Low-Latency Successive-Cancellation Polar Decoder Architectures Using 2-Bit Decoding

Bo Yuan, *Student Member, IEEE*, and Keshab K. Parhi, *Fellow, IEEE*

***Abstract*—Polar codes have emerged as important error correction codes due to their capacity-achieving property. Successive cancellation (SC) algorithm is viewed as a good candidate for hardware design of polar decoders due to its low complexity. However, for (n, k) polar codes, the long latency of SC algorithm of $(2n - 2)$ is a bottleneck for designing high-throughput polar decoder. In this paper, we present a novel reformulation for the last stage of SC decoding. The proposed reformulation leads to two benefits. First, critical path and hardware complexity in the last stage of SC algorithm is significantly reduced. Second, 2 bits can be decoded simultaneously instead of 1 bit. As a result, this new decoder, referred to as *2b-SC decoder*, reduces latency from $(2n - 2)$ to $(1.5n - 2)$ without performance loss. Additionally, *overlapped-scheduling*, *precomputation* and *look-ahead* techniques are used to design two additional decoders referred to as *2b-SC-Overlapped-scheduling decoder* and *2b-SC-Precomputation decoder*, respectively. All three architectures offer significant advantages with respect to throughput and hardware efficiency. Compared to known prior least-latency SC decoder, the 2b-SC-Precomputation decoder has 25% less latency. Synthesis results show that the proposed (1024, 512) 2b-SC-Precomputation decoder can achieve at least 4 times increase in throughput and 40% increase in hardware efficiency.**

***Index Terms*—Look-ahead, polar codes, overlapped scheduling, precomputation, successive cancellation, 2-bit decoder.**

I. INTRODUCTION

POLAR codes, as the first provable capacity-achieving codes over binary-input discrete memoryless channel (B-DMC) [1], have received significant attention among various forward error correction (FEC) codes. Due to their explicit structure and low-complexity encoding/decoding scheme, polar codes have emerged as one of the most important codes in coding theory. To date, many efforts have addressed several theoretical aspects of polar codes [2]–[9]. However, with the exception of [10]–[14], [19], not many publications have considered the VLSI design of polar decoders. In [10], an FPGA implementation of polar decoder based on the Belief-propagation (BP) algorithm was reported. Although BP decoder has particular advantages in parallel design, due to the requirement of large number of processing elements (PEs), the BP decoder is not attractive for practical applications. In [11], [12], [19], successive cancellation (SC) polar decoders were presented. These low-complexity architectures are suitable for area-stringent applications; however, due to the inherent serial nature of the SC algorithm, these SC decoders fall short due to long latency and low throughput. In [13], [14], a precomputation scheme was applied to the SC algorithm, which succeeded in reducing the overall latency from $(2n-2)$ to $(n-1)$. However, considering the penalty of increased hardware, the SC-Precomputation decoder does not show significant improvement with respect to hardware efficiency.

This paper makes three key contributions and presents three new SC decoder architectures. First, a novel reformulation of the last stage of the original SC decoder allows two bits to be decoded in the same clock cycle, which leads to a reduction in latency from $(2n - 2)$ to $(1.5n - 2)$. This architecture is referred to as the *2b-SC decoder*. Second, the use of *overlapped scheduling* technique [15] further reduces the latency to $(n-1)$. This architecture is referred to as the *2b-SC-Overlapped-scheduling decoder*. Third, the use of *precomputation* [16]–[18] and *look-ahead* [17], [18] techniques further reduces the latency from $(n-1)$ to $(3n/4-1)$. This architecture is referred to as the *2b-SC-Precomputation decoder*. Note that, among all known prior SC decoder architectures, the least achievable latency is $(n-1)$. Thus, the $(3n/4-1)$ latency of the proposed 2b-SC-Precomputation decoder is the least among all known architectures.

The remainder of the paper is organized as follows. Section II presents a brief review of the polar codes. In Section III, the reformulation for the last stage of SC decoding is developed. Then, based on this reformulation, the 2b-SC algorithm is presented. Section IV develops three different novel SC architectures based on this new algorithm. Hardware analysis and comparison are discussed in Section V. Section VI draws conclusions.

II. REVIEW OF POLAR CODES

A. Encoding Procedure

The name "polar" codes is derived from the phenomenon of channel polarization. As proved in [1], with efficient construction approach, the reliability of decoded bits will be polarized based on their different positions at the source data. Therefore, an efficient polar-based transmitter can be constructed based on the following principles: 1) sending required information bits at "good" positions, which can strongly guarantee the reliability of transmission; and 2) sending fixed "0" at "bad" positions, since after the transmission any decoded bits at these "bad" positions are highly unreliable. In [1], those "0" bits are called "frozen" bits since these are fixed and their positions are known at both the encoder and the decoder. Similarly, we call the non-frozen information bits as "free" bits in this paper. Accordingly, an

Manuscript received January 20, 2013; revised May 08, 2013, June 18, 2013; accepted July 09, 2013. Date of publication October 17, 2013; date of current version March 25, 2014. This work was supported in part by Broadcom Corporation. This paper was recommended by Associate Editor H.-C. Chang.

Authors are with Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, Minneapolis, MN 55455 USA (e-mail: yuan0103@umn.edu; parhi@umn.edu).

Digital Object Identifier 10.1109/TCSI.2013.2283779

1549-8328 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

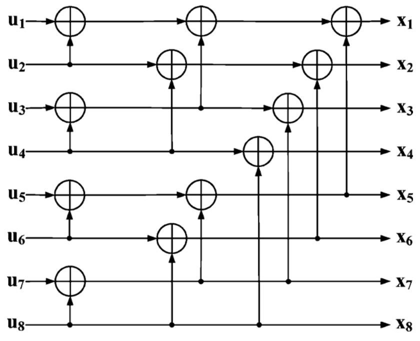


Fig. 1. An implementation of polar encoder with $n = 8$.

(n, k) polar code contains k information ("free") bits and $(n-k)$ "frozen" bits.

In general, an (n, k) polar code can be constructed from the original k-bit information message $\boldsymbol{c} = c_1^k \triangleq (c_1, c_2, \dots c_k)$ in two steps. If we denote the set of positions of frozen and free bits as $\mathbf{frz} = \{frz_1, \dots, frz_{n-k}\}$ and $\mathbf{free} = \{free_1, \dots, free_k\}$, respectively, where $1 \leq frz_i \leq n, 1 \leq free_i \leq n$, then the first encoding step is to construct an n-bit source data vector as $\boldsymbol{u} = u_1^n \triangleq (u_1, u_2, \dots u_n)$, where $u_i = c_j$, if $i = free_j$; or $u_i = 0$, if $i \in \mathbf{frz}$.

After obtaining $\boldsymbol{u}$, the second step computes the transmitted codeword $\boldsymbol{x} = x_1^n \triangleq (x_1, x_2, \dots x_n)$ by the generator matrix $\boldsymbol{G}$ [9], [10]:

$$\boldsymbol{x} = \boldsymbol{uG} \tag{1}$$

Here $\boldsymbol{G} = \boldsymbol{F}^{\otimes \boldsymbol{m}}$, where $\boldsymbol{F}^{\otimes \boldsymbol{m}}$ denotes the m-th Kronecker power of $\boldsymbol{F} = [\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}]$.

It should be noted that in some literature, the mapping from $\boldsymbol{u}$ to $\boldsymbol{x}$ is represented as $\boldsymbol{x} = \boldsymbol{uGB}$ instead of (1), where $\boldsymbol{B}$ is the bit-reverse operation. As indicated in [1], both of these two mapping approaches are equivalent and have the same performance. In this paper, we adopt (1) as the encoding equation. An example implementation for $n = 8$ polar encoder is illustrated in Fig. 1.

B. Conventional SC Decoding Algorithm

At the receiver end, corrupted by the transmission noise, the received codeword will no longer be $\boldsymbol{x}$, but change to $\boldsymbol{y} = y_1^n \triangleq (y_1, y_2, \dots y_n)$. Since the required information bits are contained in the original source data vector $\boldsymbol{u}$, the goal of polar decoding is to recover $\boldsymbol{u}$ from $\boldsymbol{y}$. In [1], it is proved that this recovery can be accomplished by the SC algorithm. With a recursive computation procedure, the SC algorithm can use the likelihood ratios (LRs) of $\boldsymbol{y}$ to output an estimated $\boldsymbol{u}$. In this paper, we denote this estimated $\boldsymbol{u}$ as $\hat{\boldsymbol{u}} = \hat{u}_1^n \triangleq (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_n)$. Here each decoded bit $\hat{u}_i$ is determined by the following decision function $\boldsymbol{h}(\cdot)$ [1]:

$$\hat{u}_i = \boldsymbol{h}(LR[\boldsymbol{y}, \hat{u}_1^{i-1}]) \tag{2}$$

where $\boldsymbol{h}(LR[\boldsymbol{y}, \hat{u}_1^{i-1}]) = 1$ if $LR[\boldsymbol{y}, \hat{u}_1^{i-1}] < 1$ and i is not frozen position; otherwise $\boldsymbol{h}(LR[\boldsymbol{y}, \hat{u}_1^{i-1}]) = 0$.

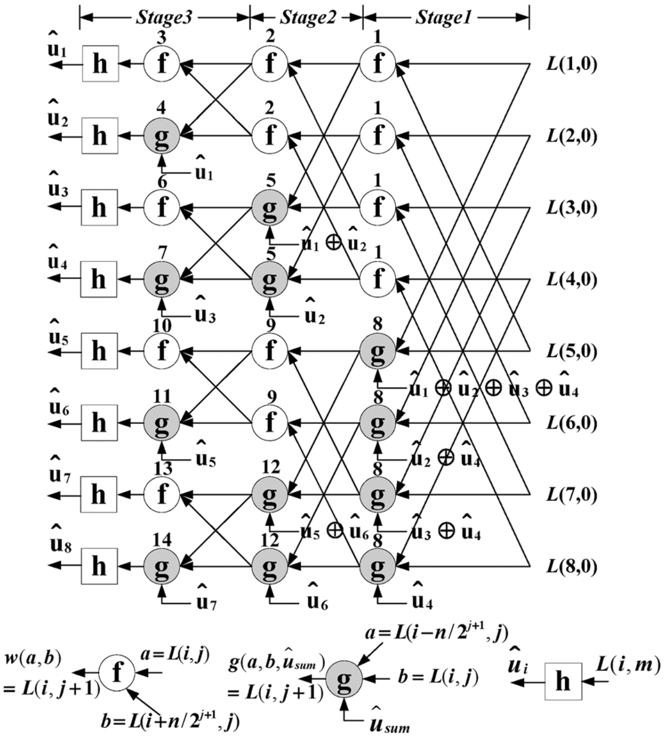


Fig. 2. The decoding procedure of conventional SC algorithm with $n = 8$.

Here $LR[\boldsymbol{y}, \hat{u}_1^{i-1}] \triangleq (W(\boldsymbol{y}, \hat{u}_1^{i-1}|u_i = 0))/(W(\boldsymbol{y}, \hat{u}_1^{i-1} |u_i = 1))$ is the LR value of the bit $\hat{u}_i$, and $W(\boldsymbol{y}, \hat{u}_1^{i-1}|u_i = s)$ is the probability that the received codeword is $\boldsymbol{y}$ and the previously decoded bits are $\hat{u}_1^{i-1} \triangleq (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_{i-1})$, given the condition that $u_i = s \in \{0, 1\}$.

