A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic

Onur Dizdar, Student Member, IEEE, and Erdal Arıkan, Fellow, IEEE

Abstract—Power and energy-efficiency characteristics of polar decoders have not been studied in most of the decoder designs suggested so far. In this paper, we propose a high-throughput energy-efficient Successive Cancellation (SC) polar decoder which is completely composed of combinational circuits. The proposed decoder is implemented using the recursive structure of SC decoding and decodes one codeword in a single clock cycle with low dynamic power consumption. The characteristics of the combinational decoder is investigated with ASIC and FPGA implementations. It is shown that the combinational decoder architecture consumes 12.31 picojoules-per-bits (pJ/b) with 2.92 gigabits-per-second (Gb/s) throughput with 90 nm technology.

Index Terms—Polar codes, successive cancellation, error correcting codes, VLSI, energy efficiency.

I. INTRODUCTION

Polar codes were recently proposed in [1] as the first channel coding method that can be proved to achieve Shannon’s channel capacity algebraically in binary erasure channels. The basic principle of polar coding is to convert the real transmission channels with arbitrary capacities into new channels that have capacities approaching either 0 or 1. This channel conversion is achieved by a matrix multiplication at the transmitter, making polar codes a type of linear block codes. The well-defined structure and error probability performance of polar codes make them a good candidate for use in emerging systems and standards. Therefore, decoding architectures for polar codes have attracted a great attention since the codes were first proposed.

A. Motivation

Studies conducted on polar decoding so far have mainly focused increasing the throughput while keeping the hardware complexity as low as possible. Increasing the throughput and decreasing the hardware complexity without making major changes in the structure of an algorithm generally brings a penalty in the power consumption. However, power consumption and energy efficiency are also important parameters along with throughput and hardware usage in designing decoder architectures, and should be analyzed more thoroughly. In this work, a SC polar decoder that mainly aims a high throughput performance with low power consumption is proposed. The proposed architecture is a purely combinational circuit implemented using the non-iterative recursive structure of SC algorithm and synchronized only when taking inputs and giving outputs. The purpose of this design is to investigate the limits of a SC decoder when hardware usage is not the primary concern. We will show that this design achieves a low power consumption with high throughput by ASIC and FPGA implementations and comparisons with state-of-the-art decoders.

B. Literature Survey

Studies for polar decoder architectures have mainly focused on three different decoding algorithms: Belief Propagation (BP), SC, and SC list decoding. BP algorithm can be used for polar decoding as any other linear block code. SC is the first proposed algorithm for decoding of polar codes in [1]. It is a non-iterative algorithm and a special case of BP, in which the messages passed between nodes are in their most reliable form once they are calculated. SC list decoding is an extension of SC decoding in which more than one SC decoding paths are kept in memory in order to increase the chance of finding the correct codeword. The architectures [2]-[5] are proposed for BP decoding of polar codes. In [2], a fully-parallel BP decoder architecture is considered. The design is implemented on GPU processors and decodes different codewords in parallel, yielding a throughput of 3.55 Mb/s for block length 1024. The design proposed by [4] uses different numbers of parallel processing units that calculate the messages in groups. The number of processing units is directly proportional with hardware complexity and inversely proportional with latency of the design. The decoder provides 27.83 Mb/s with 65 nm FPGA technology for block length 1024. Decoding delay is reduced in [4] by performing calculations in sign-magnitude form and an efficient conversion method from sign-magnitude form to two’s complement form. By this approach, the throughput and hardware efficiency are claimed to be increased by 30% and 80%, respectively, compared to [3]. An overlapped scheduling on both iteration and codeword levels is used in [5] to increase hardware utilization. Provided design results show that 2 Gb/s throughput is achieved with 45 nm technology. Architectures for SC list decoding are proposed in [6]-[9]. The implementation results on ASIC are provided in all of the works for list sizes of 2 and 4, which are practical to implement and provide significant decoding gains with respect to SC algorithm. The maximum throughput is achieved by
using 65 nm technology: 500 Mb/s with 1.06 mm² area usage for list size 2. The power consumed by this architecture is given as 395 mW, which means 790 pJ energy is required to decode one bit.

The studies in [10]-[19] consider SC polar decoding with different levels of parallelization and pipelining. In all of the studies, the architectures are composed of basic processing elements that are able to perform all the algebraic operations in SC algorithm and storage elements that store the intermediate calculated values between these processing blocks. A semi-parallel architecture with resource sharing is suggested in [10], presenting the trade-off between the hardware usage and latency. The architecture is designed to conduct less number of parallel calculations in a clock cycle than the conventional SC algorithm. This reduces the hardware usage of the architecture in return for an increase in decoding latency and storage elements. The throughput for block length 1024 is given to be 87 Mb/s in 40 nm FPGA and 246.1R Mb/s in 65 nm ASIC on 309 μm² area, most of which is spanned for storing the intermediate calculation results. Semi-parallel architecture is also used in [11] and [12], which suggest solutions to the throughput-limiting partial-sum update problem in SC decoder architectures. Throughput is shown to increase with more efficient partial-sum update designs in both studies. In [13], polar codes are represented as array codes to separate SC decoding into two parts: one part for columns and one part for rows of the generator matrix of polar code. This reduces the complexity to $O \left( \sqrt{N} \right)$ and enables a smaller update logic for partial-sums. The latency increases from $2N - 2$ (for conventional SC decoder of [11]) to approximately $2.5N$, however, as the decoder size is reduced, the use of registers instead of RAM’s to store intermediate calculations becomes feasible, increasing the allowed clock frequency. The maximum throughput is increased to 112 Mb/s in 40 nm FPGA for block length 1024.

