coefficients, and thus is well suited for sparse channels.

The BP algorithm, also known as the message passing algorithm or sum-product algorithm, has been widely used for iterative decoding of low-density parity-check (LDPC) codes [12], and its implementation has been well studied in [13]–[17]. However, the implementation of a BP detector for ISI channels is remarkably different from the LDPC decoder in the following aspects. First, LDPC codes are usually designed to have a structure that facilitates node processing in parallel and reduces the complexity of message passing during the iterative decoding process; an example of such structured codes is quasicyclic (QC) LDPC codes. In the case of LDPC decoding, the structured parity check matrix consists of purely binary values and it is effectively a constant matrix. For BP-based symbol detection in ISI channels, the analogous Tœplitz channel matrix is not constrained to the binary numbers, is not a fixed constant, and can even be time-varying. Consequently, it is difficult to exploit parallelism in the BP detector, and one must resort to serial processing. Second, the

Manuscript received January 10, 2012; revised May 9, 2012; accepted July 21, 2012. The work of Y. Peng was supported in part by the MathWorks Fellowship.

The authors are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: yjpeng@wpi.edu; xhuang@wpi.edu; klein@wpi.edu; zkjuven@gmail.com).

calculation of messages passing between nodes is significantly more complex for symbol detection than for LDPC decoding. In fact, the complexity of the BP detector increases exponentially with the number of nonzero channel coefficients, which makes it impractical to implement BP detectors even for sparse channels where the number of nonzero coefficients is on the order of 10. One solution to permit the use of near-optimal BP detectors is to first pass the received signal through a partial response equalizer (PRE) [11] which equalizes the original sparse channel to an even more sparse channel with fewer nonzero coefficients. This permits the design effort of the BP detector to be greatly mitigated while maintaining near-optimal performance [11]. Third, the channel matrix for the BP detector is Tœplitz and therefore has a banded structure which simplifies the wire routing for the message passing.

By investigating the aforementioned differences, we present an implementation of an efficient symbol detector for sparse channels as in Fig. 1. The symbol detector consists of two blocks: the adaptive PRE and the BP detector. In the PRE, the channel is first partially equalized to a target impulse response (TIR) which is designed to have only a small fixed number of nonzero coefficients. The coefficients of PRE and TIR are obtained based on the minimum mean squared error (MMSE) criterion, which is designed adaptively using the least mean square (LMS) algorithm. Once the coefficients of the PRE and TIR are computed, the partial equalization is carried out by a finite impulse response (FIR) filter, and the residual ISI is eliminated by the BP detector.

The architecture of the BP detector is based on a pipelined layer processing scheme, which allows high throughput with low complexity. The BP detector is also designed to be reconfigurable so that it can adapt to time-varying ISI channels. Since the complexity of the BP detector merely scales with the number of nonzero channel coefficients, we limit the nonzero coefficients of the TIR to keep a reasonable complexity of the BP block. Note that a system designer can specify the TIR to be sparse even when the original CIR is not sparse; as pointed out in [11], this implies that the symbol detector is general enough to work even for nonsparse channels. We implement the proposed symbol detector for quadrature phase-shift keying (QPSK) modulation. We consider a sparse channel with 30 coefficients, a PRE with 100 coefficients, and a TIR designed to have 3 nonzero coefficients. The BP detector operates on 1024 signal samples for five iterations per frame. The target frequency for synthesis is set as 515 MHz, for an equivalent throughput of 206 Mb/s. The synthesized result shows an estimated area of 3.162 mm^2 in TSMC 90-nm technology, and a total dynamic power of 1.096 W. To the best of our knowledge, this is the first implementation of a PRE/BP detector.

The rest of this paper is organized as follows. In Section II, we gives the communication system model. The PRE is described in Section III-A. The factor graph representation and BP algorithm are described in Section III-B and the implementation of the symbol detector is given in Section IV. We show the simulation performance and implementation results in Sections V and VI, followed by conclusions in Section VII.

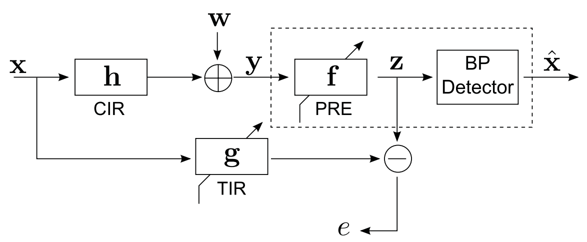


Fig. 1. System model of the sparse channel symbol detector.

II. System Model

We first introduce some notations for the description of the communication system. Assume that a length N sequence of M-ary symbols $x[k] \in \{a_0, a_1, \ldots, a_{M-1}\}$ is transmitted through a complex ISI channel whose discrete-time CIR is described as $\mathbf{h} = [h[0], h[1], \ldots, h[L_h - 1]]^T$, where L_h is the length of the CIR. Letting D_h be the number of nonzero coefficients of $\mathbf{h}$, then $D_h \ll L_h$ for sparse channels. The received signal sample at time instant k can be expressed as

$$y[k] = \sum_{i=0}^{L_h-1} h[i]x[k-i] + w[k] \tag{1}$$

where $w[k]$ is complex additive white Gaussian noise (AWGN) with noise power σ^2.

At the receiver, the symbol detector consists of two cascaded components: an adaptive PRE denoted by $\mathbf{f}$, and a BP detector. The system model is shown in Fig. 1. The goal of the adaptive PRE is to equalize the original CIR $\mathbf{h}$ to a TIR $\mathbf{g}$ which has fewer nonzero coefficients, thus making second stage detection via BP computationally efficient. The adaptive PRE and BP detector are discussed in detail in the next section.

III. Design of the Symbol Detector

A. Partial Response Equalizer

The PRE equalizes the channel to a sparse TIR, and is designed by jointly finding the set of PRE and TIR coefficients that minimize the mean squared error (MSE). The PRE is implemented as an FIR filter whose coefficients are $\mathbf{f} = [f[0], f[1], \ldots, f[L_f - 1]]^T$, where L_f is the chosen length of the PRE. Conventional wisdom for classical equalizer design [4] dictates that the equalizer length be approximately five times the channel length L_h. Here, however, the PRE has more degrees of freedom since it does not need to equalize the channel to a single spike. As we will show empirically in Section V, an equalizer length of about three times the channel length (i.e., choosing $L_f \approx 3L_h$) seems sufficient in practice.

