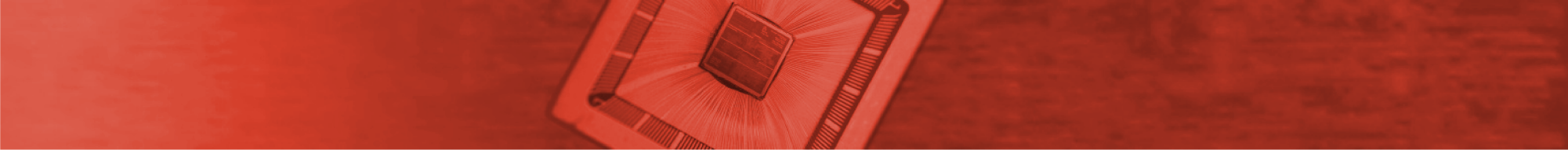


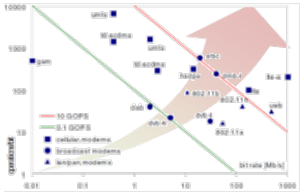
Communications Hardware In The Post-Happy-Scaling Era

Alexios Balatsoukas-Stimming
Telecommunications Circuits Laboratory
École polytechnique fédérale de Lausanne, Switzerland

March 7, 2019

Rutgers University

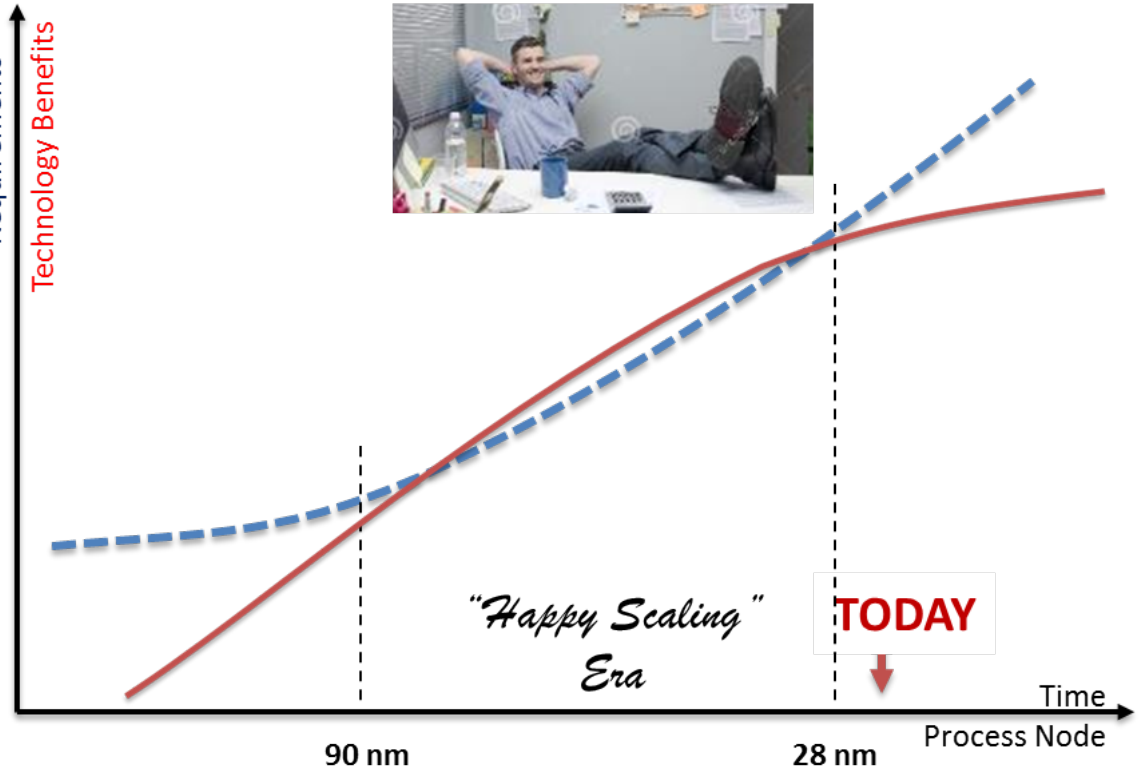
The End of the "Happy Scaling" Era

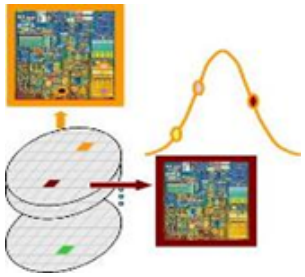


Increasing DSP algorithm complexity.

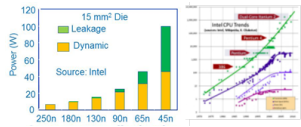
Increasing need for flexibility.

Increasing need for energy-efficiency.

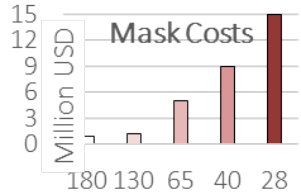




Severe variability & reliability issues.

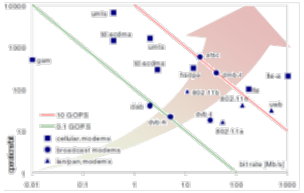


Vanishing energy & performance gains.



Skyrocketing NRE: mask & design cost.

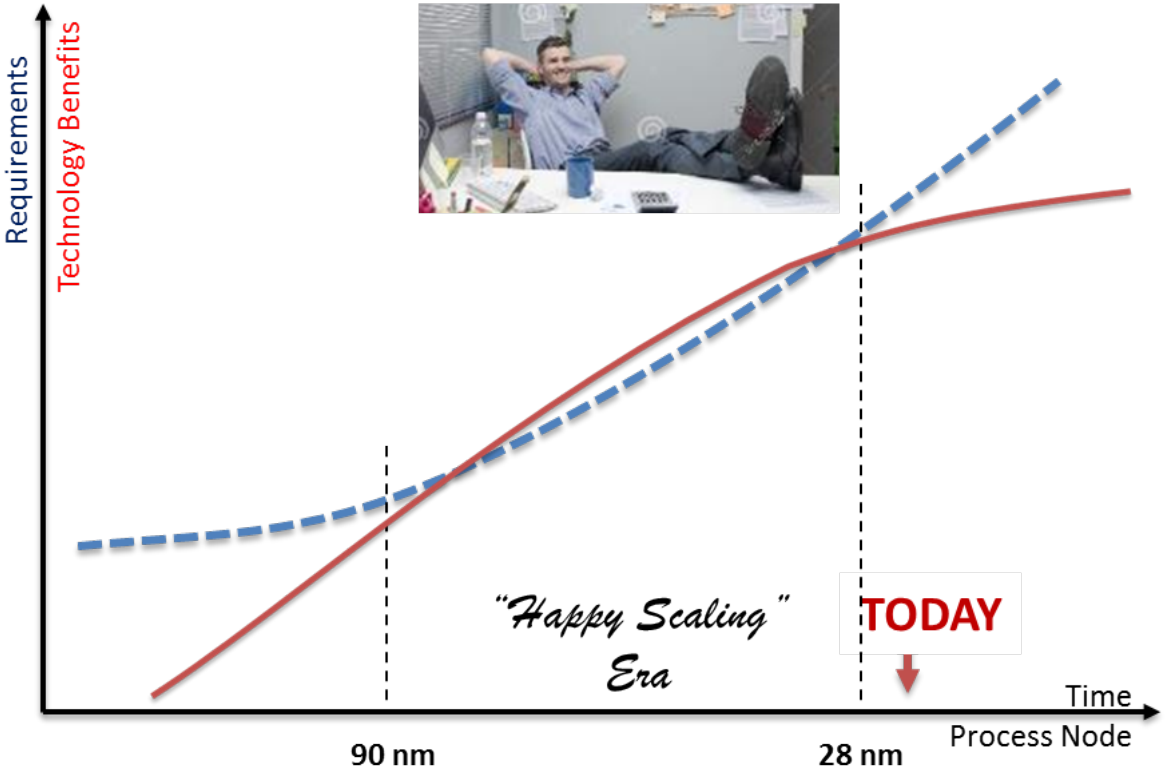
The End of the "Happy Scaling" Era

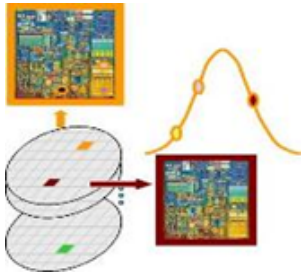


Increasing DSP algorithm complexity.

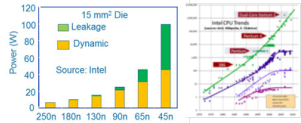
Increasing need for flexibility.

Increasing need for energy-efficiency.

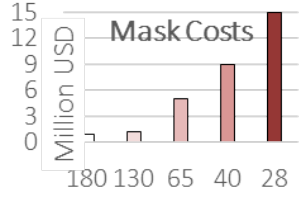




Severe variability & reliability issues.



Vanishing energy & performance gains.



Skyrocketing NRE: mask & design cost.

The Way Forward

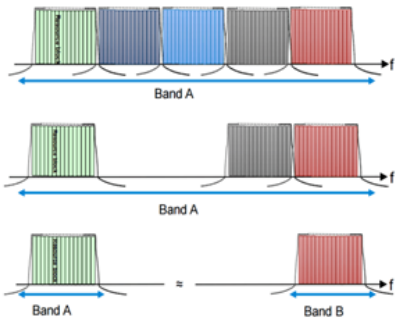
1. Algorithm/hardware co-design is **more pertinent than ever**
2. Maintaining progress will require **cross-layer** and **interdisciplinary** innovation

Outline

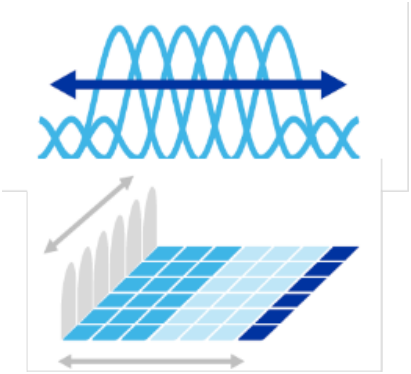
1. **Classical algorithm/hardware co-design:**
   - Hardware implementation of successive cancellation list decoding of polar codes
   - Successive cancellation flip decoding of polar codes

2. **Communications hardware meets information theory and machine learning:**
   - Polar error-correction decoding on faulty hardware
   - Terabit/s LDPC code decoders via quantized message passing
   - Neural networks for self-interference cancellation in full-duplex radios

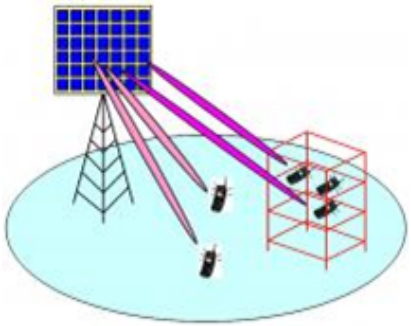
Technology Innovations in 5G



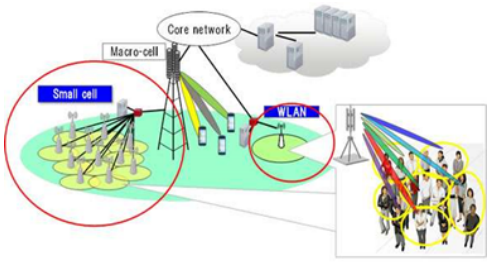
Wide Bandwidth and Carrier Aggregation



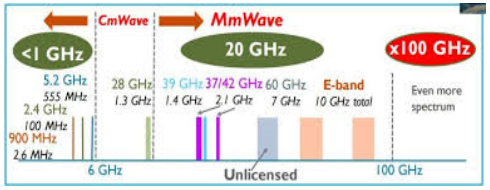
Flexible and Scalable OFDMA Air-Interface



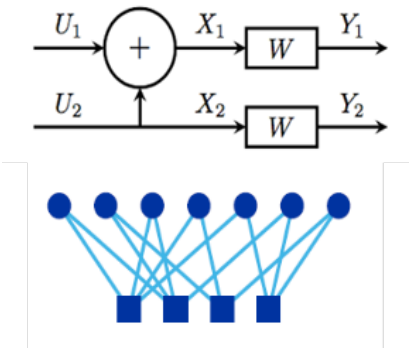
Massive MIMO



Small Cells and Advanced Cellular Concepts



New Radio Frequencies (mmWave)



Advanced Channel Codes (LDPC and polar)

Polar Codes

$\mathcal{A} = \{3, 5, 6, 7\}$

0
1
2
3
4
5
6
7

- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.

Polar Codes

$$\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}$$

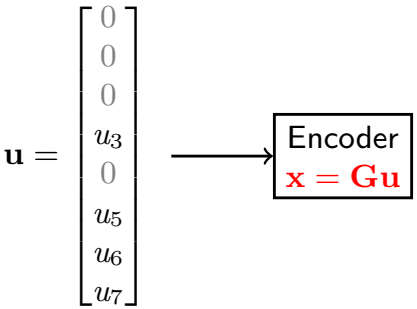
- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,

Polar Codes

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ u_3 \\ 0 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}$$

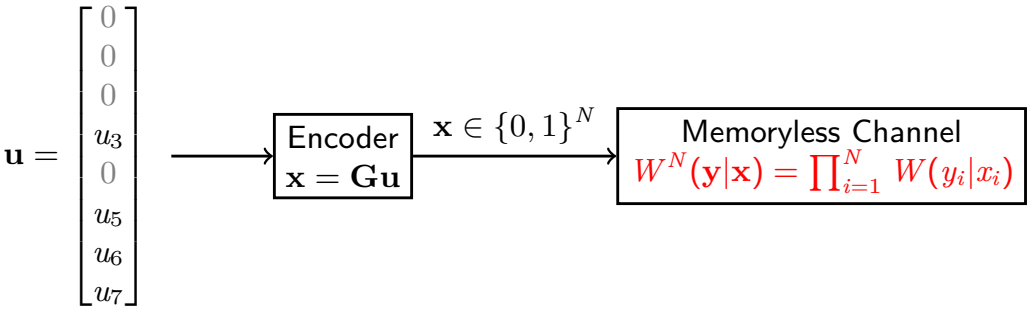
- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,
  - $\mathbf{u}_{\mathcal{A}^C} \leftarrow$ known-to-receiver frozen bits (say all-zero).

Polar Codes



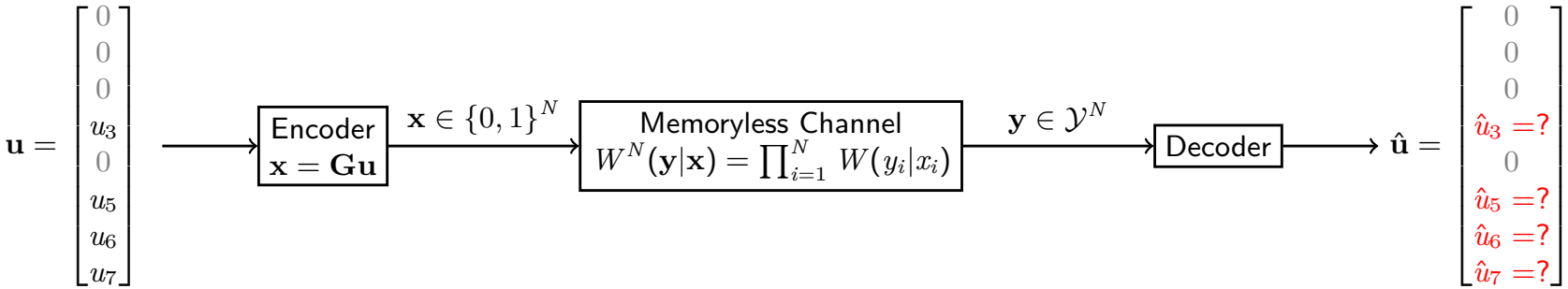
- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,
  - $\mathbf{u}_{\mathcal{A}^C} \leftarrow$ known-to-receiver frozen bits (say all-zero).
  - $\mathbf{x} \leftarrow \mathbf{Gu}$ (using $O(N \log N)$ binary additions).

Polar Codes



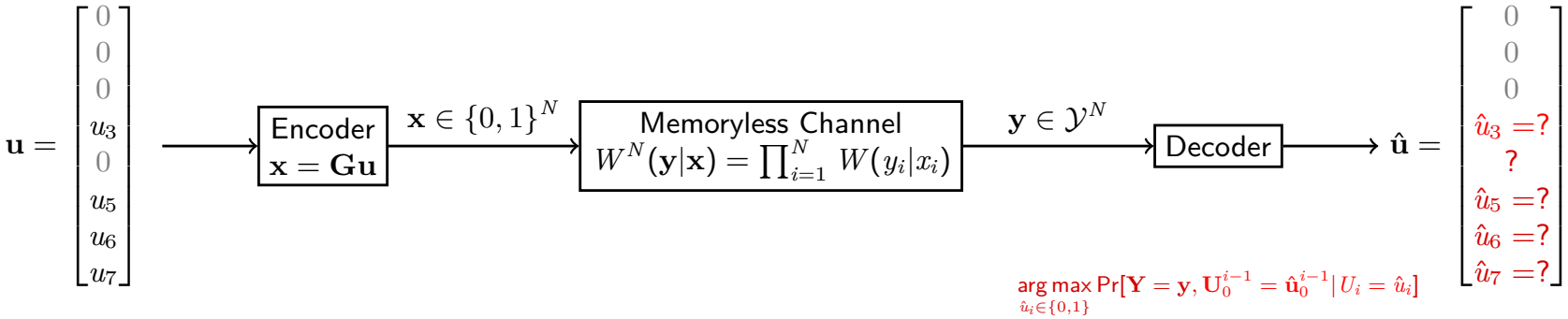
- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,
  - $\mathbf{u}_{\mathcal{A}^C} \leftarrow$ known-to-receiver frozen bits (say all-zero).
  - $\mathbf{x} \leftarrow \mathbf{Gu}$ (using $O(N \log N)$ binary additions).

Polar Codes



- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,
  - $\mathbf{u}_{\mathcal{A}^C} \leftarrow$ known-to-receiver frozen bits (say all-zero).
  - $\mathbf{x} \leftarrow \mathbf{G}\mathbf{u}$ (using $O(N \log N)$ binary additions).
- Decoding: Estimate the data bits $\hat{\mathbf{u}}_{\mathcal{A}}$.

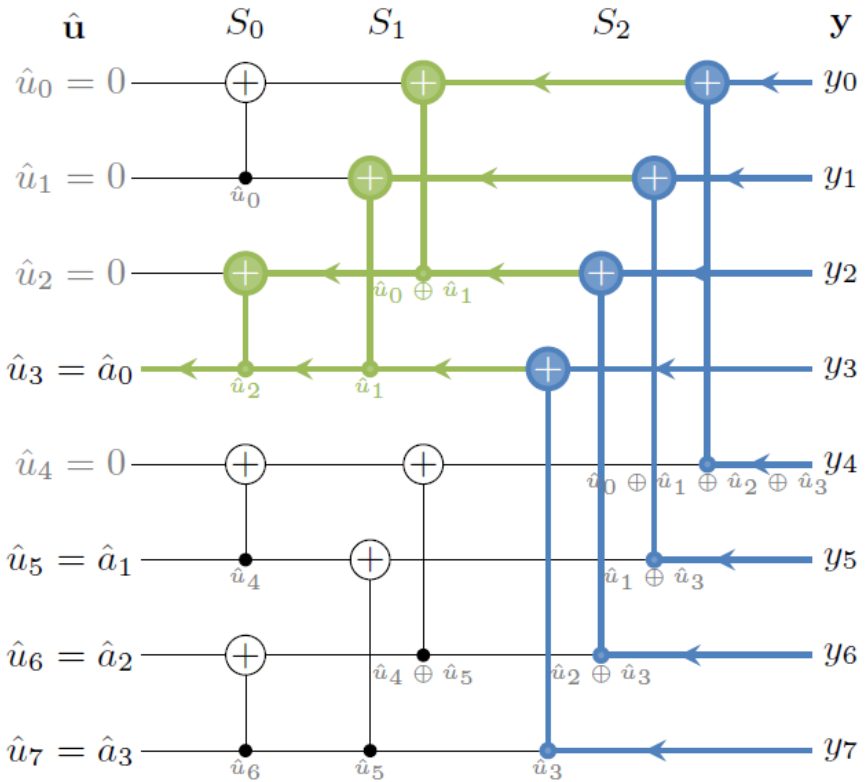
Polar Codes



- Construction:
  - Information indices: $\mathcal{A} \subset \{0, 1, \ldots, N-1\}$, $|\mathcal{A}| = NR$, $N = 2^n$.
- Encoding:
  - $\mathbf{u}_{\mathcal{A}} \triangleq [u_i, i \in \mathcal{A}]^T \leftarrow$ data bits,
  - $\mathbf{u}_{\mathcal{A}^C} \leftarrow$ known-to-receiver frozen bits (say all-zero).
  - $\mathbf{x} \leftarrow \mathbf{G}\mathbf{u}$ (using $O(N \log N)$ binary additions).
- Decoding: Estimate the data bits $\hat{\mathbf{u}}_{\mathcal{A}}$.
  - Successive Cancellation (SC) Decoding: At each level $i \in \mathcal{A}$, choose the best possible value of u_i given the past estimations and frozen bits.

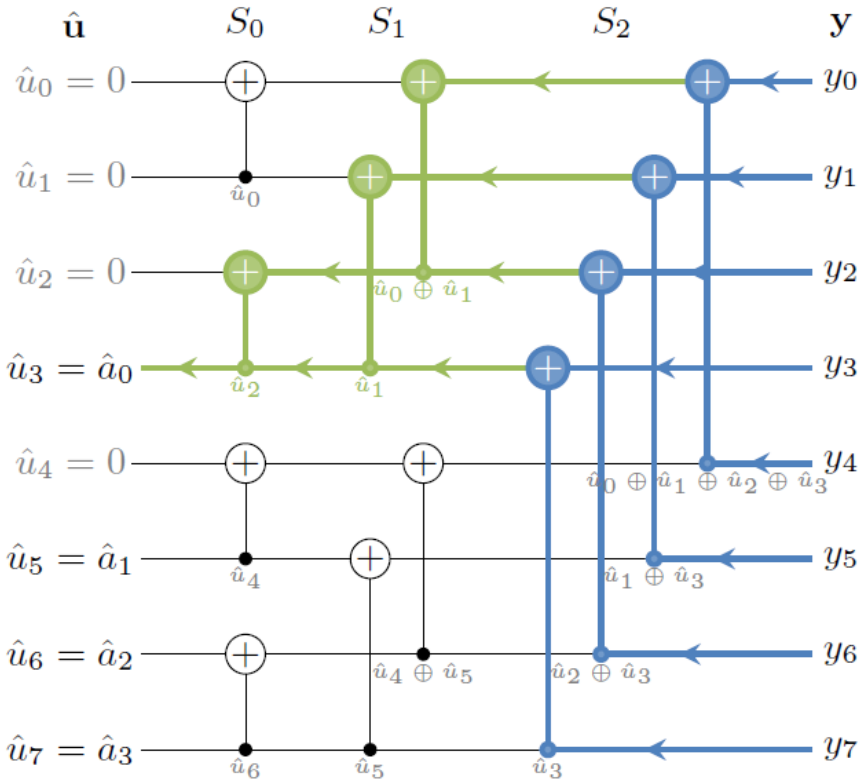
Successive Cancellation Decoding

- Successive traversal of a data dependency graph
- $N \log N$ nodes, each visited exactly once $\longrightarrow$ $O(N \log N)$ **time complexity**!
- Re-use of memory positions $\longrightarrow$ $O(N)$ **memory complexity**!

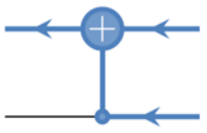


Successive Cancellation Decoding

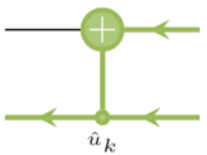
- Successive traversal of a data dependency graph
- $N \log N$ nodes, each visited exactly once $\longrightarrow$ $O(N \log N)$ **time complexity**!
- Re-use of memory positions $\longrightarrow$ $O(N)$ **memory complexity**!



