Efficient Hardware Designs for Hyperdimensional Learning

Neuroscience 299: Module 11
November 10, 2021

Mohamed S. Ibrahim

Berkeley Wireless Research Center

About Me

- BS Electrical Engineering: Ain Shams University, Cairo, Egypt: 2010
- MS Electrical Engineering: Ain Shams University, Cairo, Egypt: 2013
- MS Electrical and Computer Engineering, Duke University, Durham, NC: 2017
- PhD Electrical and Computer Engineering, Duke University, Durham, NC: 2018
  - Research: Design and optimization of “cyber-physical microfluidic biochips/systems”
- Senior SoC Design Engineer at Intel Corporation, Santa Clara, CA: June 2018—May 2021
- Postdoc at Berkeley Wireless Research Center since June 2021
  - Current research: Exploring efficient hardware solutions for HD computing beyond supervised learning, among others.

Topics

- Motivation and Goals
- Background
- Specialized HD Architectures
- Nearly General-Purpose HD Encoding Architectures

Digital Hardware

- Introducing Analog Computing and Non-Volatile Memories

Analog Hardware

Motivation and Goals

[Slides Courtesy of Jan M. Rabaey, UC Berkeley]

Rethinking Computing…

The nature of computing is changing

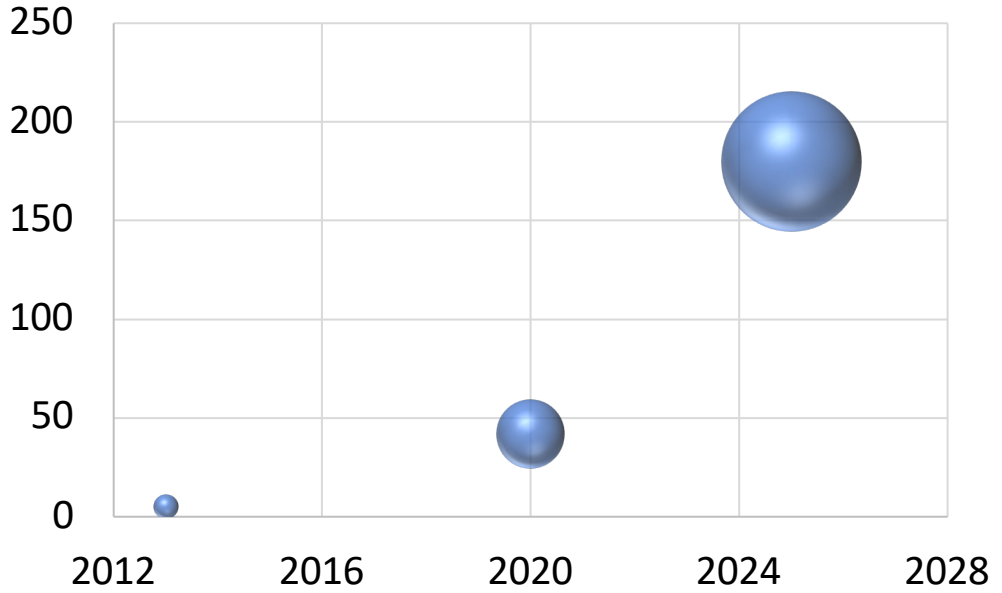
- Programming driven by data and learning, not algorithms
- Truly ubiquitous (smart world, smart humans, …)

While technologies of old are plateauing

- Traditional computer architecture limited by interconnect
- Variability and leakage constraints limit energy scaling
- Limitations of deterministic computing paradigm