From (2) it can be seen that the essence of the SC algorithm is how to determine $LR[\boldsymbol{y}, \hat{u}_1^{i-1}]$. In [1], Arıkan proposed an efficient recursive approach to compute these likelihood ratios. Fig. 2 shows the decoding procedure for an example $n = 8$ polar code. Based on the LR values of $\boldsymbol{y}$, two types of processing nodes, namely white node ($\mathbf{f}$ node) and grey node ($\mathbf{g}$ node), are employed to calculate $LR[\boldsymbol{y}, \hat{u}_1^{i-1}]$. Here $LR[\boldsymbol{y}, \hat{u}_1^{i-1}]$ in stage-3 can be calculated from the messages from stage-2, while the calculation in stage-2 needs the messages output from stage-1. Since these intermediate propagating messages are also LR values, we present a unified notation for all the LRs in this graph. The likelihood ratio output from the node at row i and stage j is denoted as $L(i, j)$. With this new notation, $LR[\boldsymbol{y}, \hat{u}_1^{i-1}]$ is now represented as $L(i, m)$, where $m = \log_2 n$. Meanwhile, the LR value for the received bit y_i can be denoted as $L\ (i, 0)$. Hence, (2) is now expressed as:

$$\hat{u}_i = \boldsymbol{h}(L(i, m)) \tag{3}$$

where $\boldsymbol{h}(L(i, m)) = 1$ if $L(i, m) < 1$ and i is not frozen position; otherwise $\boldsymbol{h}(L(i, m)) = 0$.

To calculate $L(i, m)$, in [1] Arıkan proposed to compute the following equation recursively:

$$L(i, j+1) = \begin{cases} \boldsymbol{f}(L(i, j), L(i + n/2^{j+1}, j)) & for\ f\ node \\ \boldsymbol{g}(L(i - n/2^{j+1}, j), L(i, j), \hat{u}_{sum}) & for\ g\ node \end{cases} \tag{4}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ functions were defined in [1] as:

$$\boldsymbol{f}(a,b) = \frac{1+ab}{a+b} \tag{5}$$

$$\boldsymbol{g}(a,b,\hat{u}_{sum}) = a^{1-2\hat{u}_{sum}}b. \tag{6}$$

Notice that in (6), $\hat{u}_{sum}$ is the module-2 sum of partial previous decoded bits. The term $\hat{u}_{sum}$ depicts the "successive" operation in the SC algorithm. The decision of current bit strongly depends on the estimate of previous decoded bits; therefore, the decoded bits can only be computed in a successive manner. To clearly illustrate this phenomenon, we label a specific number for each node in Fig. 2. Here each number indicates the index of the clock cycle when the corresponding node is activated. It can be seen that $\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_8$ are output from stage-3 at cycles 3, 4, 6, 7, 10, 11, 13, 14, respectively. Accordingly, this serial decoding leads to an overall latency of 14 cycles. In general, for (n,k) polar code, the decoding latency of SC decoder is $(2n-2)$.

Equations (3)–(6) describe the conventional SC decoding algorithm based on the LR representation. However, because (5) and (6) contain division and exponentiation operations, they are not attractive for hardware implementation. To solve this problem, a log-likelihood ratio (LLR)-based SC algorithm was proposed in [11] to simplify the hardware design. Accordingly, (3)–(6) in the natural domain are transformed to (7)–(11) in the logarithm domain:

$$\hat{u}_i = \boldsymbol{h}(LL(i,m)) \tag{7}$$

where $\boldsymbol{h}(LL(i,m)) = 1$ if $LL(i,m) < 0$ and i is not frozen position; otherwise $\boldsymbol{h}(LL(i,m)) = 0$.

$$LL(i,j+1) = \begin{cases} \boldsymbol{f}(LL(i,j), LL(i+n/2^{j+1},j)) & for\ f\ node \\ \boldsymbol{g}(LL(i-n/2^{j+1},j), LL(i,j), \hat{u}_{sum}) & for\ g\ node. \end{cases} \tag{8}$$

$$\boldsymbol{f}(a,b) = 2\tanh^{-1}(\tanh(a/2)\tanh(b/2)) \tag{9}$$

$$\boldsymbol{g}(a,b) = a(-1)^{\hat{u}_{sum}} + b \tag{10}$$

where LLR value is defined as $LL(i,j) \triangleq \ln(L(i,j))$.

Since (9) is still too complex for hardware design, similar to LDPC decoding, a min-sum approximation [11] can be further employed to reduce the complexity of (9):

$$\boldsymbol{f}(a,b) \approx sign(a)sign(b)\min(|a|,|b|). \tag{11}$$

In general, (7)–(11) describe the LLR version of the conventional SC algorithm.

III. The Proposed 2bit-SC Decoding Algorithm

According to [1], n, the code length of the polar code, should be large enough to guarantee the required error-correcting performance in practical applications. Since the original SC decoder requires $(2n-2)$ cycles to output a codeword, the latency with large n is not suitable for real-time high-speed applications. Therefore, design of low-latency polar decoder is an important problem to solve. In this section, using optimization at the algorithm level, we propose a novel reformulation of the last stage of the SC decoding procedure. Then, based on this reformulation, a novel *2-bit-decoding SC* (*2b-SC*) algorithm is presented. This new algorithm can decode two successive bits in the same cycle. Therefore, the latency can be reduced by 25% without any penalty on the performance or hardware complexity.

We now review the original SC algorithm under interpretation of probability. As introduced in Section II.B, the LR version of the SC algorithm is described by (3)–(6). Section III.A reviews the inherent principle of the SC algorithm in detail. This review is helpful in developing the new reduced-latency 2b-SC algorithm in Section III.B.

A. Review of SC Algorithm Under Interpretation of Probability

As indicated in [1], [9], the architectures of polar encoder (Fig. 1) and decoder (Fig. 2) can be re-defined in a unified framework. Fig. 3 illustrates this unified encoding/decoding architecture for $n=8$. Under this framework, the encoding procedure can be viewed as a left-to-right transformation from u_1^n to x_1^n. As shown in Fig. 3(a), this transformation is accomplished by computing intermediate value $q_{i,j}$. Similarly, when the probabilities of y_1^n are available at the right side of this architecture (Fig. 3(c)), the decoding procedure can be viewed as estimating those intermediate $q_{i,j}$ in the right-to-left direction. These estimated values, denoted as $\hat{q}_{i,j}$, will be finally used to calculate the leftmost $\hat{u}_1^n$, which is just the estimation of u_1^n.

Fig. 3(b) and (d) show the basic computation units of the overall architecture. For polar encoding, each unit represents an exclusive-or operation, while for decoding it represents the combination of $\boldsymbol{f}$ and $\boldsymbol{g}$ functions. When the unified architecture is in encoding phase, as shown in Fig. 3(b), it is easy to compute the outputs of the basic unit (denoted as c and d) from inputs (denoted as a and b) as:

$$c = a \oplus b \text{ and } d = b. \tag{12}$$

On the other hand, when the unified architecture is in decoding phase, since SC decoding is just the right-to-left estimation procedure for those $q_{i,j}$ (see Fig. 3(c)), we can derive the expected relationship between these estimated values in Fig. 3(d) as:

$$\hat{a} = \hat{c} \oplus \hat{d} \text{ and } \hat{b} = \hat{d}. \tag{13}$$

It should be noted that (13) can not be directly used to estimate $\hat{a}$ and $\hat{b}$. This is because the "soft" bit probability, instead of "hard" bit value, is employed in the soft-decision SC decoding. For example, in Fig. 3(d), the probability of $\hat{c}$ and $\hat{d}$ are the inputs of the basic unit to compute probability of $\hat{a}$ and $\hat{b}$. Therefore, (13) is only a "guideline" that depicts the "expected" relationship between $\hat{a}$, $\hat{b}$ and $\hat{c}$, $\hat{d}$. Next we will show how to exactly calculate the probability of $\hat{a}$ and $\hat{b}$ with the use of (13).

Now consider the probability of $\hat{a}$ denoted as $\mathrm{P}(\hat{a}=s) \triangleq \Pr(\hat{a}=s, \hat{u}_1^{i-1}|\boldsymbol{y})$ where $s \in \{0,1\}$. Notice in the case that $\hat{a}=0$, according to (13), there are two possible combinations of $\hat{c}$ and $\hat{d}$ that can make $\hat{a}$ equal to 0: $\hat{c}=0, \hat{d}=0$ or $\hat{c}=1, \hat{d}=1$.

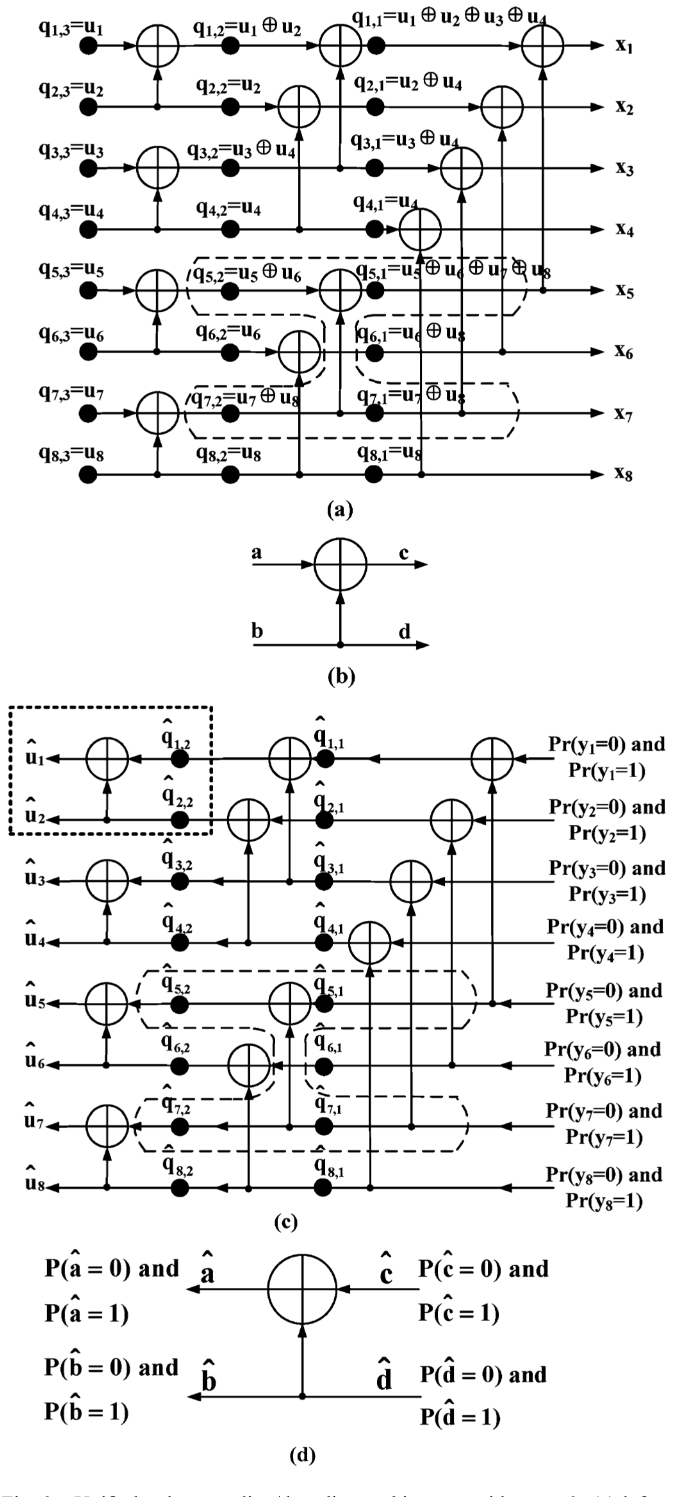


Fig. 3. Unified polar encoding/decoding architecture with $n = 8$. (a) left to right encoding procedure. (b) basic encoding computation unit. (c) right to left decoding procedure. (d) basic decoding computation unit.

