Optimized sorting network for successive cancellation list decoding of polar codes

Kun Wang, Li Li, Feng Han, Fan Feng, Jun Lin, Yuxiang Fu, and Jin Sha

School of Electronic Science and Engineering, Nanjing University,
Nanjing 210023, China

Abstract: In this paper, we propose an optimized bitonic sorting architecture and a hybrid sorting architecture for addressing the sorting problem of successive cancellation list decoders for polar codes. Since half of the $2^L$ metrics are already sorted, lots of redundant sorting operations can be deleted. According to this property and the characteristics of the sorting network, we put forward several optimization strategies to reduce the compare-and-exchange units and the pipeline depths. For the list size $L \leq 32$, the synthesis results show that the area of proposed hybrid sorter is at least 22% smaller than existing sorters, and the latency is at least 25% smaller.

Keywords: bitonic sorter, hybrid sorter, polar codes, successive cancellation list decoding

Classification: Integrated circuits

References

[1] E. Arikan: “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” IEEE Trans. Inf. Theory 55 (2009) 3051 (DOI: 10.1109/TIT.2009.2021379).
[2] I. Tal and A. Vardy: “List decoding of polar codes,” IEEE Trans. Inf. Theory 61 (2015) 2213 (DOI: 10.1109/TIT.2015.2410251).
[3] K. Niu and K. Chen: “CRC-aided decoding of polar codes,” IEEE Commun. Lett. 16 (2012) 1668 (DOI: 10.1109/LCOMM.2012.090312.121501).
[4] J. Lin and Z. Yan: “An efficient list decoder architecture for polar codes,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 23 (2015) 2508 (DOI: 10.1109/TVLSI.2014.2378992).
[5] A. Balatsoukas-Stimming, et al.: “Hardware architecture for list successive cancellation decoding of polar codes,” IEEE Trans. Circuits Syst. II, Exp. Briefs 61 (2014) 609 (DOI: 10.1109/TCSII.2014.2327336).
[6] A. Balatsoukas-Stimming, et al.: “LLR-based successive cancellation list decoding of polar codes,” IEEE Trans. Signal Process. 63 (2015) 5165 (DOI: 10.1109/TSP.2015.2439211).
[7] A. Balatsoukas-Stimming, et al.: “On metric sorting for successive cancellation list decoding of polar codes,” IEEE International Symposium on Circuits and Systems (ISCAS) (2015) 1993 (DOI: 10.1109/ISCAS.2015.7169066).

© IEICE 2017
DOI: 10.1587/elex.14.20170735
Received July 14, 2017
Accepted August 17, 2017
Publicized August 30, 2017
Copyedited September 25, 2017
1 Introduction

Polar codes are a class of capacity achieving channel codes with low complexity butterfly encoding and successive cancellation (SC) decoding algorithms [1]. However, the performances of polar codes using traditional SC decoding are unsatisfactory when the code length is short or medium. In order to solve this problem, a successive cancellation list (SCL) [2] decoding algorithm was proposed to improve the performances of polar codes with finite length. The SCL decoder has a corresponding list size $L$. At each decoding stage, $L$ decoding paths are considered concurrently, and a single output codeword is selected from the list. Moreover, the cyclic redundancy check (CRC) [3, 4] is adopted to select the output codeword, which makes the error-rate performance of polar codes competitive with low-density parity-check codes and turbo codes.

Instead of using the greedy one-time-pass search of the classic SC decoding, a breadth-first search is adopted in the SCL decoding algorithm. At each decoding step, the decoder is duplicated into two parallel paths continuing in either possible direction. Only the $L$ best paths are maintained for the final solution. Therefore, selecting the $L$ best paths out of $2L$ possible paths is an important operation of SCL decoding. However, this selecting operation may lie in the critical path of the decoder that determines the maximum operating frequency [4, 5]. As a result, considerable attention is paid to developing low-complexity sorting architectures recently [6, 7, 8]. A simplified bubble sorter and a pruned bitonic sorter were proposed in [7], and a sorting architecture which separates the odd-indexed and the even-indexed elements was described in [8].

