A 1.1μs 1.56Gb/s/mm^2 Cost-Efficient Large-List SCL Polar Decoder Using Fully-Reusable LLR Buffers in 28nm CMOS Tech.

Dongyun Kam[1], Byeong Yong Kong[2] and Youngjoo Lee[1]

[1] POSTECH, Pohang, South Korea
[2] Kongju National University, Cheonan, South Korea

Outline

- Introduction
- Implementation Challenges
- Proposed Design
- Implementation Results
- Conclusion

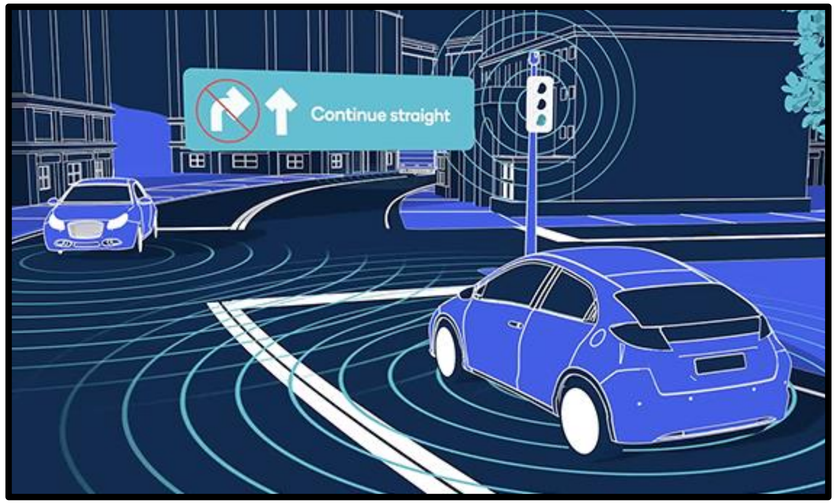
Outline

- Introduction
- Implementation Challenges
- Proposed Design
- Implementation Results
- Conclusion

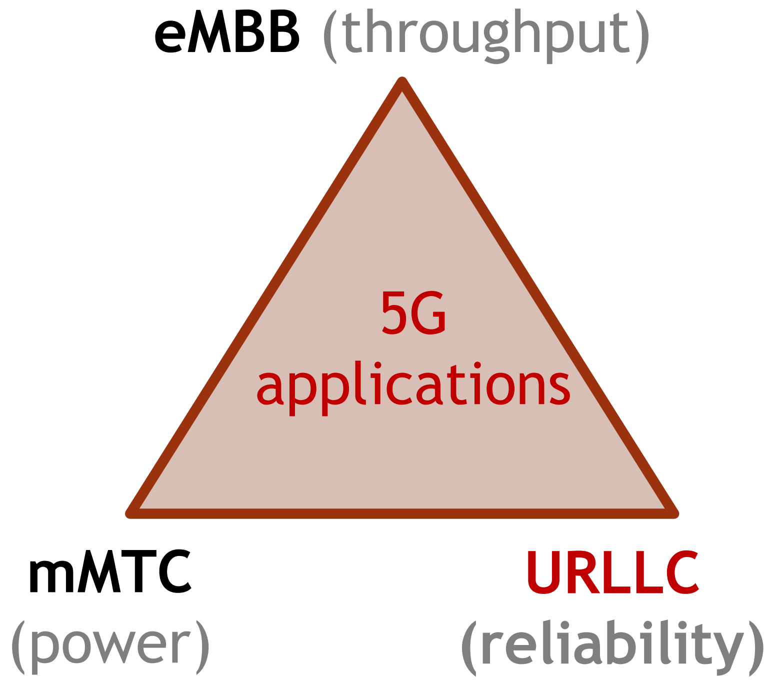
Introduction : 5G/B5G wireless communications

- 5G/B5G should support a lot of advanced applications.
- 3GPP categorizes 5G/B5G applications into three main scenarios.

[*Intel drone show*, 2018 Olympic]

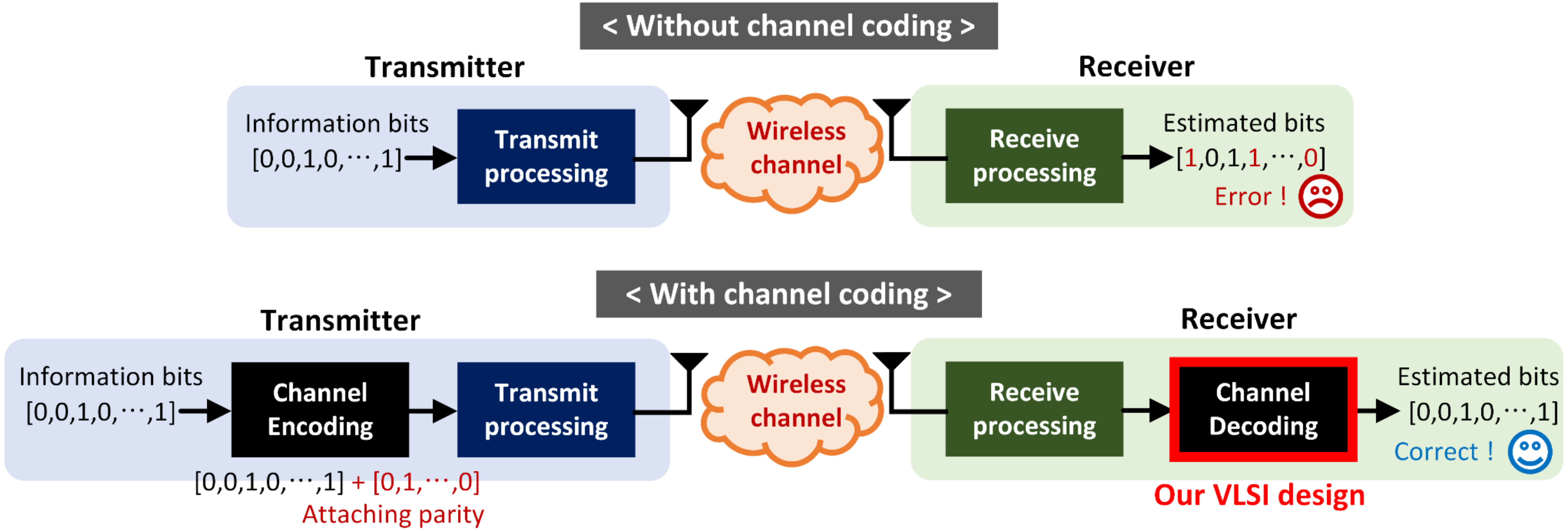


[*Qualcomm C-V2X*, 2022]



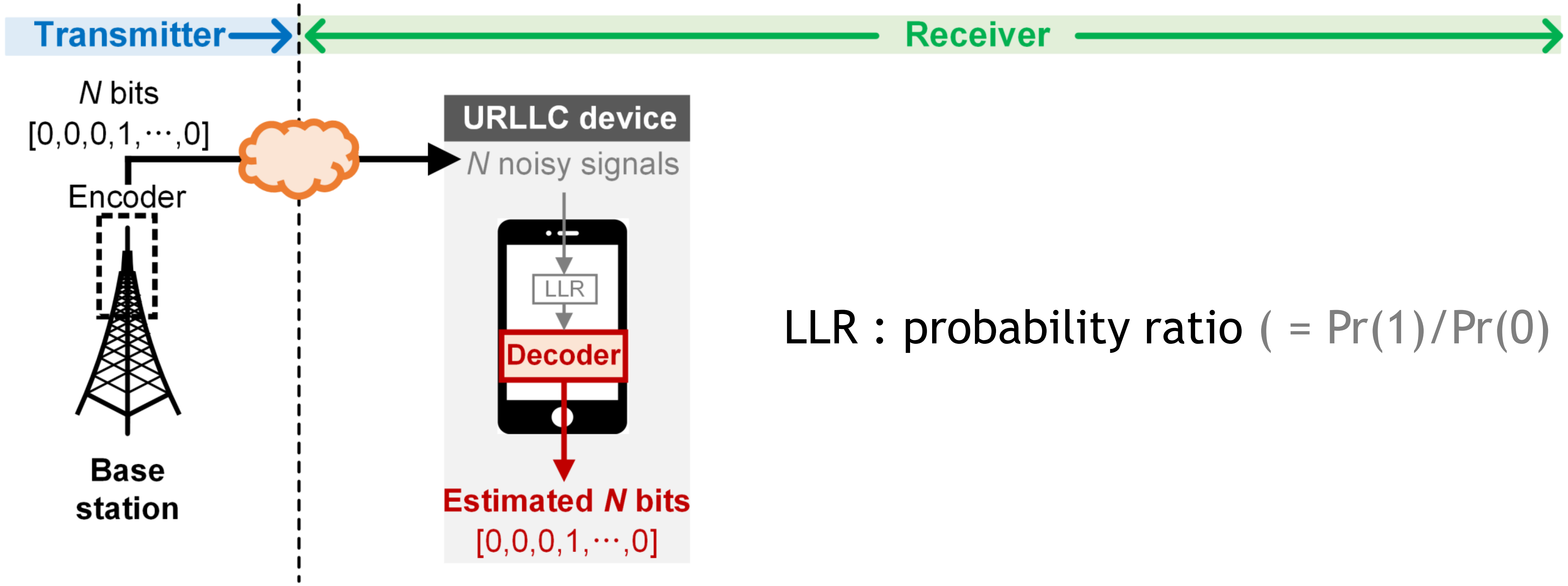
Introduction : 5G/B5G Channel coding

- Channel coding is a technique for reliable communications.
  - 5G standard coding : LDPC, **Polar codes (Esp. Polar decoder)**



Introduction : Polar decoder

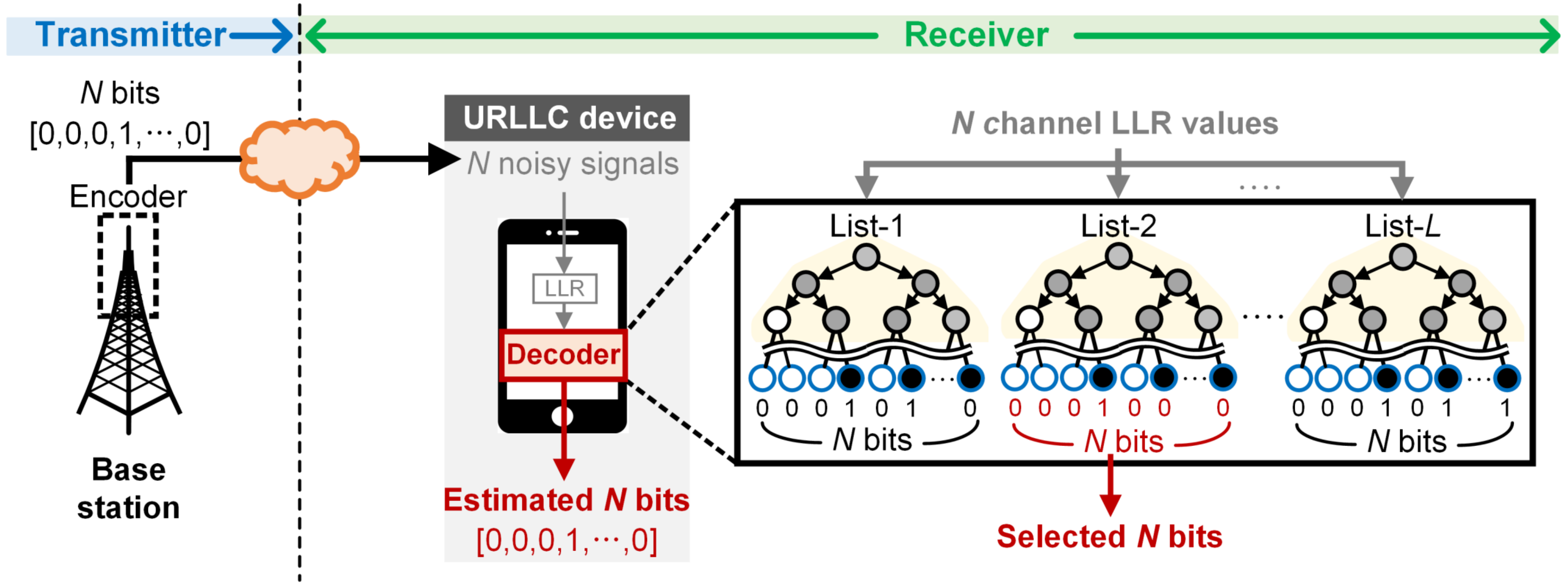
- List-*L* Successive Cancellation list (SCL) polar decoding
  - *N*-bit communications



LLR : probability ratio (= Pr(1)/Pr(0))

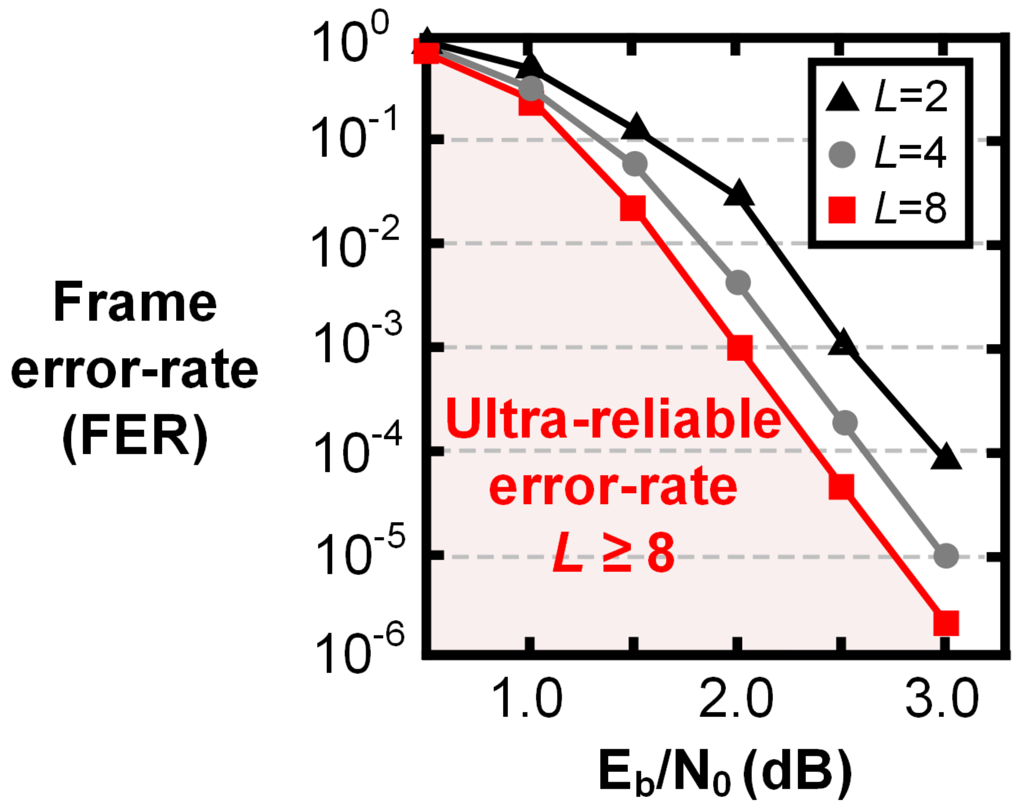
Introduction : Polar decoder

- List-*L* Successive Cancellation list (SCL) polar decoding
  - *L* parallel trees



Introduction : Large-list Polar decoder

- The URLLC requires ultra-reliability.
  - **A large-*L* SCL decoder** achieves lower error-rates.



