A Configurable Successive-Cancellation List Polar Decoder Using Split-Tree Architecture

Yaoyu Tao, *Member, IEEE*, Sung-Gun Cho, *Graduate Student Member, IEEE*, and Zhengya Zhang, *Senior Member, IEEE*

***Abstract*— Polar codes are capacity-achieving channel codes and they have recently been adopted for fifth-generation (5G) enhanced mobile broadband (eMBB) control channels. Using successive cancellation list (SCL) decoding, the error-correction performance of polar codes can surpass state-of-the-art codes of a comparable length. However, the sequential SC decoding incurs a long latency, and list decoding requires complex tracking of candidates. We present a split-tree SCL decoder that works by dividing a polar code's decoding tree to sub-trees following a split-tree decoding algorithm. The sub-trees are decoded in parallel by smaller sub-decoders that reconcile their decisions in every decoding stage. The split-tree list decoder architecture improves the throughput and latency proportionally to the split factor. By exploiting under-utilized hardware resources, we apply frame interleaving to further increase throughput and employ dynamic clock gating to reduce energy. The results are demonstrated in a 0.64-mm^2 40-nm test chip that implements a split-4, list-2, eight-frame-interleaved decoding architecture. The chip supports configurable code lengths up to 1024 bit and variable code rates. At 0.9 V and room temperature, the chip achieves 3.25 Gb/s with 42.8-mW power, or 13.2 pJ/b, and demonstrates competitive error-correction performance.**

***Index Terms*— Dynamic clock gating (CG), frame interleaving, polar codes, split-tree architecture, successive-cancellation list (SCL) decoder.**

I. Introduction

POLAR codes, discovered by Arikan [1], have recently been adopted by fifth-generation (5G) wireless technology standards. They are the first family of codes with an explicit construction that can provably achieve Shannon's channel capacity. Using successive-cancellation (SC) decoding [1], polar codes achieve the capacity when code length N approaches infinity. However, for moderate-length polar codes of practical interest, SC decoding falls short in terms of error-correction performance compared with other advanced channel codes, such as low-density parity-check (LDPC) codes.

SC list (SCL) decoding was proposed in [2] to improve the error-correction performance of polar codes. SCL decoding follows a similar schedule as SC decoding; bits are decoded sequentially as the decision of one bit depends on the previously decoded bits. However, unlike in SC decoding where only one decision per bit is kept, SCL keeps a list of L ($L > 1$) possible bit combinations in the sequential decoding. With each newly decoded bit, the list of L candidates is updated.

A larger list size L leads to better accuracy, but the complexity grows exponentially with L and the error-correction performance tends to saturate after $L = 4$. A better accuracy can also be obtained by adding a cyclic redundancy check (CRC) to aid the selection of the most likely final decoding decision from the list of L candidates [3], [4]. With $L \geq 2$ and CRC, the error-correction performance of polar codes with SCL decoding can outperform LDPC codes of a comparable code length.

Despite the promise of polar codes with SCL decoding, the sequential nature of SCL decoding and the costly tracking of a list and sorting of candidates lead to low-throughput, high-latency, and high-energy SCL decoding. State-of-the-art polar SCL decoders can hardly meet the requirements for 5G; 5G requires multi-Gb/s in throughput and sub-1 μs in latency, and the decoder chip needs to be kept compact and low power.

There are a handful of pre-silicon SCL decoder designs published in the literature recently. For simplicity and a fair comparison, we will cite the performance using a common configuration of 1024-bit code length, 1/2 code rate, a list size of $L = 2$, and 65-nm CMOS for area. The direct mapping of a log-likelihood ratio (LLR)-based CRC-aided SCL decoder design [5] was estimated to achieve a 334-Mb/s decoding throughput at an 847-MHz clock frequency. A 4-bit grouping approach was proposed in [6] to decode neighboring bits together for higher throughput, and the decoder design was estimated to achieve 395 Mb/s at 500 MHz. Other speedup techniques, such as partial look-ahead decoder [7] and multi-mode decoder designs [8], were shown in simulation to achieve 537 Mb/s and 1.21 Gb/s, respectively.

Area-saving techniques have been developed in [9] and [10]. Hashemi *et al.* [9] introduced an area-saving strategy based on interpolation-based code construction and layered decoding scheme, showing up to 50.7% area saving over a conventional SCL decoder. Mousavi *et al.* [10] presented another area-saving approach using a partial-sum network that efficiently computes the list candidates, resulting in up to 70% area reduction based on synthesis.

Manuscript received March 3, 2020; revised May 11, 2020; accepted June 14, 2020. Date of publication July 20, 2020; date of current version January 28, 2021. This article was approved by Associate Editor Vivek De. This work was supported in part by Intel and in part by NSF under Grant CCF-1054270. *(Corresponding author: Yaoyu Tao.)*

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: taoyaoyu@umich.edu; sunggun@umich.edu; zhengya@umich.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2020.3005763

Performance-enhanced SCL decoders proposed recently include symbol-decision SCL decoder [11] and sphere SCL decoder designs [12] that were shown in simulation to achieve 398 Mb/s at 500 MHz and 1.23 Gb/s at 1GHz, respectively. These designs, however, incur a high area overhead.

Complexity reduction techniques have also been developed, including simplified SCL (SSCL) and fast-SSCL-SPC [13], where the SCL tree is pruned and a single parity check (SPC) is introduced to maintain the error-correction performance. A fast-SSCL-SPC decoder design was estimated to achieve 1.86 Gb/s at 885 MHz in a 1.048-mm^2 area. A follow-up rate-flexible fast-SSCL decoder design [14] improved the results to an estimated 1.22 Gb/s at 955 MHz in a 1.45-mm^2 area. An early stopping criterion was introduced [15] to improve the fast-SSCL decoder further to a projected throughput of 2.05 Gb/s at 650 MHz. However, these complexity reduction techniques tend to hurt the error-correction performance.

When a high-speed decoder is implemented in silicon, we can expect that the measured performance to be worse, the area to be larger, and the power to be higher than the estimates, because it is difficult to fully account for all the overhead of wiring, clocking, and memory by simulation and synthesis. It is, therefore, more reliable to check silicon measurements to make comparisons. For a simple and fair comparison of silicon decoders, we will report the measured results without applying any process or voltage normalization because popular scaling formulas are no longer reflective of the reality of scaling in advanced processes. The latest SC decoder [16] was designed in 180-nm CMOS, occupying 3.17-mm^2 silicon area. The decoder delivered 655 Mb/s at 382 MHz for 1024-bit codewords. The first SCL decoder [17] was designed in a 28-nm FD-SOI technology, occupying 0.44 mm^2. The decoder achieved 614 Mb/s in throughput, 3.34 μs in latency, and 209 pJ/b in energy at 1.3 V for a 1024-bit code length and $L = 4$. The latest SCL decoder [18] was designed in 16-nm FinFET, occupying 2.27 mm^2. The chip demonstrated a 3.24-Gb/s throughput for 1024-bit codes, but no power measurements were reported. Clearly, it is still challenging to meet a multi-Gb/s throughput, sub-μs latency, sub-10-pJ/b energy, and compact silicon area all at the same time.

In this work, we present a 0.64-mm^2 configurable SCL decoder chip using a split-tree architecture in 40-nm CMOS. The decoding tree is split into four subtrees to be decoded by four sub-decoders in parallel with decision reconciliation in every stage. The new split-tree architecture improves the throughput and cuts the latency by nearly 4×. To maximize utilization, eight frames are interleaved and decoded simultaneously to increase throughput by another 8× to 3.25 Gb/s for code length up to 1024 bit and variable code rates. Dynamic clock gating (CG) reduces the peak power dissipation to 42.8 mW at 0.9 V or 13.2 pJ/b. The throughput, energy efficiency, and area efficiency are 5.3×, 15.9×, and 4.0× better, respectively, than the SCL decoder chip in a 28-nm FD-SOI process [17]. Compared with the latest SCL decoder chip [18] in a more advanced 16-nm process, our test chip achieves a similar throughput and 3× better area efficiency. These results make this chip suitable for 5G applications.

II. Background

Polar codes exhibit a polarization effect [1] when transmitted over certain types of channels and decoded using the SC algorithm. The polarization effect refers to that certain bits become highly reliable and the other bits become highly unreliable. To use polar codes, information is conveyed over the reliable bits and the unreliable ones are frozen to predetermined values.