In [14] and [15], a precomputation look-ahead method for SC decoding is proposed. This method decreases the latency to $N - 1$ by doubling the number of adders used in processing elements. The same look-ahead technique is used in [20] and [16] with increased resource utilization due to pipelining. A decoder with multiple time-saving methods is proposed in [21], which make use of decoding two bits simultaneously and precomputation of necessary calculations, reducing the latency down to $3N/4 - 1$.

Benefiting from the simplified decoding structures of high and low rate constituent codes, [17] proposes a SC decoder structure using the simplified successive cancellation (SSC) of [18] and simplified successive cancellation with maximum likelihood nodes (ML-SSC) of [19]. The decoder architecture becomes highly dependent on the frozen bit locations and code rate, which also affects the latency and resource usage. The results provided for block length 32768, for which the benefits of the proposed method become significant, show a throughput of up to 1 Gb/s for 40 nm FPGA.

C. Organization

The paper is organized as follows: In Section II, background information on polar codes and structures of encoder/decoder operations are given. Section III provides the structure of the combinational polar decoder. Implementation results are presented in Section IV. Section V propose a method to increase the throughput of synchronous decoders in an energy efficient manner by using them concatenated with combinational decoders, forming hybrid polar decoders. Section VI concludes the paper.

II. BACKGROUND

A. Problem Set-Up

In this paper, vectors are denoted by bold lowercase letters. The operations on matrices and vectors are defined over $\mathbb{F}_2$, the binary field. The addition over $\mathbb{F}_2$ is represented by the $\oplus$ operator. For any set $S \subseteq \{0, 1, \ldots, N - 1\}$, $S^c$ denotes the set with elements $\{0, 1, \ldots, N - 1\} \setminus S$. For any vector $u = (u_0, u_1, \ldots, u_{N-1})$ of length $N$ and set $S \subseteq \{0, 1, \ldots, N - 1\}$, the vector $u_S$ is defined as $u_S \overset{\text{def}}{=} [u_i : i \in S]$. The logarithmics are to the base-2 unless stated otherwise.

We consider the communication scheme in Fig. 1. Let $N$ be the length of the codewords for the polar channel coding. The uncoded bit vector of length $N$, $u$, is composed of an information part $u_A$ of length $K$, and frozen part $u_{A^c}$ of length $(N - K)$. The index set $A$ is formed according to the calculated reliability parameters as explained in [11]. We use $u_{A^c} = 0$ throughout this paper and assume it is known both at the encoder and decoder parts. We define the frozen bit vector $a$ of length $N$ with elements:

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

The uncoded bit vector $u$ is encoded using a polar encoder. The coded bit vector of length $N$, $x$, is transmitted through a binary-input memoryless channel $W$, and the channel observation vector of length $N$, $y$, is obtained at the receiver. Log-likelihood ratios (LLR’s) are calculated using the channel observations as: $\ell_i = \ln \frac{W(u_i | x=1)}{W(u_i | x=0)}$, $i \in \{0, 1, \ldots, N - 1\}$. The SC polar decoder uses the inputs $\ell$ and $a$ to calculate bit estimates vector $\hat{u}$.

![Fig. 1. Communication scheme with polar coding](image)

B. Polar Encoder

Polar codes belong to the class of linear block codes, for which the encoder can be implemented as vector-matrix multiplication with complexity $O(N^2)$. On the other hand, it is possible to represent a polar encoder with a trellis of $\log N$
stages and \( N \log N \) nodes. This representation enables to implement a polar encoder with complexity \( O(N \log N) \). The trellis representation of polar encoding operation for \( N = 8 \) is given in Fig. 2. The encoding operation begins from the left side of the graph as the uncoded bits are given as inputs to the encoder. The addition operations are performed from left to right until the coded bits are obtained as outputs of the encoder.

The trellis in Fig. 2 shows that polar encoding contains identical series of operations with different input sets after the first addition stage, i.e., a series of operations with inputs \( u_{2i} \oplus u_{2i+1} \) and an identical series of operations with inputs \( u_{2i+1} \), \( i = 0, 1, 2, 3 \). This property allows to perform encoding to a length-\( N \) input recursively by invoking the encoding function two times, each with different length-\( N/2 \) inputs. This recursive algorithm is shown in Fig. 3. The length-\( N/2 \) input for the first invoking, \( u' \), is calculated by the componentwise sum of vectors \( u_c \) and \( u_o \) as seen on line 8 where \( E = \{0, 2, \ldots, N - 2\} \) and \( O = \{1, 3, \ldots, N - 1\} \). The encoding function uses the length-\( N/2 \) vector \( u'' \) as the input for second invoking, which is simply equal to \( u_o \), as seen on line 10.

\begin{algorithm}
1: function \( x = \text{ENCODE}(u) \)
2: \( N = \text{length}(u) \)
3: if \( N = 2 \) then
4: \( x_0 \leftarrow u_0 \oplus u_1 \)
5: \( x_1 \leftarrow u_1 \)
6: return \( x \leftarrow (x_0, x_1) \)
7: else
8: \( u' \leftarrow u_c \oplus u_o \)
9: \( x' \leftarrow \text{ENCODE}(u') \)
10: \( u'' \leftarrow u_o \)
11: \( x'' \leftarrow \text{ENCODE}(u'') \)
12: return \( x \leftarrow (x', x'') \)
13: end if
14: end function
\end{algorithm}