From IoT to IoA

Digitization of society – an extremely rapidly accelerating phenomenon

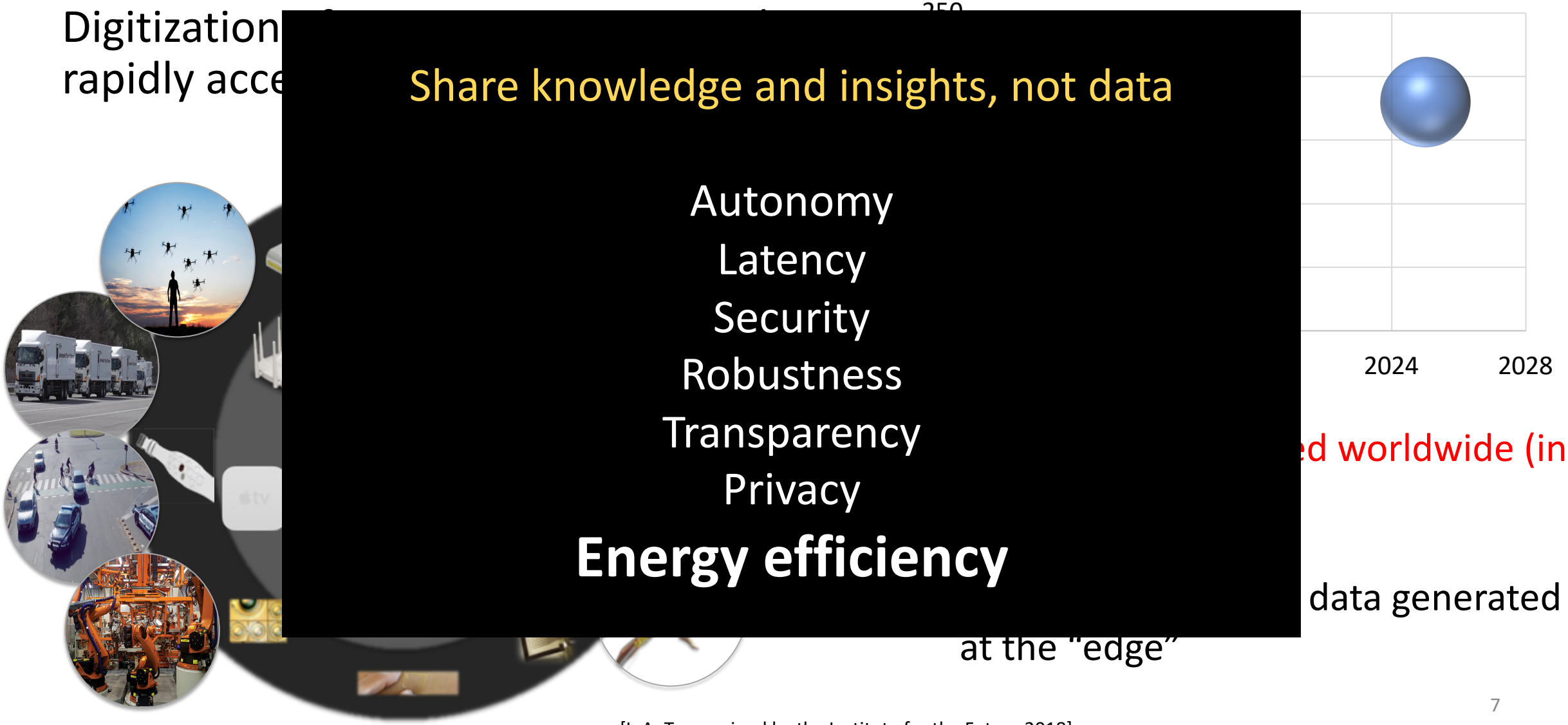


- Digital data generated worldwide (in ZettaBytes)
- Huge fraction of this data generated at the “edge”

[IoA: Term coined by the Institute for the Future 2018]

