Tb/s Polar Successive Cancellation Decoder 16nm ASIC Implementation

Altuğ Süral, E. Göksu Sezer, Ertuğrul Kolağasıoğlu, Veerle Derudder and Kaoutar Bertrand

***Abstract*—This work presents an efficient ASIC implementation of successive cancellation (SC) decoder for polar codes. SC is a low-complexity depth-first search decoding algorithm, favorable for beyond-5G applications that require extremely high throughput and low power. The ASIC implementation of SC in this work exploits many techniques including pipelining and unrolling to achieve Tb/s data throughput without compromising power and area metrics. To reduce the complexity of the implementation, an adaptive log-likelihood ratio (LLR) quantization scheme is used. This scheme optimizes bit precision of the internal LLRs within the range of 1-5 bits by considering irregular polarization and entropy of LLR distribution in SC decoder. The performance cost of this scheme is less than 0.2 dB when the code block length is 1024 bits and the payload is 854 bits. Furthermore, some computations in SC take large space with high degree of parallelization while others take longer time steps. To optimize these computations and reduce both memory and latency, register reduction/balancing (R-RB) method is used. The final decoder architecture is called optimized polar SC (OPSC). The post-placement-routing results at 16nm FinFet ASIC technology show that OPSC decoder achieves 1.2 Tb/s coded throughput on 0.79 mm^2 area with 0.95 pJ/bit energy efficiency.**

***Index Terms*—Polar codes, successive cancellation decoding, high-throughput, ASIC implementation, energy efficiency.**

I. Introduction

The Ethernet Alliance foresees a strong demand for Tb/s data throughput for data centers and mobile networks [1]. In the wireless domain, the Horizon 2020 project *Enabling Practical Wireless Tb/s Communications with Next Generation Channel Coding* (EPIC) [2] considers three well-known forward error correction (FEC) schemes turbo [3], Low Density Parity Check (LDPC) [4], [5] and polar codes [6] for extremely high speed beyond 5G applications. This paper aims to present an efficient polar code implementation to meet Tb/s throughput demand. Polar codes have attracted a significant interest from both academia and industry and recently they have been adopted for the protection of the control channel in the enhanced mobile broadband (eMBB) service for the fifth generation (5G) cellular wireless technology [7].

Polar codes are a unique family of FEC schemes which can theoretically achieve capacity in broad class of channels using a low-complexity successive cancellation (SC) decoder [6], [8]. To improve error correction performance of SC algorithm at moderate data block lengths, many algorithms are proposed with most popular one being SC-list (SCL) [9]. SCL algorithm can track a list of possible decision candidates and can achieve near ML performance with this additional complexity. Other SC based decoding algorithms SC-flip (SCF) [10] and Soft-cancellation (SCAN) [11] use an iterative approach to correct decision errors of SC. Using multiple iterations or multiple parallel decoders make these algorithms much more power hungry at Tb/s data rates.

The sequential processing nature of SC limits parallelism within decoder but promotes pipelined approach for Tb/s throughput. There are some high-throughput and pipelined SC implementations [12], [13], [14], [15] in literature. These implementations can only achieve a few Gb/s throughput, even if the unrolled implementations [14], [15] take advantage of a set of shortcuts [16], [17], [18] for speeding up the SC decoder. For a higher throughput, discrete quantization of soft-information [19] and register reduction/balancing (R-RB) [20] methods have been proposed. In [20], Tb/s throughput for polar SC decoder have been investigated. A generic problem identified for Tb/s throughput regime is power density caused by excessive switching activity in a limited core area.

This work proposes an optimized SC (OPSC) decoder architecture based on pipelining and unrolling techniques. The OPSC decoder has low implementation complexity thanks to careful register balancing and adaptive log-likelihood ratio (LLR) quantization (AQ) scheme. This scheme takes LLR distribution of each polar code segment as input and optimizes bit precision of internal LLRs. In addition to that OPSC utilizes R-RB method to optimize clock frequency by flattening time delay of pipeline stages. The post-placement-routing (post-P&R) results at 16nm FinFet ASIC technology show that OPSC decoder achieves 1.2 Tb/s coded throughput (corresponds to 1 Tb/s data throughput) on 0.79 mm^2 area with 0.95 pJ/bit energy efficiency.

A. Summary of the achievements

- OPSC decoder and a channel simulator capable of simulating very low error rates are implemented on FPGA to verify RTL code and measure error correction performance at very low BER. The FPGA implementation results show that OPSC decoder achieves 200 Gb/s throughput and 1.1×10^{-13} bit error rate (BER) at 8 dB Eb/No.
- OPSC decoder exploits AQ and R-RB methods to reduce design area and power.
- To the best of our knowledge, OPSC decoder is the first polar decoder that exceeds 1 Tb/s throughput on ASIC based on post-P&R results.

A. Süral, E. G. Sezer and E. Kolağasıoğlu are with POLARAN, Ankara, Turkey (e-mails: {altug.sural, goksu.sezer, ekolagasioglu}@polaran.com).

V. Derudder and K. Bertrand are with IMEC, Leuven, Belgium (e-mails: {veerle.derudder, kaoutar.bertrand}@imec.be).

Manuscript received September 15, 2020

- The results also show that OPSC decoder archives 0.95 pJ/bit energy efficiency at 0.79 mm^2 area.

The outline of this paper is as follows. Section II gives a short summary of polar coding and introduces the SC decoding with shortcuts and AQ. Section III presents the proposed decoder architecture including pipeline depth optimization. Section IV presents FPGA verification and communication performance of the proposed decoder. Section V presents ASIC implementation details and show comparison with state-of-the-art implementations. Finally, Section VI summarizes the main results with a brief conclusion.

