Study on Successive Cancellation Decoding of Polar Codes

Dan Le, Qiong Li, Member, IEEE, Xianyan Wu, Member, IEEE, and Xiamu Niu*, Member, IEEE

Abstract—Thanks to the property of provably capacity-achieving, the recently-discovered polar codes are being taken many attentions. The Successive Cancellation (SC) is the first and widely known decoding for polar codes. In the paper, we study the decoding beginning with its recursive formula. Compared with previous works, our study is more strict and many fresh results are presented which would be helpful to all kinds of improvements on the SC decoding. Besides, based on them a non-recursive tree SC decoding is proposed whose space complexity and latency are both $O(N)$. Simulation results on the binary erasure channel verify its correctness.

Index Terms—Polar codes, successive cancellation decoding, non-recursion, partial sum, feedback part.

I. INTRODUCTION

Polar codes introduced by Arikan [1] is the first provable class of channel capacity-achieving codes since Shannon presented the noisy channel coding theorem [2]. The standard SC decoding algorithm presented in [1] includes $N \log_2 N$ node processors and requires $N \log_2 N$ memory elements to decode one codeword, where $N$ is the code length. On the other hand, the codes ask for a large code length to achieve the channel capacity, which leads to a real challenge in some cases.

Aiming at the problem, research has progressed along two lines: (i) improving the error performance of polar codes at moderate lengths (ii) reducing SC implementation complexity. Examples of the first line include: list decoding[3], stack decoding[4], belief propagation [5], [6], non-binary polar codes[7], [8], the maximum likelihood decoding based on sphere decoding[9], and using more complex construction methods[10]. Along the second line, many efficient SC decoding implementations have been developed. For instance, [11] presented a particular scheduling enabling resource sharing to reduce complexity, [12] proposed a semi-parallel decoder for resource sharing and processor sharing at the cost of a small increase in latency, and [13] showed a overlapped architectures to decrease decoding latency. In this paper, we focus our study on the second line. Inferring from the recursive formula, many meaningful results are strictly drawn. Based on them, a non-recursive tree SC decoding is proposed. Compared with the empirical results in most of previous references, our work is more strict and more integrated.

In Section II we briefly review the traditional SC decoding and introduce some notations. Section III and Section IV separately study on two parts of SC decoder. Based on the works, a non-recursive tree SC decoding is shown in Section V whose simulation results and analysis is given by Section VI. Finally, some conclusions are drawn in Section VII.

II. TRADITIONAL SC DECODING AND SOME NOTATIONS

Polar codes are usually denoted as $(N, K, A, u_A)$, where $N$ and $K$ represent the lengths of code bits and information bits respectively, $A$ and $A^c$ represent the indices sets of information bits and frozen bits respectively, and $u_A = \{u_i | i \in A^c\}$ which is a subvector of the input vector $u_1^N$. Let the corresponding output vector through channel $W_N$ is $y_1^N$ with conditional probability $W_N (y_1^N | u_1^N)$. Then the SC decoder successively estimates the transmitted bits as follows,

$$\hat{u}_i = \begin{cases} u_i, & \text{if } i \in A^c \\ 0, & \text{if } i \notin A^c \text{ and } L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 1 \\ 1, & \text{if } i \notin A^c \text{ and } L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) < 1 \end{cases}$$

where the likelihood ratio (LR) is

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) = \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)}$$

which can be straightforwardly calculated using the recursive formula (4) according to [1]. This recursion can be continued down to block length 1 where

$$L_N^{(1)}(y_1) = \frac{W(y_1|0)}{W(y_1|1)}$$

which can be computed immediately according to the output vector $y_1^N$.

Since the logarithm likelihood ratio (LLR) is always superior to the LR in terms of hardware utilization, computational complexity, and numerical stability when the decoding algorithm is performed in real domain [13], it is employed frequently by kinds of decodings. The corresponding recursive formula is rewrittten as (5), which can be reduced to the Min-Sum update rule (6) for the purpose of the simplify implementation of the hyperbolic tangent function and its inverse function in (5). Although the above three recursive formulas are different, the dependencies are the same. Without loss of generality, we employ the recursive formula (4) to depict our idea. The results in this paper are also compatible with the recursive formula (5) and (6).

It is obvious that in (4) the calculation of an LR at length $N$ is reduced to the calculation of two LRs at length $N/2$. Besides, there are three different operations on the decoded bits $\hat{u}_1^{i-1}$, which are the XOR between the subvectors with odd
indices and even indices, the EXTRATION of the subvector with even indices, and the EXTRATION of the last element. For brevity, they are denoted as the functions $p$, $q$ and $r$, then

\[ p(\hat{u}_1^i) = \hat{u}_{2i/2}^i \oplus \hat{u}_{1i/2}^i \]
\[ q(\hat{u}_1^i) = \hat{u}_{2i/2}^i \]
\[ r(\hat{u}_1^i) = \hat{u}_i. \]

(7)

Some other notations used in the paper are as follows,

- $N = 2^n$ is the code length
- $\{a, \ldots, b\}$ represents the integers ranging from $a$ to $b$
- &\text{ is bitwise logical AND operator, } \sim \text{ is bitwise logical NOT operator.}$

Furthermore, it can be observed that in (4) there are two different data types. One is LRs, $L_N^{(i)}(\cdot, \cdot)$. The other is partial sums, $\hat{u}_{i-1}$, which is also named feedback part in some references[13] and will become much complexity during recursion. We will separately study on these two data types in the following two sections.