An $(N,\ K)$ polar code has a code length N and code rate K/N, where $N = 2^n$, $n \in \mathbb{Z}^+$. Let $u_0^{N-1} = (u_0, u_1, \ldots, u_{N-1})$ denotes the vector of input bits to the encoder. The $N - K$ least reliable bits in u_0^{N-1}, called frozen bits, are typically set to 0, whereas the remaining K bits, called free bits, are used to carry information a_0^K.

An encoder encodes u_0^{N-1} to the codeword $x_0^{N-1} = (x_0, x_1, \ldots, x_{N-1})$. The systematic encoder is mathematically described by the following equation:

$$x_0^{N-1} = u_0^{N-1} G_N = u_0^{N-1} B_N F^{\otimes n} \quad \text{for } n \geq 1 \tag{1}$$

where $G_N = B_N F^{\otimes n}$ is the $N \times N$ generator matrix, B_N is called the bit-reversal permutation matrix, and $\otimes n$ denotes the nth Kronecker power

$$F^{\otimes n} = F \otimes F^{\otimes (n-1)}, \quad F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \text{ and } F^{\otimes 0} = 1. \tag{2}$$

A polar codeword is produced by first setting the frozen bit locations to 0. For example, a vector of 4 input bits ($N = 4$ and $n = 2$) is set to $u_0^3 = [0\ a_0\ 0\ a_1]$, where bits 0 and 2 are the frozen bits and are set to 0, and bits 1 and 3 are used to carry information bits a_0 and a_1. For simplicity of understanding, assume that B_4 is identity, and then, the encoder performs the following mathematical operation:

$$\begin{aligned} x_0^3 = u_0^3 F^{\otimes 2} &= \begin{bmatrix} 0 & a_0 & 0 & a_1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} a_0 \oplus a_1 & a_0 \oplus a_1 & a_1 & a_1 \end{bmatrix} \end{aligned} \tag{3}$$

where modulo-2 operations are used and $\oplus$ represents modulo-2 addition, or XOR. The encoder produces a (4, 2) code with a block length of 4 and 2 information bits.

The mathematical operation can be illustrated in the encoding graph shown in Fig. 1. Note the regular placement of XORs between every pair of bits in the first stage and every 2-bit blocks in the second stage. Encoding is done by propagating the bit vector through the graph from left to right.

A. SC Decoding

The bits of the codeword x_0^3 are modulated and transmitted through a physical channel, where noise is injected. On the receiver side, each "bit" in the codeword is received as a multibit "soft" value, known as LLRs. A polar decoder operates on the LLRs to produce bit estimates. SC is the first and most widely used decoding algorithm for polar codes.

The SC decoding algorithm was proposed in [1]. It can be visualized by a decoding trellis, as shown in Fig. 1. The LLRs are provided on the left-hand side and the bit decisions are

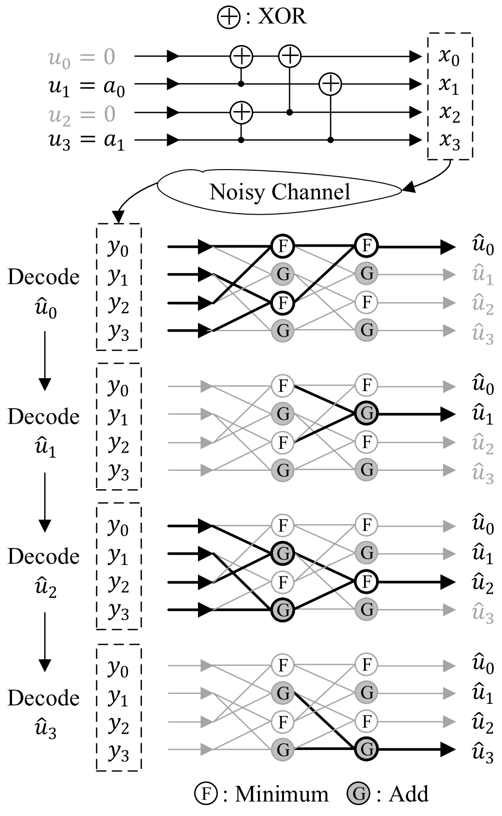

Fig. 1. Encoding graph (top) and SC decoding trellis (bottom) of a (4, 2) polar code.

made on the right-hand side. The trellis consists of $n = \log_2 N$ stages of minimum (F) and summation (G) operations. The F function receives two LLRs L_1 and L_2 and finds the minimum as follows:

$$F(L_1, L_2) = \text{sign}(L_1) \cdot \text{sign}(L_2) \cdot \min(|L_1|, |L_2|). \quad (4)$$

The G function receives the partial modulo-2 sum of all previously decoded bits, $\hat{u}_s$ (not annotated in Fig. 1) in addition to the two LLRs, and computes the conditional sum as follows:

$$G(L_1, L_2, \hat{u}_s) = L_1 \cdot (-1)^{\hat{u}_s} + L_2. \quad (5)$$

The decoding follows a bit-by-bit sequential order. An example of SC decoding is shown in Fig. 1, where in each decoding step, the highlighted paths and nodes are active. One can easily see that only a subset of F and G functions at certain stages are active at a time to decode 1 bit, leaving the other functions in the trellis idle. Assume that it takes 1 unit time per function (F or G) and the trellis is directly mapped in hardware. The latency to decode a bit is variable ranging from 1 to $\log_2 N$. It can be shown that the latency to decode an N-bit codeword is $2N - 2$. For example, decoding a 1024-bit polar code requires 2046 time units. If the trellis is directly mapped to hardware, the hardware complexity is approximately $O(N \log_2 N)$.

The SC decoding can also be represented by the depth traversal of an N-level binary tree, as shown in Fig. 2. For an $N = 4$ polar code, the binary tree consists of four levels, where the branching at each level represents the decoding of a bit to either 0 or 1. The sequential decoding starts from the top and descends. At each level i, a branch is taken, corresponding to calculating the likelihood $L(\hat{u}_i)$, based on which $\hat{u}_i$ is decoded.

For the depth traversal, we can compute the probability of the path, or the path metric (PM) $\text{PM}(\hat{u}_{i-1})$, for the traversal to reach a node representing the decision of $\hat{u}_{i-1}$. For the next step, the path can branch to one of the child nodes, $\hat{u}_i = 0$ or $\hat{u}_i = 1$. The PM can be updated as follows:

$$\text{PM}(\hat{u}_i) = \text{PM}(\hat{u}_{i-1}) + \log(1 + e^{(-1)^{\hat{u}_i} L(\hat{u}_i)}). \quad (6)$$

The goal of SC decoding is to find the most likely path or the path of the highest PM. To achieve this goal, SC decoding takes one branch at a level and keeps the path of the highest PM. The highlighted path on the left in Fig. 2 represents the survival path in SC decoding. The value associated with each node is the PM for the decoding path from the root node to that node. Note that after selecting the survival branch at a level, the other branch and its child nodes will never be considered. In the example, the SC decoder chooses the path [0 0 1 1] with a PM of 0.11.

B. SC List (SCL) Decoding

SC decoding makes one hard decision per step and never visits other possible paths. Though SC decoding is simple to implement, it is not guaranteed to find the best global path because the local optimal path may not be part of the global optimal path. In the example shown in Fig. 2, the global optimal path is [1 0 0 1] with a PM of 0.17, but SC decoding misses this path because it takes the optimal local decision in step 1.

The SCL decoding algorithm [2] overcomes this drawback by keeping a list of L candidate paths at each level in traversing the binary tree. SC decoding can be viewed as a special case of SCL decoding with $L = 1$. Fig. 2 shows SCL decoding for $L = 2$. At each level, the two most likely paths are kept. Moving to the next level, the two most likely paths branch to four child nodes, leading to four candidate paths. The SCL decoder selects the top two paths among the four to keep. In the end, the best path is selected.

In general, an SCL decoder calculates the PMs of all $2L$ child nodes ($\hat{u}_i = 0$ or $\hat{u}_i = 1$) that are connected to the L survival paths from the previous step based on (7). The decoder selects L paths among the $2L$ candidates with the highest PMs to keep. SCL decoding enhances the decoding accuracy with larger L

$$\text{PM}_j(\hat{u}_i) = \text{PM}_j(\hat{u}_{i-1}) + \log(1 + e^{(-1)^{\hat{u}_i} L(\hat{u}_i)})$$
$$0 \le j \le L - 1, \; \hat{u}_i \in \{0, 1\}. \quad (7)$$

The decoding accuracy can be further improved by concatenating polar code with CRC [3], [4]. In addition to ranking PMs, codewords corresponding to valid paths also need to pass CRC check.