Therefore, the probability for $\hat{a} = 0$ is given by:

$$\mathrm{P}(\hat{a} = 0) = \mathrm{P}(\hat{c} = 0)\mathrm{P}(\hat{d} = 0) + \mathrm{P}(\hat{c} = 1)\mathrm{P}(\hat{d} = 1). \quad (14)$$

Similarly, for the case $\hat{a} = 1$, we have

$$\mathrm{P}(\hat{a} = 1) = \mathrm{P}(\hat{c} = 0)\mathrm{P}(\hat{d} = 1) + \mathrm{P}(\hat{c} = 1)\mathrm{P}(\hat{d} = 0). \quad (15)$$

Denote the likelihood ratio of $\hat{a}$ as $LR(\hat{a})$, since $LR(\hat{a}) = (\mathrm{P}(\hat{a} = 0))/(\mathrm{P}(\hat{a} = 1))$, using (14) and (15), we have

$$\begin{aligned} LR(\hat{a}) &= \frac{\mathrm{P}(\hat{a}=0)}{\mathrm{P}(\hat{a}=1)} \\ &= \frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)}{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)} \\ &= \frac{1+LR(\hat{c})LR(\hat{d})}{LR(\hat{c})+LR(\hat{d})} \end{aligned} \quad (16)$$

where $LR(\hat{c}) = (\mathrm{P}(\hat{c} = 0))/(\mathrm{P}(\hat{c} = 1))$ and $LR(\hat{d}) = (\mathrm{P}(\hat{d} = 0))/(\mathrm{P}(\hat{d} = 1))$.

Note that (16) is just the same as (4), (5), where $LR(\hat{a}) = L(i, j+1)$, $LR(\hat{c}) = L(i, j)$ and $LR(\hat{d}) = L(i + n/2^{j+1}, j)$. This completes the derivation of the LR version of the ***f*** function based on bit probability representation and (13). Next we show how to derive LR version of the ***g*** function, which is equivalent to the calculation of probability of $\hat{b}$.

Due to the successive computation of the SC algorithm, the probability of $\hat{b}$ being 0 or 1 depends on the decision of $\hat{a}$. In the case $\hat{a} = 0$, in order to make $\hat{b}$ equal to 0, the combination of $\hat{c}$ and $\hat{d}$ can only be $\hat{c} = 0$ and $\hat{d} = 0$. Therefore, if we denote $\mathrm{P}(\hat{a} = s_1, \hat{b} = s_2) \triangleq \Pr(\hat{a} = s_1, \hat{b} = s_2, \hat{u}_1^{i-1}|\mathbf{y})$ where $s_1, s_2 \in \{0, 1\}$, then we have:

$$\mathrm{P}(\hat{a} = 0, \hat{b} = 0) = \mathrm{P}(\hat{c} = 0)\mathrm{P}(\hat{d} = 0). \quad (17)$$

Similarly, in order to make $\hat{b}$ equal to 1 under the condition that $\hat{a} = 0$, the combination of $\hat{c}$ and $\hat{d}$ can only be $\hat{c} = 1$ and $\hat{d} = 1$. Thus, we have:

$$\mathrm{P}(\hat{a} = 0, \hat{b} = 1) = \mathrm{P}(\hat{c} = 1)\mathrm{P}(\hat{d} = 1). \quad (18)$$

Based on (17) and (18), we can obtain the likelihood ratio of $\hat{b}$ for the case $\hat{a} = 0$

$$\begin{aligned} LR_{\hat{a}=0}(\hat{b}) &= \frac{\mathrm{P}(\hat{a}=0,\hat{b}=0)}{\mathrm{P}(\hat{a}=0,\hat{b}=1)} \\ &= \frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)} = LR(\hat{c})LR(\hat{d}). \end{aligned} \quad (19)$$

Now consider the probability of $\hat{b}$ when $\hat{a} = 1$. In this case, for $\hat{b}$ to be 0, $\hat{c} = 1$ and $\hat{d} = 0$. Thus:

$$\mathrm{P}(\hat{a} = 1, \hat{b} = 0) = \mathrm{P}(\hat{c} = 1)\mathrm{P}(\hat{d} = 0). \quad (20)$$

Similarly, to make $\hat{b}$ equal to 1 when $\hat{a} = 1$, the only combination of $\hat{c}$ and $\hat{d}$ is $\hat{c} = 0$ and $\hat{d} = 1$. Hence:

$$\mathrm{P}(\hat{a} = 1, \hat{b} = 1) = \mathrm{P}(\hat{c} = 0)\mathrm{P}(\hat{d} = 1). \quad (21)$$

Based on (20), (21), we have

$$\begin{aligned} LR_{\hat{a}=1}(\hat{b}) &= \frac{\mathrm{P}(\hat{a}=1,\hat{b}=0)}{\mathrm{P}(\hat{a}=1,\hat{b}=1)} \\ &= \frac{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)} = LR(\hat{c})^{-1}LR(\hat{d}). \end{aligned} \quad (22)$$

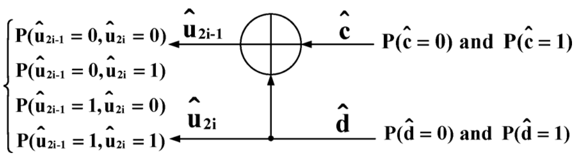

$P(\hat{u}_{2i-1}=0,\hat{u}_{2i}=0)$
$P(\hat{u}_{2i-1}=0,\hat{u}_{2i}=1)$
$P(\hat{u}_{2i-1}=1,\hat{u}_{2i}=0)$
$P(\hat{u}_{2i-1}=1,\hat{u}_{2i}=1)$
$\hat{u}_{2i-1}$
$\hat{u}_{2i}$
$\hat{c}$
$\hat{d}$
$P(\hat{c}=0)$ and $P(\hat{c}=1)$
$P(\hat{d}=0)$ and $P(\hat{d}=1)$

Fig. 4. The basic computation unit on the leftmost side (the last stage) of the overall decoding architecture.

We can derive a unified representation for LR value of $\hat{b}$ for different conditions of $\hat{a}$ from (19), (22) as:

$$LR(\hat{b}) = LR(\hat{c})^{1-2\hat{a}}LR(\hat{d}). \tag{23}$$

It can be seen that (23) is the same as (4), (6), where $LR(\hat{b}) = L(i, j+1), LR(\hat{c}) = L(i-n/2^{j+1}, j)$, $LR(\hat{d}) = L(i,j)$ and $\hat{a} = \hat{u}_{sum}$. Therefore, the LR version of $\boldsymbol{g}$ function can also be derived from (13). Note that here the equality of $\hat{a}$ and $\hat{u}_{sum}$ can be easily verified by examining the estimation characteristics of the decoding procedure. For example, in the dashed unit of Fig. 3(c), the corresponding $\hat{a}$ is $\hat{q}_{5,2}$, which is the estimation of $q_{5,2} = u_5 \oplus u_6$ (Fig. 3(a)). Therefore, $\hat{q}_{5,2}$ is equal to $\hat{u}_5 \oplus \hat{u}_6$, which is just $\hat{u}_{sum}$ of index-12 node in Fig. 2.

In this subsection, starting from (13), we have shown how the LR version of the SC algorithm can be derived under the interpretation of bit probability. This forms the basis of the new 2b-SC algorithm developed in the next subsection.

B. Proposed 2b-SC Decoding Algorithm

Section III.A discusses the general computation unit of Fig. 3(c). In this subsection, we focus on those units on the leftmost side (the last stage) of Fig. 3(c), which compute the decoded bits $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ (see Fig. 4). One of these units is highlighted with dotted rectangular line at the top left of Fig. 3(c).

According to (3), the value of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ depend not only on their LR values, but also on whether they are frozen bits or not. Therefore, since $\hat{u}_{2i-1}$ or $\hat{u}_{2i}$ can be either a free or frozen bit, we discuss four possible cases based on different frozen conditions of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$.

Case-1: None of $\hat{u}_{2i-1}$ or $\hat{u}_{2i}$ Is a Frozen Bit: In this case, since none of $\hat{u}_{2i-1}$ or $\hat{u}_{2i}$ is frozen, its value is completely determined by the probability that it is 0 or 1. Therefore, according to (17), (18), (20), (21), the probabilities of different combinations of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ can be expressed as follows:

$$\begin{aligned} P(00) &\triangleq \mathrm{P}(\hat{u}_{2i-1}=0, \hat{u}_{2i}=0) = \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)\\ P(01) &\triangleq \mathrm{P}(\hat{u}_{2i-1}=0, \hat{u}_{2i}=1) = \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\ P(10) &\triangleq \mathrm{P}(\hat{u}_{2i-1}=1, \hat{u}_{2i}=0) = \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)\\ P(11) &\triangleq \mathrm{P}(\hat{u}_{2i-1}=1, \hat{u}_{2i}=1) = \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1). \end{aligned} \tag{24}$$

Recall that in the SC algorithm, the value of the unfrozen bit $\hat{u}_j$ is determined by comparing $\mathrm{P}(\hat{u}_j = 0)$ and $\mathrm{P}(\hat{u}_j = 1)$. Equation (24) describes the joint probabilities of different combinations of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ and is the key to decoding two successive bits. Thus, $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ can be directly determined by finding the largest one among the four joint probabilities in (24).

The above hypothesis leads to two benefits. First, since a pair of bits, instead of a single bit, is determined each time, one clock cycle is saved. Considering the whole decoding procedure, this approach reduces the latency by 25%. Second, because we only need to find the largest one among four probabilities, the hardware complexity will be much less than that of the original **f** and **g** nodes. In summary, if the validity of the proposed approach can be verified, it will improve the hardware performance with respect to both latency and hardware complexity.

Motivated by the potential advantage of this hypothesis, we explore its validity. Fortunately, the proposed hypothesis is proved to be valid, and it can be verified that the decoded bit values determined by this approach are strictly equal to the outputs from the conventional SC algorithm. Therefore, we formalize this hypothesis to a proposition as follows:

Proposition 1: For arbitrary polar codes, assume the largest joint probability in (24) is $\mathrm{P}(\alpha\beta)$, and unfrozen decoded bits output from the original SC algorithm are $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$. Then $\hat{u}_{2i-1} = \alpha$ and $\hat{u}_{2i} = \beta$.