II. Polar Codes and SC Decoding Algorithm

A. Polar codes

Polar codes are a class of linear block codes that exists with a block length $N = 2^n$ for every $n \geq 1$. A polar transform matrix $G_N = G^{\otimes n}$ is the n^{th} Kronecker power of a generator matrix $G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. An input transform vector u_N consists of a user data set $u_{\mathcal{A}}$ to carry the user data d_K with K dimention and a redundancy (frozen) set $u_{\mathcal{A}^c}$ to carry the frozen bits fixed to zero. The polar-encoded codeword x_N is simply obtained as $x_N = u_N G_N$. We refer to [6] for a complete description of the polar coding technique.

B. SC Decoding Algorithm with Shortcuts

The data flow diagram of SC algorithm with certain shortcuts is shown in Fig. 1. At the start of decoding, channel log-likelihood ratio (LLR) vector ℓ_N is given to the input. For an AWGN channel W with the variance σ^2, i$^{\text{th}}$ $(1 \leq i \leq N)$ encoded symbol x_i and received symbol y_i, a channel LLR vector ℓ_i is

$$\begin{aligned} \ell_i &= \ln\left(\frac{W(y_i|x_i=0)}{W(y_i|x_i=1)}\right) \\ &= \ln\left(\frac{e^{\frac{-(y_i-1)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}\right) - \ln\left(\frac{e^{\frac{-(y_i+1)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}\right) \\ &= \frac{-(y_i-1)^2}{2\sigma^2} - \frac{-(y_i+1)^2}{2\sigma^2} \\ &= \frac{2y_i}{\sigma^2}, \end{aligned}$$

where $W(y_i|x_i)$ is the channel transition probability density function. The forward LLR calculation module consists of F and G functions to calculate the inputs ℓ_M for the first and the second half of a polar code, where M is the recursive block length parameter. Initially, $M = N$ and there are $M/2$ size-2 F_2 and G_2 functions (denoted as $\text{F}_{M/2}$ and $\text{G}_{M/2}$) at each recursion. The function $\text{F}_2(\ell_1, \ell_2)$ for any two LLR values ℓ_1 and ℓ_2 is defined as

$$\text{F}_2(\ell_1, \ell_2) = \text{sgn}(\ell_1\ell_2)\min(|\ell_1|, |\ell_2|).$$

The function $\text{G}_2(\ell_1, \ell_2, \hat{z})$ with a hard decision (HD) feedback $\hat{z}$ is

$$\text{G}_2(\ell_1, \ell_2, \hat{z}) = (1 - 2\hat{z})\ell_1 + \ell_2.$$

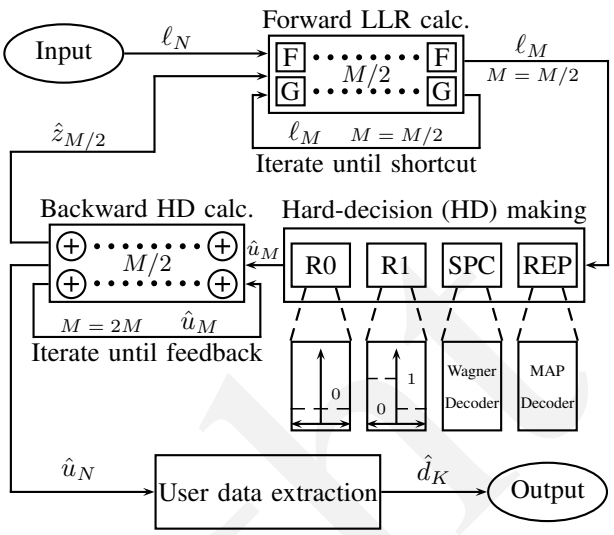

Fig. 1: Data flow diagram of polar SC decoding algorithm with shortcuts

After a bunch of F and G function iterations, the SC decoder becomes ready for making hard decisions using certain shortcuts. These shortcuts are named as Rate-0 (R0), Rate-1 (R1), SPC and REP (first introduced in [17]) for easily decodable polar code segments. For R0 shortcut all values are assigned to 0. For R1 shortcut a simple threshold function is used to assign 0 for positive LLRs and 1 for negative LLRs. Moreover, Wagner [21] and MAP [22] decoders are employed for SPC and REP nodes, respectively. After one of these shortcuts is activated, the backward HD module (can also be named as partial-sum update logic - PSUL) takes the HD estimate $\hat{u}_M$ and calculates the feedback $\hat{z}_{M/2}$ for the G functions. This module utilizes $M/2$ XOR $(\oplus)$ functions at each iteration. After all polar code segments are decoded, the final HD estimate $\hat{u}_N$ is calculated. and the estimated user data $\hat{d}_K$ is extracted from $\hat{u}_N$ at the end of decoding operation.

A formal representation of SC algorithm with shortcuts is shown in Algorithm 1. v_M is the indicator vector of the frozen coordinates for length-M polar code segments. The i$^{\text{th}}$ element of v_M is defined as

$$v_i = \begin{cases} 1, & \text{if } i \in \mathcal{A}^c \\ 0, & \text{if } i \in \mathcal{A}. \end{cases}$$

When v_M is all one, the R0 shortcut is calculated. When v_M is all zero, the R1 shortcut is calculated using a threshold function on LLRs. Both F and G functions are used element-wise for odd ℓ_M^{odd} and even ℓ_M^{even} elements of ℓ_M vector.