Outline

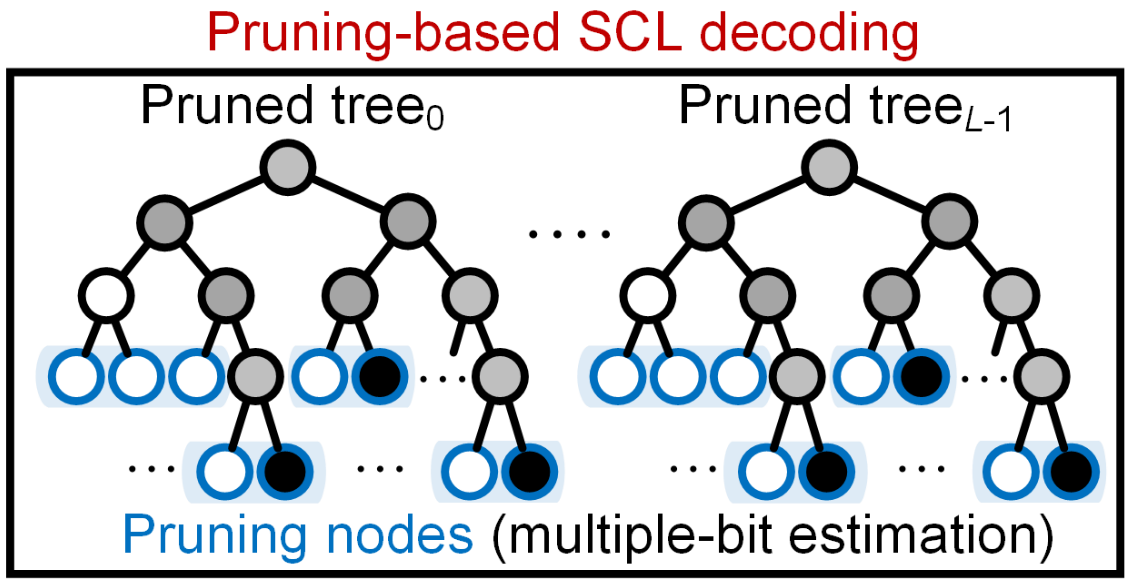
- Introduction
- Implementation Challenges
- Proposed Design
- Implementation Results
- Conclusion

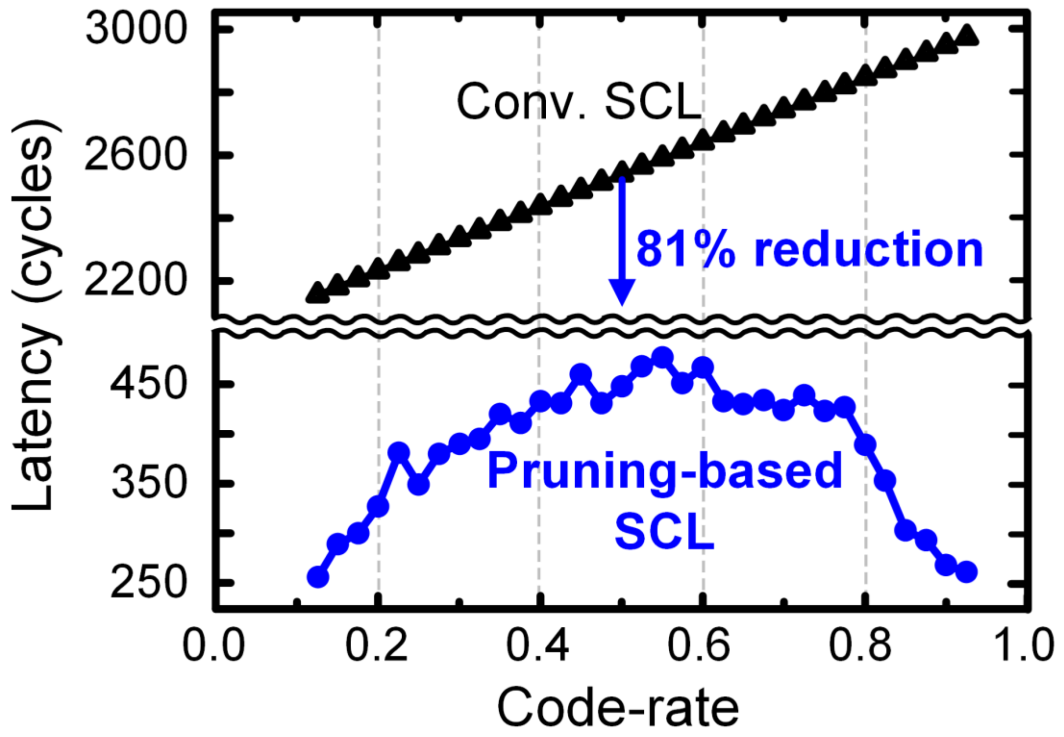
Challenges of Large-list decoders

- Implementing the large-list decoder has challenges.
  - **Challenge 1** : Long latency
  - **Challenge 2** : Huge hardware cost

Challenges of Large-list decoders

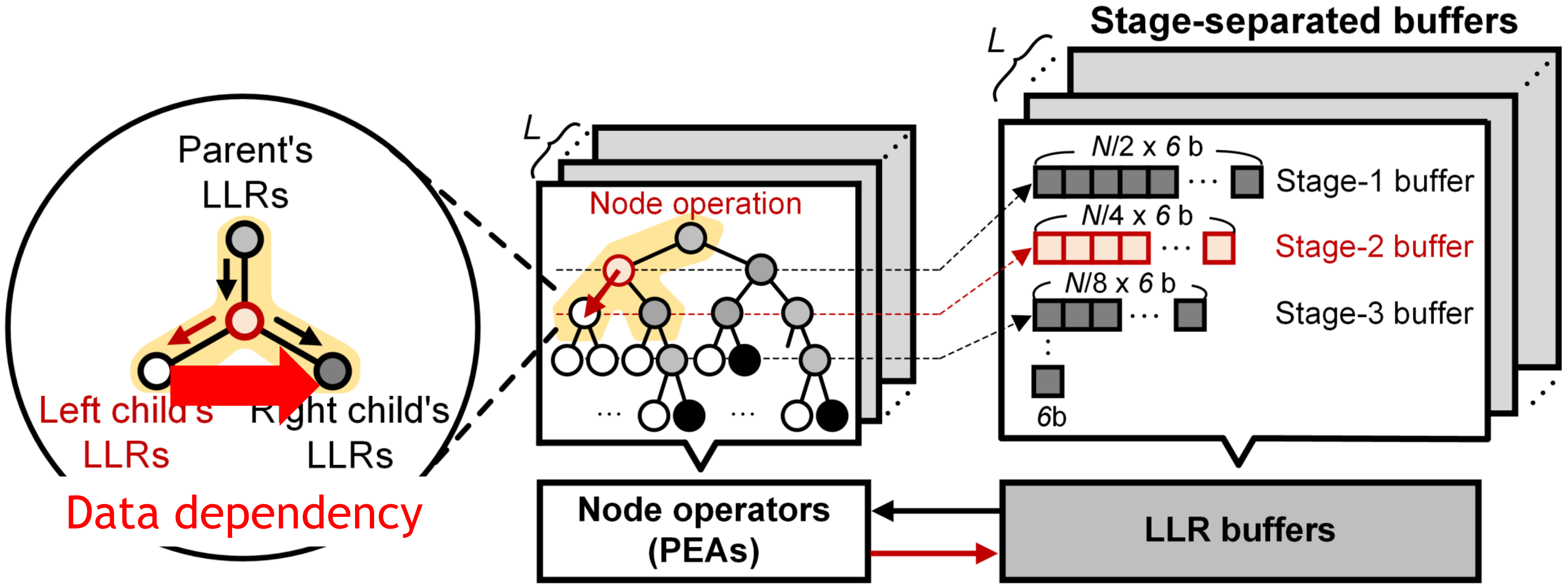
- Challenge 1 : Long latency
  - Algorithm-level solution is **the pruning-based SCL** [S. A. Hashemi, *TSP '17*].
  - There is no chip-level design for this.


Pruning-based SCL decoding
Pruned tree$_0$
Pruned tree$_{L-1}$
Pruning nodes (multiple-bit estimation)



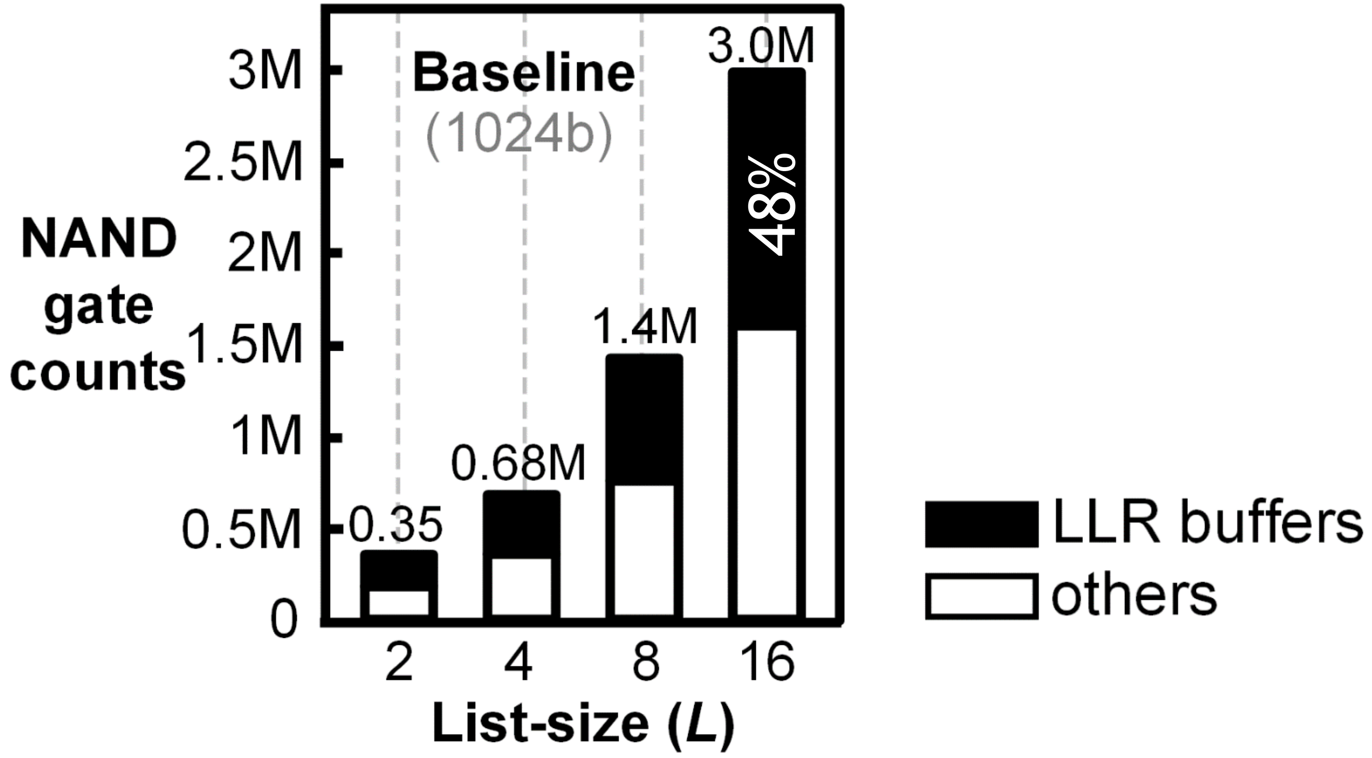
Challenges of Large-list decoders

- Challenge 2 : Huge hardware cost
  - Each decoding tree requires a stage-separated buffer.



Challenges of Large-list decoders

- Challenge 2 : Huge hardware cost
  - Each LLR buffer consists of stage-dedicated buffers.