Proof: This proposition is proved in the Appendix. □

As mentioned in the above paragraph, since the proposed hypothesis has been proved, we can obtain a fast approach to simultaneously determine unfrozen $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$: *Given the probabilities of $\hat{c}$ and $\hat{d}$, once the largest joint probability $\mathrm{P}(\alpha\beta)$ in (24) is found, $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are immediately determined as $\hat{u}_{2i-1} = \alpha$ and $\hat{u}_{2i} = \beta$.*

In practical applications, likelihood ratio, instead of probability, is used for representing soft information. Therefore, the probability-based equation (24) needs to be transformed to LR-based form:

$$\begin{aligned} LR(00) &\triangleq P(00)/P(01) = LR(\hat{c})LR(\hat{d})\\ LR(01) &\triangleq P(01)/P(01) = 1\\ LR(10) &\triangleq P(10)/P(01) = LR(\hat{d})\\ LR(11) &\triangleq P(11)/P(01) = LR(\hat{c}). \end{aligned} \tag{25}$$

To avoid potential overflow and reduce computation complexity, (25) is further transformed to the logarithm domain:

$$\begin{aligned} &LLR(00) \triangleq LLR(\hat{c}) + LLR(\hat{d}) \quad LLR(01) \triangleq 0\\ &LLR(10) \triangleq LLR(\hat{d}) \quad LLR(11) \triangleq LLR(\hat{c}). \end{aligned} \tag{26}$$

In the remainder of this paper, we will use LLR-based (26) to describe the new algorithm and its hardware architectures.

Case-2: Both $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are Frozen Bits: In this case, since both of these two bits are frozen, their values can be immediately determined as 0.

Case-3: Only $\hat{u}_{2i-1}$ Is Frozen Bit: When $\hat{u}_{2i-1}$ is frozen, $\hat{u}_{2i-1} = 0$. Then, according to (23), we have

$$LR(\hat{u}_{2i}) = LR(\hat{c})^{1-2\hat{u}_{2i-1}}LR(\hat{d}) = LR(\hat{c})LR(\hat{d}). \tag{27}$$

Under the representation of LLR, (27) becomes

$$\begin{aligned} LLR(\hat{u}_{2i}) &= LLR(\hat{c})(-1)^{\hat{u}_{2i-1}} + LLR(\hat{d})\\ &= LLR(\hat{c}) + LLR(\hat{d}). \end{aligned} \tag{28}$$

Therefore, the decision scheme for $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ in this case is

$$\begin{aligned}\hat{u}_{2i-1} &= 0\\ \hat{u}_{2i} &= \begin{cases}0 & if\ LLR(\hat{c}) + LLR(\hat{d}) \geq 0\\ 1 & if\ LLR(\hat{c}) + LLR(\hat{d}) < 0.\end{cases}\end{aligned} \tag{29}$$

Case-4: Only $\hat{u}_{2i}$ Is Frozen Bit: When $\hat{u}_{2i}$ is frozen bit, $\hat{u}_{2i} = 0$. According to (16), we have

$$LR(\hat{u}_{2i-1}) = \frac{1 + LR(\hat{c})LR(\hat{d})}{LR(\hat{c})LR(\hat{d})}. \tag{30}$$

Under the representation of LLR, (30) becomes

$$\begin{aligned}&LLR(\hat{u}_{2i-1})\\ &\quad = 2\tanh^{-1}(\tanh(LR(\hat{c})/2)\tanh(LR(\hat{d})/2)).\end{aligned} \tag{31}$$

With min-sum approximation, we have:

$$\begin{aligned}LLR(\hat{u}_{2i}) \approx\ & sign(LLR(\hat{c}))sign(LLR(\hat{d}))\\ & \times \min(|LLR(\hat{c})|, |LLR(\hat{d})|).\end{aligned} \tag{32}$$

Therefore, decision scheme for $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ in this case is

$$\begin{aligned}\hat{u}_{2i} &= 0\\ \hat{u}_{2i-1} &= \begin{cases}0 & sign(LLR(\hat{c}))sign(LLR(\hat{d})) \geq 0\\ 1 & sign(LLR(\hat{c}))sign(LLR(\hat{d})) < 0.\end{cases}\end{aligned} \tag{33}$$

Summarizing the above four cases, it can be seen that $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ can always be determined at the same time. This leads to the decision scheme (Scheme-A) for the last stage of SC decoding.

With the proposed reformulated scheme, the corresponding *2b-SC algorithm* can be developed. Fig. 5 shows the corresponding 2b-SC decoding procedure with the same $n = 8$ polar code in Fig. 2. Compared with the conventional SC scheme in Fig. 2, the proposed 2b-SC algorithm replaces the $\mathbf{f}$ and $\mathbf{g}$ nodes at stage-3 with new $\mathbf{p}$ nodes. The $\mathbf{p}$ node, whose function is described in the above Scheme-A, can output the successive $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ at the same time. Therefore, the overall latency is reduced. For example, the original latency of 14 cycles in Fig. 2 is now reduced to 10 cycles in Fig. 5. Tables I and II describe the timing information of the conventional SC and 2b-SC algorithms in detail. The original SC algorithm requires $n = 8$ cycles in stage-3 to output $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$. By employing $\mathbf{p}$ nodes to compute the decoded bits, $n/2 = 4$ cycles are saved by the 2b-SC algorithm. In general, compared with the original SC algorithm, the overall latency of 2b-SC algorithm is reduced from $(2n - 2)$ to $(1.5n - 2)$.

Scheme A: Reformulation for last stage (stage-m) computation in SC decoding

```
 1: Input: Log - Likelihood ratios $LLR(\hat{c})$ and $LLR(\hat{d})$
    from stage-$(m-1)$
 2: Judge $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are frozen bits or not
 3:   Case1: None of $\hat{u}_{2i-1}$ or $\hat{u}_{2i}$ is a frozen bit
 4:     Find the largest element among
        $\{LLR(\hat{c}) + LLR(\hat{d}), 0, LLR(\hat{d}), LLR(\hat{c})\}$
 5:     If the largest element is $LLR(\hat{c}) + LLR(\hat{d})$ :
        $\hat{u}_{2i-1} = 0, \hat{u}_{2i} = 0$
 6:     If the largest element is $0$ : $\hat{u}_{2i-1} = 0, \hat{u}_{2i} = 1$
 7:     If the largest element is $LLR(\hat{d})$ : $\hat{u}_{2i-1} =$
        $1, \hat{u}_{2i} = 0$
 8:     If the largest element is $LLR(\hat{c})$ : $\hat{u}_{2i-1} =$
        $1, \hat{u}_{2i} = 1$
 9:   Case2: Both $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are frozen bits
10:     $\hat{u}_{2i-1} = 0, \hat{u}_{2i} = 0$
11:   Case3: Only $\hat{u}_{2i-1}$ is frozen bit
12:     $\hat{u}_{2i-1} = 0$
13:     $\hat{u}_{2i} = \begin{cases}0 & \text{if } LLR(\hat{c}) + LLR(\hat{d}) \geq 0\\ 1 & \text{if } LLR(\hat{c}) + LLR(\hat{d}) < 0\end{cases}$
14:   Case4: Only $\hat{u}_{2i}$ is frozen bit
15:     $\hat{u}_{2i-1} = \begin{cases}0 & sign(LLR(\hat{c}))sign(LLR(\hat{d})) \geq 0\\ 1 & sign(LLR(\hat{c}))sign(LLR(\hat{d})) < 0\end{cases}$
16:     $\hat{u}_{2i} = 0$
17: Output: $\hat{u}_{2i-1}, \hat{u}_{2i}$
```

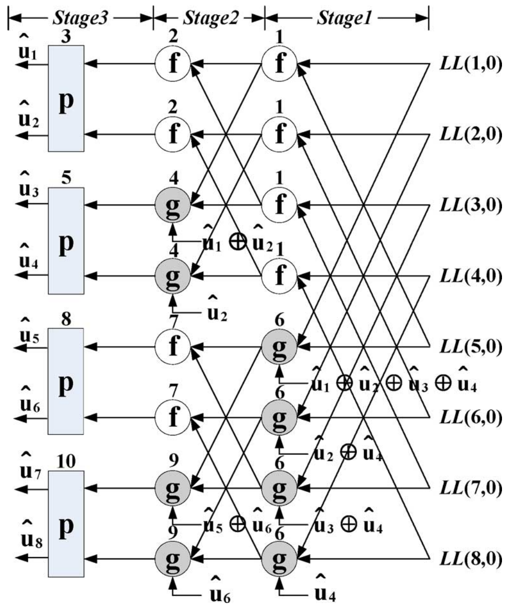


Fig. 5. The decoding procedure of 2b-SC algorithm with $n = 8$.

IV. Hardware Architectures of 2b-SC Decoder

In this section, three hardware architectures of the new 2b-SC algorithm are presented. According to Fig. 5, the overall 2b-SC decoder mainly consists of three types of processing nodes: $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{p}$ nodes. Besides these nodes, a simple partial sum generator (PSG) is also needed to generate partial sum $\hat{u}_{sum}$. Since PSG block is similar to polar encoder with simple architecture, therefore in this section we focus on the architectures of $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{p}$ nodes.

A. Processing Element (PE) for f and g Nodes

As shown in Fig. 5, $\mathbf{p}$ nodes are used in stage-m, and $\mathbf{f}$ and $\mathbf{g}$ nodes are used in other stages to calculate the propa-

TABLE I
DECODING SCHEME OF CONVENTIONAL SC [11] FOR $n = 8$ POLAR CODE

Conventional SC decoding scheme [11]														
Clock cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Stage1	f							g						
Stage2		f			g				f			g		
Stage3			f	g		f	g			f	g		f	g
Output			$\hat{u}_1$	$\hat{u}_2$		$\hat{u}_3$	$\hat{u}_4$			$\hat{u}_5$	$\hat{u}_6$		$\hat{u}_7$	$\hat{u}_8$

TABLE II
DECODING SCHEME OF 2B-SC ALGORITHM FOR $n = 8$ POLAR CODE

2-bit-decoding SC (2b-SC) scheme										
Clock cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stage1	f					g				
Stage2		f		g			f		g	
Stage3			p		p			p		p
Output			$\hat{u}_1$ & $\hat{u}_2$		$\hat{u}_3$ & $\hat{u}_4$			$\hat{u}_5$ & $\hat{u}_6$		$\hat{u}_7$ & $\hat{u}_8$

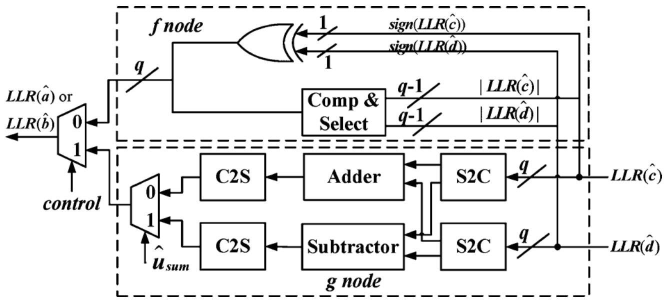


Fig. 6. The architecture of PE for **f** and **g** nodes.

gated LLR values. For simplicity of hardware design, the functions of **f** and **g** nodes are always implemented by unified processing elements (PEs) [11], [12]. Fig. 6 shows the architecture of this PE based on the LLR version of (16) and (23) with min-sum approximation. Here S2C is the block that performs the conversion from sign-magnitude form to 2's complement form, while C2S unit carries out the inverse conversion. Additionally, adder and subtractor are employed to carry out addition and subtraction between the two inputs. The corresponding sum and difference are selected by the partial sum signal $\hat{u}_{sum}$ from the PSG block. Finally, at the output end of the PE, *control* signal is used to determine the output as $LLR(\hat{a})$ or $LLR(\hat{b})$, which is propagated to the next stage. In summary, the architecture shown in Fig. 6 mainly consists of one comparator-selector, one adder, one subtractor, two multiplexers, two C2S and two S2C blocks. Accordingly, the critical path delay of PE is $T_{S2C} + T_{adder} + T_{C2S} + 2T_{MUX}$.