We denote the TIR by $\mathbf{g} = [g[0], g[1], \ldots, g[L_g - 1]]^T$, where L_g is the length of the TIR, and denote the nonzero version of the TIR by $\mathbf{g}' = [g[l_1], g[l_2], \ldots, g[l_{D_g}]]^T$, where D_g is the number of nonzero coefficients. Choosing D_g in the sparse TIR $\mathbf{g}$ is a tradeoff between error-rate performance and detector complexity. When D_g is small, the complexity of the BP detector is reduced but the overall error-rate performance is degraded. A larger value of D_g adds complexity to the BP detector but also leads to better error-rate performance. In general, the nonzero coefficients of $\mathbf{g}$ are chosen so that

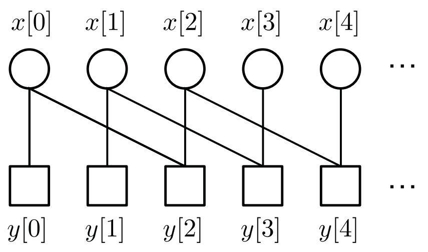
```
Algorithm 1 Summary of PRE LMS Algorithm
Parameters:
TIR at time $n$: $\mathbf{g}_n = [g_n[0], g_n[1], \ldots, g_n[L_g - 1]]^T$
Indices of nonzero coefficients of the TIR: $l_1, l_2, \ldots, l_{D_g}$
Nonzero version of the TIR at time $n$:
$\mathbf{g}'_n = [g_n[l_1], g_n[l_2], \ldots, g_n[l_{D_g}]]^T$
PRE at time $n$: $\mathbf{f}_n = [f_n[0], f_n[1], \ldots, f_n[L_f - 1]]^T$
Step size for TIR updating: $\mu_g$
Step size for PRE updating: $\mu_f$
Received symbols vector at time $n$:
$\mathbf{y}(n) = [y[n], y[n-1], \ldots, y[n - L_f + 1]]^T$
Transmitted symbols vector at time $n$:
$\mathbf{x}(n) = [x[n - l_1], x[n - l_2], \ldots, x[n - l_{D_g}]]^T$

Initialization:
Set the middle element of $\mathbf{f}_0$ to 1, and set the first element of $\mathbf{g}'_0$ to 1.

Computation:
For $n = 0, 1, \ldots$, compute:
  1) $e_n = \mathbf{f}_n{}^T \mathbf{y}(n) - \mathbf{g}'_n{}^T \mathbf{x}(n)$;
  2) $\mathbf{f}_{n+1} = \mathbf{f}_n - \mu_f \mathbf{y}(n)^H e_n$;
  3) $\mathbf{g}'_{n+1} = \mathbf{g}'_n + \mu_g \mathbf{x}(n)^H e_n$;
  4) normalize $\mathbf{g}'_{n+1}$.
```

$D_g \leq D_h \ll L_h$. We note that the case of $D_g = 1$ corresponds to classical equalization (in which case the BP detector can be reduced to a memoryless slicer), while the choice of $D_g = D_h$ corresponds to a full-blown BP detector. The locations of these nonzero coefficients in the TIR must be also placed properly. There are L_g-choose-D_g possible sets of locations, and several approaches have been proposed [18], [19] for choosing the locations to optimize various criteria. For simplicity, we adopt the approach in [11] which chooses the locations of the nonzero coefficients of the TIR to coincide with the dominant arrivals in the CIR. We note that this approach assumes that the receiver has some knowledge of the channel $\mathbf{h}$, though it need not know the CIR precisely.

The nonzero coefficients in the TIR as well as the PRE coefficients are then calculated jointly using the MMSE criterion [20], which causes the combined response of the CIR and PRE to approximate the TIR. Direct computation of the MMSE solution of the PRE and TIR involves solving a generalized eigen decomposition and requires full knowledge of the channel $\mathbf{h}$. In order to make the implementation feasible, the PRE can be designed adaptively. In [11], the recursive least squares (RLS) algorithm was employed. Here, we follow the approach of [20] by employing two LMS algorithms in tandem to jointly calculate the PRE and TIR. While this algorithm does not require knowledge of $\mathbf{h}$, it does require the availability of known training data. The dual LMS algorithm is summarized in Algorithm 1. Since only the nonzero coefficients are updated in each iteration, the TIR can be represented by the nonzero values in $\mathbf{g}'$ and their locations $l_1, l_2, \ldots, l_{D_g}$. After the convergence of the LMS algorithm, the PRE output signal at time n is $z[n] = \mathbf{f}^T \mathbf{y}(n)$, which in the absence of noise is ideally equal to $\mathbf{g}'^T \mathbf{x}(n)$.



$x[0]$ $x[1]$ $x[2]$ $x[3]$ $x[4]$ ⋯
$y[0]$ $y[1]$ $y[2]$ $y[3]$ $y[4]$ ⋯

Fig. 2. Factor graph of an example channel $[h[0], 0, h[2]]^T$.

B. BP Detection Algorithm

$$R_{m\to n}(i) = \log \frac{\sum_{\forall \mathbf{x}(m): x[n]=a_i} \exp\left\{ \frac{-\left|z[m] - \mathbf{g}'^T \mathbf{x}(m)\right|^2}{2\sigma^2} + \sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j])) \right\}}{\sum_{\forall \mathbf{x}(m): x[n]=a_0} \exp\left\{ \frac{-\left|z[m] - \mathbf{g}'^T \mathbf{x}(m)\right|^2}{2\sigma^2} + \sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j])) \right\}} \quad (2)$$

$$\approx \max_{\forall \mathbf{x}(m): x[n]=a_i} \left\{ \frac{-\left|z[m] - \mathbf{g}'^T \mathbf{x}(m)\right|^2}{2\sigma^2} + \sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j])) \right\} - \max_{\forall \mathbf{x}(m): x[n]=a_0} \left\{ \frac{-\left|z[m] - \mathbf{g}'^T \mathbf{x}(m)\right|^2}{2\sigma^2} + \sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j])) \right\}, \quad i = 0, 1, \ldots, M-1. \quad (3)$$

The residual ISI of PRE output is further mitigated by a near-optimal BP detector, based on the factor graph [12] which represents the input–output relationship of an ISI channel. Fig. 2 shows the factor graph of a simple example with CIR of $[h[0], 0, h[2]]^T$. In the factor graph, channel input symbols $x[0], x[1], \ldots,$ are represented by the circles on the top (known as variable nodes or bit nodes), and the channel output signals $y[0], y[1], \ldots,$ are denoted by the squares at the bottom (known as check nodes or function nodes). The connections (edges) between variable nodes and check nodes represent dependencies between input and output. For example, $y[3]$ is connected to $x[1]$ and $x[3]$, since $y[3] = h[0]x[3] + h[2]x[1] + w[3]$ in the example.

The BP algorithm is also referred to as a two-phase message passing or sum-product algorithm, in which check-to-variable (CTV) and variable-to-check (VTC) messages are transmitted along the edges to update each other iteratively. During the first phase, the CTV message is computed at the check nodes based on the known channel coefficients and the *a priori* probability information from the variable nodes. The updated message is then passed from each check node to its connected variable nodes. During the second phase, the variable nodes update their *a priori* information and send it back to their connected check nodes. The same procedure is repeated iteratively. After several iterations, the variable nodes are accumulated with sufficient likelihood information and a hard decision can be made for each input symbol. The operations of the BP algorithm in log-likelihood ratio

(LLR) domain are summarized as follows. Note that, since the PRE has been applied, we assume that the sparse channel is the TIR $\mathbf{g}$ (whose nonzero version is $\mathbf{g}'$), and channel input and output at time n are $x[n]$ and $z[n]$, respectively.

