Turbo Equalization with Coarse Quantization using the Information Bottleneck Method

Philipp Mohr, Jasper Brüggmann, Gerhard Bauch
Institute of Communications
Hamburg University of Technology
Hamburg, Germany
{philipp.mohr; jasper.brueggmann; bauch}@tuhh.de

***Abstract*—This paper proposes a turbo equalizer for intersymbol interference channels (ISI) that uses coarsely quantized messages across all receiver components. Lookup tables (LUTs) carry out compression operations designed with the information bottleneck method aiming to maximize relevant mutual information. The turbo setup consists of an equalizer and a decoder that provide extrinsic information to each other over multiple turbo iterations. We develop simplified LUT structures to incorporate the decoder feedback in the equalizer with significantly reduced complexity. The proposed receiver is optimized for selected ISI channels. A conceptual hardware implementation is developed to compare the area efficiency and error correction performance. A thorough analysis reveals that LUT-based configurations with very coarse quantization can achieve higher area efficiency than conventional equalizers. Moreover, the proposed turbo setups can outperform the respective non-turbo setups regarding area efficiency and error correction capability.**

I. Introduction

Transmission of digital data streams over channels with memory can cause intersymbol interference (ISI) in the received signal. Turbo equalization is a well-known technique where an equalizer and a decoder cooperate to mitigate interference with near-capacity error correction performance [1]. However, high implementation complexity arises when using optimal equalization and decoding algorithms [2].

In [3] a new type of equalization has been proposed that replaces computationally intensive parts of an arithmetic-based forward-backward equalization algorithm with lookup tables (LUTs) designed using the information bottleneck (IB) method. The IB method is a clustering framework for designing compression operations that maximize mutual information [4]. The LUTs realize compression operations to reduce the space-complexity of discrete messages exchanged within the IB equalization algorithm.

This work combines the turbo principle with the IB equalizers as depicted in Fig. 1. By introducing new LUT decompositions, the turbo feedback from the decoder is integrated into the IB equalization algorithm with low complexity. Compared to the non-turbo setup, significant performance improvements are observed for a magnetic recording and faster-than-Nyquist channel. Another contribution is the low-level hardware complexity comparison of conventionally quantized equalizers and IB equalizers with turbo and non-turbo configurations. The computational and memory complexity of all setups is compared using logic gate representations. It is shown that configurations with very coarse quantization can achieve higher area efficiency than the conventional approaches.

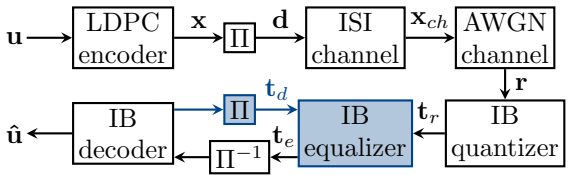


Fig. 1. Iterative equalization and decoding designed using the IB method.

II. Design of Coarsely Quantized Equalization with Decoder Feedback

A sequence of symbols $\mathbf{d}=[..,d_{-1},d_0,d_1,..]$ with mapping alphabet $d_k\in\mathcal{D}$ is transmitted over an ISI channel with taps $[h_0,h_1,..,h_L]\in\mathbb{R}^{L+1}$. The channel state s_k models the symbols in memory $[d_{k-L},..,d_{k-1}]$. The receiver observes the sequence $\mathbf{r}=[..,r_{-1},r_0,r_1,..]$ with $r_k=x_k^{ch}+n_k$ where $x_k^{ch}=\sum_{l=0}^{L}h_l d_{k-l}$ and n_k is additive white Gaussian noise (AWGN) with variance $N_0/2$.

Using the BCJR algorithm [2], optimal soft-output equalization for symbol d_k is achieved by computing

$$p(d_k|\mathbf{r}) \propto \sum_s \alpha(s|\mathbf{r}_-)\gamma(s,d_k|r_k)\beta(s'|\mathbf{r}'_+). \tag{1}$$

with the current state $s{:=}s_k$, the next state $s'{:=}s_{k+1}$, the current received symbol r_k, all previous received symbols $\boldsymbol{r}_-{:=}[r_l{:}l{<}k]$ and all following received symbols $\boldsymbol{r}'_+{:=}[r_l{:}l{>}k]$. The branch metric $\gamma(s,d_k|r_k)$ is equal to

$$p(r_k|s,d_k)p(d_k) = \frac{1}{\sqrt{\pi N_0}}\exp\left(\frac{-|r_k-x_k^{ch}|^2}{N_0}\right)p(d_k). \tag{2}$$