### III. STUDY ON LRS

According to the recursive formula (4), it can be seen that there is a total LRs at length $N/2^k$ of $N$ and a total LRs at the whole SC decoding of $N (\log_2 N + 1)$. In this section, we will show that these LRs can be stored in $2N - 1$ LR memory elements (LME) by the time multiplexing, where $2^k$ LMEs for the $N$ LRs at length $N/2^k$. Beside, an important parameter is found, which is the key to implement a non-recursive SC decoding and can also be useful in many other cases, such as determining which of (4a) and (4b) is employed to calculate LRs and determining when partial sums are reset.

#### A. Storied by the Time Multiplexing

**Lemma 1** The calculation of an LR at length $N$, $L_N^{(i)}(y_1^N, \hat{u}_{i-1}^1)$, depends on the calculation of $2^k$ LRs at length $N/2^k (1 \leq k \leq n)$, i.e.

\[ L_N^{(i/2)}(y_1^{N/2}, \hat{u}_{1,0}^{i/2}, \hat{u}_{1,e}^{i/2}, \hat{u}_{i-1}^i), \quad 1 \leq i \leq 2^k, \]

where $h_{j,k}$ is a composite function of functions $p$ and $q$. If the binary expansion of the integer $j - 1$ is $b_k b_{k-1} \cdots b_2 b_1$, then $h_{j,k} = f_k \circ f_{k-1} \circ \cdots \circ f_2 \circ f_1$, where

\[ f_a = \begin{cases} p, \text{ when } b_{k-a+1} = 0 \\ q, \text{ when } b_{k-a+1} = 1 \end{cases}, \quad 1 \leq a \leq k. \]

(8)

**Proof:**

**Basis Step:** We start with the case $k = 1$. According to (4), the calculation of an LR at length $N$, $L_N^{(i)}(y_1^N, \hat{u}_{i-1}^1)$, depends on the calculation of two LRs at length $N/2$ as follows,

\[ L_N^{(i/2)}(y_1^{N/2}, \hat{u}_{1,0}^{i/2}, \hat{u}_{1,e}^{i/2}, \hat{u}_{i-1}^i) \]
\[ L_N^{(i/2)}(y_1^{N/2}, \hat{u}_{1,0}^{i/2}, \hat{u}_{1,e}^{i/2}, N - \hat{u}_{i-1}^i) \]
\[ \text{i.e.} \]
\[ L_N^{(i/2)}(y_1^{N/2}, p(\hat{u}_{i-1}^i)), \quad L_N^{(i/2)}(y_1^{N/2}, q(\hat{u}_{i-1}^i)). \]

Let $h_{1,1} = p$ and $h_{2,1} = q$, then the Lema for the case $k = 1$ is true.

**Inductive Step:** Now we assume the truth of the case $k = m$. That is that the calculation of an LR at length $N$, $L_N^{(i)}(y_1^N, \hat{u}_{i-1}^1)$, depends on the calculation of $2^m$ LRs at length $N/2^m$ as follows,

\[ L_{N/2^m}^{(i/2^m)}(y_1^{N/2^m}, h_{j,m}(\hat{u}_{i-1}^1)), \quad 1 \leq j \leq 2^m, \]

where

\[ h_{j,m} = f_m \circ f_{m-1} \circ \cdots \circ f_2 \circ f_1 \]

and

\[ f_a = \begin{cases} p, \text{ when } b_{m-a+1} = 0 \\ q, \text{ when } b_{m-a+1} = 1 \end{cases}, \quad 1 \leq a \leq m. \]

(10)

Here $b_m b_{m-1} \cdots b_2 b_1$ is the binary expansion of the integer $j - 1$. According to the recursive formula (4), each item in
(9) is calculated as (12) when $[i/2^m]$ is odd, otherwise it is calculated as (13).

According to (12) and (13), it is obvious that the calculation of an LR at length $N/2^m$, $L_{N/2^m}^j(n=N/2^{m+1}, h_{j,m} \left(\tilde{u}_1^{i-1}\right))$, depends on the calculation of two LRs at length $N/2^{m+1}$ as follows,

$$L_{N/2^{m+1}}^j\left(y_{j-1}/N, N/2^{m+1}, h_{j,m+1} \left(\tilde{u}_1^{i-1}\right)\right), l \in \{2j - 1, 2j\}$$

where

$$h_{l,m+1} \left(\tilde{u}_1^{i-1}\right) = \left\{ \begin{array}{ll} p \left(h_{j,m} \left(\tilde{u}_1^{i-1}\right)\right), & \text{when } l = 2j - 1 \\ q \left(h_{j,m} \left(\tilde{u}_1^{i-1}\right)\right), & \text{when } l = 2j \end{array} \right. \quad (14)$$

That is

$$h_{j,m+1} = \left\{ \begin{array}{ll} p \circ f_m \circ \cdots \circ f_2 \circ f_1, & \text{when } l = 2j - 1 \\ q \circ f_m \circ \cdots \circ f_2 \circ f_1, & \text{when } l = 2j \end{array} \right.$$ \quad (15)

Meanwhile, since $j \in \{1, 2, \cdots, 2^m\}$, it can be inferred that $l \in \{1, 2, \cdots, 2^{m+1}\}$ and

$$l - 1 = \left\{ \begin{array}{ll} b_n b_{n-1} \cdots b_1 0, & \text{when } l = 2j - 1 \\ b_n b_{n-1} \cdots b_1 1, & \text{when } l = 2j \end{array} \right. \quad (16)$$

Hence the Lema for the case $k = m + 1$ is true.