1) Shortcuts for ($N = 1024, K = 854$) polar code: For a specific implementation of polar codes, code parameters are selected as $N = 1024$, $K = 854$, $R = \frac{5}{6}$. The selected polar code is constructed using Density Evolution algorithm [23] at 6.5 dB target Es/No. The number of shortcuts for this code is shown in Table I. Due to high code rate, R1 and SPC shortcuts appear more frequent than R0 and REP shortcuts. SPC and REP shortcuts are not allowed to be greater than $N_{\text{LIM}} = 32$ by design choice to keep the target clock frequency high. There are also other shortcuts discovered in [18], however

Algorithm 1: SC with shortcuts

Inputs : ℓ_M, v_M, M **Output:** $\hat{u}_M$

if $v_M = 1$ **then** // R0, R=0
  $\hat{u}_M = \mathrm{d}(\ell_M, v_M = 1) = 0$
else if $v_M = 0$ **then** // R1, R=1
  $\hat{u}_M = \mathrm{d}(\ell_M, v_M = 0)$
else if $M \leq N_{\text{LIM}}$ **and** $v_1 = 1$ **and** $v_2^M = 0$ **then**
  $\hat{u}_M = \mathrm{d}(\ell_M, v_M = 0)$ // Wagner dec.
  $p = \mathrm{mod}(\sum_{i=1}^{M} \hat{u}_i, 2)$ // R = (M-1)/M
  $r = \mathrm{argmin}(|\ \ell_M\ |)$
  $\hat{u}_r = \hat{u}_r \oplus p$
else if $M \leq N_{\text{LIM}}$ **and** $v_1^{M-1} = 1$ **and** $v_M = 0$ **then**
  $\hat{u}_M = \mathrm{d}(\sum_{i=1}^{M} \ell_i, v = 0)$ // MAP dec. R = 1/M
else // Conventional SC ∀ R
  $l_{M/2} = \mathrm{F}(\ell_M^{\text{odd}}, \ell_M^{\text{even}})$ // Fig. 4
  $\hat{z}_{M/2} = \mathrm{SC}(l_{M/2}, v_M^{\text{odd}}, \frac{M}{2})$
  $r_{M/2} = \mathrm{G}(\ell_M^{\text{odd}}, \ell_M^{\text{even}}, \hat{z}_{M/2})$ // Fig. 5
  $\hat{x}_{M/2} = \mathrm{SC}(r_{M/2}, v_M^{\text{even}}, \frac{M}{2})$
  $\hat{u}_M^{\text{odd}} = \hat{z}_{M/2} \oplus \hat{x}_{M/2}$ // Backward HD
  $\hat{u}_M^{\text{even}} = \hat{x}_{M/2}$
if $M = N$ **then** // User data extraction
  $\hat{u}_K = u_{\mathcal{A}}$
  return $\hat{u}_K$
else // Decode remaining code segments
  return $\hat{u}_M$

such shortcuts are not employed in this work due to negligible hardware gain in our target polar code.

TABLE I: Number of shortcuts in (1024,854) OPSC decoder

Node Length	Shortcuts				
	R0	R1	SPC	REP	Total
2	2	2	-	-	4
4	2	4	8	10	24
8	1	3	6	3	13
16	1	3	3	2	9
32	1	6	4	-	3
64	-	3	-	-	7
128	-	1	-	-	1
Total	68	604	256	96	1024

Due to using shortcuts, F/G function and XOR gate gains are shown in Table II. In standard SC decoder architecture, there are $N/2 = 512$ F_2 and G_2 functions at each polar code segment. Therefore, total number of required F_2 and G_2 functions are $N \log N = 10,240$. It reduces to 5276 after applying shortcuts. The gain is mostly caused by the smaller polar code segments. The number of F_2 functions are not equal to G_2 functions for small segments due to R0 shortcuts. For these specific nodes, G functions are used with all zero decision feedback. Furthermore, the required XOR gates for PSUL reduces from $N/2 \log N = 5120$ to 2672.

2) Adaptive quantization of the LLRs: Adaptive quantization is an optimization method to reduce hardware complexity by decreasing LLR bit precision of internal data paths in the SC decoder. Instead of storing and processing LLRs with a constant number of bits, a variable number of bits is used for each polar code segment. During SC decoding of polar

TABLE II: F/G function and XOR gate gain due to shortcuts

Node Length	Number of F/G functions			XOR gates in PSUL
	F	G	Total	
4	-	2	4	2
8	11	13	96	13
16	12	13	200	13
32	10	11	336	11
64	10	11	672	11
128	7	7	896	7
256	4	4	1024	4
512	2	2	1024	2
1024	1	1	1024	1
Total	2604	2672	5276	2672

codes, the polarization phenomenon becomes effective. As polarization increases, resolution can be decreased without losing performance and representing polarized code segments with constant quantization bits becomes inefficient. As polarization increases reliability, resolution of LLR can be decreased. Unlike using lookup tables as in [19] for the computation of LLRs, we use input LLR distribution statistics of each polar code segment and apply the given F and G functions with an optimized bit precision. Adaptive quantization scheme for (1024,854) polar code is shown in Fig. 2. The number of quantization bits are written on the lines between polar code segments. With the adaptive quantization, the memory complexity is reduced by 15.1% while having 4.25 bit average LLR bit precision compared to the SC decoder with 5-bit regular quantization levels. It further reduces combinational logic complexity and enables shallower pipeline depth.