An N-bit SCL decoder can be designed with an N-bit SC decoder core and additional processing logic and memory to sort and track L candidate paths. Assume that it takes 1 unit time to sort and track candidate paths after decoding each bit, the SCL decoding latency is $(2N - 2) + N$. However,

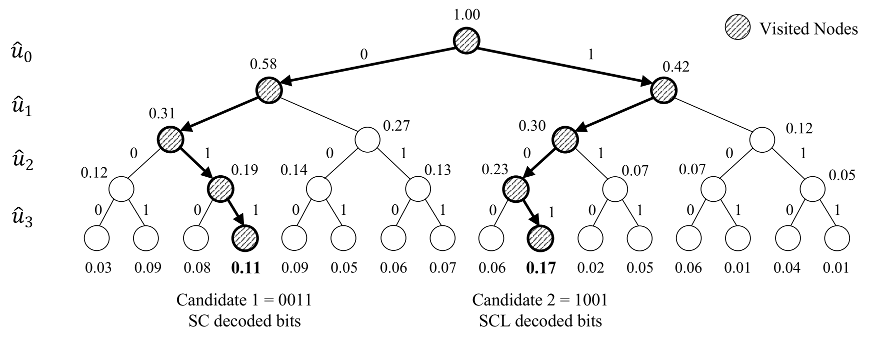

Fig. 2. SC and SCL ($L = 2$) decoding represented in a binary tree for a (4, 4) polar code.

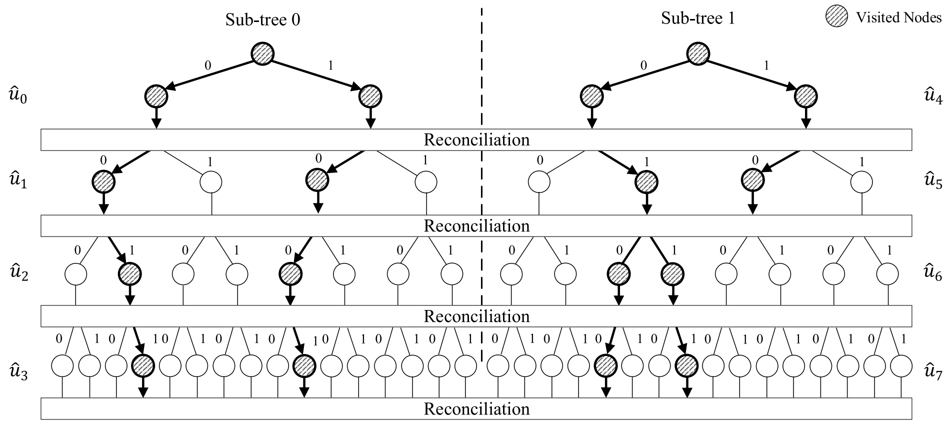

Fig. 3. ST-SCL decoding tree for (8, 8) polar code with list size $L = 2$ and split factor $M = 2$.

the sorting overhead is only incurred if a bit is a free bit. Take the 1024-bit, rate-1/2 polar code selected by the 5G enhanced mobile broadband (eMBB) standard as an example. The code has 512 free bits, so the decoding latency is $(2N-2)+N/2 = 2558$ time units or 25% longer than SC decoding.

C. Split-Tree SCL (ST-SCL) Decoding

To reduce the latency and improve the throughput of SCL decoding, an ST-SCL decoding algorithm [19] was proposed. Conceptually, the N-level decoding tree is split into M sub-trees of N/M levels, equivalent to splitting the N-bit code to M N/M-bit subcodes linked by a constraint matrix. An ST-SCL decoder consists of M N/M-bit SC sub-decoders that operate on the subcodes in parallel. Each SC sub-decoder works on an $M\times$ shorter subcode. The hardware complexity of all M sub-decoders is $O(M \cdot N/M \log_2(N/M)) = N\log_2(N/M)$. The theoretical decoding latency is reduced by a factor of M compared to SC or SCL decoding to $O(N/M)$, and the throughput is increased by the same factor. ST-SCL decoding requires an extra reconciliation stage to combine sub-decoders' local decisions.

To illustrate ST-SCL decoding, suppose that an $N = 8$ polar code is decoded with a list size of $L = 2$ and a split factor of $M = 2$. The eight-level decoding tree is split to sub-tree 0 of four levels and sub-tree 1 of four levels, as shown in Fig. 3, sub-tree 0 for $\hat{u}_0^3$ and sub-tree 1 for $\hat{u}_4^7$. The decoding of each subcode is based on a 4-bit code trellis, similar to Fig. 1. Frozen bits on each sub-tree are determined by the original eight-level decoding tree. ST-SCL decoding proceeds level by level. At level i, the decoding consists of two stages.

1) *Sub-Decoding:* SC sub-decoder 0 and sub-decoder 1 operate on their code trellises and compute the likelihood of bit $\hat{u}_i$ and $\hat{u}_{4+i}$, respectively.
2) *Reconciliation:* The PMs of the four candidate paths per sub-decoder are computed following (7), and the two survival paths per sub-decoder (called sub-paths) are selected. The sub-paths are assembled, checked for constraints, and the top two global paths are selected. The top global paths are then disassembled and distributed to the two sub-decoders.

Stage 1 is the same as in SCL decoding, but stage 2, the reconciliation, is new in ST-SCL decoding. If a bit is not a frozen bit, the sub-decoder provides L sub-paths. In decoding a level, if none of the M bits are frozen bits, there could be up to L^M possible global paths made by combinations of sub-paths. The global PMs (GPMs) for the global paths are calculated by summing the sub-path PMs.

The complexity of the reconciliation stage is proportional to L^M, limiting the maximum split factor and list size. However, some of the L^M global paths are invalid and can be removed from evaluation. For example, in Fig. 3, if u_5 is a frozen bit, the paths with $\hat{u}_5 = 1$ are not valid because $\hat{u}_5$ can only be 0. In the best case, if both $\hat{u}_i$ and $\hat{u}_{4+i}$ are frozen bits at level i, reconciliation is bypassed. Only in the case when both $\hat{u}_i$ and $\hat{u}_{4+i}$ are free bits at level i, all L^M GPMs need to be evaluated. Valid global paths are sorted by GPM. The top L global paths

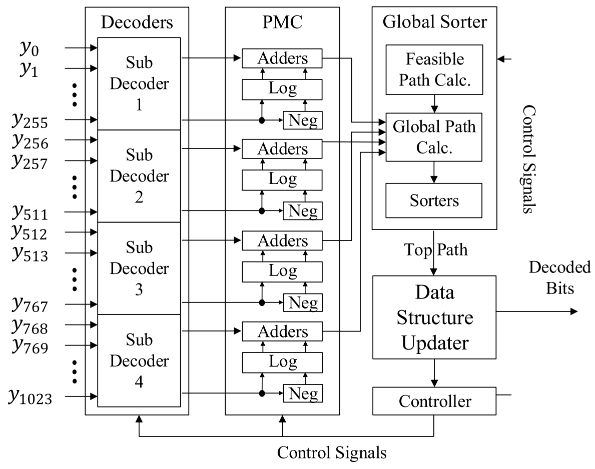

Fig. 4. Top-level architecture for a 1024-bit list-2 split-4 configurable SCL decoder.

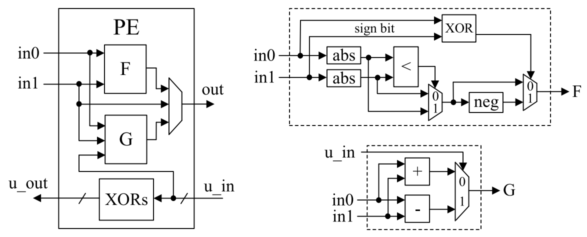

Fig. 5. PE design.

are kept and are disassembled into sub-paths and distributed to the sub-decoders.

III. ST-SCL Decoder Architecture for High Throughput and Low Latency

In this section, we present an ST-SCL decoder architecture to realize the near-theoretical latency and throughput improvements of ST-SCL decoding by an efficient reconciliation stage and a significantly higher utilization of the SC decoding hardware. A prototype ST-SCL decoder is shown in Fig. 4 for the list size $L = 2$ and the split factor $M = 4$. The prototype design supports configurable code length up to $N = 1024$ and variable code rates. In decoding a 1024-bit polar code, the input LLRs are equally split into four groups. A group of 256 LLRs is sent to an SC sub-decoder. The four SC sub-decoders operate on their decoding trellises to compute the 4-bit likelihoods in parallel.