Consequently, by the Principle of Finite Induction, the Lema is proved.

Lemma 1 indicates that the differences between the two $2^k$ LRs at length $N/2^k$ depended by two LRs at length $N$ are $[i/2^k]$ and $h_{j,k} \left(\tilde{u}_1^{i-1}\right)$. It is obvious that

$$[i/2^k] = m_i, \text{ for } i \in \{(m - 1) 2^k + 1, \cdots, m 2^k\}.$$ \quad (17)

Then what about $h_{j,k} \left(\tilde{u}_1^{i-1}\right)$?

**Lemma 2**: $h_{j,k} \left(\tilde{u}_1^{i-1}\right)$, $1 \leq j \leq 2^k$, $1 \leq k \leq n$, is a vector with the length of $[i/2^k]$, denoted as $\left(v_1, v_2, \cdots, v_{[i/2^k]}\right)$. Any element $v_a (1 \leq a \leq [i/2^k])$ is estimated as follows,

$$v_a = \bigoplus_{d \in D_{j,k,a}} u_d,$$

where

$$D_{j,k,a} = \{d \mid d = (a - 1) \cdot 2^k + 1 + c_k c_{k-1} \cdots c_1\},$$ \quad (18)

$$c_i = \left\{ \begin{array}{ll} 0, & \text{when } b_{k-i} + 1 = 0 \\ 1, & \text{when } b_{k-i} + 1 = 1 \end{array} \right.$$ \quad (19)

Proof:

**Basis Step**: We start with the case $k = 1$. In this case $j \in \{1, 2\}$, so we only need to consider $h_{1,1} \left(\tilde{u}_1^i\right)$ and $h_{2,1} \left(\tilde{u}_1^i\right)$.

Since

$$h_{1,1} \left(\tilde{u}_1^i\right) = p \left(\tilde{u}_1^i\right) = \tilde{u}_1^{2[i/2]} \oplus \tilde{u}_1^{2[i/2]}$$

$$h_{2,1} \left(\tilde{u}_1^i\right) = q \left(\tilde{u}_1^i\right) = \tilde{u}_1^{2[i/2]}$$ \quad (20)

it is obvious that their length are both $[i/2]$ and for any given $1 \leq a \leq [i/2]$

$$D_{1,1,a} = \{2a - 1, 2a\} = \{d \mid d = (a - 1) \cdot 2 + 1 + ?\}$$

$$D_{2,1,a} = \{2a\} = \{d \mid d = (a - 1) \cdot 2 + 1 + 1\}.$$

Hence the Lema for the case $k = 1$ is true.

**Inductive Step**: Now we assume the truth of the case $k = m$. Then we have

$$h_{j,m} \left(\tilde{u}_1^i\right) = (v_1, v_2, \cdots, v_{n_1}), n_1 = [i/2^m],$$

and $u_a = \bigoplus_{d \in D_{j,m,a}} u_d$, where

$$D_{j,m,a} = \{d \mid d = (a - 1) \cdot 2^m + 1 + c_m c_{m-1} \cdots c_1\},$$

$$c_i = \left\{ \begin{array}{ll} 0, & \text{when } b_{m-i} + 1 = 0 \\ 1, & \text{when } b_{m-i} + 1 = 1 \end{array} \right.$$ \quad (21)

and $b_n b_{n-1} \cdots b_1$ is the binary expansion of the integer $j - 1$.

(a) For any given $1 \leq l \leq 2^{m+1}$, when it is odd, according to (14) we have

$$h_{l,m+1} \left(\tilde{u}_1^i\right) = p \left(\tilde{u}_1^i\right) = w_{l+2}^1.$$ \quad (22)

Then $n_2 = [n_1/2] = [i/2^{m+1}]$. For any element $w_a (1 \leq a \leq n_2)$, we have

$$w_a = \bigoplus_{d \in D_{j,m,2a-1} \cup D_{j,m,2a}} \tilde{u}_d.$$ \quad (23)
and
\[ D_{j,m,2a-1} \cup D_{j,m,2a} = \{ d \mid d = (2a-2) \cdot 2^m + 1 + c_m c_{m-1} \cdots c_1 \} \cup \{ d \mid d = (2a-1) \cdot 2^m + 1 + c_m c_{m-1} \cdots c_1 \} \]
\[ \{ d \mid d = (a-1) \cdot 2^{m+1} + 1 + 0c_m c_{m-1} \cdots c_1 \} \cup \{ d \mid d = (a-1) \cdot 2^{m+1} + 1 + 1c_m c_{m-1} \cdots c_1 \} \]
\[ \{ d \mid d = (a-1) \cdot 2^{m+1} + 1 + ?c_m c_{m-1} \cdots c_1 \} , \]
so
\[ D_{l,m+1,a} = \{ d \mid d = (a-1) \cdot 2^{m+1} + 1 + ?c_m c_{m-1} \cdots c_1 \} . \]

Hence the Lema for the case that \( k = m + 1 \) and \( l \) is odd is true.