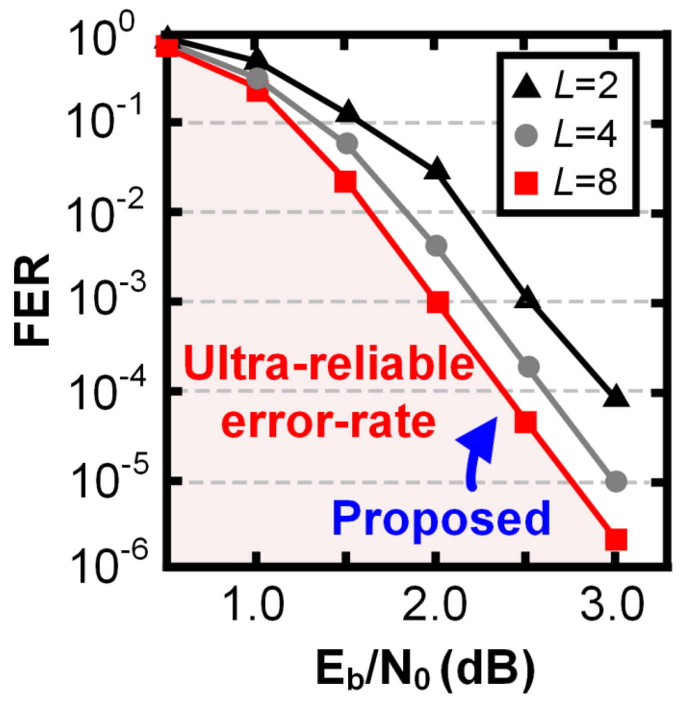


Outline

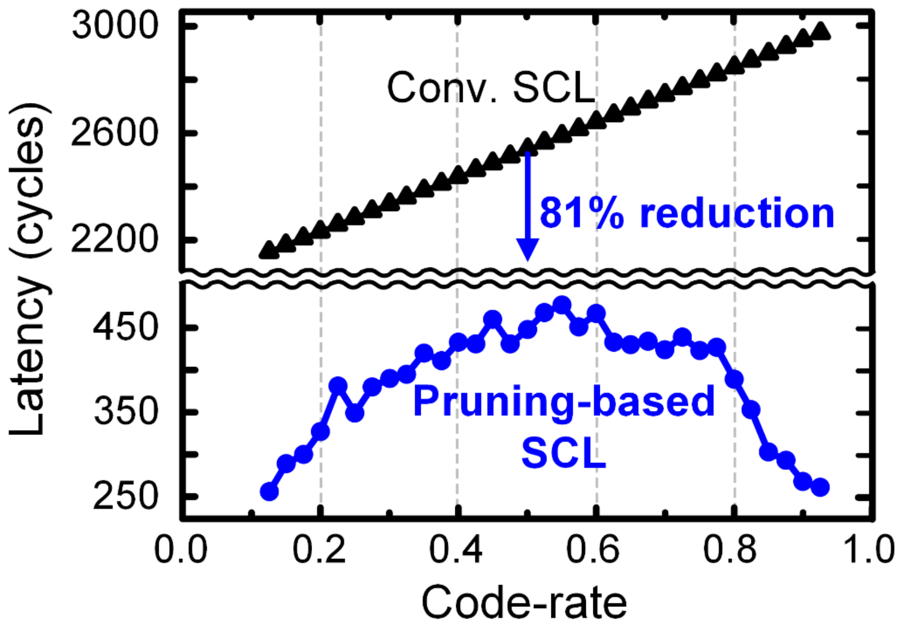
Proposed Design

- Our main contribution is to achieve three performances.

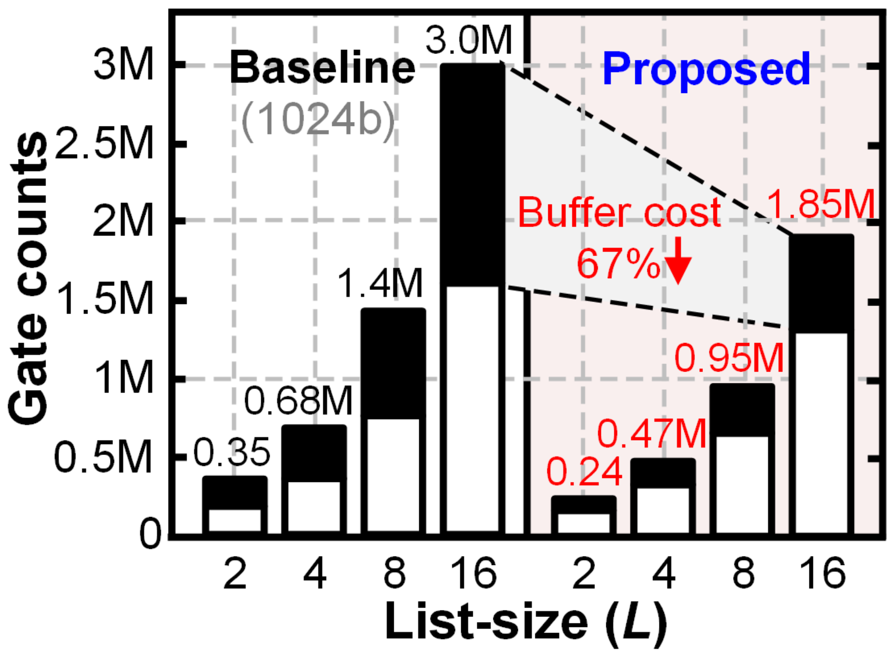
Ultra-reliability
(large-list)



Low-latency
(pruning)

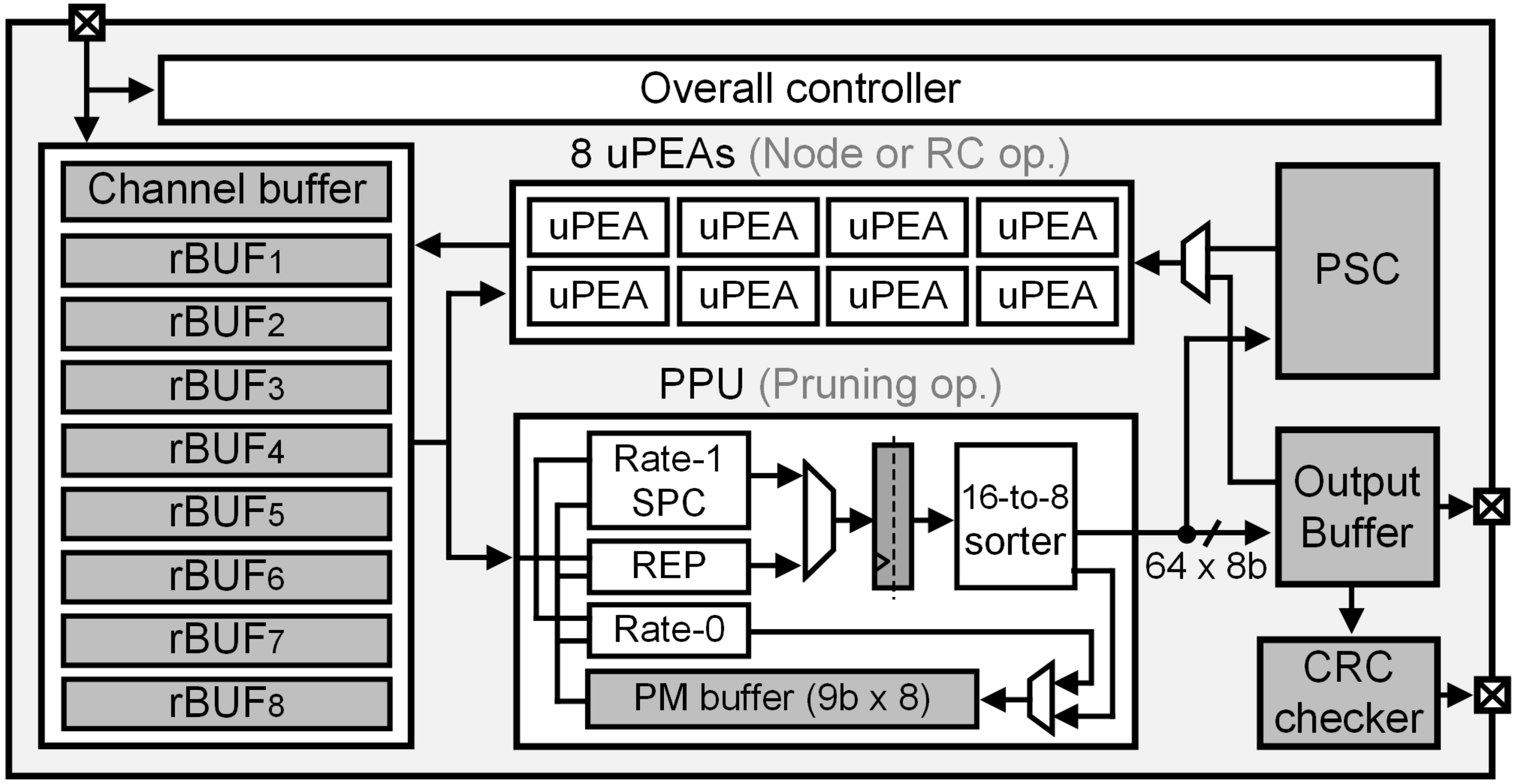


Ultra-low-cost
(Fully reusable buffer)



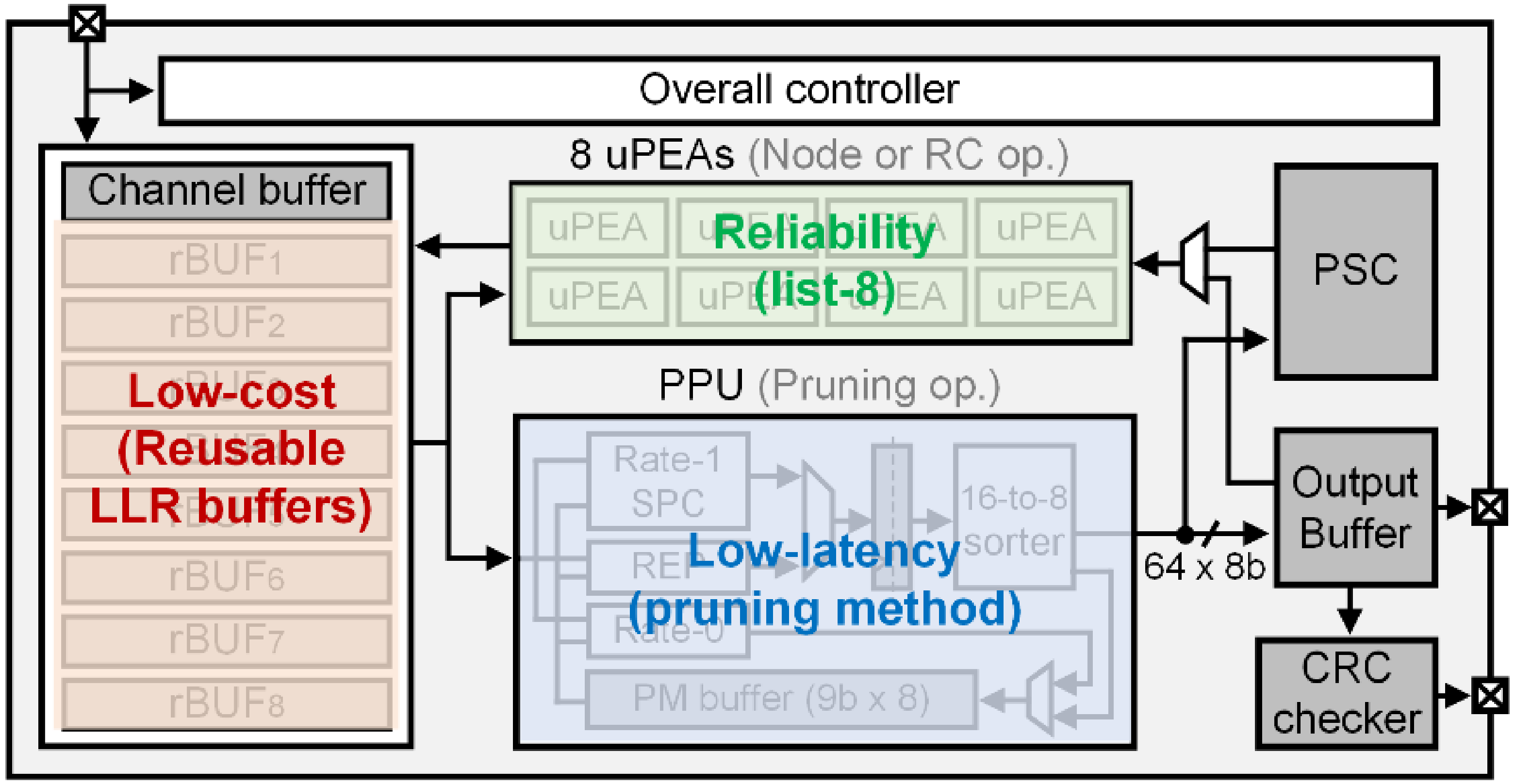
Proposed Design : Overall architecture

- This is our proposed decoder architecture (digital blocks).
  - Our design supports up to 1024b communications.