B. p Node

In Scheme-A, the decision scheme in **p** node has been described based on the LLR representation. To implement its function, a straightforward approach is to employ a sorting circuit and a signed adder. However, this method is too complex and is not hardware-efficient. After careful examination of Scheme-A, we observe that the **p** node can be implemented with a very simple method, which is described as below.

First, since the function of **p** node depends on the frozen conditions of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$, signals *frozen*1 and *frozen*2 are introduced to indicate whether $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are frozen bits or not. If $\hat{u}_{2i-1}$ is frozen, *frozen*1 will be 1, otherwise 0. Similarly, *frozen*2 will be 1 or 0 when $\hat{u}_{2i}$ is frozen or not. Secondly, the sign bits of $LLR(\hat{c})$ and $LLR(\hat{d})$ are employed for simplifying computations. Denoted as $sign(LLR(\hat{c}))$ and $sign(LLR(\hat{d}))$, these sign bits will be, respectively, 0 or 1 when the corresponding LLR values are non-negative or negative. Furthermore, the *comp* signal, which is the result of comparison between absolute value of $LLR(\hat{c})$ and $LLR(\hat{d})$, is also employed. When $|LLR(\hat{c})| \geq |LLR(\hat{d})|$, *comp* will be 1, otherwise 0. Accordingly, with the above five signals, we can obtain the truth table shown in Table III for $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ based on Scheme-A.

Then, with the help of above truth table, Boolean expression of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ can be derived as follows:

$$\hat{u}_{2i-1} = \overline{frozen1}(sign(LLR(\hat{c}) \oplus sign(LLR(\hat{d})) \tag{34}$$

$$\begin{aligned}\hat{u}_{2i} = &\overline{comp}\ \overline{frozen2}\ sign(LLR(\hat{d})) \\ &+ comp\ \overline{frozen1}\ \overline{frozen2}\ sign(LLR(\hat{d})) \\ &+ comp\ frozen1\ \overline{frozen2}\ sign(LLR(\hat{c})).\end{aligned} \tag{35}$$

Based on the (34), (35), a hardware architecture of the **p** node with q-bit quantization is shown in Fig. 7. Here $LLR(\hat{c})$ and $LLR(\hat{d})$ are represented in sign-magnitude (SM) form, and they are output from the **f** and **g** nodes in stage-$(m-1)$. In addition, since the frozen conditions of $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ have been predetermined before the transmission, signals *frozen*1 and *frozen*2 can be easily obtained from the control unit.

It can be seen that the circuit of **p** node in Fig. 7 is much simpler than that of the PE in Fig. 6. This leads to two benefits. First, since all the **f** and **g** nodes in stage- m are replaced by **p** nodes, the hardware complexity of stage- m in 2b-SC decoder (Fig. 5) is less than the original SC decoder (Fig. 2). Second, because the critical path delay of **p** node is only $T_{comp} + 2T_{AND} + 2T_{OR}$, which is much shorter than that of the PE, the latency can be

TABLE III
THE TRUTH TABLE OF $\mathbf{p}$ NODE

Input					Output	
*frozen*1	*frozen*2	*sign*(*LLR*($\hat{c}$))	*sign*(*LLR*($\hat{d}$))	*comp*	$\hat{u}_{2i-1}$	$\hat{u}_{2i}$
0	0	0	0	don't care	0	0
		1	1	don't care	0	1
		1	0	don't care	1	0
		0	1	don't care	1	1
1	1	don't care	don't care	don't care	0	0
1	0	1	1	don't care	0	1
		1	0	1		1
		1	0	0		0
		0	1	0		1
		0	1	1		0
		0	0	don't care		0
0	1	0	0	don't care	0	0
		1	1	don't care	0	
		0	1	don't care	1	
		1	0	don't care	1	

further reduced from $(1.5n - 2)$ to $(n - 1)$ as discussed in Section IV.D.

C. Overall Architecture for 2b-SC Decoder

Based on the circuits of the PE and the $\mathbf{p}$ node in Figs. 6 and 7, respectively, the overall 2b-SC decoder can be constructed as a butterfly-like architecture (Fig. 5). However, this straightforward design is not hardware-efficient. For the architecture in Fig. 5, at least half of nodes in each stage are always idle during decoding procedure. Therefore, in order to increase hardware utilization, two types of architectures, referred as tree-based and line-based architectures [11], [13], are usually used to construct overall SC decoder. In this paper we develop our 2b-SC decoder with these two approaches as well.

Fig. 8 shows the architecture of a tree-based 2b-SC decoder with $n = 8$. In this design, when a particular stage is activated, all the nodes in that stage are activated. Therefore, a total of $(n - 2)$ PEs and a single $\mathbf{p}$ node are needed.

One of the disadvantages of the tree-based architecture is that only the activated stage can achieve 100% hardware utilization in each cycle. Considering the waste of idle resource, line-based 2b-SC architecture, which merges $(m - 1)$ stages into a single stage, is illustrated in Fig. 9. In this figure, the numbers associated with the switches indicate the time index when the switches will be turned on. Compared with the tree-based architecture, the line-based architecture is attractive for moderate-speed applications due to its low hardware cost and better hardware utilization efficiency.

Besides the aforementioned tree-based and line-based architectures, overlapped architecture [11] and semi-parallel architecture [12] are two other types of architectures. In [11], the overlapped architecture was proposed to process multiple codeword to overcome the hardware underutilization of the tree-based architecture. The disadvantage of the overlapped architecture is the need for extra register/memory resource. In [12], the semi-parallel architecture was proposed to achieve low complexity by using fewer PEs. As a result, the hardware utilization is improved at the expense of increasing decoding latency.

As a general latency-reducing approach, the proposed 2b-SC decoding scheme can also be applied to the overlapped architecture in [11] and semi-parallel architecture in [12]. Similar to tree-based and line-based 2b-SC architectures, the 2b-SC version of overlapped and semi-parallel architectures can be easily developed by replacing the original last stage with our proposed $\mathbf{p}$ node. Therefore, in this paper the 2b-SC designs based on overlapped and semi-parallel architectures are not discussed in detail.

D. 2b-SC-Overlapped-Scheduling Architecture

In Section IV.B, it is observed that the $\mathbf{p}$ node has shorter critical path than the PE and this can be exploited to reduce the overall latency to $(n-1)$. This subsection explains the reason for this reduction and then develops the corresponding architecture, referred as *2b-SC-Overlapped-scheduling architecture*.

As illustrated in Fig. 5, after $\mathbf{p}$ node computes current $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$, in the next cycle, the $\mathbf{g}$ node, instead of $\mathbf{f}$ node, will be activated each time. The decoding sequence between these two nodes is illustrated in Fig. 10(a), and its example timing chart for hardware architecture is shown in Fig. 10(b). First it takes $T_{comp} + 2T_{AND} + 2T_{OR}$ for $\mathbf{p}$ node to compute $\hat{u}_3$ and $\hat{u}_4$ (see Fig. 7), and then the PSG block will use these two bits to calculate $\hat{u}_{sum}$. Finally $\hat{u}_{sum}$ is input to the PE for the computation of the $\mathbf{g}$ node (see Fig. 6). Note that here the critical path delay of the PSG block is always $2T_{XOR}$. This is because the computation of $\hat{u}_{sum}$ can be executed in a recursive manner. For example, in order to compute $\hat{u}_{sum} = \hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3 \oplus \hat{u}_4$, because $\hat{u}_1$ and $\hat{u}_2$ have been obtained and $\hat{u}_1 \oplus \hat{u}_2$ has been computed and stored in the PSG block in the previous cycle, only two exclusive-or operations are needed to obtain $\hat{u}_{sum}$ from $\hat{u}_3$ and $\hat{u}_4$.

After a careful examination of the decoding sequence in Fig. 10(a), it is found that the computations of $\mathbf{p}$ and $\mathbf{g}$ nodes can be overlapped. The new decoding sequence with *overlapped scheduling* [15] is shown in Fig. 10(c). Here the computations of the $\mathbf{p}$ and $\mathbf{g}$ nodes are carried out in the same clock cycle; therefore, one cycle can be saved each time. The validity of the proposed overlapped scheduling

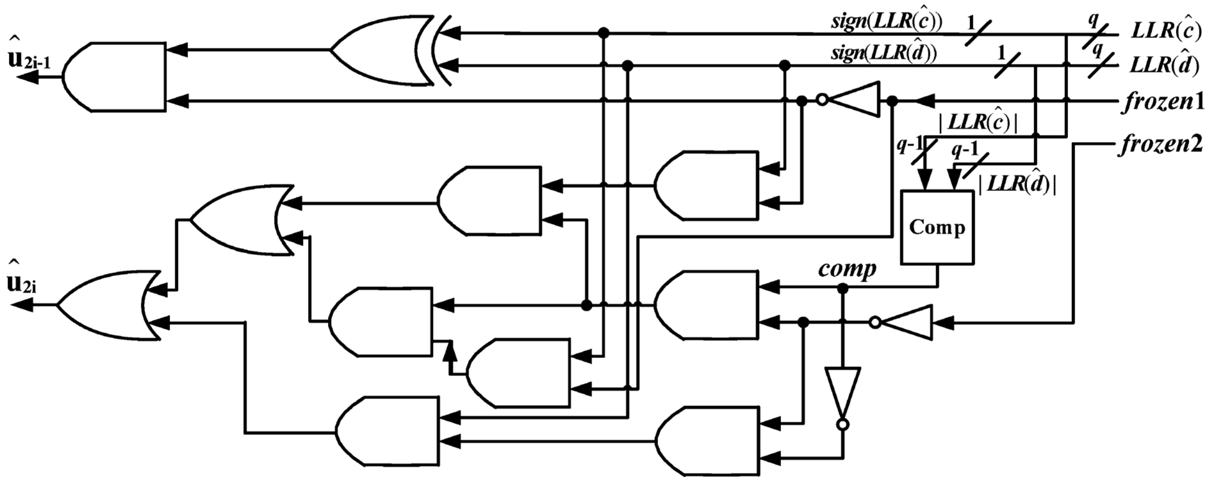


Fig. 7. The architecture of $\mathbf{p}$ node.