The probability $p(d_k)$ can be considered as extrinsic information from the decoder. In the first equalizer run this probability corresponds to the prior probabilities. The forward and backward metrics, $\alpha(s|\mathbf{r}_-)$ and $\beta(s'|\mathbf{r}'_+)$ in (1), are computed recursively with $\boldsymbol{r}'_-{:=}[r_l{:}l{\leq}k]$ and $\boldsymbol{r}_+{:=}[r_l{:}l{\geq}k]$ according to

$$\alpha(s'|\mathbf{r}'_-) = \sum_{d_{k-L}}\gamma(s,d_k|r_k)\alpha(s|\mathbf{r}_-) \text{ and} \tag{3}$$

$$\beta(s|\mathbf{r}_+) = \sum_{d_k}\gamma(s,d_k|r_k)\beta(s'|\mathbf{r}'_+). \tag{4}$$

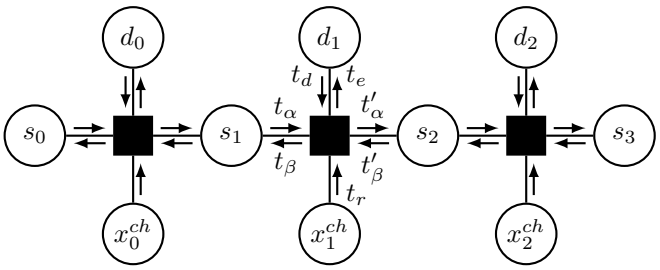


Fig. 2. Factor graph with compressed exchanged messages.

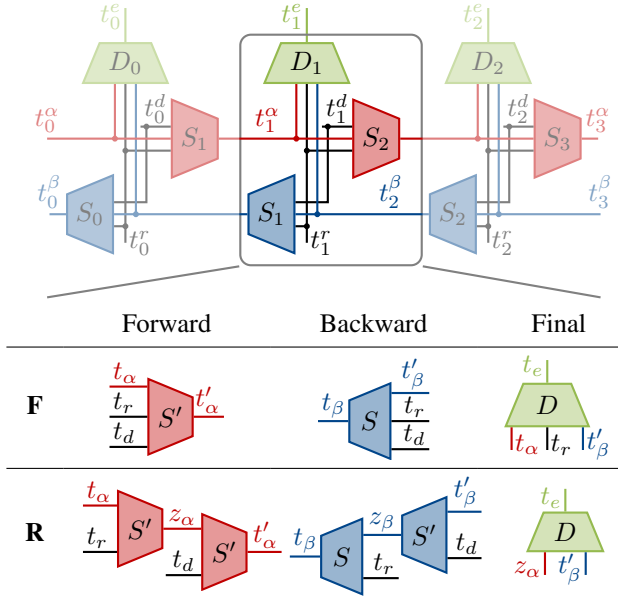


Fig. 3. IB graph and LUT decompositions.

A. Forward and Backward Updates with Coarse Quantization

In [3] it has been proposed to replace the high-resolution vector metrics $\alpha(s|\mathbf{r}_-)$ and $\beta(s'|\mathbf{r}'_+)$ with finite alphabet messages $t_\alpha \in \mathcal{T}^{w_\alpha}$ and $t'_\beta \in \mathcal{T}^{w_\beta}$ using $\mathcal{T}^w = \{1, \ldots, 2^w\}$ with w bits. Furthermore, the channel message, extrinsic feedback message from the decoder and message from the equalizer are assumed to be quantized as $t_r \in \mathcal{T}^{w_r}$, $t_d \in \mathcal{T}^{w_d}$ and $t_e \in \mathcal{T}^{w_e}$. Hence, the computations in (3) and (4) can be replaced with lookup operations defined by the deterministic conditional probabilities $p(t'_\alpha|t_\alpha, t_r, t_d)$ and $p(t_\beta|t'_\beta, t_r, t_d)$. The factor graph with quantized exchanged messages is shown in Fig. 2.

All compressed messages t_α, t'_β, t_r and t_d can be modeled as realizations of the random variables T_α, T'_β, T_r and T_d, respectively. Similarly, the symbol d_k, current state s and next state s' are modeled as realizations of the random variables D, S and S', respectively.

The forward lookup shall maximize the preserved mutual information between S' and T'_α according to

$$\max_{p(t'_\alpha|t_\alpha,t_r,t_d)} I(S';T'_\alpha). \tag{5}$$

The objective (5) can be solved using an IB algorithm where $[T_\alpha, T_r, T_d]$, S' and T'_α are classified as the observed variable Y, relevant variable X and compressed variable T, respectively [3]. The required input joint distribution $p(x, y)$ for the design is obtained as

$$p(s', t_\alpha, t_r, t_d) = \sum_{d_{k-L}} p(t_r|s, d_k)p(d_k, t_d)p(s, t_\alpha). \tag{6}$$

In Fig. 3 the factor graph of Fig. 2 is depicted as an IB graph. The trapezoidal nodes reflect an IB setup where the inputs are the observed variables, the label inside of the node is the relevant variable and the output is the compressed variable [3].