C. SC Polar Decoder

We define the functions
\[
\begin{align*}
  f(x, y) &= \text{sgn}(x) \text{sgn}(y) \min \{|x|, |y|\}, \\
  g(x, y, z) &= y + (1 - 2z) x,
\end{align*}
\]
as the min-sum approximations to the equations (75) and (76) in [1], respectively. The function \( \text{sgn}(x) \) is defined as:
\[
\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases}
\]
The trellis for SC polar decoding is given in Fig. 4. The decoding and encoding trellises are identical, except the inputs (channel observation LLR’s) are taken from the right and the outputs (bit estimates) are provided from the left sides of the trellis in decoding operation. The intermediate LLR’s at each stage are calculated by either \( f(\cdot) \) or \( g(\cdot) \) functions given by [1] and [2], respectively, using the LLR’s at the previous stages. The function labels on each branch represent the function that is used to calculate LLR of the stage to its left using two LLR’s of the stage to its right. As the encoding function, SC decoding function can be implemented recursively. This recursive algorithm is shown in Fig. 5.

The functions \( f_{N/2}(\cdot) \) and \( g_{N/2}(\cdot) \) are multiple-input multiple-output versions of [1] and [2], i.e.,
\[
\begin{align*}
  f_{N/2}(\ell) &= (f(\ell_0, \ell_1), \ldots, f(\ell_{N-2}, \ell_{N-1})), \\
  g_{N/2}(\ell, v) &= (g(\ell_0, \ell_1, v_0), \ldots, g(\ell_{N-2}, \ell_{N-1}, v_{N/2-1})).
\end{align*}
\]
The operator \( d(x, a) \) is used for decision-making with the calculated LLR’s and frozen bits, and defined as:
\[
d(x, a) = \begin{cases} 0, & \text{if } a = 0 \text{ or } x \geq 0 \\ 1, & \text{otherwise} \end{cases}
\]

Similar to the case of recursive encoding function, an input of length-\( N \) can be decoded by invoking the decoding function two times with different length-\( N/2 \) inputs. The input vector for the first invoking, \( \ell' \), is obtained using [3] on the received input vector \( \ell \), as seen on line 8. The output vector \( \hat{u}' \) forms the first half of the estimated bit vector, and is also passed to the encoder function to obtain \( v \), the partial-sum vector.
function \( \hat{u} = \text{DECODE}(\ell, a) \)

1. \( N = \text{length}(\ell) \)
2. if \( N == 2 \) then
3. \( \hat{u}_0 \leftarrow d(\text{sgn}(f_0) \oplus \text{sgn}(f_1), a_0) \)
4. \( \hat{u}_1 \leftarrow d(g(\ell_0, \ell_1, \hat{u}_0), a_1) \)
5. return \( \hat{u} \leftarrow (\hat{u}_0, \hat{u}_1) \)
6. else
7. \( \ell' \leftarrow f_{N/2}(\ell) \)
8. \( a' \leftarrow (a_0, \ldots, a_{N/2-1}) \)
9. \( \hat{u}' \leftarrow \text{DECODE}(\ell', a') \)
10. \( v \leftarrow \text{ENCODER}(\hat{u}') \)
11. \( \ell'' \leftarrow g_{N/2}(\ell, v) \)
12. \( a'' \leftarrow (a_{N/2}, \ldots, a_{N-1}) \)
13. \( \hat{u}'' \leftarrow \text{DECODE}(\ell'', a'') \)
14. return \( \hat{u} \leftarrow (\hat{u}', \hat{u}'') \)
15. end if
16. end function

Fig. 5. Recursive algorithm for SC polar decoder

(line[11]). The input vector for the second invoking of decoding function, \( \ell'' \) is calculated using (5) with received inputs and partial-sums, as seen on line[12]. The decoding of a length-\( N \) input is completed when decoding function is invoked with the calculated \( N/2 \) inputs once again (line[14]).

D. Power Consumption is CMOS Digital Circuits

Power dissipated in CMOS circuits is composed of two parts named static and dynamic powers [(22)]. Static power is dissipated by the transistors even if there is no voltage switching in the circuit. It is due to the leakage currents in transistors and the potential difference in diffusion regions and substrates. The most commonly used method to reduce static power is power gating, which involves cutting the operating voltage to the unused parts of the circuit.

On the other hand, dynamic power is dissipated when the capacitances in the circuit are charged or discharged due to a switching voltage, and can be formulated as:

\[
P_{\text{dynamic}} = \alpha CV_{DD}^2 f,
\]

where \( \alpha \) is called the activity factor and represents the average portion of the circuit that switches with the switching voltage, \( C \) is the internal capacitance of the overall circuit, \( V_{DD} \) is the drain voltage, and \( f \) is the operating frequency. Activity factor depends on the architecture design. Internal capacitance depends on the architecture design and also the process technology.

III. Decoder Implementation Using Combinational Logic

The recursive decoding algorithm explained in Fig. 5 can be implemented using only combinational circuits. The main motivations for such an architecture are high throughput and low power consumption. In this section, we are going to present the details of this purely combinational architecture and analyze its complexity and latency.