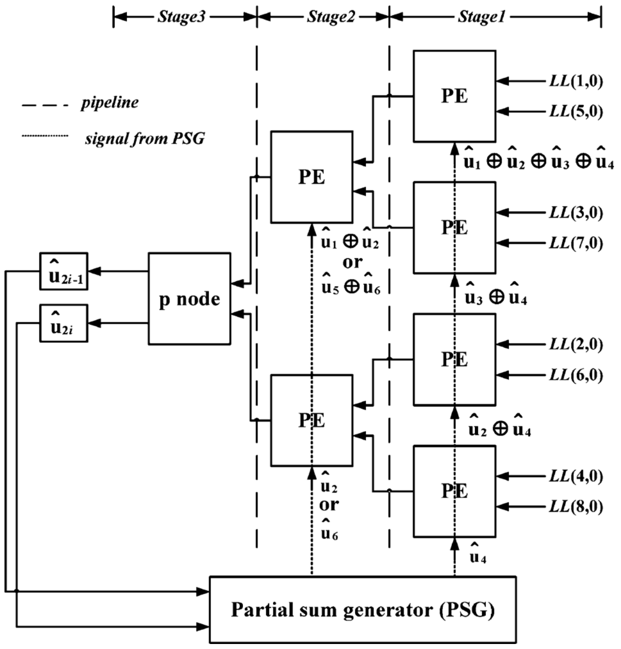


Fig. 8. The tree-based 2b-SC architecture with $n = 8$.

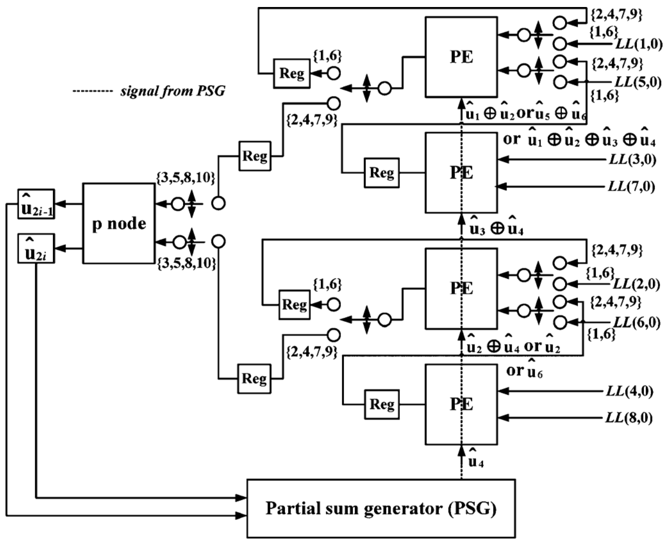


Fig. 9. The line-based 2b-SC architecture with $n = 8$.

is shown in Fig. 10(d). The arrival time of $\hat{u}_{sum}$ for PE is $T_{pnode} + 2T_{XOR}$, which is much less than its maximum allowable arrival time $T_{S2C} + T_{adder} + T_{C2S}$ (according to Fig. 6). For example, with 5-bit quantization and FreePDK 45 nm standard CMOS technology, synthesis results show that $T_{S2C} + T_{adder} + T_{C2S} = 0.9539$ ns while $T_{pnode} + 2T_{XOR}$ is only 0.5417 ns. Therefore, the overlapped computation of $\mathbf{p}$ node and $\mathbf{g}$ node in the PE can be accurately carried out without timing conflict. Considering $\mathbf{p}$ node is activated for 0.5 n cycles, this overlapped scheduling approach reduces the overall latency to $(1.5n - 2)$-$(0.5n - 1) = (n - 1)$. Table IV shows a scheme of the 2b-SC-Overlapped-scheduling decoder for $n = 8$ polar code. Based on this scheme, the corresponding tree-based and line-based architectures can also be easily derived from Fig. 8 and Fig. 9 by removing the registers between the $\mathbf{p}$ node and the PSG block.

E. 2b-SC-Precomputation Architecture

In [13], *precomputation* technique [16]–[18] was exploited to reduce the overall latency of the original SC algorithm. The essential idea of this method is to merge the computation of $\mathbf{f}$ and $\mathbf{g}$ nodes in the same stage. Table V shows a schedule of the SC-Precomputation decoding scheme. In each clock cycle, the computations of $\mathbf{f}$ and $\mathbf{g}$ nodes are carried out at the same time. As a result, the overall latency is 50% less than that of the conventional scheme in Table I. Moreover, in order to implement the precomputation scheme, [13] proposed to employ merged PEs (see Fig. 11). Different from conventional 2-input 1-output PE (Fig. 6), this modified 2-input 3-output PE can calculate the exact output of $\mathbf{f}$ node and 2 output candidates of $\mathbf{g}$ node at the same time. The valid output of the $\mathbf{g}$ node is selected and propagated to the next stage when corresponding $\hat{u}_{sum}$ is available. For details of the SC-Precomputation algorithm and architecture, the reader is referred to [13].

Although SC-Precomputation decoder in [13] has saved half of the clock cycles, with the help of the reformulation ($\mathbf{p}$ node) of the last stage in Section III.B, further reduction on latency can

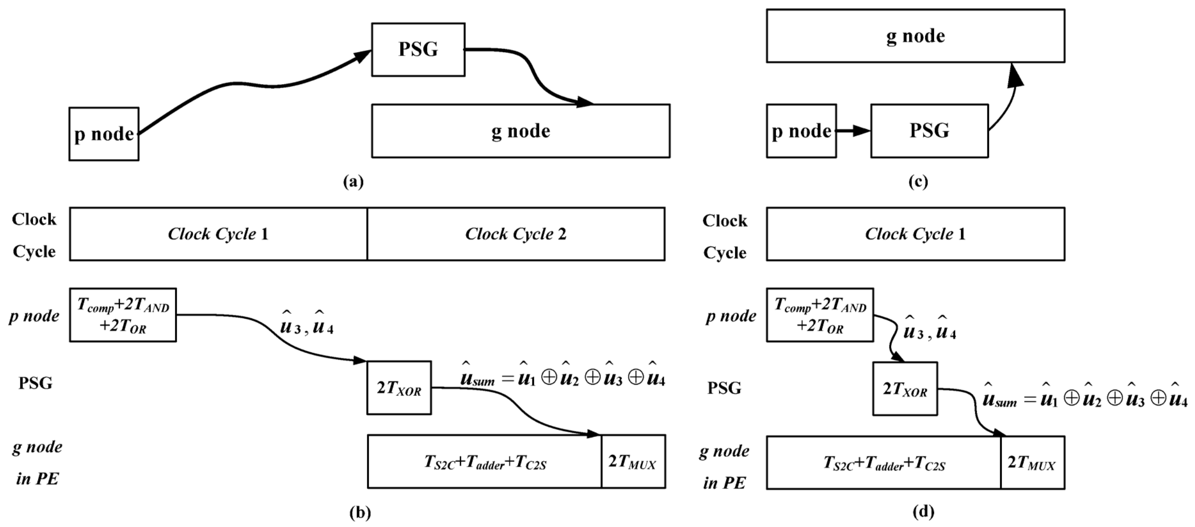


Fig. 10. (a) Original 2b-SC decoding sequence between $\mathbf{p}$ and $\mathbf{g}$ nodes. (b) Example timing chart for original decoding scheme. (c) Decoding sequence between $\mathbf{p}$ node and $\mathbf{g}$ node in PE after overlapped scheduling. (d) Example timing chart after overlapped scheduling.

TABLE IV
OVERLAPPED SCHEDULING OF 2B-SC FOR $n = 8$ POLAR CODE

Overlapped scheduling of 2b-SC decoding scheme							
Clock cycle	1	2	3	4	5	6	7
Stage1	f			g			
Stage2		f	g		f	g	
Stage3			p	p		p	p
Output			$\hat{u}_1$ & $\hat{u}_2$	$\hat{u}_3$ & $\hat{u}_4$		$\hat{u}_5$ & $\hat{u}_6$	$\hat{u}_7$ & $\hat{u}_8$

TABLE V
DECODING SCHEMES OF SC-PRECOMPUTATION [13] FOR $n = 8$ POLAR CODE

SC-Precomputation decoding scheme[13]							
Clock cycle	1	2	3	4	5	6	7
Stage1	Merged f&g						
Stage2		Merged f&g			Merged f&g		
Stage3			Merged f&g	Merged f&g		Merged f&g	Merged f&g
Output			$\hat{u}_1$ & $\hat{u}_2$	$\hat{u}_3$ & $\hat{u}_4$		$\hat{u}_5$ & $\hat{u}_6$	$\hat{u}_7$ & $\hat{u}_8$

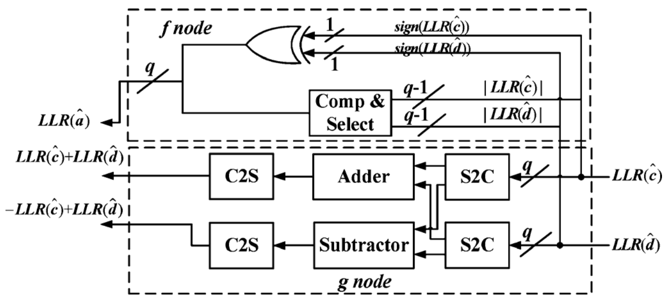


Fig. 11. The architecture of merged PE for SC-Precomputation decoding in [13].

be obtained. Recall that the function of the $\mathbf{p}$ node is to output 2 bits in one cycle; therefore, the merged computations for $\mathbf{f}$ and $\mathbf{g}$ nodes in the last stage of SC-Precomputation scheme (Table V) can be completely replaced by the $\mathbf{p}$ node. In addition, since the critical path of the $\mathbf{p}$ node is short, the computation of $\mathbf{p}$ node in adjacent cycles can be merged into one cycle. Table VI shows the example decoding scheme of this 2b-SC-Precomputation decoder. Based on this new scheme, the overall latency is further reduced from $(n-1)$ to $(3n/4-1)$.

When merging two successive computations of $\mathbf{p}$ nodes into one cycle, a potential problem is the increase of critical path delay. Because the longest data path between two successive computations of $\mathbf{p}$ nodes is longer than that in the merged PE in Fig. 11, a straightforward implementation of the merge operation will increase the critical path delay. To solve this problem, *look-ahead* technique [17], [18] is applied to the last stage. An example of this reformulation is illustrated in Fig. 12. By using look-ahead technique, the critical path of the last stage is reduced from $2T_{pnode}+T_{PSG}+T_{MUX}$ in Fig. 12(a) to $T_{pnode}+T_{PSG}+T_{4-1MUX}$ in Fig. 12(b), which is smaller than the longest path delay in the PE. The validity of this assumption has been verified by synthesis results. With 5-bit quantization and 45 nm technology, $T_{pnode}+T_{PSG}+T_{4-1MUX}=0.6738$ ns

TABLE VI
DECODING SCHEMES OF 2B-SC-PRECOMPUTATION BEFORE LOOK-AHEAD REFORMULATION FOR $n = 8$ POLAR CODE

2b-SC-Precomputation decoding scheme before look-ahead reformulation							
Clock cycle	**1**	**2**	**3**		**4**	**5**	
Stage1	Merged f&g						
Stage2		Merged f&g			Merged f&g		
Stage3			p	p		p	p
Output			$\hat{u}_1$ & $\hat{u}_2$	$\hat{u}_3$ & $\hat{u}_4$		$\hat{u}_5$ & $\hat{u}_6$	$\hat{u}_7$ & $\hat{u}_8$