The design procedure must keep track of the underlying probability distribution

$$p(s', t'_\alpha) = \sum_{t_\alpha,t_r,t_d} p(t'_\alpha|t_\alpha, t_r, t_d)p(s', t_\alpha, t_r, t_d). \tag{7}$$

The forward LUTs are designed over multiple recursions. In the first design step, $p(s) = 1/|\mathcal{D}|^L$ and $p(t_\alpha|s) = 1/2^{w_\alpha}$. After a certain amount of recursions (around 10 to 50), the mutual information $I(S';T'_\alpha)$ converges. Then, a post processing step, proposed as the *static design phase* in [3], finds a single LUT that can be used in all recursions.

Similarly, the backward mapping $p(t_\beta|t'_\beta, t_r, t_d)$ is designed using the joint distribution

$$p(s, t'_\beta, t_r, t_d) = \sum_{d_k} p(t_r|s, d_k)p(d_k, t_d)p(s', t'_\beta). \tag{8}$$

The backward output joint distribution yields

$$p(s, t_\beta) = \sum_{t_\beta,t_r,t_d} p(t_\beta|t'_\beta, t_r, t_d)p(s, t'_\beta, t_r, t_d). \tag{9}$$

After the static design phase for forward and backward LUTs, a single final mapping $p(t_e|t'_\beta, t_r, t_\alpha)$ is designed using

$$p(d_k, t_\alpha, t_r, t'_\beta) = \sum_{d_k} p(t'_\beta|s')p(s, d_k, t_r)p(t_\alpha|s) \tag{10}$$

where $p(t_\alpha|s)$ and $p(t'_\beta|s')$ are obtained after a certain amount of recursions with the static LUTs. Then, the output joint distribution assumed for the decoder design yields

$$p(d_k, t_e) = \sum_{t'_\beta,t_r,t_\alpha} p(t_e|t'_\beta, t_r, t_\alpha)p(d_k, t_\alpha, t_r, t'_\beta). \tag{11}$$

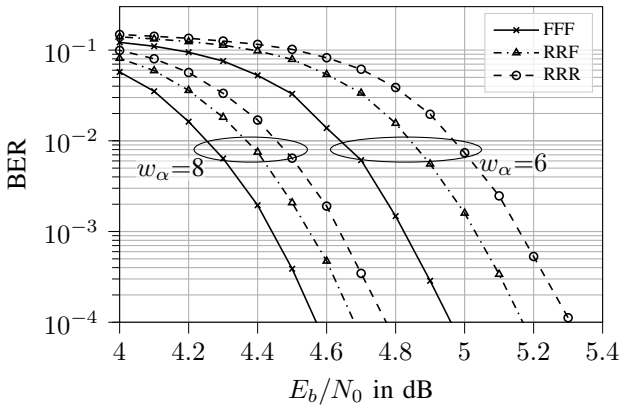


Fig. 4. Performance of simplified structures for forward, backward and final update. Each update can use a full (F) or reduced (R) LUT structure.

TABLE I
NUMBER OF LUT ENTRIES (w_r=5, w_d=3).

	Forward/Backward		Final	
Variant	8 bit	6 bit	8 bit	6 bit
Full (F)	66k	16k	2097k	131k
Reduced (R)	10k	2.5k	66k	4k

B. Decomposition of Three-Input Tables

Fig. 3 proposes simplified structures, labeled reduced (R), that shrink the overall LUT size through concatenation of two-input LUTs. The reduction in size causes a performance loss compared to the three-input LUTs, labeled full (F), investigated in Fig. 4. The simulation results are obtained for a magnetic recording channel with taps $[.5, .5, -.5, -.5]$, BPSK modulation, a rate $1/2$ regular LDPC code used in [3] and 3 turbo iterations each with 10 decoder iterations.

It can be observed that the decomposition of the forward and backward three-input LUT in variant RRF leads to .1 dB and .2 dB loss compared to variant FFF for w_α=w_β=8 bits and w_α=w_β=6 bits, respectively. As in [3], the decomposition of the final LUT in variant RRR causes an additional loss of .08 dB and .13 dB, respectively. Thus, accepting moderate performance loss enables major savings in terms of LUT size as shown in Table I. The decomposition of forward and backward LUTs reduces the number of LUT entries from $2^{w_\alpha+w_r+w_d}$ to $2^{w_\alpha}(2^{w_r}+2^{w_d})$. Decomposing the final LUT leads to a reduction from $2^{w_\alpha+w_r+w_\beta}$ to $2^{w_\alpha+w_\beta}$.