1) Calculating CTV Messages: The extrinsic message from check node m to variable node n is computed as (2) if the two nodes are connected, where $\mathbf{x}(m) = [x[m-l_1], x[m-l_2], \ldots, x[m-l_{D_g}]]^T$ is the data vector and $Q_{j\to m}$ is the *a priori* information from variable node j for check node m, and $\mathcal{N}(m)$ is the set of variable nodes connected with check node m. We use the mapping function ψ: $\{0, 1, \ldots, M-1\} \to \{a_0, a_1, \ldots, a_{M-1}\}$ to denote the modulation on the M-ary constellation, and the demapping function is expressed as ψ^{-1}: $\{a_0, a_1, \ldots, a_{M-1}\} \to \{0, 1, \ldots, M-1\}$. The nonlinear log-sum-exponential function in (2) could be implemented with lookup tables (LUTs), but it is not desirable because the LUTs would require a large memory. As a good approximation, the calculation in (2) can be simplified as (3) by Jacobian logarithm with negligible performance loss [21]. Note that for M-ary modulation, there are $M-1$ LLRs $R_{m\to n}(1), R_{m\to n}(2), \ldots, R_{m\to n}(M-1)$ need to be calculated from check node m to variable node n, and $R_{m\to n}(0) = 0$ by (3).

2) Calculating VTC Messages: After receiving the updated extrinsic message from the check nodes, the variable nodes update the *a priori* information for the next iteration. The *a priori* information at variable node n for check node m is calculated as

$$Q_{n\to m}(i) = \sum_{j\in\mathcal{M}(n)\backslash m} R_{j\to n}(i), \qquad i = 0, 1, \ldots, M-1 \tag{4}$$

where $\mathcal{M}(n)$ is the set of check nodes connected with variable node n. Note that for M-ary modulation, there are $M-1$ LLRs $Q_{n\to m}(1), Q_{n\to m}(2), \ldots, Q_{n\to m}(M-1)$ need to be obtained from variable node n to check node m, and $Q_{n\to m}(0) = 0$ by (2) and (4). Then we go back to the first step for the next iteration.

3) Summing Up and Decision: After iterating the above two steps several times, the accumulated LLRs for variable node n are

$$\Lambda_n(i) = \sum_{j\in\mathcal{M}(n)} R_{j\to n}(i), \qquad i = 0, 1, \ldots, M-1. \tag{5}$$

By comparing these LLR sums, an estimate of the transmitted symbol sequence $x[n]$ can be made by

$$\hat{x}[n] = \psi(\arg\max_i \Lambda_n[i]), \qquad i = 0, 1, \ldots, M-1.$$

IV. Architecture of the Symbol Detector

A. Overall Architecture of the Proposed Symbol Detector

In this section, we present a low-complexity high-throughput architecture of the proposed symbol detector. The overall system architecture of the proposed sparse channel detector is shown in Fig. 3. The LMS block runs the adaptive algorithm to obtain the PRE coefficients $\mathbf{f}$ and TIR coefficients $\mathbf{g}$. Since the BP algorithm processes symbol detection frame by frame, a serial-to-parallel (S/P) converter is placed before the BP block to buffer N symbols where N is the frame length.

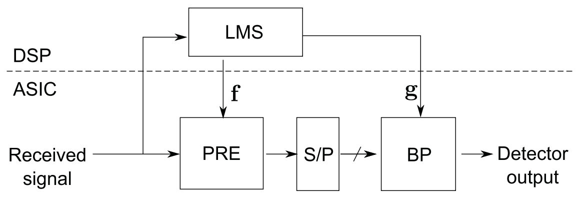


Fig. 3. Architecture of the proposed symbol detector.

We further assume that the coherence time of the channel is sufficiently large so that the channel can be considered static during each frame of N symbol periods. Consequently, the LMS algorithm needs to be applied only once per frame. We use a digital signal processor (DSP) to process the LMS update calculation for several reasons. Since the PRE length L_f can be very long, an application-specific integrated circuit (ASIC) implementation would consume excessive circuit area. A DSP is also more accurate for recursive algorithms if floating-point is used. In addition, the LMS adaptation is performed only once per block of training sequence and a regular DSP or embedded processor is sufficient to handle the computation. Thus, a DSP provides an efficient solution for LMS block as in Fig. 3.

For high data rate application, we suggest the implementation of both PRE and BP algorithms in a field-programmable gate array (FPGA) or a customized ASIC. Since the length of the PRE is usually large, a circuit implementation can exploit pipelining and parallelism for the large filter. As the BP detection algorithm is very complex, a customized circuit implementation is the best solution to achieve high throughput. The architectures of the PRE and BP blocks are discussed in the following subsections.

B. PRE as a Folded FIR Filter

The folded FIR filter architecture is selected as an efficient implementation of the PRE block. Since the BP block is the most complex and limits the overall detector throughput, the PRE is designed such that the filter output matches the input rate requirement of the BP block. For each frame, the BP detector takes N samples and processes for n_{it} iterations, before outputting the detected symbols. Considering the pipelined layer processing architecture, which will be discussed in Section IV-C, applied to the BP detector, it takes $N \times n_{\text{it}}$ cycles to process one frame of received symbols. The function of the S/P block is to buffer the next frame while the current frame is being processed, thus the PRE block needs to output only one sample per n_{it} clock cycles. For area efficiency, we implement a partly serial FIR filter using the folding technique [22]. The folding factor of the FIR filter is n_{it}, that is, one multiplier-accumulator (MAC) is allocated for n_{it} filter coefficients. Hence, the number of MACs used in the PRE is $\lceil L_f/n_{\text{it}} \rceil$. The block diagram of the folded filter design is given in Fig. 4.

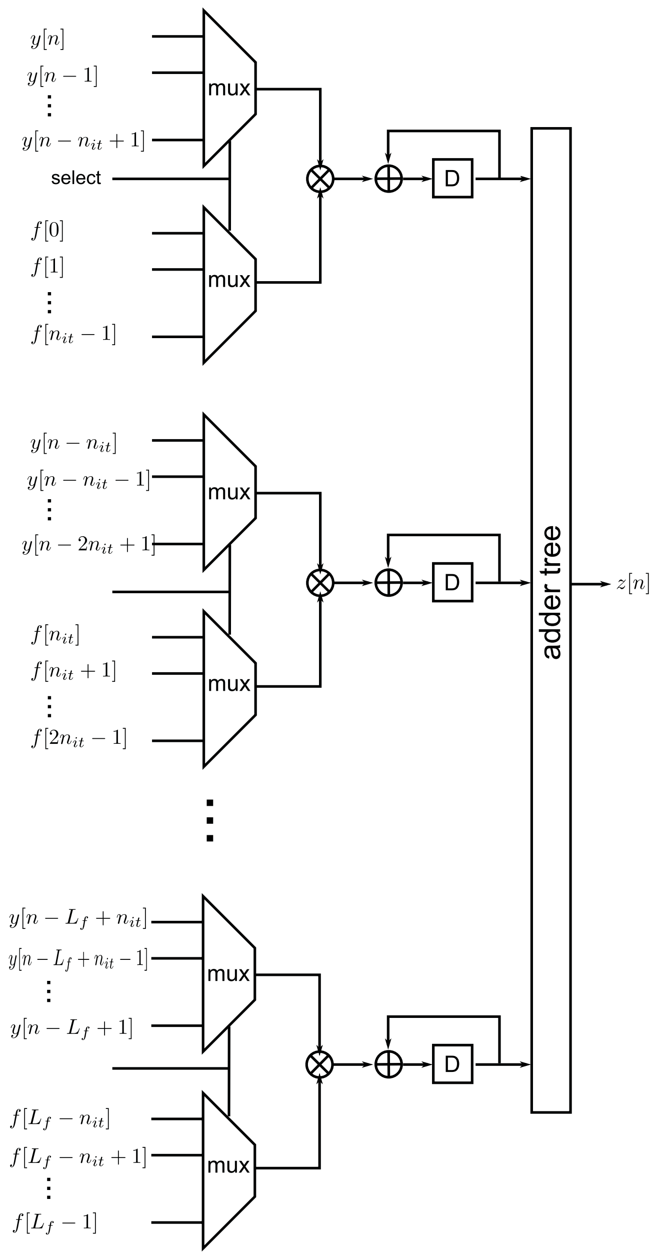


Fig. 4. Block diagram of the PRE folded filter design.