(b) With the same virtue, the case that \( k = m + 1 \) and \( l \) is even can also be proved.

Hence, Combining (a) and (b), the Lema is inferred to be true for the case \( k = m + 1 \).

Consequently, by the Principle of Finite Induction, the Lema is proved.

According to Lema 2 we have
\[ h_{j,k} \left( \hat{u}_1^{i-1} \right) = h_{j,k} \left( \hat{u}_1^{(i-1)/2^k} \right) \cdot 2^k . \] (19)

Hence
\[ h_{j,k} \left( \hat{u}_1^{i-1} \right) \equiv h_{j,k} \left( \hat{u}_1^{(m-1)/2^k} \right) , \]
for \( \forall i \in \{ (m-1) \cdot 2^k + 1, \ldots, m2^k \} . \) (20)

Combining (16), (20) and Lema 1, it is concluded that the \( 2^k \)
LRs at length \( N/2^k \) are shared by the \( 2^k \) LRs at length \( N \), which is stated as Theorem 1.

**Theorem 1** For any given \( 1 \leq k \leq n \), the \( 2^k \) LRs at length \( N/2^k \),
\[ \begin{align*}
L^{(m)}_{N/2^k} \left( y \right)_{(j-1)-N/2^k+1}^{j-N/2^k} h_{j,k} \left( \hat{u}_1^{(m-1)/2^k} \right) , \\
1 \leq j \leq 2^k .
\end{align*} \] (21)
are shared by the \( 2^k \) LRs at length \( N \),
\[ \begin{align*}
L^{(i)}_N \left( y \right)_{N, \hat{u}_1^{i-1}}^{N, \hat{u}_1^{i-1}} , \\
(m-1) \cdot 2^k < i \leq m2^k ,
\end{align*} \]
where \( 1 \leq m \leq N/2^k \).

Let \( h_{1,0} \) be an identity function. Then the discussions above of \( h_{j,k} (1 \leq k \leq n) \) are also true for \( k = 0 \). Based on \( h_{j,k} \), the general form of LRs in SC decoder is expressed as follows,
\[ \begin{align*}
L^{(m)}_{N/2^k} \left( y \right)_{(j-1)-N/2^k+1}^{j-N/2^k} h_{j,k} \left( \hat{u}_1^{(m-1)/2^k} \right) , \\
1 \leq j \leq 2^k , \\
1 \leq m \leq N/2^k , \\
0 \leq k \leq n,
\end{align*} \] (22)
which are in total of \( N (\log_2 N + 1) \). These LRs are at \( n + 1 \)
different lengths and there are \( N \) LRs at each length. Since SC decoder successively calculates \( \hat{u}_1 (1 \leq i \leq N) \), just one LR at length \( N \) is activated per time. According to Lema 1, the number of activated LRs at length \( N/2^k \) per time is \( 2^k \). Therefore the \( N \) LRs at length \( N/2^k \) \( (0 \leq k \leq n) \) can be stored in \( 2^k \) LMEs by the time multiplexing, denoted as \( LR_{j,k} \), \( 1 \leq j \leq 2^k \). In this case, a total of \( 2N - 1 \) LMEs is needed to store all the LRs. These LMEs are depended in the tree form as shown in Figure 1 where \( N = 8 \). The Dec. with frame is a decision according to (1). The white cycles represent the channel LRs estimated by (3) which remain invariant once being calculated, and the gray cycles represent the other LRs which are computed over and over again according to their two children nodes and (4a) or (4b) determined by the value of \( i \). Theorem 1 indicates that the smaller \( k \) means the more frequency recalculation of \( LR_{j,k} \), \( 1 \leq j \leq 2^k \).

**B. Non-recursive Calculation**

**Theorem 2** In SC decoder, when the \( i^{th} \) LR at length \( N \) is being calculated, only the LRs at length \( N, N/2, \ldots, N/2^{z_i} \) should be updated, while the LRs at length \( N/2^{z_i+1}, N/2^{z_i+2}, \ldots, 1 \) can be shared, where \( z_i \) is the number of the consecutive zeros in the end of the binary expansion of the integer \( i-1 \). It is noted that \( z_i = n \) when \( i = 1 \). Specifically, the updates can be performed beginning with the LRs at length \( N/2^{z_i} \), and in sequence till ending with the LRs at length \( N \) by the formula (4).

**Proof:** According to the Theorem 1, it is obvious that the \( i^{th} \) and \((i-1)^{th}\) LR at length \( N \) can not share the \( 2^k \) LRs at length \( N/2^k \) if and only if there exists an integer \( m \) satisfying that \( i-1 = m \cdot 2^k \). Since \( \hat{z}_i \) is the number of the consecutive zeros in the end of the binary expansion of the integer \( i-1 \), we have
\[ i-1 = \begin{cases} 0, & \text{if } i = 1 \\ m_o \cdot 2^{\hat{z}_i}, & \text{otherwise} \end{cases} \] (23)
where \( m_o \) is odd.

