electronics

Article

Partial Order-Based Decoding of Rate-1 Nodes in Fast Simplified Successive-Cancellation List Decoders for Polar Codes

Lucas Johannsen [1,*], Claus Kestel [2], Oliver Griebel [2], Timo Vogt [1] and Norbert Wehn [2]

[1] Faculty of Engineering, Koblenz University of Applied Sciences, 56075 Koblenz, Germany; vogt@hs-koblenz.de

[2] Microelectronic Systems Design Research Group, University of Kaiserslautern, 67663 Kaiserslautern, Germany; kestel@eit.uni-kl.de (C.K.); griebel@eit.uni-kl.de (O.G.); wehn@eit.uni-kl.de (N.W.)

* Correspondence: johannsen@hs-koblenz.de

Abstract: Polar codes are the first family of error-correcting codes that can achieve channel capacity. Among the known decoding algorithms, Successive-Cancellation List (SCL) decoding supported by a Cyclic Redundancy Check (CRC) shows the best error-correction performance at the cost of a high decoding complexity. The decoding of Rate-1 nodes belongs to the most complex tasks in SCL decoding. In this paper, we present a new algorithm that largely reduces the number of considered candidates in a Rate-1 node and generate all required candidates in parallel. For this purpose, we use a partial order of the candidate paths to prove that only a specified number of candidates needs to be considered. Further complexity reductions are achieved by an extended threshold-based path exclusion scheme at the cost of negligible error-correction performance loss. We present detailed Application-Specific Integrated Circuit (ASIC) implementation data on a 28 nm Fully Depleted Silicon on Insulator (FD-SOI) Complementary Metal-Oxide-Semiconductor (CMOS) technology for decoders with code length 128. We show that the new decoders outperform state-of-the-art reference decoders. For list size 8, improvements of up to 158.8% and 62.5% in area and energy efficiency are observed, respectively.

Keywords: polar code; CRC-aided successive-cancellation list decoding; Rate-1 node; ASIC decoder implementations; parallel high-throughput architectures

check for updates

Citation: Johannsen, L.; Kestel, C.; Griebel, O.; Vogt, T.; Wehn, N. Partial Order-Based Decoding of Rate-1 Nodes in Fast Simplified Successive-Cancellation List Decoders for Polar Codes . *Electronics* **2022**, *11*, 560. https://doi.org/10.3390/electronics11040560

Academic Editors: Byeong Yong Kong and Hoyoung Yoo

Received: 23 December 2021
Accepted: 3 February 2022
Published: 12 February 2022

1. Introduction

Polar codes are the first error-correction codes that were proven to achieve the capacity of Binary Symmetric Memoryless Channels (BSMCs) [1]. The low complexity Successive-Cancellation (SC) decoding algorithm [1] traverses the Polar Factor Tree (PFT) in a depth-first manner [2]. Due to this sequential behavior, SC decoders have a high latency. Tree pruning is used in Fast Simplified SC (Fast-SSC) decoding [2,3], where nodes with predefined bit patterns, i.e., nodes that represent certain constituent codes, are decoded directly. Thus, tree pruning reduces the latency of SC-based decoders.

One drawback of SC decoding is its limited error-correction performance for finite code lengths. To overcome error propagation in SC decoding, SC flip decoders rerun SC decoding with different bit decisions in a set of unreliable bit positions [4]. Successive-Cancellation List (SCL) decoding considers L concurrent decoding paths in parallel to close the gap between SC decoding and maximum-likelihood decoding [5], where L denotes the list size. The consideration of both values of an information bit, 0 and 1, splits the particular decoding path. Sorting is needed to keep track of the L most probable decoding paths. Concatenating the Polar code with an outer Cyclic Redundancy Check (CRC) further improves the error-correction performance by assisting the final path selection [5,6]. The drawback of SCL decoding is a high computational complexity caused by the usage of L parallel SC decoders and sorting.

The reduction of the number of path splittings is one approach to reduce the computational complexity of SCL decoding. In [7], a condition for path splitting in conventional SCL decoding was presented along with an additional path pruning scheme to further reduce the average number of decoding paths. In [8], an auxiliary path metric was introduced for a set of unreliable information bits in CRC aided SCL (CA-SCL) decoding to determine whether a path split leads to an erroneous path. Together with a threshold, these metrics are used to judge whether path splitting is necessary or not. In [9], an additional path contraction mechanism further reduces the number of paths.

Although Refs. [7–9] reduce the computational complexity of SCL decoders in terms of the number of path splittings, they do not exploit the potential of low latency and complexity reduction by tree pruning as in Fast-SSC decoding. In Fast Simplified SCL (Fast-SSCL) decoding, the Fast-SSC decoding algorithm was adapted for the use in SCL decoding [10,11]. The key enabler for Fast-SSCL decoding according to [11] is the use of the information about the order of the input paths and the order of bit estimations in Rate-1 nodes. A parallel decoding approach for Rate-1 nodes was introduced in [12] to enable high-throughput hardware implementations of Fast-SSCL decoders. However, due to numerous candidate paths for increasing list sizes, the maximum node size of Rate-1 nodes was restricted. The authors of [13] present a node reliability-based path expansion scheme using precomputed flipping patterns. While the node reliability results from code construction, the determination of these patterns remains open. The approach of an adaptive candidate selection method based on the reliability of the input paths was presented in [14]. However, they only presented an example for a Rate-1 node in Simplified SCL (SSCL) decoding with list size $L = 8$, while a formal presentation of an algorithm is missing. An advanced parallel decoding algorithm for Rate-1 nodes in Fast-SSCL decoders was presented in [15]. Based on a Minimum-Combinations (MC) set relative to the list size, redundant path splittings are eliminated, and a significant latency reduction is achieved.

In this work, we focus on the optimization of parallel Rate-1 node decoding by incorporating flipping combinations related to the order of input paths and make the following contributions:

- We present a new algorithm for the candidate generation in Rate-1 nodes of SCL decoders based on a partial order of the candidates. With this algorithm, the number of candidates is significantly reduced. We show that the presented algorithm preserves the error-correction performance of conventional SCL decoding.
- We introduce an extended threshold-based candidate selection scheme for practical SCL decoder implementations, which further reduces the number of considered decoding paths with negligible error-correction performance degradation. Compared to the state-of-the-art, this scheme can reduce the number of considered candidates and the sorter complexity in terms of the number of comparators by up to 56.3% and 81.7%, respectively.
- We present new architectures for Rate-1 nodes, which are based on the presented candidate generation algorithms. Using these architectures in unrolled Fast-SSCL decoder implementations significantly reduces the implementation costs. We present a set of decoder implementations in a 28 nm Fully Depleted Silicon on Insulator (FD-SOI) Complementary Metal-Oxide-Semiconductor (CMOS) technology for a Polar code with $N = 128$ to compare the implementation costs in terms of area efficiency and energy efficiency with state-of-the-art decoders. Improvements of up to 158.8% and 62.5% in the aforementioned metrics are observed, respectively.

The remainder of this paper is structured in four sections. We describe the background of Polar code decoding in Section 2. Within four steps, we develop an algorithm to generate the required candidate paths in the parallel decoding of Rate-1 nodes in Section 3. An additional threshold is introduced to further reduce the number of candidates. In Section 4, we analyze the impact of the candidate reduction on the error-correction performance and present estimations on the complexity reduction, together with a comparison of hardware implementation results of decoders with different list sizes. Section 5 concludes this paper.

2. Background

Polar codes are linear block codes denoted by $\mathcal{P}(N, K)$, with code length $N = 2^n$, K information bits and code rate $R = K/N$. Polar codes use the phenomenon of channel polarization [1] to divide bit-channels into reliable and noisy channels. Information bits are sent over reliable channels specified in the information set $\mathcal{I}$. The frozen set $\mathcal{F} = \mathcal{I}^C$ of length $N - K$ specifies the unreliable channels, which are used to include redundancy. Bit channels correspond to the rows of the generator-matrix $\boldsymbol{G}_N = \boldsymbol{G}_2^{\otimes n}$. Here, the polarization kernel $\boldsymbol{G}_2 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ and $\otimes n$ denotes the n-th Kronecker product. The bits of an input vector $\boldsymbol{u}$ of length N at the positions of $\mathcal{F}$, called frozen bits, are set to zero. The codeword $\boldsymbol{x}$ is calculated from $\boldsymbol{u}$ by

$$\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}_N. \tag{1}$$

2.1. SC and Fast-SSC Decoding

The original SC decoding process [1] for Polar codes of length N can be obtained by message passing between nodes of a balanced binary tree [2] with $2N - 1$ nodes, named Polar Factor Tree (PFT). Figure 1 shows a PFT for $N = 16$. For non-systematic Polar codes, the decoded bits $\hat{u}_i$, $i \in [0, N-1]$ are estimated in the leaf nodes at layer $s = 0$ of the PFT. The N received channel values $\boldsymbol{y}$ are input to the root node ($s = n$) as channel Log-Likelihood Ratios (LLRs) $\alpha_i = \log(Pr(y_i|x_i = 0)/Pr(y_i|x_i = 1))$. In layer s of the PFT, a node v of size $N_v = 2^s$ receives a vector $\boldsymbol{\alpha}^v$ of N_v LLRs from the parent node and returns a hard decision vector $\boldsymbol{\beta}^v$ of the same length. The messages passed to the left and the right child nodes, $\boldsymbol{\alpha}^l$ and $\boldsymbol{\alpha}^r$, are computed by the f- and g-functions as defined in [16], respectively. The message to the parent node is the partial sum vector $\boldsymbol{\beta}^v$, calculated by the combination of $\boldsymbol{\beta}^l$ and $\boldsymbol{\beta}^r$, denoted by h-function. In this way, the PFT is traversed depth first with priority to the left child, since its result is used to determine the message to the right child. In the leaf nodes, $\boldsymbol{\beta}^v$ only has one element, and the estimated bit $\hat{u}_i$ determined by

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } \alpha_i^v \geq 0 \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

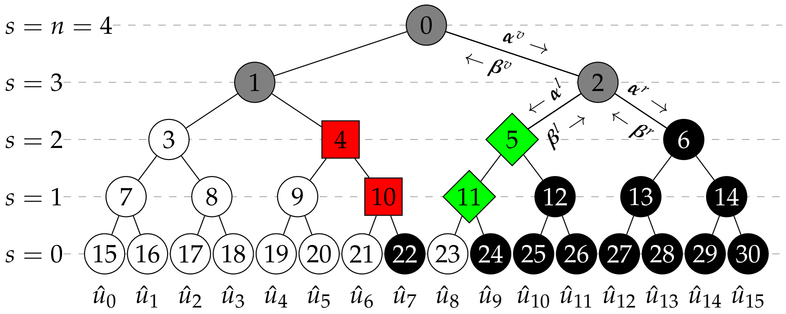

$s = n = 4$
$s = 3$
$s = 2$
$s = 1$
$s = 0$
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
$\boldsymbol{\alpha}^v \rightarrow$
$\leftarrow \boldsymbol{\beta}^v$
$\leftarrow \boldsymbol{\alpha}^l$
$\boldsymbol{\beta}^l \rightarrow$
$\boldsymbol{\alpha}^r \rightarrow$
$\leftarrow \boldsymbol{\beta}^r$
$\hat{u}_0$ $\hat{u}_1$ $\hat{u}_2$ $\hat{u}_3$ $\hat{u}_4$ $\hat{u}_5$ $\hat{u}_6$ $\hat{u}_7$ $\hat{u}_8$ $\hat{u}_9$ $\hat{u}_{10}$ $\hat{u}_{11}$ $\hat{u}_{12}$ $\hat{u}_{13}$ $\hat{u}_{14}$ $\hat{u}_{15}$

Figure 1. Polar factor tree for $\mathcal{P}(16, 8)$ with $\{u_0, \ldots, u_6, u_8\} \in \mathcal{F}$. Rate-0, Rate-1, Repetition code (REP) and Single Parity-Check code (SPC) nodes with a node size restriction of $N_{v,\max} = 4$ are represented as white and black circles, red squares and green rhombs, respectively.