- Two simple soft information update operations:



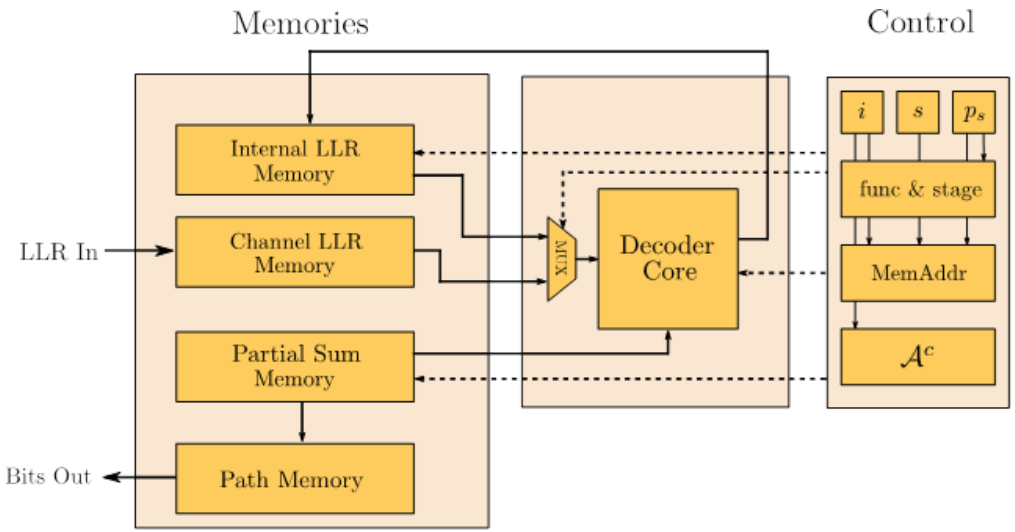
$$f(\alpha_i, \alpha_j) \approx \text{sgn}(\alpha_i)\text{sgn}(\alpha_j) \min(|\alpha_i|, |\alpha_j|)$$



$$g(\alpha_i, \alpha_j, \hat{u}_k) = \alpha_j + (-1)^{\hat{u}_k} \alpha_i$$

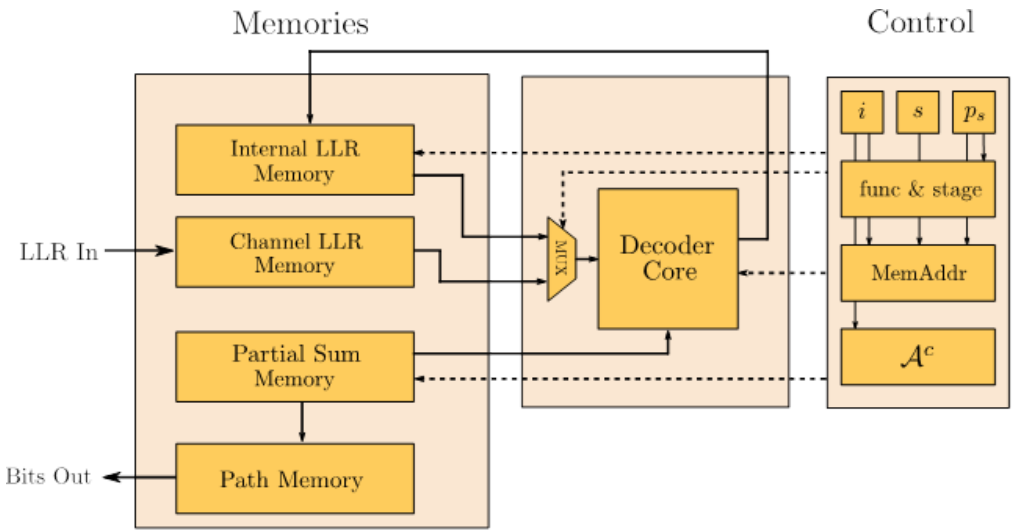
Hardware Implementation of SC Decoding

- **Decoder Core:** contains P processing elements that implement update rules
- **Memories:** store soft information, partial sums, and decoded codeword
- **Controller:** organizes memory reads and writes, and update rule selection



Hardware Implementation of SC Decoding

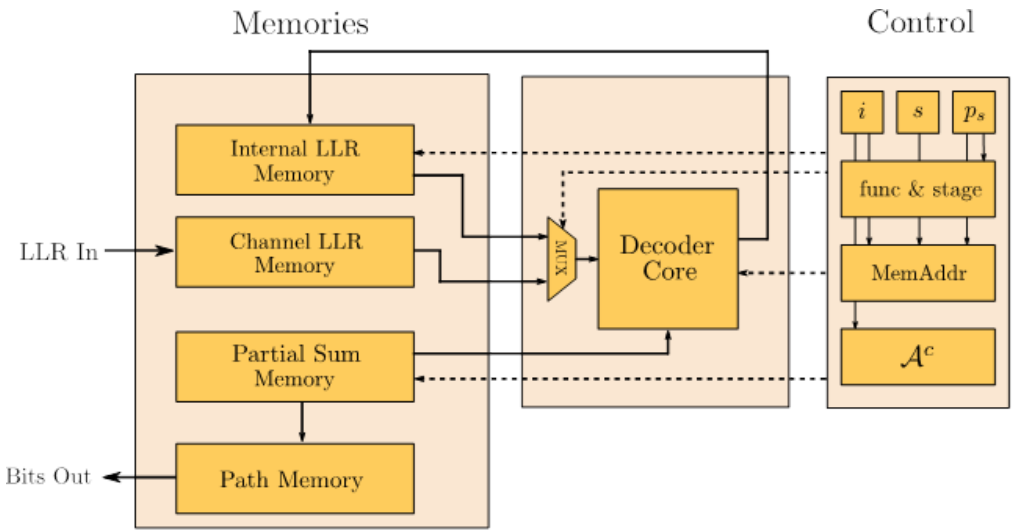
- **Decoder Core:** contains P processing elements that implement update rules
- **Memories:** store soft information, partial sums, and decoded codeword
- **Controller:** organizes memory reads and writes, and update rule selection



- **Simple** and **flexible** architecture
- **Compact** and **energy-efficient**

Hardware Implementation of SC Decoding

- **Decoder Core:** contains P processing elements that implement update rules
- **Memories:** store soft information, partial sums, and decoded codeword
- **Controller:** organizes memory reads and writes, and update rule selection



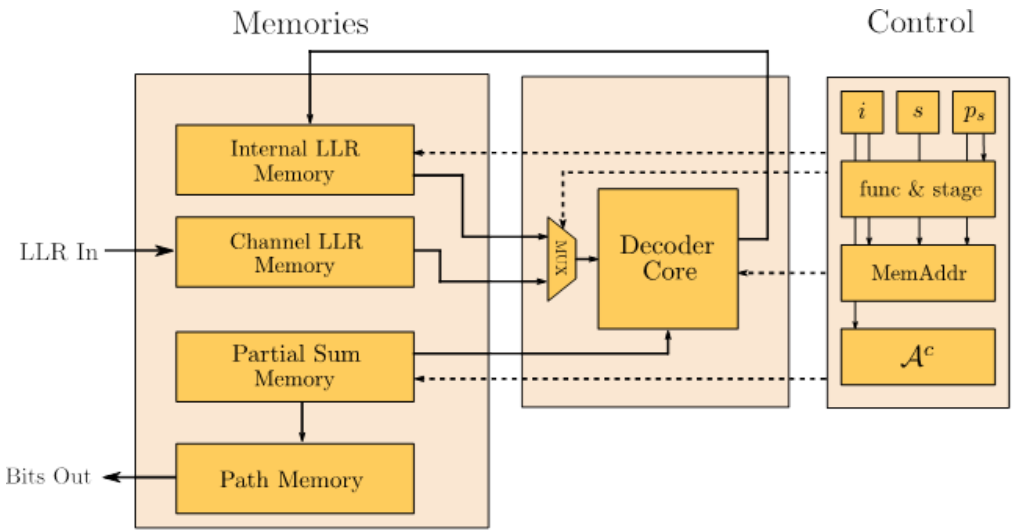
- **Simple** and **flexible** architecture
- **Compact** and **energy-efficient**

Two main challenges with SC decoding:

1. Low throughput due to sequential nature
2. Mediocre error-correcting performance due to error propagation

Hardware Implementation of SC Decoding

- **Decoder Core:** contains P processing elements that implement update rules
- **Memories:** store soft information, partial sums, and decoded codeword
- **Controller:** organizes memory reads and writes, and update rule selection



- **Simple** and **flexible** architecture
- **Compact** and **energy-efficient**

Two main challenges with SC decoding:

1. Low throughput due to sequential nature
2. Mediocre error-correcting performance due to error propagation

Successive Cancellation List Decoding

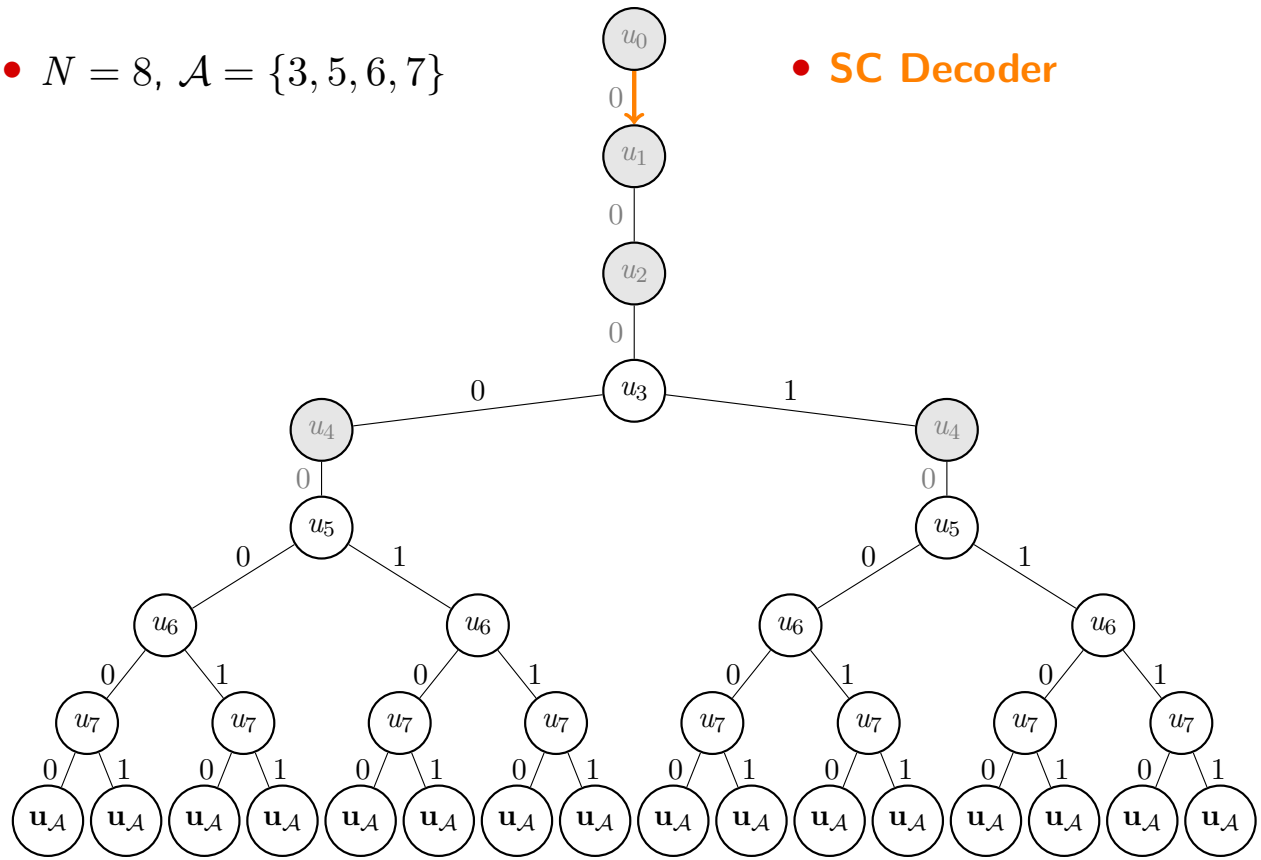
- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