A. Hardware Description

1) Combinational Decoder: As explained in Sec. II-D dynamic power consumption can be decreased by reducing the operating clock frequency. On the other hand, reducing the clock frequency in a synchronous digital circuit often results in reduction of its throughput. The aim of this work is to design an SC polar decoder that provides \( N \) decoded bits as output in one clock cycle in order to maintain a high throughput while dissipating low dynamic power. Decoding a single codeword in a single clock cycle requires cascading a large number of basic logic blocks without any registers in-between. The maximum operating frequency of such a circuit will be significantly reduced due to the increased combinational delay with respect to the conventional synchronous decoders. On the other hand, the operating frequency should be kept large enough in order to maintain a high throughput.

The basic logic blocks required for SC polar decoding are comparator and adder/subtractor blocks, required by (1) and (2) respectively. In a purely combinational decoder, we assign a separate logic block to each of the algebraic operations required to decode one codeword. The outputs of the decoder settle according to the provided inputs after an amount of time which is equal to the combinational delay of the overall circuit. The inputs are kept constant during this time interval since no registers are used between the logic blocks.

In order to operate properly, combinational circuits should not contain any loops in the design. Therefore, the operations on lines 10 and 14 of Fig. 5 should be performed by cascading two different \( N/2 \)-input decoders. This architecture is shown in Fig. 6. As the algorithm suggests, these \( N/2 \)-input decoders are implemented using four different \( N/4 \)-input decoders, and so on; until a base decoder for a relatively small \( N \) is reached (e.g., length-4 input decoder).

The lack of registers and RAM blocks between each calculation stage makes the architecture feasible in terms of hardware usage. This also saves time from read/write operations and clock setup/hold times. Sign-magnitude representation is used to eliminate absolute value and sign-changing operations, which are frequent and in large numbers if any other representation is used. The function \( g_{N/2}(\cdot) \) is implemented using the look-ahead method: it is composed of parallel adder and subtractor pairs each of which carry out addition and subtraction for given inputs, and choose the desired output by a multiplexer according to the partial sum input. This reduces the amount of delay required to prepare the inputs of the second decoder after the outputs of the first decoder settle. Comparators are used for implementation of the function \( f(\cdot) \) and for making odd-numbered bit decisions instead of adder/subtractor blocks, i.e., any odd-numbered bit estimate \( \hat{u}_{2i+1} = d[g(\lambda_1, \lambda_2, \hat{u}_{2i}), a_{2i+1}] \) can also be calculated as:

\[
\hat{u}_{2i+1} = \begin{cases} 0 & \text{if } a_{2i+1} = 0, \\ \text{sgn}(\lambda_2) & \text{if } a_{2i+1} = 1 \text{ and } |\lambda_2| \geq |\lambda_1|, \\ \text{sgn}(\lambda_1) \oplus \hat{u}_{2i+1} & \text{otherwise}. \end{cases}
\]
expressions relating the inputs and outputs of the decoder can easily be found as:

\[
\hat{u}_0 = d \left[ \text{sgn}(\ell_0) \oplus \text{sgn}(\ell_1) \oplus \text{sgn}(\ell_2) \oplus \text{sgn}(\ell_3), a_0 \right],
\hat{u}_1 = d \left[ g(f(\ell_0, \ell_1), f(\ell_2, \ell_3), \hat{u}_0), a_1 \right],
\hat{u}_2 = d \left[ \text{sgn} \left( g(\ell_0, \ell_1, \hat{u}_0) \right) \oplus \text{sgn} \left( g(\ell_2, \ell_3, \hat{u}_1) \right), a_2 \right],
\hat{u}_3 = d \left[ g \left( g(\ell_0, \ell_1, \hat{u}_0), g(\ell_2, \ell_3, \hat{u}_1) \right), a_3 \right].
\] (9)

The combinational polar decoder using the equation set (9) is illustrated in Fig. 7. The decoder for any \(N\) value can be implemented as in Fig. 6 using the base decoder in Fig. 7.

The combinational architecture is feasible for SC since it is a non-iterative algorithm and can be implemented without feedback loops. The disadvantage of purely combinational architecture is the increased hardware usage since each operation in decoding is carried out by a unique adder/subtractor or comparator circuit.

2) Pipelined Combinational Decoder: Resource reuse and pipelining are not applicable to the combinational decoder design described in Section III-A1 However, registers can be added before both of the two \(N/2\)-input decoders in the recursive architecture to enable pipelining, as shown in Fig. 8. The parameter \(Q\) in Fig. 8 denotes the quantization bit number for calculations; therefore, size-\(N \times Q\) registers are required to store channel LLR’s of a length-\(N\) codeword, and size-\(N/2 \times 1\) registers are required to store partial-sums. The channel observation LLRs \(\ell_1\), \(\ell_2\), and \(\ell_3\) correspond to different codewords, and \(v_1\) is the vector of partial-sums for the codeword \(\ell_1\). The output \(\hat{u}_0'\) is the vector of the first half of estimated bits for the codeword \(\ell_2\) and the output \(\hat{u}_0''\) is the vector of the second half of estimated bits for the codeword \(\ell_1\). Pipelining stages can be increased further to increase throughput by adding registers between the four \(N/4\)-input decoders that form the two \(N/2\)-input decoders, and so on; with the cost of increasing complexity and power.

B. Analysis

In this section, the complexity and delay of combinational decoder will be investigated with the help of the structure in Fig. 5.

1) Complexity: The complexity of the combinational decoder can be expressed in terms of the total number of comparators, adders and subtractors in the design since they are the basic building blocks of the architecture with similar complexities.