Given the symbol detector running at frequency f and the number of BP iterations n_{it}, we can estimate a throughput of f/n_{it} symbol per second. For M-ary modulation, the bit rate is

$$\text{Throughput (bps)} = \left(\frac{f}{n_{\mathrm{it}}}\right)\log_2 M. \tag{6}$$

C. Pipelined Architecture for Layer Processing

Prior to the discussion of the hardware architecture for the BP detector, we first give the Tœplitz channel matrix which shows dependency between the input and output. As an example, Fig. 5 shows the matrix for a channel with only three nonzero coefficients. The channel matrix is similar to parity-check matrix in quasicyclic LDPC codes. Each check node corresponds to a column index, and its related variable nodes correspond to the row indices. The architecture design for a BP detector can be referenced to the existing LDPC decoder design [13]–[17]. Here we adopt a sequential BP algorithm, which is performed in a way that each check node and its connected variable nodes are treated as a layer. Within each layer processing, we update the LLRs from the check node (e.g., check node m) to its neighboring variable nodes $\mathcal{N}(m)$, and then update the accumulated LLRs associated to these variable nodes. These updated messages are then used in the next layer processing. As an illustration, the sequential BP processing is shown in Fig. 6. Previous studies [23] also have shown that layered BP algorithm improves the convergence speed by reducing the number of iterations.

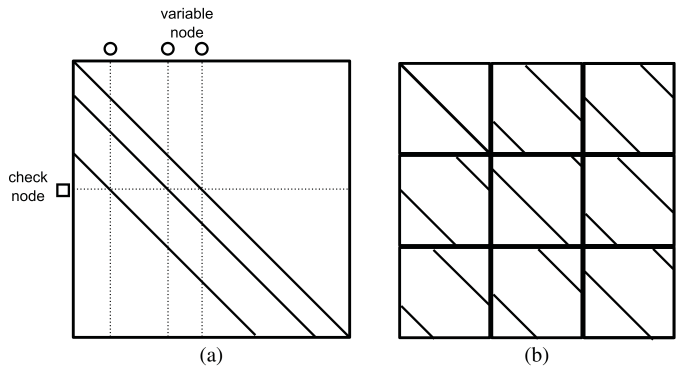


Fig. 5. (a) Tœplitz channel matrix versus (b) parity-check matrix.

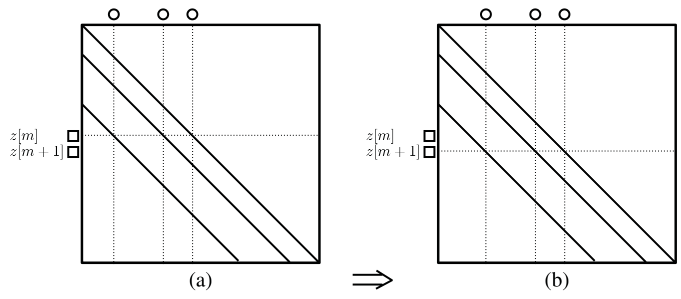


Fig. 6. Illustration of sequential BP processing. (a) Current layer. (b) Next layer.

However, there are major differences between the BP detection algorithm and LDPC decoding algorithm. The structure of detector channel matrix is time-varying, while LDPC parity-check matrix is usually fixed. In LDPC decoders, the structured property of the parity-check matrix can be decomposed into cyclic permutation submatrices, which allows for tradeoff between hardware complexity and decoding throughput using the partially parallel architecture [15], [17]. In contrast, the parallelism is difficult to exploit for the BP detector since the channel matrices are different from frame to frame. If multiple layers are processed simultaneously, it is possible that one variable node will receive the updated messages from different check nodes, and memory writing conflict will occur.

Another major difference between the BP detection algorithm and LDPC decoding algorithm is in the calculation of CTV messages. The calculation for the BP detector operates in the Euclidean space, while for LDPC decoding it operates in GF(2). Therefore, CTV calculations are much more complicated in a BP detector. Recalling that the number of nonzero coefficients is D_g and the modulation order is M, there are D_g elements in $\mathbf{x}(m)$ and hence M^{D_g} instances of

$$\left\{\frac{-\left|z[m]-\mathbf{g}'^T\mathbf{x}(m)\right|^2}{2\sigma^2}+\sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j]))\right\}$$

need to be calculated in order to obtain the LLRs passing from a check node to its connected variable nodes. If each instance requires three multiplications, there are $3M^{D_g}$ multiplications required for one CTV LLR calculation. Given that each layer updates the messages sequentially, computing resource can be reused such that the total number of multipliers is

$$n_{\text{mult}} = 3M^{D_g}. \tag{7}$$

In a typical case, e.g., $D_g = 3$, and $M = 4$ (QPSK), we need to allocate $3 \times 4^3 = 192$ multipliers to calculate the CTV LLRs. For the practical implementation of a BP detector, the parameters M and D_g should be reasonably small. Since the number of multipliers is exponential with respect to the constellation size M, the BP detector approach is typically suited for lower order quadrature amplitude modulation (QAM) systems, such as binary phase-shift keying (BPSK) and QPSK modulations.

The problem in the sequential BP algorithm is that each layer processing takes a long time to execute due to the computational complexity according to (3). The long processing latency for each layer results in a very low throughput rate. The bottleneck can be overcome by pipelining the layer processing unit. For layered processing scheme, the messages on the mth layer are updated first, and messages on the $(m+1)$th layer are processed next. The two successive layer processings can be pipelined without problem as there is no dependency. Note that the independence between layers can be easily satisfied for sparse channels since the nonzero coefficients are often widely separated. It is also observed from the channel matrix that the maximum depth of the pipeline is equal to the shortest distance between the locations of any two adjacent nonzero coefficients.

D. Cache-Memory Architecture for the BP Detector

The overall architecture of the BP detector is shown in Fig. 7. The BP detector mainly consists of a layer processing unit (LPU) responsible for layer message update, the LLR sum memory and CTV message memory which are used to store the BP messages, a cache which is a temporary storage for the LLRs of the nodes currently being processed, and a decision unit for estimating the input symbols. Note that control signals on each blocks are omitted for clarity.