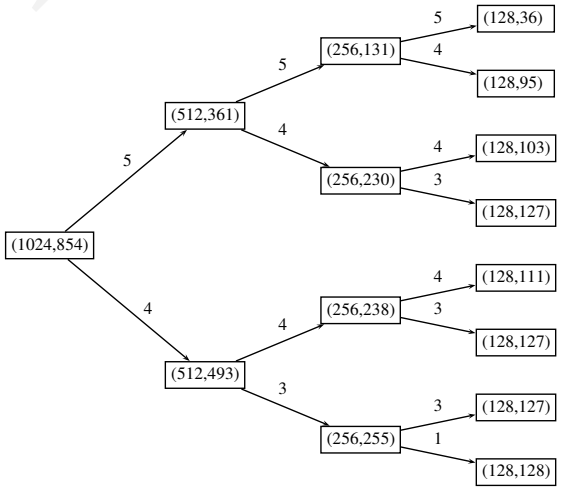

Fig. 2: Adaptive quantization of the constituent codes of SC(1024,854) for $128 \leq M \leq 1024$. The number of quantization bits are written on the lines.

III. OPSC Decoder Architecture

The proposed OPSC decoder architecture exploits pipelining and unrolling techniques to achieve Tb/s data rate while keeping implementation complexity in check. The enabling methods for OPSC decoder are as follows.

- Systematic polar code for improved BER performance.
- Min-sum decoding for simpler arithmetic operations with small area and power dissipation.

- Adaptive quantization of internal LLRs to reduce computational and memory complexity.
- Bit-reversed order computation to operate on neighboring LLRs.
- Unrolled and pipelined architecture for high throughput.
- Fully-parallel SC architecture with multi-bit hard decisions using shortcuts.
- R-RB using pipeline depth optimization for minimum delay and power. Pipeline depth of OPSC decoder is optimized as 158 for FPGA and 60 for ASIC.

The OPSC decoder denoted as OPSC(N, K), consists of two sub-decoders which have the same block length $\frac{N}{2}$ with a different payload $K_i = \frac{N}{2}R_i$. In general, OPSC(N, K) is decoded in four steps: calculation of F functions, OPSC$_1(\frac{N}{2}, K_1)$, calculation of G functions and OPSC$_2(\frac{N}{2}, K_2)$. AQ is applied after F and G functions to reduce LLR bit resolution from Q to Q' bits. For example, the recursive OPSC decoder architecture for $N = 16$, $K = 9$ polar code segment is shown in Fig. 3. Choice of lower layer code rates is exemplary. The ℓ_{16} LLRs at the input with $16 \times Q$ bits are stored during processing duration of OPSC$(8, 2)$ decoder (denoted as $\mathcal{L}$(OPSC$_1$)) until $\hat{z}_8$ becomes ready at the input of function block G. Likewise, $\hat{z}_8$ is stored until $\hat{x}_8$ is ready. Buffer memory structure is used to access data faster than other alternatives such as SRAM.

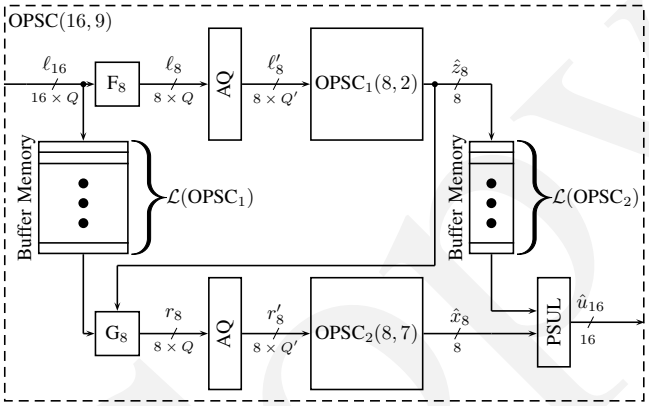

Fig. 3: OPSC(16,9) architecture

Similar hardware architecture is utilized to decode all polar code segments recursively. When shortcuts are detected, OPSC$_1$ and OPSC$_2$ blocks are replaced. For example, OPSC$_2(8, 7)$ is replaced with an SPC shortcut.

A. Building blocks of the OPSC decoder architecture

The basic building blocks of OPSC decoder are F and G functions. F$_N$ function consists of $N/2$ copies of F$_2$ function shown in Fig. 4. The F$_2$ function contains a compare-and-select (C&S) logic and an XOR gate.

G$_2$ function is shown in Fig. 5. The G$_2$ function contains an adder, a subtractor and a multiplexer. The select input $\hat{z}$ of the multiplexer may have longer delay than the $\ell'_1 + \ell'_2$ and $\ell'_2 - \ell'_1$ inputs due to XOR gate chain in PSUL. To avoid timing problems, both results are calculated and the correct one is chosen. Since LLRs are stored in sign-magnitude form, the G$_2$ function also utilizes two sign-magnitude to twos-complement converters (S2C) at the input and a twos-complement to sign-magnitude converter (C2S) at the output.

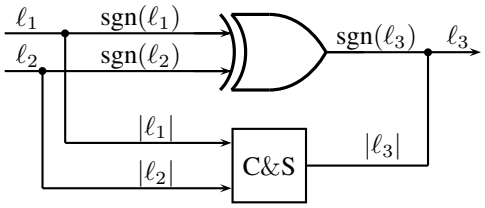

Fig. 4: F$_2$ function.