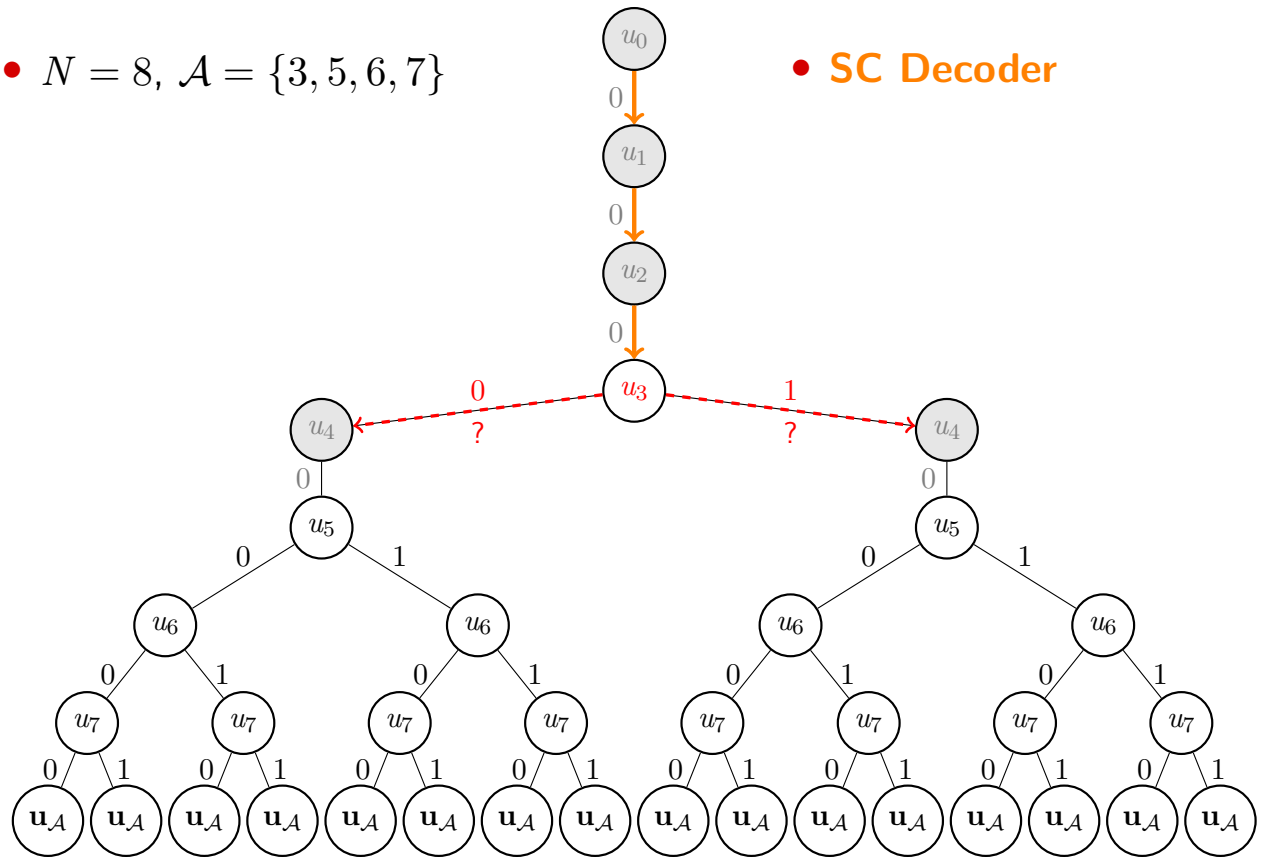


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

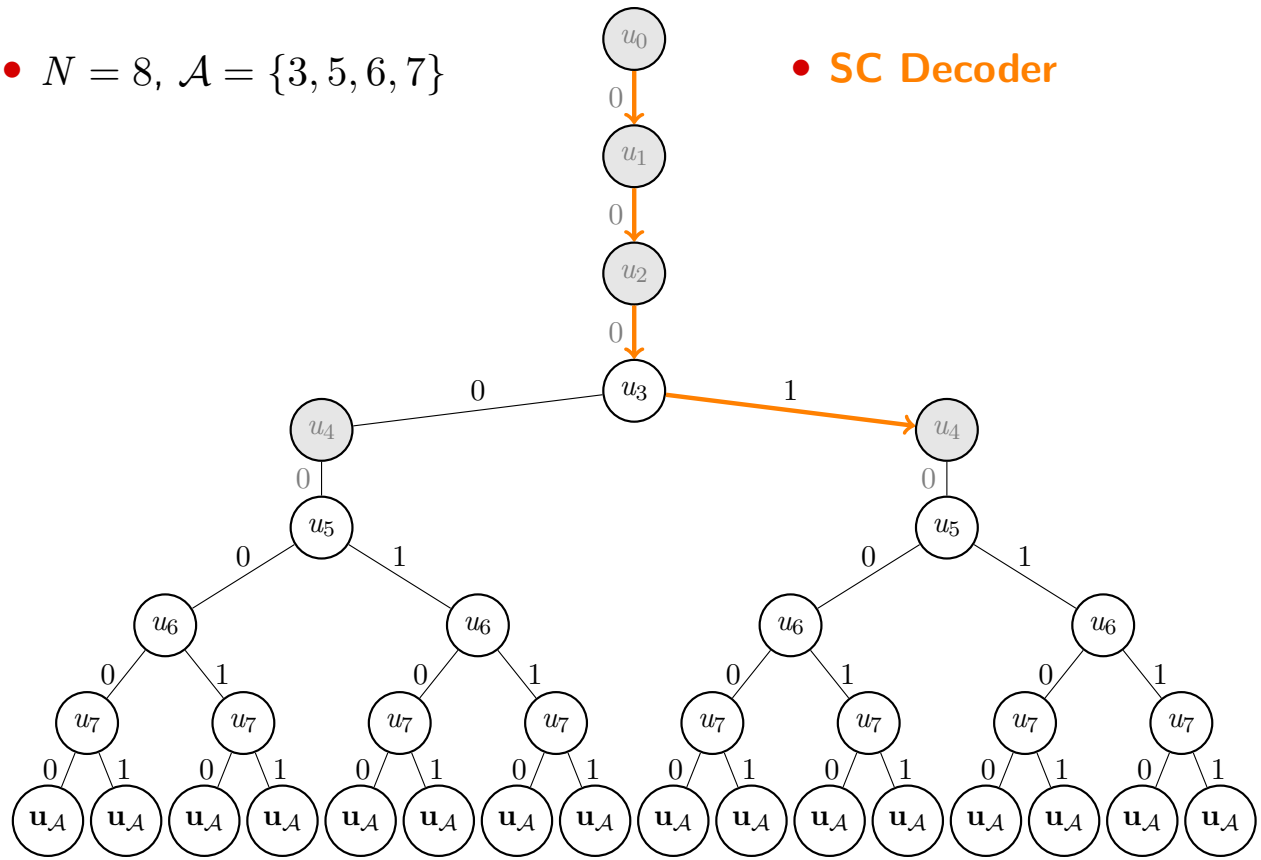


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

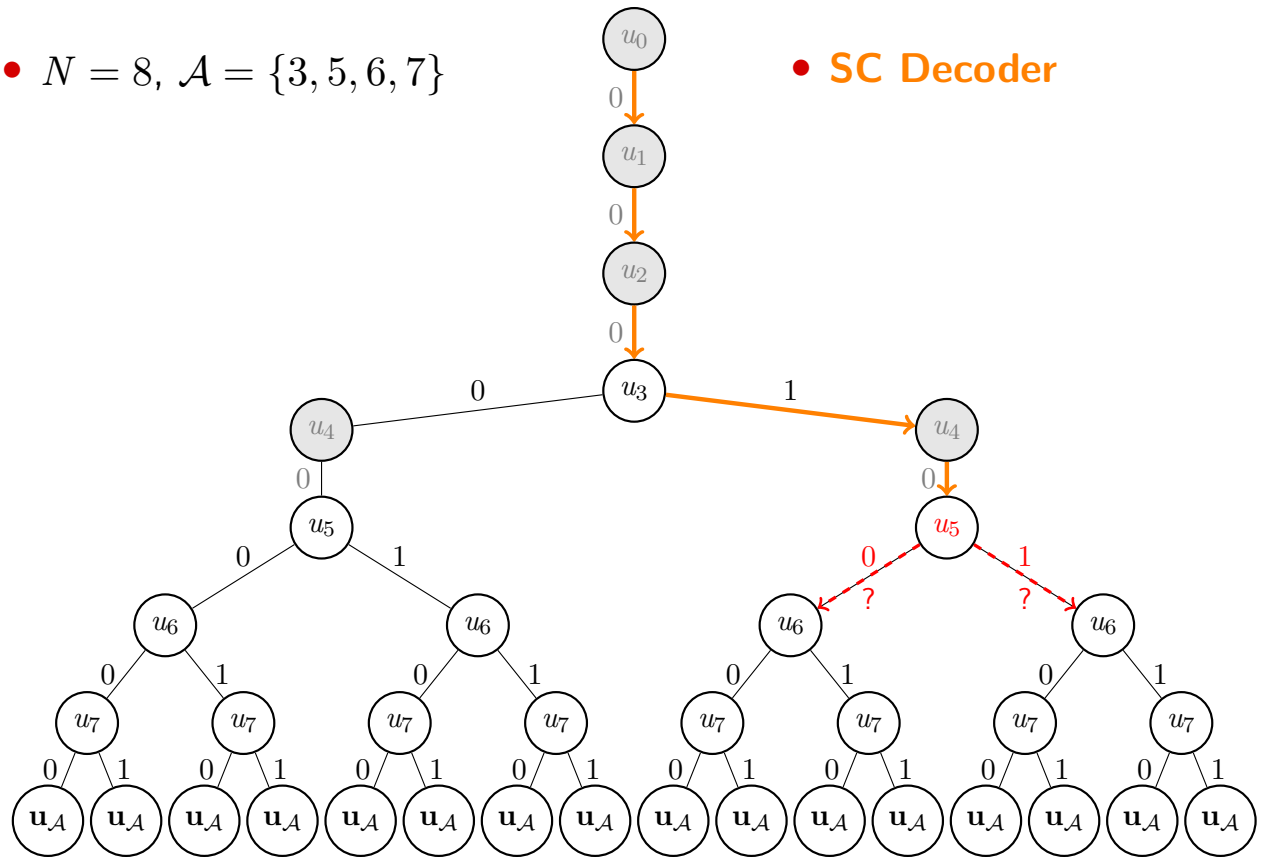


[1] I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

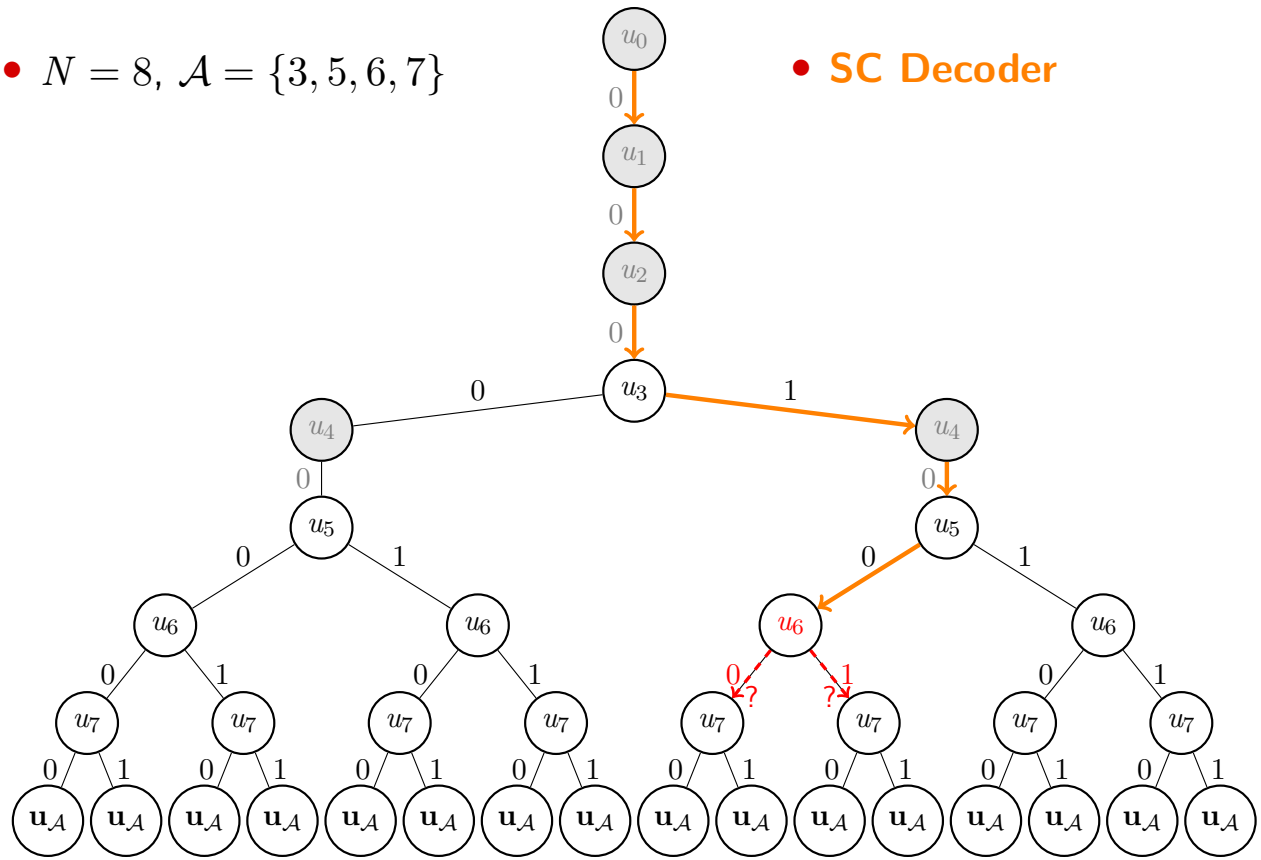


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

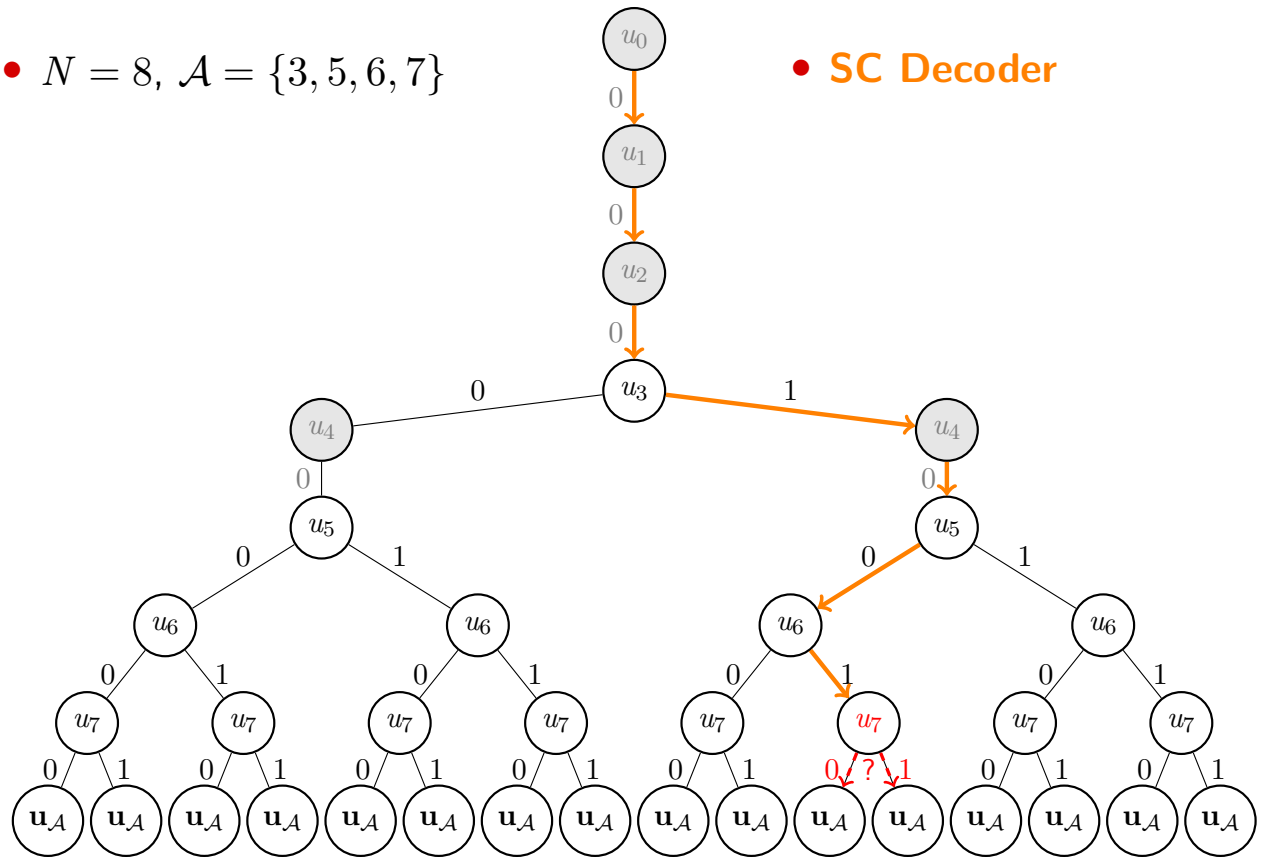


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

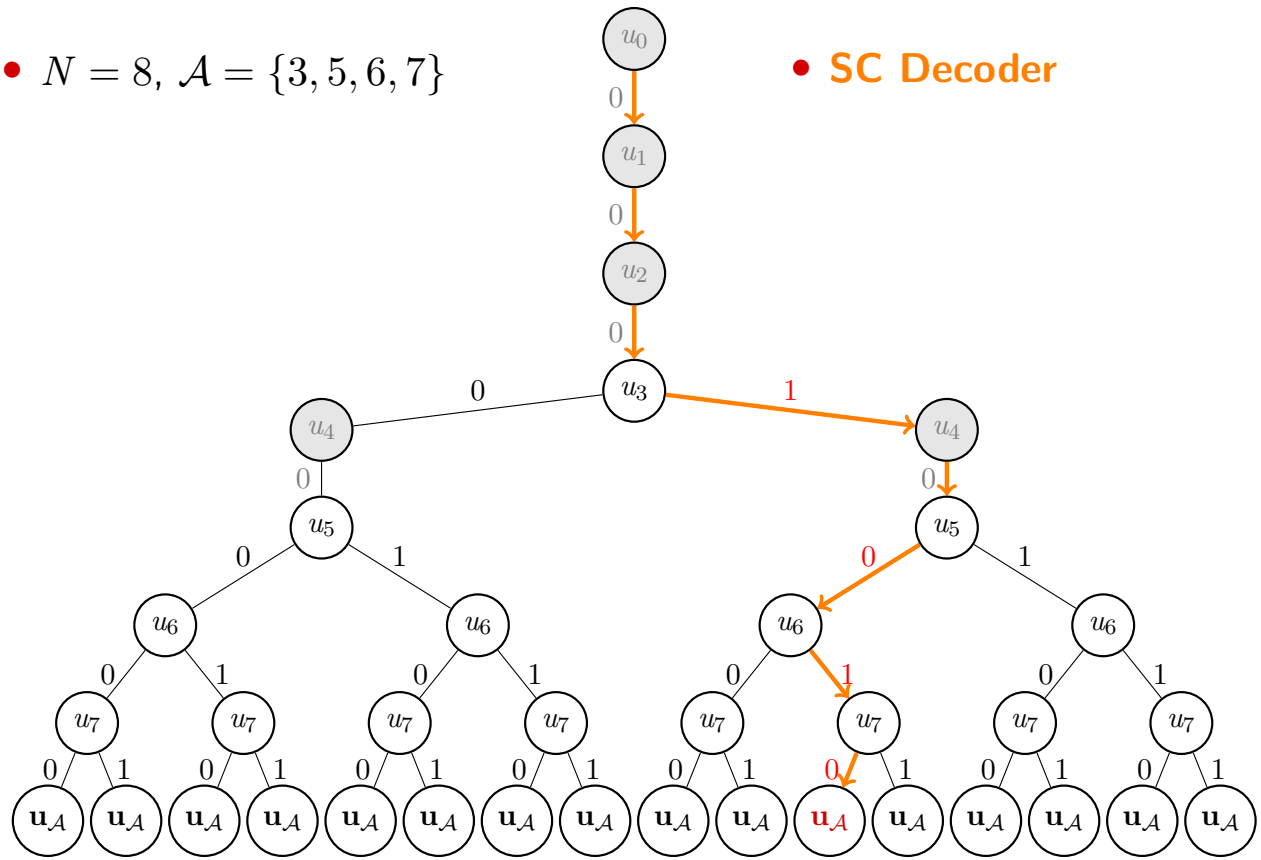


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$
- **SC Decoder**

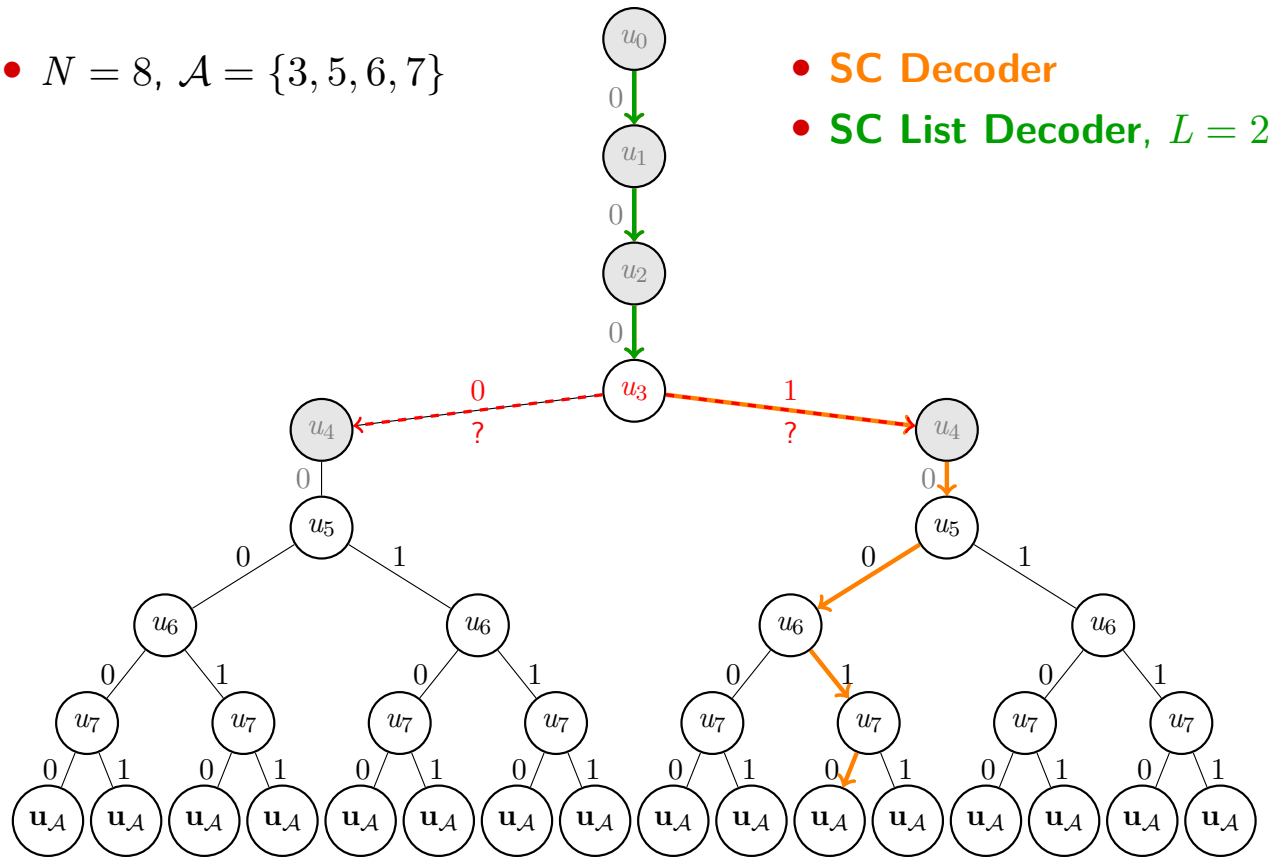


1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



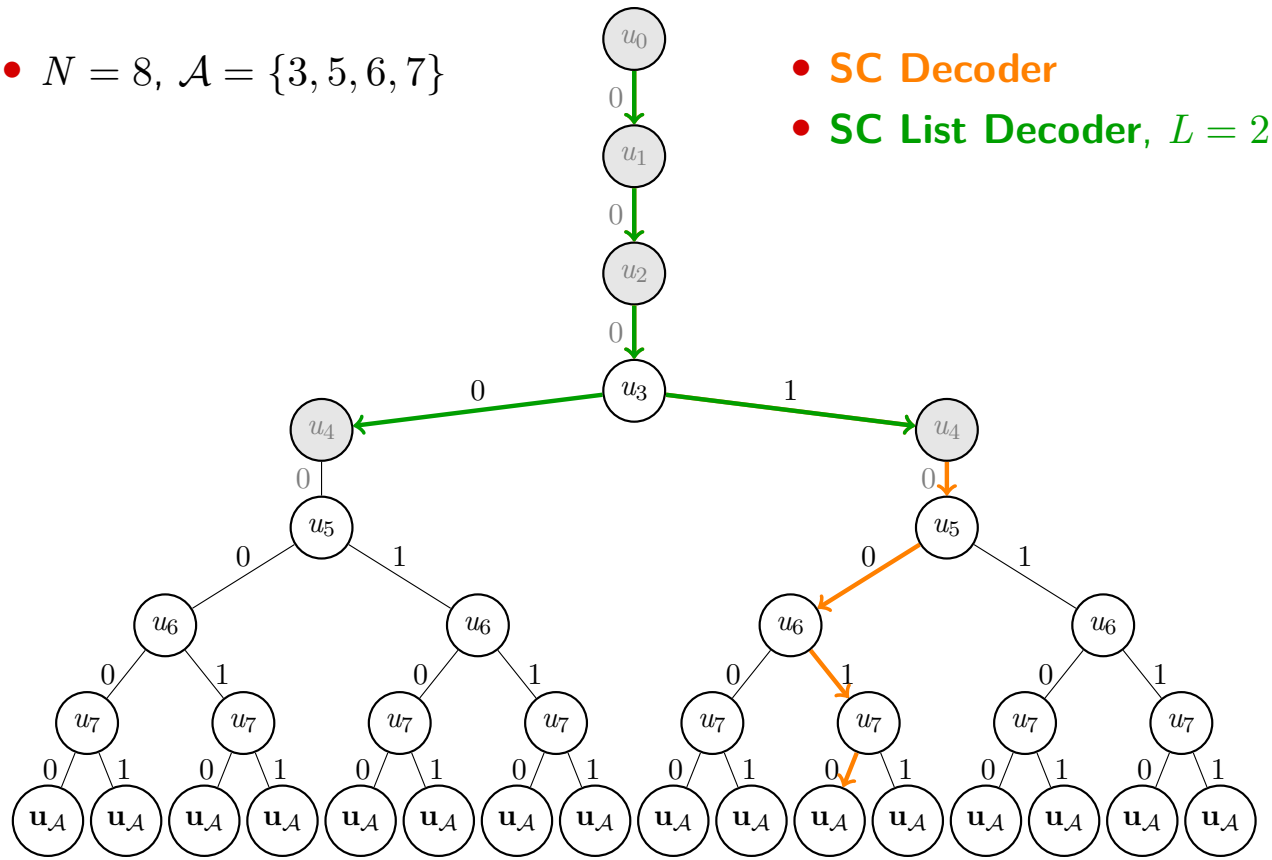
- **SC Decoder**
- **SC List Decoder**, $L = 2$

[1] I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



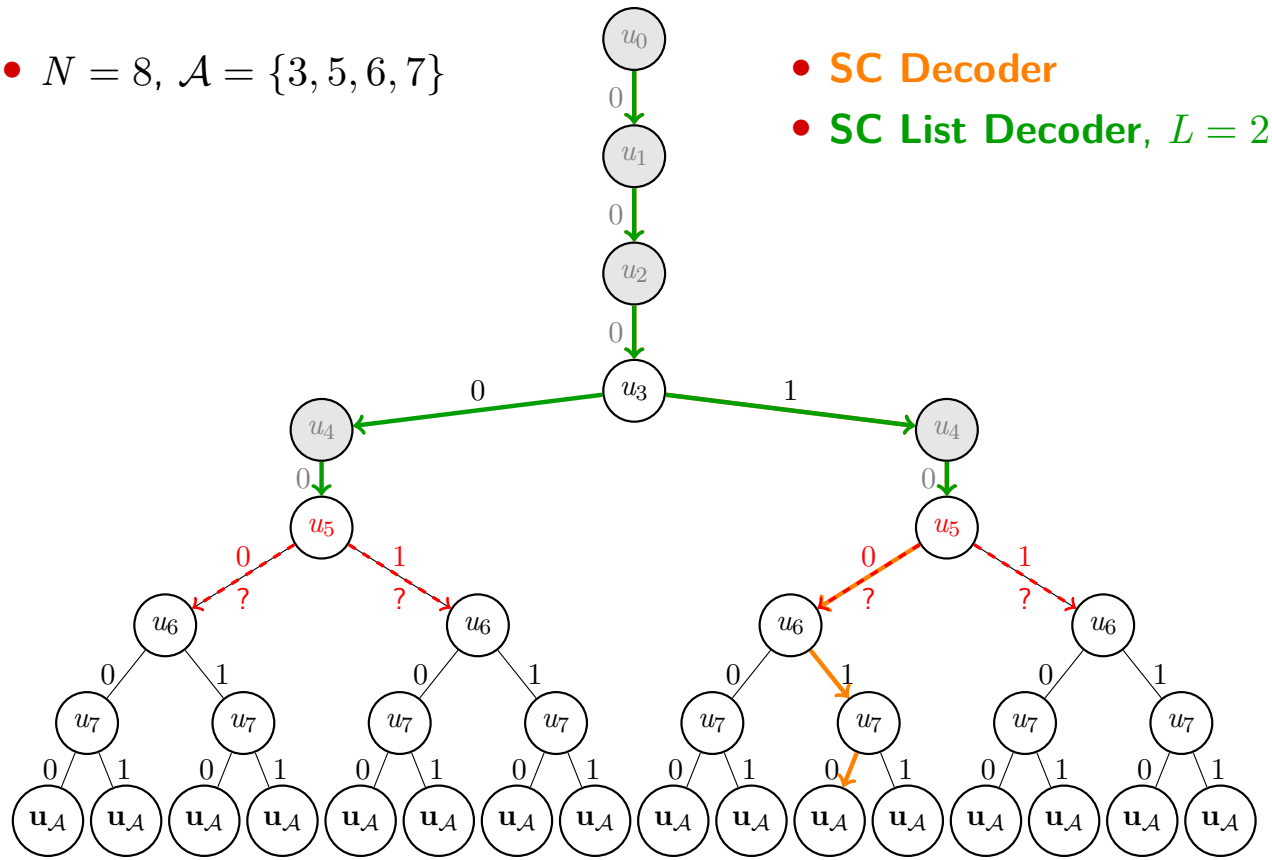
- **SC Decoder**
- **SC List Decoder**, $L = 2$

1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



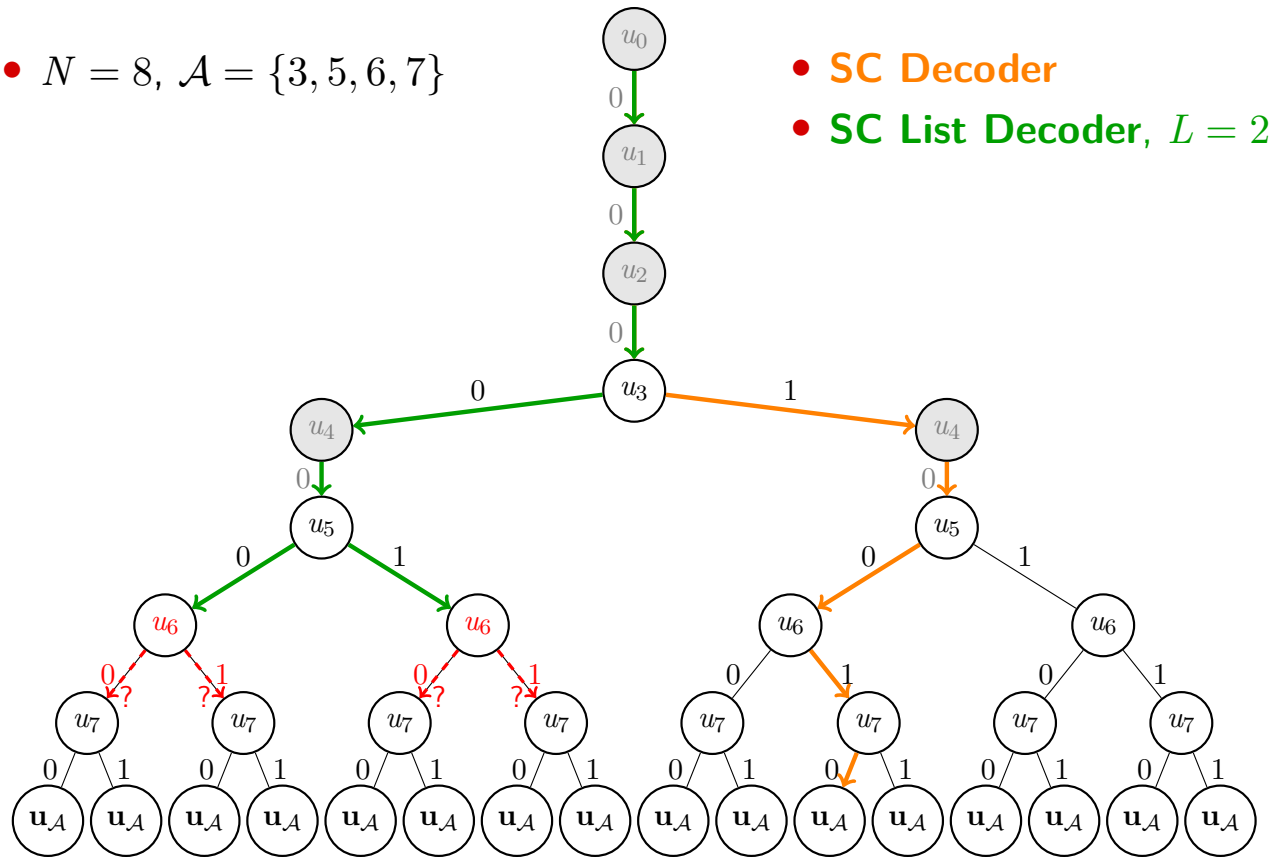
- **SC Decoder**
- **SC List Decoder**, $L = 2$