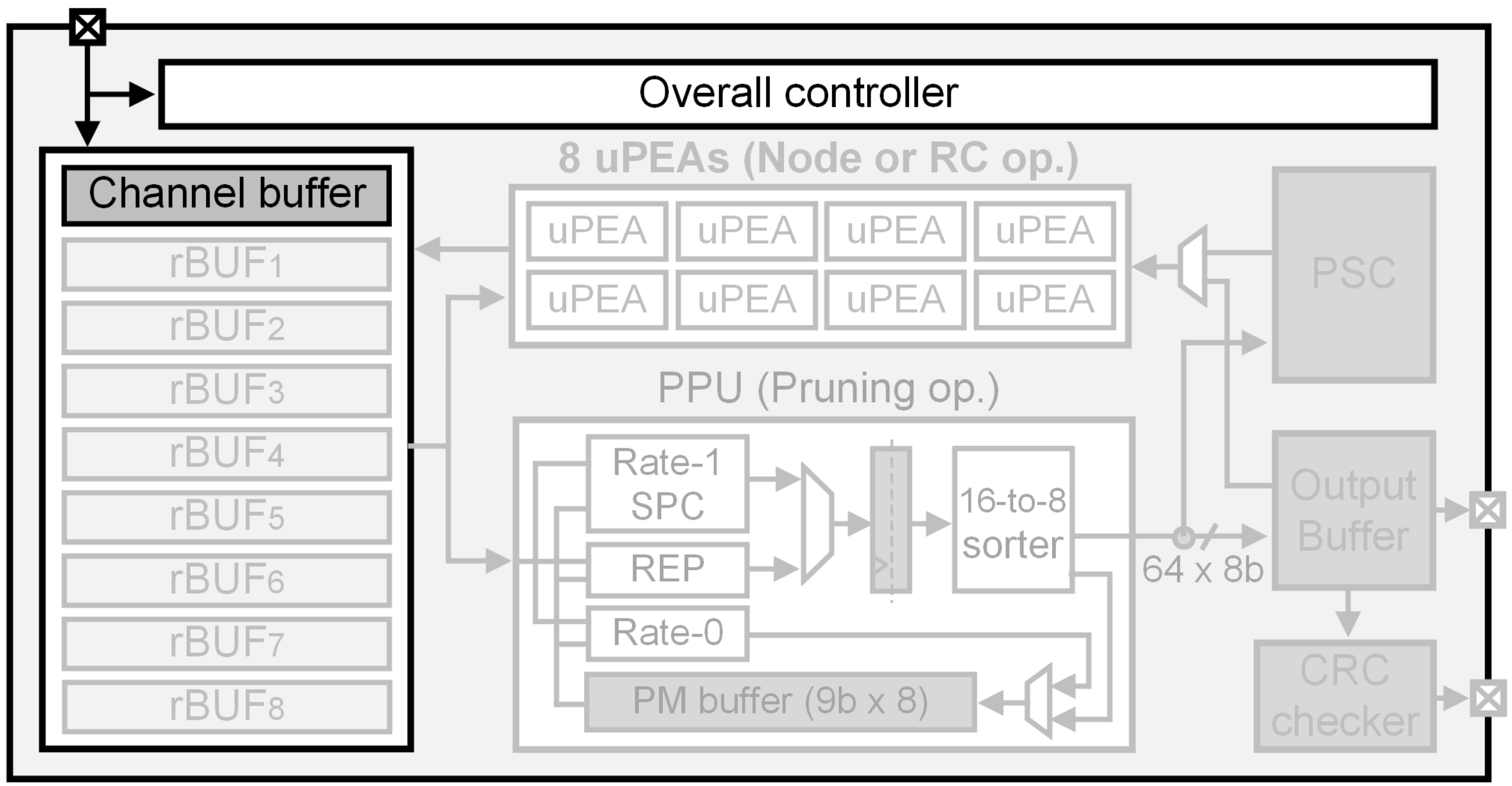
Proposed Design : Overall architecture

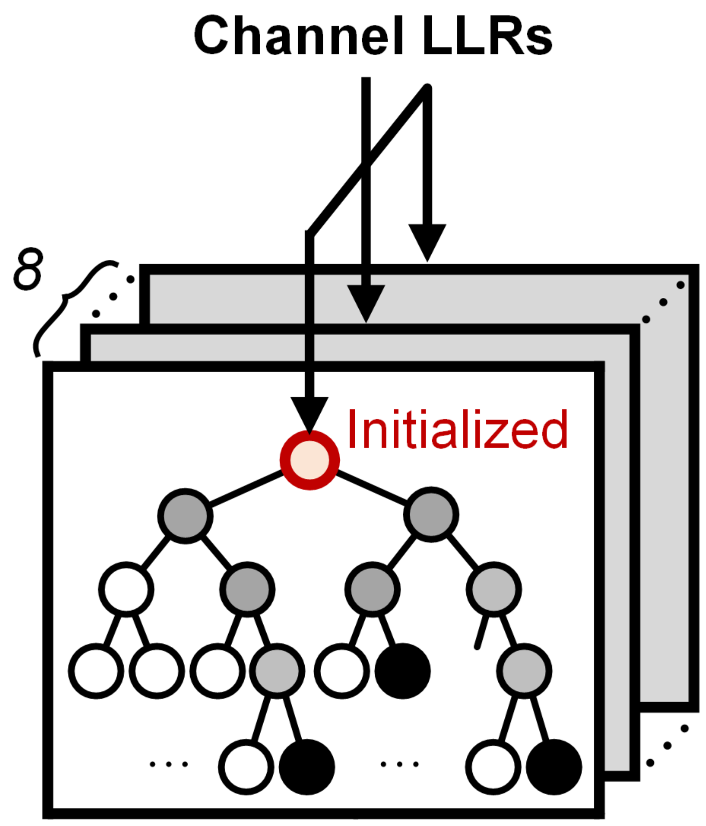
- Which block achieves each performance ?
  - Low-cost (reusable LLR buffers), Reliability (list-8), Low-latency (pruning)



Proposed Design : Decoding procedure

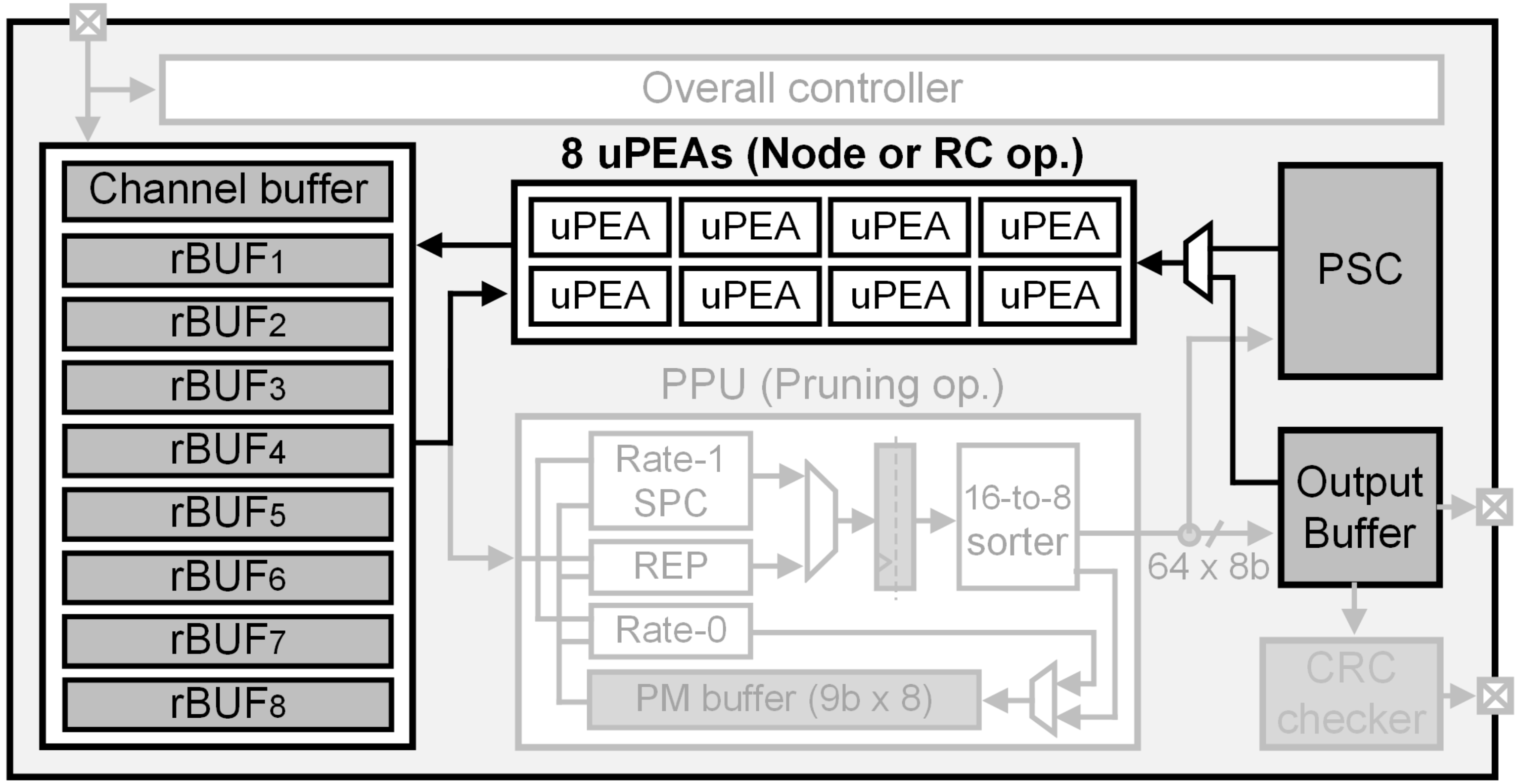
- First, we initialize the channel buffer with channel LLRs.

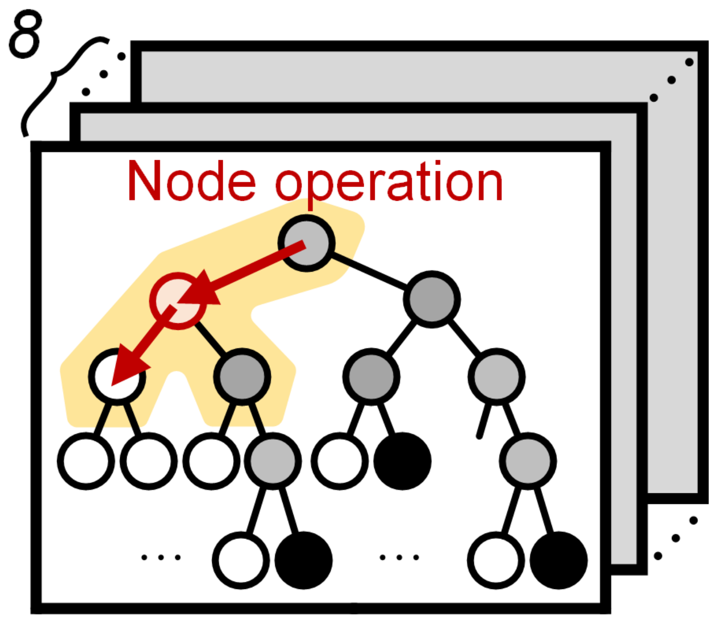




Proposed Design : Decoding procedure

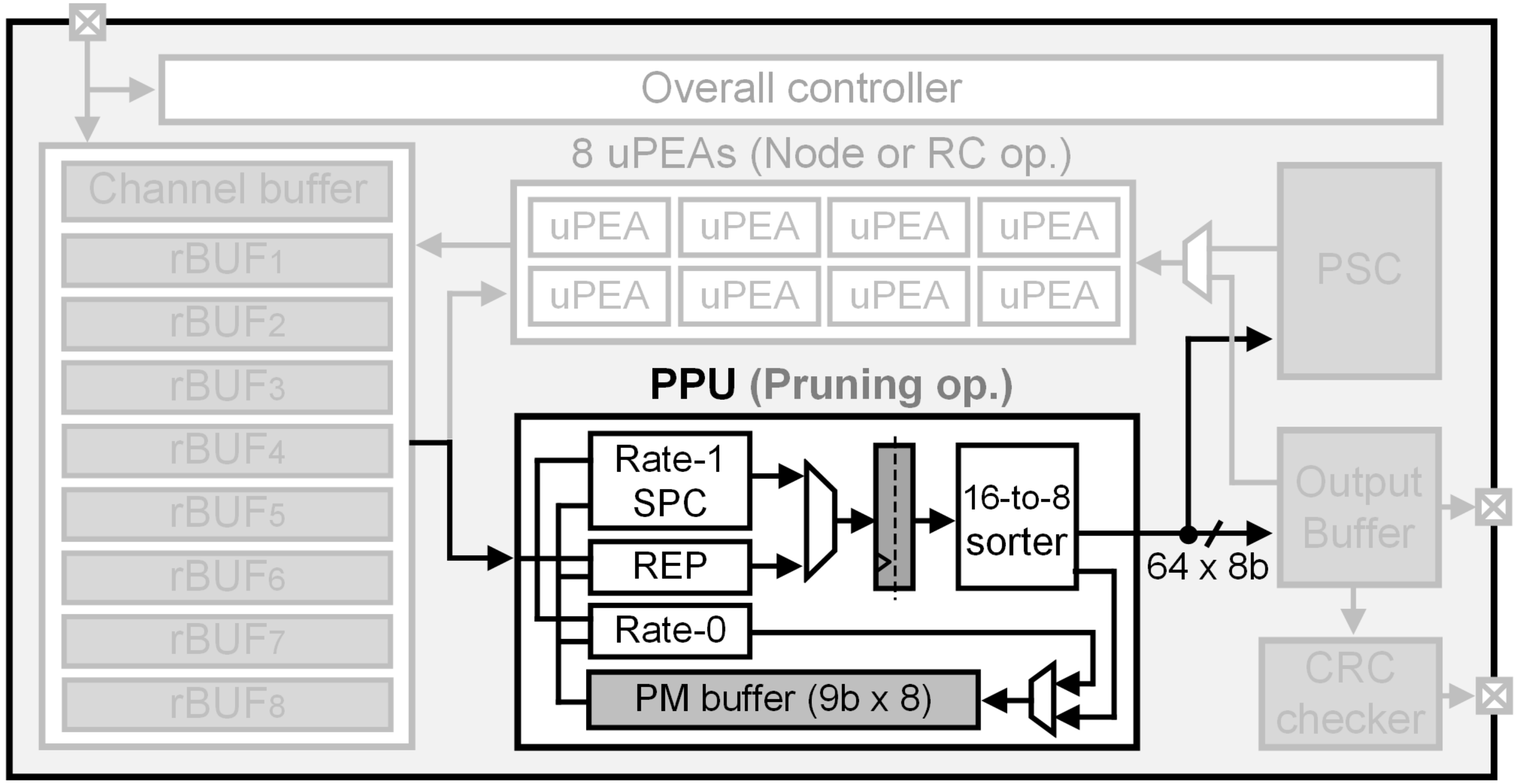
- Our design provides the list-8 SCL decoding.
  - With 8 processing element arrays

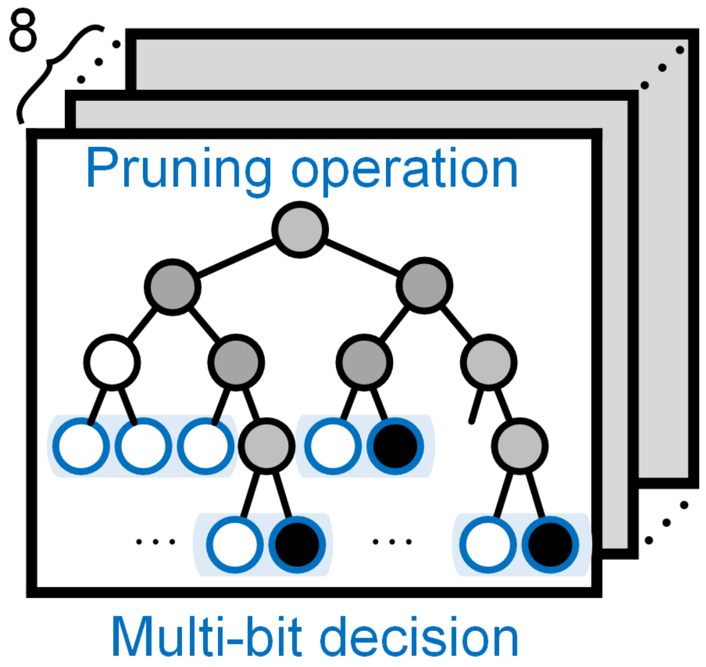




Proposed Design : Decoding procedure

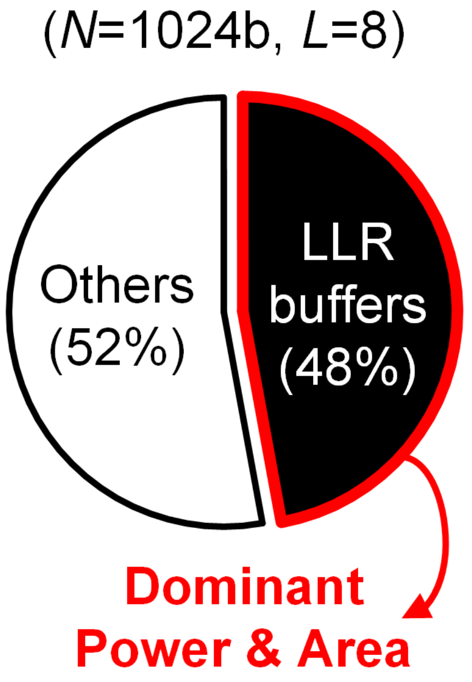
- Our design supports the pruning operations.
  - With the pruning processing unit