From IoT to IoA

Digitization
rapidly acce



Share knowledge and insights, not data

Autonomy
Latency
Security
Robustness
Transparency
Privacy
Energy efficiency

ed worldwide (in

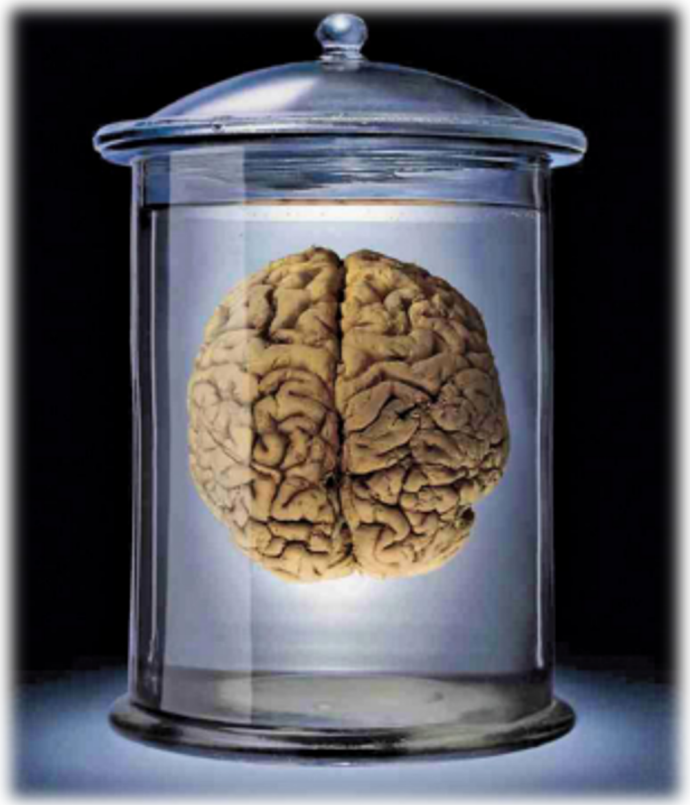
data generated

at the "edge"

[IoA: Term coined by the Institute for the Future 2018]

The Neuroscience Promise

An Amazing Computational Engine



2-3 orders more efficient than today's silicon equivalent (>10^{16} FLOPS with ~20 W)

Robustness in presence of component failure and variations

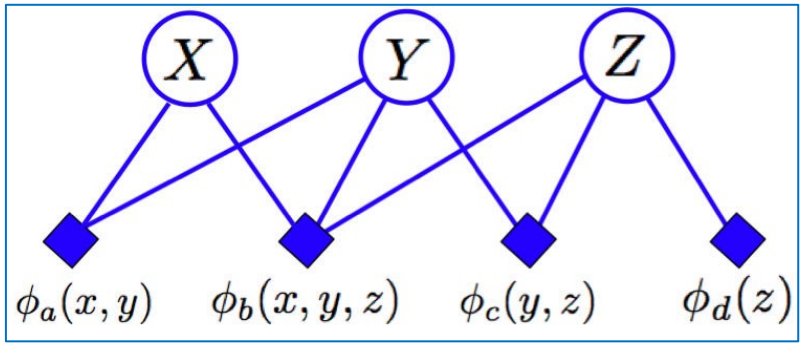
- Neural response is highly variable

Amazing performance with mediocre components

- E.g., sensory pathways– auditory, olfactory, vision, …

Still marginally understood, let alone "cloned"