Rate-0 nodes, whose leafs are all frozen bits, or Rate-1 nodes, whose leafs are all information bits, can be decoded without the need of traversing their subtrees, which is denoted by Simplified SC (SSC) decoding [2]. Alike, PFT tree pruning can be carried out for nodes that represent Repetition code (REP) nodes and Single Parity-Check code (SPC) nodes, resulting in Fast-SSC decoding [3]. The PFT in Figure 1 shows Rate-0, Rate-1, REP and SPC nodes with a node size restriction of $N_{v,\max} = 4$ as white and black circles, red squares and green rhombs, respectively. In Fast-SSC decoding (with the aforementioned restriction), the subtrees of these nodes in layer $s = 2$ need not be traversed. This tree pruning largely reduces the size of the PFT and thus the number of visited nodes.

2.2. SCL Decoding

To improve the error-correction performance of Polar codes, SCL decoding with list size L has been introduced in [5]. In the PFT, instead of vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, lists of L vectors are passed among the nodes. Therefore, a second index $l \in [0, L-1]$ is introduced, e.g., the partial-sums of a node v are denoted by $\boldsymbol{\beta}^{v,l}$. In layer $s = 0$, at the leaves of the unpruned PFT, both possible values, 0 and 1, for an information bit estimation are considered. Therefore, the number of decoding paths doubles for every bit estimation. When the number of paths exceeds the list size L, unreliable paths must be discarded by sorting. The reliability of each path is rated by a Path Metric (PM), which is updated at every bit estimation. For LLR-based SCL decoding [17], the PMs are initialized with 0. When estimating bit i, each PM of a path with index $p \in [0, 2L-1]$ that proceeds the input path $l \in [0, L-1]$, is updated by

$$\mathrm{PM}_i^p = \begin{cases} \mathrm{PM}_{i-1}^l + \left|\alpha_i^{v,l}\right|, & \text{if } \beta_i^{v,l} \neq \mathrm{HDD}(\alpha_i^{v,l}) \\ \mathrm{PM}_{i-1}^l, & \text{otherwise,} \end{cases} \tag{3}$$

with $\mathrm{HDD}(\boldsymbol{\alpha}^v)$ being Hard Decision Decoding (HDD) on $\boldsymbol{\alpha}^v$. Since small PMs belong to probable paths, those paths with low PM values survive the sorting step. After the last bit decision, the path with the smallest PM is chosen as the output of the decoder.

A CRC included in the input vector $\boldsymbol{u}$ as an outer code assists the path selection. The output path in SCL decoding is chosen based on a valid CRC, which further improves the error-correction performance.

2.3. Fast-SSCL Decoding and Rate-1 Nodes

Similar to the PFT optimization for the SC algorithm, optimized Rate-0, Rate-1, REP and SPC nodes also exist for the SCL algorithm [10,11,18–20]. The calculation of PMs and the candidate generation, i.e., proper path splitting, in the optimized nodes are the main tasks in adapting the PFT pruning techniques to SCL decoding. State-of-the-art Fast-SSCL decoding was introduced in [11] with the focus on the decoding of Rate-1 nodes. Instead of considering all $L \cdot 2^{N_v}$ possible candidates in a Rate-1 node of length N_v and list size L, the authors of [11] proved that the number of path splits needed to avoid any error-correction performance degradation is

$$P = \min(N_v, L-1). \tag{4}$$

The number of all $L \cdot 2^P$ possibly considered candidates is reduced by (4), when $L - 1 < N_v$. However, the Rate-1 node decoding algorithm in [11] is derived from list sphere decoding [17] and involves a sorting step after each bit estimation where the decoding path is split. Consequently, for each of the P path splits, the number of paths doubles, while only the L most probable ones are considered after $2L$ to L sorting in the following splitting step. Thus, the number of visited paths in a Rate-1 node of [11] is

$$L_{\mathrm{out,s}} = L \cdot (P+1). \tag{5}$$

In [12], a parallel decoding implementation of a Rate-1 node is used. The corresponding architecture is shown in Figure 2. To achieve low latency and high throughput, all the relevant candidate partial sums $\boldsymbol{\beta}^{l,j}$ according to (4) are estimated in parallel. The 2^P possible candidates originating from input path $l \in [0, L-1]$ are indexed by $j \in [0, 2^P - 1]$. For the parallel processing, $2^P - 1$ splitting patterns for each of the L input paths are determined in the *Candidate Generator*. These patterns are used to generate the candidate partial sums by applying the bitwise XOR operation with the HDD estimates $\boldsymbol{\beta}^{l,0}$ of the dedicated LLR vectors $\boldsymbol{\alpha}^{v,l}$. Consequently,

$$L_{\text{out,p}} = L \cdot 2^P \tag{6}$$

candidates are considered and need to be sorted to extract the L most probable partial sums $\boldsymbol{\beta}^{v,l}$.

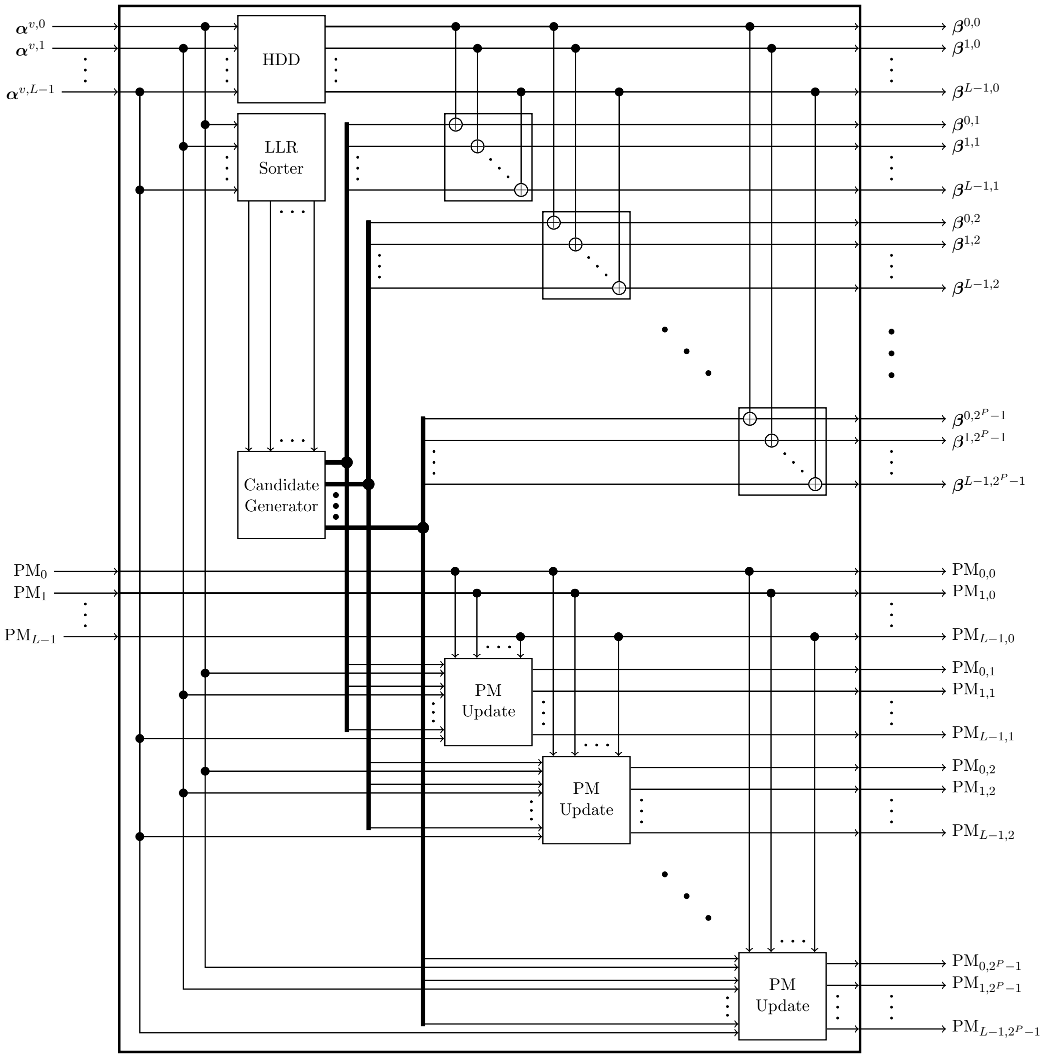


Figure 2. Parallel Rate-1 node architecture.

The notation of PMs is as follows: $\mathrm{PM}_{l,j}$ denotes the PM of the candidate path with index j originating from input path l. To estimate $\mathrm{PM}_{l,j}$, Equation (3) is restated as

$$\mathrm{PM}_{l,j} = \mathrm{PM}_l + \sum_{\substack{i=0 \\ \beta_i^{l,j} \neq \beta_i^{l,0}}}^{N_v-1} \left| \alpha_i^{v,l} \right|. \tag{7}$$

Note that the PM of an HDD estimate is always equal to the input PM and written as $\mathrm{PM}_{l,0}$. PM_l with a single index is used to denote the l-th PM in a sorted list of PMs.

Due to the exponential increase of the number of candidate paths with N_v for $N_v > L$, in [12], a restriction on the maximal node size of Rate-1 nodes $N_{v,Rate\text{-}1_{max}} = 4$ was set for $L > 4$. This limits tree pruning to layer $s = 2$ for Rate-1 nodes and thus the optimization of the PFT. This especially affects the rightmost Rate-1 nodes, which tend to be large due to polarization being predominant at high bit channels' indices.

Concerning the rightmost Rate-1 nodes (e.g., node $v = 6$ in Figure 1), it is important to recollect their relation to CRC. Without the presence of an CRC, the rightmost Rate-1 nodes only need to proceed with the most probable path [7]. In contrast, using a CRC requires the maintenance of the list of L paths to preserve error-correction performance compared to CA-SCL decoding.