(a) If \( i = 1 \), then for all \( 0 \leq k \leq n = z_i \) there exists the integer 0 satisfying that \( i-1 = 0 \cdot 2^k \), which means in this case all the LRs at all the lengths should be estimated. Since the \( 2^{z_i} \) LRs at length 1 are channel LRs, (3) indicates that they can be estimated immediately. Then the \( 2^{z_i-1} \) LRs at length 2 can be directly computed according to (4). Afterwards the LRs at length \( N/2^{z_i-2}, N/2^{z_i-3}, \ldots, N \) can be updated in sequence according to (4).
(b) Otherwise, for any given integer \( k > z_i \), there does not exist any integer \( m \) satisfying that \( i - 1 = m \cdot 2^k \), which means that the \( i^{th} \) and \( (i - 1)^{th} \) LR at length \( N \) share the \( 2^k \) LRs at length \( N/2^k \) when \( k > z_i \). On the other hand, for any given integer \( k \leq z_i \), there exists the integer \( m_i = 2z_i - k \cdot m_o \) satisfying that \( i - 1 = m_i \cdot 2^k \), which means that the \( i^{th} \) and \( (i - 1)^{th} \) LR at length \( N \) can not share the \( 2^k \) LRs at length \( N/2^k \) when \( k \leq z_i \). Therefore only the LRs at length \( N, N/2, \ldots, N/2^{z_i} \) should be updated. Since the shared \( 2^{z_i+1} \) LRs at length \( N/2^{z_i+1} \) has been calculated during the estimation of the \( (i - 1)^{th} \) LR at length \( N \), the \( 2^{z_i} \) LRs at length \( N/2^z \) can be directly computed according to (4). Afterwards the LRs at length \( N/2^{z_i-1}, N/2^{z_i-2}, \ldots, N \) can be updated in sequence according to (4).

Combining (a) and (b), the Theorem is proofed. 

In the traditional SC decoder, in order to estimate \( \hat{u}_i \) the computation of LR at length \( N \) is firstly activated, which in turn activates the LRs at length \( N/2 \). The LRs at length \( N/2 \) then activate the LRs at length \( N/4 \) which activate the LRs at length \( N/8 \). The process continues till the LRs at certain length assumed as \( N/2^k \) have been estimated. Then the computation is sequentially performed back from length \( N/2^{k-1} \) to length \( N \), and \( \hat{u}_i \) is determined according to (1). Different from the traditional SC decoding, Theorem 2 shows a non-recursive method. In fact, the recursion in the traditional SC decoding is replaced by the calculation of \( z_i \), which can be estimated efficiently by Bit Twiddling Hacks. Furthermore, we will illustrate other roles of \( z_i \) in the following.

C. Determine the Calculation Formula

Theorem 1 demonstrates that \( LR_{j,k}, 1 \leq j \leq 2^{k} \), store \( 2^k \) LRs at length \( N/2^k \) shown as (21) whose calculation formulas are chosen according to \( m \). When \( m \) is odd, (4a) is employed to calculate these \( 2^k \) LRs. When \( m \) is even, (4b) is used. The parity of \( m \) can be determined by the following three methods.

(a) Firstly, since \( m = \lceil i/2^k \rceil \), we have

\[
i - 1 \in \left\{ (m - 1) \cdot 2^k, \ldots, (m - 1) \cdot 2^k + 2^k - 1 \right\},
\]

which means that \( m \) is odd when the \( (k + 1)^{th} \) least significant bit (LSB) of \( i - 1 \) is 0, and \( m \) is even when the \( (k + 1)^{th} \) LSB of \( i - 1 \) is 1. That is

\[
m = \begin{cases} 
    \text{odd}, & \text{when } (i - 1) \& 2^k = 0 \\
    \text{even}, & \text{when } (i - 1) \& 2^k = 1 
\end{cases}.
\]

(b) Secondly, \( m \) ranges from 1 to \( N/2^k \), increases by 1 once these \( 2^k \) LRs are recalculated, and its parity is hereby changed. Thus a flag bit can be employed to record the change, which determines whether \( m \) is even or not, is initialized as FALSE and flipped whenever these \( 2^k \) LRs are recalculated. The Discriminant formula is given by

\[
m = \begin{cases} 
    \text{odd}, & \text{when flag = FALSE} \\
    \text{even}, & \text{when flag = TRUE} 
\end{cases}.
\]

(c) Thirdly, based on (23)

\[
m = \left\lfloor \frac{i}{2^k} \right\rfloor = \begin{cases} 
    \lceil 1/2^k \rceil, & \text{if } i = 1 \\
    \left[ m_o \cdot 2^{z_i - k} + 1/2^k \right], & \text{otherwise} 
\end{cases}
\]

where \( m_o \) is odd. Obviously, if \( i = 1 \), then \( m \) is equal to 1 for all the \( k \), otherwise \( m \) is even for \( k = z_i \) and odd for \( k < z_i \). Here the case of \( k > z_i \) are not considered because the corresponding LRs can be shared according to Theorem 2 and need not be recalculated. Hence the third Discriminant formula is written as

\[
m = \begin{cases} 
    \text{even}, & \text{if } i \neq 1 \text{ and } k = z_i \\
    \text{odd}, & \text{otherwise} 
\end{cases}.
\]

To the best of our knowledge, the scheme a is mentioned in some references[12] while the other two schemes have not been stated so far. The scheme a need not extra storage but its additional calculation is dependent on the global variant \( i \). The additional calculation of scheme b is local which is not referenced to any global variants, however it need extra \( n \) bits storage. The scheme c need neither extra storage nor additional calculation when combining with proposed non-recursive tree SC decoding because the estimation of \( z_i \) can be shared. Accordingly, we adopt the scheme c in the proposed non-recursive tree SC decoding.