[1] I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



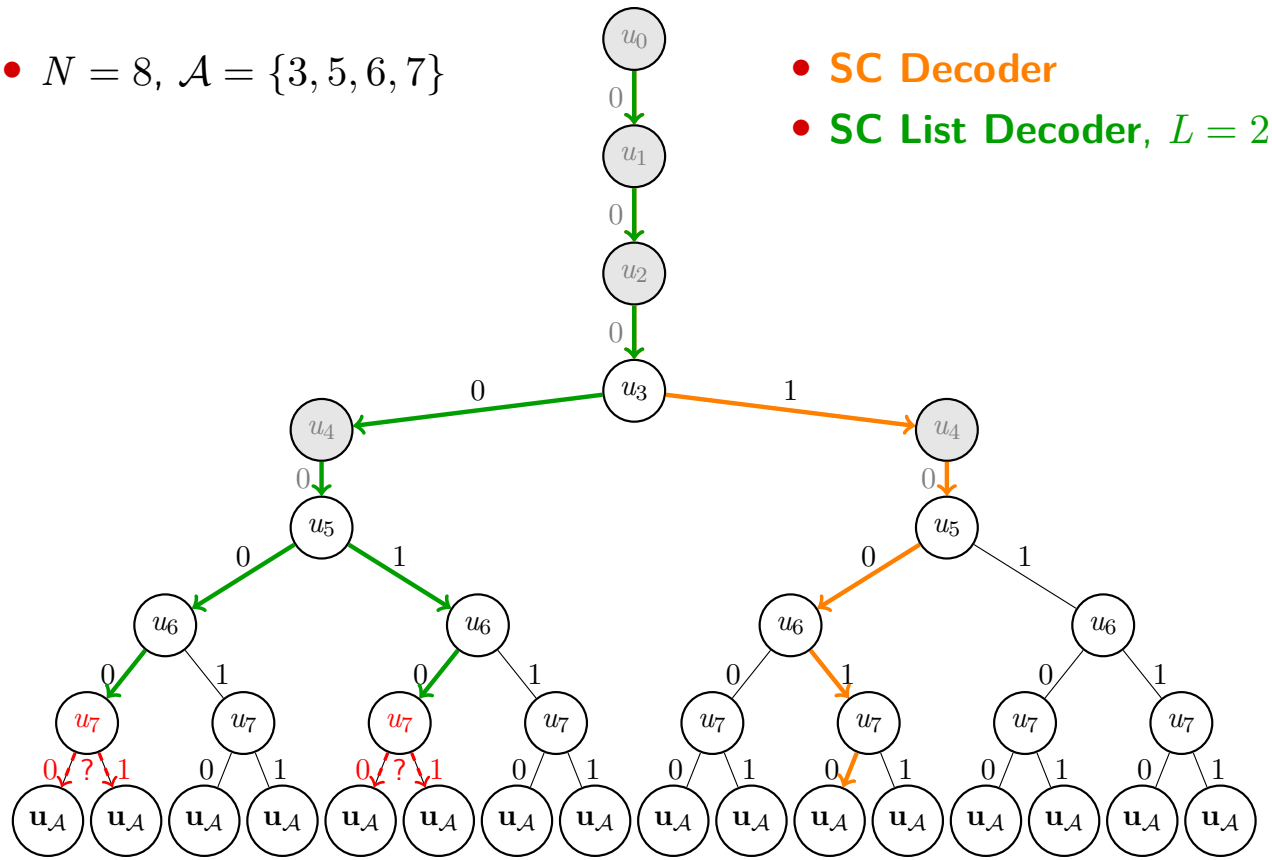
- **SC Decoder**
- **SC List Decoder**, $L = 2$

1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



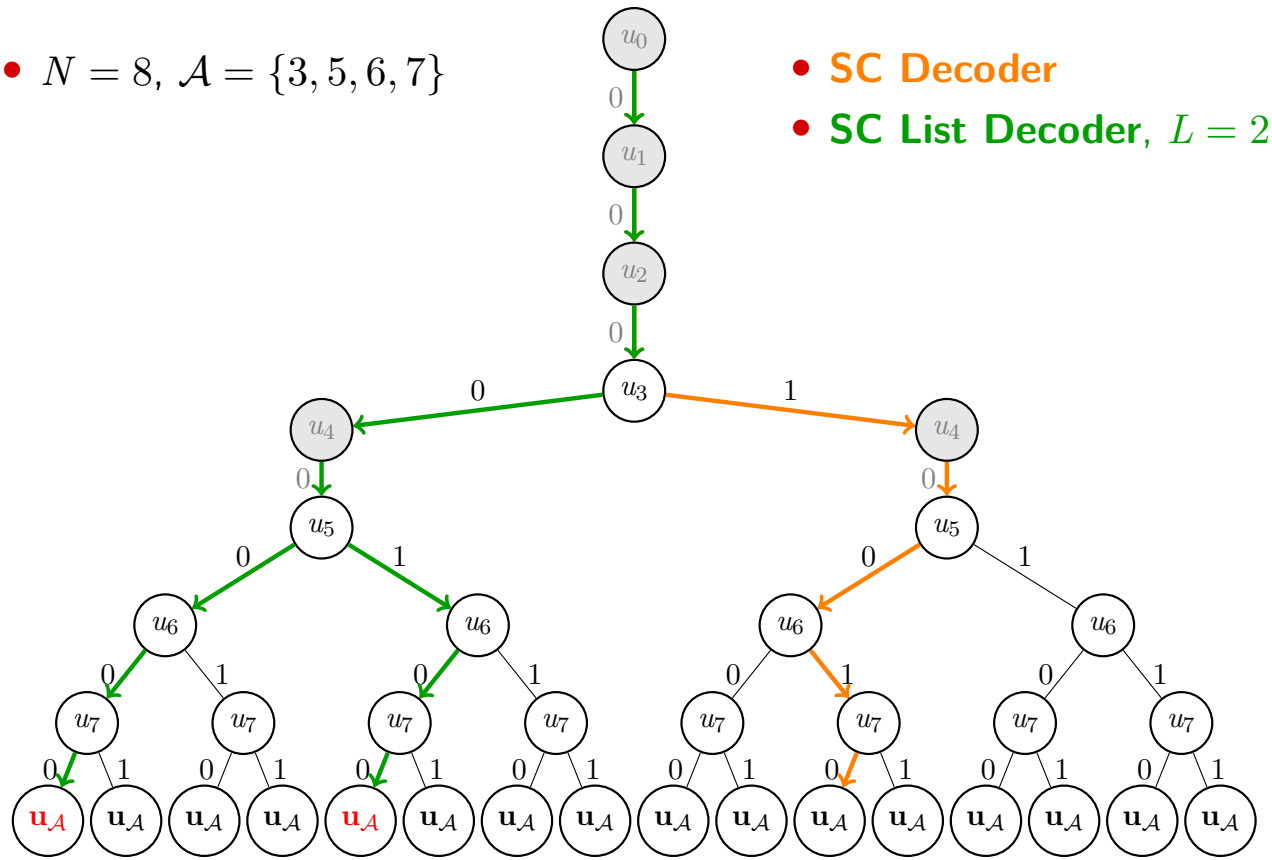
- **SC Decoder**
- **SC List Decoder**, $L = 2$

[1] I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



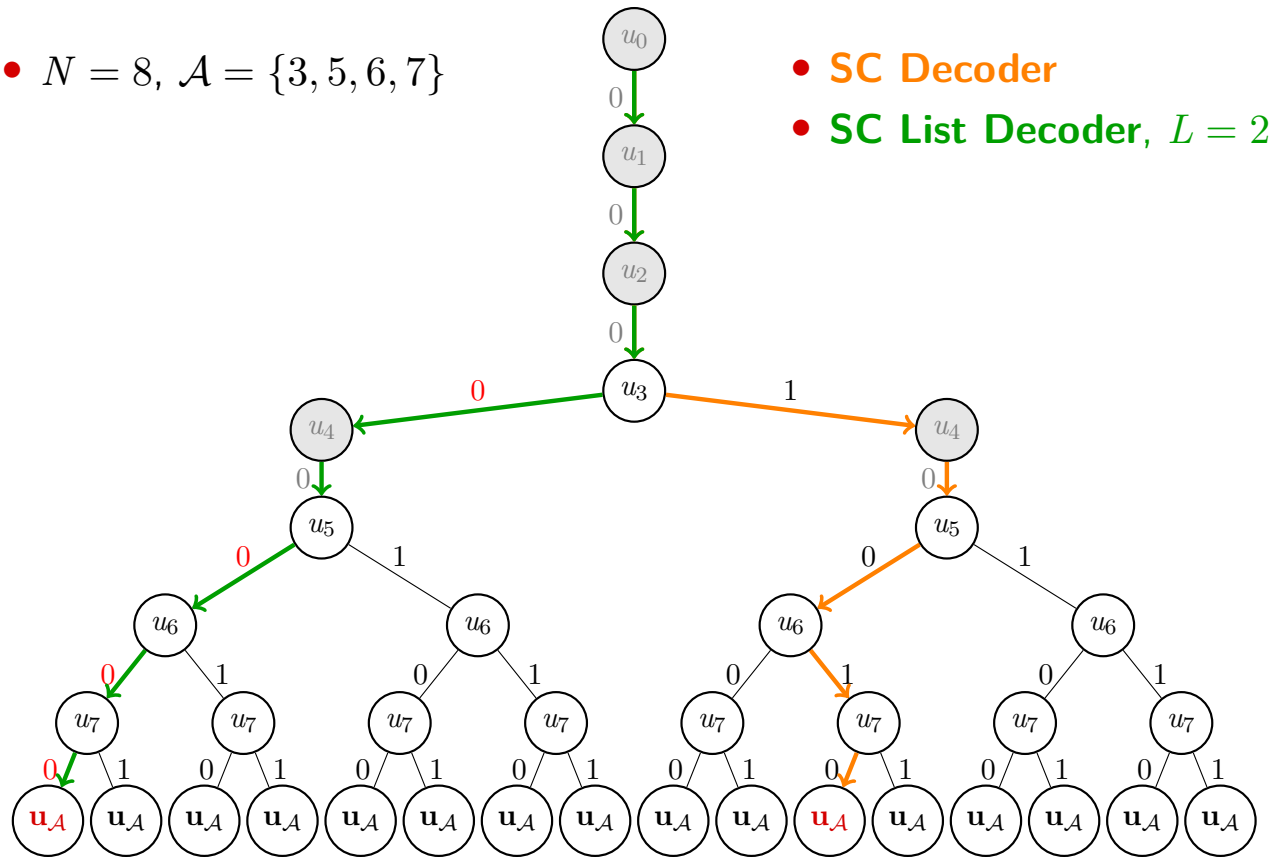
- **SC Decoder**
- **SC List Decoder**, $L = 2$

1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Successive Cancellation List Decoding

- **SC Decoding:** past errors can never be corrected
- **SCL Decoding:** up to L simultaneous paths on the decoding tree
  - **Time complexity:** $O(LN \log N)$, **Memory complexity:** $O(LN)$

- $N = 8$, $\mathcal{A} = \{3, 5, 6, 7\}$



- **SC Decoder**
- **SC List Decoder**, $L = 2$

1 I. Tal, A. Vardy, "List decoding of polar codes," IEEE Transactions on Information Theory, May 2015

Hardware Implementation of SCL Decoding

What changes w.r.t. SC decoding?

- Perform computations for L paths simultaneously
- Compute and sort path metrics to keep L best paths at each step

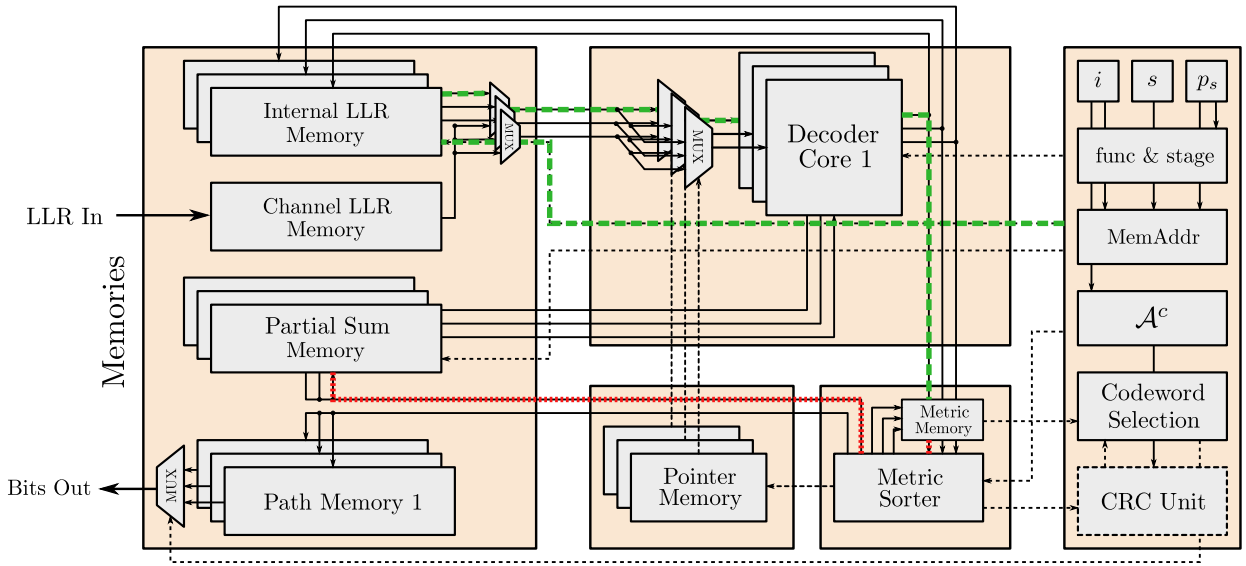
Hardware Implementation of SCL Decoding

What changes w.r.t. SC decoding?

- Perform computations for L paths simultaneously (highly parallelizable!)
- Compute and sort path metrics to keep L best paths at each step

What we need:

1. L decoder cores
2. L SC memories
3. A path metric sorter



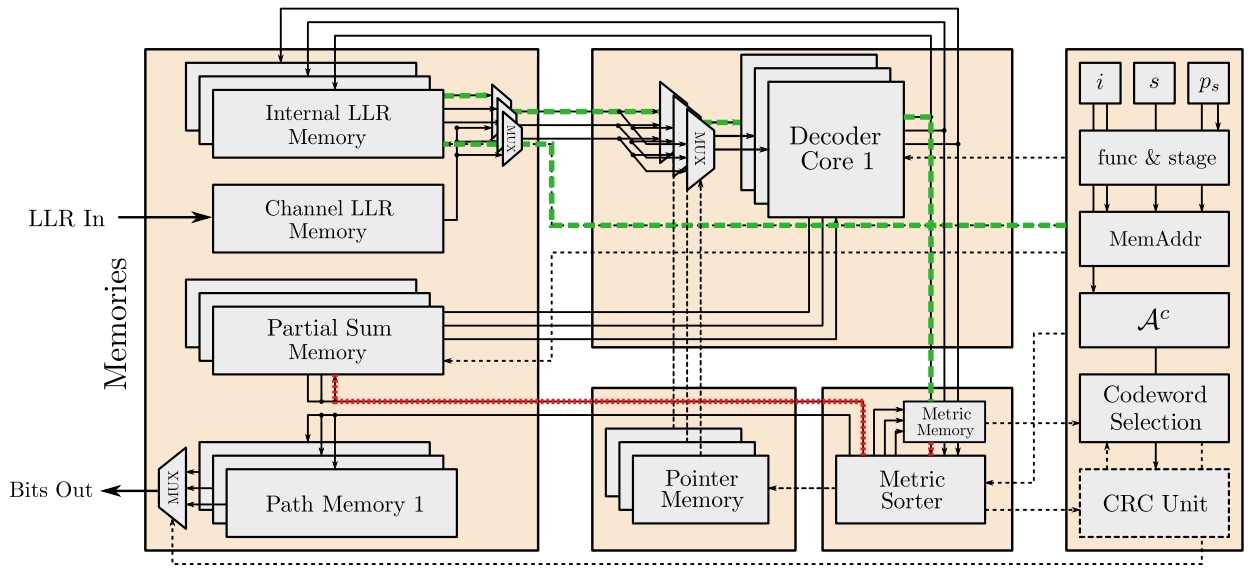
Hardware Implementation of SCL Decoding

What changes w.r.t. SC decoding?

- Perform computations for L paths simultaneously
- Compute and sort path metrics to keep L best paths at each step

What we need:

1. L decoder cores
2. L SC memories
3. A path metric sorter



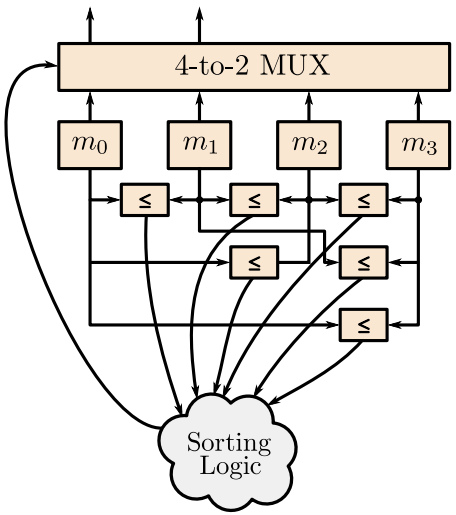
We proved an arithmetic re-formulation of SCL decoding that makes the hardware implementation **up to 67% more hardware-efficient**!

1. A. Balatsoukas-Stimming, M. Bastani Parizi, A. Burg, "LLR-based successive cancellation list decoding of polar codes," IEEE Transactions on Signal Processing, Oct. 2015

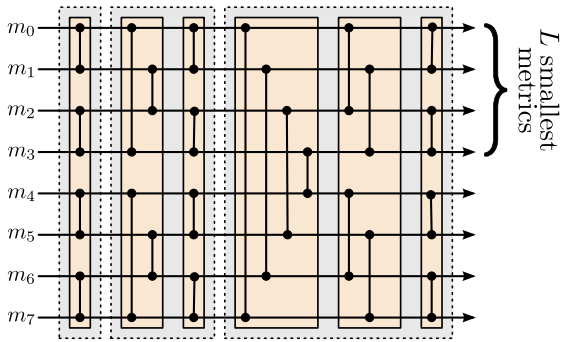
Optimized Metric Sorting for SCL Decoding

- Metric sorter lies on the **critical path** of SCL decoders

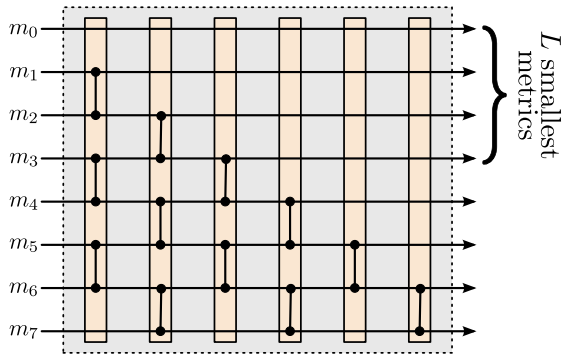
Radix-$2L$ Sorter



Bitonic Sorter



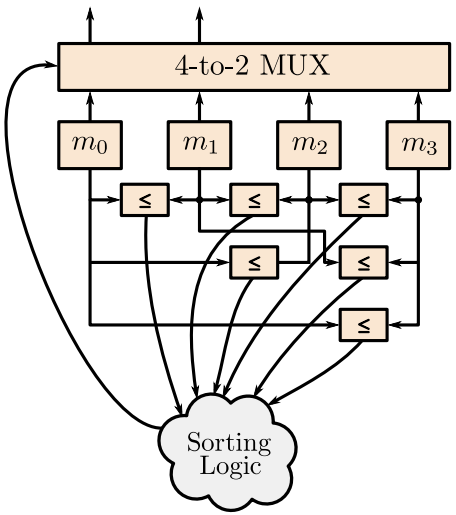
Bubble Sorter



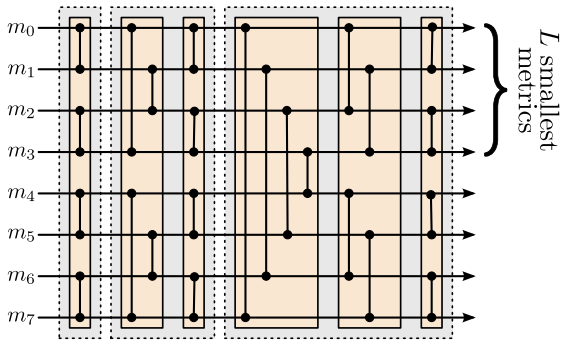
Optimized Metric Sorting for SCL Decoding

- Metric sorter lies on the **critical path** of SCL decoders
- Exploit reformulated metric properties to simplify the sorter:
  1. When forking, the L new path metrics are augmented versions of the old L ones
  2. Just need the L best among $2L$, no need for the L best to be sorted

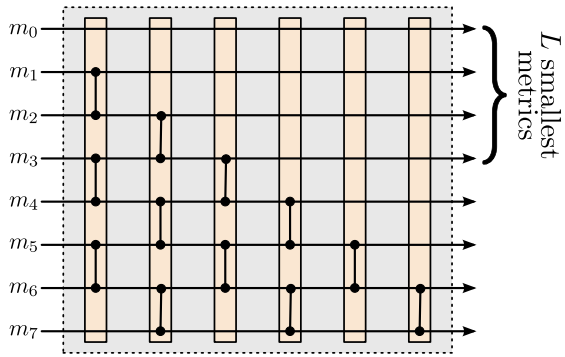
Radix-$2L$ Sorter



Bitonic Sorter



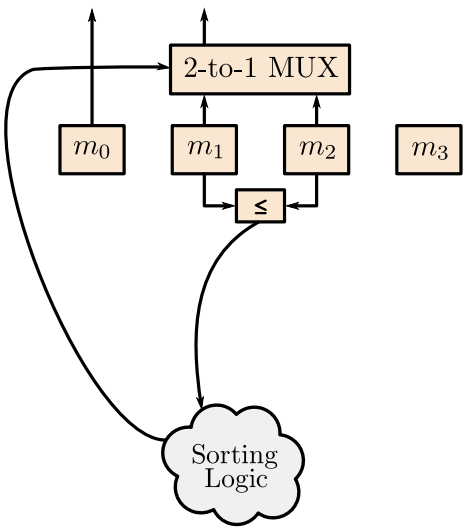
Bubble Sorter



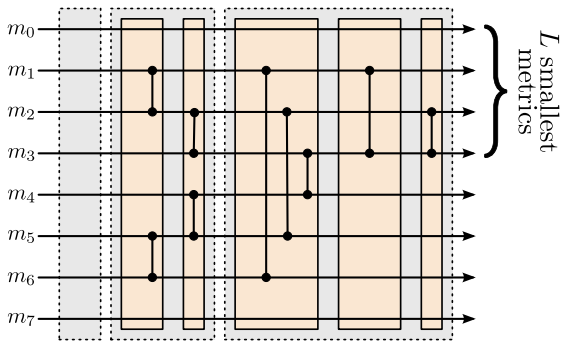
Optimized Metric Sorting for SCL Decoding

- Metric sorter lies on the **critical path** of SCL decoders
- Exploit reformulated metric properties to simplify the sorter:
  1. When forking, the L new path metrics are augmented versions of the old L ones
  2. Just need the L best among $2L$, no need for the L best to be sorted

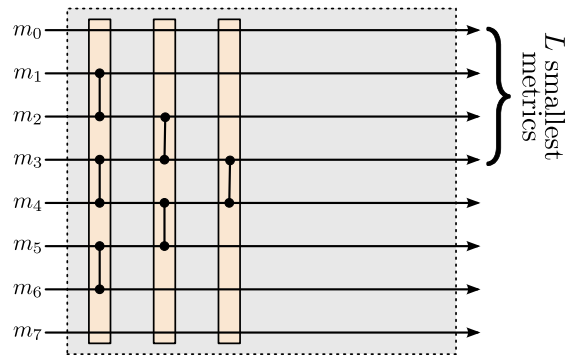
Radix-$2L$ Sorter



Bitonic Sorter



Bubble Sorter



Significant improvement in the **area and operating frequency** of the decoder!

1. A. Balatsoukas-Stimming, M. Bastani Parizi, A. Burg, "On metric sorting for successive cancellation list decoding of polar codes," IEEE International Symposium on Circuits and Systems, May 2015

Successive Cancellation Flip Decoding

SCL Decoding

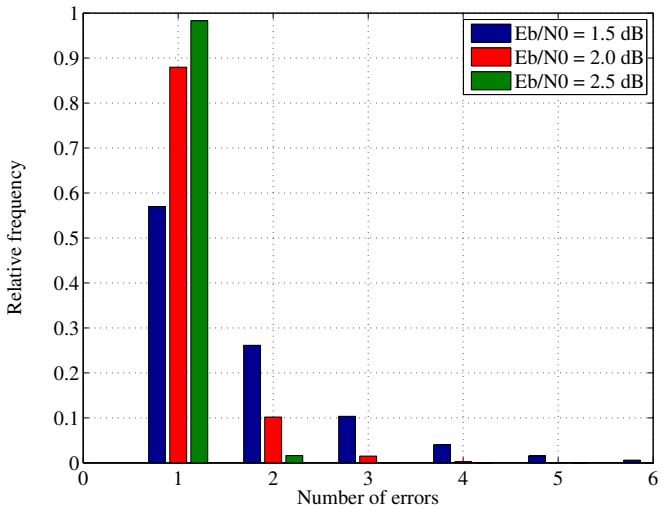
Most of the computations and memory are **wasted most of the time**!

Successive Cancellation Flip Decoding

SCL Decoding

Most of the computations and memory are **wasted most of the time**!