3. Candidate Generation in Parallel Decoding of Rate-1 Nodes

In this section, we show that it is possible to exclude paths in Rate-1 nodes of Fast-SSCL decoders. For that purpose, we incorporate the order of the list of incoming candidates and the indices of the least reliable LLRs, i.e., the positions where bit flips are carried out. Therefore, we exploit the relations between the $L_{\mathrm{out,p}}$ possible paths. In contrast to [11], we consider both the information about the order of the input paths and the order of sorted LLR magnitudes simultaneously and in relation to each other. We reduce the number of considered candidate paths to decrease the computational complexity of the Rate-1 node itself and the subsequent sorter unit. The theorems and their proofs in Section 3.1 are based on considerations about the relations between the PMs of candidates with different numbers of flipped bits. In Section 3.2, the findings of these considerations are combined with the knowledge about the indices of the bits that have to be flipped, to describe a Partial Order (PO) of the candidates. This PO is used to generate only the candidate paths that need to be considered in Rate-1 nodes to preserve error-correction performance of conventional SCL decoding. Furthermore, threshold-based path pruning of [20] is combined with PO-based candidate generation in Section 3.3 denoted by PO with threshold S (POS) candidate generation. This threshold is extended in Section 3.4 resulting in Extended POS (ExPOS) candidate generation to further reduce the number of generated candidates and enable more efficient implementations with negligible error-correction performance loss.

Throughout this section, we often use the wording "path a" or "candidate b" to abbreviate the phrases "the path with index a" or "the candidate with index b".

We use $j \in [0, 2^P - 1]$ to index all 2^P possible candidates originating from input path l. The P bits of the binary expansion of j are used inside the *Candidate Generator* (Figure 2) to construct the flipping patterns for the generation of the candidate partial sums $\boldsymbol{\beta}^{l,j}$. Each 1 in the binary expansion of j stands for a flipped bit position. Specifically, a less significant 1 in the binary expansion of j is mapped to a less reliable bit location in $\boldsymbol{\alpha}^{v,l}$ known from the result of the *LLR Sorter* (Figure 2). Thus, j indicates the positions of flipped bits in $\boldsymbol{\beta}^{l,j}$. The Hamming weight $\mathrm{hw}(j)$, defined as the number of ones in the binary expansion of j, gives us the number of flipped bits in candidate j.

3.1. Path Exclusion Based on PM Relations

We start our considerations by investigating the relations between PMs. The PM calculation in (7) represents a cost-function, thus PMs are monotonically increasing. Therefore, in a Rate-1 node, candidates originating from input path l and generated by flipping bits compared to $\boldsymbol{\beta}^{l,0}$ can never obtain a lower PM than $\boldsymbol{\beta}^{l,0}$ itself. Since the input paths are sorted, i.e.,

$$\mathrm{PM}_0 \leq \mathrm{PM}_1 \leq \cdots \leq \mathrm{PM}_{L-1}, \tag{8}$$

the codeword estimated by HDD of input path l always remains more probable than the one of input path $l+1$, which also holds for equivalence since we use a stable sorter.

Theorem 1. *In Fast-SSCL decoding with list size L, there is no need for path splitting in the least reliable input path $L-1$ of a Rate-1 node.*

Proof. We prove Theorem 1 by contradiction. Let us assume that the two most probable paths originating from input path $L-1$ survive sorting. These paths are obtained by the HDD estimate of $\boldsymbol{\alpha}^{v,L-1}$ and its fork with one flipped bit at the least reliable bit position, respectively. The corresponding LLR is $\alpha_{min}^{v,L-1}$. After sorting, these paths have the indices p and q, respectively. The corresponding PMs are

$$\mathrm{PM}_p = \mathrm{PM}_{L-1,0}, \tag{9}$$

$$\mathrm{PM}_q = \mathrm{PM}_{L-1,0} + |\alpha_{min}^{v,L-1}| = \mathrm{PM}_{L-1,1}, \tag{10}$$

where $0 \leq p < q < L$ must apply due to the assumption that these candidates survive. Now, we show that there are at least L paths with PMs which are less than or equal to PM_q. Let us consider a path that originates from input path l and corresponds to the HDD estimate of the LLR vector $\boldsymbol{\alpha}^{v,l}$. After sorting, this path has the index s and its PM is

$$\mathrm{PM}_s = \mathrm{PM}_{l,0}. \tag{11}$$

With (8), we can conclude that

$$\mathrm{PM}_s \leq \mathrm{PM}_q, \tag{12}$$

which results in $s < q$. Since we have L input paths, s can assume L values (including $p = L - 1$), for which (12) is true. Thus, $q \geq L$, which contradicts the assumption that path q survives. Theorem 1 is consequently proven. □

From the considerations of Theorem 1, it can be deduced that the minimum index of the candidate partial sum $\boldsymbol{\beta}^{l,0}$ is l. To a priori exclude further paths without the knowledge of the LLR values themselves, a metric is required, which rates all $L_{\mathrm{out,p}}$ candidate paths. For that purpose, we introduce $m'_{l,j}$ as the smallest possible index that a candidate j originating from input path l can have after estimating and sorting all the $L_{\mathrm{out,p}}$ possible candidates. The proof of Theorem 2 will show that $m'_{l,j}$ can be calculated by

$$m'_{l,j} = l + 2^{\mathrm{hw}(j)} - 1. \tag{13}$$

Theorem 2. *In Fast-SSCL decoding with list size L, the number of required candidates originating from input path l to a Rate-1 node of length N_v is*

$$L^{l}_{out,m'} = \sum_{\substack{j=0 \\ m'_{l,j}<L}}^{2^P-1} 1. \tag{14}$$

Proof. We start this proof by discussing why $m'_{l,j}$ is the smallest possible index of candidate j originating from path l. It follows from (4) that the number of candidates originating from path l is 2^P. These candidates, indexed by j, are generated by the HDD estimate of l-th LLR vector ($j = 0$) and flipping the bits in the P least reliable positions in all possible combinations. We now consider the candidates where exactly one, two and x bits are flipped:

1. *One flipped bit:* According to (7), all PMs of the candidates with exactly one flipped bit are greater than or equal to $\mathrm{PM}_{l,0}$ of the HDD estimation candidate of path l because the one corresponding $|\alpha^{v,l}_{i_{flip,j}}|$ is added. Thus, the minimal index these candidates can have after sorting is $l + 1$. The candidate index j in this case is a power of 2 because $\mathrm{hw}(2^n) = 1$ for all $n \in \mathbb{N}_0$.
2. *Two flipped bits:* The PMs of the candidates generated by flipping exactly two bits are greater than or equal to $\mathrm{PM}_{l,0}$ and greater than or equal to the two PMs of the candidates where the bits at the same two positions were flipped separately. To clarify, let us suppose three different candidates with indices σ, τ, ω exhibit $\mathrm{hw}(\sigma) = \mathrm{hw}(\tau) = 1$ and $\mathrm{hw}(\omega) = 2$, respectively. These two bits in which candidates σ and τ differ from the HDD candidate 0 are flipped simultaneously in candidate ω. Then,

$$\mathrm{PM}_{l,\sigma} = \mathrm{PM}_{l,0} + |\alpha_{i_\sigma}^{v,l}| \tag{15}$$

$$\mathrm{PM}_{l,\omega} = \mathrm{PM}_{l,0} + |\alpha_{i_\sigma}^{v,l}| + |\alpha_{i_\tau}^{v,l}| \tag{16}$$

$$\mathrm{PM}_{l,\tau} = \mathrm{PM}_{l,0} \qquad + |\alpha_{i_\tau}^{v,l}| \tag{17}$$

and

$$\mathrm{PM}_{l,\sigma} \leq \mathrm{PM}_{l,\omega} \geq \mathrm{PM}_{l,\tau}. \tag{18}$$

Considering a new candidate ρ with $\mathrm{hw}(\rho) = 1$, the relation between candidates ω and ρ is an open question. A priori, the relation between $\mathrm{PM}_{l,\omega}$ and $\mathrm{PM}_{l,\rho}$ is unknown, since $|\alpha_{i_\rho}^{v,l}|$ can be smaller than, greater than, or equal to $|\alpha_{i_\sigma}^{v,l}| + |\alpha_{i_\tau}^{v,l}|$. Therefore, we can only state with certainty that the three candidates $j = 0$, σ and τ are always more likely than candidate ω. Consequently, the minimal index candidates with exactly two flipped bits can have after sorting is $l + 3$.

3. *x flipped bits:* In general, the number of PMs known to be less than or equal to $\mathrm{PM}_{l,j}$ with $\mathrm{hw}(j) = x$ equals $2^x - 1$. This is the case because the number of candidates generated by the combinations of flipping the same x bits is 2^x and candidate j is the one where all x bits are flipped simultaneously. It follows that the minimal index candidate j can have after sorting all 2^x forks is $l + 2^x - 1$.

It can be seen that (13) describes the smallest possible index of a candidate j, depending on the number of flipped bits and the index l of its input path. With the knowledge of $m'_{l,j}$ being the smallest possible index of candidate j, we can conclude that each path with $m'_{l,j} > L - 1$ will be discarded, and therefore it is unnecessary to consider these paths. Theorem 2 is consequently proven. □

Remark 1. *Theorem 1 is a corollary to Theorem 2, but it was described first to build on.*

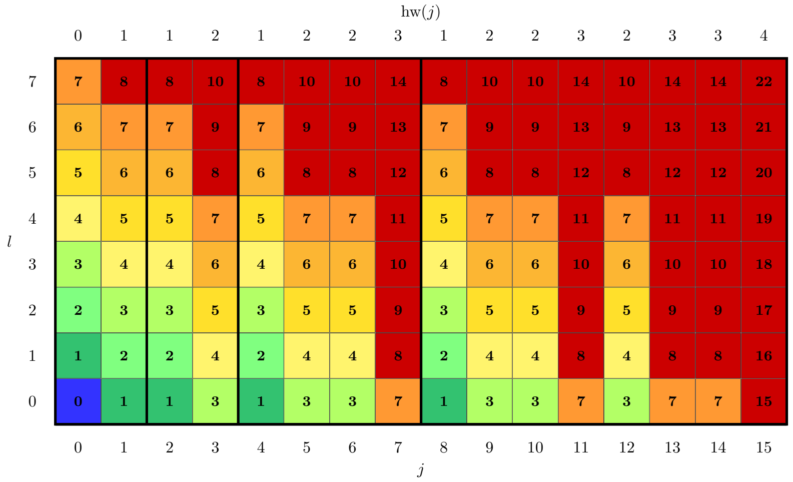
As an example, Figure 3 shows all the $m'_{l,j}$ for $L = 8$ and $P = 4$. All the candidates corresponding to a red colored $m'_{l,j}$ can be excluded a priori. With Theorem 2, the total number of required candidates in a Rate-1 node of length N_v in Fast-SSCL decoding with list size L consequently is

$$L_{\mathrm{out},m'} = \sum_{l=0}^{L-1} L_{\mathrm{out},m'}^{l}. \tag{19}$$

Furthermore, the number of required candidates from input path l as given in (14) is restated as a function of l, L and P by

$$L_{\mathrm{out},m'}^{l} = \sum_{a=0}^{\lfloor \mathrm{ld}(L-l) \rfloor} \binom{P}{a}. \tag{20}$$

There are two different approaches for the candidate generation, a serial one in [11] and a parallel one in this work. In a parallel implementation of a Rate-1 node, it is not known a priori which candidate paths would survive the intermediate sorting steps in [11] and, therefore, would be further expanded. However, with Theorem 1, for every bit estimation step in [11], the number of paths can be reduced from $2L$ to $2L - 1$. Theorem 2 shows that the number of paths can be further reduced, but the theorem is not directly applicable to the serial approach, since it is based on a global view.



hw(j)	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	4
l = 7	7	8	8	10	8	10	10	14	8	10	10	14	10	14	14	22
6	6	7	7	9	7	9	9	13	7	9	9	13	9	13	13	21
5	5	6	6	8	6	8	8	12	6	8	8	12	8	12	12	20
4	4	5	5	7	5	7	7	11	5	7	7	11	7	11	11	19
3	3	4	4	6	4	6	6	10	4	6	6	10	6	10	10	18
2	2	3	3	5	3	5	5	9	3	5	5	9	5	9	9	17
1	1	2	2	4	2	4	4	8	2	4	4	8	4	8	8	16
0	0	1	1	3	1	3	3	7	1	3	3	7	3	7	7	15
j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 3. Smallest possible candidate indices $m'_{l,j}$ for $L = 8$ and $P = 4$ with candidates $j \in [0, 2^P - 1]$ and list index $l \in [0, L-1]$.