The LPU is the core computing unit executing message updating expressed in (3)–(5). We note that the VTC messages Q used in (3) can be obtained by LLR sum Λ and CTV messages R

$$Q_{n\to m}(i) = \Lambda_n(i) - R_{m\to n}(i), \qquad i = 0, 1, \ldots, M-1$$

according to (4) and (5). In order to minimize the hardware cost, we only store and update Λ instead of Q since the memory for Q is about D_g times as large as the memory

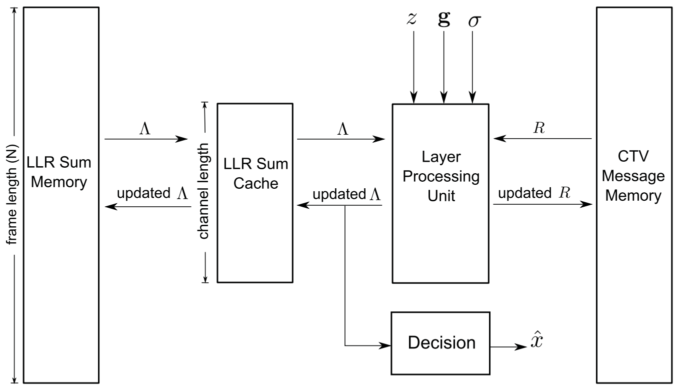


Fig. 7. Overall architecture of the BP detector.

for Λ. In each layer processing, the LPU first calculates the VTC messages Q needed in (3), and then updates the CTV messages R and LLR sums Λ. The architecture of LPU is given in Fig. 8. The calculation of

$$\max\left\{\frac{-\left|z[m]-\mathbf{g}'^T\mathbf{x}(m)\right|^2}{2\sigma^2}+\sum_{j\in\mathcal{N}(m)\backslash n} Q_{j\to m}(\psi^{-1}(x[j]))\right\}$$

$\forall\mathbf{x}(m)$: $x[n] = a_i$ is carried out in M square blocks for $i = 0, 1, \ldots, M-1$, respectively. The architecture for each square block is shown in the dashed box. Since there are M^{D_g-1} different cases of $\mathbf{x}(m)$, M^{D_g-1} branches are implemented for each case. A pipeline architecture is implemented in the LPU design such that it can take a new signal input every clock cycle.

The updated R from LPU are stored back in the CTV message memory, and they will be fetched from the memory in the next iteration. There are $(M-1)$ LLRs for each connection, and thus $(M-1)ND_g$ LLRs for the entire factor graph. The size of CTV message memory is $(M-1)ND_g \times q_R$, where q_R is the word length of the CTV LLR.

The cache is a register bank containing only the accumulated LLRs of L_h variable nodes being processed. This is similar to a "sliding window" that only needs to fetch one new LLR from the Λ memory when the LPU moves to process the next layer. The updated Λ from the LPU consists of D_g sets of LLRs corresponding to D_g connected variable nodes, which is illustrated in Fig. 6. The updated Λ may be used for the following layer processing in a short time. For efficient memory operations, they are written back to the cache to replace the old values. Only the Λ related to the leftmost variable node will be used in the next iteration, and they are stored back in the Λ memory.

The LLR sum memory is used to store the content of Λ. There are $(M-1)$ LLRs for each variable nodes. Thus the size of the LLR sum memory is $(M-1)N \times q_\Lambda$ where q_Λ is the word length of the LLR sum.

Particularly, if the number of iterations is reached, the updated Λ from the LPU do not need to be stored back to the cache or memory. They will be delivered directly to the decision block for symbol decision and output.

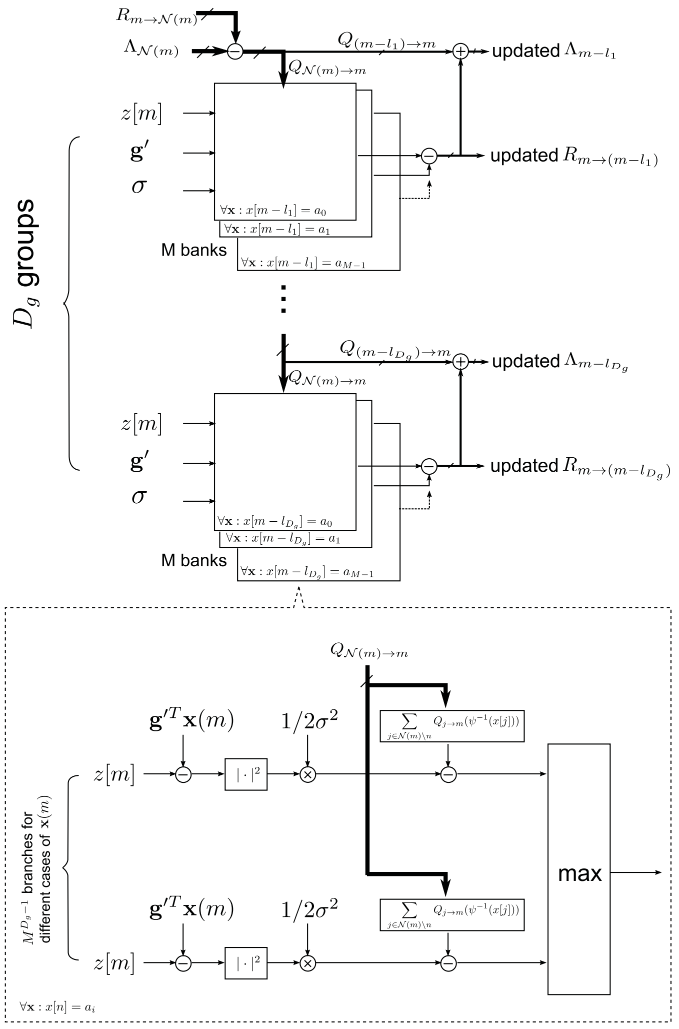

$R_{m\to\mathcal{N}(m)}$
$\Lambda_{\mathcal{N}(m)}$
$Q_{(m-l_1)\to m}$
updated Λ_{m-l_1}
$Q_{\mathcal{N}(m)\to m}$
$z[m]$
$\mathbf{g}'$
σ
updated $R_{m\to(m-l_1)}$
D_g groups
$\forall \mathbf{x}: x[m-l_1]=a_0$
$\forall \mathbf{x}: x[m-l_1]=a_1$
M banks
$\forall \mathbf{x}: x[m-l_1]=a_{M-1}$
$Q_{(m-l_{D_g})\to m}$
updated $\Lambda_{m-l_{D_g}}$
updated $R_{m\to(m-l_{D_g})}$
$\forall \mathbf{x}: x[m-l_{D_g}]=a_0$
$\forall \mathbf{x}: x[m-l_{D_g}]=a_1$
$\forall \mathbf{x}: x[m-l_{D_g}]=a_{M-1}$
$\mathbf{g}'^T\mathbf{x}(m)$
$1/2\sigma^2$
$\sum_{j\in\mathcal{N}(m)\setminus n} Q_{j\to m}(\psi^{-1}(x[j]))$
$|\cdot|^2$
max
M^{D_g-1} branches for different cases of $\mathbf{x}(m)$
$\forall \mathbf{x}: x[n]=a_i$

Fig. 8. Detailed architecture of the layer processing unit.