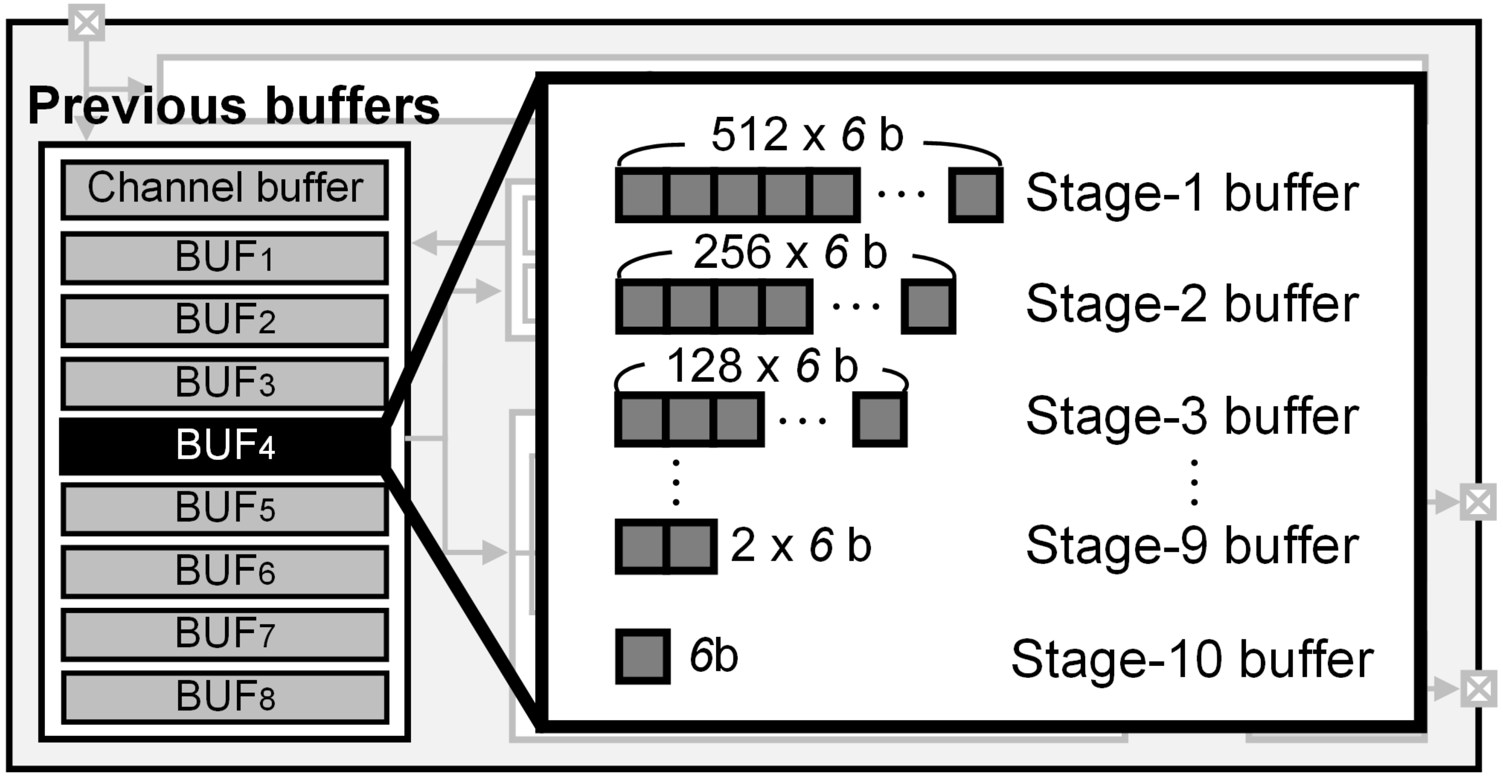




Proposed Design : Previous Buffer

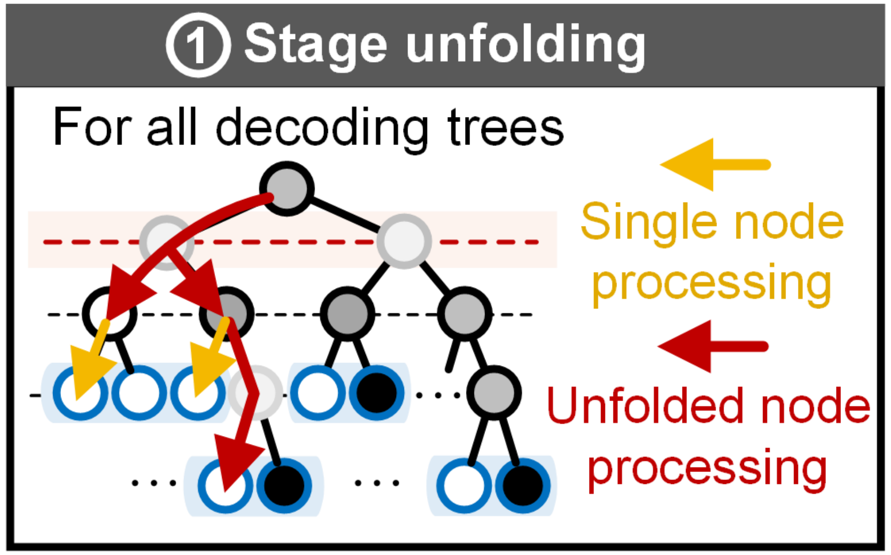
- We need to reduce the buffer overheads.
  - To implement large-list SCL decoder (for ultra-reliability)

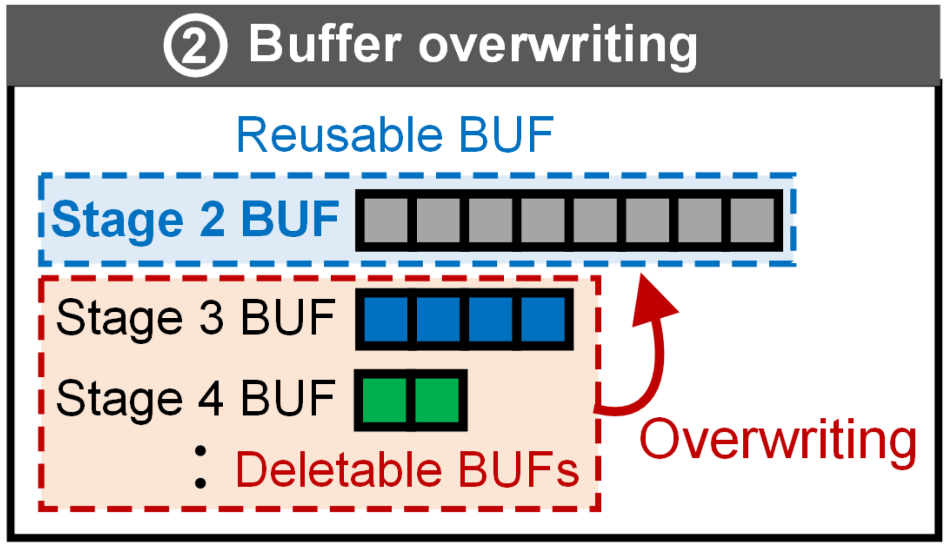




Proposed Design : Buffer reduction methods

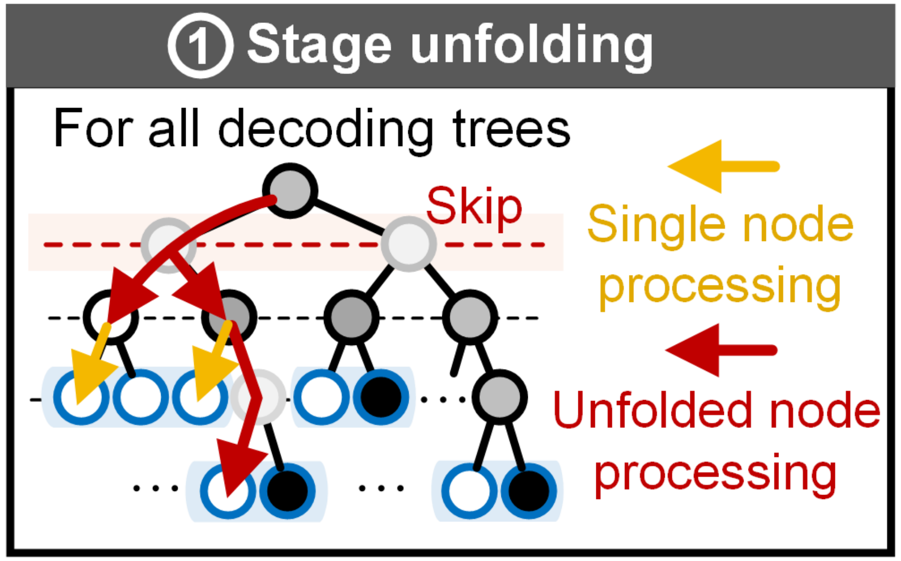
- Fully-reusable LLR-buffers (rBUFs)
  - Two buffer-reduction methods make rBUFs.
  - They reduce the buffer size by **67%** totally.

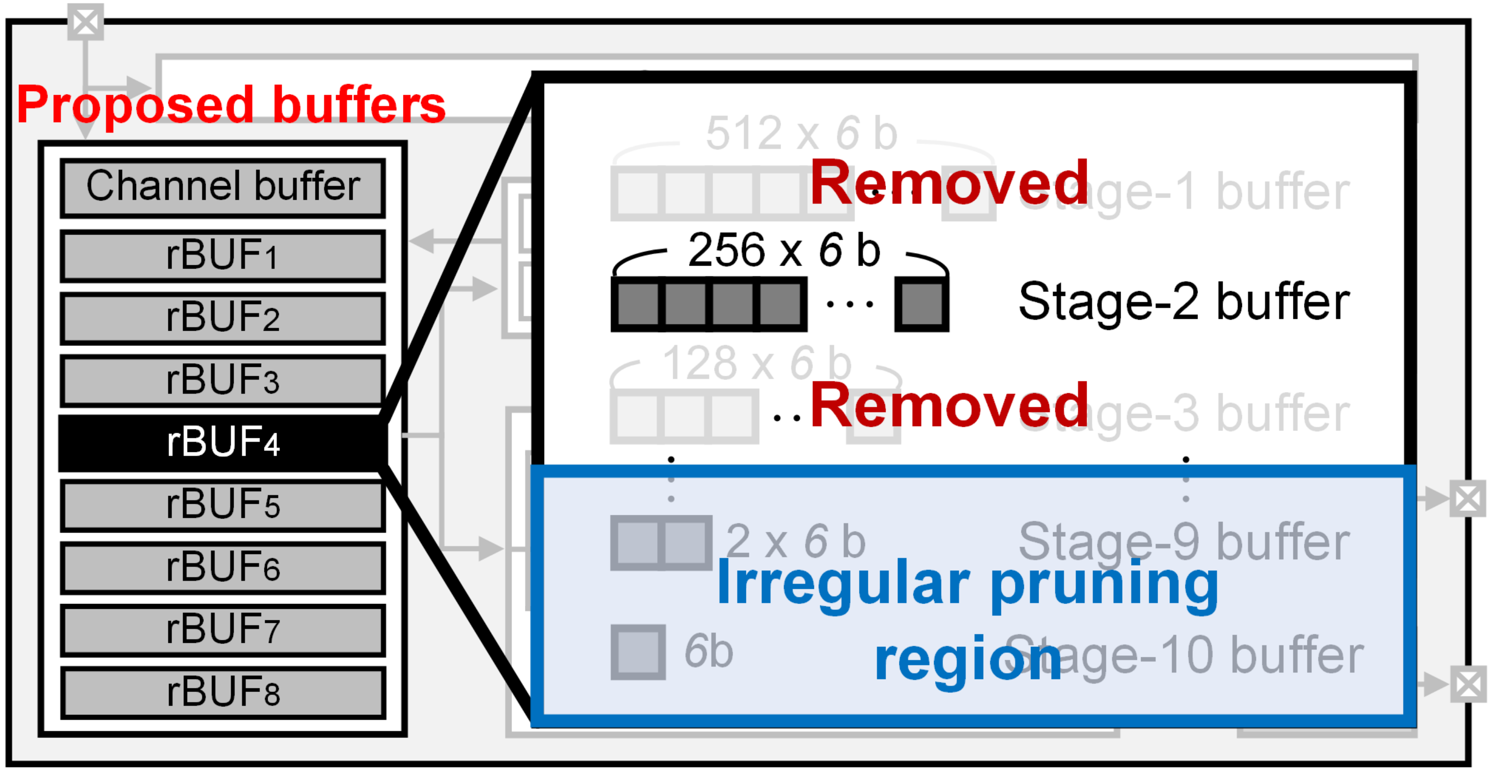




Proposed Design : Buffer reduction methods

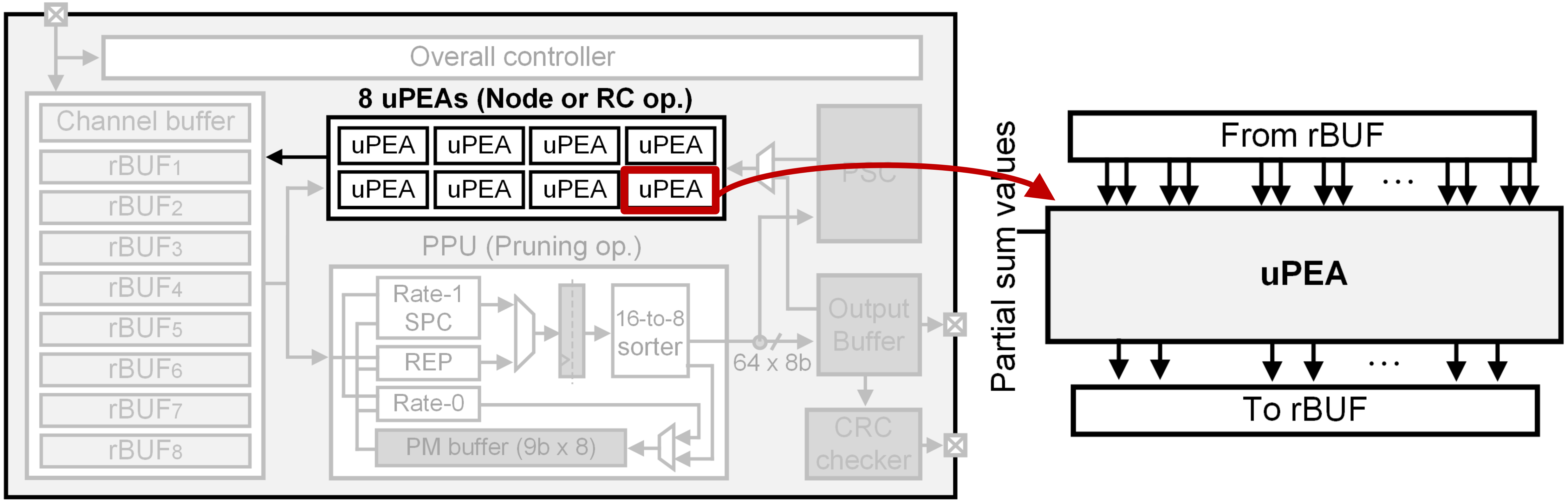
- Method 1 : Stage unfolding
  - This method supports serial node operations at a time. ($\longrightarrow$)