3.2. Candidate Generation Based on Partially Ordered Candidate Sets

Compared to the $L_{\text{out,p}}$ candidates according to (6), many candidates can be excluded by the method described in the previous section. However, there are still many paths that are visited, especially when L increases. Taking into account that the flipped bit locations need to be determined anyway by identifying the P least reliable LLRs in each $\boldsymbol{\alpha}^{v,l}$, the knowledge about the relations between the LLRs can be used to exclude further paths. Due to this identification by an *LLR-sorter*, as shown in Figure 2, we can consider the P least reliable LLRs of input path l as sorted values, i.e.,

$$|\alpha_0^{v,l}| \leq |\alpha_1^{v,l}| \leq \cdots \leq |\alpha_{P-1}^{v,l}|. \tag{21}$$

Following the proof of Theorem 1 in [21], where the Partial Order (PO) of Polar code bit-channels from [22] is restated, we describe a similar PO for PMs. Therefore, we write the candidate index j in its binary expansion and, for better readability, we then write the index l of the input path as a superscript. From (21), we deduce the following lemma:

Lemma 1. *For $P = 2$ and $\text{hw}(j) = 1$, we have the PM-relation $PM_{01}^l \leq PM_{10}^l$.*

Replacing a more significant 0 with the next less significant 1 in the binary expansion of j corresponds to flipping a bit in a position of a more reliable LLR instead of flipping a bit in the position of the next less reliable LLR. Therefore, for any $p^a \in \{0,1\}^a$, $q^b \in \{0,1\}^b$ and $a, b \geq 0$, we can generalize Lemma 1 to

$$\text{PM}_{p^a 01 q^b}^l \leq \text{PM}_{p^a 10 q^b}^l. \tag{22}$$

Theorem 3. *In Fast-SSCL decoding, the PMs of all possible candidates in a Rate-1 node originating from input path l are related as*

$$PM_{p^a 0 r^c 1 q^b}^l \leq PM_{p^a 1 r^c 0 q^b}^l, \tag{23}$$

for all $p^a \in \{0,1\}^a$, $r^c \in \{0,1\}^c$, $q^b \in \{0,1\}^b$ and $a, c, b \geq 0$.

Proof. We prove Theorem 3 by induction. For the initial case, $c = 0$, Equation (23) is equal to (22). For the induction step, we assume that (23) is true for $c = d, d > 0$ and let $c = d + 1$. Then, an additional bit $\delta \in \{0,1\}$ is appended and $r^{d+1} = r^d \delta$. There are two cases to consider for the additional bit δ:

1. If $\delta = 0$, the relations

$$\begin{aligned}\mathrm{PM}^{l}_{p^a 0 r^{d+1} 1 q^b} = \mathrm{PM}^{l}_{p^a 0 r^{d} 0 1 q^b} &\overset{(A)}{\leq} \mathrm{PM}^{l}_{p^a 0 r^{d} 1 0 q^b} \\ &\overset{(B)}{\leq} \mathrm{PM}^{l}_{p^a 1 r^{c} 0 0 q^b} = \mathrm{PM}^{l}_{p^a 1 r^{d+1} 0 q^b},\end{aligned} \tag{24}$$

hold. Here, (A) follows from (22) and (B) is based on the induction hypothesis for $c = d$.

2. If $\delta = 1$, the relations

$$\begin{aligned}\mathrm{PM}^{l}_{p^a 0 r^{d+1} 1 q^b} = \mathrm{PM}^{l}_{p^a 0 r^{d} 1 1 q^b} &\overset{(C)}{\leq} \mathrm{PM}^{l}_{p^a 1 r^{d} 0 1 q^b} \\ &\overset{(D)}{\leq} \mathrm{PM}^{l}_{p^a 1 r^{c} 1 0 q^b} = \mathrm{PM}^{l}_{p^a 1 r^{d+1} 0 q^b},\end{aligned} \tag{25}$$

hold. Here, (C) is based on the induction hypothesis for $c = d$ and (D) follows from (22).

Combining cases (1) and (2), we obtain

$$\mathrm{PM}^{l}_{p^a 0 r^{d+1} 1 q^b} \leq \mathrm{PM}^{l}_{p^a 1 r^{d+1} 0 q^b}, \tag{26}$$

for all $r^{d+1} \in \{0,1\}^{d+1}$, $p^a \in \{0,1\}^a$, $q^b \in \{0,1\}^b$ and $a, b \geq 0$, which completes the induction step. □

The PO of PMs described in Theorem 3 is directly related to the binary expansions of the candidates indices j, for which reason we call it the PO of candidates. An example of a partially ordered set of candidates for $P = 5$ and a constant number of flipped bits $x = \mathrm{hw}(j) = 2$ is given in Figure 4. We chose these parameters because the resulting graph depicts the correlation between the partial order and the binary expansion of j concretely. We introduce the stage z as the index in a chain of the partially ordered set of all candidates with constant $x = \mathrm{hw}(j)$. For such a candidate, we write $j_{x,z}$. The smallest candidate index for a given x is

$$j_{x,0} = 2^x - 1. \tag{27}$$

From $j_{x,z}$ within stage z, $j_{x,z+1}$ is obtained by swapping a more significant 0 with the next less significant 1 in the binary expansion of $j_{x,z}$. We redefine the smallest possible candidate index originally given in (13) as

$$m_{l,j} = m'_{l,j} + z. \tag{28}$$

The values of the redefined $m_{l,j}$ for $L = 8$ and $P = 4$ are shown in Figure 5. In addition, the smallest possible index $m_{l,j}$ is displayed above the directed graph (Figure 4), to show how it is increased for each stage z.

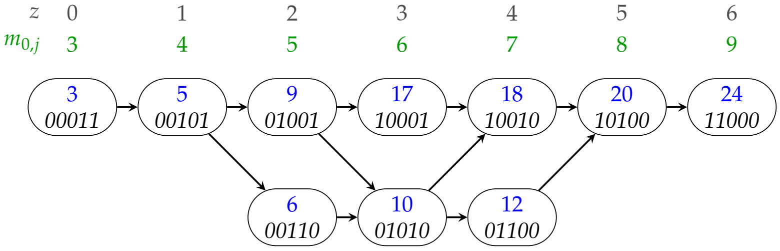
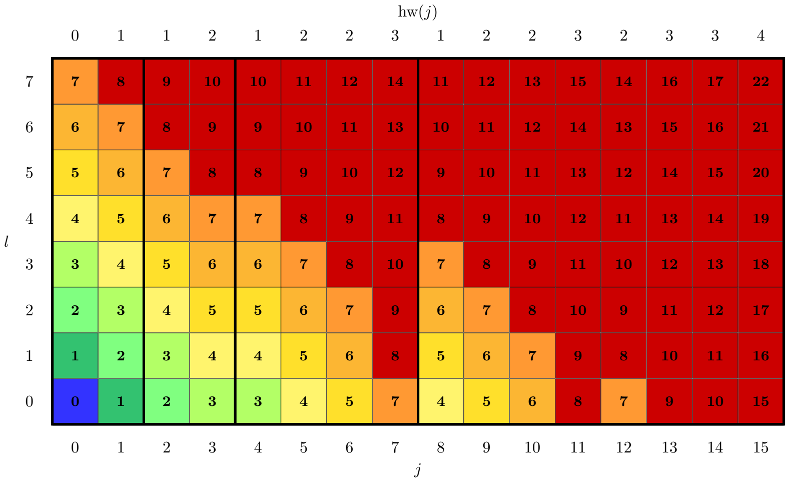


Figure 4. Directed graph of the partially ordered set of candidates j_2 (blue) with corresponding binary expansion (black) for $P = 5$ and $x = 2$.



hw(j)	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	4
7	7	8	9	10	10	11	12	14	11	12	13	15	14	16	17	22
6	6	7	8	9	9	10	11	13	10	11	12	14	13	15	16	21
5	5	6	7	8	8	9	10	12	9	10	11	13	12	14	15	20
4	4	5	6	7	7	8	9	11	8	9	10	12	11	13	14	19
3	3	4	5	6	6	7	8	10	7	8	9	11	10	12	13	18
2	2	3	4	5	5	6	7	9	6	7	8	10	9	11	12	17
1	1	2	3	4	4	5	6	8	5	6	7	9	8	10	11	16
0	0	1	2	3	3	4	5	7	4	5	6	8	7	9	10	15
l / j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 5. Smallest possible candidate indices $m_{l,j}$ for $L = 8$ and $P = 4$ with candidates $j \in [0, 2^P - 1]$ and list index $l \in [0, L - 1]$ calculated by the PO-based method.

Based on the observation that $m_{l,j_{x,z}} < m_{l,j_{x,z+1}}$ when $j_{x,z+1}$ is obtained by swapping a more significant 0 with the next less significant 1 in the binary expansion of $j_{x,z}$, we formulate Algorithm 1. It uses the PO to select the required candidates. Algorithm 1 only generates the candidates with $m_{l,j} < L$, i.e., those candidates whose $m_{l,j}$ are not colored in red in Figure 5.