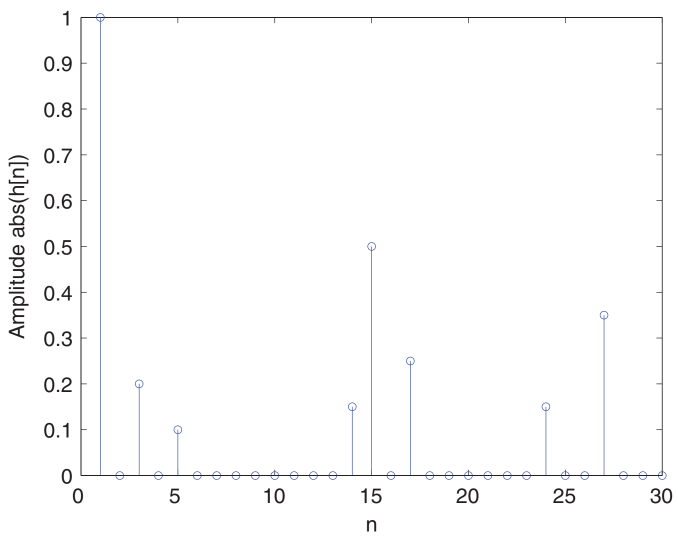


Fig. 9. Sparse channel coefficients for the experimental setup.

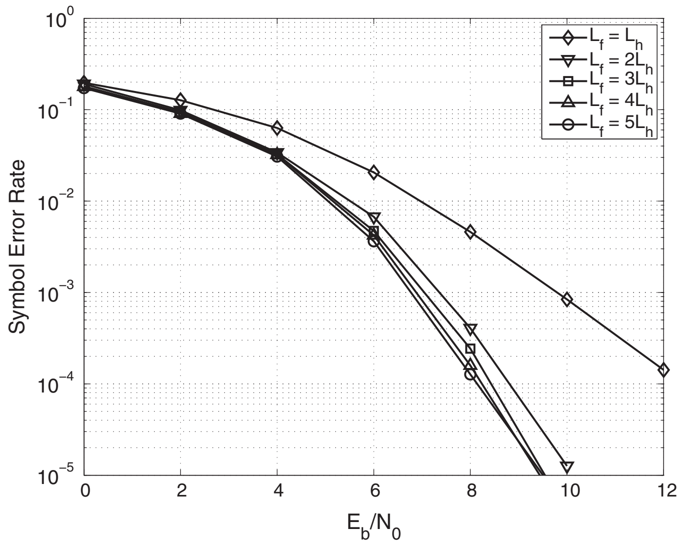


Fig. 10. SER performance for different lengths of PRE L_f.

E. Interconnect Network

The interconnect network between the memory units and the LPU is determined by the channel matrix. Since the structure of channel matrix becomes trivial after the PRE, memory access control and interconnection network in the BP detector are much simpler than the LDPC decoder. Given the TIR $\mathbf{g}' = [g[l_1], g[l_2], \ldots, g[l_{D_g}]]^T$, the mth layer is associated with the $(m-l_1)$th, $(m-l_2)$th, $\ldots$, and $(m-l_{D_g})$th variable nodes, i.e., $\mathcal{N}(m) = \{m-l_1, m-l_2, \ldots, m-l_{D_g}\}$. When the sample $z[m]$ is in process in the LPU, $R_{m\to(m-l_1)}, R_{m\to(m-l_2)}, \ldots, R_{m\to(m-l_{D_g})}$ are loaded from the CTV memory, and $\Lambda_{m-l_1}, \Lambda_{m-l_2}, \ldots, \Lambda_{m-l_{D_g}}$ are loaded from the LLR sum cache. For the next layer process, the read/write addresses for the memories are updated automatically by an increment of 1.

In practice, the sparse channel $\mathbf{h}$ may vary from time to time, and so do its corresponding TIR $\mathbf{g}$ in both values and locations of coefficients. The BP detector can easily adapt to time-varying channels just by changing the coefficients in LPUs and write/read addresses of the memories accordingly.

V. Simulation Results and Error Performance

The performance of the proposed symbol detector is evaluated by simulations in terms of symbol error rate (SER) versus signal-to-noise ratio per bit E_b/N_0. In particular, we consider a sparse channel with $L_h = 30$ taps, which is shown in Fig. 9. The effect of the transmitter pulse shaping and the receiver matched filter has been included in the channel model. QPSK modulation is selected, so that $M = 4$, and the transmitted symbol has unit power. Each time the BP detector processes one frame with 1024 symbols.

Fig. 10 shows the performance for different lengths of PRE L_f, where $n_{\text{it}} = 5$ iterations of the message passing are applied and the number of nonzero coefficients in the TIR is $D_g = 3$. As expected, the performance improves with longer L_f, though the improvement is minimal once the length reaches $L_f = 3L_h$. Recall that the complexity of LMS and PRE will also increase with L_f. In our implementation, the length of the PRE is chosen as $L_f = 100 \approx 3L_h$, and the

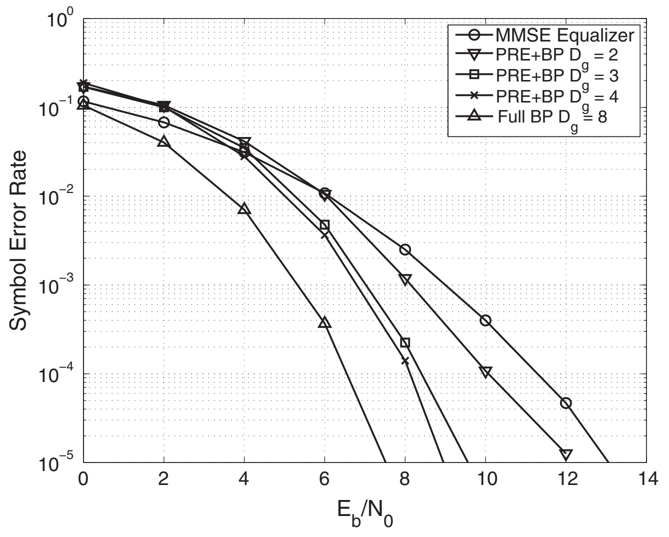


Fig. 11. SER performance for different number of nonzero coefficients in TIR.

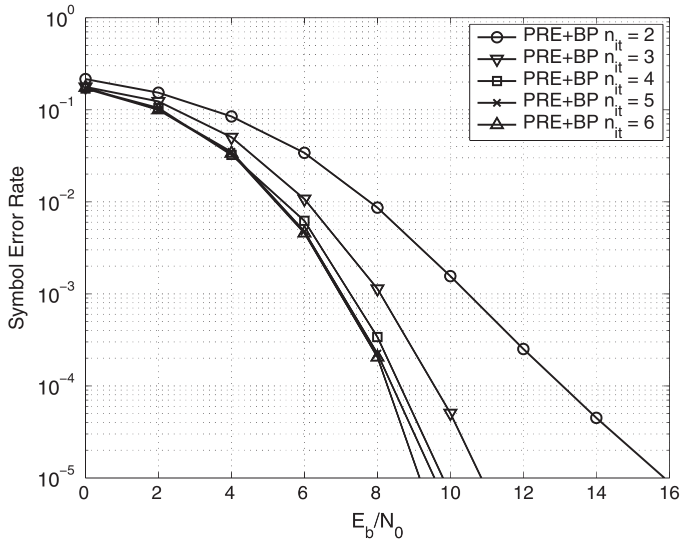


Fig. 12. SER performance for different numbers of BP iterations.

