Partitioned Successive-Cancellation List Decoding of Polar Codes

Seyyed Ali Hashemi*, Alexios Balatsoukas-Stimming†, Pascal Giard*, Claude Thibeault◊, Warren J. Gross*

*McGill University, Montréal, Québec, Canada
†École polytechnique fédérale de Lausanne, Lausanne, Switzerland
◊École de technologie supérieure, Montréal, Québec, Canada

March 23, 2016
What is the problem?

- Polar codes are state-of-the-art codes with interesting properties
- Successive-Cancellation List (SCL) decoding can outperform LDPC codes
- SCL requires large amount of memory
  - High memory requirement translates into high area occupation

In this talk:

We reduce the memory usage of SCL and improve its performance!
Polar Codes

- First family of codes which can provably achieve the channel capacity with explicit construction and low-complexity decoding\(^1\)
- The encoding process consists of recursive application of a linear transformation of \(G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\) to get the generator matrix \(G^\otimes n\)
  - The channels are then sorted as being either *good* or *bad*
  - As the number of channels increases, the fraction of good channels tends to the channel capacity
- Different decoding schemes are available:
  - Successive-Cancellation (SC)
  - SC List

---

For a rate $\frac{K}{N}$ code $P(N, K)$, the $K$ best channels are found and the information bits are assigned to those channels.

The $N - K$ worse channels are set to a predefined value (usually 0). These are called *frozen bits* $\mathcal{F}$. 
Successive-Cancellation Decoding

\[
\alpha_I[i] = \text{sgn}(\alpha[i])\text{sgn}(\alpha[i + 2^{s-1}]) \min(|\alpha[i]|, |\alpha[i + 2^{s-1}]|),
\]
\[
\alpha_R[i] = \alpha[i + 2^{s-1}] + (1 - 2\beta_I[i])\alpha[i],
\]
\[
\beta[I] = \begin{cases} 
\beta_I[i] \oplus \beta_R[i], & \text{if } i < 2^{s-1} \\
\beta_R[i + 2^{s-1}], & \text{otherwise},
\end{cases}
\]
SC List Decoding

- SC decoding
  \[ \hat{u}_i = \begin{cases} 
  0, & \text{if } i \in \mathcal{F} \text{ or } \alpha_i \geq 0, \\
  1, & \text{otherwise}. 
\end{cases} \]

- SCL decoding
  \[ \hat{u}_i = \begin{cases} 
  0, & \text{if } i \in \mathcal{F} \\
  0 \text{ or } 1, & \text{otherwise}. 
\end{cases} \]

  \[ PM_i' = \begin{cases} 
  PM_{i-1}', & \text{if } \hat{u}_i = \frac{1}{2} (1 - \text{sgn}(\alpha_i')), \\
  PM_{i-1}' + |\alpha_i'|, & \text{otherwise}, 
\end{cases} \]

- Cyclic Redundancy Check (CRC) code can be used to help SCL find the correct candidate.
**Partitioned SCL Decoding**

- The decoder tree is broken into partitions (subtrees)
- SCL decoding is performed only on the partitions
- Standard SC rules are applied to the remainder of the decoding tree
- Each partition outputs a single candidate codeword which is selected with the help of a CRC and then sent to the next partition
- The decoding process starts with the standard SC update rules
- The decoder does not require memory to store $L$ entire trees of internal LLRs
- Only $L$ copies of the partitions on which SCL decoding is performed are stored
Partitioned SCL Decoding

level $n$

CRC-aided SCL

level $n - 1$

CRC-aided SCL

level $n - 2$

CRC-aided SCL

CRC-aided SCL

CRC-aided SCL

CRC-aided SCL
Memory Requirements

\[ M_{SC} = (2N - 1) Q_\alpha + N - 1 \]

\[ M_{SCL} = (N + (N - 1) L) Q_\alpha + L Q_{PM} + (2N - 1) L \]

\[ M_{PSCL} = \left( \sum_{k=0}^{P-1} \frac{N}{2^k} + \left( \frac{N}{2^{P-1}} - 1 \right) L \right) Q_\alpha + L Q_{PM} \]

\[ + \sum_{k=1}^{P-2} \frac{N}{2^k} + \left( \frac{N}{2^{P-2}} - 1 \right) L \]
Memory Requirements

Number of Partitions

Memory Bits

10^5

PSCL2
PSCL4
PSCL8
SC Bound
SCL2 Bound
SCL4 Bound
SCL8 Bound
Performance Results $\mathcal{P}(2048, 1024)$

![Graphs showing FER and BER for different codes](image)

- **FER**
  - SC
  - SCL2-CRC32
  - PSCL(2, 2)-CRC16
  - SCL4-CRC32
  - PSCL(4, 2)-CRC8
  - PSCL(4, 4)-CRC8

- **BER**
  - SC
  - SCL2-CRC32
  - PSCL(2, 2)-CRC16
  - SCL4-CRC32
  - PSCL(4, 2)-CRC8
  - PSCL(4, 4)-CRC8
## Implementation Results $P(2048, 1024)$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total (mm$^2$)</th>
<th>Memory (mm$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>0.723</td>
<td>0.413</td>
</tr>
<tr>
<td>SCL2-CRC32</td>
<td>1.563</td>
<td>0.702</td>
</tr>
<tr>
<td>SCL4-CRC32</td>
<td>3.075</td>
<td>1.214</td>
</tr>
<tr>
<td>PSCL(2, 2)-CRC16</td>
<td>1.189</td>
<td>0.540</td>
</tr>
<tr>
<td>PSCL(4, 2)-CRC8</td>
<td>0.909</td>
<td>0.415</td>
</tr>
<tr>
<td>PSCL(4, 4)-CRC8</td>
<td>1.356</td>
<td>0.543</td>
</tr>
</tbody>
</table>
Conclusions

- We proposed a novel PSCL decoding algorithm for polar codes.
- The code is broken into partitions and each partition is decoded with a CRC-aided SCL decoder.
- The memory requirements of PSCL decoder is significantly smaller than that of an SCL decoder.
- At equivalent error-correction performance, PSCL leads to memory and total area savings of 41% and 42% compared to a similar SCL decoder implementation.
- PSCL enables a coding gain of approximately 0.25 dB at a BER of $10^{-5}$ while occupying 13% less total area than the SCL decoder.

In short:
PSCL achieves better performance and reduces memory usage at the same time!
Thank you!