Algorithm 1 Partial order based candidates generation.

```
procedure GenerateCandidates($L$, $P$)
    for $l \leftarrow 0, L-1$ do
        $candidates[l] \leftarrow \{0\}$
        for $x \leftarrow 1, P$ do        ▷ loop over the number of flipped bits, HDD ($x = 0$) is already in candidates list
            $z \leftarrow 0$                ▷ stage in directed graph
            $j_x[z] \leftarrow \{2^x - 1\}$       ▷ $j_{x,0}$ according to (27)
            $m_{l,j_x} \leftarrow l + 2^x - 1$     ▷ $m_{l,j_{x,0}}$ according to (13)
            while $m_{l,j_x} < L$ do
                                            ▷ Update candidates list
                $candidates[l] \leftarrow candidates[l] \cup j_x[z]$
                for $j_{x,z} \in j_x[z]$ do         ▷ Generate new candidates
                    for $c \leftarrow 0, x-1$ do      ▷ $x$ possible occurrences of 01
                        $j_{x,z+1}[c] \leftarrow$ Repl_01_10($j_{x,z}$, $x - 1 - c$)
                    end for
                    $j_x[z+1] \leftarrow j_x[z+1] \cup j_{x,z+1}$
                end for
                $z \leftarrow z + 1$              ▷ move to next stage in directed graph
                $m_{l,j_x} \leftarrow m_{l,j_x} + 1$
            end while
        end for
    end for
    return $candidates$
end procedure

procedure Repl_01_10($cand$, $o$)
    $result \leftarrow$ Replace $o$-th occurrence of 01 by 10 in binary representation of
            $cand$
    return $result$
end procedure
```

Theorem 4. *In Fast-SSCL decoding with list size L, the number of required candidates originating from input path l to a Rate-1 node of length N_v is*

$$L^{l}_{out,m} = \sum_{\substack{j=0 \\ m_{l,j}<L}}^{2^P-1} 1. \tag{29}$$

Proof. To prove Theorem 4, we need to verify that Equation (28) represents the smallest possible index of a candidate after sorting. Since we have proven (13) in Theorem 2, it is open to prove that the increment by z is valid. This, however, follows directly from Theorem 3. Thus, Theorem 4 is also proven. □

With Theorem 4, the total number of required candidates in a Rate-1 node of length N_v in Fast-SSCL decoding with list size L consequently is observed by (19), when $m'_{l,j}$ and $L^{l}_{\text{out},m'}$ are replaced by m_l, j and $L^{l}_{\text{out},m}$, respectively, as

$$L_{\text{out},m} = \sum_{l=0}^{L-1} L^{l}_{\text{out},m}. \tag{30}$$

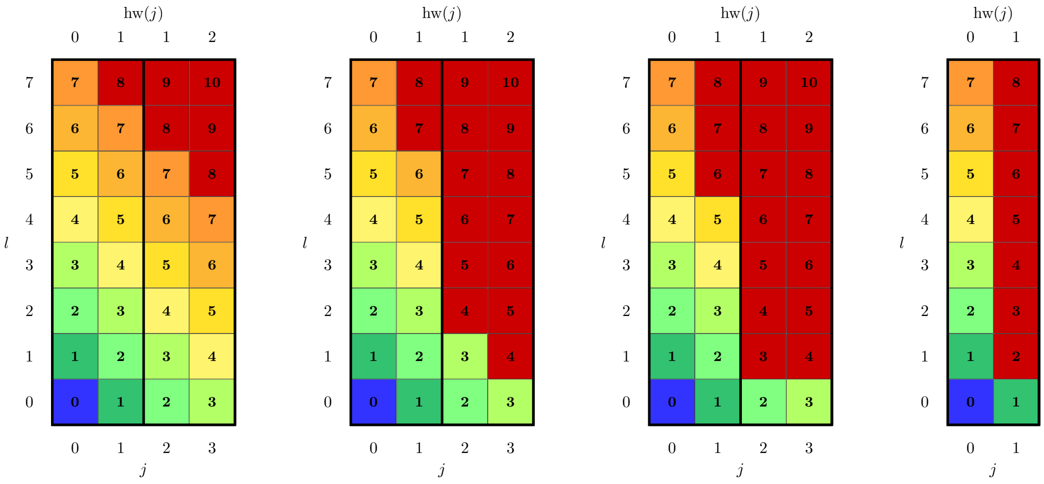
Fast-SSCL decoding with the use of Rate-1 nodes with PO-based candidates generation is denoted by PO-based Fast-SSCL (PO-Fast-SSCL) decoding.

3.3. Partial Order-Based Candidate Generation with Threshold

As described in [20], fewer path forks are required to match the error-correction performance of conventional SCL decoding in practical decoder implementations. Thus, Ref. [20] introduced a threshold $S_{\text{Rate-1}}$ to limit the number of path splits. They determined $S_{\text{Rate-1}}$ heuristically by analyzing the error-correction performance for different $S_{\text{Rate-1}}$ and $\mathcal{P}(1024, 512)$. For $L \in \{2, 4, 8\}$, an associated threshold $S_{\text{Rate-1}} \in \{1, 1, 2\}$ was set, respectively, to replace $L - 1$ in (4):

$$P = \min(N_v, S_{\text{Rate-1}}). \tag{31}$$

POS-based candidate generation is denoted as the adoption of the threshold $S_{\text{Rate-1}}$ in the use of Algorithm 1. POS-based candidate generation results in a vertical cutting-off of the candidates. This is shown in Figure 6a, which represents the left part of Figure 5.



(a) (b) (c) (d)

Figure 6. Smallest possible candidate indices $m_{l,j}$ for $L = 8$, calculated by (**a**) POS algorithm with $S_{\text{Rate-1}} = 2$, (**b**) ExPOS algorithm with $S_{\text{Rate-1}} = 2$, $k_c = 3$, (**c**) ExPOS algorithm with $S_{\text{Rate-1}} = 2$, $k_c = 4$, (**d**) ExPOS algorithm with $S_{\text{Rate-1}} = 1$, $k_c = 8$.

Fast-SSCL decoding with the use of PO-based candidate generation in Rate-1 nodes, together with (31), is denoted by POS-based Fast-SSCL (POS-Fast-SSCL) decoding. For the estimation of $S_{\text{Rate-1}}$, we refer to the first loop of Algorithm 2 and its description in Section 3.4.

3.4. Extended Threshold Partial Order-Based Candidate Generation

To further reduce the number of candidates for practical decoder implementations, the approach of limiting the number of paths by a threshold is extended. For that purpose, the rates of surviving the sorting step for each candidate j in the Rate-1 nodes of POS-Fast-SSCL decoders are investigated. The surviving rate h_j is obtained as the empirical probability of the event that candidate j from input path l survives sorting, normalized to the sum over all proceeded candidates divided by L.

As an example, Figure 7 shows the means of the surviving rates h_j of candidates j in Rate-1 nodes with $L = 32$ and $P = S_{\text{Rate-1}} = 5$ as a heat map. The values were observed by a total of 1.5×10^6 samples, extracted from POS-Fast-SSCL decoders for a code with $N = 1024$ and $R \in \{1/3, 1/2, 5/6\}$ during decoding data received at Signal-to-Noise-Ratios (SNRs) of $E_b/N_0 \in \{0, 3, 6, 9\}$. The normalization in the color map is divided into two slopes: The first half of the color range is mapped linearly to values from 0 to 0.05 and the second half is mapped linearly to values from 0.05 to 1.0.

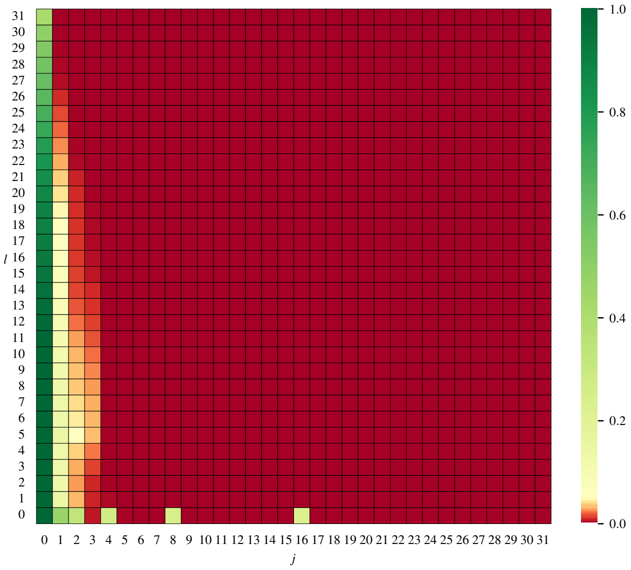


Figure 7. Empirical probability of surviving the sorting step h_j for candidates j in Rate-1 nodes with $L = 32$ and $P = 5$.

It can be observed that these surviving rates do not directly map to $m_{l,j}$. Candidate $j = 0$ from input paths 0 to 9 always survives, and from input paths $l > 9$, the surviving rates decrease from 1.0 to 0.4. In general, the highest surviving rate of each j is observed for candidates originating from input path $l = 0$. The surviving rates of candidates with $x = 1$ decrease with increasing j and decreasing l. An exception is $j = 2$, where a hot spot around $l = 5$ appears, but candidate $j = 2$ from input path $l = 0$ still has the highest surviving rate. For $j \geq 4$ and $x = 1$, the surviving rates for $l \geq 1$ are smaller than 0.002. The aforementioned hot spot also appears for $j = 3$, but here the candidate originating from input path $l = 0$ does not exhibit the highest surviving rate.

Similar observations were also made for other list sizes, as well as for the separate samples of the parameters R and E_B/N_0 that were averaged in Figure 7. In summary, the investigation of the surviving rates revealed that:

- Candidates with $j = 3$ are the only needed candidates with $x = 2$. All other candidates with $x \geq 2$ can be discarded.

- The number of candidates with equal j can be decreased linearly until only the candidate originating from path $l = 0$ needs to be considered.

We approximate an exclusion of the candidates with the smallest surviving rates by a new threshold θ_j to replace L in (29) and in Algorithm 1, line 4. The observed linear decrease is implemented by a gradient k_c. The threshold for the exclusion of candidate j with $x = \mathrm{hw}(j)$ flipped bits and originating from input path l is determined by

$$\theta_j = \min(L, \max(m_{0,j}, L - k_c \cdot j + x) + 1). \tag{32}$$

The gradient k_c needs to be determined for each combination of N, R, L and $S_{\text{Rate-1}}$.

To consider the described findings about the surviving rates, Algorithm 1 is adapted at the given lines as follows:

line 4: Set $x = 1$ instead of looping over the number of flipped bits.

line 8: Replace the stopping condition by $m_{l,j} < \theta_j$.

line 19: Append candidate $j = 3$ when $P >= 1$ and $m_{l,3} < \theta_3$ according to (32).

We call the adapted algorithm ExPOS-based candidate generation. Fast-SSCL decoding with the use of ExPOS-based candidate generation in Rate-1 nodes is denoted by ExPOS-based Fast-SSCL (ExPOS-Fast-SSCL) decoding.

To determine the thresholds in ExPOS-Fast-SSCL decoding, an iterative approximation is provided in Algorithm 2. The iterative approximation is based on Monte Carlo error-correction performance simulations. The first loop of Algorithm 2 determines $S_{\text{Rate-1}}$ by comparing the error-correction performance of POS-Fast-SSCL decoding and SCL decoding. $S_{\text{Rate-1}}$ is initialized with 0 and increased until the performance loss at a specified target Frame Error Rate (FER) or Bit Error Rate (BER) is smaller than a specified Δ. Once a sufficient $S_{\text{Rate-1}}$ is found, it is used in the second loop to determine k_c as the relevant parameter for the calculation of the internal threshold θ_j for the candidate selection. The approximation proceeds in the same way as for $S_{\text{Rate-1}}$, with k_c being initialized with L and decreased until the error-correction performance of ExPOS-Fast-SSCL decoding is close enough to SCL decoding.

Algorithm 2 Iterative threshold approximation.

```
procedure ThresholdApproximation($\mathcal{P}$, $L$, target FER/BER, $\Delta$)
    Run error-correction performance simulation with SCL decoding.
    for $S_{Rate\text{-}1} \leftarrow 0, L-1$ do          ▷ Start with minimal number of path splits
        Run error-correction performance simulation
        with POS-Fast-SSCL decoding.
        if Performance loss at target FER/BER $< \Delta$ then
            break
        end if
    end for
    for $k_c \leftarrow L, 0$ do          ▷ Start with maximal gradient for candidate exclusion
        Run error-correction performance simulation
        with ExPOS-Fast-SSCL decoding.
        if Performance loss at target FER/BER $< \Delta$ then
            break
        end if
    end for
    return $S_{Rate\text{-}1}$, $k_c$
end procedure
```

The computational effort for each simulation depends on the choice of the target FER/BER. However, the maximum number of needed simulation runs is L for the estimation of $S_{\text{Rate-1}}$ and $L+1$ for the estimation of k_c.