III. PROPOSED VS. CONVENTIONAL EQUALIZATION

Logarithmic probabilities $\bar{p}:=\log p$ are considered for a conventional equalizer. Then, the forward recursion yields [5]

$$\bar{\alpha}(s') = \overset{*}{\max_{d_{k-L}}} \left(\bar{\alpha}(s) + \frac{1}{N_0}\bar{\gamma}(s,d_k) + \bar{p}(d_k|t_d) \right) \qquad (12)$$

In (12), for any two inputs $a, b \in \mathbb{R}^2$, we recursively apply [6]

$$\overset{*}{\max}(a,b) = \max(a,b) + \log(1+e^{-|a-b|}). \qquad (13)$$

The final update computes the LLR for the decoder as

$$\bar{p}(d_k|\mathbf{r}) = \overset{*}{\max_{s}} \left(\bar{\alpha}(s) + \frac{1}{N_0}\bar{\gamma}(s,d_k) + \bar{\beta}(s') \right). \qquad (14)$$

The metric computation $\bar{\gamma}(s,d_k)$ depends on the observation model. The Forney observation model directly uses the channel observations r_k for computing the branch metric [5], [7]

$$\bar{\gamma}(s,d_k) = -|r_k - x^{ch}_{s,d_k}|^2 \quad \text{with } x^{ch}_{s,d_k} = \sum_{l=0}^{L} h_l d_{k-l}. \qquad (15)$$

The Ungerboeck observation model [5], [8] applies a prefilter leading to $r'_k = \sum_\kappa h'_\kappa r_{k-\kappa}$ where $h'_l = h^*_{-l}$. Then,

$$\bar{\gamma}(s,d_k) = 2\mathrm{Re}\{d_k \cdot r'_k\} - \gamma_{s,d_k} \qquad (16)$$

with the function $\gamma_{s,d_k} = g_0|d_k|^2 + \mathrm{Re}\left\{ d^*_k \sum_{l=1}^{L} g_l d_{k-l} \right\}$ where $g_k = \sum_{i=-L}^{L} h_i h^*_{i-k}$. Another variant, termed channel shortening, applies a modified prefilter h' leading to a shortened model g_k with smaller memory L [9]. However, the approach reduces performance and the modified prefilter involves approximately 30 taps leading to additional complexity.

A. Complexity Reduction of The Conventional Equalizer

The Forney model involve an expensive $|.|^2$-operation in (15) which must be computed for every state transition. The multiplication implementation in the Ungerboeck model (16) is very simple with BPSK modulation as $2\mathrm{Re}\{d_k r'_k\} \in \{\pm 2r'_k\}$, also mentioned in [8]. Hence, overall the Forney metric demands higher complexity with $3{\cdot}2^{L+1}$ multiplications per equalizer run compared to the Ungerboeck model that only uses $L{+}1$ multiplications in the prefilter operation. For a magnetic recording channel with the channel taps $[.5, .5, -.5, -.5]$, the prefilter can avoid multiplications completely. When neglecting the correction term in (13) complexity can be saved. Then, e.g., the forward recursion simplifies to

$$\tilde{\alpha}(s') \approx \max_{d_{k-L}} \left(\bar{\alpha}(s) + \bar{\gamma}(s,d_k) \right) + N_0 \bar{p}(d_k|t_d) \qquad (17)$$

where $N_0\bar{p}(d_k|t_d)$ can be implemented with an LUT because the decoder feedback t_d is coarsely quantized. A normalization $\bar{\alpha}(s') = \tilde{\alpha}(s') - \tilde{\alpha}(s'{=}1)$ limits the numeric range.

The channel message r_k can be uniformly quantized with spacing Δ. The spacing Δ is preserved within all equalizer operations. A clipping operation limits the bit width of every entry in the forward metrics $\bar{\alpha}(s)$ to $\bar{w}_\alpha$. The bit width of the vector metrics is $w_\alpha = (2^L{-}1)\bar{w}_\alpha$ as $\bar{\alpha}(s{=}1)$ is always 0.

B. Magnetic Recording Channel

Fig. 5 evaluates the error rate performance for a magnetic recording channel with the same setup as in section II-B using decomposition RRR (cf. Fig. 4). The number of turbo iterations N_{it} is configured as 2,1 or 0. Similar decoder complexity is established across all setups by distributing a total budget of 20 decoder iterations as $(5, 5, 10)$, $(10, 10)$ and (20), respectively. If not mentioned otherwise, all setups use LUT-based decoders with 4 bit for the exchanged messages as in [3]. The decoder uses the existing messages in memory from previous decoder runs.

The non-turbo IB equalizer proposed in [3] is outperformed by up to .55 and .8 dB when using 1 or 2 turbo iterations, respectively. The IB equalizers are configured with 5 bits for the channel message and 9 bits for the state metrics.

A conventional equalizer requires 7 bits for the channel message and 11 bits for each state metric entry (in total 77 bits with 8 states) to maintain similar performance as its non-quantized variant. Although the number of bits is decreased by an order of magnitude, the IB equalizers suffer only from .03 dB, .05 dB and .1 dB performance loss compared to the conventional equalizer at 0, 1 and 2 turbo iterations, respectively. The increasing difference can be explained by the decomposition structure RRR, which is used more frequently according to the number of turbo iterations causing a higher cumulative loss.