Number of comparators: Comparators are used in two different types of operations as explained in Section III-A1 the function \(f(\cdot)\) given in (11) and the decisions for odd-indexed bits as described in (8). We define \(c_N\) as the number of comparators used for the function \(f(\cdot)\) in an \(N\)-input combinational decoder. Using the algorithm in Fig. 5 and \(c_4 = 2\) by Fig. 7 \(c_N\) can be found as:

\[
c_N = 2c_{N/2} + \frac{N}{2}
= 2 \left( 2c_{N/4} + \frac{N}{4} \right) + \frac{N}{2}
= 2 \left( 2 \left( 2c_{N/8} + \frac{N}{8} \right) + \frac{N}{4} \right) + \frac{N}{2}
\]

\[
= \frac{N}{4}c_4 + \log \left( \frac{N}{4} \right) \frac{N}{2}
= \frac{N}{2} \log \left( \frac{N}{2} \right). \tag{10}
\]

Next, we define \(s_N\) as the number of comparators used for making odd-indexed bit decisions in an \(N\)-input combinational decoder. Again, using \(s_4 = 2\) by Fig. 7 we can find:

\[
s_N = 2s_{N/2} = 2(2s_{N/4}) = \ldots = \frac{N}{4}s_4 = \frac{N}{2}. \tag{11}
\]

Number of adders and subtractors: The adders and subtractors are used to implement the function \(g(\cdot)\) given in (2), for which the output is selected via a multiplexer according to the result of the corresponding partial sum. We define \(r_N\) as the total
number of adders and subtractors in an $N$-input combinational decoder. With calculations similar to the ones in (10) and using $r_4 = 4$ by Fig. 8 $r_N$ can be found as $r_N = N \log (N/2)$.

We add the number of comparators, adders, and subtractors used in the design to obtain a complexity measure as:

$$c_N + s_N + r_N = N \left( \frac{3}{2} \log N - 1 \right), \quad (12)$$

making the complexity order $c_N = O(N \log N)$.

2) Combemional Delay: Let $D_N$ denote the combinational delay of an $N$-input decoder. By Fig. 8 we can interpret $D_N$ as

$$D_N = 2D_{N/2} + \delta_c + 2\delta_m + E_{N/2} + \tau_N$$

where $\delta_c$ is the delay of a comparator and $\delta_m$ is the delay of a multiplexer, which are summed to form the delay of function $f()$. The function $g()$ adds another $\delta_m$ delay, since the design uses look-ahead adder/subtractors and addition/subtraction operation occurs at the same time as input LLR comparisons of $f()$ function occur. The maximum delay for the partial-sum calculated by $N/2$-input encoder is denoted by $E_{N/2}$ and can be written as:

$$E_{N/2} = \log \left( \frac{N}{2} \right) \delta_x,$$

with $\delta_x$ denoting the propagational delay of a single exclusive-or gate. It is assumed here that $N/2$ bits are exclusive-ored at $\log (N/2)$ stages. Finally, $\tau_N$ represents the non-deterministic change in the interconnect delay of the $N$-input decoder with respect to the $N/2$-input decoder at the recursion step where $N$-input decoder is obtained from $N/2$-input decoders. We can express $D_N$ in terms of the delays of basic blocks and base decoder as follows:

$$D_N = \frac{N}{4}D_4 + \left( \frac{N}{4} - 1 \right)(\delta_c + 2\delta_m) + \left( \frac{3N}{4} - \log N - 1 \right)\delta_x + T_N,$$

where $T_N$ is the routing delay of the overall circuit. Using Fig. 8 and assuming $\delta_c \geq 2\delta_x + \delta_a$, it can be found that $D_4 = 3\delta_x + 4\delta_m + \delta_x + 2\delta_a$ with $\delta_a$ representing the delay of an and gate. Substituting $D_4$ in (14), we obtain

$$D_N = (N-1)(\delta_c) + \left( \frac{3N}{2} - 1 \right) \delta_m + (N - \log N - 1)\delta_x + \frac{N}{2}\delta_a + T_N.$$ \hspace{1cm} (15)

The interconnect delay of the overall design, $T_N$, cannot be expressed in a deterministic form in terms of $N$ since the routing process does not follow a predictable scheme with increasing hardware usage and design area, especially in FPGA implementations.

IV. IMPLEMENTATION RESULTS

In this section, the implementation results of combinational and pipelined combinational decoders are presented on ASIC and FPGA. The implemented architectures use $Q = 5$ for LLR calculations throughout the decoding operation. The metrics used in performance evaluation are throughput, energy-per-bit, and hardware efficiency. We define the throughput as the rate of decoded bits (frozen and information) per second, and calculate it as:

$$\text{TP} = \frac{N}{D_N}.$$ \hspace{1cm} (16)

The energy-per-bit is calculated by

$$\text{Energy per bit} = \frac{\text{Power}[W]}{\text{Throughput}[b/s]}$$

and the hardware efficiency is

$$\text{Hardware Efficiency}[b/s/m^2] = \frac{\text{Throughput}[b/s]}{\text{Area}[m^2]}.$$ 

A. ASIC Implementation

Table I gives the implementation results for combinational decoders of block lengths from $2^5$ to $2^{15}$ with 90 nm 1.0 V technology. The input LLR’s, frozen bit vector, and output bit estimates of the combinational decoder are stored in registers in order to reduce the input/output pins of the design. The results in Table I verify the theoretical analyses for complexity and delay. It is expected from (12) that the ratio of the number of components of an $N$-input decoder to an $N/2$-input decoder should converge to 2 as $N$ increases. The area usage and number of cells given in Table I confirm this trend. The throughput of the decoders slightly decrease as $N$ increases, which is due to the additive delays on top of $2D_{N/2}$ in (15) and throughput equation (16).