In Figure 6b–d, $m_{l,j}$ is shown resulting from ExPOS-based candidate generation. The 17 candidates, considered with $S_{\text{Rate-1}} = 2$ and $k_c = 3$, are the ones not colored in red in Figure 6b. When $k_c = 4$, the number of candidates decreases to 15 as shown in Figure 6c. Setting $S_{\text{Rate-1}} = 1$ and $k_c = 8$ results in Figure 6d, where only the eight HDD estimates and the one candidate $j = 1$ of input path $l = 0$ are considered.

4. Results and Discussion

This section presents the results of error-correction performance simulations and complexity estimations in terms of the number of candidates generated in Rate1-nodes with different parameters and the number of comparators needed in the dedicated sorting units. Furthermore, we present hardware implementation results to show that applying POS- and ExPOS-based candidate generation in the Rate-1 node architectures results in decoder implementations that outperform state-of-the-art decoders.

4.1. Error-Correction Performance

To show the error-correction performance of the presented algorithms, we evaluate Polar codes with code length $N = 1024$ and code rates $R \in \{1/6, 1/2, 5/6\}$. All codes are constructed according to [23] with a design SNR of 0 dB for an Additive White Gaussian Noise (AWGN) channel. Furthermore, all codes are concatenated with an outer CRC of length $C = 16$, which is placed in the most reliable bit channels of $\mathcal{F}$. We restrict the presented results to the aforementioned parameters, due to a large design space. However, further simulations have shown that the presented results are transferable to other Polar codes $\mathcal{P}(N, K)$. Note that the choice of C affects the error-correction performance twofold. On the one hand, CRC bits assist the selection of the correct decoding path with larger list sizes, but, on the other hand, they degrade the performance of the inner polar code by increasing its rate [17]. However, the choice of CRC is not a subject of this study and does not influence the conclusions on the presented work.

Error-correction performance simulations are carried out with floating point precision. The applied simulation model uses Binary Phase Shift Keying (BPSK) modulation and an AWGN channel, which is commonly used in literature for the analysis of error-correction performance to enable fair comparisons of error-correcting codes. Using other modulations schemes or channel models is possible but is beyond the scope of this work. We simulate a maximum of 10^8 pseudo-random information words. A stopping condition of 10^3 erroneous decoded blocks is applied.

To show that PO-Fast-SSCL decoding matches the error-correction performance of SCL decoding, the FERs for code rates $R \in \{1/6, 1/2, 5/6\}$ of both decoding algorithms are shown in Figure 8.

In Figures 9–11, all plots exhibit the FER of LLR-based SCL decoding [17] with list sizes $L \in \{2, 4, 8, 16\}$ as reference. The curves labeled with "POS" and "ExPOS" result from POS-Fast-SSCL and ExPOS-Fast-SSCL decoding, respectively. All values in the set of $S_{\text{Rate-1}}$ belong to the corresponding value in the set of list sizes L. We omit the subscript of $S_{\text{Rate-1}}$ in the legends of the plots for lack of space. For the sets of k_c, the same applies as for the sets of $S_{\text{Rate-1}}$.

The combination of $L = 2$ and $S_{\text{Rate-1}} = 1$ is a special case, since $S_{\text{Rate-1}} = P = L - 1$, and, therefore, PO-based candidate generation directly provides the minimal number of candidates. Thus, PO-, POS- and ExPOS-based candidate generation are equal for $L = 2$.

As input to the threshold approximation for ExPOS-Fast-SSCL decoding presented in Algorithm 2, we chose the target FER $= 10^{-5}$ and $\Delta = 0.02$ dB.

In Figures 9–11, the curves plotted in violet and orange represent the FER observed for the parameter sets of $S_{\text{Rate-1}}$ being rated as insufficient and sufficient, respectively, by Algorithm 2 for POS-Fast-SSCL decoding. Accordingly, the curves plotted in red and green correspond to insufficient and sufficient values for $S_{\text{Rate-1}}$ and k_c in ExPOS-Fast-SSCL decoding. Additional FER curves are plotted in blue for $L = 8$ as corner cases with an error-correction performance loss equal to Δ at the selected target FER.

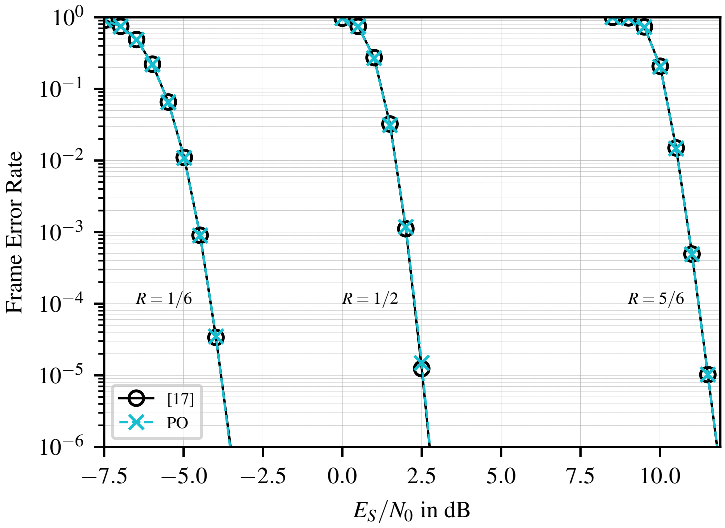


Figure 8. Error-correction performance of Polar codes with $N = 1024$ and $R \in \{1/6, 1/2, 5/6\}$, under SCL and PO-Fast-SSCL decoding with $L = 16$.

For a code rate $R = 1/6$, the FER plots for $L \in \{2, 4, 8, 16\}$ are presented in Figure 9. It can be seen that $S_{\text{Rate-1}} \in \{0, 0, 1, 1\}$ is sufficient to avoid an error-correction performance loss $> \Delta$ for the presented list sizes, respectively. Note that $S_{\text{Rate-1}} = 0$ implies that no splitting is carried out in Rate-1 nodes, and, therefore, ExPOS-Fast-SSCL decoding is not feasible. POS-Fast-SSCL decoding with $S_{\text{Rate-1}} = 0$ for $L = 8$ exhibits a loss equal to Δ at FER $= 10^{-5}$, which increases for decreasing FER. For ExPOS-Fast-SSCL decoding with $L \in \{8, 16\}$, $S_{\text{Rate-1}} = 1$ and $k_c = L$ are sufficient. This results in considering only the candidates with $x = 0$ and the one most probable candidate with $x = 1$ originating from input path $l = 0$ (see also Figure 6d).

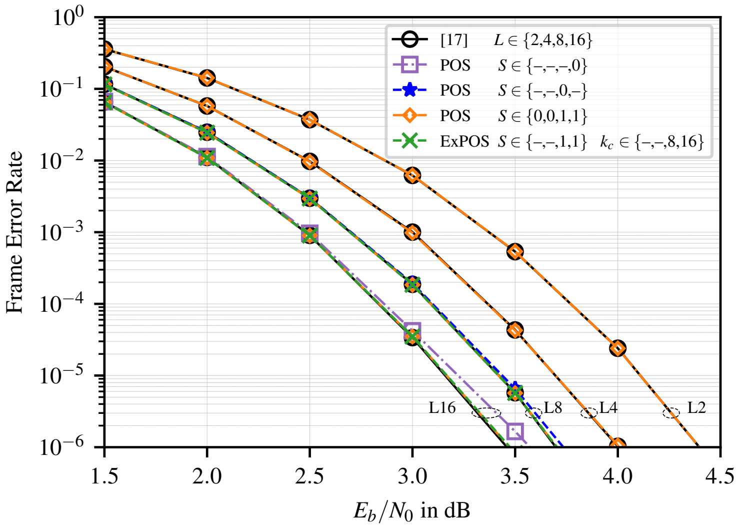


Figure 9. Error-correction performance of Polar code $N = 1024$, $R = 1/6$.

Figure 10 shows the FER plot of code rate $R = 1/2$. The values of $S_{\text{Rate-1}} \in \{1, 1, 2\}$ for $L \in \{2, 4, 8\}$ correspond to the results of [20]. For $L = 16$, we find $S_{\text{Rate-1}} = 2$ for POS-Fast-SSCL decoding to be sufficient to preserve error-correction performance at FER $= 10^{-5}$ with respect to the chosen $\Delta = 0.02$ dB. Setting $S_{\text{Rate-1}} = 1$ results in a FER performance degradation of 0.09 dB. For ExPOS-Fast-SSCL decoding with $L \in \{4, 8, 16\}$ and the appropriate values of $S_{\text{Rate-1}}$, we observe an adequate error correction with $k_c \in \{3, 4, 4\}$, respectively. Excluding more candidates by setting $k_c \in \{4, 6, 5\}$ results in an error-correction performance degradation $> \Delta$. As a corner case, we point out the combination $L = 8$, $S_{\text{Rate-1}} = 2$ and $k_c = 5$, where the performance loss is equal to Δ at FER $= 10^{-5}$. This can be considered negligible, especially when a smaller FER, e.g., 10^{-3} is targeted.

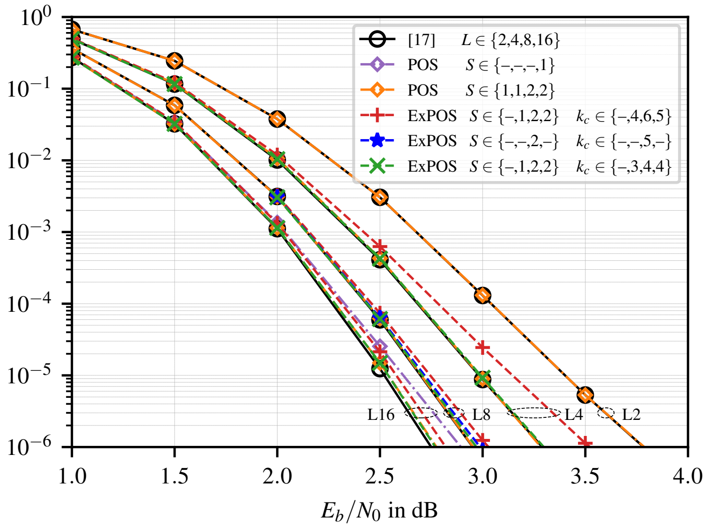


Figure 10. Error-correction performance of Polar code $N = 1024$, $R = 1/2$.

The observations for code rate $R = 5/6$, presented in Figure 11, correspond to those for code rate $R = 1/2$. Based on the sufficient $S_{\text{Rate-1}} \in \{1, 2, 2, 3\}$ in POS-Fast-SSCL decoding with $L \in \{2, 4, 8, 16\}$, it can be seen that ExPOS-Fast-SSCL decoding with $k_c \in \{3, 3, 6\}$, respectively, preserves the error-correction performance with respect to Δ. An error-correction performance degradation $> \Delta$ is observed for $k_c \in \{4, 5, 7\}$. As a corner case, $k_c = 4$ for $L = 8$ results in an FER performance loss equal to Δ. However, the error-correction degradation for the presented ExPOS cases with $L \in \{8, 16\}$ is close to Δ. Therefore, $k_c \in \{4, 5\}$ with $L = 8$ and $k_c = 7$ with $L = 16$ should also be considered for a small target FER, e.g., 10^{-3}, since these values provide an adequate error-correction performance in that FER region.