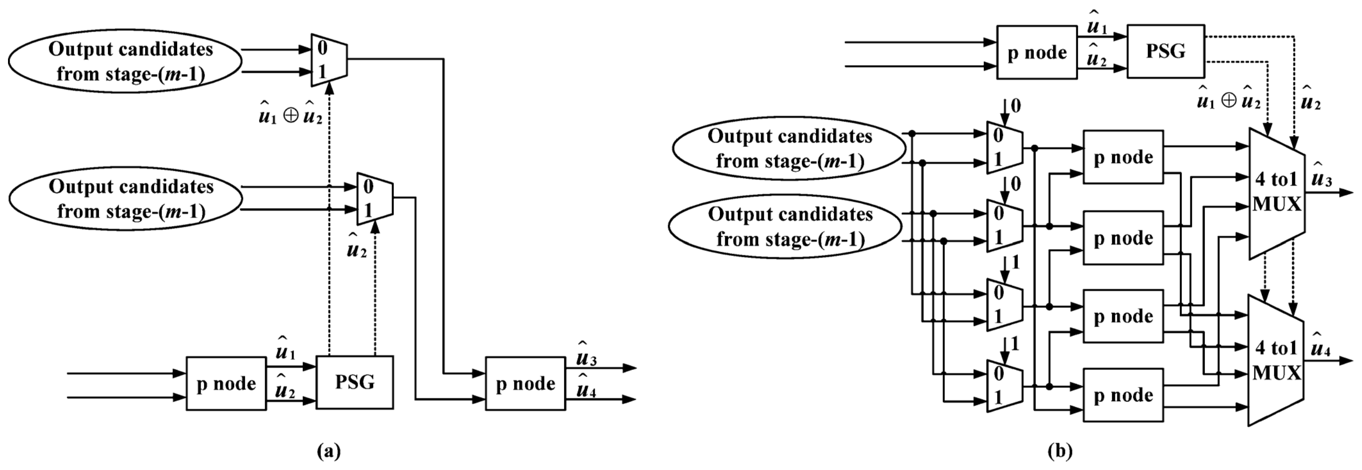

Output candidates from stage-(m-1)
Output candidates from stage-(m-1)
0
1
$\hat{u}_1 \oplus \hat{u}_2$
$\hat{u}_2$
p node
$\hat{u}_1$
$\hat{u}_2$
PSG
p node
$\hat{u}_3$
$\hat{u}_4$
(a)
p node
$\hat{u}_1$
$\hat{u}_2$
PSG
$\hat{u}_1 \oplus \hat{u}_2$
$\hat{u}_2$
Output candidates from stage-(m-1)
Output candidates from stage-(m-1)
0
1
p node
4 to1 MUX
$\hat{u}_3$
$\hat{u}_4$
(b)

Fig. 12. (a) Original design for two successive computations of $\mathbf{p}$ nodes in the last stage (stage- m). (b) Look-ahead reformulation.

TABLE VII
DECODING SCHEMES OF 2B-SC-PRECOMPUTATION AFTER LOOK-AHEAD REFORMULATION FOR $n = 8$ POLAR CODE

2b-SC-Precomputation decoding scheme after look-ahead reformulation							
Clock cycle	**1**	**2**	**3**		**4**	**5**	
Stage1	Merged f&g						
Stage2		Merged f&g			Merged f&g		
Stage3			p			p	
			p			p	
Output			$\hat{u}_1$ & $\hat{u}_2$, $\hat{u}_3$ & $\hat{u}_4$			$\hat{u}_5$ & $\hat{u}_6$, $\hat{u}_7$ & $\hat{u}_8$	

TABLE VIII
HARDWARE COMPARISON OF DIFFERENT TREE-BASED AND LINE-BASED (n, k) SC DECODERS

Tree-based and Line-based Architecture						
Hardware		**SC-Precomputation [13]**	**SC [11]**	**2b-SC**	**2b-SC with Overlapped-scheduling**	**2b-SC with Precomputation**
# of PE	**Tree-based**	n-1	n-1	n-2	n-2	n-2
	Line-based	n/2	n/2	n/2	n/2	n/2
# of p node		0	0	1	1	5
# of 1-bit REG		~3qn	~qn	~qn	~qn	~3qn
Latency (cycle)		n-1	2n-2	1.5n-2	n-1	0.75n-1
Throughput (Normalized)		2	1	1.33	2	2.67

while T_{PE} is about 0.9539 ns. Therefore, the critical path delay of overall 2b-SC-Precomputation decoder will be the same as that of the SC-Precomputation decoder. Table VII shows the example decoding scheme of 2b-SC-Precomputation after look-ahead reformulation.

V. HARDWARE ANALYSIS AND COMPARISON

In this section, we analyze hardware performance of the proposed 2b-SC architectures and compare them with the state-of-the-art designs. Tables VIII shows the required hardware resource, latency and throughput of different (n, k) polar tree-based and line-based SC architectures, respectively. In this table all the list designs are assumed to be constructed based on the same PE with q-bit quantization scheme. Notice that non-uniform quantization scheme similar to those in [20], [22] can be used to achieve smaller word length.

From Table VIII it can be seen that that the normalized throughput of the 2b-SC, 2b-SC-Overlapped-scheduling, 2b-SC-Precomputation decoders are 1.33, 2, and 2.67, respectively, where these are normalized to the SC decoder in [11]. Compared with SC design in [11], the 2b-SC and 2b-SC-Overlapped-scheduling decoders have much shorter decoding latency. Since the critical path remains the same, this reduction in latency can lead to increased throughput. Meanwhile, unlike SC-Precomputation decoders [13], the 2b-SC and 2b-SC-Overlapped-scheduling decoders succeed in

TABLE IX
IMPLEMENTATION RESULTS OF DIFFERENT (1024, 512) SC DECODERS WITH 5-BIT QUANTIZATION

Design	[19]*	[12]	Tree-based 2b-SC-Precomputation
CMOS Technology	180nm	65nm	45nm
Total gate counts	183637	214370**	338499
Frequency (MHz)	150	500	750
Decoding latency (cycle)	1560†	2080†	767
Throughput (Mbps)	49	123	500
TSNT (Mbps/Kgate) (scaled to 45nm)‡	1.07	0.83	1.48

* Results in [19] are measurement results.

** Gate count is calculated based on the area information in [12] and unit gate area in TSMC 65nm CMOS library.

† Decoding latency is calculated based on the equation (12) in [12].

‡ Technology scaled normalized throughput, referred as TSNT in [21], is defined by *Throughput*(technology/45nm)/Total gate count*

reducing latency without requiring any extra registers. Therefore, these two decoders maintain low complexity. Besides, by applying precomputation technique to the 2b-SC design, the latency of the 2b-SC-Precomputation architecture is reduced to $(3n/4 - 1)$. To the best of our knowledge, this is the shortest decoding latency among all known SC decoders. Since **p** node occupies very small area of the whole decoder $(<0.01\%)$, the proposed 2b-SC-Precomputation decoder has about 30% higher normalized throughput than the SC-Precomputation decoder in [13] with the same complexity.

Additionally, in order to demonstrate the advantage of the proposed architectures, we have implemented our designs for polar (1024, 512) code with Verilog HDL. Here tree-based 2b-SC-Precomputation architecture is selected for implementation. After developing the RTL models, we synthesize our decoders with FreePDK 45 nm standard CMOS library by using Synopsys Design Complier.

Table IX lists the implementation results of reported polar (1024, 512) SC decoders. Notice that [19] used a speculative method to achieve 2 bits output in one cycle. Compared with the hardware-based method in [19], our proposed 2b-SC approach is more general since it reformulates the algorithm. As a result, this reformulation reduces the critical path of the last stage, and then enables the reduced-latency 2b-SC-Overlapped-scheduling and 2b-SC-Precomputation architectures.

From Table IX it can be seen that our design can achieve at least twice reduction in latency as well as 4 times increase in throughput. When scaling to the same technology (45 nm), the technology scaled normalized throughput (TSNT) metric, defined as throughput per Kgate, increases by at least 40% for our design. Notice that the designs in [12], [19] are based on semi-parallel architecture while our design is based on tree architecture. If the proposed 2b-SC-Precomputation design is also implemented on the same low-complexity semi-parallel architecture, the advantage of our design on hardware performance will be further improved. We estimate that the semi-parallel-based 2b-SC, 2b-SC with overlapped scheduling and 2b-SC-Precomputation decoders require latencies of around $1.5n$, n and $0.75n$ with area overhead of 0, 0, and 40%, respectively. Therefore, these architectures offer the throughput/area advantages by factors 1.33, 2 and 1.92, respectively, as compared to semi-parallel architecture in [12].

Due to the generality of 2b-SC decoding scheme, it can be widely applied to current and future SC decoders, independent of the design of the **f** and **g** nodes. In summary, the proposed 2b-SC decoding algorithm and architectures are very attractive for hardware implementations of low-latency SC decoders.

VI. CONCLUSION

In this paper, a novel reformulation for the last stage of the SC decoding is proposed. Based on this reformulation, a reduced-latency 2b-SC decoding algorithm is presented. In addition, with the use of overlapped scheduling and precomputation approaches, the decoding latency of 2b-SC design is further reduced. Analysis shows that the proposed 2b-SC architectures have significant advantages with respect to both throughput and hardware efficiency. Future work will be directed towards design of polar list decoders using our proposed 2-bit decoding approach.

APPENDIX

To prove $\hat{u}_{2i-1} = \alpha$ and $\hat{u}_{2i} = \beta$, we show that $\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$ corresponds to the largest probability among P(00), P(01), P(10) and P(11). Since $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ can be either 0 or 1, we discuss four possible cases:

Case A-1: $\hat{u}_{2i-1} = 0$ and $\hat{u}_{2i} = 0$: Recall that $\hat{u}_{2i-1}$ and $\hat{u}_{2i}$ are the outputs from the SC algorithm. Therefore, according to (3), when $\hat{u}_{2i-1} = 0, L(2i-1, m) = LR(\hat{u}_{2i-1}) \geq 1$.

According to (3), (16),

$$\begin{aligned} L(2i-1,m) &= LR(\hat{u}_{2i-1}) \\ &= \frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)}{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)} \\ &\geq 1. \end{aligned}$$

Thus,

$$\begin{aligned} &\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1) \\ &\qquad \geq \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1). \quad \text{(A1)} \end{aligned}$$

Now we show that the largest probability $\mathrm{P}(\alpha\beta)$ must be $P(00) = \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)$ or $P(01) = P(c=1)P(d=1)$.

Proposition-A1: Given (A1), among $\mathrm{P}(00), \mathrm{P}(01), \mathrm{P}(10)$ and $\mathrm{P}(11)$, the largest probability $\mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(00)$ or $\mathrm{P}(01)$.

Proof: If $\mathrm{P}(\alpha\beta)$ is not $\mathrm{P}(00)$ or $\mathrm{P}(01)$, without loss of generality, assume $\mathrm{P}(\alpha\beta)$ is $\mathrm{P}(10) = \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)$. Since $\mathrm{P}(\alpha\beta) = \mathrm{P}(10)$ is the largest probability, and the sum of

$\mathrm{P}(\hat{c}=1)$ and $\mathrm{P}(\hat{c}=0)$ is equal to some non-negative value x, then we have:

$$\begin{aligned}&\mathrm{P}(10)>\mathrm{P}(00)\\&\quad\Rightarrow \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)>\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)\\&\quad\Rightarrow x-\mathrm{P}(\hat{c}=0)>\mathrm{P}(\hat{c}=0)\Rightarrow \mathrm{P}(\hat{c}=0)<x/2.\end{aligned}\tag{A2}$$

Similarly, we can get:

$$\begin{aligned}&\mathrm{P}(10)>\mathrm{P}(01)\\&\quad\Rightarrow \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)>\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow \mathrm{P}(\hat{d}=0)>y-\mathrm{P}(\hat{d}=0)\Rightarrow \mathrm{P}(\hat{d}=0)>y/2\end{aligned}\tag{A3}$$

where y is the non-negative sum of $\mathrm{P}(\hat{d}=1)$ and $\mathrm{P}(\hat{d}=0)$.