D. Complexity

As discussed in Section III-A, there are in total of \( N (\log_2 N + 1) \) LRs in the whole SC decoding, which can be stored in \( 2N - 1 \) LMEs by the time multiplexing. Hence the space complexity is \( O(N) \).

Let \( Z_j = \{ i | z_i = j, 1 \leq i \leq N \} \) which indicates the set of \( i (1 \leq i \leq N) \) satisfying that the number of the consecutive zeros in the end of the binary expansion of \( i - 1 \) is \( j \), then

\[
|Z_j| = \begin{cases} 
    1, & \text{if } j = n \\
    2^{n-j} - 1, & \text{if } 0 \leq j < n 
\end{cases}.
\]

In order to estimated \( \hat{u}_i \), the number of the calculations of LRs is

\[
S_i = \sum_{k=0}^{z_i} 2^k = 2^{z_i+1} - 1,
\]

according to Theorem 2. Then the total of the calculations of LRs in the whole SC decoding is

\[
\sum_{i=1}^{N} S_i = \sum_{j=0}^{n} |Z_j| (2^{j+1} - 1) = N (\log_2 N + 1).
\]

In a fully parallel environment, the \( 2^k \) LRs at length \( N/2^k \) can be calculated concurrently in one time slot because their calculations are independent. Then the time slots of calculating the total latency for the calculations of LRs is given by

\[
\sum_{i=1}^{N} (z_i + 1) = \sum_{j=0}^{n} |Z_j| \cdot (j + 1) = 2N - 1.
\]

1http://graphics.stanford.edu/~seander/bithacks.html. Please note the "∼" in the website when clicking it, which may need to manually retype.
IV. Study on Partial Sums

A partial sum, in fact, is the mod-2 sum of some estimated \( \hat{u}_i \) and can be stored by one bit. Since the general form of all LRs in SC decoding is as (22), the general form of partial sums is

\[
r(\hat{u}_{j,k}(2m-1)2^k) = \begin{cases} \hat{u}_1 & 1 \leq j \leq 2^k, 1 \leq m' \leq N/2^{k+1}, \ 0 \leq k < n, \end{cases}
\]

which according to (4). It can be seen that there is a total partial sum of \((N\log_2 N)/2\). In this section, we will show that all the partial sums can be stored in \(N-1\) bits by the time multiplexing and discuss the calculations of them.

A. Stored by the Time Multiplexing

According to Lemma 2, \(r(\hat{u}_{j,k}(2m-1)2^k))\) can be estimated as follows,

\[
r(\hat{u}_{j,k}(2m-1)2^k)) = v_{2m-1} = \sum_{d \in D_{j,k,2m-1}} \hat{u}_d.
\]

Since \(0 \leq c_k \leq 1 \leq 2^k-1\), it can be inferred that \(D_{j,k,2m-1} \subseteq \{(2m-2)2^k+1, \ldots, (2m-1)2^k\}, 1 \leq j \leq 2^k\).

Hence \(r(\hat{u}_{j,k}(2m-1)2^k))\), \(1 \leq j \leq 2^k\), are calculated during the estimation of \(\hat{u}_{j,k}(2m-2)2^k+1\), and are used during the estimation of \(\hat{u}_{j,k}(2m-1)2^k+1\). The union of the calculation range and the use range

\[
\{(2m-2)2^k+1, \ldots, (2m-1)2^k\} \cup \\
\{(m-1)2^k+1, \ldots, m'2^k\}
\]

is named as the working range of these \(2^k\) partial sums.

Likewise, the working range of \(r(\hat{u}_{j,k}(2m+1)2^k))\), \(1 \leq j \leq 2^k\), is

\[
\{(2m' + 1)2^k+1, \ldots, (2m'+1)2^k\} \cup \\
\{(m'+1)2^k+1, \ldots, (m'+2)2^k\}
\]

It is obvious that the working ranges in (30) and (31) are disjoint, which means that the two \(2^k\) partial sums can be stored in the same memory by the time multiplexing. Thus the \(N/2\) partial sums

\[
r(\hat{u}_{j,k}(2m-1)2^k)) \in \{1 \leq j \leq 2^k, 1 \leq m' \leq N/2^{k+1}\}
\]

can be stored in \(2^k\) bits by the time multiplexing. For brevity, the \(2^k\) storage bits are denoted as \(r_{j,k}\) \((1 \leq j \leq 2^k)\). Then the storage for all \((N\log_2 N)/2\) partial sums just is \(N-1\) bits. Different from the tree dependency of LRs, these \(r_{j,k}\) \((1 \leq j \leq 2^k, 0 \leq k < n)\) are calculated independently. Each of them is attached to an unique LR expect channel LRs as illustrated in Figure 2.