The reconciliation stage is divided to three steps.

1) *PM Calculation:* For each sub-decoder, the PMs of $2L = 4$ candidate paths are calculated following (7) and $L = 2$ sub-paths are selected.
2) *Enumeration and Sorting:* Based on frozen bit information, valid combinations from the $L^M = 16$ sub-path combinations are enumerated, and the GPMs are calculated. The GPMs are sorted to select the top $L = 2$ global paths.
3) *Update:* The top global paths are disassembled and distributed to the four sub-decoders.

The three steps are carried out by PM calculator (PMC), global sorter (GS), and data structure updater (DSU) blocks shown in Fig. 4. We discuss the details of the sub-decoding stage and the reconciliation stage in the following.

A. Sub-Decoder Design

Each SC sub-decoder decodes a polar code of length up to $N = 256$ bits ($n = 8$) by recursively passing through an eight-stage decoder trellis following the sequential order, as shown in Fig. 1. We group an F and a G function in a processing element (PE), as shown in Fig. 5. It consists of an F function described in (4), a G function described in (5) and XORs to compute partial modulo-2 sums of decoded bits. Back routing is needed to feed decoded bits back to G functions. The back routing is implemented by routers between the decoding stages.

A direct mapping of the decoding trellis produces an eight-stage sub-decoder architecture, and each stage consists of 128 PEs. However, the hardware utilization of a direct-mapped architecture is very low. A well-known pattern in SC decoding is that stage i has at most 2^{n-1-i} PEs active at the same time. For example, in Fig. 1, for an $N = 4$ bit polar code ($n = 2$), stage 0 has two PEs active and stage 1 has one PE active in decoding $\hat{u}_0$. Therefore, instead of the direct mapping, we design the sub-decoder by instantiating only 2^{n-1-i} or 2^{7-i} PEs in decoding stage i (named D$_i$), as shown in Fig. 6. In total, the eight stages contain $\sum_{i=0}^{7} 2^{7-i} = 255$ PEs, instead of $128 \times 8 = 1024$ PEs for a direct-mapped architecture.

To achieve a high clock rate, the sub-decoder is pipelined to eight stages, aligned with decoding stages, D0–D7. The pipeline boundaries are shown as dotted lines in Fig. 6. The 255 pipeline registers, L0–L255, store intermediate likelihoods that are propagated forward, and 255 state registers, U0–U255, store the partial modulo-2 sums of the decoded bits that are routed back.

To support a shorter code length, decoding stages can be bypassed. For example, to support a 512-bit code, the code is split into four 128-bit subcodes to be decoded by the four sub-decoders. In each sub-decoder, stage D0 is bypassed by forwarding the 128 input LLRs to the stage D1 PEs. Multiplexers are placed at the inputs to the stage D1 PEs to select either the bypassed input LLRs or the intermediate LLRs from the D0 and D1 router. Bypassing stage D0 shortens the latency and increases the throughput by reducing the pipeline depth from eight to seven stages. The clock inputs to the bypassed D0 and D1 pipeline registers are gated to save power. Similarly, to support a 256-bit code, in each sub-decoder, stages D0 and D1 are bypassed by forwarding the 64 input LLRs to stage D2 PEs. Both D0 and D1 and D1 and D2 pipeline registers are clock gated to save power.

B. Reconciliation Design

A three-stage reconciliation is done by PMC, GS, and DSU. For each sub-decoder, the PMC takes the soft decision of a bit (LLR of a bit decision being 0) from each sub-decoder output and the L survival paths to compute the PMs of $2L$ candidate

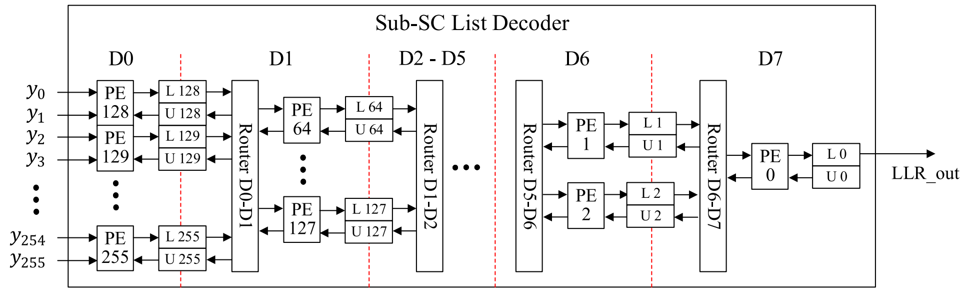

Fig. 6. 256-bit SCL sub-decoder design.

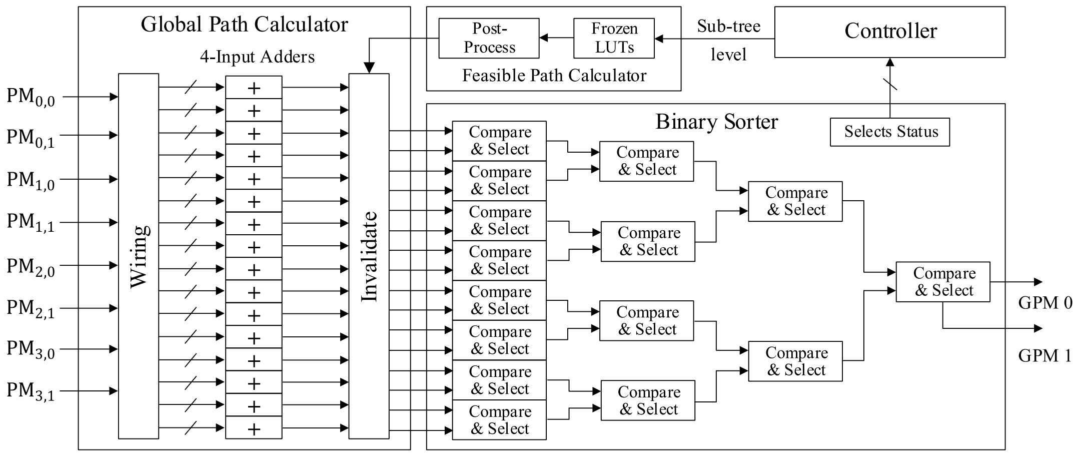

Fig. 7. Split-tree reconciliation GS architecture for $L = 2$ and $M = 4$.

paths by (7). To reduce the complexity of exponentiation and logarithm evaluations, the PMC employs a piecewise linear approximation in the following equation:

$$\log(1+e^x) \approx \begin{cases} 0, & \text{for } x \leq -1 \\ 0.5(x+1), & \text{for } -1 < x < 1 \\ x, & \text{for } x \geq 1. \end{cases} \quad (8)$$

The PMC consists of four sets of hardware, one set per sub-decoder. A set consists of one negation block to compute the LLR of a bit decision being 1, two log-approximation blocks, and four adders to compute the PMs of four candidate paths. From the four candidate paths, the top two candidate paths are selected to be passed on to the GS.

The GS is shown in Fig. 7. It consists of a feasible path calculator, a global path calculator, and a binary sorter. The feasible path calculator uses frozen bit lookup tables (LUTs) to generate control signals for selecting only the valid global paths. The LUTs can be reconfigured to support different code lengths and rates. To save area, the LUTs are implemented as four copies of length-256 cyclic shift registers to store frozen bit indicators for each subcode of up to 256 bits.

The global path calculator sums up all combinations of sub-paths using L^M four-input adders. For $M = 4$ and $L = 2$, there are a total of $L^M = 16$ possible global paths. The complexity of wiring to route the local PMs and the number of adders increase exponentially with the split factor M, limiting the practical M to 4. The 16 GPMs are filtered by the feasible path calculator. The GPMs of the invalid paths are set to the minimum value. The filtered GPMs undergo a four-stage binary sorter to select the top and the second top global paths. The sub-paths that are present in the top and the second top global paths are recorded. Note that the second top global path is approximated by the smaller GPM of the final-stage comparator and this approximation introduces negligible performance loss in mid-to-high SNR regime.

Finally, the DSU disassembles the top L global paths to constitute sub-paths and distributes them to the sub-decoders. The disassembling is done by marking the corresponding local PMs as visited and updating the list state registers in controller. The back-propagation XORs also updates the state registers Us in sub-decoders. The worst case DSU delay happens when the newly decoded bits back-propagate through all eight stages of XOR network inside sub-decoders.

