A Two Phase Successive Cancellation Decoder Architecture for Polar Codes

Alptekin Pamuk
Department of Electrical-Electronics Engineering
Bilkent University
Ankara, TR-06800, Turkey
alptekin@ee.bilkent.edu.tr

Erdal Arıkan
Department of Electrical-Electronics Engineering
Bilkent University
Ankara, TR-06800, Turkey
arikan@ee.bilkent.edu.tr

***Abstract*—We propose a two-phase successive cancellation (TPSC) decoder architecture for polar codes that exploits the array-code property of polar codes by breaking the decoding of a length-N polar code into a series of length-$\sqrt{N}$ decoding cycles. Each decoding cycle consists of two phases: a first phase for decoding along the columns and a second phase for decoding along the rows of the code array. The reduced decoder size makes it more affordable to implement the core decoder logic using distributed memory elements consisting of flip-flops (FFs), as opposed to slower random access memory (RAM), leading to a speed up in clock frequency. To minimize the circuit complexity, a single decoder unit is used in both phases with minor modifications. The re-use of the same decoder module makes it necessary to recall certain internal decoder state variables between decoding cycles. Instead of storing the decoder state variables in RAM, the decoder discards them and calculates them again when needed. Overall, the decoder has $O(\sqrt{N})$ circuit complexity excluding RAM, and a latency of approximately $2.5N$. A RAM of size $O(N)$ is needed for storing the channel log-likelihood variables and the decoder decision variables. As an example of the proposed method, a length $N = 2^{14}$ bit polar code is implemented in an FPGA and the synthesis results are compared with a previously reported FPGA implementation. The results show that the proposed architecture has lower complexity, lower memory utilization with higher throughput, and a clock frequency that is less sensitive to code length.**

***Index Terms*—Error correcting codes, polar codes, successive cancellation decoding, decoding complexity.**

I. Introduction

Polar codes were introduced in [1] as a class of codes that achieve the capacity of binary-input memoryless symmetric channels using low-complexity encoders and decoders. The decoder used in [1] was a successive cancellation (SC) decoder. Some implementation aspects of the SC decoder were discussed in an early follow-up work [2]. Since then the SC decoder and many of its variants (including belief propagation (BP) decoders) have been the subject of intense research, aimed at improving the performance of the basic SC decoder. This line of work was motivated by potential practical applications of polar coding and has emphasized efficient hardware or software implementations. A notable work of this type is [3], in which a VLSI implementation architecture was given for the SC decoder. In related work, [4], a semi-parallel SC decoder implementation was described, with synthesis results for an FPGA and a TSMC 65 nm process. In [5], first results concerning an FPGA implementation of a BP decoder for polar codes was reported and the complexity of the resulting implementation was compared with that of a decoder for the IEEE 802.16e Convolutional Turbo Code (CTC) code, also implemented on the same FPGA. That comparison showed a complexity advantage in favor of polar codes.

In this work, we describe a new architecture for the implementation of SC decoding. The proposed TPSC decoder architecture exploits the fact that polar codes can be expressed as product codes. As a result, the decoding of an N-bit polar code can be divided into two phases where each phase a shorter polar code is decoded. This approach gives rise to two advantages. First, a smaller partial sum update logic (PSUL) is used. The term PSUL, borrowed from [4], refers to the propagation of decoder decisions to parts of the decoder circuit where they are needed to enable further calculations. The PSUL is indicated as the main cause of hardware complexity and low clock frequency in [4]. The second advantage of using smaller decoder units is to make it more affordable to use FFs as storage elements integrated into the decoder fabric, instead of the more abundant but slower RAM. Further details about the decoder and its relation to previous work will be given in the following sections.

The organization of the rest of the paper is as follows. Section II gives a brief account of polar codes. Details of the TPSC decoder are given in Section III with references to earlier related work. Finally, synthesis results for the TPSC decoder are given in Section IV and compared with an earlier work.