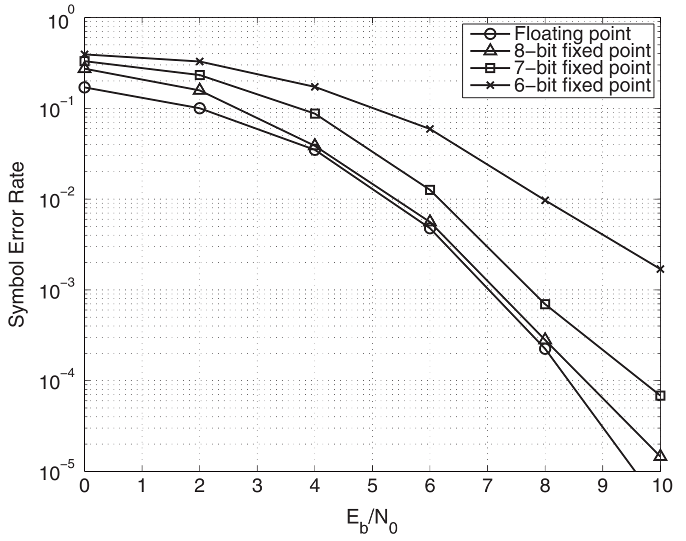


Fig. 13. SER performance for different fixed-point word lengths.

number of MACs consumed by the PRE is $\lceil L_f/n_{\text{it}} \rceil = 20$. Considering the complex filter for QPSK modulation, the number of multipliers is $20 \times 4 = 80$.

Fig. 11 shows the performance for different numbers of nonzero coefficients in TIR, where $n_{\text{it}} = 5$ iterations of the message passing are applied. To evaluate the effectiveness of the symbol detector, we provide the performance of full BP detector with $D_g = 8$ and the linear equalizer with memoryless slicer for comparison. At an SER of 10^{-5}, the detector exhibits performance 5.5 dB better than the linear equalizer when $D_g = 8$, 4 dB better when $D_g = 4$, around 3.5 dB better when $D_g = 3$, and 1 dB better when $D_g = 2$. As discussed in Section IV-C, the complexity of BP detector increases exponentially with D_g. By (7), the multiplier utilization is 48, 192, 768, and 196 608 for $D_g = 2$, $D_g = 3$, $D_g = 4$, and $D_g = 8$, respectively. Thus in our implementation example, we use $D_g = 3$ to ensure a good complexity/performance tradeoff.

Fig. 12 shows the effect of different numbers of BP iterations, where the TIR has $D_g = 3$ nonzero coefficients. We can see that the performance improvement is marginal when n_{it} is larger than 5. Since more iterations means more computing time and thus lower throughput, we suggest using $n_{\text{it}} = 5$ in practice.

VI. Detector Implementation Results

The detector implementation in ASIC includes two main modules—the PRE block and BP block. The same set of parameters as in Section V is applied for hardware implementation: the channel length is 30; the number of nonzero taps in TIR is 3; the number of iterations in BP detection is 5. The pipelined layered processing architecture as described in Section IV-C is adopted. The cache-memory architecture presented in Section IV-D is also implemented to store and transfer messages efficiently in the BP detector.

Fixed-point quantization is evaluated for the received signals and the BP messages. Fig. 13 shows the SERs for different fixed-point quantizations. For 8-bit word length, the fixed-point performance is 0.2 dB away from the floating point result at an SER of 10^{-4}. The SER performance for 7-bit word length is about 1 dB worse than the floating point result. Here, we apply 8-bit word length quantization in this implementation.

The LMS block is implemented on a TI TMS320C6748 DSP, which operates at 456 MHz. The result shows that 10 779 clock cycles are needed for each calculation in Algorithm 1. Assuming that 300 received samples are required for PRE calculation, the PRE and TIR information can be obtained within 7.09 ms.

The proposed symbol detector is implemented with TSMC 90-nm technology and synthesized by a Synopsys Design Compiler. The synthesis result shows that the maximum clock frequency is up to 515 MHz, the core area is 3.162 mm^2, and the estimated dynamic power is 1.096 W. The overall throughput is about 206 Mb/s calculated by (6).

TABLE I
COMPARISON OF HARDWARE RESOURCE UTILIZATION BY DIFFERENT DETECTORS

Detector scheme	PRE/BP detector	ML detector	MMSE equalizer
Multiplier	$4\lceil L_f/n_{\mathrm{it}}\rceil$+$3M^{D_g}$	$2\ M^{L_h}$	$4\ L_f$
Adder	$4\lceil L_f/n_{\mathrm{it}}\rceil + (3+D_g)M^{D_g} + 3(M-1)D_g$	$4\ M^{L_h}$	$4\ L_f$
Comparator	$(M^{D_g-1}-1)MD_g$	$(M-1)M^{L_h-1}$	—
Memory (bits)	$(M-1)ND_g \cdot q_R + (M-1)N \cdot q_\Lambda$	$20L_h \log_2 M \cdot M^{L_h-1}$	—
Throughput (bit/s)	$f \log_2 M/n_{\mathrm{it}}$	$f \log_2 M$	$f \log_2 M$
Latency (cycles)	$((L_f+L_h)/2+N)n_{\mathrm{it}}$	$10\ L_h$	$(L_f+L_h)/2$

TABLE II
COMPARISON OF HARDWARE COMPLEXITY OF DIFFERENT DETECTOR IMPLEMENTATIONS

Detector scheme	PRE/BP detector	MMSE equalizer
Multiplier	272	400
Adder	491	400
Comparator	180	—
Memory (kBytes)	10	—
Throughput (bit/s)	0.4 f	2 f
Latency (cycles)	5445	65

With respect to the implementation complexity, we compare the hardware resource utilization for three schemes in Table I: the proposed detector, the ML detector, and the MMSE equalizer. Complex received signals and complex channel coefficients are considered. For comparison purposes, the ML detector is implemented based on the classic VA. As the implementation of Viterbi-based decoder has been well studied [24]–[27], we simply extend the classic architecture to the Viterbi-based detector by calculating the Euclidean distance instead of the Hamming distance for branch metric calculation. Four banks of memory are used for pipelined output, and the depth for each bank is equal to the traceback length, which is chosen as $5L_h$ in this table. For the MMSE equalizer, we consider a fully parallel FIR filter implementation using multipliers and adders. The latencies for different schemes are compared. The decision delay is estimated as $(L_f + L_h)/2$ for the MMSE equalizer and the PRE. The latency of the ML detector depends on the traceback scheme. We estimate the latency as twice the traceback length, i.e., $10L_h$. Table II shows the actual number of multipliers, adders, comparators, and memory use in our implementation of the proposed detector. Since direct implementation of a Viterbi-based ML detector for channel length of 30 is extremely expensive in computations and not practical, we omit the listing of resource usage of the ML detector in Table II.

Owing to the iterative nature of the BP algorithm, the PRE/BP detector has the drawbacks of longer processing latency and reduced throughput. However, it demonstrates much improved error performance with affordable complexity. For the parallel implementation, the complexity of an MMSE equalizer is about the same as that of the BP detector, but its performance is about 3.5 dB worse at an SER of 10^{-5} than that of the BP detector ($D_g = 3$, $n_{\mathrm{it}} = 5$), which is significant in practical communication systems.