In this paper, we propose an optimized bitonic sorter and a hybrid sorter for the sorting problem in SCL decoder. Through analyzing characteristics of the sorting network and the particularity of application, several optimization methods are proposed. By comparing our proposed sorters with the existing sorters, the hybrid architecture performs better on the hardware resources and latencies. For the most common range $L \leq 32$, the synthesis results show that the area of proposed hybrid sorter is at least 22% less than the pruned odd-even sorter, and the latency is at least 25% less.

2 Problem formulation

The sorting problem is to choose the $L$ smallest metrics out of $2L$ possible candidates. Let $M = [m_0, m_1, \ldots, m_{2L-1}]$ denote the metrics of the $2L$ child candi-
dates to be sorted. These $2L$ candidates are not arbitrary. Half of them are already sorted in the previous step and the rest are obtained by adding positive real values to the existing $L$ metrics. More specifically, let $o_0 \leq o_1 \leq \ldots \leq o_{L-1}$ be the $L$ sorted path metrics from the previous step of SCL decoding. Then, the metrics of the $2L$ child candidates in $M$ are computed by the following two equations:

$$m_{2l} = o_l$$

$$m_{2l+1} = o_l + a_l$$

where $a_l \geq 0$, for all $l = 0, 1, \ldots, L - 1$. Thus, these $2L$ elements of $M$ have the following two properties:

$$m_{2l} \leq m_{2(l+1)}$$

$$m_{2l} \leq m_{2l+1}$$

where $0 \leq l \leq L - 2$ for Eq. (3), and $0 \leq l \leq L - 1$ for Eq. (4).

Therefore, these two particular properties can be used to simplify the hardware implementation of selecting the $L$ best paths. Moreover, selecting the $L$ smallest path metrics is sufficient. There is no need to sort these $L$ smallest metrics.

### 3 Related work

Considering the importance of the metric sorter in the SCL decoders, several kinds of sorters have been proposed in the literature. In [7], Alexios Balatsoukas-Stimming proposed a simplified bubble sorter and a pruned bitonic sorter, as shown in Fig. 1 and Fig. 2. Each vertical line represents a compare-and-exchange (CAE) unit that has two inputs and two outputs. If the values of upper input are smaller than the lower line, they are directed to the corresponding outputs; otherwise, they are exchanged. The latency of a sorting network is proportional to its depth, which is described on the top of these figures. As mentioned in the previous section, the known particular properties of metrics are exploited to simplify the general bubble sorter and bitonic sorter. As shown in Fig. 1, all CAE units in dotted line can be removed in the simplified bubble sorter, while all the depicted CAE units are required in the full bubble sorter. Thus, for $2L = 8$ inputs, the simplified bubble sorter needs 6 CAE units and 3 CAE unit stages process. Fig. 2 shows that for $2L = 16$ inputs, the pruned bitonic sorter needs 46 CAE units and 9 CAE unit stages process.

However, the simplified bubble sorter and the pruned bitonic sorter do not make full use of the fact that half of the metrics are already sorted in the previous step of SCL decoding. In order to avoid performing redundant sorting operations on the metrics that are already sorted, Byeong Yong Kong [8] proposed an efficient sorting architecture which separates the odd-indexed and the even-indexed elements. In order to further reduce the complexity of implementation, the author adopted the odd-even sorting network as the basic architecture which is simpler than the bitonic sorter. Fig. 3 shows that for $2L = 16$ inputs, Byeong Yong Kong’s pruned odd-even sorter needs 27 CAE units and 9 CAE unit stages process.
4 Our approach

In this section, we present two kinds of sorting network for the sorting problem in SCL decoders. Since all of these existing sorters ignored that the selected $L$ smallest path metrics do not need to be sorted, we can make full use of this characteristic to further reduce the hardware complexity.