- **Observation:** In SC, most erroneous frames are due to **one wrong decision**

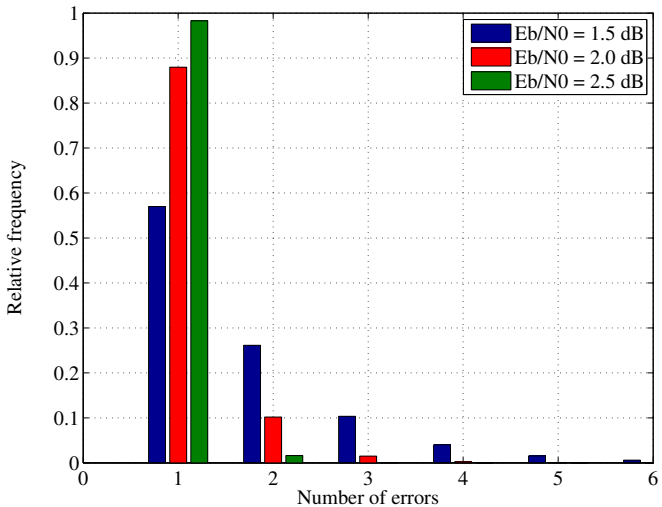


Successive Cancellation Flip Decoding

SCL Decoding

Most of the computations and memory are **wasted most of the time**!

- **Observation:** In SC, most erroneous frames are due to **one wrong decision**



- **Successive Cancellation Flip** (SCF) decoding algorithm:
  1. Perform SC decoding and track T most unreliable decisions
  2. Use a CRC to identify erroneous decoding
  3. Re-run SC up to T times, each time flipping the most unreliable decision

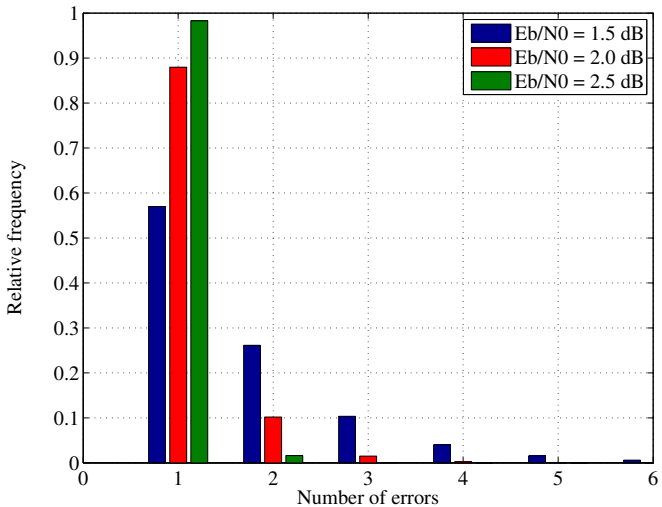
[1] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in Asilomar Conf. on Signals, Systems, and Computers, Nov. 2014

Successive Cancellation Flip Decoding

SCL Decoding

Most of the computations and memory are **wasted most of the time**!

- **Observation:** In SC, most erroneous frames are due to **one wrong decision**



- **Successive Cancellation Flip** (SCF) decoding algorithm:
  1. Perform SC decoding and track T most unreliable decisions
  2. Use a CRC to identify erroneous decoding
  3. Re-run SC up to T times, each time flipping the most unreliable decision

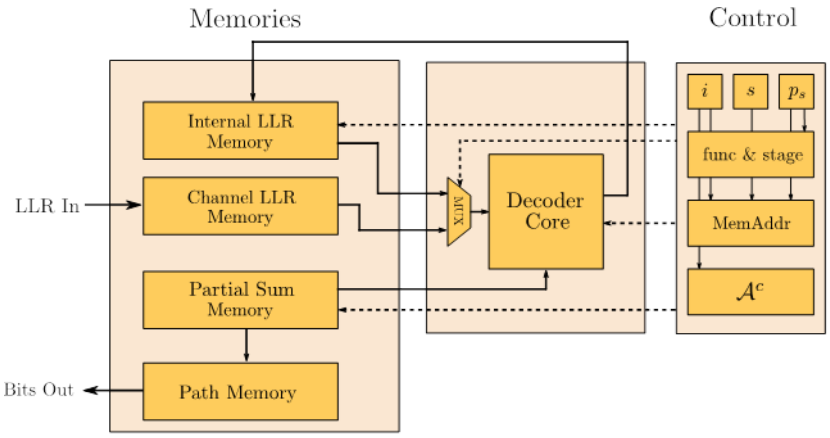
Error-correcting performance in-between SC and SCL, but:

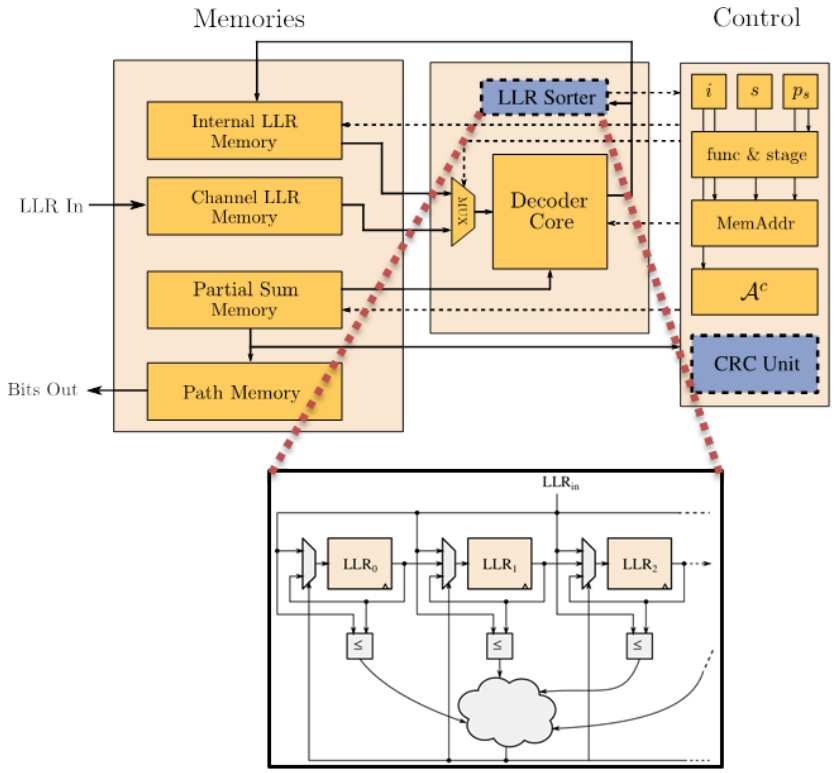
- **Memory complexity** of SC decoding
- (Average) **time complexity** of SC decoding (at high SNR)

1 O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in Asilomar Conf. on Signals, Systems, and Computers, Nov. 2014

Hardware Implementation of SCF Decoding

- **Simple:** Add an insertion sorter and a CRC unit to an SC decoder

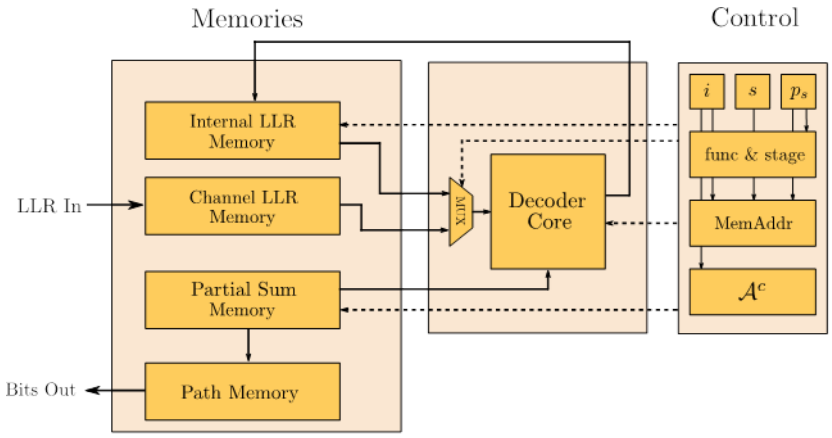


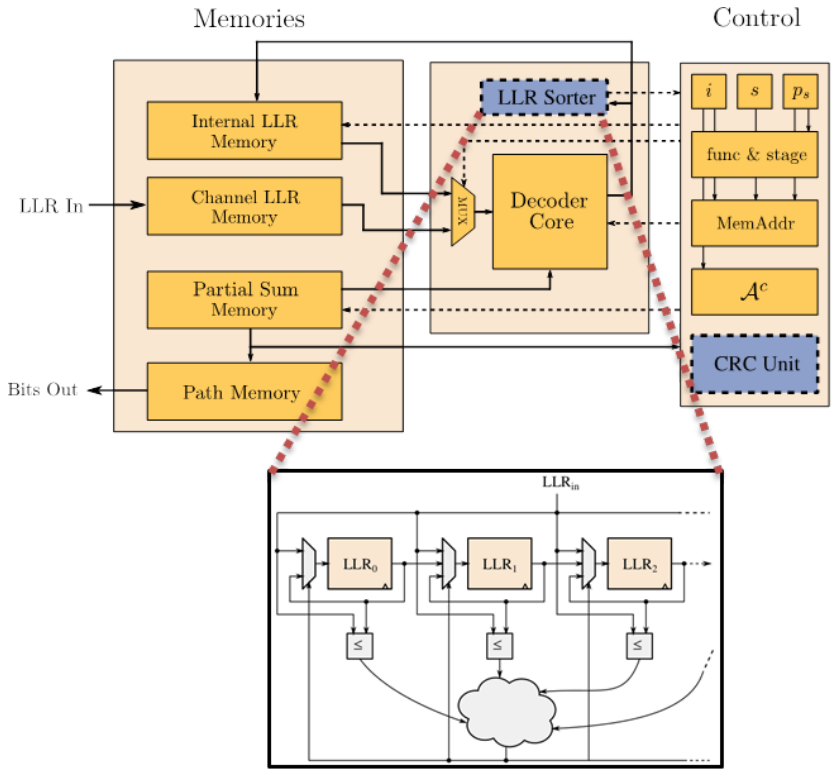


Negligible area overhead!
No impact on latency!

Hardware Implementation of SCF Decoding

- **Simple:** Add an insertion sorter and a CRC unit to an SC decoder



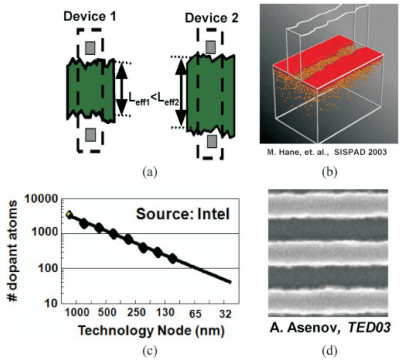


Negligible area overhead!
No impact on latency!

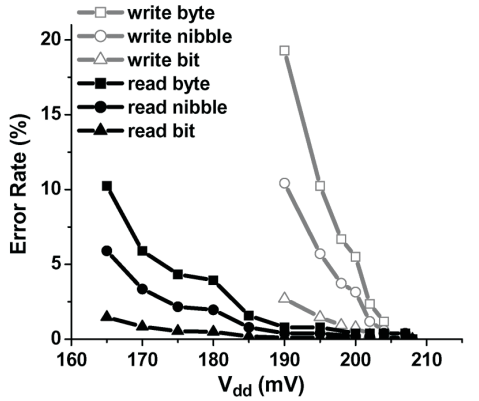
Our proposed SCF decoding algorithm is being considered by the 3GPP as an ultra-low power option for massive machine type communications for IoT in 5G

1 Huawei and HiSilicon, "Computational and implementation complexity of channel coding schemes," 3GPP TSG RAN WG1 Meeting #86, Tech. Rep. R1-167213, Aug. 2016

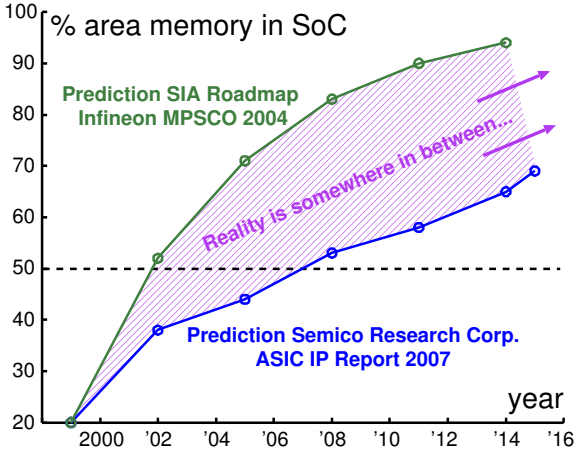
VLSI Circuits are Becoming Unreliable



Process variation
[K. Roy et al., 2010]

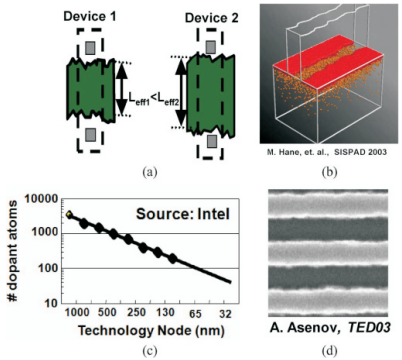


Soft-errors [B. Zhai et al., 2008]

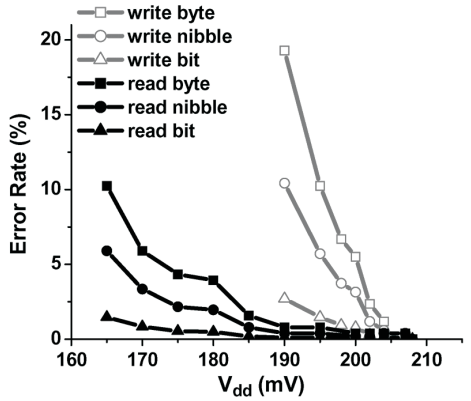


- Devices suffer from **defects** due to **parameter variations** and "soft errors"
- These issues **compromise reliable operation** and **prevent effective power-reduction techniques** (e.g., voltage scaling)

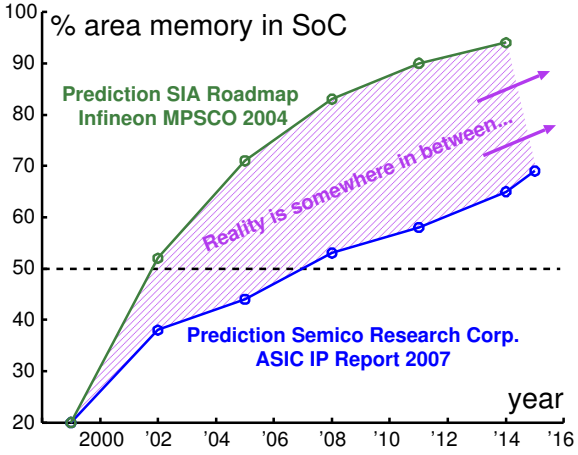
VLSI Circuits are Becoming Unreliable



Process variation
[K. Roy et al., 2010]

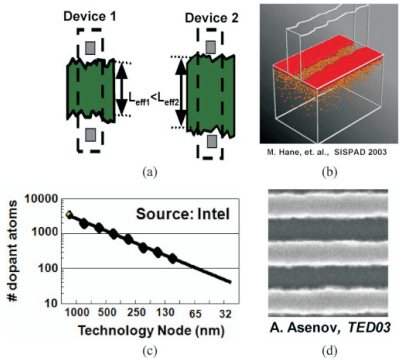


Soft-errors [B. Zhai et al., 2008]

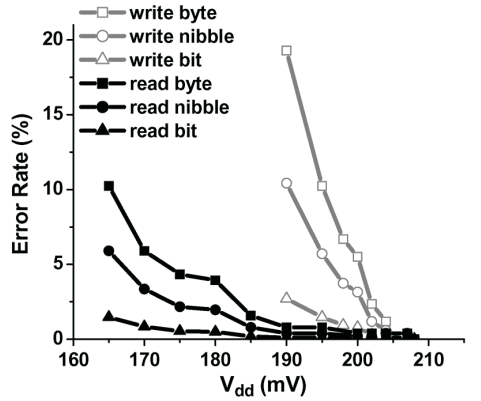


- Devices suffer from **defects** due to **parameter variations** and "soft errors"
- These issues **compromise reliable operation** and **prevent effective power-reduction techniques** (e.g., voltage scaling)
- Memories are **particularly sensitive to process variations** and **dominate area and power consumption** of modern systems-on-chip

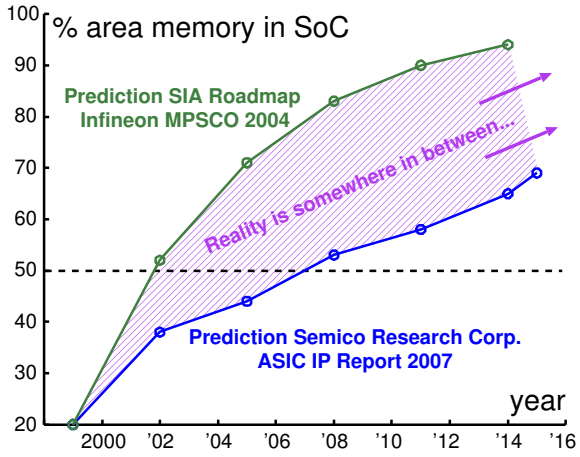
VLSI Circuits are Becoming Unreliable



Process variation
[K. Roy et al., 2010]



Soft-errors [B. Zhai et al., 2008]



- Devices suffer from **defects** due to **parameter variations** and "soft errors"
- These issues **compromise reliable operation** and **prevent effective power-reduction techniques** (e.g., voltage scaling)
- Memories are **particularly sensitive to process variations** and **dominate area and power consumption** of modern systems-on-chip

What to do?

- Hardware protection to avoid errors is costly in terms of area and power
- Discarding faulty chips can decrease the yield dramatically

Approximate Computing

- **Fortunately,** many applications deal with data that is already **stochastic** and/or degrade **gracefully** when data is corrupted

Approximate Computing

- **Fortunately,** many applications deal with data that is already **stochastic** and/or degrade **gracefully** when data is corrupted

Example: error-correcting codes

- Faulty successive cancellation decoding of polar codes
- Faulty min-sum decoding of LDPC codes
- Faulty windowed min-sum decoding of spatially-coupled LDPC codes
- Throughput-oriented construction of polar codes

1. A. Balatsoukas-Stimming and A. Burg, "Faulty successive cancellation decoding of polar codes for the binary erasure channel," IEEE Transactions on Communications, Jun. 2018
2. A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," IEEE Communications Letters, May 2014
3. J. Mu, A. Vosoughi, J. Andrade, A. Balatsoukas-Stimming, G. Karakonstantis, A. Burg, G. Falcao, V. Silva, and J. R. Cavallaro, "The impact of faulty memory bit cells on the decoding of spatially-coupled LDPC codes," in Asilomar Conference on Signals, Systems, and Computers, Nov. 2015
4. A. Balatsoukas-Stimming, G. Karakonstantis, A. Burg, "Enabling complexity-performance trade-offs for successive cancellation decoding of polar codes," International Symposium on Information Theory (ISIT), Jun. 2014

Approximate Computing

- **Fortunately,** many applications deal with data that is already **stochastic** and/or degrade **gracefully** when data is corrupted

Example: error-correcting codes

- Faulty successive cancellation decoding of polar codes
- Faulty min-sum decoding of LDPC codes
- Faulty windowed min-sum decoding of spatially-coupled LDPC codes
- Throughput-oriented construction of polar codes

1. A. Balatsoukas-Stimming and A. Burg, "Faulty successive cancellation decoding of polar codes for the binary erasure channel," IEEE Transactions on Communications, Jun. 2018
2. A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," IEEE Communications Letters, May 2014
3. J. Mu, A. Vosoughi, J. Andrade, A. Balatsoukas-Stimming, G. Karakonstantis, A. Burg, G. Falcao, V. Silva, and J. R. Cavallaro, "The impact of faulty memory bit cells on the decoding of spatially-coupled LDPC codes," in Asilomar Conference on Signals, Systems, and Computers, Nov. 2015
4. A. Balatsoukas-Stimming, G. Karakonstantis, A. Burg, "Enabling complexity-performance trade-offs for successive cancellation decoding of polar codes," International Symposium on Information Theory (ISIT), Jun. 2014

Faulty Polar Decoding over the Binary Erasure Channel

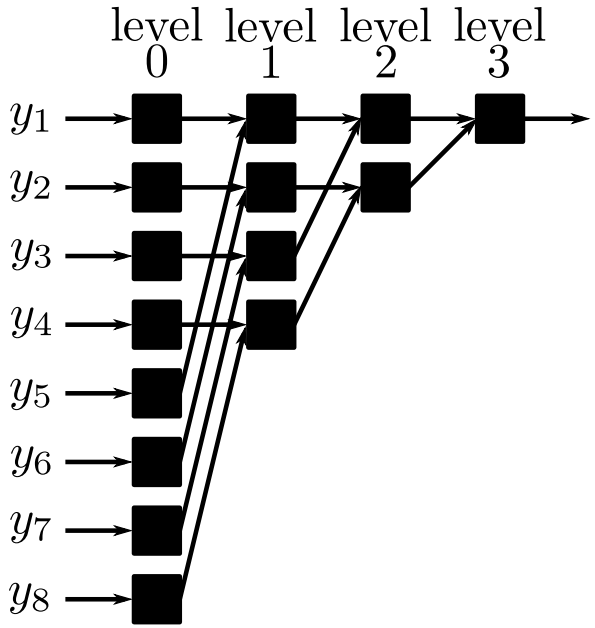
- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p

Faulty Polar Decoding over the Binary Erasure Channel

- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p

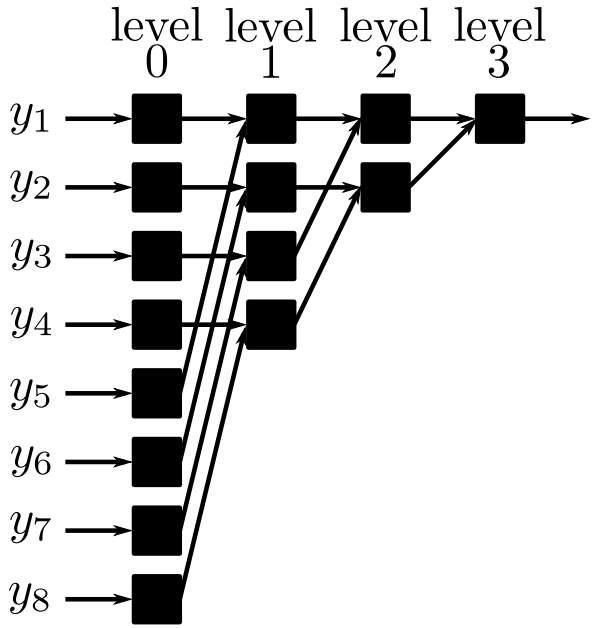
Density Evolution

- Erasure pr. f **nodes**: $T^f(\epsilon) = 2\epsilon - \epsilon^2$
- Erasure pr. g **nodes**: $T^g(\epsilon) = \epsilon^2$



Faulty Polar Decoding over the Binary Erasure Channel

- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p



Density Evolution

- Erasure pr. f **nodes**: $T^f(\epsilon) = 2\epsilon - \epsilon^2$
- Erasure pr. g **nodes**: $T^g(\epsilon) = \epsilon^2$

- What happens when **memory is unreliable**?

Fault model

Additional erasures in **non-erased** outputs w.p. δ

Faulty Polar Decoding over the Binary Erasure Channel

- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p

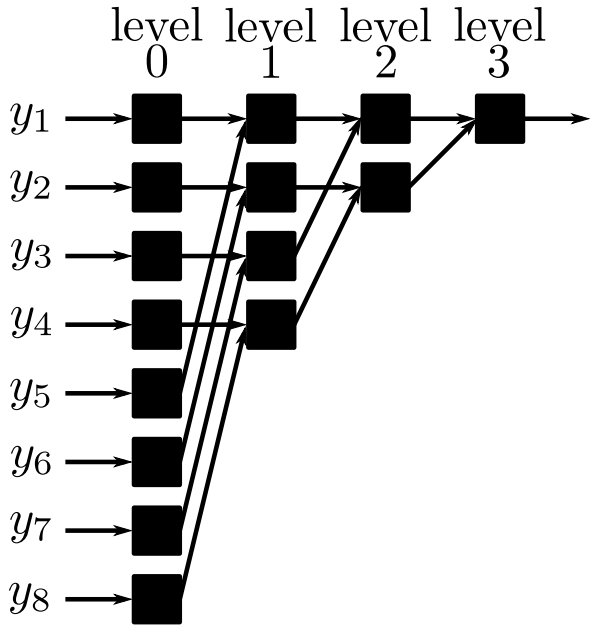
Density Evolution

- Erasure pr. f **nodes**: $T^f(\epsilon) = 2\epsilon - \epsilon^2 + (1-2\epsilon+\epsilon^2)\delta$
- Erasure pr. g **nodes**: $T^g(\epsilon) = \epsilon^2 + (1-\epsilon^2)\delta$

- What happens when **memory is unreliable**?