B. Calculation

Different from the calculation of LRs which is one time, the estimation of partial sums is the cumulative mod-2 sum, which leads the estimation of \(r_{j,k}\) should consist of two components. One is to reset \(r_{j,k}\) at the appropriate time. According to the working ranges in (30) and (31), it can be inferred that the reset happens at the moment when \(\hat{u}_{m',2^k+1}\) has been estimated while \(\hat{u}_{m',2^k+1+1}\) is being calculated. In other words, the reset component can be depicted as

\[
r_{j,k} = \begin{cases} 0, & i \text{ if } (i-1) \& 2^{k+1} = 0, \\
\hat{u}_{j,k}, & \text{otherwise} \end{cases} \quad (32)
\]

where \(i\) is the index of the being estimated \(\hat{u}_i\). (32) indicates that all the \(k+1\) LSBs of \(i-1\) are zeros. In the view of \(z_i\), (32) can be rewritten as follows

\[
r_{j,k} = \begin{cases} 0, & i \text{ if } k < z_i, \\
\hat{u}_{j,k}, & \text{otherwise} \end{cases} \quad (33)
\]

The second component is the cumulative mod-2 sum of certain \(\hat{u}_i\). Since \(r_{j,k}\) stores the \(N/2^{k+1}\) partial sums

\[
r(\hat{u}_{j,k}(2m-1)2^k)) \in \{1 \leq m' \leq N/2^{k+1}\}
\]

It can be inferred that \(i \in D_{j,k}\) if and only if both the following two conditions are standing.

\[
(i-1) \& 2^k = 0, \quad (35)
\]
where the function $\text{Reverse}(\cdot)$ converts the most significant bits (MSB) of a given integer to its LSBs, and vice versa. (35) indicates that the $(k+1)^{th}$ LSB of $i-1$ must be 0, and (36) indicates that $c_i$ must meet (17). Hence the cumulative sum component can be summarized as

$$r_{j,k} = \begin{cases} \hat{u}_i, & \text{if both (35)(36) are satisfied} \\ r_{j,k}, & \text{otherwise} \end{cases}$$

(37)

C. Complexity

As discussed in Section IV-A, there are in total of $(N\log_2 N)/2$ partial sums in the whole SC decoding, which can be stored in $N – 1$ bits by the time multiplexing. Hence the space complexity is $O(N)$.

According to (34) and (17), it can be inferred that the number of mod-2 sum operations of $r_{j,k}$, $1 \leq j \leq 2^k$, is

$$\sum_{j=1}^{2^k} |D_{j,k}| = \frac{N}{2^{k+1}} \sum_{i=0}^{k} 2^i \binom{k}{i} = \frac{3^k N}{2^{k+1}}.$$  

Then the total calculations of partial sums during the whole SC decoding are

$$\sum_{k=0}^{n-1} \sum_{j=1}^{2^k} |D_{j,k}| = N \left( \frac{3}{2} \log_2 N - 1 \right).$$  

(38)

Figure 3 shows the comparison of the total calculations between LRs as (28) and partial sums as (38). It can be seen that the total calculations of partial sums is far more than that of LRs when $N$ is large.

![Figure 3](image-url)

Fig. 3. The comparison of total calculations between LRs and partial sums.

In a fully parallel environment, all $r_{j,k}$, $1 \leq j \leq 2^k$, $0 \leq k < n$, can be calculated concurrently in one time slot because their calculations are independent. Then the total latency for the calculations of partial sums is $N$ which is less than that of LRs given by (29).

V. NON-REFUGOUS TREE SC DECODING

Based on the results in Section III and Section IV, a non-recursive tree SC decoding is presented in Algorithm 1. Some steps in the algorithm consist of more than one actions which are ended with semicolon, for instance Step 3 includes two actions. The actions in the same step can be performed concurrently. Hence it is obvious that the latency of the algorithm is $O(N)$.

Algorithm 1 The non-recursive tree SC decoding

1: calculate in parallel the $2^k$ channel LRs at length 1. Each of them is stored as $LR_{j,n}$, $1 \leq j \leq 2^k$, and is estimated according to (3) and $y_j$;
2: for $k = n - 1$; $k ≥ 0$; $k -= 0$
3: calculate in parallel the $2^k$ LRs at length $N$. Each of them is stored as $LR_{j,k}$, $1 \leq j \leq 2^k$, and is estimated according to (4a), $LR_{2j-1,k+1}$ and $LR_{2j,k+1}$; Reset in parallel the $2^k$ partial sums $r_{j,k}$, $1 \leq j \leq 2^k$
4: end for
5: determine $\hat{u}_i$ according to (1) and $LR_{1,0}$; calculate $z_2$
6: for $i = 2$; $i ≤ N$; $i += 1$
7: perform in parallel the cumulative sum component of all partial sums $r_{j,k}$, $1 \leq j \leq 2^k$, $0 \leq k < n$, according to (37);
8: calculate in parallel the $2^i$ LRs at length $\frac{N}{2^i}$ to update $LR_{2i,k}$, $1 \leq j \leq 2^i$. Each of them is estimated according to (4b), $LR_{2j-1,z_i+1}$, $LR_{2j,z_i+1}$ and $r_{z_i}$;
9: for $k = z_i - 1$; $k ≥ 0$; $k -= 0$
10: calculate in parallel the $2^k$ LRs at length $\frac{N}{2^k}$ to update $LR_{j,k}$, $1 \leq j \leq 2^k$. Each of them is estimated according to (4a), $LR_{2j-1,k+1}$ and $LR_{2j,k+1}$; Reset in parallel the $2^k$ partial sums $r_{j,k}$, $1 \leq j \leq 2^k$
11: end for
12: determine $\hat{u}_i$ according to (1) and $LR_{1,0}$; calculate $z_{i+1}$;
13: end for

Besides, some additional notes about the algorithm are listed as follows,

(a) The calculation of LRs is non-recursive and the dependency between them is a complete binary tree as shown in Figure 1, which are the reasons that the algorithm is called non-recursive tree SC decoding. The non-recursion instead of recursion can decrease the latency of SC decoding to some extend.