4.1 Optimized architecture based on bitonic sorter

Bitonic sorter is one of the fastest sorting networks [9, 10], which is composed of a number of bitonic mergers ($BM$) to recursively merges two ascending sets into a single sorted set. For example, the 16-input bitonic sorter as shown in Fig. 4 has eight parallel 2-input bitonic merger ($BM - 2$) units, four parallel $BM - 4$ units,
two parallel $BM - 8$ units, and one $BM - 16$ unit. An arbitrary sequence is divided into eight pairs and each pair is sorted into a bitonic sequence by $BM - 2$ unit. The adjacent bitonic sequences are merge-sorted repeatedly through all stages until the entire sequence is sorted.

![Diagram](image)

**Fig. 4.** A general bitonic sorter when $2L = 16$.

A $2L$-input bitonic merger is used to form a sorted sequence of length $2L$ from two sorted sequences of length $L$. It consists of a half-cleaner and two modified $L$-input bitonic mergers. The half-cleaner produces two bitonic subsequences, where all elements in the upper half are smaller than or equal to those of the lower half. The subsequences themselves are sorted by the two modified $L$-input bitonic mergers, respectively. Based on the architecture of the bitonic sorter and the particular properties of the sorting problem in SCL decoders, several optimization methods are proposed as follow:

(i) As stated in the preceding section, we only need to find out the $L$ smallest values of the $2L$ metrics during SCL decoding, and these selected metrics do not need to be sorted. Since the $L$ smallest metrics are picked out after the half-cleaner of the last $BM - 2L$ unit, the two modified $BM - L$ units in the last $BM - 2L$ unit can be removed. Furthermore, as was in the simplified bitonic sorter, the results of all CAE units in the first $BM - 2$ are already known from Eq. (4). Thus, the first $BM - 2$ can also be completely removed from the sorting network. As shown in Fig. 5, the blue dotted vertical lines are used to indicate the CAE units that are deleted by this optimization method;

(ii) Let $\{b_i\}$ denote the elements before the last $BM - 2L$ in a $2L$-input bitonic sorter, as shown in Fig. 4. We can easily know that $\{b_i(0 \leq i \leq L - 1)\}$ and $\{b_i(L \leq i \leq 2L - 1)\}$ are already sorted in the previous bitonic mergers. Deduced from Eq. (3) and Eq. (4), $b_L$ is equal to $m_L$ and is the smallest element in $\{b_i(L \leq i \leq 2L - 1)\}$. It also can be inferred that there are at least $L/2$ elements equal to or less than $m_L$ from Eq. (1) and Eq. (2). So $\{b_i(i = 0, 1, \ldots, L/2 - 1)\}$ are a sorted list of the $L/2$ smallest elements in $M$. Thus in the last $BM - 2L$, all CAE units involving these inputs can be removed. This inference can be further expanded to all bitonic mergers in this application. For each $BM - N$ in this
application, the upper \( N/4 \) inputs are already the smallest, so we can reduce all involved CAE units which are the orange dotted vertical lines in Fig. 5;

(iii) Since selecting the \( L \) smallest path metrics is sufficient, there is no need to require anything about the order. As soon as any metric is found belonging to the smallest \( L \) metrics, the subsequent CAE units can be deleted. Take a \( 2L = 16 \) inputs bitonic sorter in Fig. 4 for example, the four smallest elements of \( M \) are picked out after the half-cleaner of the upper \( BM = 8 \). So the CAE unit which is the red dotted vertical line in Fig. 5 can be removed. In addition, a 64-input bitonic sorter is used to illustrate this method more clearly. The upper half of a 64-input bitonic sorter is shown in Fig. 6, and the red dotted vertical lines represent the CAE units that are deleted by this optimization method. It can be seen that the deleted parts in each top \( BM - N \) are equivalent to a modified \( BM - N/4 \);

(iv) On the other hand, the subsequent CAE units can also be deleted when the metric is found not belonging to the smallest \( L \) metrics. Take a \( 2L = 16 \) inputs bitonic sorter in Fig. 4, for instance. In the previous description, we have known

---

Fig. 5. Optimized bitonic sorter when \( 2L = 16 \).