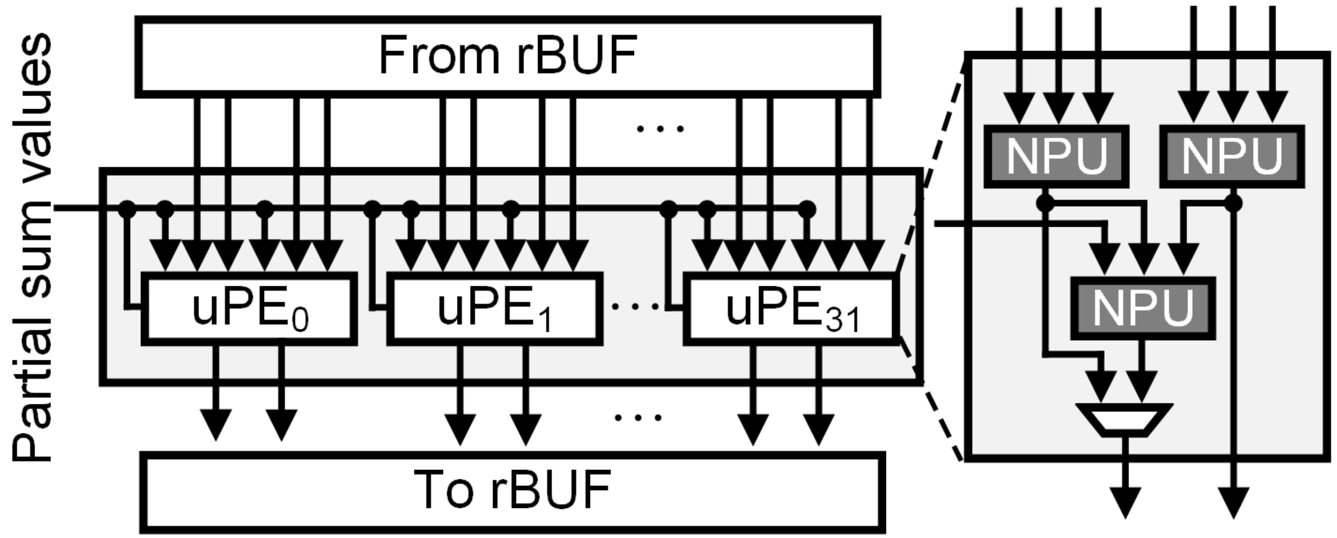
Proposed Design : Buffer reduction methods

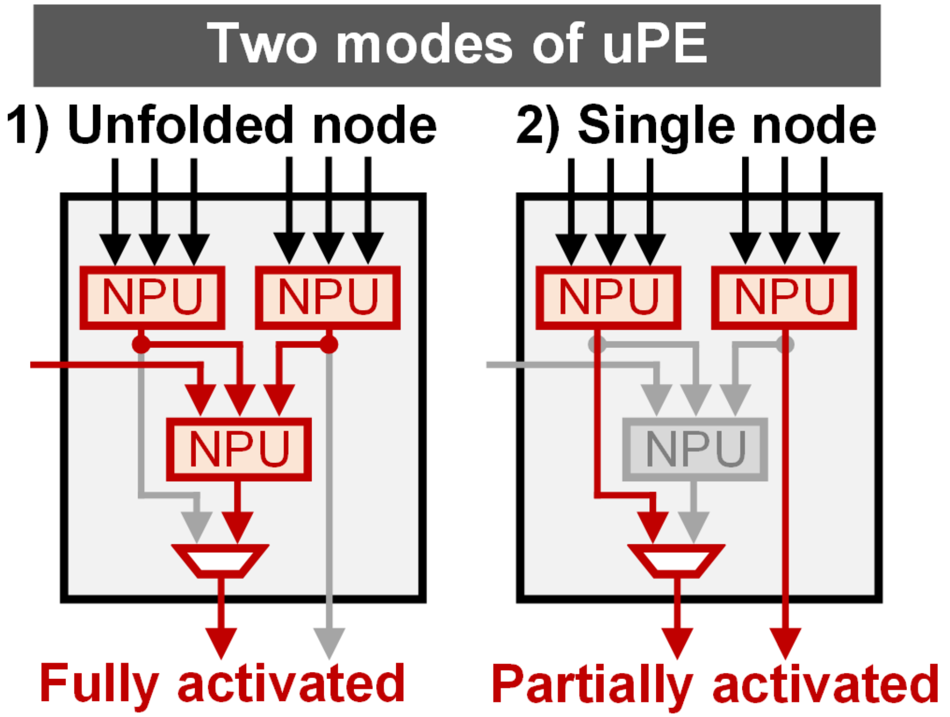
- Method 1 : Stage unfolding
  - We propose **a new node operator** supporting the proposed stage-unfolding.
  - We dedicate 8 unfolded-processing element arrays (uPEAs).



Proposed Design : Buffer reduction methods

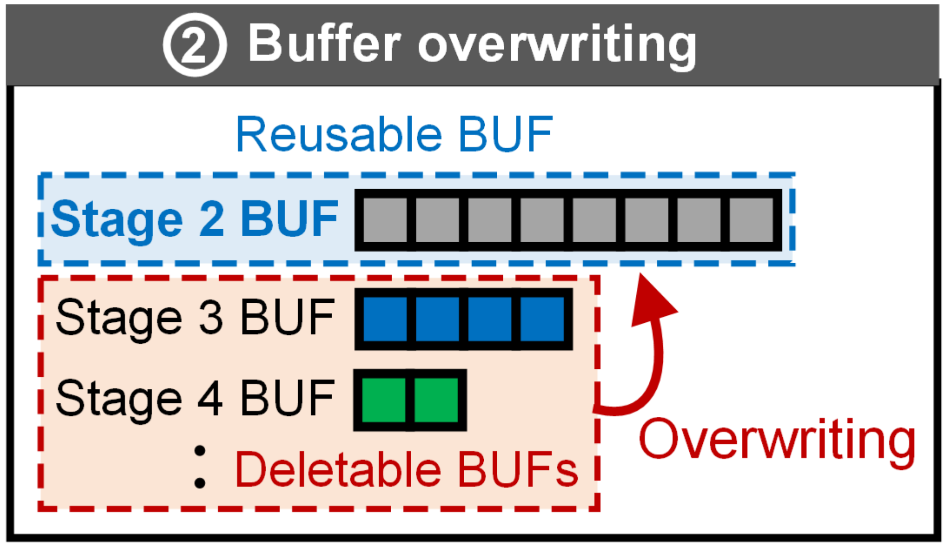
- Method 1 : Stage unfolding
  - Each uPE has three node processing units (NPUs).
  - Each uPE supports two node types.

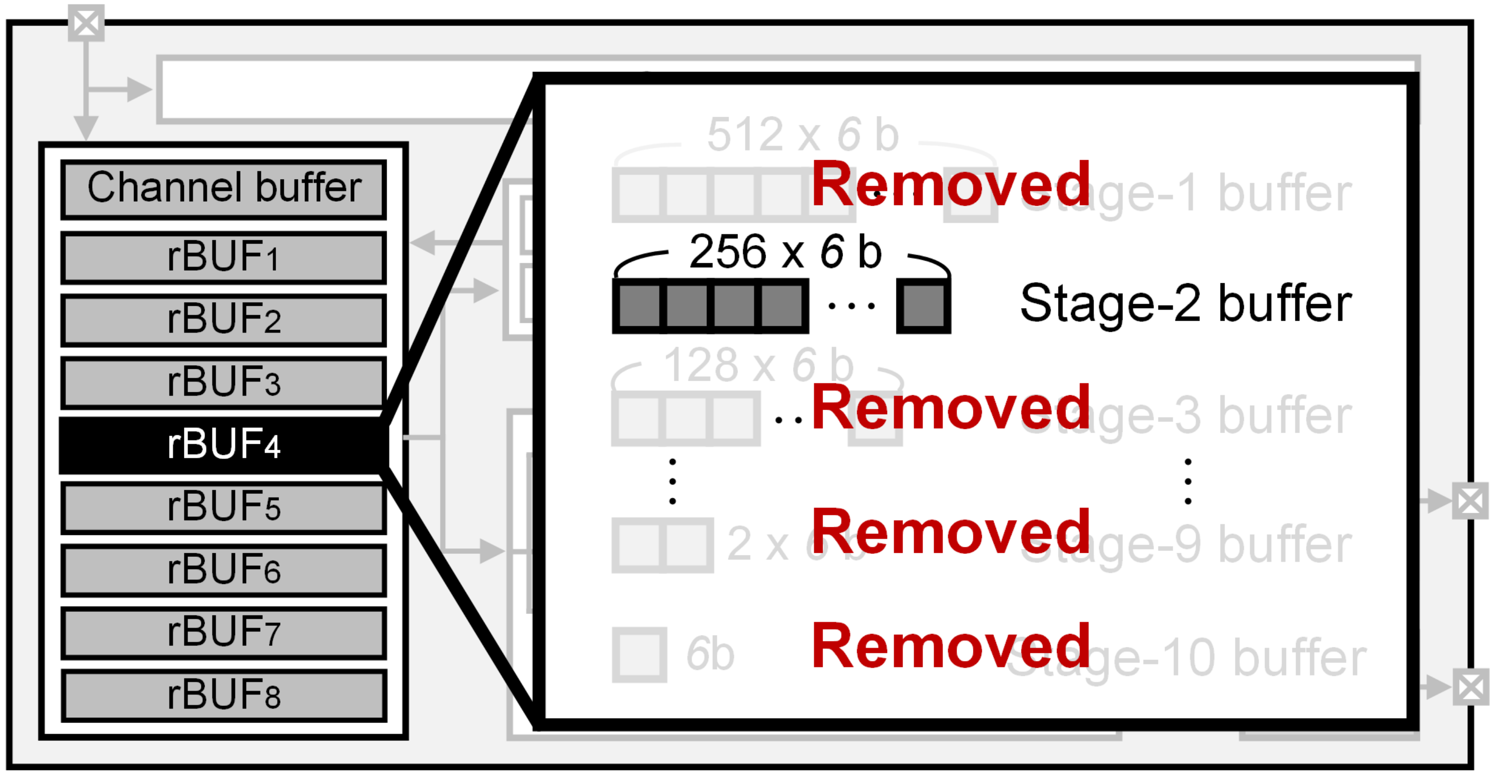




Proposed Design : Buffer reduction methods

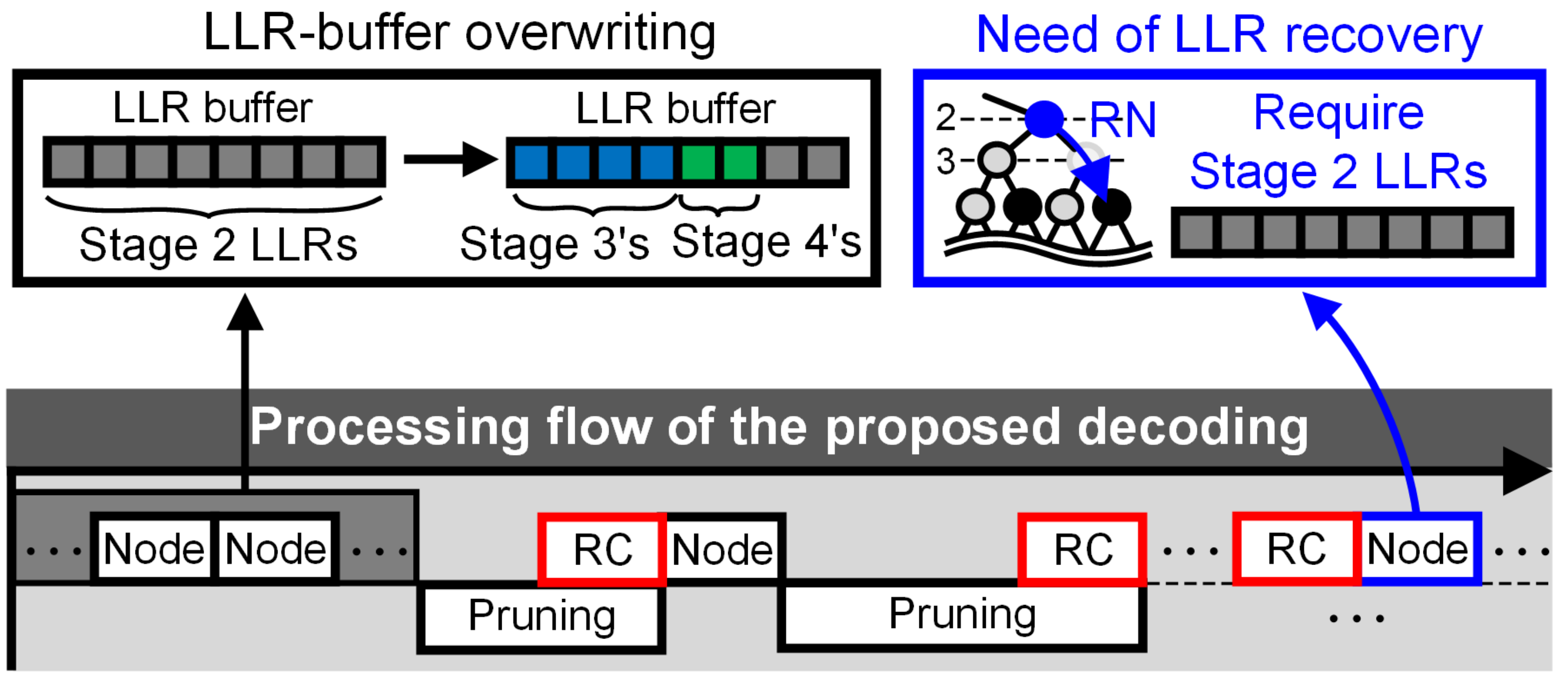
- Method 2 : LLR buffer overwriting + recovery
  - We just **reuse the stage-2 buffer** for all stages.