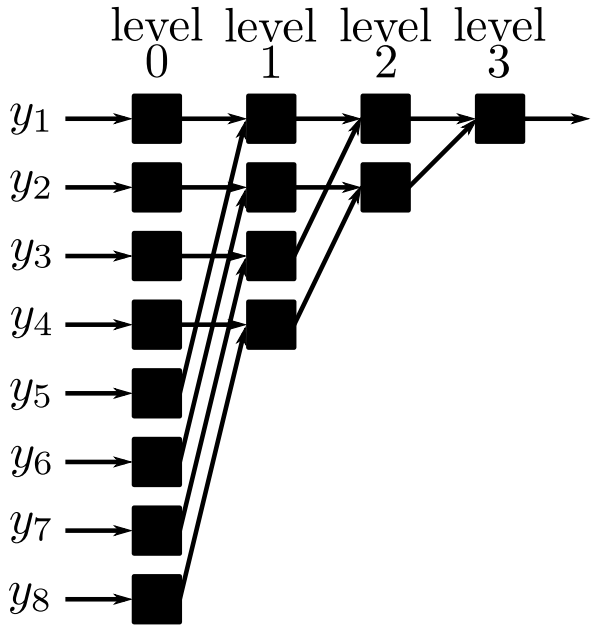
Fault model

Additional erasures in **non-erased** outputs w.p. δ



Faulty Polar Decoding over the Binary Erasure Channel

- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p



Density Evolution

- Erasure pr. f **nodes**: $T^f(\epsilon) = 2\epsilon - \epsilon^2 + (1-2\epsilon+\epsilon^2)\delta$
- Erasure pr. g **nodes**: $T^g(\epsilon) = \epsilon^2 + (1-\epsilon^2)\delta$

- What happens when **memory is unreliable**?

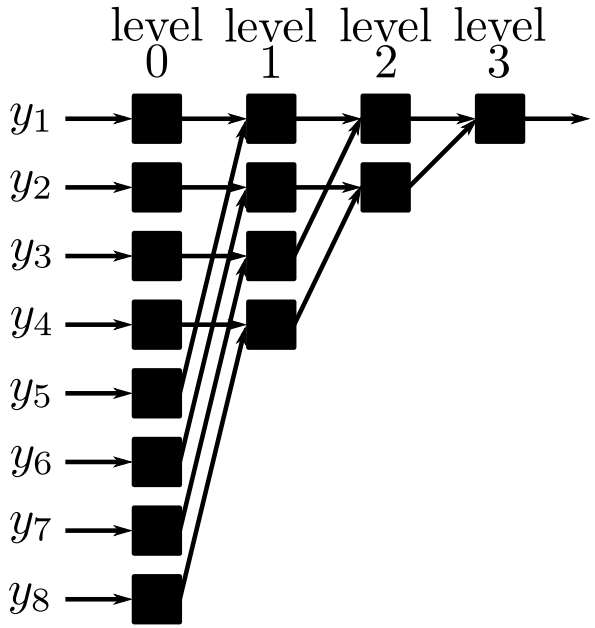
Fault model

Additional erasures in **non-erased** outputs w.p. δ

- Polarization process: $\epsilon_j = \begin{cases} T^f(\epsilon_{j-1}) & \text{w.p. } 1/2, \\ T^g(\epsilon_{j-1}) & \text{w.p. } 1/2, \end{cases} \qquad \epsilon_0 = p.$

Faulty Polar Decoding over the Binary Erasure Channel

- Binary Erasure Channel (BEC):
  - **Input**: 0 or 1
  - **Output**: input w.p. $(1-p)$, erasure w.p. p



Density Evolution

- Erasure pr. f **nodes**: $T^f(\epsilon) = 2\epsilon - \epsilon^2 + (1-2\epsilon+\epsilon^2)\delta$
- Erasure pr. g **nodes**: $T^g(\epsilon) = \epsilon^2 + (1-\epsilon^2)\delta$

- What happens when **memory is unreliable**?

Fault model

Additional erasures in **non-erased** outputs w.p. δ

- Polarization process: $\epsilon_j = \begin{cases} T^f(\epsilon_{j-1}) & \text{w.p. } 1/2, \\ T^g(\epsilon_{j-1}) & \text{w.p. } 1/2, \end{cases} \qquad \epsilon_0 = p.$

Theorem (**All channels** become asymptotically **useless**)

Under faulty SC decoding over the BEC, $\epsilon_j \xrightarrow{\text{a.s.}} 1$ *for any* $\delta > 0$.

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_{\mathsf{p}} = (n+1) - n_{\mathsf{u}}$ levels for n_{u} fixed: protected fraction is **constant**

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_{\mathsf{p}} = (n+1) - n_{\mathsf{u}}$ levels for n_{u} fixed: protected fraction is **constant**

Theorem (**Full reliability** by protecting a **constant fraction** of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_{\mathsf{p}} = (n+1) - n_{\mathsf{u}}$ levels for n_{u} fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

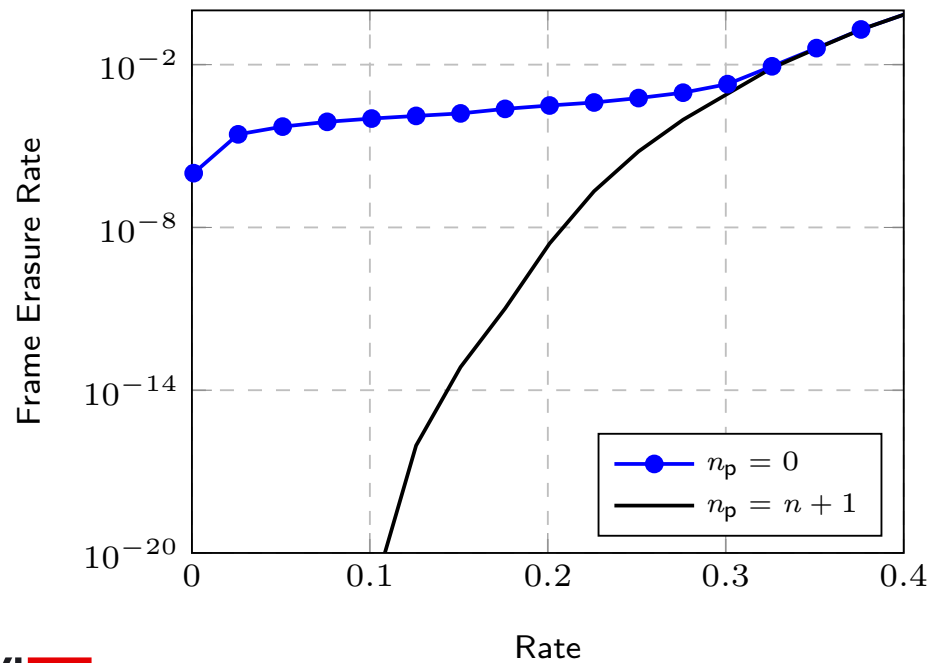
1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed $n_u < n+1$, ϵ_j **converges a.s.** to a random variable $\epsilon_\infty \in \{0,1\}$.*
2. ***Rate loss**: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$*



% of protected MEs ($n = 10$):

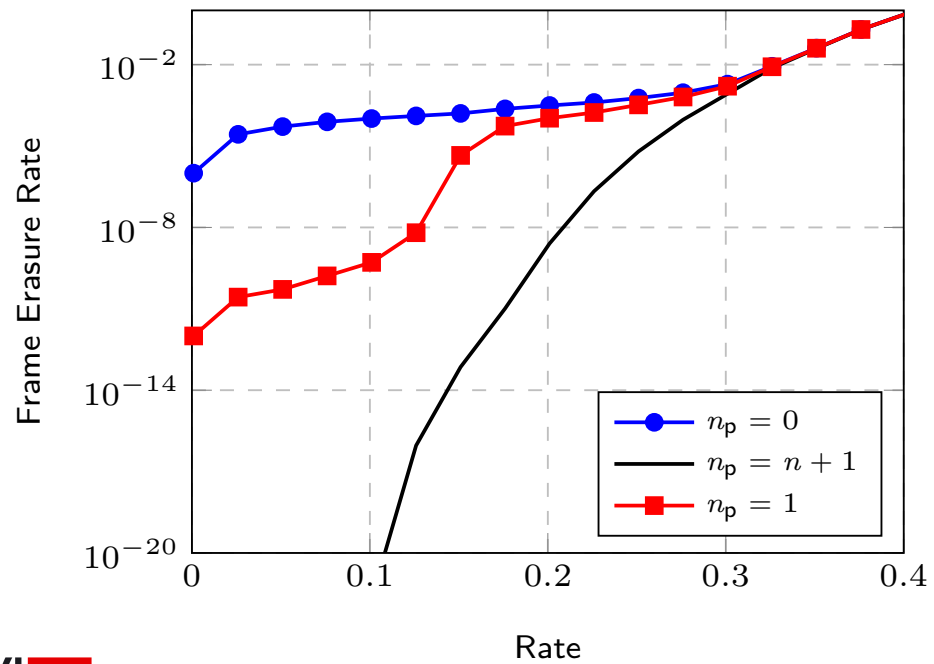
- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$



% of protected MEs ($n = 10$):

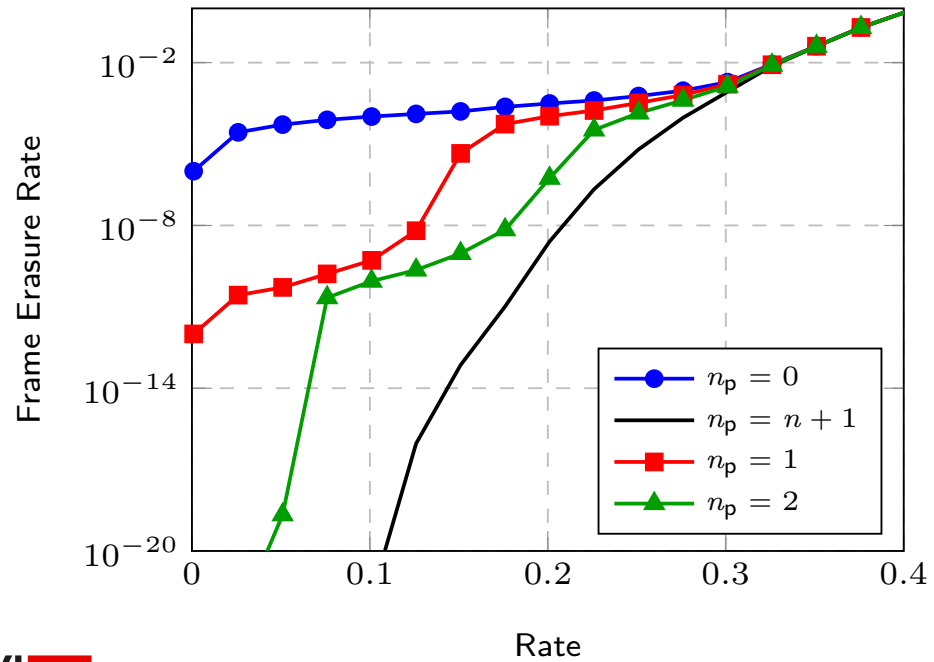
- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed $n_u < n+1$, ϵ_j **converges a.s.** to a random variable $\epsilon_\infty \in \{0,1\}$.*
2. ***Rate loss**: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$*



% of protected MEs ($n = 10$):

- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = 2$: 0.15%

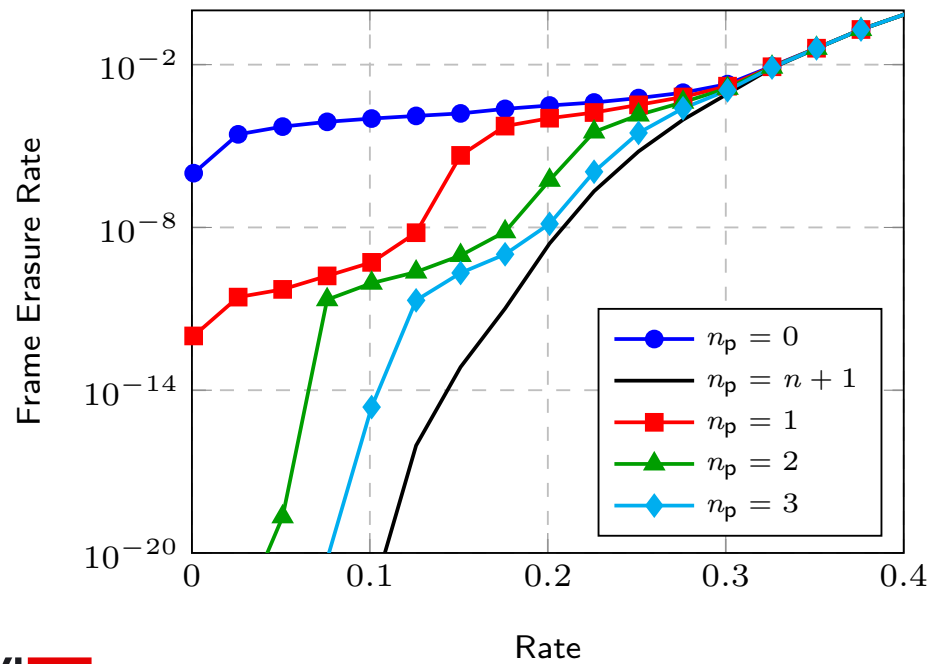
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$



% of protected MEs ($n = 10$):

- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = 2$: 0.15%
- $n_\mathsf{p} = 3$: 0.34%

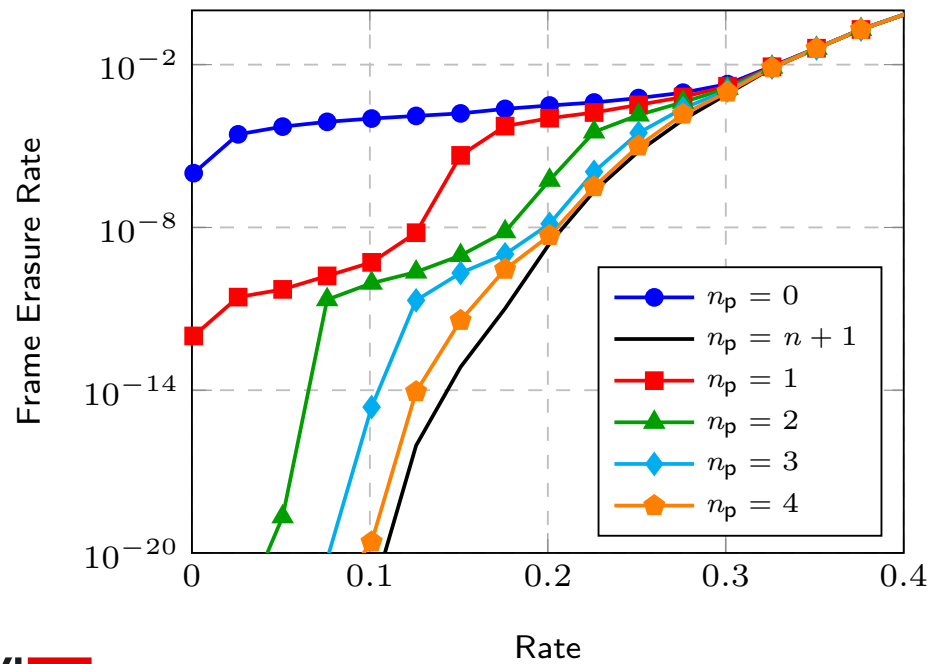
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$



% of protected MEs ($n = 10$):

- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = 2$: 0.15%
- $n_\mathsf{p} = 3$: 0.34%
- $n_\mathsf{p} = 4$: 0.73%

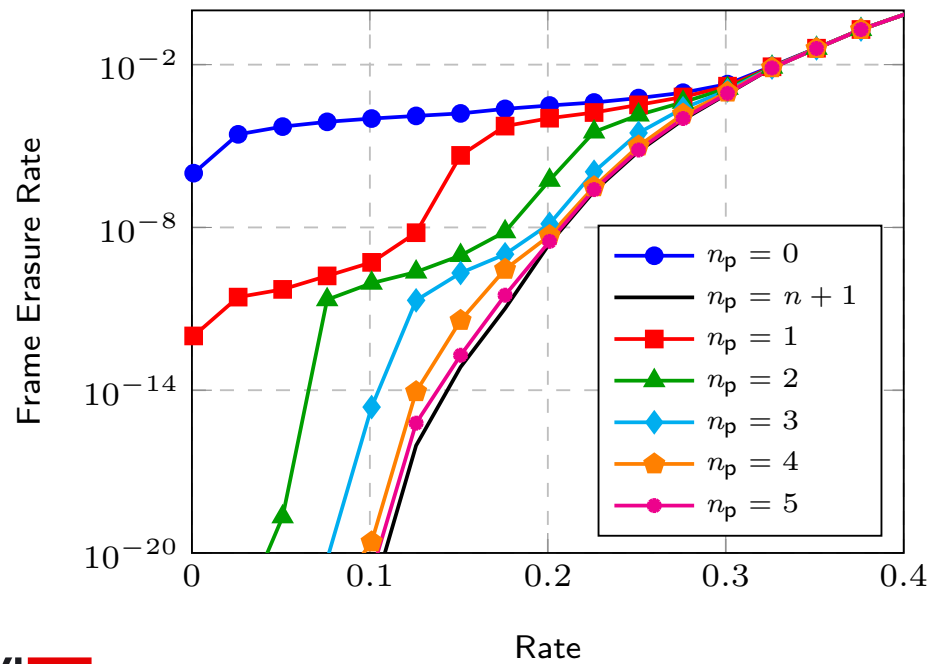
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$



% of protected MEs ($n = 10$):

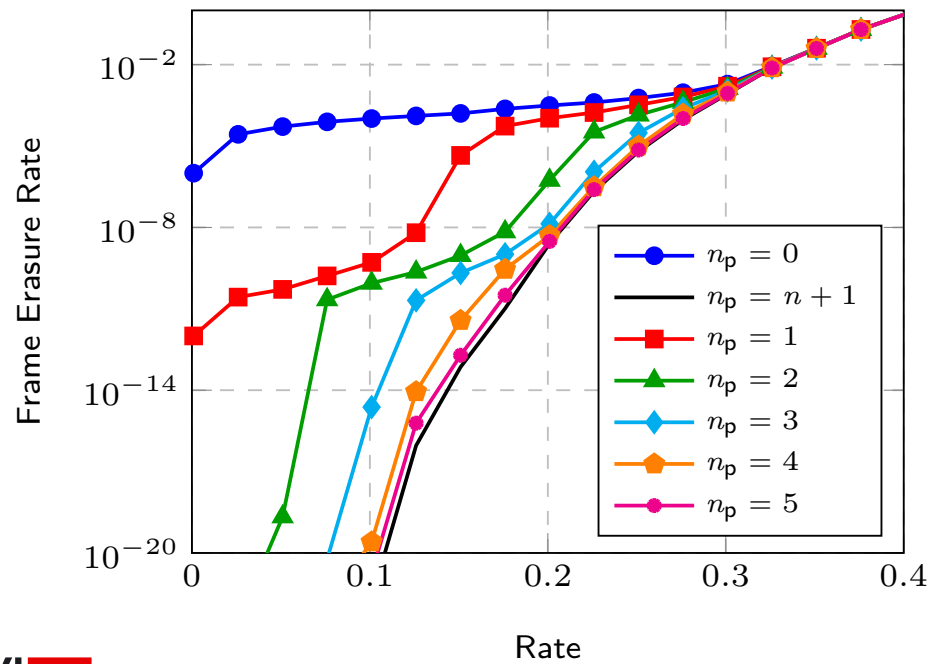
- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = 2$: 0.15%
- $n_\mathsf{p} = 3$: 0.34%
- $n_\mathsf{p} = 4$: 0.73%
- $n_\mathsf{p} = 5$: 1.51%
- $n_\mathsf{p} = n+1$: 100.00%

Error-free transmission via unequal error protection

- **Concept:** Nodes closer to the root contribute more to the erasure rate reduction
- Protect $n_\mathsf{p} = (n+1) - n_\mathsf{u}$ levels for n_u fixed: protected fraction is **constant**

Theorem (Full reliability by protecting a constant fraction of the decoder)

1. *For any fixed* $n_u < n+1$, ϵ_j ***converges a.s.*** *to a random variable* $\epsilon_\infty \in \{0,1\}$.
2. ***Rate loss***: $P(\epsilon_\infty = 0) = \underbrace{(1-\delta)^{n_u}}_{\text{mult. penalty}} \underbrace{(1-p)}_{\text{BEC cap.}}$



% of protected MEs ($n = 10$):

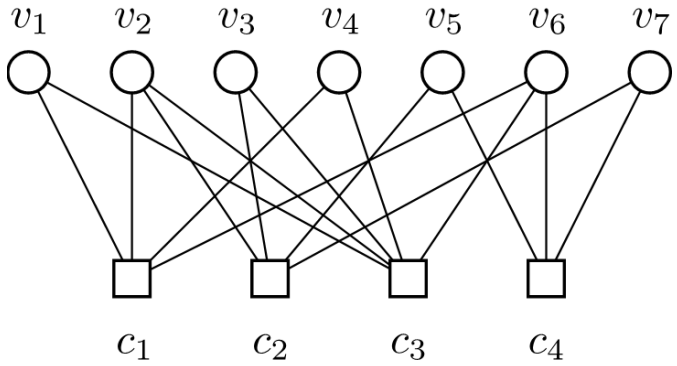
- $n_\mathsf{p} = 0$: 0.00%
- $n_\mathsf{p} = 1$: 0.05%
- $n_\mathsf{p} = 2$: 0.15%
- $n_\mathsf{p} = 3$: 0.34%
- $n_\mathsf{p} = 4$: 0.73%
- $n_\mathsf{p} = 5$: 1.51%
- $n_\mathsf{p} = n+1$: 100.00%

Min-Sum Decoding of LDPC Codes

- LDPC codes are linear block codes with a sparse parity-check matrix

Min-Sum Decoding of LDPC Codes

- LDPC codes are linear block codes with a sparse parity-check matrix
- An LDPC code can be represented as a Tanner graph with:
  - **Variable nodes** (VNs)
  - **Check nodes** (CNs)



Min-Sum Decoding

$$\text{Variable-to-check messages:} \quad \Phi_v(L, \boldsymbol{\mu}) = L + \sum_i \mu_i,$$

$$\text{Check-to-variable messages:} \quad \Phi_c(\boldsymbol{\mu}) = \prod_j \operatorname{sign} \mu_j \, \min |\boldsymbol{\mu}|.$$

Finite-Alphabet Message Passing Decoding

- In hardware implementations of MS decoding, uniform quantization is used.

Finite-Alphabet Message Passing Decoding

- In hardware implementations of MS decoding, uniform quantization is used.

Conventional Message-Passing

$$\text{Decoding Algorithm} \xrightarrow{\text{defines}} \text{Quantization}$$

- **Efficient** arithmetic circuits, but **suboptimal** error-correcting performance.

Finite-Alphabet Message Passing Decoding

- In hardware implementations of MS decoding, uniform quantization is used.

Conventional Message-Passing

$$\text{Decoding Algorithm} \xrightarrow{\text{defines}} \text{Quantization}$$

- **Efficient** arithmetic circuits, but **suboptimal** error-correcting performance.

Finite-Alphabet Message-Passing

$$\text{Quantization} \xrightarrow{\text{defines}} \text{Decoding Algorithm}$$

- Updates are implemented as optimized look-up tables (LUTs).
- Potential for **significant bit-width reduction and performance improvement**.