Fig. 6. The upper half of a 64-input bitonic sorter.
that $b_8$ is equal to $m_8$ and there are at least four elements equal to or less than $m_8$. So there are at least eight elements equal to or less than $b_{12}$. Then any element in \{ $b_i$ ($i = 12, 13, \ldots, 15$) \} does not belong to the $L$ smallest elements. Furthermore, we can easily infer that the lowest input of the lowest $BM - 4$ and the lowest two inputs of the lower $BM - 8$ can never be among the $L$ smallest elements. All CAE units involving these can be removed. As shown in Fig. 5, the green dotted vertical lines represent the removed CAE units by this optimization method. The removed parts in each bottom $BM - N$ are equivalent to a modified $BM - N/2$.

For $2L = 16$ inputs, the proposed bitonic sorter which is composed of the solid vertical lines in Fig. 5 needs 20 CAE units and 6 CAE unit stages process.

### 4.2 Proposed hybrid architecture

Odd-even sorter is another fast sorting networks, which is similar to the bitonic sorter. The odd-even sorter is composed of a number of odd-even mergers ($OEM$), and the odd-even merger is composed of a half-cleaner and a rear network. All the solid vertical line and the dotted vertical line in Fig. 7 form a general 16-input odd-even sorter. The sorter has eight parallel 2-input odd-even merger ($OEM - 2$) units, four parallel $OEM - 4$ units, two parallel $OEM - 8$ units, and one $OEM - 16$ unit. It can be recognized that the odd-even merger is simpler than the bitonic merger.

In Fig. 7, we apply all the proposed optimization methods to a general 16-input odd-even sorting network. The optimized 16-input odd-even sorting network needs 23 CAE units and 8 CAE unit stages process, which are both larger than the proposed bitonic sorting network. This is mainly because the data exchange between the upper half and the lower half still exists after the half-cleaner of the $OEM - 16$. Optimization method (i) can only delete all $OEM - 2$ units. According to optimization method (ii) and (iv), the entire half-cleaner of the $OEM - 16$ can be removed. Another thing to be noted is that the CAE units optimized by method (iv) in the 16-input rear network of the OEM-16 are equivalent to an 8-input rear network.

To achieve a better performance, we adopt a hybrid architecture by replacing the last $2L$-input odd-even merger with a bitonic merger. A 16-input optimized
hybrid architecture is shown in Fig. 8. This architecture needs 20 CAE units and 6 CAE unit stages process, which is equal to the proposed bitonic sorting network. But with the increase of $L$, the hybrid architecture requires less CAE units than the bitonic one. Detailed analysis about resources and latencies will be described in the next section.

As can be seen from Fig. 8, the optimization method (i) has the same effect in the hybrid sorter as in the bitonic sorter. The optimization method (ii) removes half of the CAE units in the last 16-input half-cleaner. For each remaining $N$-input odd-even merger, the CAE units that are related to the upper $N/4$ inputs are optimized by method (ii). Half of the $N$-input half-cleaner and an equivalent $N/2$-input rear network are removed by the optimization method (iv) in each bottom $N$-input odd-even merger. However, the optimization method (iii) does not work in the 16-input hybrid sorter. So we use a 64-input hybrid sorter to illustrate, the upper half of a 64-input hybrid sorter is shown in Fig. 9. The deleted parts in each top OEM – $N$ ($N \geq 16$) are equivalent to a $N/4$-input rear network.

Fig. 8. Proposed hybrid sorter when $2L = 16$.

Fig. 9. The upper half of a 64-input hybrid sorter.
5 Results

In this section, we first analyze the pipeline depth in the proposed sorting networks. The pipeline depth of a $2L$-input bitonic merger is $\log_2 2L$ and a $2L$-input general bitonic sorter has $\log_2 2L$ consecutive bitonic mergers. So, a $2L$-input bitonic sorter has $\log_2 2L \times (\log_2 2L + 1)/2$ CAE stages. Since the CAE units in the first stage and the two modified $BM - L$ in the last $BM - 2L$ are removed in the proposed bitonic sorter, the total pipeline depth for $2L$ inputs is $\log_2 2L \times (\log_2 2L + 1)/2 - \log_2 2L$. The hybrid architecture has the same pipeline depth.