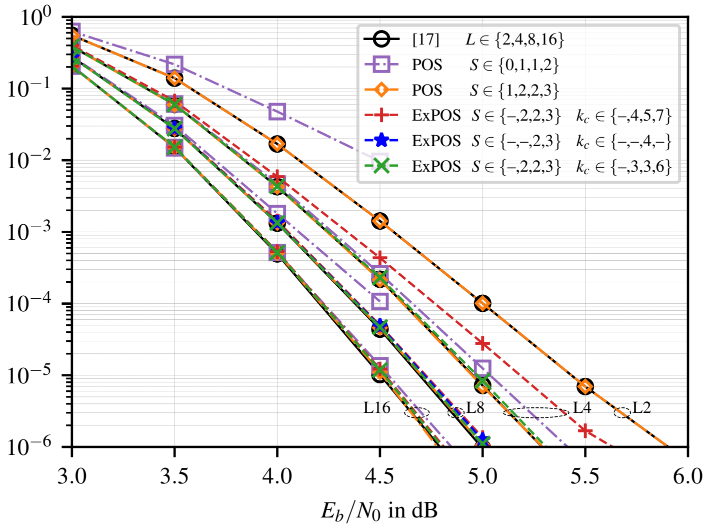


Figure 11. Error-correction performance of Polar code $N = 1024$, $R = 5/6$.

4.2. Complexity Estimations

In this section, we analyze the complexity reduction gained by ExPOS-based candidate generation compared to the serial approach of Fast-SSCL with speed optimization, i.e., with threshold $S_{\text{Rate-1}}$, as presented in [20], and to the parallel approach of MC set-based Fast-SSCL in [15]. Therefore, we evaluate the total number of considered paths L_{out} within the decoding of a Rate-1 node with $N_v > S_{\text{Rate-1}}$ and the number of comparators needed for sorting the considered paths.

For [20], L_{out} is calculated according to (5) with $P = S_{\text{Rate-1}}$ according to (31). Since the decoding of a Rate-1 node proceeds serially in [20], P intermediate sorting steps are involved to reduce the number of paths from $2L$ to L. Under the assumption that the whole decoding process is unrolled for high-throughput decoders, P sorting units need to be instantiated in a Rate-1 node. Then, the number of comparators in all the P sorters is

$$P \cdot \binom{2L}{2}. \tag{33}$$

For a fair comparison with [15], we need to adopt the threshold $S_{\text{Rate-1}}$ to their MC set-based candidate generation scheme. For $L \in \{2, 4, 8\}$, they provide details about the MC sets $\mathcal{C}_L$ that represent the indices combinations to generate the candidates in parallel by flipping the corresponding bits. The cardinality of the sets $|\mathcal{C}_L|$ is the number of candidates of each input path. These sets, restated in our candidate denomination, are $\mathcal{C}_{L=2} = \{0, 1, 2, 3\}$ with $|\mathcal{C}_2| = 4$, $\mathcal{C}_{L=4} = \{0, 1, 2, 3, 4\}$ with $|\mathcal{C}_4| = 5$ and $\mathcal{C}_{L=8} = \{0, 1, 2, 3, 4, 5, 6, 8, 9, 7, 16, 32, 64\}$ with $|\mathcal{C}_8| = 13$. Adopting $S_{\text{Rate-1}}$ to these sets corresponds to eliminating all values that have a 1 in positions $\geq S_{\text{Rate-1}}$ of their binary expansion. Considering $S_{\text{Rate-1}} = 1$ as an example, we obtain $\mathcal{C}_{S=1,L=2} = \{0, 1\}$ with $|\mathcal{C}_{1,2}| = 2$. Consequently, for [15] with threshold adoption, we obtain

$$L_{\text{out,MC}} = L \cdot |\mathcal{C}_{S,L}|. \tag{34}$$

In the parallel implementation of this work, the number of considered paths in Rate-1 nodes in ExPOS-Fast-SSCL decoding, analog to (19) and (30), is

$$L_{\text{out,ExPOS}} = \sum_{l=0}^{L-1} \sum_{\substack{j=0 \\ m_{l,j} < \theta_j}}^{2^P-1} 1. \tag{35}$$

Due to the parallel processing in [15] and in this work, only one bigger sorter is needed, whose number of comparators is

$$\binom{L_{\text{out}}}{2}, \tag{36}$$

where L_{out} is set to $L_{\text{out,MC}}$ and $L_{\text{out,ExPOS}}$, respectively.

Table 1 presents the calculated values of the number of paths according to (5), (34) and (35), and the values of the number of comparators according to (33) and (36). For the comparison, we selected different combinations of L, $S_{\text{Rate-1}}$ and k_c, for which the error-correction performance is close to the one of original SCL decoding as shown in Section 4.1. We observe reductions in the number of visited paths from 14.6 % to 53.1 % compared to [20] and 25.0 % to 56.3 % compared to [15] with threshold. In terms of the number of comparators, the savings range from 17.3 % to 72.6 % and 46.4 % to 81.7 %, compared to [20] and [15] with threshold, respectively.

4.3. Implementation Results

Unrolled and fully pipelined SCL decoder Application-Specific Integrated Circuit (ASIC) implementations achieve high throughput [12]. Therefore, the depth first traversal of the PFT is implemented in an unrolled manner. Each vector function, executed by a node of the PFT, corresponds to an instantiated computational kernel. Register balancing reduces the amount of required memory in such decoders and, thus, improves area and energy efficiency.

Table 1. Comparison of number of paths and comparators.

L	$S_{\text{Rate-1}}$	k_c	Number of Paths			Reduction of Paths [%] Related to		Number of Comparators			Reduction of Comparators [%] Related to	
			ExPOS Equation (35)	[20] Equation (5)	[15] △ Equation (34)	[20]	[15] △	ExPOS Equation (36)	[20] Equation (33)	[15] △ Equation (36)	[20]	[15] △
2	1	2	3	4	4	25.0	25.0	3	6	6	50.0	50.0
4	1	3	6	8	8	25.0	25.0	15	28	28	46.4	46.4
	2 ⋆	3	8	12	16	33.3	50.0	28	56	66	50.0	57.6
8	1 ⋆	8	9	16	16	43.8	50.0	36	120	120	70.0	70.0
	2	3	17	24	32	29.2	46.9	136	240	496	43.3	72.6
	2	4	15	24	32	37.5	53.1	105	240	496	56.3	78.8
	2	5	14	24	32	41.7	56.3	91	240	496	62.1	81.7
16	1 ⋆	16	17	32	- ⋄	46.9	- ⋄	136	496	- ⋄	72.6	- ⋄
	2 ⋆	4	41	48	- ⋄	14.6	- ⋄	820	992	- ⋄	17.3	- ⋄
	3 ⋆	6	33	64	- ⋄	48.4	- ⋄	528	1488	- ⋄	64.5	- ⋄
	3 ⋆	7	30	64	- ⋄	53.1	- ⋄	435	1488	- ⋄	70.8	- ⋄

△ [15] with threshold adoption. ⋆ not given in [20], but considered here on the basis of simulation results. ⋄ not enough details given in [15] to compute this value.

The architectures of the decoders presented in this section are automatically generated by the framework presented in [12] and implemented in a 28 nm FD-SOI CMOS technology under worst case Process, Voltage and Temperature (PVT) conditions (125 °C, 0.9 V for timing, 1.0 V for power). Synthesis is performed with the Synopsys Design Compiler, and Placement and Routing (PAR) is carried out with the Synopsys IC-Compiler. Simulations of the post-PAR netlists are executed for verification and the generation of the stimuli for power estimation. Power numbers are calculated with back-annotated wiring data at an SNR of $E_b/N_0 = 4\,\text{dB}$. A constraint of 64 Gbit/s coded throughput was set for all designs. A quantization of 6 bit is used for channel values and internal LLRs, PMs are quantized with 8 bit.

Table 2 shows the post-PAR results of decoder implementations with $N = 128$, $R = 1/2$ and $L \in \{2, 4, 8\}$. ExPOS-Fast-SSCL and PO-Fast-SSCL decoder implementations are presented in comparison with the implementations of [12]. As described in Section 4.1, for $L = 2$, PO- and ExPOS-based candidate generation is equal. Thus, only one decoder implementation with $L = 2$ is listed as a ExPOS-Fast-SSCL decoder in Table 2. The parameters $S_{\text{Rate-1}}$ and k_c for the ExPOS-Fast-SSCL decoders were determined according to Algorithm 2. The values for $S_{\text{Rate-1}}$ in the columns of PO-Fast-SSCL decoders and [12] equal $L - 1$, related to (4).

The PO-Fast-SSCL and ExPOS-Fast-SSCL decoder architectures with $L \in \{2, 4\}$ correspond exactly to the ones presented in [12], where the sub-architectures of the Rate-1 nodes with their dedicated sorters were replaced. For the ExPOS-Fast-SSCL decoders with $L = 8$, the restriction $N_{v,\text{Rate-1}_{\max}} = 4$ for Rate-1 nodes from [12] can be dropped, since the number of generated candidates is bounded to $L_{\text{out}} = 15$ (see (35) and Table 1). Thus, the effect of enhanced tree pruning additionally improves the implementation results. This also holds for the PO-Fast-SSCL decoder implementation with $L = 8$, which exhibits a maximal $L_{\text{out}} = 59$ for Rate-1 nodes according to (30).

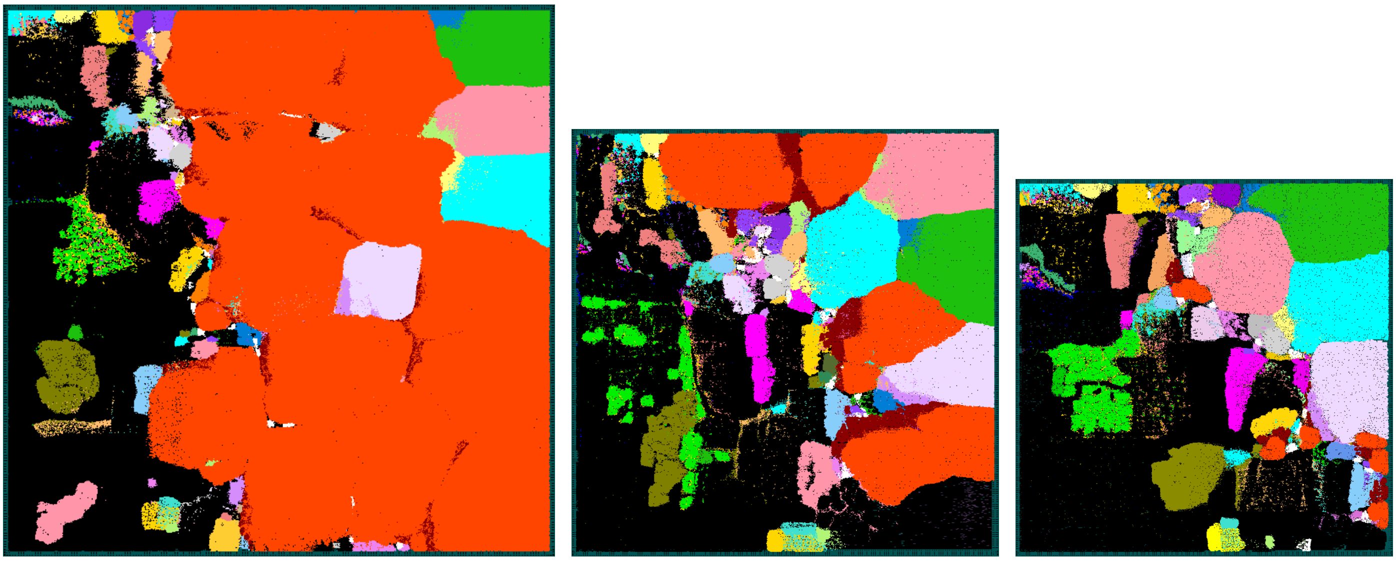
It can be observed that PO-Fast-SSCL and ExPOS-Fast-SSCL decoding improve the area efficiency and energy efficiency of the decoder implementations. With increasing list size, the impact increases. Since ExPOS-Fast-SSCL decoding exhibits the largest improvements with a negligible error-correction performance loss, we focus on these decoders. The area efficiency for $L \in \{2, 4, 8\}$ increases by $\{4.2\%, 10.6\%, 158.8\%\}$, respectively. The according energy efficiency values are improved by $\{9.0\%, 11.1\%, 62.5\%\}$.