II. Polar Codes

A. Notation

The codes considered are over the binary field $\mathbb{F}_2$ and so are all vector and matrix operations. Boldface uppercase (lowercase) letters are used to denote matrices (vectors). For any matrix $\mathbf{A}$, $\mathbf{A}^{\otimes n}$ denotes the nth Kronecker power of $\mathbf{A}$. For any vector $\mathbf{u} = (u_1, \ldots, u_N)$ and set $\mathcal{A} \subset \{1, \ldots, N\}$, the notation $\mathbf{u}_\mathcal{A}$ denotes the sub-vector of $\mathbf{u}$ consisting of coordinates in $\mathcal{A}$, *i.e.*, $\mathbf{u}_\mathcal{A} = (u_i : i \in \mathcal{A})$. The function $\sigma(x)$ is defined as $\sigma(x) = 0$ if $x \geq 0$ and $\sigma(x) = 1$ otherwise.

978-1-4799-0446-4/13/$31.00 ©2013 IEEE

B. Polar Encoding

For any $N = 2^n$ with $n \geq 1$, a length-N polar code is defined by the linear mapping

$$\mathbf{x} = \mathbf{u}\mathbf{G}_N, \quad \mathbf{G}_N = \mathbf{F}^{\otimes n}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{u}$ and $\mathbf{x}$ are row vectors of size $1 \times N$, representing the source word and the codeword, respectively. A rate K/N polar code is specified by a K-element set $\mathcal{A} \subset \{1, \ldots, N\}$ which serves to split the source vector $\mathbf{u}$ into two parts: a part $\mathbf{u}_{\mathcal{A}}$ which carries data and its complement $\mathbf{u}_{\mathcal{A}^c}$ which is frozen. The decoder knows the frozen part and tries to estimate the free part. We assume throughout that the frozen part $\mathbf{u}_{\mathcal{A}^c}$ is fixed as zero. For capacity-achieving performance on a given channel, the set $\mathcal{A}$ needs to be chosen with care, as described in [1]; however, for the purposes of the present paper, the set $\mathcal{A}$ can be anything.

C. Successive Cancellation Decoding

We consider a decoder architecture which is based on the uniform graphical representation of polar codes as described in [2], [5]. Specifically, we use the representation shown in Fig. 1, which is one of several such representations given in [5]. The decoding of polar codes will be described in relation

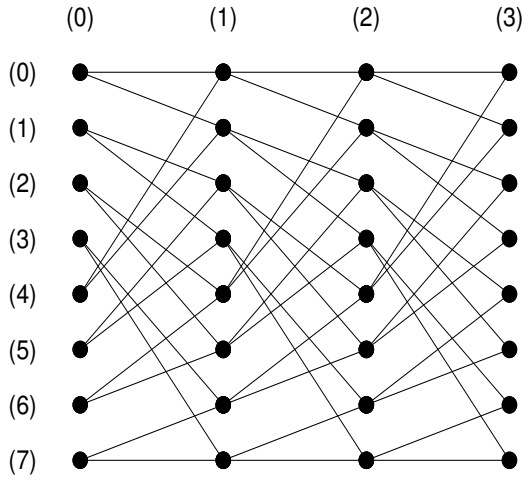

Fig. 1. Uniform decoding graph for an 8-bit SC decoder.

to this graph. For a polar code of length $N = 2^n$, there are N rows and $n+1$ columns in the associated graph. The left-most column (numbered 0) corresponds to the source level and the right-most column (numbered n) to the channel level. For each $0 \leq i \leq N-1$ and $0 \leq j \leq n$, the node in the ith row and the jth column is associated with two decoder variables: a likelihood ratio (LLR) $\lambda_{i,j}$ and a hard decision (HD) $\hat{u}_{i,j}$. The right-most LLR variables $(\lambda_{i,n} : i \in \{0, \ldots, N-1\})$ are received from the channel and constitute the decoder input. The remaining LLR values are calculated by the formulas

$$\lambda_{i,j} = \begin{cases} f(\lambda_{2i,j+1}, \lambda_{2i+1,j+1}), & i < N/2; \\ g(\lambda_{2i,j+1}, \lambda_{2i+1,j+1}, \hat{u}_{i-N/2,j}), & i \geq N/2, \end{cases}$$

where

$$\begin{aligned} f(a,b) &= (1 - \sigma(ab)) \min(|a|, |b|) \\ g(a,b,\hat{u}) &= b + (1 - 2\hat{u})a \end{aligned}$$

(The function f is one of several possible approximations to the exact LLR calculation. The method described here can be applied with other approximations or the exact formula.)