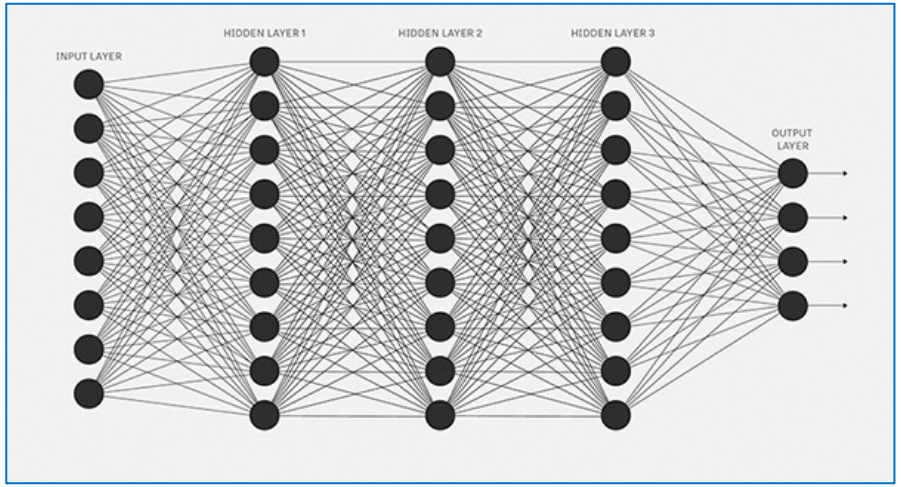
Learning-Based Computational Models



Bayesian Machine learning

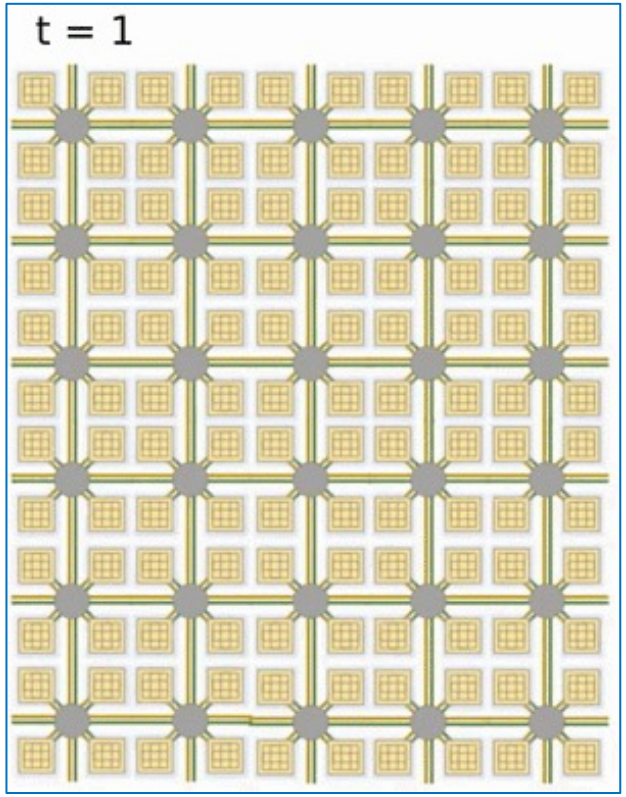
(Believe propagation, reinforcement learning, graphical models, support-vector machines)

Model building non-trivial
Executed on standard processors (graph analytics)



Neuromorphic computing

Bottom-up, Networks of neurons

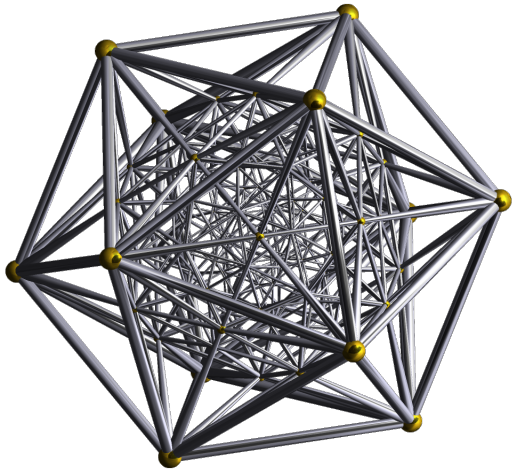


Deep Neural Nets

Learning compute and data hungry
Separate from inference
Complex

High-dimensional computing (Holographic)

Computing with patterns, one-shot learning



Brain-Like Principles:

1- Memory and computing are interconnected

Memory-centric algorithms: Highly parallel; Approximate computing

2- Resilient computing (Redundancy as a feature)

Brain-like robust algorithms: Low SNR; High variability

Brain-Like Principles:

1- Memory and computing are interconnected

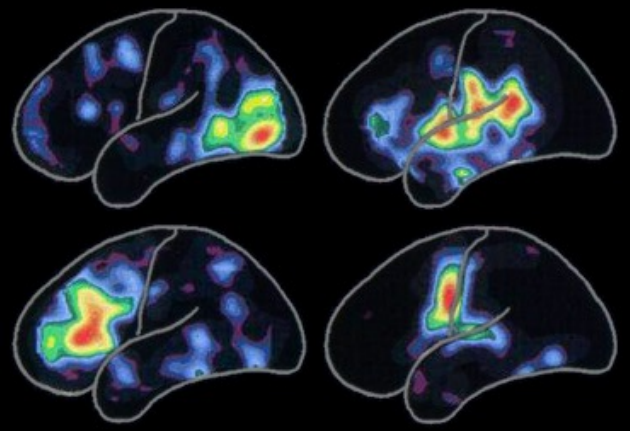
Memory-centric algorithms: Highly parallel; Approximate computing

2- Resilient computing (Redundancy as a feature)

Brain-like robust algorithms: Low SNR; High variability

Hyperdimensional vectors (N > 1000) as basic computational symbols

- represent patterns rather then numbers
- can be *approximate – that is,* can be compared for similarity