1. R. Ghanaatian, A. Balatsoukas-Stimming, C. Müller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588 Gbps LDPC decoder based on finite-alphabet message passing," IEEE Transactions on Very Large Scale Integration Systems, Feb. 2018
2. M. Meidlinger, A. Balatsoukas-Stimming, A. Burg, and G. Matz, "Quantized message passing for LDPC codes," in Asilomar Conference on Signals, Systems, and Computers, May 2015
3. A. Balatsoukas-Stimming, M. Meidlinger, R. Ghanaatian, G. Matz, and A. Burg, "A fully-unrolled LDPC decoder based on quantized message passing," in IEEE International Workshop on Signal Processing Systems (SiPS), Oct. 2015

Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

Local **maximization of mutual information** between messages and codeword bits.

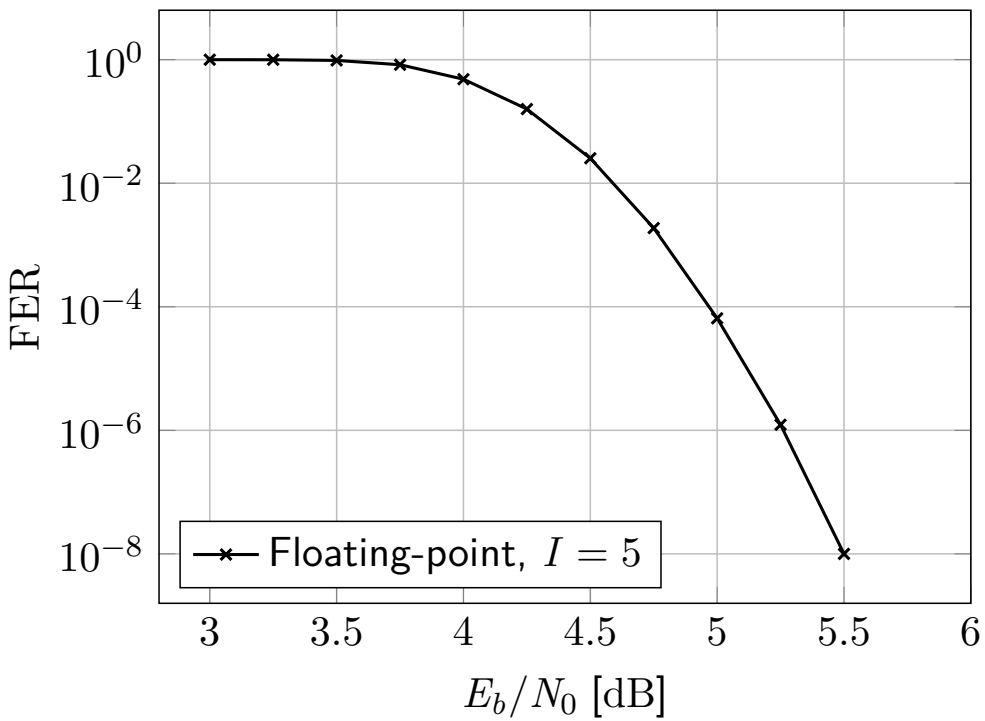
Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

Local **maximization of mutual information** between messages and codeword bits.

- IEEE 802.3an LDPC code ($d_v = 6,\ d_c = 32,\ N = 2048, I_{\text{max}} = 5$)



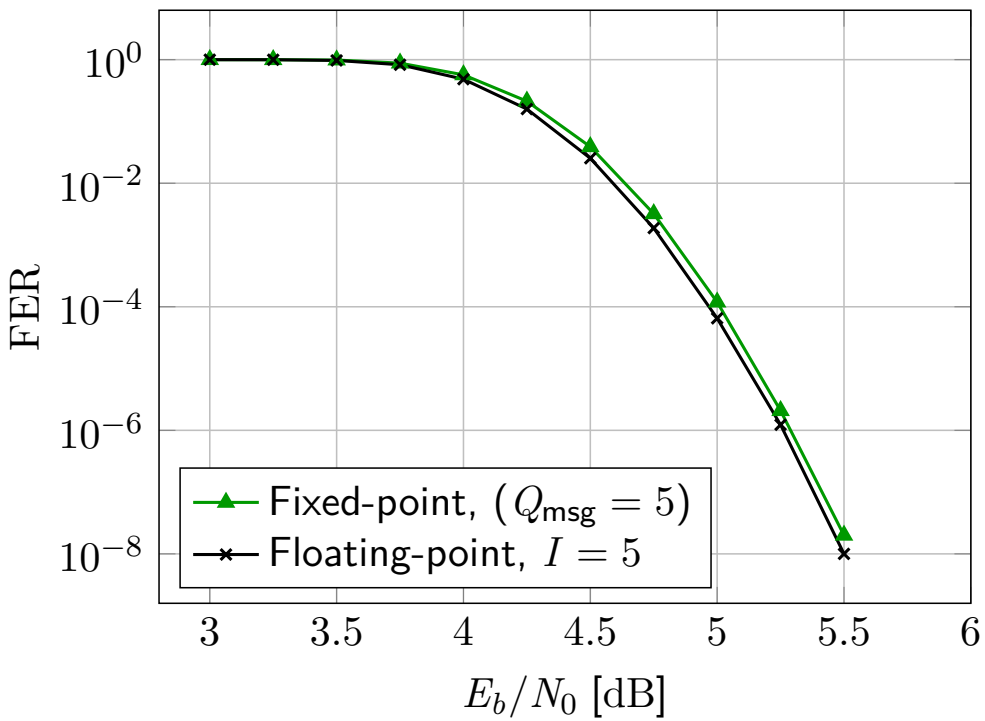
Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

Local **maximization of mutual information** between messages and codeword bits.

- IEEE 802.3an LDPC code ($d_v = 6,\ d_c = 32,\ N = 2048, I_{\text{max}} = 5$)



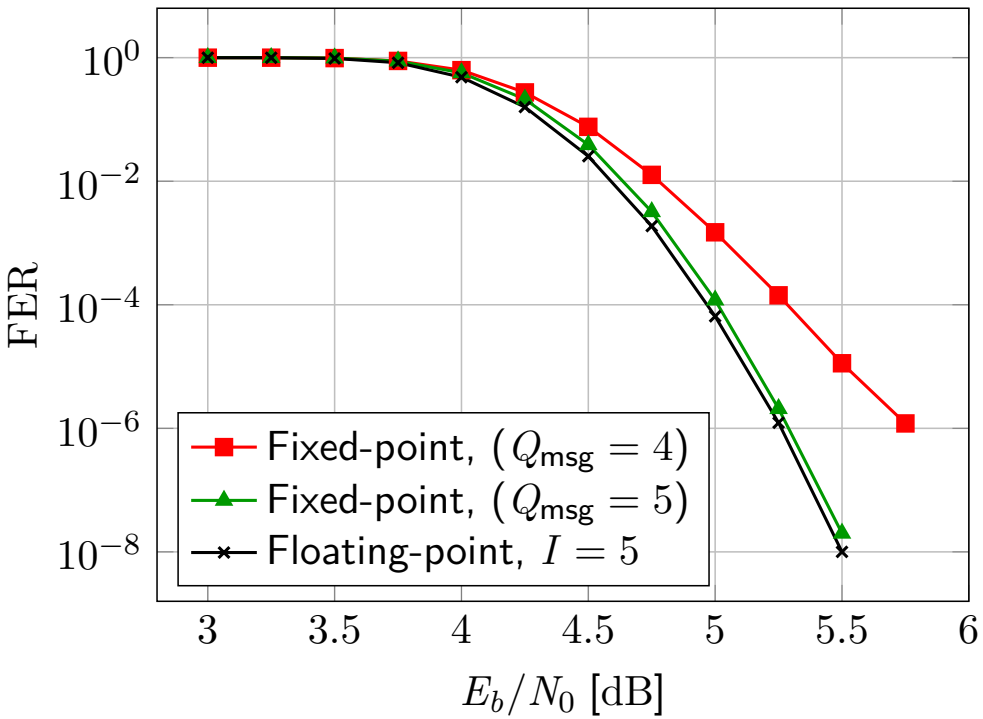
Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

Local **maximization of mutual information** between messages and codeword bits.

- IEEE 802.3an LDPC code ($d_v = 6,\ d_c = 32,\ N = 2048, I_{\text{max}} = 5$)



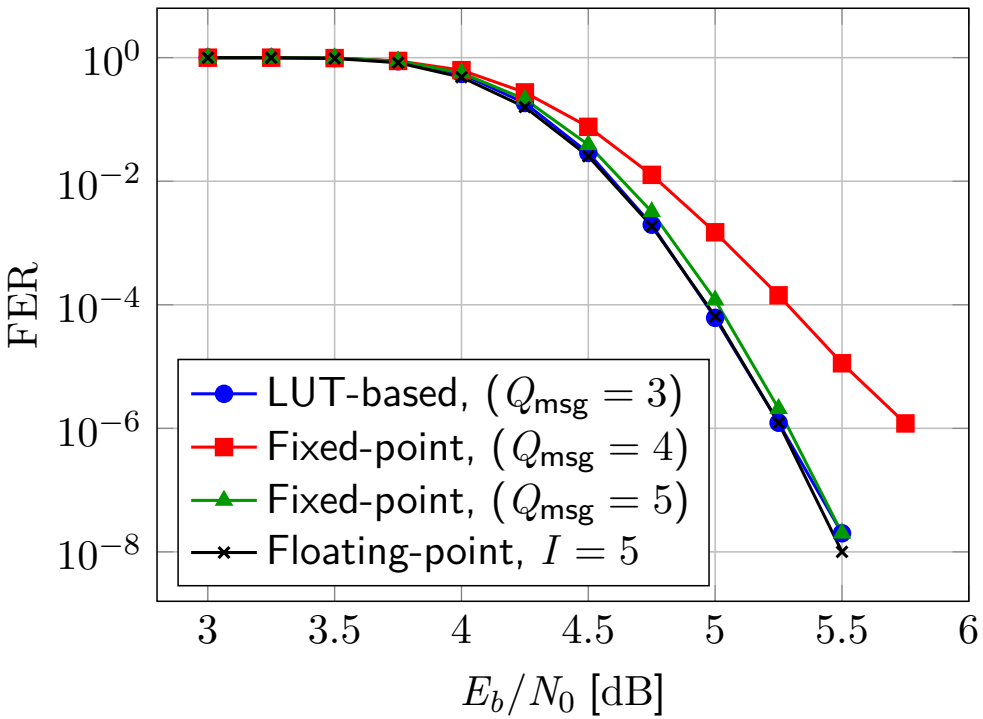
Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

Local **maximization of mutual information** between messages and codeword bits.

- IEEE 802.3an LDPC code ($d_v = 6,\ d_c = 32,\ N = 2048, I_{\text{max}} = 5$)



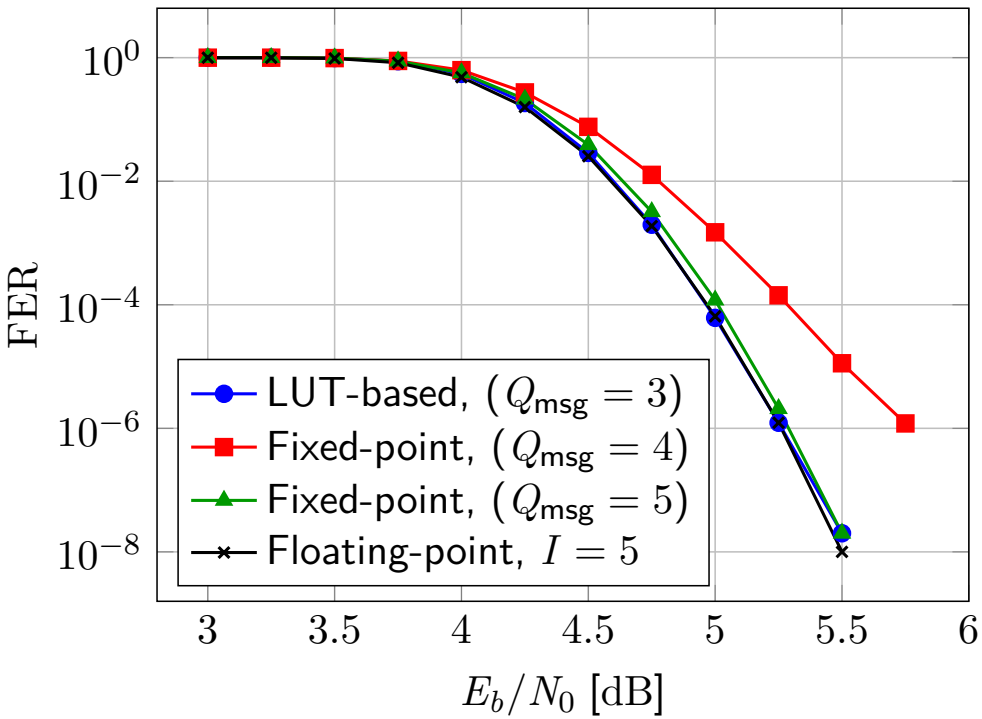
Look-Up Table Design

- Our method is based on an **information theoretic** criterion.

LUT Design Principle

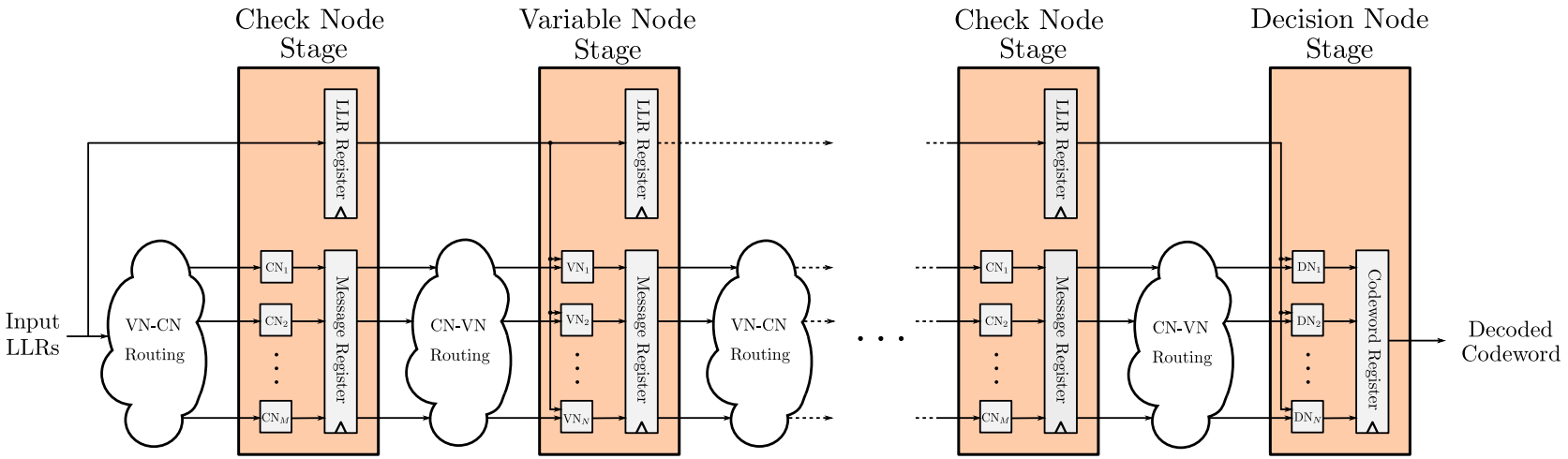
Local **maximization of mutual information** between messages and codeword bits.

- IEEE 802.3an LDPC code ($d_v = 6,\ d_c = 32,\ N = 2048, I_{\text{max}} = 5$)



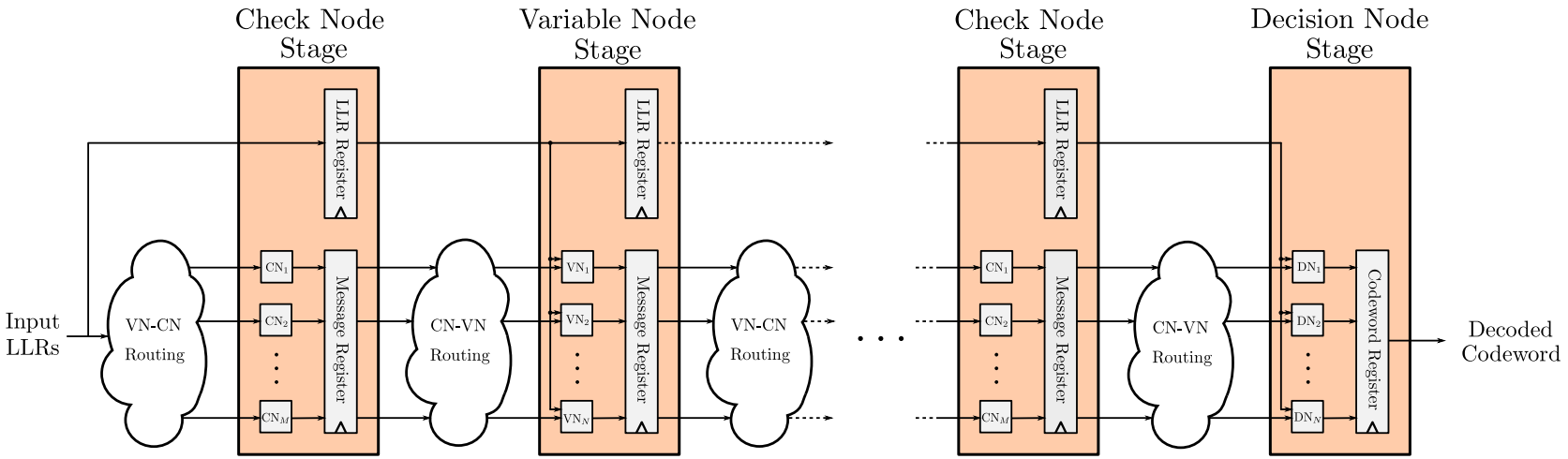
40% message bit-width reduction with identical FER performance.

Fully Unrolled LDPC Decoder Hardware Architecture



- One CN stage and one VN stage per iteration: $2I_{\text{max}}$ pipeline stages.

Fully Unrolled LDPC Decoder Hardware Architecture



- One CN stage and one VN stage per iteration: $2I_{\text{max}}$ pipeline stages.

	Quant. MS	LUT-based	
Area (mm^2)	35.63	33.79	**−5%**
Throughput (Gbps)	1014	1665	**+64%**
Area Eff. (Gbps/mm^2)	28.46	49.27	**+73%**

Bi-directional Wireless Communications

Time-division duplexing (TDD)

Wasted time resources: switching interval

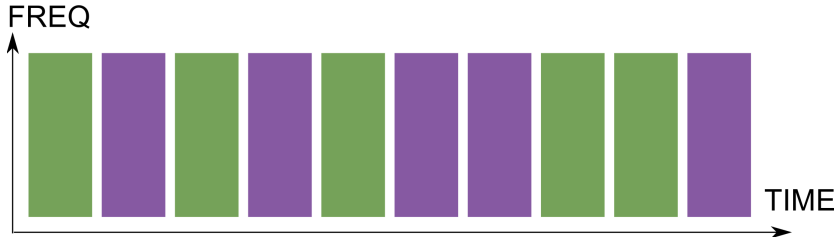
Frequency-division duplexing (FDD)

Wasted frequency resources: guard bands

Bi-directional Wireless Communications

Time-division duplexing (TDD)

Wasted time resources: switching interval



Frequency-division duplexing (FDD)

Wasted frequency resources: guard bands

In-Band Full-Duplex (IBFD)

Up to twice the throughput wrt TDD & FDD!

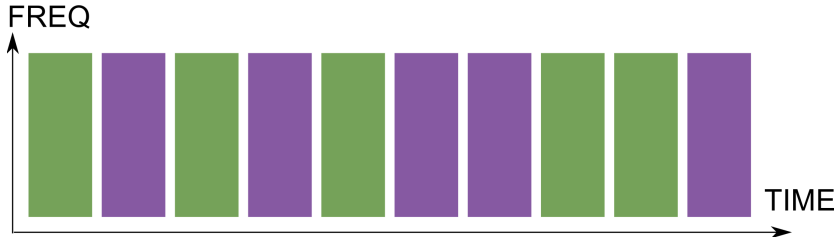
No additional bandwidth

No wasted time or frequency resources

Bi-directional Wireless Communications

Time-division duplexing (TDD)

Wasted time resources: switching interval



Frequency-division duplexing (FDD)

Wasted frequency resources: guard bands

In-Band Full-Duplex (IBFD)

Up to twice the throughput wrt TDD & FDD!

No additional bandwidth

No wasted time or frequency resources

Fundamental Challenge: Self-interference is much stronger than the desired signal!

Why is Cancellation Challenging?

- **In principle**, cancellation is easy since digital transmitted signal is known!

Why is Cancellation Challenging?

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story!

Why is Cancellation Challenging?

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story!
  - Analog components introduce non-linearities that make **digital cancellation difficult**

Why is Cancellation Challenging?

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story!
  - Analog components introduce non-linearities that make **digital cancellation difficult**

- Consider a state-of-the-art non-linear cancellation model:

$$y(n) = \sum_{\substack{p=1,\\ p \text{ odd}}}^{P} \sum_{q=0}^{p} \sum_{m=0}^{M+L-1} \underbrace{h_{p,q}(m)}_{\text{params}} \underbrace{x(n-m)^q x^*(n-m)^{p-q}}_{\text{basis functions}}$$

1. D. Korpi, L. Anttila, and M. Valkama, "Nonlinear self-interference cancellation in MIMO full-duplex transceivers under crosstalk," EURASIP Journal on Wireless Communications and Networking, Feb. 2017

Why is Cancellation Challenging?

- **In principle**, cancellation is easy since digital transmitted signal is known!
- **In practice**, the digital signal does not tell the whole story!
  - Analog components introduce non-linearities that make **digital cancellation difficult**

- Consider a state-of-the-art non-linear cancellation model:

$$y(n) = \sum_{\substack{p=1,\\ p \text{ odd}}}^{P} \sum_{q=0}^{p} \sum_{m=0}^{M+L-1} \underbrace{h_{p,q}(m)}_{\text{params}} \underbrace{x(n-m)^q x^*(n-m)^{p-q}}_{\text{basis functions}}$$

Idea

Why not use a **neural network** that **extracts structure** from training data?