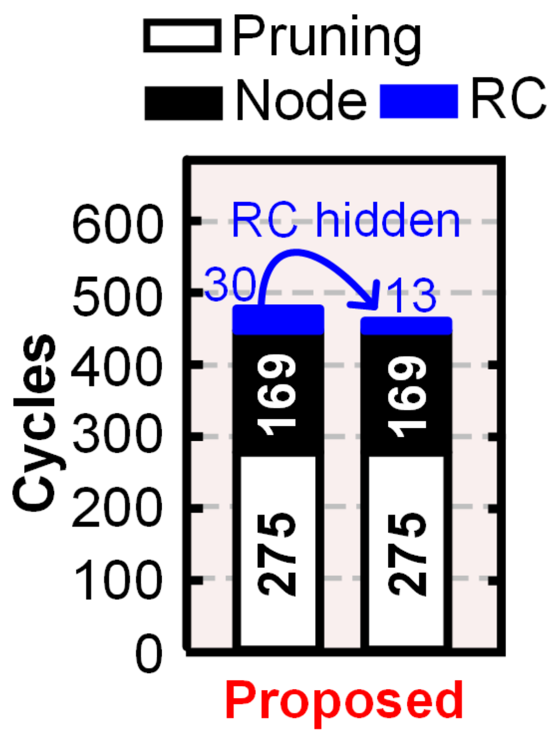




Proposed Design : Buffer reduction methods

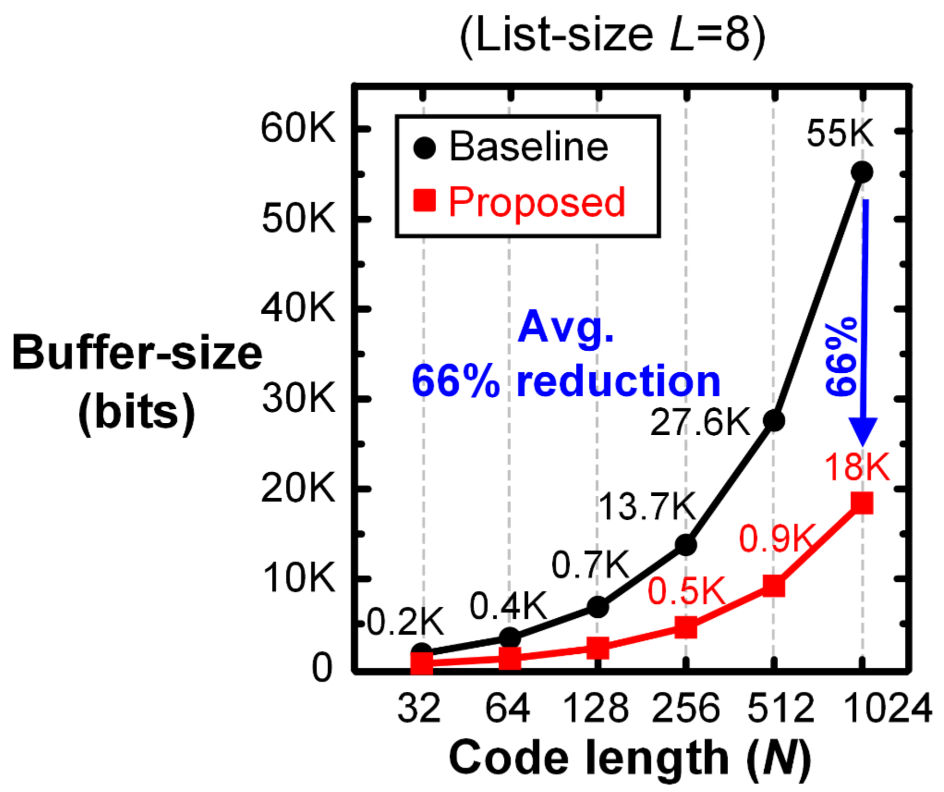
- Method 2 : LLR buffer overwriting + recovery
  - We can hide the recovery processing into no-dependency time.

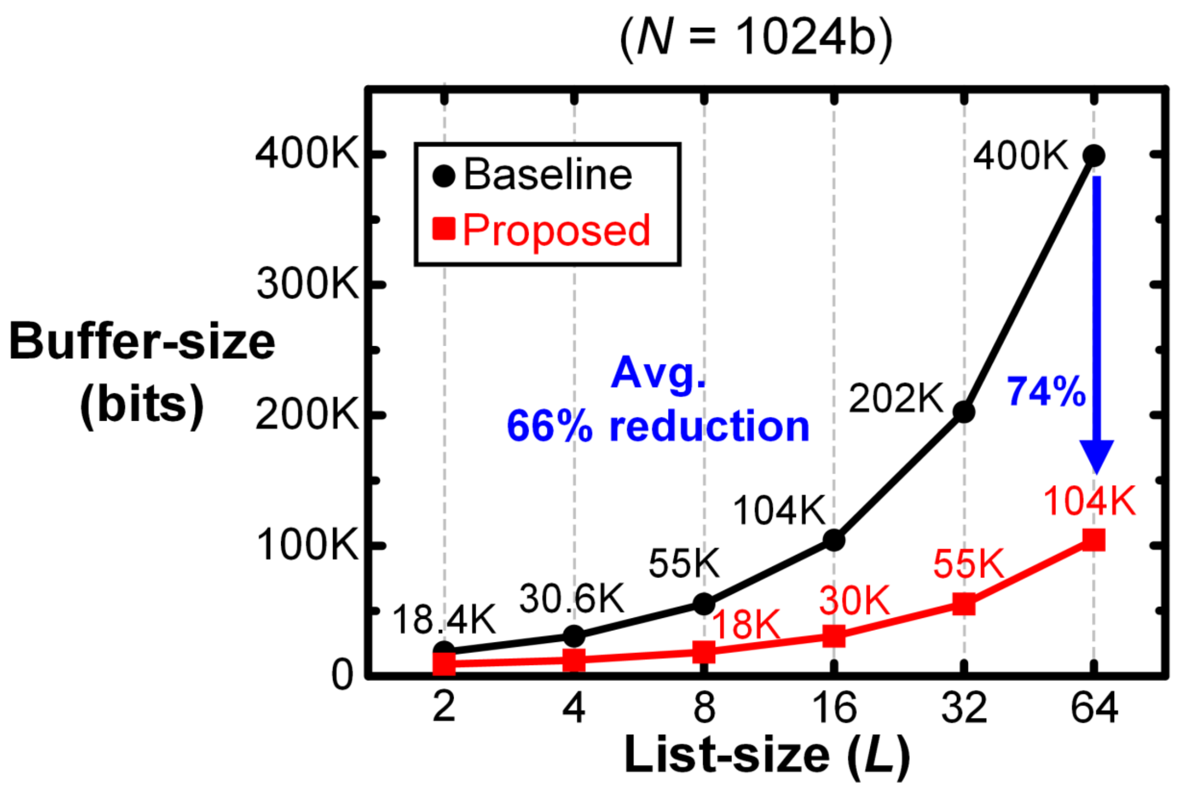




Proposed Design : Buffer reduction methods

- Our methods reduce the buffer overhead by **66%**.
  - For different code lengths and list-sizes



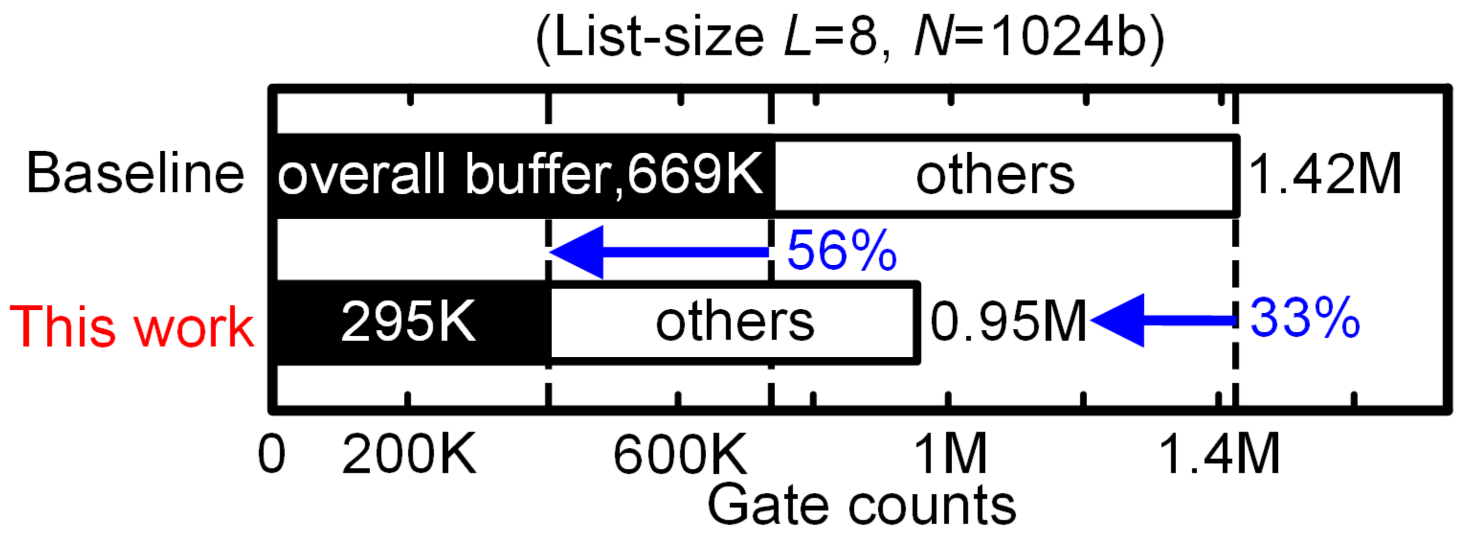


Outline

- Introduction
- Implementation Challenges
- Proposed Design
- **Implementation Results**
- Conclusion

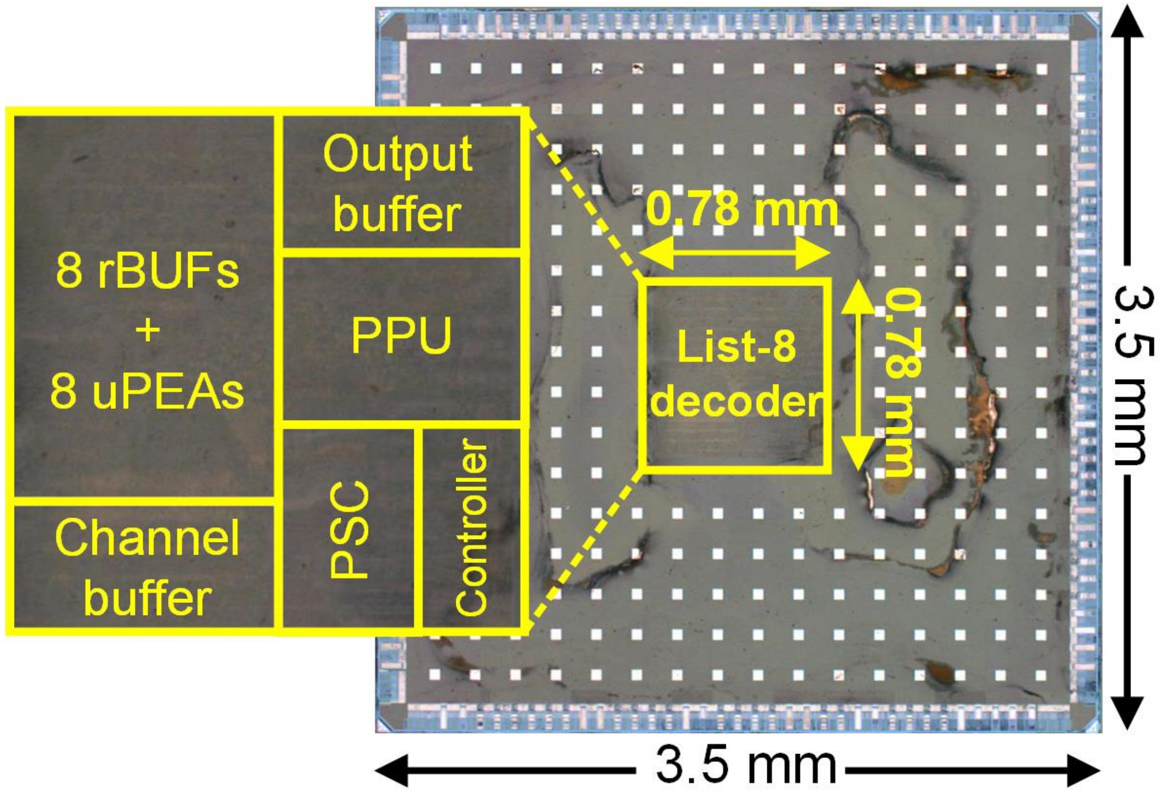
Implementation Results : Prototype decoder

- Our prototype decoder supports the list-8 and $N \leq$ 1024b.
  - The buffer gate-counts are reduced by **56%.**
  - Overall decoder cost is reduced by **33%.**