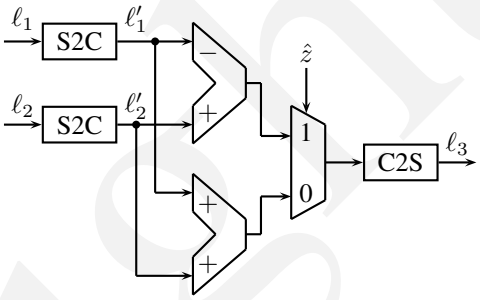

Fig. 5: G$_2$ function.

1) Complexity analysis: The time complexity of fully-parallel standard SC decoder is

$$T_{\mathrm{N}} = 2T_{\mathrm{N}/2} + 2 = \sum_{i=1}^{n} 2^i = 2N - 2 = \Theta(N).$$

The memory complexity of unrolled and fully-pipelined standard SC decoder is

$$\begin{aligned} M_{\mathrm{N}} &= 2M_{\mathrm{N}/2} + (Q + 0.5)(N^2 - N) + NQ \\ &= (Q + 0.5)\big((2 - 2^{-\log N})N^2 - N \log N - N\big) \\ &\quad + QN \log N \\ &= \Theta(N^2 Q). \end{aligned}$$

where $M_2 = 2Q + 1$. This formula shows that in the most general case, memory complexity increases almost quadratically with N. This memory is dominantly used in buffers, where soft decision values are stored. Size of these buffers decreases significantly after applying AQ and R-RB as shown in Table III. The final memory complexity of OPSC decoder is significantly smaller than the conventional SC decoder.

B. Register reduction/balancing

Pipeline stages are important to shorten the critical path and, thus, increase the clock frequency. However, excessive pipeline stages may also increase memory complexity. To reduce the pipeline depth, we merge a set of consecutive short paths of the SC decoder as much as possible based on the combinational delay from timing simulations. Register reduction is challenging for SC decoding algorithm due to its sequential essence. We overcome this problem by estimating delay of each computation and exploiting shortcuts for parallel processing. This method enables the decoder to perform multiple calculations within a single clock cycle with remarkably

reduced latency and memory consumption. Table III shows that LLR buffer memory of OPSC decoder is reduced from 1.1 Mb to 380 Kb using R-RB at $Q = 5$ bits. After applying R-RB, the pipeline depth becomes 60. The results include shortcuts without AQ. Applying the proposed AQ scheme in Fig. 2 further reduces the LLR buffer memory to 361 Kb. Including PSUL memory, total buffer memory becomes 380 Kb. The memory gain of AQ is marginal, because R-RB scheme has already reduced the memory significantly.

TABLE III: Buffer memory of OPSC decoder

Node Length	Buffer memory depth w/o R-RB		Buffer memory depth with R-RB	
	LLR	PSUL	LLR	PSUL
4	-	18	-	-
8	22	29	-	5
16	44	36	15	14
32	60	44	20	15
64	76	50	29	22
128	93	49	29	21
256	103	53	34	20
512	103	46	37	17
1024	112	-	41	-
Total size	1.1 Mb	49 Kb	380 Kb	19 Kb

IV. FPGA Verification and Performance

Error correction performance of OPSC decoder was verified on Xilinx (xcvu37p-fsch2892-2L-e-es1) FPGA demo board. To attain real-time verification, FPGA architecture is developed for both polar systematic encoder and OPSC decoder. The rest of this section presents FPGA test platform and error correction performance of OPSC decoder.

A. FPGA test platform

Polar decoder implementations were verified on FPGA test platform shown in Fig. 6. The test platform supports 200 Gb/s information throughput at 234 MHz clock frequency. A linear feedback shift register (LFSR) array generates $K = 854$ bit pseudo random data for each transmitted polar codeword. A systematic polar encoder generates $N = 1024$ bit encoded data from the pseudo random data. The encoded data, x_i, consists of 854 systematic bits and 170 parity bits, where both bits are mapped to BPSK symbols (s_i) using the mapping rule in Eq. 1. The symbols are accumulated with the additive white Gaussian noise (AWGN) generated by a build-in Gaussian random number generator. BPSK demodulator generates LLR values with $Q = 5$ bits in the form of sign-magnitude. The polar SC decoder processes LLRs and produces information bit estimates, which are compared with the original data to produce error statistics.

$$s_i = \begin{cases} 1, & \text{if } x_i = 0 \\ -1, & \text{otherwise.} \end{cases} \quad (1)$$

B. FPGA performance results

The frame error statistics in Fig. 7 show that the FPGA implementation of OPSC decoder causes less than 0.1 dB performance loss compared to floating-point software simulation (without AQ). The performance loss is caused by

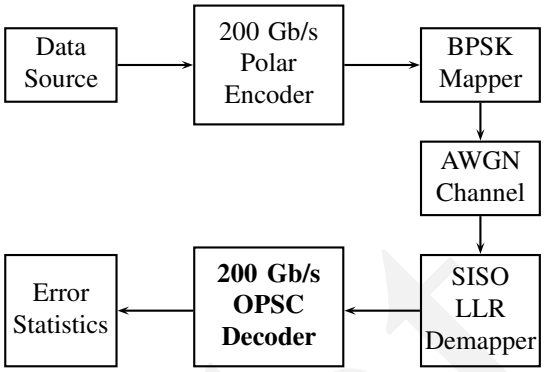

Fig. 6: Block diagram of FPGA test platform

5-bit quantization of LLRs on FPGA. The proposed AQ scheme causes almost 0.1 dB more performance loss, which is tolerable due to hardware implementation gains.