It is seen from Table I that the power consumption tends to saturate as $N$ increases. To study this behaviour, a more detailed power consumption analysis is given in Table II. Table II shows that the static power consumption of the combinational decoder approximately doubles as $N$ increases. This is expected since the static power consumption is directly related to the number of cells in the design, which also approximately doubles as given in Table I. On the other hand, the dynamic power consumption (which is the significant part...
It should be noted that the design results for the combinational decoder are obtained using 90 nm technology, whereas the results provided by [12] and [23] are for 65 nm technology. This creates a disadvantage for combinational decoder in all of the studied metrics. The area spanned by combinational decoder is seen to be the highest from Table III. However, hardware efficiency is also high because of the increased throughput. Two sets of throughput and power consumption metrics are presented for [23]: one with higher power and throughput, and one with lower ones. The energy required for combinational decoder to decode one bit is smaller than the one for [23], even when it is operated in the lower power state. The throughput of the combinational decoder is only smaller than the throughput of [23] when it is operated in the higher power state.

B. FPGA Implementation

Table IV shows the implementation results of purely combinational decoder on Xilinx Virtex6-XC6VLX550T (40 nm) FPGA core. The FPGA implementation strategy is adjusted to favor speed performance of the design.

### Table I
**ASIC Implementation Results**

| N  | 2^5 | 2^10 | 2^14 | 2^18 | 2^9 | 2^10 |
|----|-----|------|------|------|-----|------|
| Technology | 90 nm, 1.0 V |
| Area [mm^2] | 0.061 | 0.153 | 0.338 | 0.759 | 1.514 | 2.213 |
| Power [mW] | 7.07 | 18.05 | 26.90 | 32.25 | 36.02 | 32.75 |
| Maximum Delay [ns] | 9.56 | 19.31 | 39.80 | 81.03 | 175.02 | 358.00 |
| Frequency [MHz] | 101.4 | 51.7 | 25.1 | 12.3 | 5.71 | 2.79 |
| Throughput [Gb/s] | 3.34 | 3.31 | 3.21 | 3.16 | 2.92 | 2.86 |
| Hard. Eff. [Mb/s/mm^2] | 54754 | 21633 | 94579 | 4163 | 1928 | 890 |
| Engy.-per-bit [pJ/b] | 2.11 | 5.45 | 8.38 | 10.20 | 12.33 | 11.45 |

### Table II
**Power Consumption**

| N  | 2^5 | 2^10 | 2^14 | 2^18 | 2^9 | 2^10 |
|----|-----|------|------|------|-----|------|
| Comb. Decoder | Stat. [mW] | 248.5 | 701.8 | 1572.8 | 3505.0 | 6896.8 | 14846.7 |
| Stat. [mW] | 701.8 | 1572.8 | 3505.0 | 6896.8 | 14846.7 |
| Dyn. [mW] | 153.7 | 22.3 | 24.6 | 23.8 | 23.4 |
| Throughput [Gb/s] | 57 | 26.9 | 32.25 | 36.02 | 32.75 |
| Hard. Eff. [Mb/s/mm^2] | 54754 | 21633 | 94579 | 4163 | 1928 | 890 |
| Engy.-per-bit [pJ/b] | 2.11 | 5.45 | 8.38 | 10.20 | 12.33 | 11.45 |

### Table III
**Comparison with State-of-the-Art Polar Decoders**

| This Work | [12] | [23] |
|-----------|------|------|
| Decoder Type | SC | SC | BP |
| Block Length | 1024 | 1024 | 1024 |
| Technology | 90 nm | 65 nm | 65 nm |
| Area [mm^2] | 3.213 | 0.68 | 1.476 |
| Voltage [V] | 1.0 | 1.2 | 1.0 | 0.425 |
| Frequency [MHz] | 2.79 | 1010 | 300 | 50 |
| Power [mW] | 32.75 | - | 477.5 | 18.6 |
| Throughput [Mb/s] | 2860 | 397 | 476.6 | 179.3 |
| Engy.-per-bit [pJ/b] | 11.45 | - | 10.21 | 23.8 |
| Hard. Eff. [Mb/s/mm^2] | 890 | 7.30* | 3168 | 528 |

* Not given in the original work, calculated with the provided metrics of total power consumption) stabilizes as N increases. This is explained as follows: the dynamic power is directly related to the load capacitance, activity factor, and operating frequency of the circuit, as expressed in (7). As N increases, the number of gates are doubled asymptotically while the operating frequency is approximately halved as a result of the recursive architecture. Activity factor is expected to stay constant since the switching capacitance ratio to the overall capacitance is the same for all N. Therefore, the product of these values saturate as N increases. This also causes a saturation in the dynamic power consumption. The results show that a very low power is required to operate the decoders with throughputs on the order of Gb/s and the energy consumption is reduced to the order of pJ/b.