When $2L = 4$, only one CAE unit is required for the sorting problem. Then, we will derive the formulae for the numbers of CAE units in the proposed sorting networks when $2L \geq 8$.

A general $2L$-input bitonic sorter contains $\log_2 2L$ stages of bitonic merger, and each $2^i$-input ($i = 1, 2, \ldots, \log_2 2L$) bitonic merger contains $i \times 2^{i-1}$ CAE units. Thus, the total number of CAE units in a general $2L$-input bitonic sorter is

$$\sum_{i=1}^{\log_2 2L} \frac{2L}{2^i} \times i \times 2^{i-1} = \sum_{i=1}^{\log_2 2L} 2L \times \frac{i}{2}.$$  

By using the optimization method (i), the first and the last bitonic merger of a general $2L$-input bitonic sorter are removed, except the last $2L$-input half-cleaner which contains $2L/2$ CAE units. So the number of the rest CAE units after optimization method (i) is

$$\sum_{i=2}^{\log_2 2L-1} 2L \times \frac{i}{2} + 2L/2.$$  

Next, we will analyze the optimization method (ii). In an $N$-input bitonic merger, there are $N/4$ CAE units that are related to the top $N/4$ inputs in the first and the second CAE stages, and $N/8$ CAE units that are related to the top $N/4$ inputs in each of the remaining CAE stages. Thus, the number of CAE units involved with the top $N/4$ inputs in an $N$-input bitonic merger is $2 \times N/4 + (\log_2 N - 2) \times N/8$. For each $2^i$-input ($i = 2, 3, \ldots, \log_2 2L - 1$) bitonic merger in the rest of the bitonic sorter, the number of CAE units eliminated by optimization method (ii) is $i \times 2^{i-3} + 2^{i-2}$. Moreover, $2L/4$ CAE units in the last half-cleaner can be removed by method (ii). As described in the preceding section, an equivalent modified $BM - N/4$ in an $N$-input bitonic merger can be optimized by method (iii) and an equivalent modified $BM - N/2$ can be optimized by method (iv). So, the numbers of the removed CAE units directed by optimization method (iii) and (iv) in each bitonic merger stage of the rest are $(i - 2) \times 2^{i-3}$ and $(i - 1) \times 2^{i-2}$, respectively. Let $R_{pbs}$ denote the total number of CAE units used in our proposed bitonic sorter, then the resource consumption can be calculated by the following equation:

$$R_{pbs} = \sum_{i=2}^{\log_2 2L-1} 2L \times \frac{i}{2} + 2L/2 - \sum_{i=2}^{\log_2 2L-1} \frac{2L}{2^i} \times (i \times 2^{i-3} + 2^{i-2}) - 2L/4 - \sum_{i=2}^{\log_2 2L-1} (i - 2) \times 2^{i-3} - \sum_{i=2}^{\log_2 2L-1} (i - 1) \times 2^{i-2}.$$