The conventional equalizer performance can be improved by .27 dB using the accurate $\overset{*}{\max}$ implementation. Further gains of .1 dB in the configuration 'BCJR&BP dec.' are obtained using a high-resolution belief propagation LDPC decoder instead of a 4-bit LUT decoder.

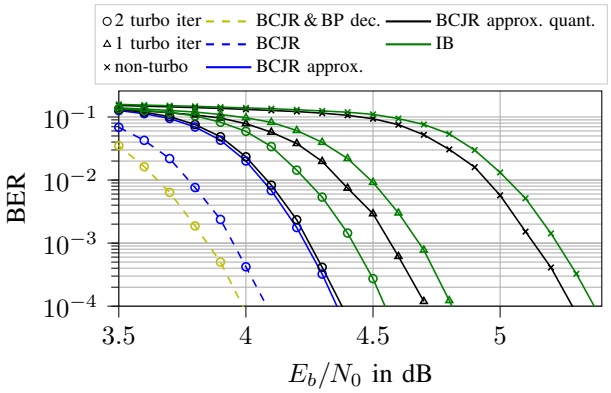


Fig. 5. EPR4 channel: Conventional vs. IB (w_α=9).

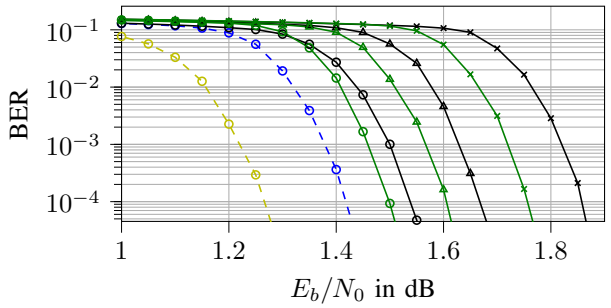


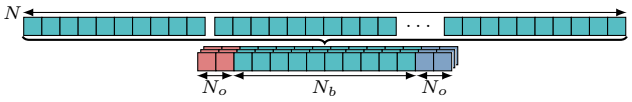
Fig. 6. FTN channel: Conventional vs. IB (w_α=8).

C. Faster-Than-Nyquist Channel

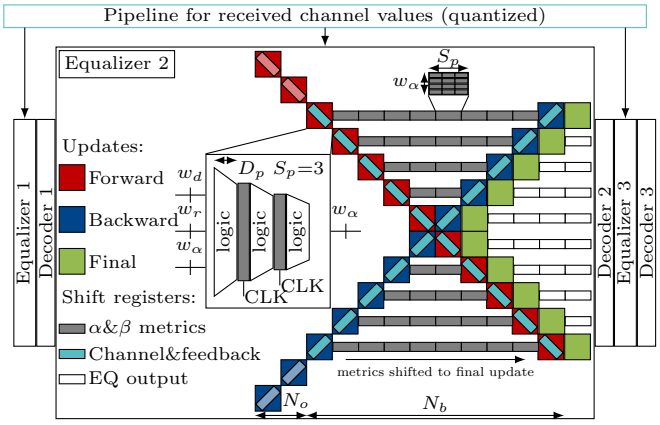
Fig. 6 depicts the error correction performance for a channel with ISI resulting from faster-than-Nyquist (FTN) signaling as in [3]. A 64800-length DVB-S2 LDPC code with rate $1/2$ is used. The number of decoder iterations is distributed as (50), $(25, 25)$ and $(16, 17, 17)$ for 0, 1 and 2 turbo iterations, respectively. As the impulse response is long $\mathbf{h}$=[.8907,.4088,-.1919,.0510,-.0040, .0045,-.0076,.0039,-.0014,.0019,-.0020,.0014], all equalizers perform a shortening approach to reduce the equalizer memory. The conventional equalizers perform channel shortening leading to 4 states (memory 2). The IB equalizers truncate the model after 3 taps. The non-turbo IB equalizer [3] is outperformed by .17 dB and .27 dB when using 1 or 2 turbo iterations, respectively. The gains are smaller than in the previous section III-B as the ISI is less severe. All quantized BCJR equalizers with the approximated $\overset{*}{\max}$ implementation are outperformed by .2 .1, and .07 dB for 0, 1 and 2 turbo iterations.

IV. Hardware Complexity

This section estimates and compares the hardware complexity of the LUT-based and arithmetic-based equalizers. The complexity is typically expressed in terms of area efficiency, i.e., the ratio of throughput to chip area. The forward and backward recursions can cause high delays when equalizing N symbols sequentially. To reduce the delay and to improve throughput, P sub-blocks with length N_b are equalized in parallel as shown in Fig. 7a. Each sub-block involves an initialization phase using N_o overlapping symbols of the adjacent



(a) Division into sub-blocks for increasing parallelism.



(b) 3 unrolled turbo iterations using an X-shaped structure [10].