The three-step reconciliation, including PMC, GS, and DSU, occupies only 0.02 mm^2 in 40-nm CMOS when synthesized at a 500-MHz clock frequency.

C. Pipeline and Hardware Utilization

In the ST-SCL decoder, the four sub-decoders operate in parallel and feed to the three-step reconciliation. The sub-decoder is pipelined to eight stages, from D0 to D7; and the three-step reconciliation is pipelined to three stages, PMC, GS, and DSU, abbreviated by P, S, and U, respectively. What complicates the design is that the decoding of any given bit follows a different set of pipeline stages. The irregularity is due to two factors: 1) the variable path through the trellis for

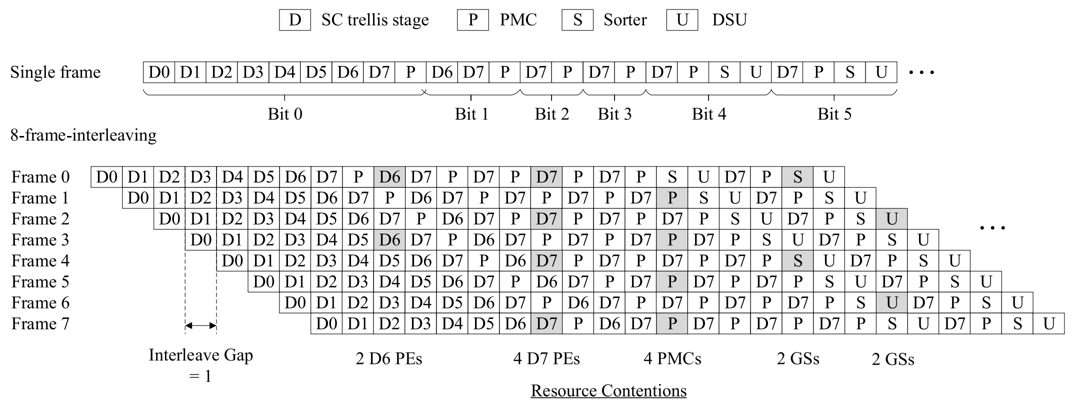

Fig. 8. ST-SCL decoding pipeline for a single frame (top) and eight-frame interleaving (bottom).

decoding a bit, as shown in Fig. 1, and 2) S and U stages can be bypassed if all four bits at a given level are frozen bits.

An example is shown in Fig. 8. Decoding the 4 bits in level 0 by the four sub-decoders requires going through all eight stages of the trellis, corresponding to D0–D7 pipeline stages; decoding the 4 bits in level 1 by the four sub-decoders requires only the last two stages of the trellis, corresponding to D6 and D7 pipeline stages; and so on. In decoding level 0, all four bits are frozen bits, so S and U pipeline stages are bypassed; similarly, in decoding levels 1–3, S and U stages are also bypassed. The irregularity is handled by routers in the sub-decoders and the bypass switches in reconciliation.

To estimate latency and throughput, we use the 1024-bit, rate-1/2 polar code selected by the 5G eMBB standard as an example. The code has 512 frozen bits. We split the code into $M = 4$ subcodes to be decoded by four 256-bit sub-decoders in parallel. Since the latency of SC decoding is $2N-2$ for an N-bit code, decoding a 256-bit subcode requires 510 clock cycles. The reconciliation latency depends on the frozen bit pattern. Among 256 decoding levels, 101 levels involve four bits that are all frozen bits and the S and U stages are bypassed, and the remaining 155 levels involve at least one free bit. Therefore, the reconciliation latency is $101 + 155 \times 3 = 566$ clock cycles. In total, ST-SCL decoding requires $510 + 566 = 1076$ cycles to decode a 1024-bit code or 47% times faster than SC decoding.

The hardware utilization is low if one frame is processed at a time. During sub-decoding, P, S, and U stages are idle, and during reconciliation, D0–D7 stages are idle. Furthermore, during sub-decoding, only one of D0–D7 stages is active at a time. If we define utilization as the average fraction of active hardware units at a given clock cycle, sub-decoders' PE utilization is only 1.57%, PMC's utilization is 23.8%, and GS and DSU utilization are 14.4%. There is ample room to increase the utilization for improving the efficiency and the throughput of the hardware.

IV. Frame Interleaving to Enhance Throughput and Efficiency

To increase the sub-decoders' PE utilization, a straightforward way is to fold the eight stages of 255 PEs in Fig. 6 to one stage of 128 PEs. Folding reduces the PE count to approximately half, allowing the PE utilization to be doubled to 3.14%, which is still low. Complex wiring, muxes, and control logic have to be added to support PE reuse, costing an estimated 24% extra area and 21% longer clock period based on synthesis.

A better approach is to exploit the pipeline to accommodate decoding of multiple frames at the same time. Through frame interleaving, the same hardware is used to process more workload, improving the hardware utilization and increasing the throughput proportional to the number of frames. However, resource contentions may occur. Suppose that eight frames are launched over eight consecutive cycles (interleave gap = 1), as shown in the eight-frame-interleaved pipeline chart in Fig. 8. The highlighted parts show contentions for a hardware unit. Resolving the contentions requires multiple copies of hardware units, including PMCs, GSs, and DSUs. For example, two copies of D6 and four copies of D7 are required in each sub-decoder, and four copies of PMCs and two copies of GSs and DSUs are required in reconciliation.

We studied the optimal number of frames for interleaving by first checking the amount of hardware addition, as shown in Fig. 9. With more frames, more hardware units are needed for the worst case resource contention scenarios. However, due to the general low utilization of the baseline architecture, resource contentions due to frame interleaving are relatively infrequent. As a result, only a small number of hardware units need to be added to the baseline architecture, making frame interleaving a relatively low-cost approach to increasing both throughput and efficiency. For example, to support eight-frame interleaving, only five additional PEs are needed on top of the 255 PEs in each sub-decoder, and four copies of PMCs and two copies of GSs and DSUs are needed for reconciliation. What is not shown in Fig. 9 is that N-frame interleaving also requires N copies of state registers in each sub-decoder as well as muxes to select frames.

Besides hardware duplication, frame interleaving also requires extra control and dispatchers, which become more expensive with more frames. Interleaving more frames produces a higher throughput, but the silicon area increases too. We use chip synthesis in a 40-nm CMOS technology at room temperature and the nominal voltage 0.9 V to evaluate the

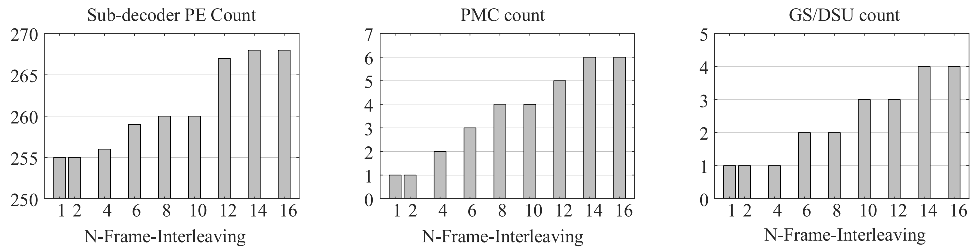

Fig. 9. Allocation of additional hardware units to support N-frame interleaving.

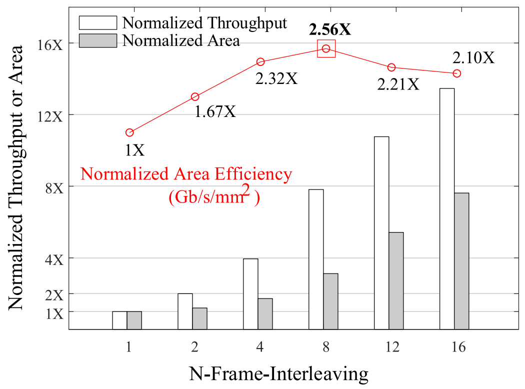

Fig. 10. Throughput and area of frame-interleaved designs. Silicon area is obtained from chip synthesis in 40-nm CMOS at room temperature and the nominal 0.9-V supply voltage.

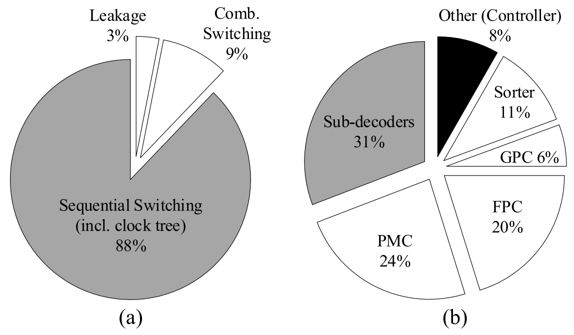

Fig. 12. (a) Power breakdown of a split-4, list-2, eight-frame-interleaved ST-SCL decoder. (b) Detailed breakdown of the sequential switching power of the ST-SCL decoder. Power is obtained from chip synthesis in 40-nm CMOS at room temperature and the nominal 0.9-V supply voltage.