(5)
A general $2L$-input odd-even sorter also contains $\log_2 2L$ odd-even merger stages, and each $2^i$-input ($i = 1, 2, \ldots, \log_2 2L$) odd-even merger contains $2^{i-1} \times (i - 1) + 1$ CAE units. The number of CAE units in a $2^i$-input rear network is $2^{i-1} \times (i - 1) + 1 - 2^{i/2} = 2^{i-1} \times (i - 2) + 1$. Similarly, the first odd-even merger and the last bitonic merger (except the half-cleaner) in the full hybrid sorter can be removed by adopt the optimization method (i). For a $2L$-input hybrid sorter, the number of the remaining CAE units after optimization method (i) is $\sum_{i=2}^{\log_2 2L-1} \frac{2L}{2^i} \times [2^{i-1} \times (i - 1) + 1] + 2L/2$. Then, we will analyze the effect of the optimization method (ii). In an $N$-input odd-even merger, $N/4$ CAE units are related to the top $N/4$ inputs in the first CAE stage, no CAE unit is related to the top $N/4$ inputs in the second CAE stage, and $N/8$ CAE units are related in each of the remaining CAE stages. Thus, the number of CAE units involved with the top $N/4$ inputs is $N/4 + (\log_2 N - 2) \times N/8 = \log_2 N \times N/8$. For each $2^{i}$-input ($i = 2, 3, \ldots, \log_2 2L - 1$) odd-even merger in the rest of the full hybrid sorter, the number of CAE units eliminated by optimization method (ii) is $i \times 2^{i-3}$. In addition, $2L/4$ CAE units in the last half-cleaner can be removed by method (ii). Moreover, the method (iii) can optimize an equivalent $N/4$-input rear network in an $N$-input bitonic merger, and the method (iv) can optimize half of the $N$-input half-cleaner and an equivalent $N/2$-input rear network. So, the numbers of the removed CAE units directed by optimization method (iii) and (iv) in each odd-even merger stage of the rest are $(i - 4) \times 2^{i-3} + 1$ and $(i - 2) \times 2^{i-2} + 1$, respectively. Let $R_{phs}$ denote the total number of CAE units used in the proposed hybrid sorter, then the resource consumption can be calculated as follows:

$$R_{phs} = \sum_{i=2}^{\log_2 2L-1} \frac{2L}{2^i} \times [2^{i-1} \times (i - 1) + 1] + 2L/2$$

$$- \sum_{i=2}^{\log_2 2L-1} \frac{2L}{2^i} \times i \times 2^{i-3} - 2L/4$$

$$- \sum_{i=2}^{\log_2 2L-1} [(i - 4) \times 2^{i-3} + 1]$$

$$- \sum_{i=2}^{\log_2 2L-1} [(i - 2) \times 2^{i-2} + 1].$$

The numbers of CAE units and CAE stages with different $L$ are presented in Table I. We observe that the number of CAE units of the proposed hybrid sorter is less than that of the proposed bitonic sorter with the increasing of $L$. The simplified bubble sorter [7] and the pruned odd-even sorter [8] are selected to compare with our two proposed sorters. For different $L$, the CAE stages of the two proposed architectures are the smallest. According to the derived formulae, we can figure out that the depth of the two proposed architecture are $\log_2 2L - 1$ less than the pruned odd-even sorter. The number of CAE units of the proposed bitonic sorter is smaller than the pruned odd-even sorter for $L \leq 64$, and the hybrid architecture demands the smallest CAE units for different $L$. It is clear that the proposed hybrid sorter shows a better performance than the other sorters.
Table II presents the synthesis results for the pruned odd-even sorter [8] and our proposed architectures. The resultant areas and latencies are based on the worst timing library in a 40-nm CMOS technology. The bits wide of the list elements are assumed to be 8. The latency of the two proposed sorter are nearly the same. For the most common range $L/C_20 \leq 32$, the area of the proposed hybrid sorter is at least 22% smaller than that of the pruned odd-even sorter, and the latency is at least 25% smaller.

6 Conclusion

In this paper, we adopted bitonic sorter as a basic building block and proposed several optimization methods. We take full advantage of the characteristics of the sorting network and the particular properties of the sorting problem in SCL decoding algorithms. Then we presented a hybrid sorter which combines the odd-even sorting network and bitonic sorting network. The formula analysis shows that, for various $L$, the numbers of CAE units and the pipeline depth of the hybrid architecture are smaller than that of the existing sorters. Compared to the pruned odd-even sorter, the reduction of area and latency is more than 22% and 25% respectively for $L \leq 32$.

Acknowledgments

This work was supported by National Nature Science Foundation of China under Grant No. 61176024, 61370040; The project on the Integration of Industry, Education and Research of Jiangsu Province BY2015069-05; The key Research and Development Program of Jiangsu Province under Grant No: BE2015153; Supported by the Fundamental Research Funds for the Central Universities under Grant No. 021014380030; The Project Funded by the PAPD of Jiangsu Higher Education Institutions (PAPD).