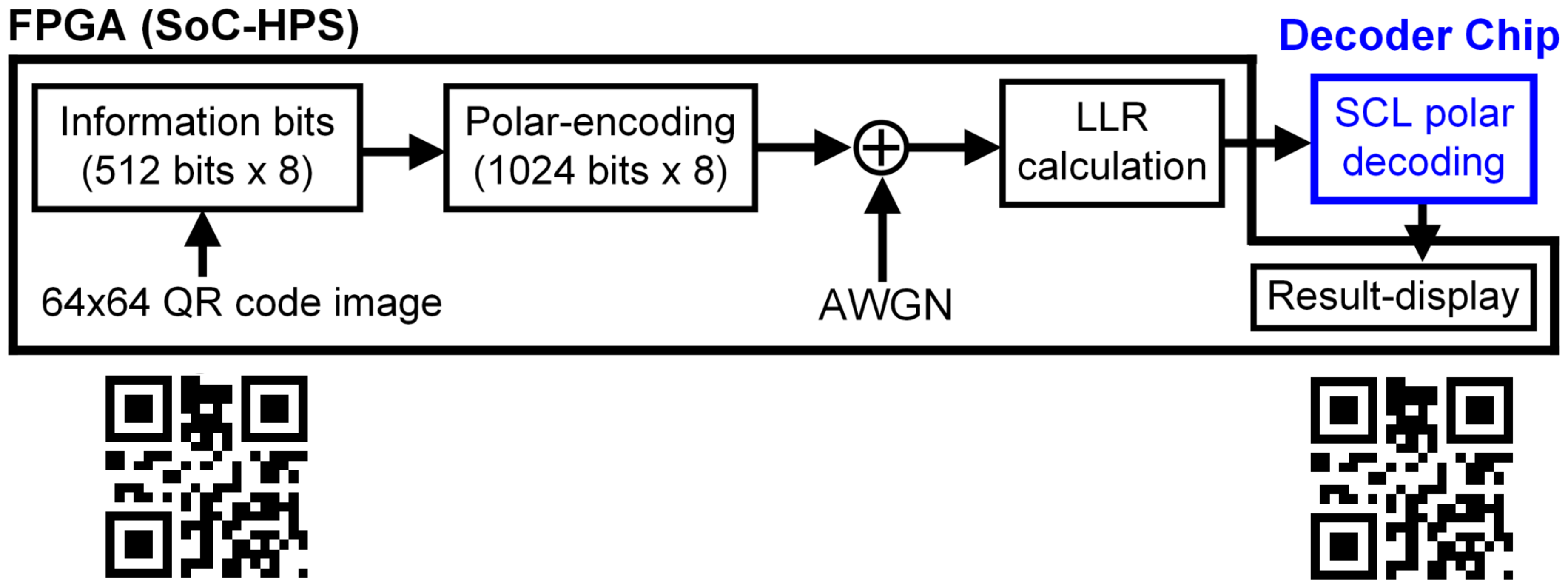
Implementation Results : Chip Microphotograph

- Our decoder chip was implemented in 28nm CMOS.



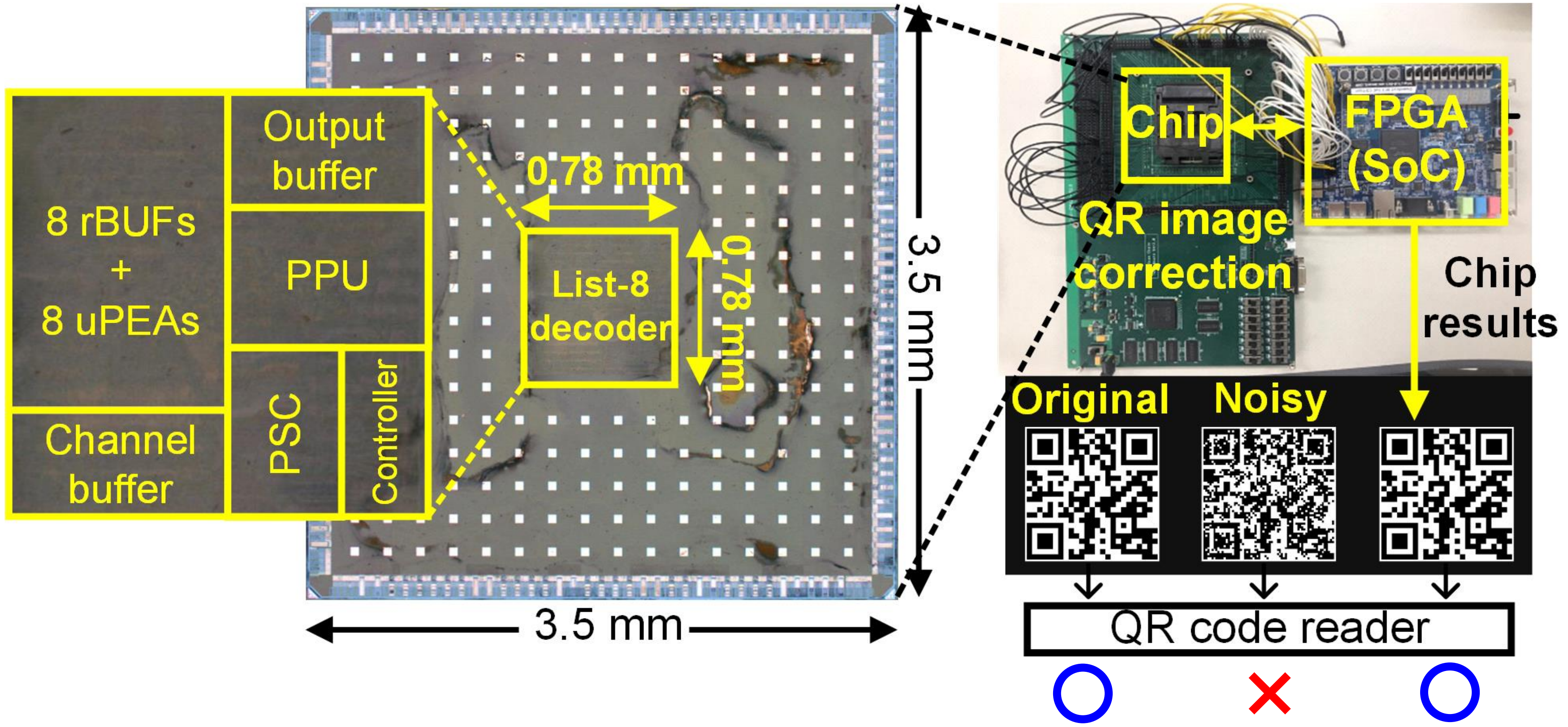
Implementation Results : Chip Verification

- Verification scenario
  - This Verification scenario is to correct a noisy 64 x 64 QR-code image.



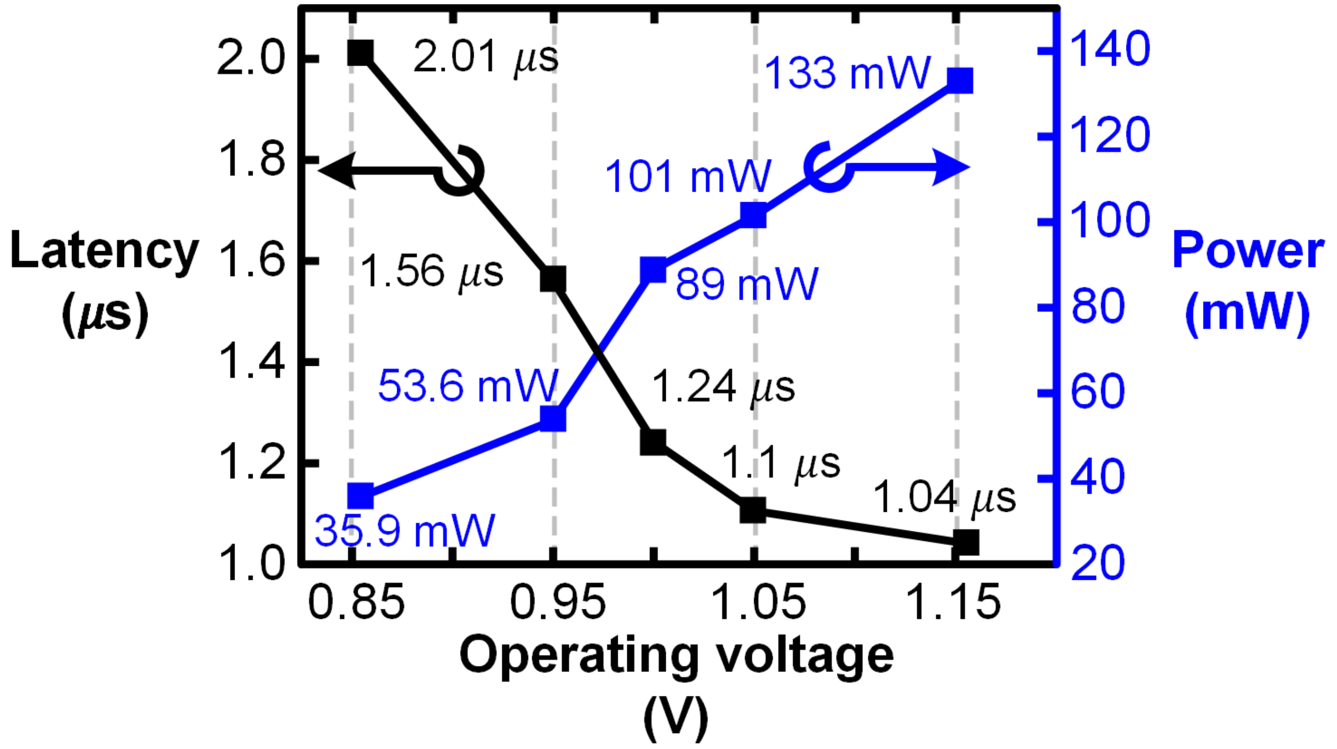
Implementation Results : Chip Verification

- FPGA-based verification platform.
  - Finally, we checked the corrected result is valid for a QR-code reader.



Implementation Results : Performance

- Voltage-scaling results
  – We have measured latency and power in different operating voltages.

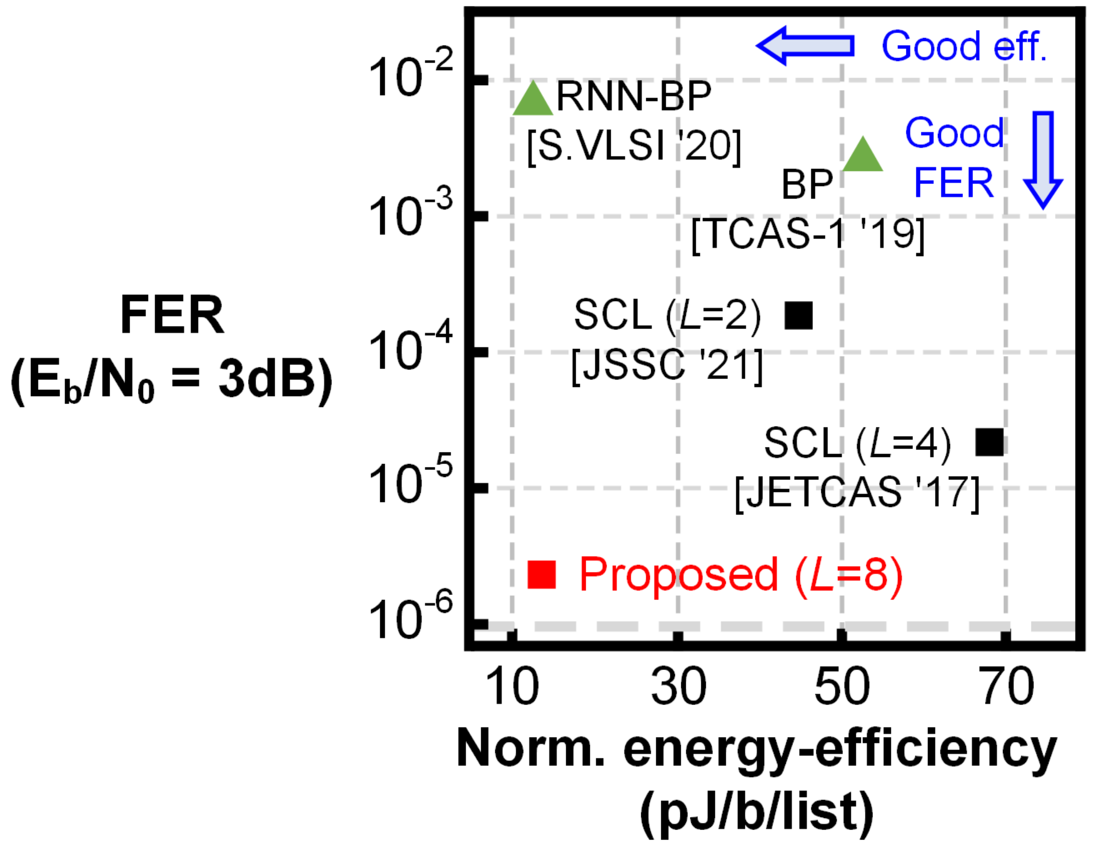


Implementation Results : Comparison Table

	This work	JETCAS '17	JSSC '21	S.VLSI '20	TCAS-I '19
Technology (nm)	28	28	40	40	40
Algorithm	SCL	SCL+SCF	Split SCL	RNN-BP	BP
Code length (bits)	up to 1024	1024	up to 1024	up to 256	1024
Code rate	variable	variable	variable	variable	0.5
List size	**8**	**4**	**2**	**1**	**1**
E_b/N_0 for FER=10^{-5}	**2.75**	**3.25**	**3.6**	**6.0**	**4.8**
Pruning	**O**	**X**	**X**	**X**	**X**
Core area (mm^2)*	0.595	0.44	0.317	0.23	1.11
Voltage (V)	1.05	1.3	0.9	0.9	0.9
Latency (μs)**	1.1	3.34	2.26	0.31	N/A
Throughput (Mb/s)**	925	307	453	823	7610
Power (mW)**	101.4	128.3	42.8	12.8	422.7
Area-efficiency (Gb/s/mm^2)**†	1.56 **[12.5]**	0.69 [2.76]	4.17 [8.3]	10.4 [10.4]	19.9 [19.9]
Energy-efficiency (pJ/b)**†	109.5 **[13.7]**	272.9 [68.2]	89.9 [44.9]	12.9 [12.9]	52.9 [52.9]

Implementation Results : Evaluation Chart

- Our chip design achieves both the best energy-efficiency and FER.



Conclusion

- **Cost-efficient large-list SCL decoder**
  - Supports the pruning-based decoding (ultra-low-latency).
  - Supports the list-8 SCL decoding (ultra-reliability).
  - Supports the proposed reusable LLR buffers (low-cost).

- **Implementation in 28nm CMOS**
  - Achieves 1.1μs by supporting the pruning-based decoding.
  - Achieves 12.5 Gb/s/mm^2/list and 13.7 pJ/b/list due to the reusable buffers.
  - Be Successfully demonstrated by a QR-image and a FPGA-based platform.
  - Provides the best solution for 5G and next generation URLLC scenarios.

Thank you

QnA