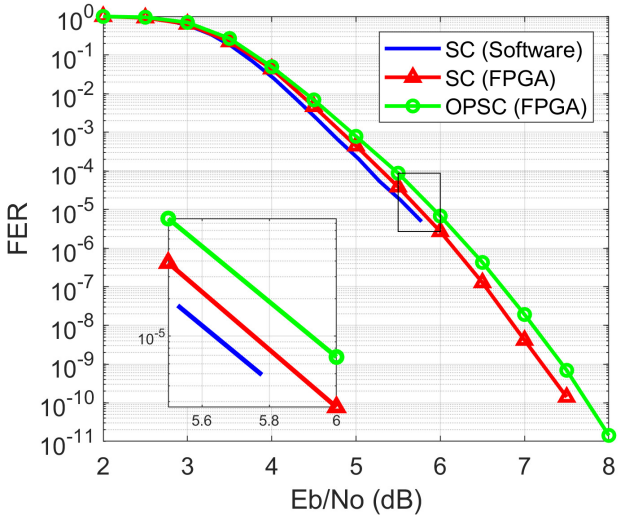

Fig. 7: FER performance of (1024,854) polar code

BER performance results in Fig. 8 show that a coding gain of $13.93 - 7.72 = 6.21$ dB is attained at 10^{-12} BER relative to uncoded transmission.

FPGA implementation results of polar encoder and OPSC decoder are shown in Table IV. Both encoder and decoder have 234 MHz clock frequency. To support this frequency, OPSC decoder utilizes register array memory in the form of LUT for storing internal LLRs. The received LLRs are stored in 143 block random access memory (BRAM) with 16K capacity. The results also show that OPSC decoder utilizes only 7.34 % of the FPGA in terms of LUT consumption. OPSC decoder can fit low-cost FPGA boards such as Xilinx Artix-7.

TABLE IV: FPGA resource utilization

	LUT	FF	Power (mW)	Latency (ns)
Polar encoder	1848	1825	36	8.5
OPSC decoder	95653	50843	2373	672

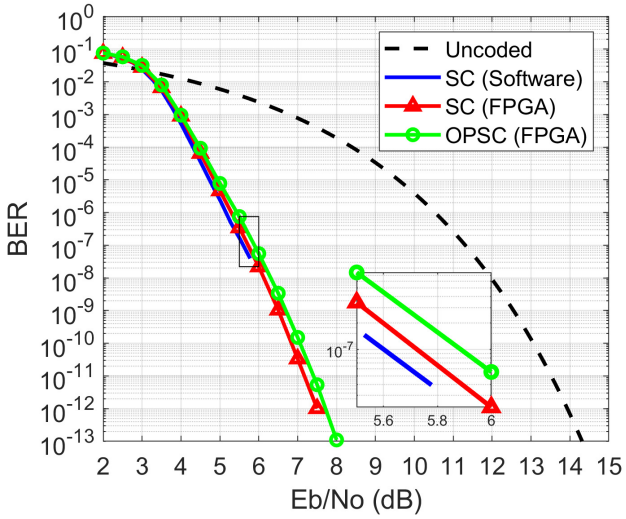

Fig. 8: BER performance of (1024,854) polar code

V. ASIC Implementation

This section presents ASIC implementation procedure and results. The general information about the ASIC implementation is as follows.

- The TSMC 16nm CMOS logic FinFet library (BWP16P90) is used for RTL synthesis and backend implementation.
- The process-voltage-temperature (PVT) values are 0.8 V and 25°C. Although timing is satisfied at 0.7 V at the end of RTL synthesis, 0.8 V is chosen for backend implementation to make timing closure easier.
- The logic cells with a set of thresholds RVT, LVT, OPT-LVT and ULVT are used.
- The setup and hold time of each design are verified for typical, worst-C, worst-RC, best-C and best-RC design corners.
- The power results are estimated accurately by generating signal activity factors using a testbench. The testbench simulates consecutive decoding of 1000 codewords transmitted at 0-9dB Eb/No. For each Eb/No value 100 codewords are tested.
- To achieve timing-clean results, a noticeable number of buffers and inverters have been added to the design.
- The final implementation results are obtained at the end of a timing-clean P&R.

A. Synthesis

The 0.8V TSMC 16nm logic FinFET plus 1P13M process is used for implementation. Physical synthesis is performed with Cadence Genus. The OPSC has been implemented for a clock frequency of 1.2 GHz. To cope with the high clock frequency, retiming has been adopted to move the pipeline registers across the combinatorial logic. In addition, during synthesis fine-grained clock gating has been performed to reduce the dynamic power.

Initially, OPSC decoder was synthesized with 0.7V and 0.8V supply voltages at 1.2 GHz clock frequency. Synthesis results in Table V show that 73 paths have timing violations at 0.7V. These violations can be fixed by reducing clock frequency or increasing the number of pipeline stages. The former is not possible to achieve our Tb/s throughput target. The latter increases the number of DFF and therefore complexity of clock distribution network. This may cause routing problems at backend implementation stage. Since we want to keep pipeline depth as small as possible,the backend study is performed for only OPSC decoder at 0.8V.

TABLE V: Synthesis results of OPSC decoder at 0.7V and 0.8V supply voltages

Supply Voltage (V)	Cell Area (um^2)	# of DFF	WNS (ps)	TNS (ps)	# of violating paths
0.8	462,386	397,030	0	0	0
0.7	468,132	395,935	-5.7	-166.3	73

B. Floorplan

Physical floorplan for OPSC decoder is shown in Fig. 9. Input and output pins are placed at top and bottom of the chip, respectively. Rectangular shape is adopted to increase area utilization under flat placement. The total area of the chip is $(1.255)(0.63) = 0.79$ mm^2.