Fig. 7. Turbo equalizer setup with N_b=10 and N_o=2.

sub-blocks. It can be observed in [3] that an overlap length of N_o=10 is sufficient to avoid performance loss for a magnetic recording channel. A conceptual hardware implementation for 3 unrolled turbo iterations is depicted in Fig. 7b. The forward and backward recursions can be carried out using an X-shaped structure [10]. We divide the logic gates that carry out the update operations into a number of levels N_L. The logic depth per pipeline stage D_p defines the number of logic gate levels between a pair of consecutive pipeline stages. A lower value for the logic depth D_p allows higher clock frequency and better logic utilization at the cost of additional shift registers. We use a small value D_p=8 throughout the analysis to ensure high logic utilization. In this way, it is assumed that all implementations use the same clock frequency. This means that the area efficiency only depends on the chip area.

The chip area is determined by the number of transistors used to realize the update logic, the pipeline stages and the shift registers for storing the state metrics. In this work, interconnects are neglected to avoid extensive hardware synthesis and dependence on a specific hardware platform. Table II lists the assumed transistor count of logic operations.

The overall transistor count for N_{it} turbo iterations is

$$\xi_{eq}^{(N_b,N_o)} = \sum_{i=0}^{N_{it}} \left(\xi_{update}^{(i,N_b)} + \xi_{memory}^{(i,N_b,N_o)} \right) \tag{18}$$

where the update logic and memory logic are normalized by the number of equalizer outputs N_b per clock cycle as

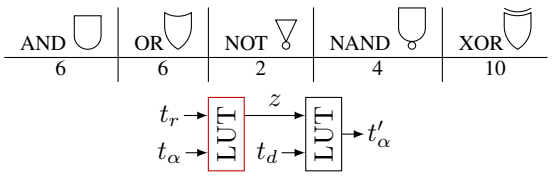
$$\xi_{update}^{(i,N_b,N_o)} = \frac{1}{N_b} \left((N_b+N_o)(\xi_\alpha^{(i)}+\xi_\beta^{(i)}) + N_b \xi_e^{(i)} \right) \text{ and} \tag{19}$$

$$\xi_{memory}^{(i,N_b)} = \frac{1}{N_b} (S_p(w_\alpha^{(i)}+w_\beta^{(i)}) + S_{p,e} w_e) \xi_{DFF} \sum_{k=1}^{N_b/2-1} 2k \,. \tag{20}$$

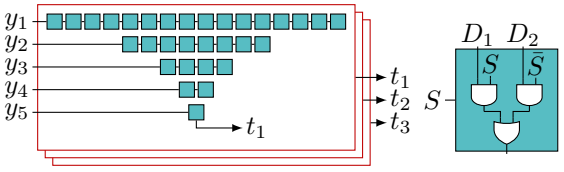
The transistor count for forward, backward and final update (a single square box in Fig. 7b) is ξ_α^i, ξ_β^i and ξ_e^i, respectively,

TABLE II
REQUIRED CMOS TRANSISTORS PER LOGIC GATE [11].

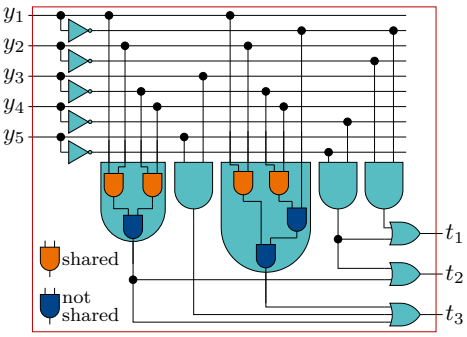
AND	OR	NOT	NAND	XOR
6	6	2	4	10



(a) Decomposition into two LUTs.



(b) LUT with multiplexer implementation.



(c) LUT with two-level and multi-level implementation.

Fig. 8. LUT-based equalizer operations ($w_\alpha = 3, w_r = 2$).

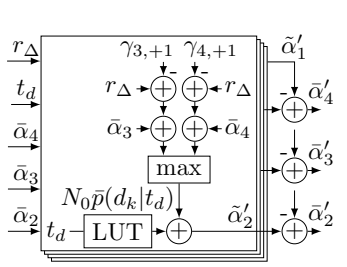
also comprising chip area for the pipeline stages. The memory complexity $\xi_{memory}^{(i,N_b)}$ increases linearly with the number of pipeline stages $S_p{=}\lceil N_L/D_p \rceil$ as each pipelined sub-block requires dedicated shift registers for the metrics. The shift registers consist of multiple D-Flipflops (each 4 NAND, 1 NOT, $\xi_{DFF}{=}18$). The choice of N_b is a compromise between update and memory complexity. We remark that the latency changes with N_b, but is not evaluated further since the area efficiency is our main focus. Thus, in each setup individual optimization of the sub-block length is done according to

$$\min_{N_b} \xi_{eq}^{(N_b, N_o=10)}. \tag{21}$$

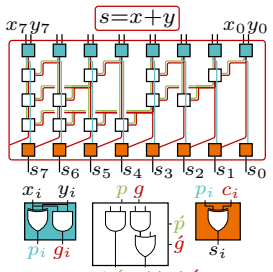
A. Proposed Hardware with Lookup Operations

In the LUT-based equalizer the forward recursion performs a lookup operation with the forward message t_α from the previous recursion step, the quantized channel message t_r and the feedback message t_d from the decoder. As discussed in section II-B, the LUT size is significantly reduced with a decomposition into two LUTs (cf. Fig. 8a).