TABLE III
COMPARISON OF THE ASIC AND DSP IMPLEMENTATIONS

	ASIC	DSP
Technology	90 nm	65 nm
Core voltage	1 V	1.3 V
Power	1.096 W	0.66 W
Throughput	206 Mb/s	632.5 kb/s

For comparison, the PRE/BP detector is also implemented on the C6748 DSP processor. For one symbol output, it costs an average of 1442 cycles, which leads to a throughput of 632.5 kb/s only. The comparison of the ASIC and DSP implementations are given in Table III. The technology, core voltage, and power consumption of the DSP are provided in [28] and [29]. It is shown that the ASIC implementation of the PRE/BP detector has significantly higher throughput than for the DSP implementation.

VII. CONCLUSION

For symbol detection under sparse channels, the optimal ML sequence estimator is impractical due to its prohibitively high complexity, while the MMSE linear equalizer is incapable of providing a superior performance. To trade off between the performance and complexity, we investigated the design of a near-optimal detector for sparse channels. This two-stage detector consists of the PRE and BP detector. The PRE first equalizes the sparse channel to a TIR which has a limited number of nonzero channel coefficients. Next the residual ISI is eliminated by the BP detector whose complexity depends merely on the number of nonzero channel coefficients. By exploiting the characteristic of sparse channels, a pipelined layer processing scheme was adopted in the BP detector to achieve high throughput. The PRE/BP detector was implemented with TSMC 90-nm technology for the first time. The implementation result showed a maximum throughput of 206 Mb/s, with an area of 3.162 mm^2, and a total dynamic power of 1.096 W.

REFERENCES

[1] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Marine Technol. Soc. J.*, vol. 42, no. 1, pp. 103–114, 2008.

[2] S. Ariyavisitakul, N. Sollenberger, and L. Greenstein, "Tap-selectable decision-feedback equalization," *IEEE Trans. Commun.*, vol. 45, no. 12, pp. 1497–1500, Dec. 1997.

[3] W. Schreiber, "Advanced television systems for terrestrial broadcasting: Some problems and some proposed solutions," *Proc. IEEE*, vol. 83, no. 6, pp. 958–981, Jun. 1995.

[4] J. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000.

[5] N. C. McGinty, R. A. Kennedy, and P. Hocher, "Parallel trellis Viterbi algorithm for sparse channels," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 143–145, May 1998.

[6] N. Benvenuto and R. Marchesani, "The Viterbi algorithm for sparse channels," *IEEE Trans. Commun.*, vol. 44, no. 3, pp. 287–289, Mar. 1996.

[7] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Trans. Commun.*, vol. 37, no. 5, pp. 428–436, May 1989.

[8] M. V. Eyuboglu and S. U. H. Qureshi, "Reduced-state sequence estimation with set partitioning and decision feedback," *IEEE Trans. Commun.*, vol. 36, no. 1, pp. 13–20, Jan. 1988.

[9] M. N. Kaynak, T. M. Duman, and E. M. Kurtas, "Belief propagation over frequency selective fading channels," in *Proc. IEEE Veh. Technol. Conf.*, Sep. 2004, pp. 1367–1371.

[10] G. Colavolpe and G. Germi, "On the application of factor graphs and the sum-product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818–825, May 2005.

[11] S. Roy, T. M. Duman, and V. K. McDonald, "Error rate improvement in underwater mimo communications using sparse partial response equalization," *IEEE J. Ocean Eng.*, vol. 34, no. 2, pp. 181–201, Apr. 2009.

[12] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[13] X. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2. 2001, pp. 1036–1038.

[14] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[15] T. Zhang and K. Parhi, "VLSI implementation-oriented (3, k)-regular low-density parity-check codes," in *Proc. IEEE Workshop Signal Process. Syst.*, May 2001, pp. 25–36.

[16] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *Proc. IEEE Global Telecommun. Conf.*, vol. 5. Jan. 2001, pp. 3019–3024.

[17] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 1, pp. 104–114, Jan. 2007.

[18] A. Gomaa and N. Al-Dhahir, "A new design framework for sparse FIR MIMO equalizers," *IEEE Trans. Commun.*, vol. 59, no. 8, pp. 2132–2140, Aug. 2011.

[19] R. Machado, A. Klein, and R. Martin, "Sparsening filter design for iterative soft-input soft-output detectors," *EURASIP J. Wireless Commun. Netw.*, 2012, doi: 10.1186/1687-1499-2012-72.

[20] D. Falconer and F. Magee, "Adaptive channel memory truncation for maximum likelihood sequence estimation," *Bell Syst. Tech. J.*, vol. 52, no. 9, pp. 1541–1562, 1973.

[21] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2. Jun. 1995, pp. 1009–1013.

[22] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[23] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2004, pp. 107–112.

[24] R. Cypher and C. Shung, "Generalized trace-back techniques for survivor memory management in the Viterbi algorithm," *J. VLSI Signal Process.*, vol. 5, no. 1, pp. 85–94, Jan. 1993.

[25] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.

[26] G. Feygin and P. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 425–429, Mar. 1993.

[27] Y. Gang, A. T. Erdogan, and T. Arslan, "An efficient pre-traceback architecture for the Viterbi decoder targeting wireless communication applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 9, pp. 1918–1927, Sep. 2006.

[28] *TMS320C6748 Fixed/Floating-Point DSP Data Sheet*, Texas Instruments Inc., Dallas, TX, 2011.

[29] C. Lam, "TMS320C6748/46/42 power consumption summary," Texas Instruments, Dallas, TX, Appl. Rep. SPRABF9, 2010.

Yanjie Peng received the B.S. degree in electronic science and technology from the Huazhong University of Science and Technology, Wuhan, China, and the M.S. degree in microelectronics from Fudan University, Shanghai, China, in 2005 and 2008, respectively. He has been pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, since August 2008.

His current research interests include circuits and systems design for wireless communication and signal processing.

Xinming Huang (M'01–SM'09) received the Ph.D. degree in electrical engineering from Virginia Tech, Blacksburg, in 2001.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA. He was a Technical Staff Member with the Wireless Advanced Technology Laboratory, Bell Laboratories, Lucent Technologies, Murray Hill, NJ, from 2001 to 2003. His current research interests include circuits and systems, with emphasis on reconfigurable computing, wireless communication, and embedded systems.

Dr. Huang was a recipient of the Central Bell Laboratory Annual Excellence and Teamwork Award in 2002, the IBM Faculty Fellowship Award in 2004, and the DARPA Young Faculty Award in 2007.

Andrew G. Klein (S'95–M'98) received the B.S. degree from Cornell University, Ithaca, NY, and the M.S. degree from the University of California, Berkeley, and the Ph.D. degree from Cornell University in 2006, all in electrical engineering.

He has been with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute (WPI), Worcester, MA, since 2007, where he is currently an Assistant Professor. He was a Post-Doctoral Researcher with Supélec/LSS, Paris, France, from 2006 to 2007.

Kai Zhang received the B.E. and M.S. degrees from Xi'an Jiaotong University, Xi'an, China, and the Ph.D. degree in electrical and computer engineering from Worcester Polytechnic Institute, Worcester, MA, in 2012.

He is currently a Design Engineer with Analog Devices, Inc., Norwood, MA. His current research interests include VLSI design for error correction codes and wireless communication.