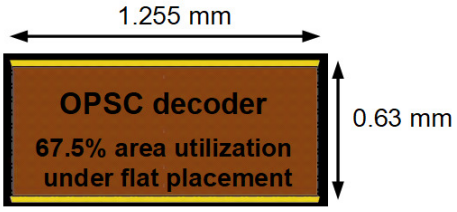

Fig. 9: Physical floorplan of the OPSC decoder

C. Placement and Routing

Virtual silicon tape out is a technique to conceptually validate on silicon without making a real tape out. It consists of going through all the implementation and signoff phases and doing all the simulations required to validate the design without send it to the fabrication. In this study, we have completed virtual silicon design flow up to post-place-and-route stage with timing closure.

The physical implementation on this design was done with Cadence INNOVUS tool, using the exact same libraries as the synthesis and using the signoff timing recommendations provided by TSMC for OCV. Timing is closed only on SSG0.72V and TT0.8V corners in all temperatures. We also followed all the recommendations in terms of power and layout signoff. Additional cells added through different steps of the implementation is shown in Table VI.

The spread of cells across libraries are shown in Table VII.

D. ASIC implementation results

The implementation results of the OPSC decoder in Table VIII show that the decoder utilizes 906K instances in 0.5

TABLE VI: Additional cells at post-place, post-cts and post-route steps

	Post-place	Post-cts	Post-route
Density	60.48	66.95	67.49
Cells to fix setup/transition	596	6119	450
Latency average	N/A	600 ps	600 ps
Cells added to fix hold	N/A	110452	116018

TABLE VII: Cell types of OPSC decoder

	RVT	LVT	OPT-LVT	ULVT
Number of cells	637,149	207,964	234	61,618
% of cells	70.27	22.94	0.03	6.8

mm^2 cell area. Even after our optimizations, the design is still register dominated due to deeply pipelined architecture to cope with 1.2 GHz clock frequency. Since register cells are usually larger than the other cells, 70% of the area is occupied by registers. The second largest area belongs to combinational logic to process LLRs and produce hard decisions. The total power dissipation is 1.2 W, while the leakage power is only 2.4 mW. The registers clocked at 1.2 GHz has 52.8% of the total power dissipation. Hold-fix and setup-fix buffers also have significant low power dissipation.

TABLE VIII: Implementation results of the OPSC decoder

Cell type	Instances		Area		Power	
	Value	%	um^2	%	mW	%
Registers	401,132	44.2	357,327	69.8	616.2	52.8
Inverters	29,575	3.3	4,055	0.8	35.3	3.0
Buffers	154,379	17.0	52,087	10.2	261.4	22.4
Clk. latches	353	0.04	453	0.1	6.0	0.5
Comb. logic	321,292	35.4	97,683	19.1	248.6	21.3
Total	906,731	100	511,605	100	1,167	100

E. ASIC implementation comparison with other high throughput polar decoders

A comparison of the proposed OPSC decoder implementation with the state-of-the-art polar SC decoder implementations is shown in Table IX. The results are scaled to the same 0.8V supply voltage and 16nm process technology for a fair comparison. As for common scaling factors given in [24], the area is scaled in proportion to the square of the process ratio; the power is scaled in proportion to the square of voltage ratio and linear to process ratio; energy efficiency is scaled in proportion to the square of the process ratio times the square of the voltage ratio.

The synthesis results show that OPSC decoder has a noticeable area efficiency, energy efficiency, and latency advantage compared to others. The unrolled implementation [25] provides immense throughput at high clock frequency; however, it consumes too much power. The combinational decoder implementation [26] is favorable in terms of power and power density; however, it has extremely low throughput to satisfy the high throughput requirements of certain applications.

The comparison of the post-P&R results of this work with the results of fabricated ASICs is shown in Table X. The scaled results show that this work has ultra-low latency and it can achieve Tb/s throughput under a reasonable area and power budget. The area efficiency result of this work is more than 10 times greater than the others. It is remarkable for this work to achieve 1.2 W power and 0.95 pJ/bit energy efficiency at 16nm FinFet technology.

TABLE IX: Synthesis results comparison with the high throughput polar decoders

Implementation	This work	[25]	[26]
ASIC Technology	16nm	28nm	90nm
Block Length	1024	1024	1024
Code Rate	0.83	0.5	Flex.
Supply Voltage (V)	0.8	1.0	1.3
Coded Throughput (Gb/s)	1229	1275	2.6
Frequency (MHz)	1200	1245	2.5
Latency (μs)	0.05	0.29	0.40†
Latency (Clock Cycles)	60	365	1
Area (mm^2)	0.47	4.63	3.21
Power (mW)	1072	8793	191
Scaled to 16nm and 0.8V using the common scaling factors in [24].			
Coded Throughput (Gb/s)	1229	**2231**	14.4
Frequency (MHz)	1200	**2179**	14
Latency (μs)	**0.05**	0.17	0.07
Area (mm^2)	0.47	1.51	**0.10**
Area Eff. ($Gb/s/mm^2$)	**2590**	1477	142
Power (mW)	1072	3216	**13**
Power Density (mW/mm^2)	2260	2128	**126**
Energy Eff. (pJ/bit)	**0.87**	1.44	0.89