1. D. Korpi, L. Anttila, and M. Valkama, "Nonlinear self-interference cancellation in MIMO full-duplex transceivers under crosstalk," EURASIP Journal on Wireless Communications and Networking, Feb. 2017
2. A. Balatsoukas-Stimming, "Non-linear digital self-interference cancellation for in-band full-duplex radios using neural networks," IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Jun. 2018

Self-Interference Cancellation Using Neural Networks

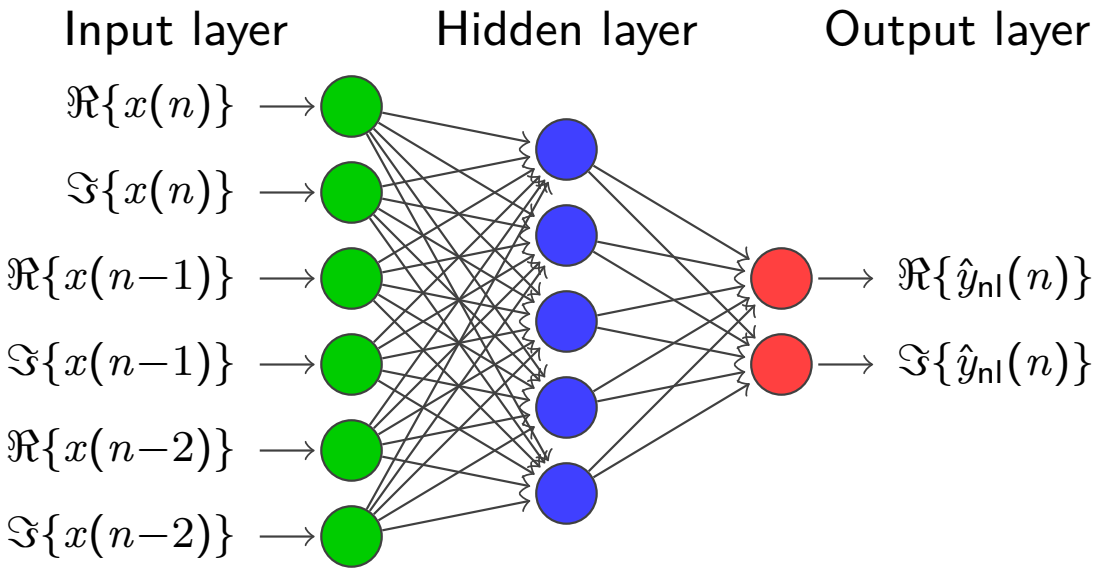
- **Two-step cancellation:**

Self-Interference Cancellation Using Neural Networks

- **Two-step cancellation:**
  1. Use standard linear digital cancellation: $\hat{y}_{\text{lin}}(n) = \sum_{m=0}^{M+L-1} \hat{h}_{1,1}(m)x(n-m)$

Self-Interference Cancellation Using Neural Networks

- **Two-step cancellation:**
  1. Use standard linear digital cancellation: $\hat{y}_{\text{lin}}(n) = \sum_{m=0}^{M+L-1} \hat{h}_{1,1}(m)x(n-m)$
  2. Train a neural network to reproduce and cancel $y_{\text{nl}}(n) \approx y(n) - \hat{y}_{\text{lin}}(n)$

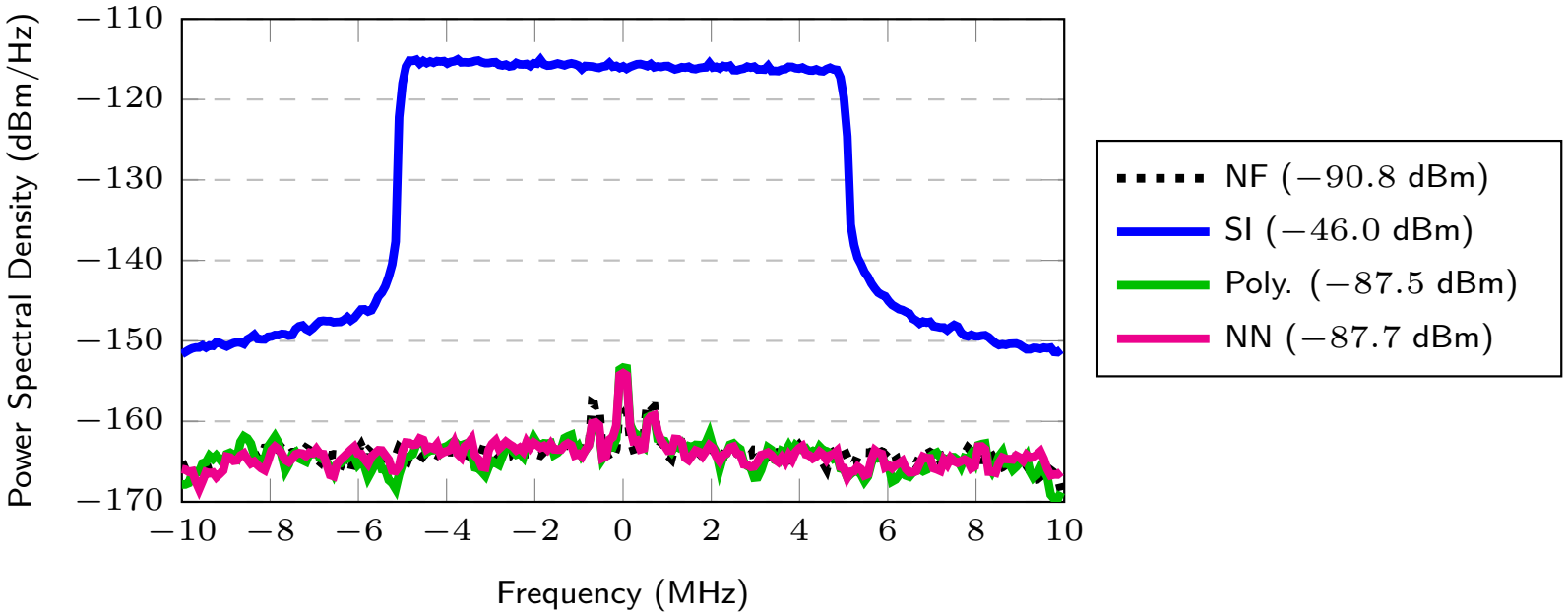


Experimental Cancellation Results

- 10 MHz OFDM signal, QPSK modulation

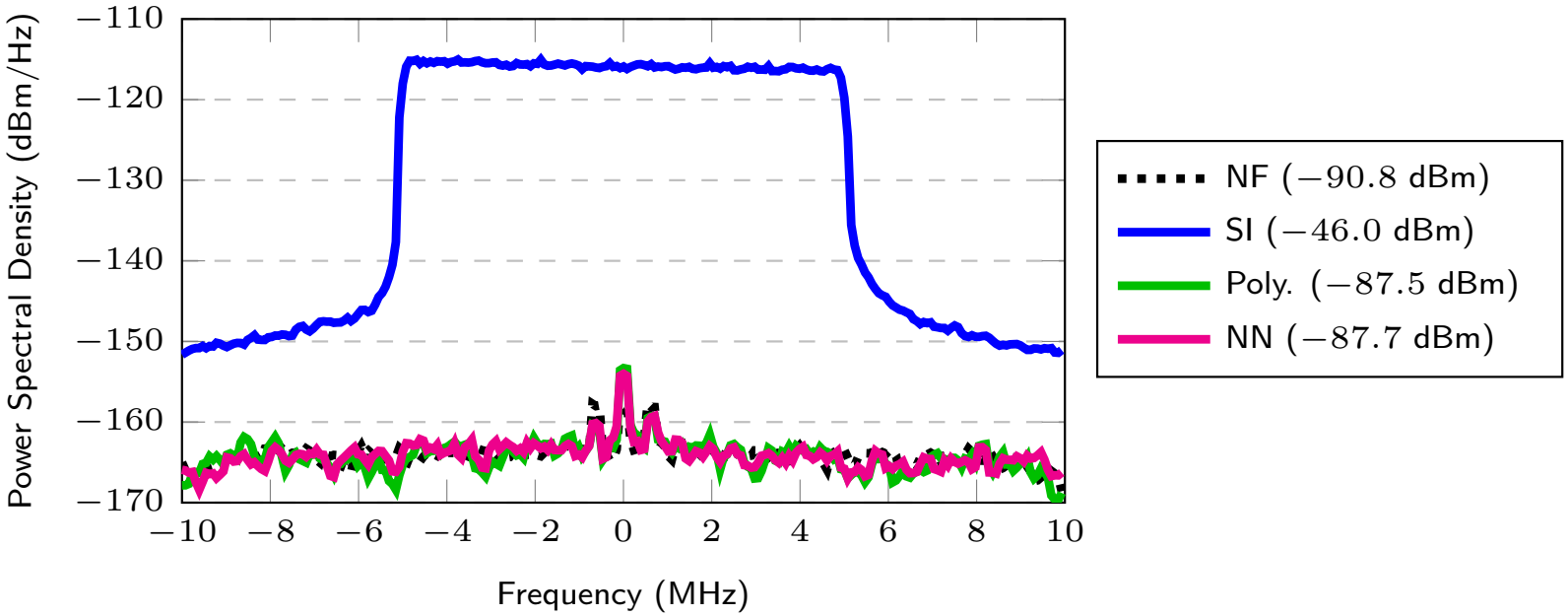
Experimental Cancellation Results

- 10 MHz OFDM signal, QPSK modulation



Experimental Cancellation Results

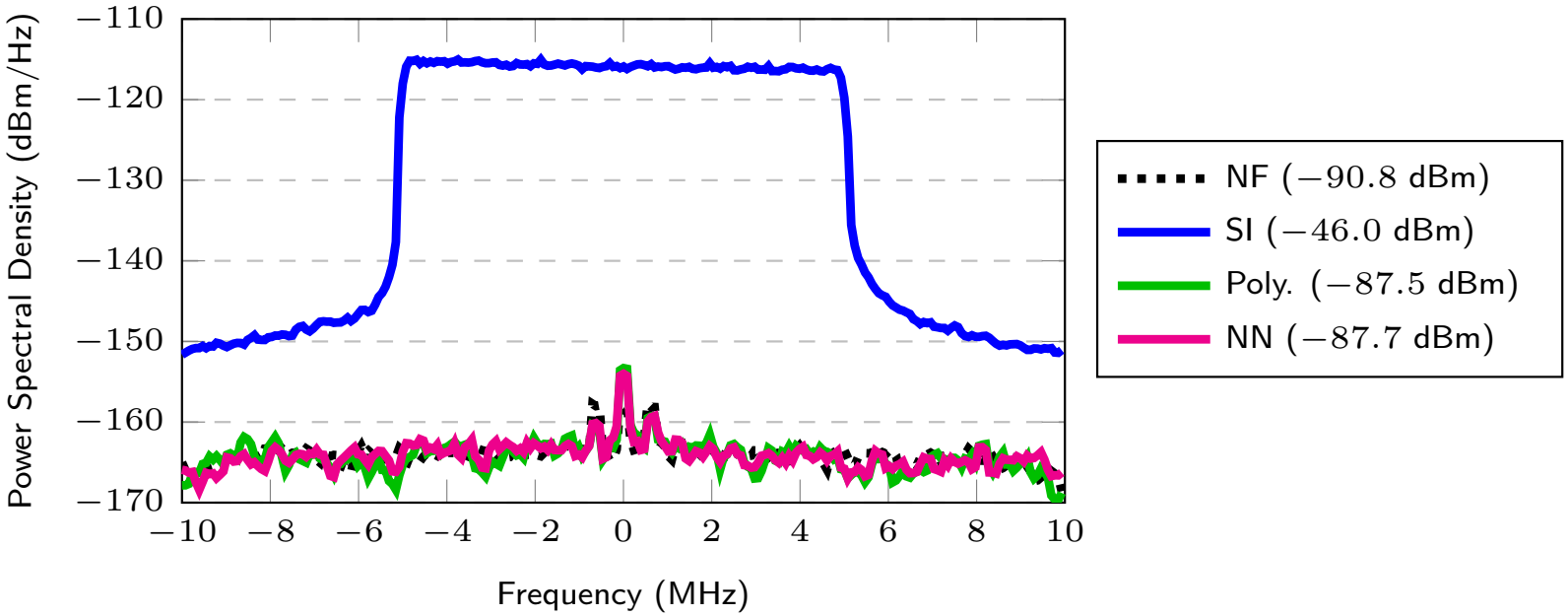
- 10 MHz OFDM signal, QPSK modulation



	Poly.	NN	Gain
Add.	1818	593	**67%**
Mult.	780	543	**30%**

Experimental Cancellation Results

- 10 MHz OFDM signal, QPSK modulation



	Poly.	NN	Gain
Add.	1818	593	**67%**
Mult.	780	543	**30%**

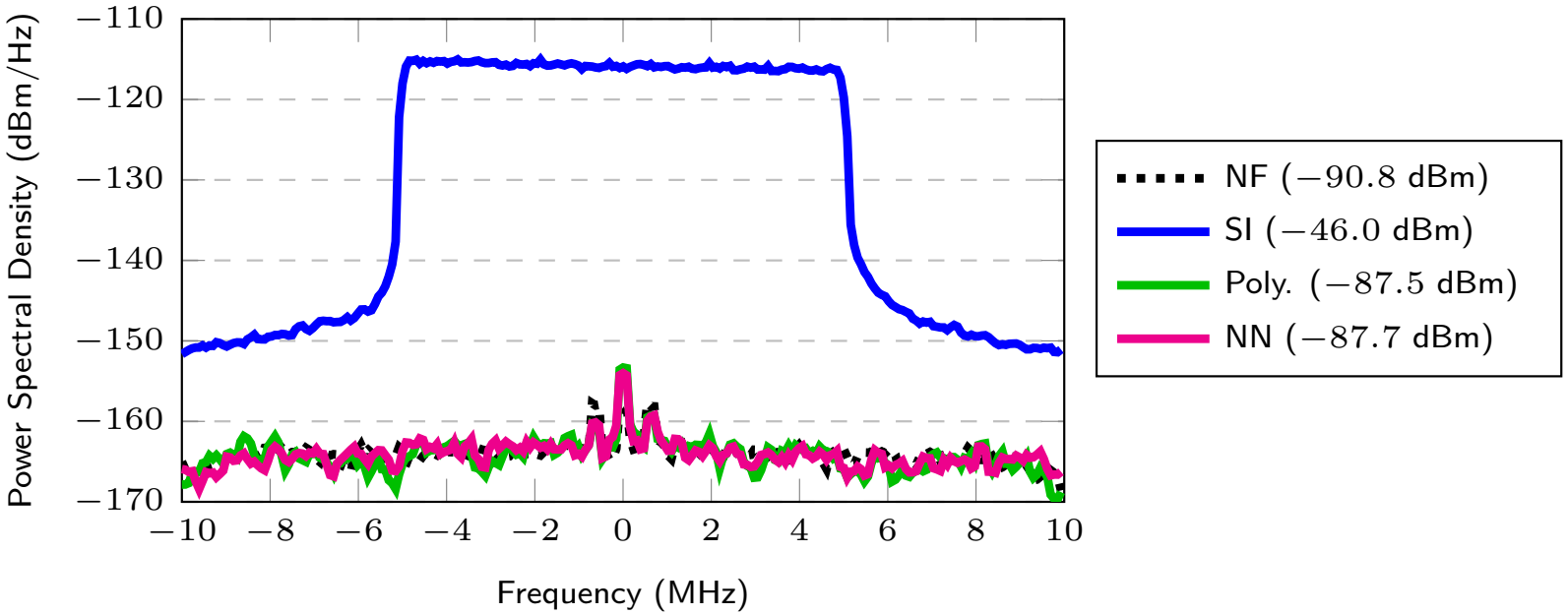
	Poly.	NN	Gain
Area (mm^2)	0.36	0.32	**−11%**
T/P (MS/s)	17.4	27.8	**+60%**

Placed-and-routed, 28 nm FD-SOI, slow-slow corner, 125° C

1 Y. Kurzo. A. Burg, A. Balatsoukas-Stimming, "Design and implementation of a neural network aided self-interference cancellation scheme for full-duplex radios," Asilomar Conference on Signals, Systems, and Computers, Nov. 2018.

Experimental Cancellation Results

- 10 MHz OFDM signal, QPSK modulation



	Poly.	NN	Gain
Add.	1818	593	**67%**
Mult.	780	543	**30%**

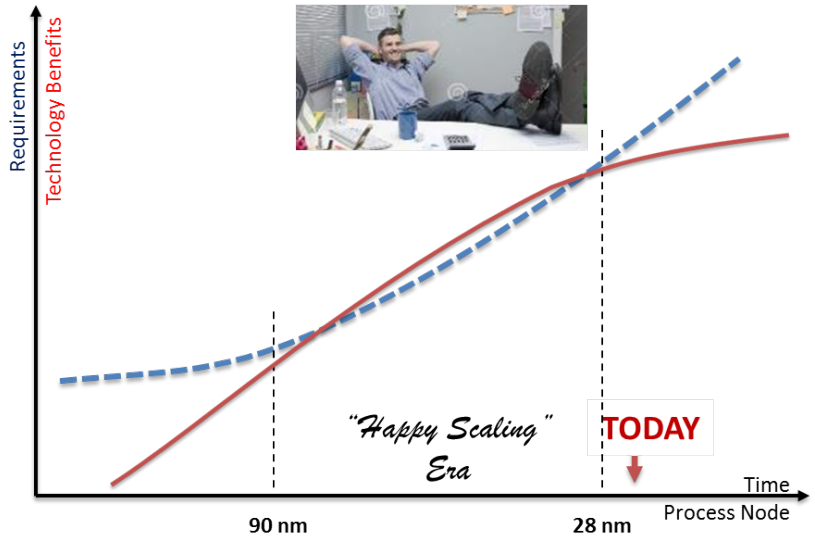
	Poly.	NN	Gain
Area (mm^2)	0.36	0.32	**−11%**
T/P (MS/s)	17.4	27.8	**+60%**

Placed-and-routed, 28 nm FD-SOI, slow-slow corner, 125° C

Identical cancellation performance with **significantly lower complexity**!

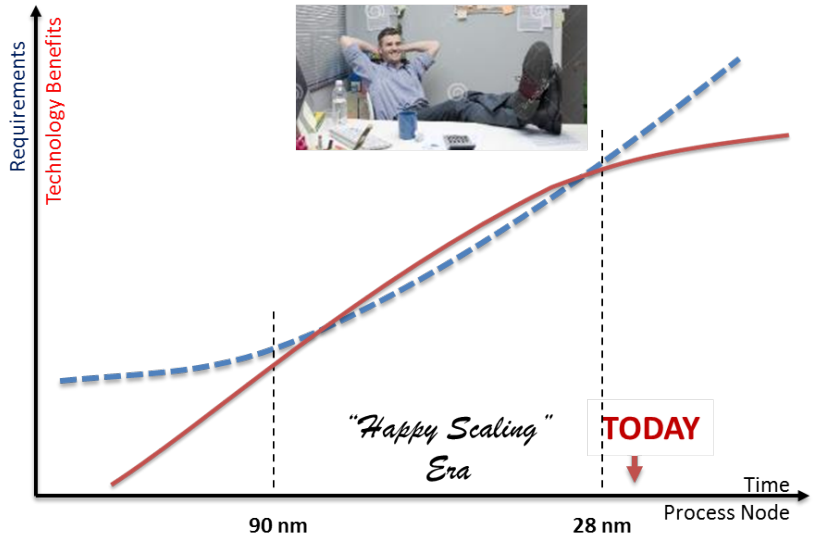
1 Y. Kurzo. A. Burg, A. Balatsoukas-Stimming, "Design and implementation of a neural network aided self-interference cancellation scheme for full-duplex radios," Asilomar Conference on Signals, Systems, and Computers, Nov. 2018.

Conclusions



- A combination of approaches is necessary to maintain progress in the **post-happy-scaling era**, including:
  1. **Classical algorithm/hardware co-design**
  2. **Approximate computing**
  3. **Tools from other disciplines**

Conclusions

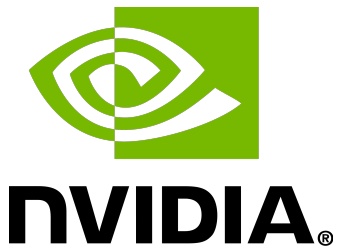


- A combination of approaches is necessary to maintain progress in the **post-happy-scaling era**, including:
  1. **Classical algorithm/hardware co-design**
  2. **Approximate computing**
  3. **Tools from other disciplines**

The end of the happy scaling era creates new **challenges and opportunities** for innovation!

Acknowledgments

Funding:

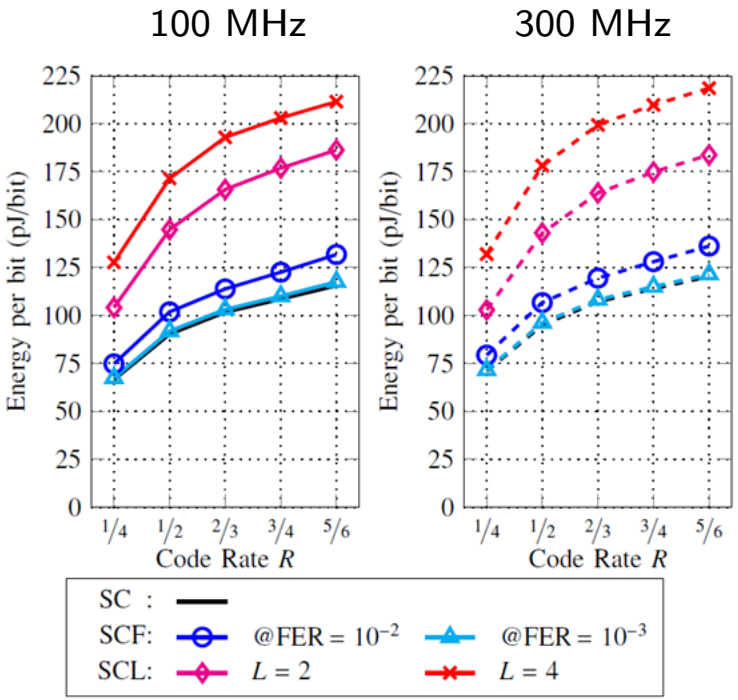


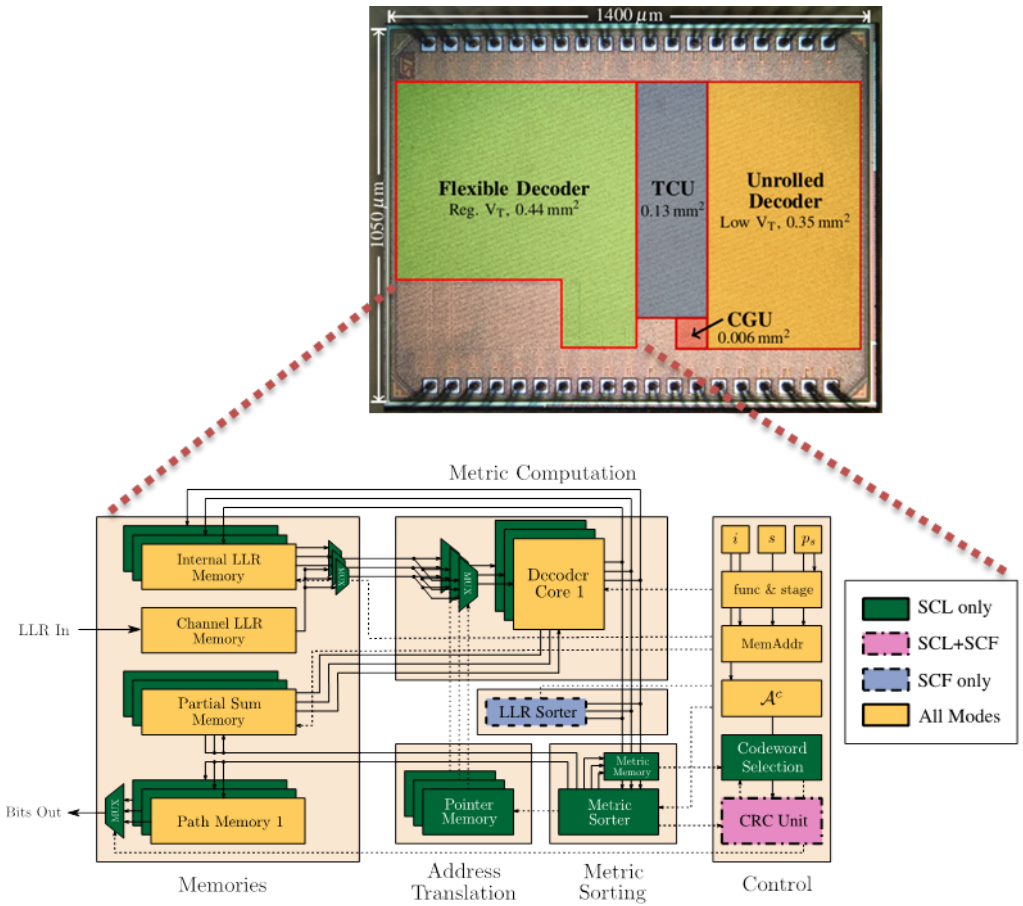
Collaborators:

- O. Afisiadis, K. Alexandris, A. Austin, M. Bastani Parizi, P. Belanovic, A. Burg, R. Ghanaatian, P. Giard, G. Karakonstantis, Y. Kurzo, M. Van Beirendonck (EPFL)
- W. J. Gross, S. A. Hashemi (McGill University)
- J. R. Cavallaro (Rice University)
- G. Matz, M. Meidlinger (TU Vienna)
- T. Podzorny, J. Uythoven (CERN)
- C.-H. Chen, F. Sheikh (Intel Labs)

Bringing It All Together

- PolarBear: Manufactured ASIC in **ST 28 nm FD-SOI**
  - SC, SCF, and SCL decoding on the same chip
  - Run-time algorithm selection for **energy-proportional operation**





1 P. Giard, A. Balatsoukas-Stimming, C. Müller, A. Bonetti, C. Thibeault, W. J. Gross, P. Flatresse, A. Burg, "PolarBear: A 28-nm FD-SOI ASIC for decoding of polar codes," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Dec. 2017