(b) It is straightforward to determine which formula of (4a) and (4b) should be employed to estimate a LR, which need neither extra storage nor additional calculation. This straightforward determination can also decrease the latency of SC decoding to some extend.

(c) The application of time division multiplexing ensures that the space complexity of LRs and partial sums are both $O(N)$. Moreover, Step 8 indicates that the dependency between partial sums and LRs expect channel LRs is biunique as shown in Figure 2.

(d) The reset component of partial sums is divided into several parts to keep pace with the calculation of LRs. Each part
resets $2^k$ partial sums one time as illustrated in Step 3 and Step 10. The component can also be performed one time in parallel as required.

(e) $z_i$ is a helpful parameter. Firstly, it is the key to implement a non-recursive SC decoding as shown in Theorem 2. Secondly, it leads to a straightforward method to select the calculation formula of a LR as shown in (26). Third, it indicates the implementation of the reset component of partial sums as shown in (33).

VI. EXPERIMENTAL RESULTS AND ANALYSIS

To verify the correctness of proposed non-recursive tree SC decoding, we implement it on GPU. Here the adopted recursive formula is the Min-Sum update rule as (6) instead of (4). An example at block length $2^{10}$ on a binary erasure channel (BEC) with erasure probability 0.5 is shown as Figure 4. It can be seen that the simulation results are bounded by the upper bound and lower bound of the probability of block error, which demonstrates the correctness of proposed non-recursive tree SC decoding. On the other hand, the phenomenon that the simulation results are approaching the upper bound when the rate become smaller may be caused by the employment of Min-Sum update rule instead of LRs.

![Rate versus reliability for non-recursive tree SC decoding at block length $2^{10}$ on a BEC with erasure probability 0.5.](image)

VII. CONCLUSION

Deriving from the recursive formula of SC decoding, we separately study in details on LRs and partial sums which are the two important parts of SC decoding. A non-recursive tree SC decoding is proposed based on these study, whose space complexity and latency are both $O(N)$. It can also decrease the latency of SC decoding to some extent. Compared with the empirical results in most of previous references, this paper is not only more strict but also presents many meaningful and fresh results. For instance, (a) a non-recursive implementation of SC decoding is first proposed. (b) A straightforward method to determine the calculation formula of a LR is presented which need neither extra storage nor additional calculation. (c) An integrated method for the estimation of partial sums is given which consists of the reset component and the cumulative sum component. Our work would be helpful to all kinds of improvements on the SC decoding.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Grant Number: 61301099, 61361166006) and the Fundamental Research Funds for the Central Universities (Grant Number: HIT. NSRIF. 2013061, HIT. KISTP. 201416, HIT. KISTP. 201414).

REFERENCES

[1] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” Information Theory, IEEE Transactions on, vol. 55, no. 7, pp. 3051–3073, 2009.
[2] C. E. Shannon, “A mathematical theory of communication,” The Bell Technical Journal, vol. 27, no. 4, pp. 379–423, 1948.
[3] I. Tal and A. Vardy, “List decoding of polar codes,” in Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on. IEEE, 2011, pp. 1–5.
[4] K. Niu and K. Chen, “Stack decoding of polar codes,” Electronics letters, vol. 48, no. 12, pp. 695–697, 2012.
[5] N. Hassami, S. B. Korada, and R. Urbanke, “Performance of polar codes for channel and source coding,” in Information Theory, 2009. ISIT 2009. IEEE International Symposium on. IEEE, 2009, pp. 1488–1492.
[6] E. Arikan, “A performance comparison of polar codes and reed-muller codes,” IEEE Commun. Lett, vol. 12, no. 6, pp. 447–449, 2008.
[7] R. Mori and T. Tanaka, “Non-binary polar codes using reed-solomon codes and algebraic geometry codes,” in Information Theory Workshop (ITW), 2010 IEEE. IEEE, 2010, pp. 1–5.
[8] S. Kahraman, E. Viterbo, and M. E. Celebi, “Folded successive cancellation decoding of polar codes,” in Communications Theory Workshop (AusCTW), 2014 Australian. IEEE, 2014, pp. 57–61.
[9] S. Kahraman and M. E. Celebi, “Code based efficient maximum-likelihood decoding of short polar codes,” in Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on. IEEE, 2012, pp. 1967–1971.
[10] S. B. Korada, E. Sasoglu, and R. Urbanke, “Polar codes: Characterization of exponent, bounds, and constructions,” Information Theory, IEEE Transactions on, vol. 56, no. 12, pp. 6253–6264, 2010.
[11] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, “Hardware architectures for successive cancellation decoding of polar codes,” in Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011, pp. 1665–1668.
[12] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, “A semi-parallel successive-cancellation decoder for polar codes,” Signal Processing, IEEE Transactions on, vol. 61, no. 2, pp. 289–299, 2013.
[13] C. Zhang and K. K. Parhi, “Low-latency sequential and overlapped architectures for successive cancellation polar decoder,” Signal Processing, IEEE Transactions on, vol. 61, no. 10, pp. 2429–2441, 2013.