The HD variables are calculated successively in accordance with the following rules.

$$\hat{u}_{i,j} = \begin{cases} 0, & j = 0 \text{ and } i \in \mathcal{A}^c; \\ \sigma(\lambda_{i,j}), & j = 0 \text{ and } i \in \mathcal{A}; \\ \hat{u}_{i/2,j-1} \oplus \hat{u}_{i/2+N/2,j-1}, & j \neq 0 \text{ and } i \text{ even}; \\ \hat{u}_{(i-1)/2+N/2,j-1}, & j \neq 0 \text{ and } i \text{ odd}. \end{cases}$$

The specific order of calculations in SC decoding as described in [1] ensures that the interdependencies among the LLR and HD variables do not lead to a computational lock-up state. In fact, a certain degree of freedom exists in the schedule of calculations as mentioned in [1]. Specifically, the LLR values $\{\lambda_{i,j} : 0 \leq i \leq N-1\}$ at level j can be calculated in batches of size 2^j, for any $0 \leq j \leq n$. Such parallelization has been exploited in [3] and [4] to give a range of implementation options, offering trade-offs between time and hardware complexity.

III. A Two-Phase Successive Cancellation Decoder

In this section, we describe the TPSC decoder architecture for polar codes. This architecture exploits the fact that polar codes can be factored into the product of smaller polar codes. We first make this notion more precise before describing the details of the proposed decoder.

A. Polar Codes as Array Codes

An N-bit polar code can be constructed as a code that maps a source array of size $N_1 \times N_2$ to a codeword array of the same size for any N_1 and N_2 such that $N = N_1 N_2$. To see this, write the source vector $\mathbf{u}$ in (1) in the form

$$\mathbf{u} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{N_1-1} \\ u_{N_1} & u_{N_1+1} & \cdots & u_{2N_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{(N_2-1)N_1} & u_{(N_2-1)N_1+1} & \cdots & u_{N_2N_1-1} \end{bmatrix}.$$

Encode this array row by row using the matrix G_{N_1} to obtain an interim array

$$\mathbf{v} = \begin{bmatrix} v_0 & v_1 & \cdots & v_{N_1-1} \\ v_{N_1} & v_{N_1+1} & \cdots & v_{2N_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{(N_2-1)N_1} & v_{(N_2-1)N_1+1} & \cdots & v_{N_2N_1-1} \end{bmatrix}.$$

Next, encode $\mathbf{v}$ column by column using G_{N_2} to obtain

$$\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N_1-1} \\ x_{N_1} & x_{N_1+1} & \cdots & x_{2N_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(N_2-1)N_1} & x_{(N_2-1)N_1+1} & \cdots & x_{N_2N_1-1} \end{bmatrix}.$$

It is not difficult to see that the array $\mathbf{x}$, serialized into a vector, satisfies (1).

B. The Two Phase Successive Cancellation Decoding Algorithm

The TPSC decoder exploits the above structure by splitting the decoding into two phases, first along the columns then along the rows. The TPSC algorithm is most readily applicable to polar codes for which the code length $N = 2^n$ is a power of 4. Then, one considers the product-form representation with $N_1 = N_2 = \sqrt{N}$ and defines $\sqrt{N}$ decoding cycles (DCs). Each DC consists of a phase-1 (P1) decoding cycle, which works column-wise on the code array, followed by a phase-2 (P2) decoding cycle, which works row-wise. Every DC terminates with the estimation of $\sqrt{N}$ source bits. Fig. 2 illustrates the four DCs in decoding a code of length $N = 16$. The active edges processed by the P1 and P2 decoders in each DC are indicated by the blue and red colors, respectively.