Recall that for (A1):

$$\begin{aligned}&\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\&\quad\geq \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow \mathrm{P}(\hat{c}=0)(\mathrm{P}(\hat{d}=0)-\mathrm{P}(\hat{d}=1))\\&\quad\geq \mathrm{P}(\hat{c}=1)(\mathrm{P}(\hat{d}=0)-\mathrm{P}(\hat{d}=1))\\&\quad\Rightarrow (\mathrm{P}(\hat{c}=0)-\mathrm{P}(\hat{c}=1))(\mathrm{P}(\hat{d}=0)-\mathrm{P}(\hat{d}=1))\geq 0\\&\quad\Rightarrow (2\mathrm{P}(\hat{c}=0)-x)(2\mathrm{P}(\hat{d}=0)-y)\geq 0.\end{aligned}\tag{A4}$$

However, with (A2) and (A3) we know that $(2\mathrm{P}(\hat{c}=0)-x)(2\mathrm{P}(\hat{d}=0)-y)<0$, which contradicts (A4). Therefore, $\mathrm{P}(\alpha\beta)$ can not be $\mathrm{P}(10)$. Similarly, it can be proved that $\mathrm{P}(\alpha\beta)$ can not be $\mathrm{P}(11)$. Therefore, $\mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(00)$ or $\mathrm{P}(01)$. □

After proving the above proposition-A1, we now show $\mathrm{P}(00)$ must be larger than $\mathrm{P}(01)$. Since $\hat{u}_{2i-1}=0$ and $\hat{u}_{2i}=0$, according to (3), (23), we can get

$$\begin{aligned}&LR(\hat{u}_{2i})=LR(\hat{c})^{1-2\hat{u}_{2i-1}}LR(\hat{d})\\&\quad=LR(\hat{c})LR(\hat{d})\geq 1\\&\quad\Rightarrow \frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)}\geq 1\\&\quad\Rightarrow \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)\geq \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow \mathrm{P}(00)\geq \mathrm{P}(01).\end{aligned}\tag{A5}$$

Since it has been proved that $\mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(00)$ or $\mathrm{P}(01)$, then with (A5), we have $\mathrm{P}(\alpha\beta)=\mathrm{P}(00)=\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$.

Case A-2: $\hat{u}_{2i-1}=0$ and $\hat{u}_{2i}=1$: Similar to the case A-1, when $\hat{u}_{2i-1}=0, \mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(00)$ or $\mathrm{P}(01)$.

For $\hat{u}_{2i-1}=0$ and $\hat{u}_{2i}=1$, according to (3), (23), we can obtain

$$\begin{aligned}&LR(\hat{u}_{2i})=LR(\hat{c})^{1-2\hat{u}_{2i-1}}LR(\hat{d})\\&\quad=LR(\hat{c})LR(\hat{d})<1\\&\quad\Rightarrow \frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)}<1\\&\quad\Rightarrow \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)<\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow P(00)<P(01).\end{aligned}\tag{A6}$$

Since $\mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(00)$ or $\mathrm{P}(01)$, in this case, $\mathrm{P}(\alpha\beta)=\mathrm{P}(01)=\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$.

Case A-3: $\hat{u}_{2i-1}=1$ and$\hat{u}_{2i}=0$: When $\hat{u}_{2i-1}=1$, according to (3), (16), we have

$$\begin{aligned}&L(2i-1,m)=LR(\hat{u}_{2i-1})<1\\&\quad\Rightarrow LR(\hat{u}_{2i-1})\\&\quad=\frac{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)}{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)}<1\\&\quad\Rightarrow \mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=1)\\&\quad<\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)+\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1).\end{aligned}\tag{A7}$$

Similar to the proof of proposition-A1, it is easy to prove: From (A7) the $\mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(10)=\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)$ or $\mathrm{P}(11)=\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)$.

Then, consider $\hat{u}_{2i}=0$, with (23), we can obtain that

$$\begin{aligned}&LR(\hat{u}_{2i})=LR(c)^{1-2\hat{u}_{2i-1}}LR(d)\\&\quad=LR(c)^{-1}LR(d)\geq 1\\&\quad\Rightarrow \frac{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)}\geq 1\\&\quad\Rightarrow \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)>\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow \mathrm{P}(10)>\mathrm{P}(11).\end{aligned}$$

Therefore, $\mathrm{P}(\alpha\beta)=\mathrm{P}(10)=\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$.

Case A-4: $\hat{u}_{2i-1}=1$ and $\hat{u}_{2i}=1$: Similar to the case A-3, when $\hat{u}_{2i-1}=1, \mathrm{P}(\alpha\beta)$ must be $\mathrm{P}(10)$ or $\mathrm{P}(11)$.

For $\hat{u}_{2i-1}=1$ and $\hat{u}_{2i}=1$, according to (3), (23), we can obtain

$$\begin{aligned}&LR(\hat{\mathrm{u}}_{2\mathrm{i}})=LR(c)^{1-2\hat{\mathrm{u}}_{2\mathrm{i}-1}}LR(d)\\&\quad=LR(c)^{-1}LR(d)<1\\&\quad\Rightarrow \frac{\mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)}{\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)}<1\\&\quad\Rightarrow \mathrm{P}(\hat{c}=1)\mathrm{P}(\hat{d}=0)<\mathrm{P}(\hat{c}=0)\mathrm{P}(\hat{d}=1)\\&\quad\Rightarrow \mathrm{P}(10)<\mathrm{P}(11).\end{aligned}$$

Hence the largest probability $\mathrm{P}(\alpha\beta)=\mathrm{P}(11)=\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$.

Summarizing the above four cases, we can conclude that $\mathrm{P}(\alpha\beta)=\mathrm{P}(\hat{u}_{2i-1}\hat{u}_{2i})$ holds all the time. Therefore, $\hat{u}_{2i-1}=\alpha$ and $\hat{u}_{2i}=\beta$. Thus, proposition 1 is proved. □

REFERENCES

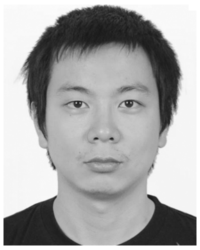
[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, 2010.

[3] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, Jul. 2009.

[4] I. Tal and A. Vardy, "How to construct polar codes," May 2011, arXiv: 1105.6164v1.

[5] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, 2011.
[6] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2011, pp. 1–5.
[7] K. Niu and K. Chen, "Stack decoding of polar codes," *Elect. Lett.*, vol. 48, no. 12, pp. 695–696, 2012.
[8] E. Arıkan, "Systematic polar coding," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 860–862, 2011.
[9] E. Arıkan, "Polar codes: A pipelined implementation," in *Proc. 4th Int. Symp. on Broad. Commun. ISBC 2010*, Jul. 2010, pp. 11–14.
[10] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *Proc. 8th Int. Symp. on Wireless Commun. Syst. (ICWCS)*, Nov. 2011, pp. 437–441.
[11] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *Proc. IEEE ICASSP*, May 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," *IEEE Trans. Signal Processing*, vol. 61, no. 2, pp. 289–299, Jan. 2013.
[13] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *Proc. Int. Conf. Commun.*, June 2012, pp. 3471–3475.
[14] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Processing*, vol. 61, no. 10, pp. 2429–2441, May 2013.
[15] K. K. Parhi and D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data flow programs via optimum unfolding," *IEEE Trans. Comput.*, vol. 40, no. 2, pp. 178–195, Feb. 1991.
[16] K. K. Parhi, "Pipelining in algorithms with quantizer loops," *IEEE Trans. on Circuits and Systems*, vol. 38, no. 7, pp. 745–754, Jul. 1991.
[17] K. K. Parhi, "Design of multi-gigabit multiplexer loop based decision feedback equalizers," *IEEE Trans. VLSI Syst.*, vol. 13, no. 4, pp. 489–493, Apr. 2005.
[18] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY, USA: Wiley, 1999.
[19] A. Mishra, A. Raymond, L. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, and W. J. Gross, "A successive cancellation decoder ASIC for a 1024-bit polar code in 180 nm CMOS," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, 2012, pp. 205–208.
[20] Z. Cui and Z. Wang, "A 170 Mbps (8176, 7156) quasi-cyclic LDPC decoderimplementation with FPGA," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 5095–5098.
[21] H.-Y. Hsu, A.-Y. Wu, and J.-C. Yeo, "Area-efficient VLSI design of reed-solomon decoder for 10GBased-LX4 optical communication systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 11, pp. 1245–1249, Nov. 2006.
[22] D. Oh and K. K. Parhi, "Minsum decoder architecture with reduced word-length for LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 105–115, Jan. 2010.



Bo Yuan received the B.S. degree in physics and M.S. degree in microelectronics from Nanjing University, Nanjing, China in 2007 and 2010, respectively. He is now working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, MN, USA.

His research interests include VLSI architecture and algorithm design for high-speed low-power communication and digital signal processing systems.

Keshab K. Parhi (S'85–M'88–SM'91–F'96) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, in 1982, the M.S.E.E. degree from the University of Pennsylvania, PA, USA, in 1984, and the Ph.D. degree from the University of California, Berkeley, CA, USA, in 1988.

He has been with the University of Minnesota, Minneapolis, since 1988, where he is currently Distinguished McKnight University Professor and Edgar F. Johnson Professor in the Department of Electrical and Computer Engineering. He has published over 500 papers, has authored the textbook *VLSI Digital Signal Processing Systems* (Wiley, 1999) and coedited the reference book *Digital Signal Processing for Multimedia Systems* (Marcel Dekker, 1999). His research addresses VLSI architecture design and implementation of signal processing, communications and biomedical systems, error control coders and cryptography architectures, high-speed transceivers, and ultra wideband systems. He is also currently working on intelligent classification of biomedical signals and images, for applications such as seizure prediction and detection, schizophrenia classification, and diabetic retinopathy screening.

Dr. Parhi is the recipient of numerous awards including the 2013 Distinguished Alumnus Award from IIT, Kharagpur, India, 2013 Graduate/Professional Teaching Award from the University of Minnesota, 2012 Charles A. Desoer Technical Achievement award from the IEEE Circuits and Systems Society, the 2004 F. E. Terman award from the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, the 2001 IEEE W. R. G. Baker prize paper award, and a Golden Jubilee medal from the IEEE Circuits and Systems Society in 2000. He has served on the editorial boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS and TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, *VLSI Systems, Signal Processing, Signal Processing Letters*, and *Signal Processing Magazine*, and served as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS (2004–2005 term), and currently serves on the Editorial Board of the *Journal of VLSI Signal Processing*. He has served as technical program co-chair of the 1995 IEEE VLSI Signal Processing workshop and the 1996 ASAP conference, and as the general chair of the 2002 IEEE Workshop on Signal Processing Systems. He was a distinguished lecturer for the IEEE Circuits and Systems society during 1996–1998. He served as an elected member of the Board of Governors of the IEEE Circuits and Systems society from 2005 to 2007.