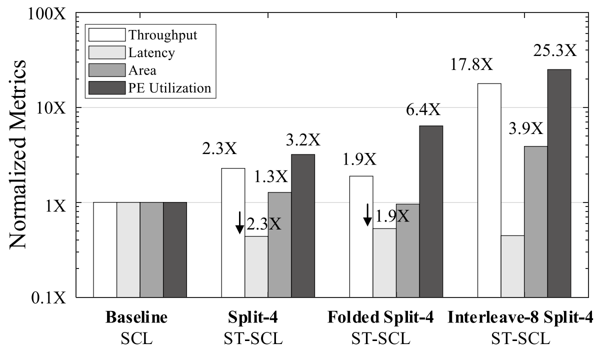

Fig. 13. Chip design optimization summary based on chip synthesis in 40-nm CMOS at room temperature and the nominal 0.9-V supply voltage.

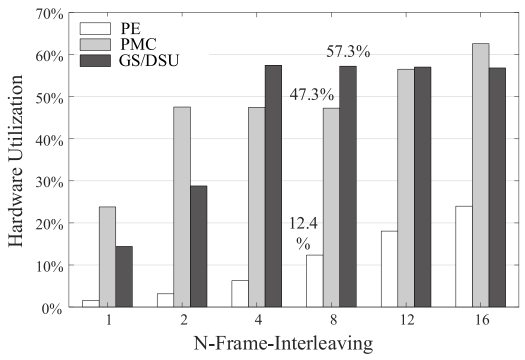

Fig. 11. Improved hardware utilization with frame interleaving.

area with a different number of interleaved frames. As shown in Fig. 10, the increased throughput initially outpaces the increase in silicon area until it reaches eight frames. If we use area efficiency, i.e., throughput/area, as the metric, eight-frame interleaving is optimal, as it increases the throughput by 7.8× and the area by only three times over the baseline architecture.

As shown in Fig. 11, with eight-frame interleaving, two copies of D6 and four copies of D7 are used in each sub-decoder, and the PE utilization increases by 7.8×, from 1.57% to 12.4%. Four copies of PMCs, two copies of GSs and DSUs are also needed, with a utilization of 47.3%, 57.3%, and 57.3%, respectively, which are 2×, 4×, and 4× higher than the baseline.

Frame interleaving proportionally increases the number of state registers. To estimate the power consumption, the split-4, list-2 eight-frame-interleaved ST-SCL decoder was synthesized and placed and routed in a 40-nm CMOS process. Fig. 12(a) shows the power breakdown of the decoder. The switching power of the sequential circuits is the dominant portion, claiming 88% of the total power. Further breakdown of the switching power of sequential circuits in Fig. 12(b) shows that the switching power of the sub-decoders, the PMCs, and the sorters account for more than 90% of the sequential switching power.

V. Summary of Decoder Design Optimization Steps

We summarize the decoder design optimization steps based on 40-nm CMOS synthesis for the code length of 1024 bit and a list size of 2 in Fig. 13 and Table I. The conventional SCL decoder is set as the baseline. The baseline runs at the maximum clock rate of 720 MHz to achieve a 240-Mb/s throughput

TABLE I
CHIP DESIGN OPTIMIZATION SUMMARY BASED ON CHIP SYNTHESIS IN 40-nm CMOS AT ROOM TEMPERATURE AND THE NOMINAL 0.9-V SUPPLY VOLTAGE

Decoder Configuration	Max Clock (MHz)	Throughput (Gb/s)	Latency (ms)	Area (mm2)	PE Utilization	Area Efficiency (Gb/s/mm2)
Baseline SCL	720	0.240	0.43	0.138	0.49%	1.74
Split-4 ST-SCL	575	0.547	0.19	0.176	1.57%	3.11
Folded Split-4 ST-SCL	476	0.453	0.23	0.132	3.14%	3.43
Interleave-8 Split-4 ST-SCL	565	4.274	0.19	0.537	12.4%	7.96

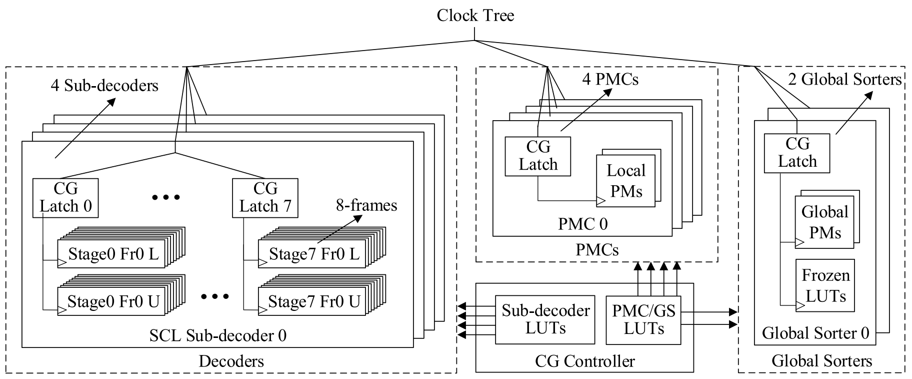

Fig. 14. CG design for split-4, eight-frame-interleaved ST-SCL polar decoder.

and a 0.43-ms latency with a core area of 0.138 mm^2. The PE utilization is only 0.49%.

The split-4 ST-SCL decoder enhances the throughput and latency by 2.3× to 547 Mb/s and 0.19 ms, respectively, while incurring a 30% area penalty. Compared with the baseline, the ST-SCL decoder increases the PE utilization to 1.57% and the area efficiency to 3.11 Gb/s/mm^2. Folding the ST-SCL decoder produces 1.9× higher throughput and lower latency compared with the baseline. Folding also increases the PE utilization to 3.14%.

The split-4 ST-SCL decoder with eight-frame-interleaving boosts the throughput by 17.8× to 4.27 Gb/s and shortens the latency by 2.3× compared with the baseline. The area efficiency is 4.57× better than the baseline. The PE utilization is increased to 12.4%.

We apply a per-block CG strategy to reduce the active power consumption of sequential circuits by exploiting the idle cycles. The PE utilization of the sub-decoders is 12.4%, and the utilization of the reconciliation stage is approximately 50%. By systematically gating the clocks to unused hardware units, the active power is reduced proportionally.

Adding per-block CG increases the area by 2.6% for the eight-frame-interleaved split-4 ST-SCL decoder. CG is implemented with a CG controller sending clock enables to sub-decoders, PMCs, and GSs, as shown in Fig. 14. Clock enable patterns are determined by code configurations, including code length, code rate, and frozen bit locations. For each code, clock enable patterns are pre-computed based on stages D0–D7 active/idle patterns, and they are stored in LUTs inside the CG controller. The decoder top controller sends the code configuration to the CG controller, and the CG controller outputs clock enable signals by reading from the LUTs. In the test chip design, we disable clock input to a sub-decoder stage if the stage will be idle for at least three consecutive cycles to avoid frequent off/on switching. For the shorter code lengths of 512 and 256 bit, CG latch 0 and latch 1 inside sub-decoders will switch off the clock inputs to the bypassed stages. The CG controller also stores the number of required PMCs and GSs for each cycle and disables the clock inputs to unused PMCs and GSs to save power.

VI. DECODER CHIP IMPLEMENTATION AND MEASUREMENT RESULTS

A test chip for the split-4, list-2, eight-frame-interleaved, configurable polar ST-SCL decoder supporting code length up to 1024 bit and variable code rates was implemented in 40-nm CMOS. The chip microphotograph is shown in Fig. 15. The chip measures 0.91 mm × 0.91 mm, and the decoder core measures 0.70 mm × 0.91 mm or 0.64 mm^2. The chip incorporates input buffers to provide input vectors and output buffers to collect the decoded bits. An on-chip CPU with UART interface enables testing of various code lengths, code rates, number of interleaving frames, and CG. It also supports optional post-processing. The chip is verified to be fully functional for the code lengths of 512 and 1024 bit and the code rates of 1/2, 2/3, 3/4, and 5/6.