Mathematical properties of high-dimensional spaces
in remarkable agreement with behaviors observed in brain

Background

[Slides Courtesy of Mohsen Imani, UC Irvine]

HD Operations: Bundling & Binding

1 Bundling (+): Represents a set

$$H = [V_1 + V_2 + V_3]$$

$\delta\langle H, V_1\rangle >> 0$ V_1 in H ✓

$\delta\langle H, V_4\rangle \simeq 0$ V_4 in H ✗

D=10,000

V_1 = [−1 +1 −1 −1 −1 −1 −1 ...]
V_2 = [+1 −1 +1 +1 +1 +1 −1 ...]
V_3 = [−1 −1 −1 +1 +1 +1 −1 ...]
V_4 = [+1 +1 −1 −1 −1 +1 +1 ...]

Nearly-orthogonal vectors
$\delta\langle V_i, V_j\rangle \simeq 0$

Bundling is like a memory: remember the input information

What's the *H* capacity? $\mathbf{H} = \mathbf{V}_1 + \mathbf{V}_2 + \cdots + \mathbf{V}_P$

2 Binding (*): associative two information

$H = A * R$

A **Orthogonal**
$\delta\langle H, A\rangle \simeq 0$
$\delta\langle H, R\rangle \simeq 0$

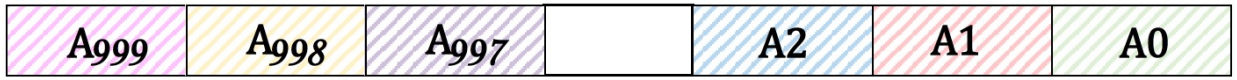
B **Reversible**
$H * A = R$
$H * R = A$

A = [−1 +1 −1 +1 −1 −1 −1 ...]
R = [+1 −1 +1 +1 +1 −1 −1 ...]
*
H = [−1 −1 −1 +1 −1 +1 +1 ...]

HD Operations: Permutation

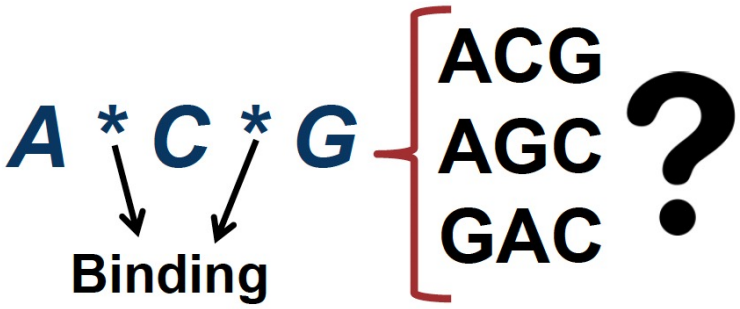
❸ Permutation (ρ):

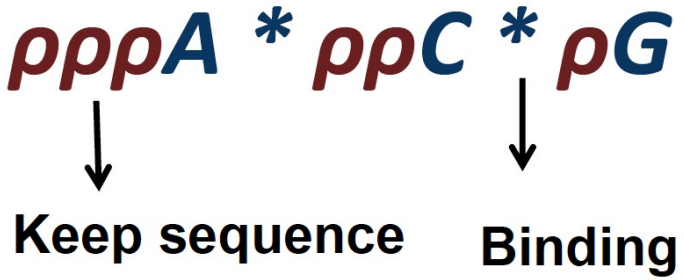
Makes a dissimilar vector by rotating: $\delta\langle A, \rho A\rangle \simeq 0$