Table III shows the comparison of combinational decoder with two state-of-the-art polar decoders proposed in [12] and [23]. It should be noted that the results of BP decoder in [23] are given for an SNR value of 4 dB, for which BP decoder makes 6.57 iterations per codeword. However, the required number of iterations for BP increases for smaller SNR values, which will in turn decrease throughput and hardware efficiency and increase the decoding energy. On the other hand, SC decoders perform with the same metrics for all SNR values. It should be noted that the design results for the combinational decoder are obtained using 90 nm technology, whereas the results provided by [12] and [23] are for 65 nm technology. This creates a disadvantage for combinational decoder in all of the studied metrics. The area spanned by combinational decoder is seen to be the highest from Table III. However, hardware efficiency is also high because of the increased throughput. Two sets of throughput and power consumption metrics are presented for [23]: one with higher power and throughput, and one with lower ones. The energy required for combinational decoder to decode one bit is smaller than the one for [23], even when it is operated in the lower power state. The throughput of the combinational decoder is only smaller than the throughput of [23] when it is operated in the higher power state.

### Table IV
**FPGA Implementation Results (Purely Combinational)**

| N  | TP [Gb/s] | LUT | FF |
|----|-----------|-----|----|
| 2^5 | 1.05 | 1479 | 169 |
| 2^6 | 0.88 | 1918 | 206 |
| 2^7 | 0.85 | 5126 | 392 |
| 2^8 | 0.82 | 14517 | 783 |
| 2^9 | 0.75 | 35152 | 1564 |
| 2^10 | 0.73 | 77154 | 3909 |
| 2^11 | 0.60 | 193456 | 6151 |

The registers seen in Table IV are used for input/output and frozen bit vector storage, and small logic circuits and are not used in the main combinational decoder structure. The decrease in throughput of the FPGA design with respect to that of ASIC is mainly due to the high routing delays in FPGA, which may go up to 90% of the total delay.
is established using LUT’s, which are also used to implement the logic of the design. Therefore, mapping a large connected combinational block results in high routing delays in FPGA’s. Table [V] shows the implementation results of the pipelined architecture given in Fig. [VII] which is more suitable for FPGA implementation since registers are used between large combinational circuit blocks. The throughput gains with respect to purely combinational architecture are also given in Table [V]. The results in Table [V] show that the throughput is over 1 Gb/s for practical values of blocklength. The pipelining stages can be increased further to increase the throughput at the cost of complexity.

V. HYBRID LOGIC POLAR DECODERS

The number of calculations that can be done in parallel decreases each time SC decoder gets close to decision-making stages, i.e., leftmost stages of \( D \). This reduces hardware utilization and throughput of synchronous SC decoders. In hybrid logic polar decoders, a combinational decoder is concatenated with a synchronous decoder to do the calculations at these stages with smaller delays and increase throughput in an energy-efficient manner.

The concatenation is done by using an \( N \)-input synchronous decoder to calculate the LLR’s up to the stage where there are \( N’ \) operations that can be processed in parallel, and then pass these LLR’s to an \( N’ \)-input combinational decoder. The \( N’ \)-input combinational decoder outputs \( N’ \) bit estimates, which are also used to calculate the partial sums for further operations of the synchronous decoder. This operation is repeated \( N/N’ \) times until \( N \) bit estimates are obtained by the hybrid logic decoder. Fig. [VIII] shows the algorithm of a hybrid logic decoder. For the \( i^{th} \) run of synchronous and combinational decoders, where \( 0 \leq i \leq N/N’ - 1 \): the LLR vector passed from synchronous decoder to combinational decoder, output vector, and frozen bit vector are denoted by \( \lambda^{(i)} = (\lambda^{(i)}_0, \ldots, \lambda^{(i)}_{N’-1}) \), \( \hat{u}^{(i)} = (\hat{u}^{(i)}_{N’}, \ldots, \hat{u}^{(i+1)}_{N’-1}) \), and \( \alpha^{(i)} = (\alpha^{(i)}_{N’}, \ldots, \alpha^{(i+1)}_{N’-1}) \), respectively.

During the operation of combinational decoder, the synchronous decoder waits for \( \left[ \frac{\Lambda_{\alpha}}{t_{\alpha}} \right] \) clock cycles, where \( t_{\alpha} \) is the operating clock period of sequential decoder. The latency gain of a hybrid logic decoder to that of the synchronous decoder can be calculated as follows: let the latency of an \( N’ \)-input synchronous decoder be \( L_S(N) \). The latency reduction obtained by using an \( N’ \)-input combinational decoder instead of synchronous decoder is \( L_{\text{red}}(N’) = L_S(N) - \left[ \frac{\Lambda_{\alpha}}{t_{\alpha}} \right] \) for one use of combinational decoder. Then, the latency gain of a hybrid logic decoder with respect to the synchronous decoder is calculated as follows:

\[
g = \frac{L_S(N)}{L_S(N) - (N/N’) L_{\text{red}}(N’)}
\]

As an example, we consider the semi-parallel SC decoder proposed in [10] with latency:

\[
L_{SP}(N) = 2N + \frac{N}{P} \log\left(\frac{N}{4P}\right)
\]

where \( P \) is the number of processing elements in the decoder. The implementation results in [10] are given for a Stratix IV FPGA with similar properties to the Virtex6 FPGA used in this work. Combining (17) and (18), and using the results in Table [VI] we obtain the gains and throughput values of SP and Hybrid Logic SP decoders for different \( N \) and \( P \) values as in Table [VI].

| \( N \) | \( P \) | \( TP_{SP} \) | \( N’ \) | \( g \) | \( TP_{HLSP} \) |
|---|---|---|---|---|---|
| \( 2^{10} \) | 64 | 85 | \( 2^{5} \) | 5.90 | 501.50 |
| \( 2^{10} \) | 64 | 85 | \( 2^{5} \) | 6.50 | 552.50 |
| \( 2^{10} \) | 64 | 85 | \( 2^{5} \) | 7.22 | 613.70 |
| \( 2^{11} \) | 64 | 83 | \( 2^{6} \) | 8.70 | 473.10 |
| \( 2^{11} \) | 64 | 83 | \( 2^{6} \) | 9.60 | 572.70 |
| \( 2^{11} \) | 64 | 83 | \( 2^{6} \) | 9.60 | 572.70 |

It should be noted that the gains and throughput values given in Table [VI] are obtained by theoretical calculations and may differ depending on the implementation and partial-sum update logic ([13], [20]). Moreover, the gain will be smaller for decoders that spend less clock cycles in the final stages of decoding operation, such as [13] and [21]. However, the hybrid logic decoder is useful for decoding large codeword lengths, for which the hardware usage is high for the combinational polar decoder and latency is high for the synchronous polar decoders.