1) Implementation with Multiplexers: In Fig. 8b, the LUT is implemented as a selection network, generating a w_α-bit output t for the input y using a tree structure. The bit values of y are denoted by y_1 to y_w. The LUT entries LUT_1 to LUT_{2^w} initialize the selection process. In the first stage, y_1 causes each 2:1 multiplexer to forward one of its two inputs from the memory cells. In the next stages, y_2 to y_w further proceed the selection until the output t is generated. Hence, $w_\alpha{+}w_r$ multiplexer stages are required.



(a) Conventional hardware structure of the forward update with 4 states.



(b) Ladner-Fischer adder [12].

Fig. 9. Arithmetic-based equalizer implementation.

The second LUT operates similarly using the w_α-bit intermediate message t and the w_d-bit feedback message t_d from the decoder, requiring another $w_\alpha{+}w_d$ multiplexer stages. However typically, $w_d{<}w_r$. Hence, the first LUT requires most of the complexity.

2) Implementation with Two-Level Logic: The LUT behavior can be represented using logic expressions. In this paper we consider the disjunctive normal form (DNF) generated by constructing a truth table defining a bit-wise input-output relationship. We make use of a logic optimization tool [13] that implements the Espresso algorithm [14] to approach a minimal expression of the DNF. Fig. 8c depicts an exemplary implementation of the minimized DNF using logic AND as well as OR gates.

3) Implementation with Shared Multi-Level Logic: The multi-input AND gates in Fig. 8c, termed nodes, can be realized with several two-input AND gates. Among the nodes, some of the internal operations, indicated by the orange color in Fig. 8c, have the same input signal. To improve efficiency, all those operations can be replaced with a single operation whose output signal is utilized by multiple nodes. We developed an effective algorithm that maximizes utilization of the single operations. In each iteration step, the algorithm selects the AND combination of two signals that maximizes the number of subsequent operations using the combined signal. The algorithm terminates when all inputs are combined.

4) Exploiting channel symmetries: As proposed in [3], channel symmetries can be used to enforce a symmetric design of the LUT. Then, the LUT can be implemented as

$$t = \mathrm{LUT}(y) = \begin{cases} \mathrm{LUT}'(y_{2:w}) & y_1 = 0 \\ \neg\,\mathrm{LUT}'(\neg y_{2:w}) & y_1 = 1 \end{cases} \tag{22}$$

where $y_{2:w}$ denotes the bits y_2 to y_w and $\neg$ is a bit inversion. Halving the LUT size saves 2^{w-1} multiplexers in Fig. 8b and approximately halves the gate count in Fig. 8c. The conversion (22) requires only $w{-}1$ multiplexers at input and output.

B. Comparison to Conventional Equalizer Implementations

Fig. 9a depicts the forward update that implements the behavior of an arithmetic-based log-domain equalizer as in (12) and (14) with simplifications from section III-A. Most of the complexity lies within the adder structure shown in Fig. 9b, also realizing major part of the maximum operation.

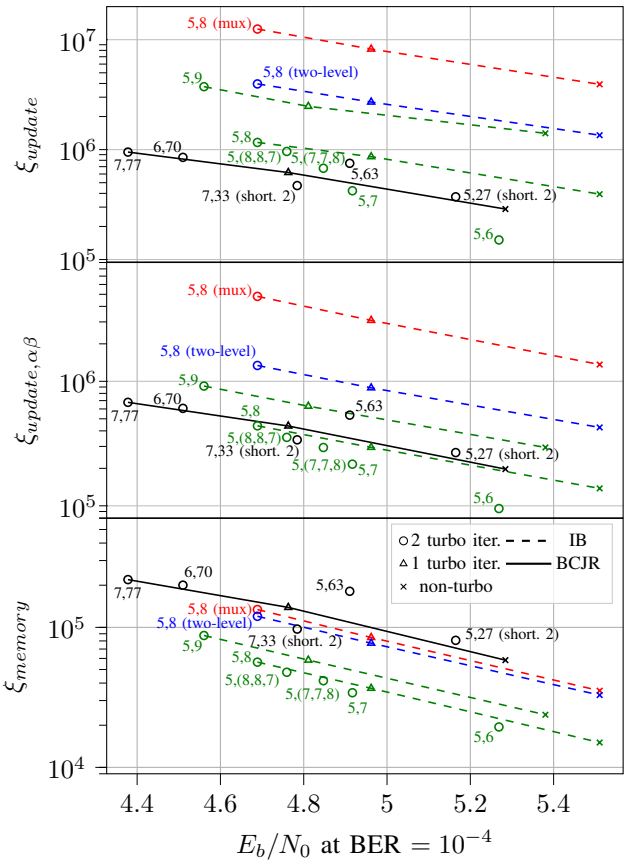


Fig. 10. Complexity (transistor count per equalized symbol) vs. performance.