†Not presented in the paper, calculated from the presented results

TABLE X: Implementation results comparison with fabricated ASICs

Implementation	This work†	[27]	[28]
ASIC Technology	16nm	28nm	16nm
Block Length	1024	1024	32768
Code Rate	0.83	0.85	0.85
Supply Voltage (V)	0.8	0.9	0.9
Coded Throughput (Gb/s)	1229	9.23	4.44
Frequency (MHz)	1200	451	1000
Latency (μs)	0.05	0.63	-
Latency (CCs)	60	283	-
Area (mm^2)	0.79	0.35	0.35
Power (mW)	1167	10.6	-
Scaled to 16nm and 0.8V using the common scaling factors in [24].			
Coded Throughput (Gb/s)	**1229**	16.16	4.44
Frequency (MHz)	**1200**	789	1000
Latency (μs)	**0.05**	0.36	-
Area (mm^2)	0.79	**0.11**	0.35
Area Eff. ($Gb/s/mm^2$)	**1554**	143	12.7
Power (mW)	1167	**5**	-
Power Density (mW/mm^2)	1473	**43**	-
Energy Eff. (pJ/bit)	0.95	**0.30**	-

†Post-P&R results are given for this work.

VI. Conclusion

In this paper, an optimized implementation of SC decoder based on AQ and R-RB methods is proposed for polar codes. These methods not only reduce implementation complexity, power and area but also enable Tb/s throughput on ASIC. Hardware architectures of (1024,854) OPSC decoder are developed for FPGA and ASIC. For FGPA, 200 Gb/s throughput is achieved and hardware verification of OPSC decoder is completed by measuring 1.1×10^{-13} BER at 8 dB Eb/No. The ASIC implementation results show that OPSC decoder achieves 1.2 Tb/s coded throughput, 1554 $Gb/s/mm^2$ area efficiency and 0.95 pJ/b energy efficiency. When OPSC decoder is compared with other fabricated ASICs for polar codes, it has

16 times more throughput, 7.2 times less latency and 10 times better area efficiency than the best alternative implementation.

ACKNOWLEDGMENT

This work has been supported by EPIC project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 760150. The authors would like to thank Prof. Norbert Wehn and Mr. Claus Kestel for useful discussions and comments.

REFERENCES

[1] "Ethernet roadmap 2019." [Online]. Available: https://ethernetalliance.org/wp-content/uploads/2019/08/EthernetRoadmap-2019-Side1-ToPrint.pdf.

[2] "EPIC - Enabling practical wireless Tb/s communications with next generation channel coding." [Online]. Available: https://epic-h2020.eu/results.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070 vol.2, May 1993.

[4] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, January 1962.

[5] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–, Aug 1996.

[6] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, pp. 3051–3073, July 2009.

[7] 3GPP, "New radio NR; multiplexing and channel coding (3GPP TS 38.212, Version 15.7.0)," tech. rep., 2019.

[8] E. Sasaoglu, E. Telatar, and E. Arikan, "Polarization for arbitrary discrete memoryless channels," pp. 144 – 148, 11 2009.

[9] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, pp. 2213–2226, May 2015.

[10] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, pp. 2116–2120, 2014.

[11] U. U. Fayyaz and J. R. Barry, "Low-complexity soft-output decoding of polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 958–966, 2014.

[12] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Transactions on Signal Processing*, vol. 61, pp. 2429–2441, May 2013.

[13] O. Dizdar and E. Arıkan, "A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, pp. 436–447, March 2016.

[14] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "237 gbit/s unrolled hardware polar decoder," *Electronics Letters*, vol. 51, no. 10, pp. 762–763, 2015.

[15] P. Giard, G. Sarkis, C. Thibeault, and W. J. Gross, "Multi-mode unrolled architectures for polar decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, pp. 1443–1453, Sept 2016.

[16] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Communications Letters*, vol. 15, pp. 1378–1380, December 2011.

[17] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 946–957, May 2014.

[18] M. Hanif and M. Ardakani, "Fast successive-cancellation decoding of polar codes: Identification and decoding of new nodes," *IEEE Communications Letters*, vol. 21, pp. 2360–2363, Nov 2017.

[19] S. A. A. Shah, M. Stark, and G. Bauch, "Design of quantized decoders for polar codes using the information bottleneck method," in *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*, pp. 1–6, Feb 2019.

[20] A. Süral, E. G. Sezer, Y. Ertuğrul, O. Arıkan, and E. Arıkan, "Terabits-per-second throughput for polar codes," in *2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)*, pp. 1–7, Sep. 2019.

[21] R. Silverman and M. Balser, "Coding for constant-data-rate systems," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, pp. 50–63, September 1954.

[22] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Comm.*, vol. 47, pp. 673–680, May. 1999.

[23] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[24] C. Wong and H. Chang, "Reconfigurable turbo decoder with parallel architecture for 3gpp lte system," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, pp. 566–570, July 2010.

[25] P. Giard, C. Thibeault, and W. J. Gross, *High-Speed Decoders for Polar Codes*. Springer, 2017. pp. 66–67.

[26] O. Dizdar, *High Throughput Decoding Methods and Architectures for Polar Codes with High Energy-Efficiency and Low Latency*. PhD thesis, Bilkent University, 2017.

[27] P. Giard, A. Balatsoukas-Stimming, T. C. Müller, A. Bonetti, C. Thibeault, W. J. Gross, P. Flatresse, and A. Burg, "Polarbear: A 28-nm fd-soi asic for decoding of polar codes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, pp. 616–629, Dec 2017.

[28] X. Liu, Q. Zhang, P. Qiu, J. Tong, H. Zhang, C. Zhao, and J. Wang, "A 5.16gbps decoder asic for polar code in 16nm finfet," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 1–5, Aug 2018.