Table 2. Results of Fast-SSCL decoders for Polar codes with $N = 128$, $K = 64 + 6$, and $C = 6$.

	ExPOS			PO		[12]		
L	2	4	8	4	8	2	4	8
$S_{\text{Rate-1}}$	1	2	2	3	7	1	3	7
k_c	2	3	4	-	-	-	-	-
$N_{v,\text{Rate-1}_{\max}}$	-	-	-	-	-	-	-	4
Frequency [MHz]	502	503	500	503	442	502	499	418
Throughput [Gbps]	64	64	64	64	57	64	64	53
Latency [ns]	41.8	63.7	90.1	65.5	108.5	41.8	64.1	141.3
Area [mm^2]	0.22	0.51	1.47	0.55	1.89	0.23	0.56	3.15
Area Efficiency [Gbps/mm^2]	**295**	**125**	**44**	**117**	**30**	**283**	**113**	**17**
Power Total [W]	0.23	0.50	1.51	0.54	1.82	0.25	0.56	3.34
Energy Efficiency [pJ/bit]	**3.52**	**7.78**	**23.59**	**8.31**	**32.18**	**3.87**	**8.75**	**62.46**

The layouts of the three decoders with $L = 8$ presented in Table 2 are shown in Figure 12. The scaling of the figures corresponds to the area values of the decoders. The coloring is consistent for all three layouts: Computational kernels (f-, g- and h-functions, Rate-0, REP and SPC nodes, their dedicated sorters) are represented by different colors. The delay line memory and CRC update functions are colored in black and white, respectively. Rate-1 nodes and their dedicated sorters are highlighted in dark-red and orange-red, respectively.

The number of Rate-1 nodes and their dedicated sorters decreases comparing the implementation of [12] (Figure 12a) with PO-Fast-SSCL and ExPOS-Fast-SSCL (Figure 12b,c) decoder implementations. This is due to enhanced tree pruning, enabled by being able to drop the restrictions on $N_{v,\text{Rate-1}_{\max}}$. Comparing Figure 12a,b, the effect of reducing L_{out} of the Rate-1 nodes by PO-based candidate generation is visible in the decrease in the area occupied by each Rate-1 node sorter, colored in orange-red. By ExPOS-based candidate generation, this effect is intensified and can easily be seen comparing the layouts in Figure 12b,c.



(**a**) Fast-SSCL: 3.15 mm^2 (**b**) PO-Fast-SSCL: 1.89 mm^2 (**c**) ExPOS-Fast-SSCL: 1.47 mm^2

Figure 12. Layouts of (**a**) Fast-SSCL decoder [12], (**b**) PO-Fast-SSCL decoder, and (**c**) ExPOS-Fast-SSCL decoder with $L = 8$ for Polar code $\mathcal{P}(128, 70)$ and $C = 6$; computational kernels are represented by different colors, delay line memory is colored in black; Rate-1 nodes and their dedicated sorter units are highlighted in dark-red and orange-red, respectively; all layouts have the same scaling.

5. Conclusions

In this work, we have proven that the upper bound for the number of candidates in the decoding of Rate-1 nodes required to preserve error-correction performance in Fast-SSCL decoders can be decreased. We described a partial order for the PMs of all possible candidates and used it to propose an algorithm for the generation of these required candidates. The presented upper bound for the number of candidates is significantly smaller than the state-of-the-art upper bound.

For practical decoder implementations, we introduced an extended threshold for the candidate selection, which reduces the number of paths by up to 53.1%. The resulting ExPOS-Fast-SSCL decoding exhibits a minor error-correction performance degradation, which can be tuned in trade-off with the candidate reduction. Unrolled decoder hardware implementations for high throughput benefit from these reductions, which was shown by a set of decoder implementations for $N = 128$ and $L \in \{2, 4, 8\}$ in a 28 nm FD-SOI CMOS technology. The highest improvements of 158.8% regarding the area efficiency and 62.5% in terms of energy efficiency were observed for $L = 8$ within the set of the presented implementations. The presented algorithms and the corresponding architectures of Rate-1 nodes enable unrolled high-throughput decoder architectures with list sizes $L > 2$ to approach reasonable implementation costs.

In future research, the proposed algorithm may be applied to other node types besides Rate-1 nodes, in order to further improve Fast-SSCL decoder implementations. Moreover, the results may also be combined with other complexity reduction mechanisms or be used in other decoder architectures like partially unrolled decoders for different target platforms, e.g., Field Programmable Gate Arrays (FPGAs).

Author Contributions: Conceptualization, L.J., C.K., O.G., T.V. and N.W.; methodology, L.J.; software, L.J., C.K. and O.G.; validation, L.J., C.K. and O.G.; formal analysis, L.J., C.K. and O.G.; investigation, L.J.; resources, L.J.; data curation, L.J.; writing—original draft preparation, L.J., C.K., O.G., T.V. and N.W.; visualization, L.J.; supervision, T.V. and N.W.; project administration, T.V. and N.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by the German Federal Ministry for Economic Affairs and Energy (BMWi) under grant ZF4013407PR7 and the German Federal Ministry of Education and Research (BMBF) under grant 16KIS1185 (FunKI).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors would like to thank Jhon Jimenez and Timo Lehnigk-Emden, Creonic GmbH, 67655 Kaiserslautern, for helpful discussions on the relations between PMs, on which the presented findings in Section 3.1 are based.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arikan, E. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073. [CrossRef]
2. Alamdar-Yazdi, A.; Kschischang, F.R. A Simplified Successive-Cancellation Decoder for Polar Codes. *IEEE Commun. Lett.* **2011**, *15*, 1378–1380. [CrossRef]
3. Sarkis, G.; Giard, P.; Vardy, A.; Thibeault, C.; Gross, W.J. Fast Polar Decoders: Algorithm and Implementation. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 946–957. [CrossRef]
4. Afisiadis, O.; Balatsoukas-Stimming, A.; Burg, A. A Low-Complexity Improved Successive Cancellation Decoder for Polar Codes. In Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 2116–2120. [CrossRef]
5. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [CrossRef]
6. Niu, K.; Chen, K. CRC-Aided Decoding of Polar Codes. *IEEE Commun. Lett.* **2012**, *16*, 1668–1671. [CrossRef]
7. Zhang, Z.; Zhang, L.; Wang, X.; Zhong, C.; Poor, H.V. A Split-Reduced Successive Cancellation List Decoder for Polar Codes. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 292–302. [CrossRef]

8. Wang, X.; Zhang, H.; Li, J.; Bao, X.; Xie, K. An Improved Path Splitting Decision-Aided SCL Decoding Algorithm for Polar Codes. *IEEE Commun. Lett.* **2021**, *25*, 3463–3467. [CrossRef]
9. Peng, Y.; Bao, J.; Liu, X. An Improved Path Splitting Strategy on Successive Cancellation List Decoder for Polar Codes. *IET Commun.* **2021**, *15*, 1198–1209. [CrossRef]
10. Sarkis, G.; Giard, P.; Vardy, A.; Thibeault, C.; Gross, W.J. Fast List Decoders for Polar Codes. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 318–328. [CrossRef]
11. Hashemi, S.A.; Condo, C.; Gross, W.J. Fast Simplified Successive-Cancellation List Decoding of Polar Codes. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6. [CrossRef]
12. Kestel, C.; Johannsen, L.; Griebel, O.; Jimenez, J.; Vogt, T.; Lehnigk-Emden, T.; Wehn, N. A 506Gbit/s Polar Successive Cancellation List Decoder with CRC. In Proceedings of the 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, London, UK, 31 August–3 September 2020; IEEE: London, UK, 2020; pp. 1–7. [CrossRef]
13. Lee, H.Y.; Pan, Y.H.; Ueng, Y.L. A Node-Reliability Based CRC-Aided Successive Cancellation List Polar Decoder Architecture Combined With Post-Processing. *IEEE Trans. Signal Process.* **2020**, *68*, 5954–5967. [CrossRef]
14. Kim, D.; Kim, S.; Kim, I.; Kim, M.G. Reliability Based Candidate Selection of List Decoding for Polar Code. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021; pp. 1–5. [CrossRef]
15. Zhao, Y.; Yin, Z.; Wu, Z.; Xu, M. Minimum-Combinations Set-Based Rate-1 Decoder for Fast List Decoding of Polar Codes. *IEEE Commun. Lett.* **2021**, *25*, 3185–3189. [CrossRef]
16. Leroux, C.; Raymond, A.J.; Sarkis, G.; Gross, W.J. A Semi-Parallel Successive-Cancellation Decoder for Polar Codes. *IEEE Trans. Signal Process.* **2013**, *61*, 289–299. [CrossRef]
17. Balatsoukas-Stimming, A.; Parizi, M.B.; Burg, A. LLR-Based Successive Cancellation List Decoding of Polar Codes. *IEEE Trans. Signal Process.* **2015**, *63*, 5165–5179. [CrossRef]
18. Hashemi, S.A.; Condo, C.; Gross, W.J. Simplified Successive-Cancellation List Decoding of Polar Codes. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 815–819. [CrossRef]
19. Hashemi, S.A.; Condo, C.; Gross, W.J. A Fast Polar Code List Decoder Architecture Based on Sphere Decoding. *IEEE Trans. Circuits Syst. Regul. Pap.* **2016**, *63*, 2368–2380. [CrossRef]
20. Hashemi, S.A.; Condo, C.; Gross, W.J. Fast and Flexible Successive-Cancellation List Decoders for Polar Codes. *IEEE Trans. Signal Process.* **2017**, *65*, 5756–5769. [CrossRef]
21. Wu, W.; Siegel, P.H. Generalized Partial Orders for Polar Code Bit-Channels. *IEEE Trans. Inf. Theory* **2019**, *65*, 7114–7130. [CrossRef]
22. Schürch, C. A Partial Order for the Synthesized Channels of a Polar Code. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 220–224. [CrossRef]
23. Tal, I.; Vardy, A. How to Construct Polar Codes. *IEEE Trans. Inf. Theory* **2013**, *59*, 6562–6582. [CrossRef]