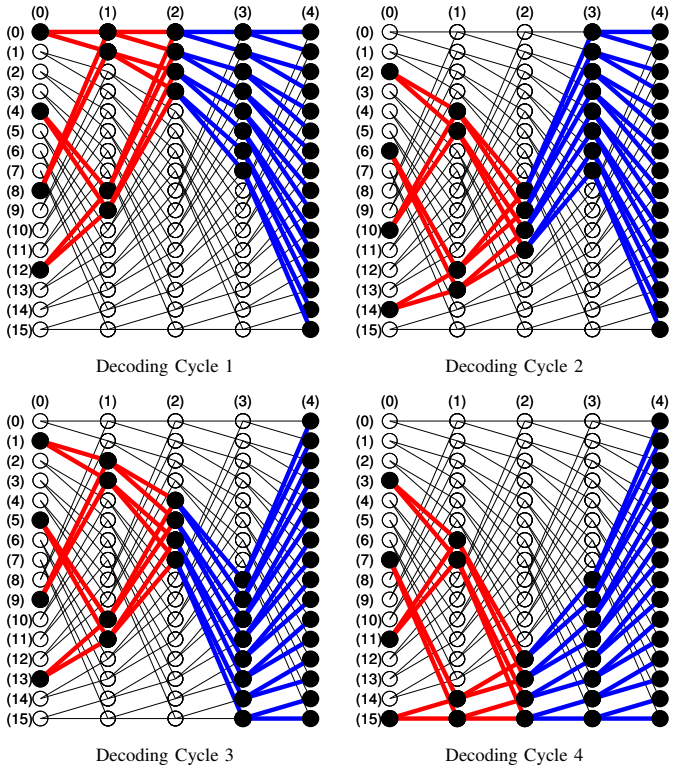

Fig. 2. Active edges in decoder graph in various DCs for a 16-bit code.

In general, the decoding graph consists of $n+1$ levels for a code of length $N = 2^n$. The P1 decoder works on code segments between levels $n/2$ and n, while the P2 decoder works between levels 0 and $n/2$. The two decoders interface at level $n/2$ and exchange information with each other but they are otherwise independent. There are two types of memory used by the decoders: flip-flops (FFs) and random access memory (RAM). FFs are faster than RAMs but the FF storage capacity is nowhere as abundant as the RAM capacity in typical FPGAs. In the proposed TPSC decoder, FFs are used for calculations internal to P1 and P2 decoders. RAM is used for storing the channel LLRs and the HDs exchanged at level $n/2$ between the P1 and P2 decoders. The details of the P1 and P2 decoders are described next, starting with the P2 decoder since any standard polar decoder can be used as a P2 decoder, while the proposed P1 decoder has some novel features.

1) Phase-2 Decoder: The P2 decoder receives LLR inputs at level $n/2$ from the P1 decoder and terminates by generating $\sqrt{N}$ HDs at level 0. Here, we use a fully parallel decoder for P1. To be more specific, we use the pipelined tree (PT) decoder architecture proposed in [3], with some modifications as shown in Fig. 3 for $\sqrt{N} = 8$. As in the original PT decoder, the P1 decoder here has 2^j processing elements (PEs) between levels j and $(j+1)$, where each PE is capable of computing the functions f and g.

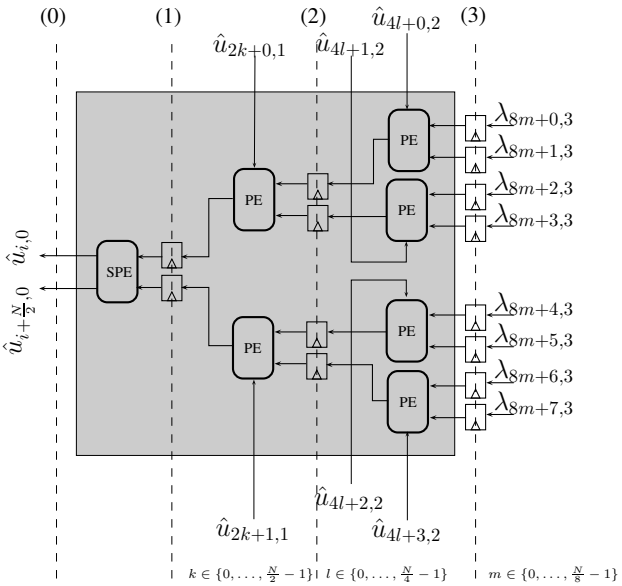

Fig. 3. P2 decoder architecture for $\sqrt{N} = 8$.