VI. CONCLUSION

In this work, we proposed a purely combinational architecture for SC polar decoders, which aim to provide a
high throughput with low power consumption. The proposed combinational decoder outputs a codeword in a single clock cycle and operates at low clock frequencies, which reduces the dynamic power consumption of the architecture. We show that combinational decoder provides throughputs on the order of Gb/s with an energy consumption on the order of pJ/b. The decoder is shown to be hardware efficient even though the hardware usage is high. A pipelined combinational decoder and hybrid logic decoders are also presented as possible variations of combinational decoder.

REFERENCES

[1] E. Arikan, “Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” IEEE Trans. on Inf. Theory, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
[2] R. Bharath Kumar and N. Chandrachoodan, “A GPU implementation of belief propagation decoder for polar codes,” in Signals, Systems and Computers (ASC), 2014 Conference Record of the Forty Sixth Asilomar Conference on, Nov 2012, pp. 1272–1276.
[3] A. Pamuk, “An FPGA implementation architecture for decoding of polar codes,” 8th Int. Symp. Wireless Commun. Syst., Aachen, pp. 437–441, Nov. 2011.
[4] B. Yuan and K. Parhi, “Architecture optimizations for BP polar decoders,” Proc. of IEEE Int. Conf. Acoust., Speech Signal Process., pp. 2654–2658, May 2013.
[5] ———, “Architectures for polar BP decoders using folding,” in Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, June 2014, pp. 205–208.
[6] A. Balatsoukas-Stimming, A. Raymond, W. Gross, and A. Burg, “Hardware architecture for list successive cancellation decoding of polar codes,” Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 61, no. 8, pp. 609–613, Aug 2014.
[7] J. Lin and Z. Yan, “Efficient list decoder architecture for polar codes,” in Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, June 2014, pp. 1022–1025.
[8] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, “LLR-based successive cancellation list decoding of polar codes,” in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, May 2014, pp. 3903–3907.
[9] B. Yuan and K. Parhi, “Low-latency successive-cancellation list decoders for polar codes with multibit decision,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. PP, no. 99, pp. 1–1, 2014.
[10] C. Leroux, A. Raymond, G. Sarkis, and W. Gross, “A semi-parallel successive-cancellation decoder for polar codes,” Signal Processing, IEEE Transactions on, vol. 61, no. 2, pp. 289–299, Jan 2013.
[11] A. Raymond and W. Gross, “A scalable successive-cancellation decoder for polar codes,” Signal Processing, IEEE Transactions on, vol. 62, no. 20, pp. 5339–5347, Oct 2014.
[12] Y. Fan and C.-Y. Tsui, “An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation,” Signal Processing, IEEE Transactions on, vol. 62, no. 12, pp. 3165–3179, June 2014.
[13] A. Pamuk and E. Arikan, “A two phase successive cancellation decoder architecture for polar codes,” in Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on, July 2013, pp. 957–961.
[14] C. Zhang, B. Yuan, and K. K. Parhi, “Reduced-latency SC polar decoder architectures,” CoRR, vol. abs/1111.0704, 2011. [Online]. Available: http://arxiv.org/abs/1111.0704
[15] C. Zhang, “Low-latency low-complexity channel decoder architectures for modern communication systems,” Univ. of Minnesota, 2012.
[16] C. Zhang and K. Parhi, “Interleaved successive cancellation polar decoders,” in Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, June 2014, pp. 401–404.
[17] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. Gross, “Fast polar decoders: Algorithm and implementation,” Selected Areas in Communications, IEEE Journal on, vol. 32, no. 5, pp. 946–957, May 2014.
[18] A. Alamdar-Yazdi and F. Kschischang, “A simplified successive-cancellation decoder for polar codes,” IEEE Commun. Letters, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.
[19] G. Sarkis and W. Gross, “Increasing the throughput of polar decoders,” IEEE Commun. Letters, vol. 17, no. 4, pp. 725–728, Apr. 2013.
[20] C. Zhang and K. Parhi, “Low-latency sequential and overlapped architectures for successive cancellation polar decoder,” Signal Processing, IEEE Transactions on, vol. 61, no. 10, pp. 2429–2441, May 2013.

[21] B. Yuan and K. Parhi, “Low-latency successive-cancellation polar decoder architectures using 2-bit decoding,” Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 61, no. 4, pp. 1241–1254, April 2014.
[22] N. Weste and D. Harris, Integrated Circuit Design. Pearson, 2011.
[23] Y. S. Park, Y. Tao, S. Sun, and Z. Zhang, “A 4.68gb/s belief propagation polar decoder with bit-splitting register file,” in VLSI Circuits, 2014 Symposium Digest of Technical Papers, 2014 Symposium on, June 2014, pp. 1–2.