For comparing complexity and performance, we assume a magnetic recording channel setup (cf. section III-B). In Fig. 10 the number of transistors for various equalizer parts is depicted versus E_b/N_0 at a target bit error rate of 10^{-4}. The configurations are labeled with the number of bits $(w_r, w_{\alpha/\beta})$ used for the channel message and forward/backward metrics. The IB setups (5,(8,8,7)) and (5,(7,7,8)) use different bit widths for the individual turbo iterations and demonstrate another option for trading complexity and performance.

The complexity for the update ξ_{update} reveals that only the IB equalizer with shared logic gates (cf. Fig. 8c) can achieve similar or even lower implementation complexity than the arithmetic-based equalizers. For example, the (5,6) setup with 3 turbo iterations requires half the complexity of the conventional (7,77) setup without turbo iterations at similar performance.

The final LUT is significantly larger than the forward/backward LUT according to table I. Hence, considering only the forward/backward update complexity with $\xi_{update,\alpha\beta}$ even the higher bit width (5,8) setup can achieve lower complexity than the conventional forward/backward update. Thus, a hybrid design with forward/backward LUTs and final arithmetic update would be a viable option.

Finally, the memory complexity ξ_{memory} shows that all IB equalizers require significantly less memory logic as the forward/backward metrics are smaller by a factor of 3 to 13. The memory complexity varies among the configurations as it depends on the sub-block length N_b which is optimized in (21). Reducing the bit width of the conventional equalizers causes severe performance degradation for (6,70) or (5,63) while saving only small amounts of complexity. Instead using channel shortening (7,33,(short. 2)) can offer a better performance-complexity trade-off (neglecting the prefilter in Fig. 10). However, comparing the range of numbers for ξ_{memory} with ξ_{update} clearly shows that the area-efficiency of the forward-backward algorithm is update-bound and not memory-bound. Thus, the major memory savings with the IB equalizer have only small impact on the overall complexity.

V. Conclusions

This paper presented turbo equalizers utilizing two-input LUTs as compression updates to generate coarsely quantized metrics. The information bottleneck method was employed for the LUT design. Significant performance improvements of up to .8 dB were achieved with the turbo over the non-turbo setups. The LUT equalizers reduced the memory requirements for buffering metrics by up to an order of magnitude compared to conventional arithmetic-based equalizers. However, the overall complexity, including the logic for update operations, was only lower under very coarse quantization.

References

[1] M. Tuchler, R. Koetter, and A. Singer, "Turbo equalization: principles and new results," *IEEE Transactions on Communications*, vol. 50, no. 5, pp. 754–767, 2002.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[3] P. Mohr, M. Stark, and G. Bauch, "Information Bottleneck Receivers for ISI Channels," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 3370–3375.

[4] J. Lewandowsky, "The information bottleneck method in communications," Ph.D. dissertation, 2020.

[5] G. Colavolpe and A. Barbieri, "On MAP symbol detection for ISI channels using the Ungerboeck observation model," *IEEE Communications Letters*, vol. 9, no. 8, pp. 720–722, 2005.

[6] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings IEEE International Conference on Communications ICC '95*, vol. 2, 1995, pp. 1009–1013 vol.2.

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

[8] G. Ungerboeck, "Adaptive Maximum-Likelihood Receiver for Carrier-Modulated Data-Transmission Systems," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 624–636, 1974.

[9] F. Rusek and A. Prlja, "Optimal Channel Shortening for MIMO and ISI Channels," *IEEE Transactions on Wireless Communications*, vol. 11, no. 2, pp. 810–818, 2012.

[10] S. Weithoffer, C. A. Nour, N. Wehn, C. Douillard, and C. Berrou, "25 Years of Turbo Codes: From Mb/s to beyond 100 Gb/s," in *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, 2018, pp. 1–6.

[11] Z. Gajda and L. Sekanina, "Reducing the number of transistors in digital circuits using gate-level evolutionary design," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007, pp. 245–252.

[12] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. A K Peters, 2001.

[13] C. Drake, "PyEDA: Data Structures and Algorithms for Electronic Design Automation." in *SciPy*, 2015, pp. 25–30.

[14] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI synthesis*. Springer Science & Business Media, 1984, vol. 2.