The modified PT architecture used here substitutes a special PE (SPE) in place of a regular PE in order to improve latency. The SPE calculates the HDs,

$$\hat{u}_{i,0} = \begin{cases} 0, & i \in \mathcal{A}^c; \\ \sigma(\lambda_{2i,1}) \oplus \sigma(\lambda_{2i+1,1}), & i \in \mathcal{A}, \end{cases}$$

and

$$\hat{u}_{i+\frac{N}{2},0} = \begin{cases} 0, & i+\frac{N}{2} \in \mathcal{A}^c; \\ \sigma(\lambda_{2i+1,1} + (1-2\hat{u}_{i,0})\lambda_{2i,1}), & i+\frac{N}{2} \in \mathcal{A}. \end{cases}$$

in parallel, reducing the latency of the original PT decoder from $2\sqrt{N} - 2$ to $1.5\sqrt{N} - 2$ CCs. The SPE also avoids

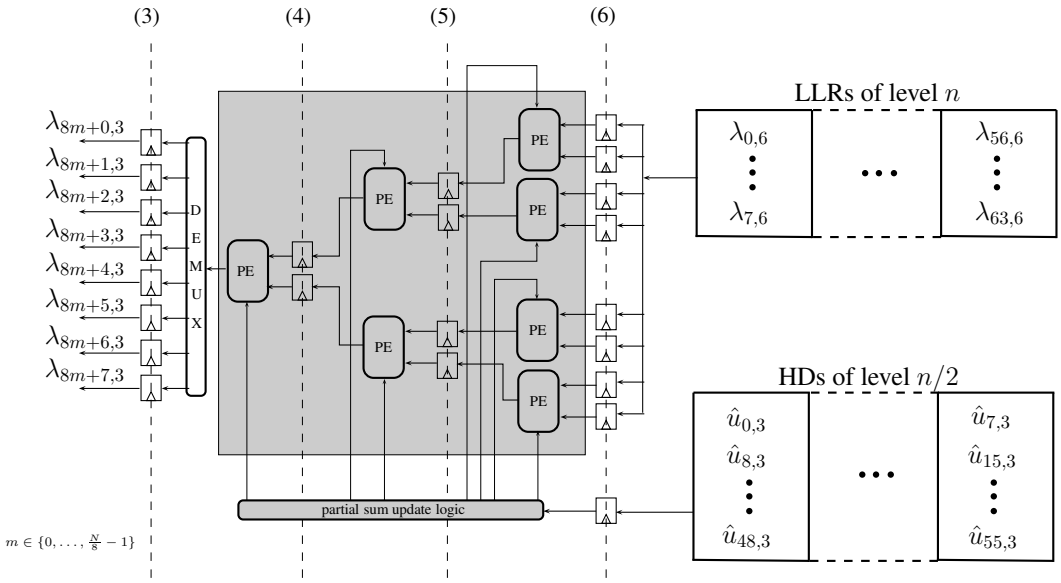

Fig. 4. P1 decoder architecture for $\sqrt{N} = 8$.

calculating the LLRs $\lambda_{i,0}$ at level 0 since only their signs are needed. The short-cut used in SPE is similar to the look-ahead technique proposed in [7], [8]. Finally, the PSUL, which concerns propagating the HDs to the right in the decoder graph, is implemented in a manner similar to [4], except here the PSUL uses a two-bit input since the SPE produces two HDs in one CC.

The P2 decoder stores the LLRs and HDs in FFs (as opposed to RAM) in order to improve the clock speed. At the end of the decoding cycle, the P2 decoder hands over the HDs to the P1 decoder through a RAM while it discards the LLRs. RAM is also used by the P2 decoder to keep track of the identity of the frozen source bits. As pointed out before, FFs are faster but relatively scarce compared to RAM; so by breaking the decoding of a length-N polar code into the decoding of length-$\sqrt{N}$ polar codes, the decoding architecture proposed here makes FF storage more affordable.