A. Measurement Results

The bit error rate (BER) and frame error rate (FER) for decoding a split-4, list-2, 1024-bit rate-1/2 code are plotted in Fig. 16. The 6-bit quantized decoder uses an 8-bit CRC

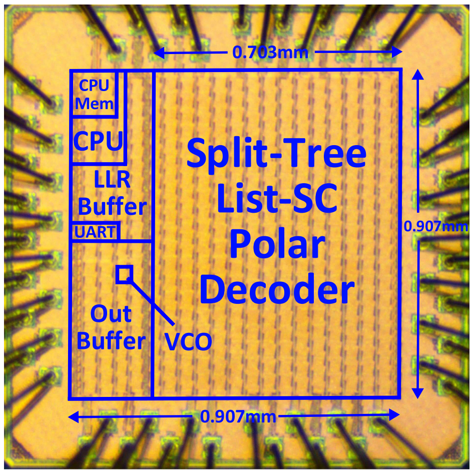

Fig. 15. Microphotograph of the decoder test chip fabricated in a 40-nm CMOS technology.

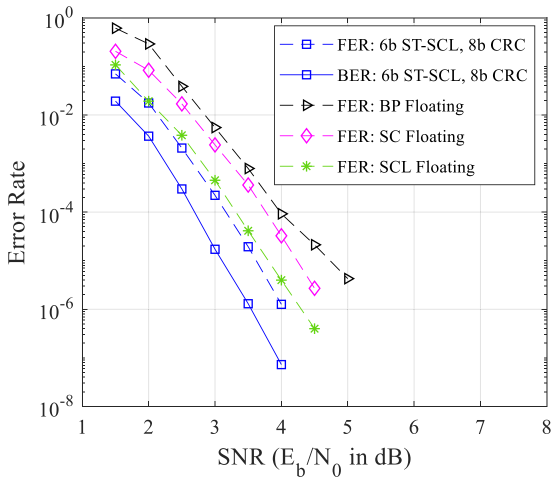

Fig. 16. BER and FER performance of 1024-bit rate-1/2 ST-SCL decoder with split factor 4 and list size 2 using 6-bit quantization and 8-bit CRC.

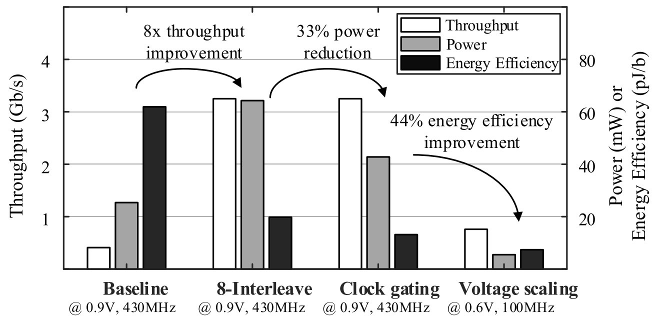

Fig. 17. Measured throughput, power, and energy efficiency of various configurations for the 40-nm ST-SCL decoder chip in decoding 1024-bit rate-1/2 polar codes at room temperature.

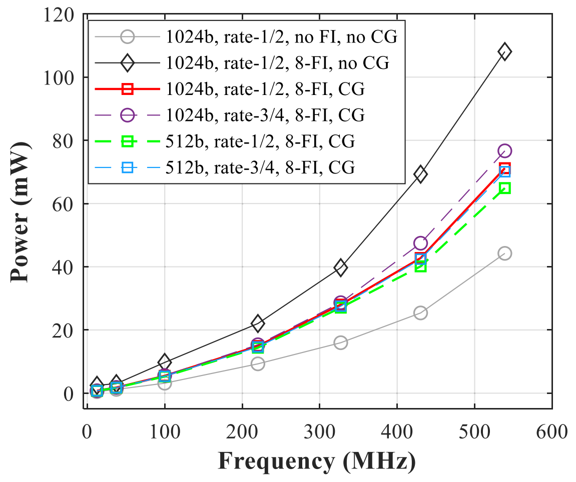

Fig. 18. Measured power of the 40-nm test chip for decoding 512- and 1024-bit polar codes at room temperature and different supply voltages.

to assist with final path selection for a better error-correction performance. The design achieves an FER of 10^{-5} at 3.55 dB, demonstrating 0.15- and 0.65-dB coding gains over the floating-point SCL ($L = 2$) decoder and the floating-point SC decoder, respectively. Compared with the floating-point belief propagation (BP) decoder, our design provides a 1.1-dB coding gain.

The decoder test chip runs at a maximum clock frequency of 430 MHz at a 0.9-V nominal supply voltage and room temperature when decoding 1024-bit, rate-1/2 polar codes. Fig. 17 summarizes the measured throughput and power consumption of the test chip. In the baseline design without frame interleaving and CG, the decoder delivers a 407-Mb/s throughput. It consumes 25.39-mW power, which translates to an energy efficiency of 61.92 pJ/b. To achieve a higher throughput, eight-frame interleaving is enabled to provide an 8× throughput to 3.25 Gb/s at a power consumption of 64.29 mW. The energy efficiency is improved to 19.80 pJ/b, due to the efficient sharing and reusing of under-utilized hardware. The power increase from the baseline to the eight-frame-interleaved design is mainly attributed to three factors: 1) the number of state registers (Ls and Us) in the sub-decoders is increased by 8×; 2) PE, PMC, and GS/DSU utilization are increased to 12.4%, 47.3%, and 57.3%, respectively; and 3) decoder controller and router switching activities are increased to support decoding eight frames in parallel. Among the three factors, factor 1) contributes the most power increase. The sub-decoders are estimated to consume 31% of the total power, 90% of which is sequential power consumed by state registers. Compared with a decoder that supports only one frame at a time, the eight-frame-interleaved design requires eight sets of state registers, and the total chip power increases by about 2.2× (calculated from 31% × 90% ×8) due to the sub-decoder's state register increase. Combining factors 2) and 3), the power of the eight-frame interleaved decoder increases by 2.5× over the single-frame decoder.

CG can be enabled to reduce the power consumption to 42.80 mW and improve the energy efficiency to 13.17 pJ/b. Scaling the supply voltage from 0.9 to 0.6 V reduces the maximum clock frequency from 430 to 100 MHz and further improves the energy efficiency to 7.40 pJ/b.

Fig. 18 shows the power consumption for decoding 1024- and 512-bit codes of rate 1/2 and 3/4, and the effect of frame interleaving and CG. The power was recorded at

TABLE II
COMPARISON OF STATE-OF-THE-ART POLAR DECODERS

	This Work	TCAS-I'20 [20]	TVLSI'19 [21]	arXiv'18 [18]	JETCAS'17 [17]	TCAS-II'19 [22]	ASSCC'18 [16]	TCAS-I'19 [23]
Design	silicon	synthesis	synthesis	silicon	silicon	layout	silicon	silicon
Code	up to 1024b variable rate	1024b rate-1/2	1024b rate-1/2	up to 2^{15}b variable rate	1024b rate-1/2	1024b rate-1/2	1024b rate-1/2	1024b rate-1/2
Decoding Algorithm	Split-tree SCL	SCL-flip/ fast-SCL-flip	stack SCL	SCL	SCL	SC	SC	BP
List Size	2	2	2	8	4	1	1	-
Quantization	6b	6b	6b	6b	6b	5b	custom [24]	5b
Process	40nm CMOS	65nm CMOS	90nm CMOS	16nm FinFet	28nm FD-SOI	180nm CMOS	180nm CMOS	40nm CMOS
Decoder Area (mm^2)	0.637	0.56	0.44	2.27	0.44	1.95	3.17	0.704
Supply (V)	0.9	1.0	-	0.9	1.3	1.8	1.8	0.9
Frequency (MHz)	430	-	1077	1000	721	447	382	500
Throughput (Gb/s)	3.25	1.51	0.764	3.24	0.61	0.30	0.66	7.61
Power (mW)	42.80	-	-	-	128.3	1073 @200MHz	-	422.7
Area Efficiency (Gb/s/mm^2)	5.10	2.71	1.74	1.43	1.39	0.154	0.208	10.81
Energy Efficiency (pJ/b)	13.17	-	-	-	209	7994 @200MHz	-	55.58
**Normalized to 40nm, 0.9V*								
Area Efficiency (Gb/s/mm^2)	5.10	11.63	19.82	0.092	0.48	14.03	18.95	10.81
Energy Efficiency (pJ/b)	13.17	-	-	-	143.1	444.1	-	55.58

* General scaling-theory is used to scale area, frequency (and throughput), and power by $1/s^2$, s, and $1/u^2$ respectively, where s is dimension scale factor and u is voltage scale factor.

the lowest operating voltage at each frequency. From the baseline without frame interleaving to eight-frame interleaving, the power increases as expected, but the per-block CG effectively lowers the power. For a given code length, decoding a higher code rate (in this case 3/4) consumes slightly higher power, due to more switching activities to process more free bits. For a given code rate, decoding a shorter code length costs less power, due to the sub-decoders' bypassing of trellis stages.