Finally, we note that the P2 decoder employs $(\sqrt{N}-2)$ PEs and one SPE, for a total hardware complexity of $O(\sqrt{N})$.

2) Phase-1 Decoder : For P1 decoder, we aim for an architecture that has the same order of complexity as the P2 decoder in terms of hardware and latency. We achieve this by proposing a P1 decoder that is very similar to the P2 decoder. The proposed P1 decoder is shown in Fig. 4 for $\sqrt{N} = 8$.

All PEs in the P1 decoder are identical, unlike the P2 decoder that has an SPE at level 0. The P1 decoder uses RAM to store the channel LLRs and the HDs received from the P2 decoder; it uses FFs to store the LLRs and HDs that it computes internally. The inputs to the PSUL are read from RAM and all partial sums are calculated in one CC. At the end of each P1 decoder cycle, the LLRs at level $n/2$ become inputs to the P2 decoder that takes over.

In this architecture, P1 and P2 decoders are mostly identical and the same hardware with minor adjustments can serve for both tasks. No provision is made for the P1 decoder to save its LLRs at the end of the decoding cycle, other than passing those LLRs at level $n/2$ to the P2 decoder. This necessitates recalculation of the discarded LLRs when they are needed again in the future. The alternative to discarding the LLRs would be to save them in RAM, which might reduce the time complexity if data can be transferred fast enough between the FFs and the RAM.

As for the latency of the P1 decoder, note that it takes $\frac{n}{2}+1$ CCs for the first LLR to appear at the output (level $n/2$). The remaining $\sqrt{N}-1$ LLRs at level $n/2$ are calculated in the next $\sqrt{N}-1$ CCs. Therefore, the latency of the first phase is $\sqrt{N}+\frac{n}{2}$.

The P1 decoder employs $(\sqrt{N}-1)$ PEs and has a total hardware complexity of $O(\sqrt{N})$.

C. Overall Latency and Complexity

The latency of one DC is the sum of the latencies of P1 and P2 decoders, which is $2.5\sqrt{N}+\frac{n}{2}-2$ CCs. Since there are $\sqrt{N}$ DCs, the total latency is $2.5N+\sqrt{N}(\frac{n}{2}-2)$ which is approximately $2.5N$ for large N.

The overall circuit complexity of the TPSC decoder equals the complexity of the P1/P2 decoder, which is $O(\sqrt{N})$, plus the complexity of the RAM units, which is $O(N)$. The sub-linear complexity $O(\sqrt{N})$ relates to the most expensive logic elements, such as FFs and look-up tables (LUTs).

D. Heuristics to Improve Latency

In each decoding cycle the P2 decoder encounters a polar code of some rate which varies between 0 and 1. When the P2 decoder encounters a code of rate 0 or 1, a short-cut in decoding can be introduced as in [6]. If the code rate is 0, the HDs in that cycle can be pre-computed. If the code rate is 1, one can simply turn the LLRs at P2 decoder input to HDs.

The additional hardware required for taking advantage of these short-cuts is insignificant compared to the overall complexity.

An empirical study on the number of occurrences of rate 0 and 1 codes at the input of the P2 decoder are given in Table I. Each polar code in the table is constructed using the formulas for a binary erasure channel with erasure probability 0.5. This table shows that the special cases occur often enough that they can reduce the latency significantly.

TABLE I
FREQUENCY OF RATE 0 AND 1 CODES IN P2 DECODING

Code Rate		1/3		1/2		2/3		3/4	
P2 Code Rate		0	1	0	1	0	1	0	1
N	64	3	1	1	1	1	3	0	4
	1024	12	1	7	7	1	12	1	14
	16384	53	6	35	35	6	53	0	55

IV. SYNTHESIS RESULTS AND COMPARISONS

This section reports some FPGA synthesis results for the TPSC decoder and compares them with those for the semi parallel successive cancellation (SPSC) decoder of [4], which is the only reference we could find with a comparable implementation study. The results are presented in Table II. The FPGA used for synthesis was an ALTERA STRATIX IV EP4SGX530KH40C2 device. All decoders in the table use a $Q = 5$ bit precision for representing the LLRs. The table shows that TPSC fares better than SPSC in several respects: it uses fewer FPGA resources (LUT, FF, RAM), has a faster clock speed f, and a better throughput (T/P) for any coding rate R. Furthermore, the clock frequency of TPSC decreases with increasing code length at a significantly slower rate than that of SPSC.

TABLE II
FPGA SYNTHESIS RESULTS FOR TPSC AND SPSC.

Decoder	N	PE	LUT	FF	RAM (bits)	f (MHz)	T/P (Mbps)
TPSC	64	8	620	338	320	240	$> 87\,R$
TPSC	1024	32	1940	748	7136	239	$> 112\,R$
SPSC	1024	16	2888	1388	11904	196	$87\,R$
SPSC	1024	64	4130	1691	15104	173	$85\,R$
TPSC	16384	128	7815	3006	114560	230	$> 118\,R$
SPSC	16384	64	29897	17063	184064	113	$53\,R$

To explain these results several remarks are in order. First, it should be noted that the TPSC used in this study uses the heuristic method mentioned in III-D, which explains why TPSC has a better throughput than SPSC. Second, the lower RAM usage of TPSC is explained by the fact that TPSC resorts to recalculations instead of storing LLRs in RAM. Third, the faster clock speed of TPSC is explained by the fact that TPSC uses a PSUL of size $O(\sqrt{N})$ while SPSC uses a PSUL of size $O(N)$. In [4], the PSUL size is identified as an important factor in determining the clock speed, which explains the better performance of TPSC in this regard. Fourth, the smaller PSUL of TPSC also helps bring down its complexity significantly; this is most evident in the comparison between TPSC and SPSC at block-length 16384 where TPSC uses twice as many PEs but has a smaller LUT/FF consumption; the larger LUT/FF utilization of SPSC can only be attributed to PSUL.

V. FUTURE WORK

The TPSC decoder architecture presented in this study was based on the representation of a polar code as a two-dimensional array code. It would be of interest to study the extension of the ideas presented in this paper to the case where a polar code of length $N = N_1 N_2 \cdots N_m$ is represented as an m-dimensional array code with a code of length N_i along the ith dimension.

ACKNOWLEDGMENT

This work was supported in part by TÜBİTAK under grant 110E243 and in part by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM# (contract n.318306).

REFERENCES

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, pp. 3051–3073, July 2009.

[2] E. Arıkan, "Polar codes: A pipelined implementation," *Proc. 4th Int. Symp. Broadband Communications (ISBC, 2010)*, Melaka, Malaysia, 11-14 July 2010.

[3] C. Leroux, I. Tal, A. Vardy, W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," The 36th International Conference on Acoustics, Speech and Signal Processing (ICASSP, 2011), Prague, May 22-27, 2011.

[4] C. Leroux, A. J. Raymond, G. Sarkis, W. J. Gross, "A Semi-Parallel Successive-Cancellation Decoder for Polar Codes," Signal Processing, IEEE Transactions on , vol. 61, no. 2, pp. 289-299, Jan. 15, 2013.

[5] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," The 8th International Symposium on Wireless Communication Systems, (ISWCS 2011), Aachen, 2011.

[6] Z. L. Huang, C. J. Diao, M. Chen, "Latency reduced method for modified successive cancellation decoding of polar codes," Electronics Letters , vol. 48, no. 23, pp. 1505-1506, 8 Nov. 2012.

[7] C. Zhang, B. Yuan, K. K. Parhi, "Reduced-latency SC polar decoder architectures," 2012 IEEE International Conference on Communications (ICC), pp. 3471-3475, 10-15 June 2012.

[8] A. Mishra, A. J. Raymond, L. G. Amaru, G. Sarkis, C. Leroux, P. Meinerzhagen, A. Burg, W. J. Gross, "A Successive Cancellation Decoder ASIC for a 1024-bit Polar Code in 180nm CMOS," Proceedings of IEEE Asian Solid-State Circuits Conference (A-SSCC 2012), Nov. 12-14, 2012, Kobe, Japan.