B. Comparisons

The ST-SCL decoder test chip is compared with the state-of-the-art polar decoder designs in Table II, including both synthesis results (where no test chip was fabricated and power was not reported) and silicon measurements. Compared with the most recent synthesis results of SCL decoders [20], [21], our ST-SCL chip outperforms by more than 2.15× in throughput and 1.88× in area efficiency than [20] before normalization. After technology normalization to 40-nm and 0.9-V supply voltage, the area efficiency of [20] and [21] surpass our design, which is mainly due to three factors: 1) [20] and [21] do not support variable code lengths and rates; 2) [20] and [21] use SSCL or modified SCL decoding algorithms with performance loss; and 3) [20] and [21] are synthesis results only without silicon measurements. Only silicon results capture the layout and wiring congestion overheads that can be significant in high-throughput decoder designs.

Compared with the recent fabricated silicon SCL polar decoders [17], [18] in more advanced 28- and 16-nm technology nodes, our design exceeds the throughput reported in [17] and [18]. After technology normalization, our design achieves an order of magnitude better area efficiency (in Gb/s/mm^2) and an order of magnitude better energy efficiency (in pJ/b) than [17] and [18] (note that [18] did not report power, and the energy efficiency cannot be estimated).

Compared with the much simpler SC decoder designs [16], [22], the ST-SCL decoder delivers a better error-correction performance as shown in Fig. 16, and the energy efficiency is more than an order of magnitude better after technology normalization. Compared with the most recent BP decoder synthesis [23], the ST-SCL decoder achieves more significant coding gain as shown in Fig. 16, and the energy efficiency is still 4.2× better.

VII. CONCLUSION

We present a fabricated test chip in a 40-nm CMOS technology that implements an ST-SCL decoder for polar codes. In this design, a given polar code is split into four sub-codes and decoded separately with smaller sub-decoders followed by a reconciliation step in every decoding stage. Taking advantage of the under-utilized PEs in the sub-decoders, eight frames are interleaved and decoded in parallel to achieve a high throughput and area efficiency. The decoder supports variable code lengths up to 1024 bit and variable code rates by programming the control LUTs. Per-block CG is implemented to further reduce the power consumption and improve energy efficiency. The 0.64-mm^2 test chip is measured to achieve a decoding throughput of 3.25 Gb/s at 430 MHz at the nominal supply voltage of 0.9 V, consuming 13.17 pJ/b, and it demonstrates a competitive error-correction performance. Voltage and frequency scaling of the chip to 0.6 V and 100 MHz further improves the energy efficiency to 7.4 pJ/b at a reduced throughput of 760 Mb/s. The test chip outperforms the state-of-the-art SCL polar decoder chips in throughput, and its normalized energy efficiency and area efficiency are an order of magnitude better than the latest published work.

Acknowledgment

The authors would like to thank Dr. Farhana Sheikh and collaborators at Intel Labs for advice.

References

[1] E. Arikan, "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, Jul. 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, Jul. 2011, pp. 1–5.

[3] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, Oct. 2012.

[4] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, Dec. 2012.

[5] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[6] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2268–2280, Oct. 2015.

[7] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2378–2391, Jun. 2016.

[8] C. Xiong, J. Lin, and Z. Yan, "A multimode area-efficient SCL polar decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 12, pp. 3499–3512, Dec. 2016.

[9] S. A. Hashemi, M. Mondelli, S. H. Hassani, C. Condo, R. L. Urbanke, and W. J. Gross, "Decoder partitioning: Towards practical list decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3749–3759, Sep. 2018.

[10] M. Mousavi, Y. Fan, C.-Y. Tsui, J. Jin, B. Li, and H. Shen, "Efficient partial-sum network architectures for list successive-cancellation decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 66, no. 14, pp. 3848–3858, Jul. 2018.

[11] C. Xiong, J. Lin, and Z. Yan, "Symbol-decision successive cancellation list decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 64, no. 3, pp. 675–687, Feb. 2016.

[12] S. A. Hashemi, C. Condo, and W. J. Gross, "A fast polar code list decoder architecture based on sphere decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 12, pp. 2368–2380, Dec. 2016.

[13] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, Nov. 2017.

[14] S. A. Hashemi, C. Condo, M. Mondelli, and W. J. Gross, "Rate-flexible fast polar decoders," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5689–5701, Nov. 2019.

[15] D. Kim and I.-C. Park, "A fast successive cancellation list decoder for polar codes with an early stopping criterion," *IEEE Trans. Signal Process.*, vol. 66, no. 18, pp. 4971–4979, Sep. 2018.

[16] H.-Y. Yoon, S.-J. Hwang, and T.-H. Kim, "A 655 Mbps successive-cancellation decoder for a 1024-bit polar code in 180 nm CMOS," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2018, pp. 281–284.

[17] P. Giard *et al.*, "PolarBear: A 28-nm FD-SOI ASIC for decoding of polar codes," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 7, no. 4, pp. 616–629, Dec. 2017.

[18] X. Liu *et al.*, "A 5.16Gbps decoder ASIC for polar code in 16nm FinFET," 2018, *arXiv:1807.01451*. [Online]. Available: http://arxiv.org/abs/1807.01451

[19] B. Li, H. Shen, and D. Tse, "Parallel decoders of polar codes," 2013, *arXiv:1309.1026*. [Online]. Available: http://arxiv.org/abs/1309.1026

[20] F. Ercan, T. Tonnellier, and W. J. Gross, "Energy-efficient hardware architectures for fast polar decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 1, pp. 322–335, Jan. 2020.

[21] W. Song *et al.*, "Efficient successive cancellation stack decoder for polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2608–2619, Nov. 2019.

[22] R. Shrestha and A. Sahoo, "High-speed and hardware-efficient successive cancellation polar-decoder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 7, pp. 1144–1148, Jul. 2019.

[23] Y.-T. Chen, W.-C. Sun, C.-C. Cheng, T.-L. Tsai, Y.-L. Ueng, and C.-H. Yang, "An integrated message-passing detector and decoder for polar-coded massive MU-MIMO systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 3, pp. 1205–1218, Mar. 2019.

[24] H.-Y. Yoon and T.-H. Kim, "Efficient successive-cancellation polar decoder based on redundant LLR representation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 12, pp. 1944–1948, Dec. 2018.

Yaoyu Tao (Member, IEEE) is currently pursuing the Ph.D. degree with the University of Michigan, Ann Arbor, MI, USA.

He is also with the Wireless Research and Development Team, Qualcomm, as a Staff Engineer, where he was promoted from Senior Engineer to Staff Engineer in October 2019. His research interests expand to VLSI circuits, architecture and systems for computing, communications, signal processing, and machine learning applications.

Sung-Gun Cho (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Michigan, Ann Arbor, MI, USA.

From 2012 to 2015, he was with SK Hynix Inc., Icheon, South Korea, where he was involved in system-on-chip (SoC) design and implementation of error control coding. His research interests include energy-efficient, high-performance architecture, VLSI circuits and systems for neuromorphic computing, error control coding, and machine learning applications.

Zhengya Zhang (Senior Member, IEEE) received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley (UC Berkeley), Berkeley, CA, USA, in 2005 and 2009, respectively.

He has been a Faculty Member with the University of Michigan, Ann Arbor, MI, USA, since 2009, where he is currently an Associate Professor with the Department of Electrical Engineering and Computer Science. His research interest includes low-power and high-performance VLSI circuits and systems for computing, communications, and signal processing.

Dr. Zhang was a recipient of the David J. Sakrison Memorial Prize from UC Berkeley in 2009, the National Science Foundation CAREER Award in 2011, the Intel Early Career Faculty Award in 2013, and the University of Michigan College of Engineering Neil Van Eenam Memorial Award in 2019. He has been an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS since 2015. He has been serving on the Technical Program Committees of the Symposium on VLSI Circuits and the IEEE Custom Integrated Circuits Conference (CICC) since 2018. He was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS from 2013 to 2015 and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS from 2014 to 2015.