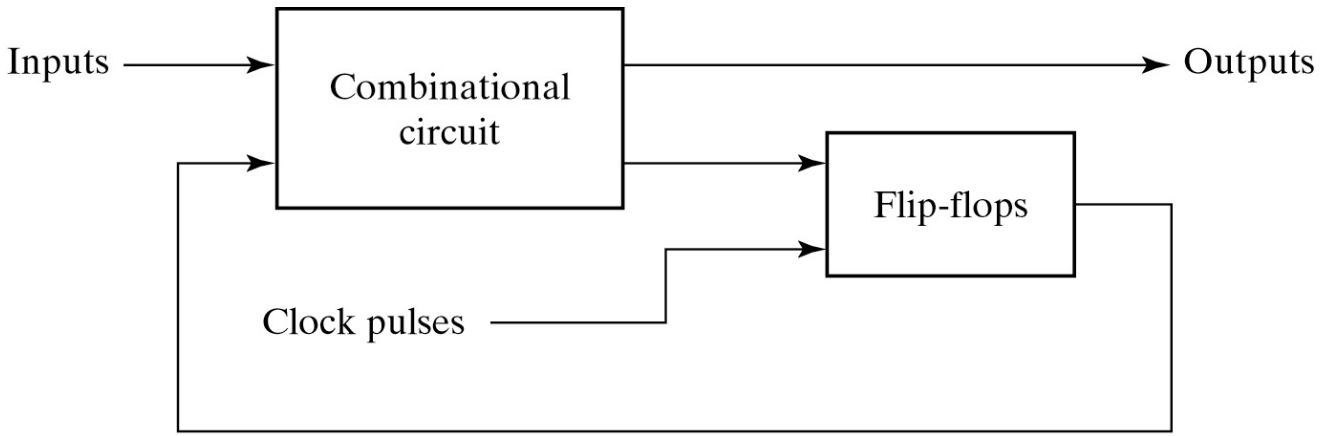
Good for representing sequences

- A trigram "ACG" is encoded by:

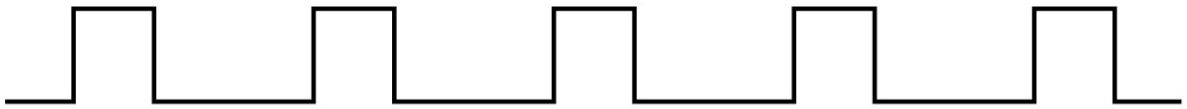




Digital Circuits



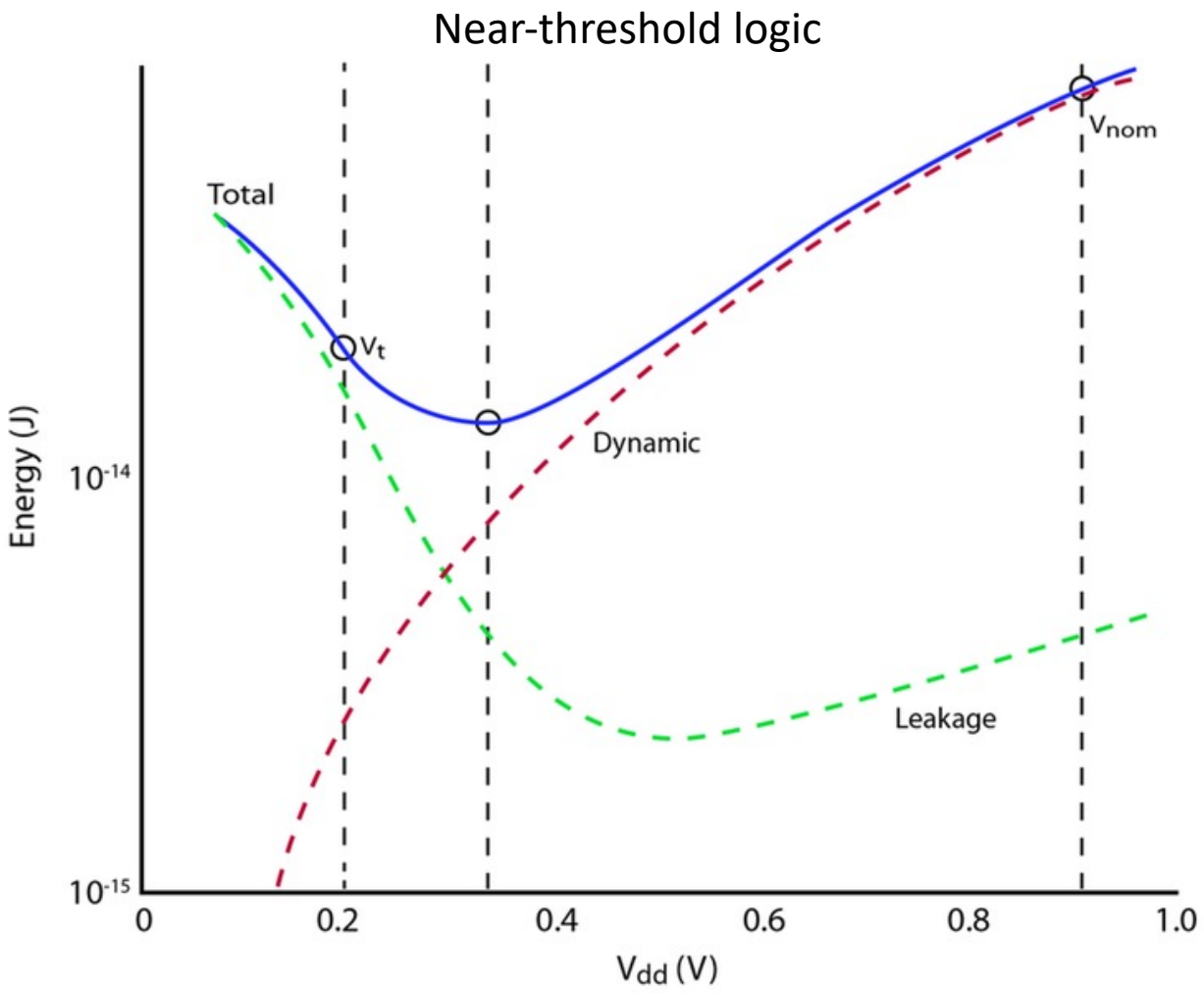
(a) Block diagram



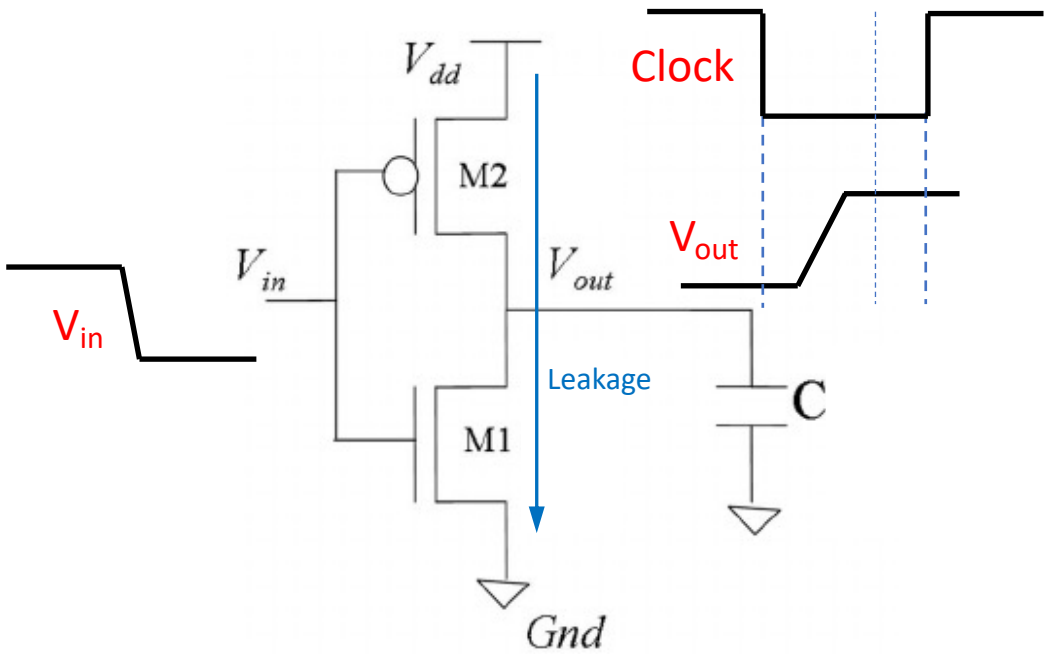
(b) Timing diagram of clock pulses

- Combinational circuit: Output determined solely by inputs
- Sequential circuits: Output determined by inputs AND previous outputs

Digital Circuits



[Source: David Blaauw, U Michigan, Ann Arbor]



- Gates are made of CMOS transistors, which require voltage source (VDD) to function
- What happens when we increase or decrease VDD?
  - Energy efficiency
  - Performance

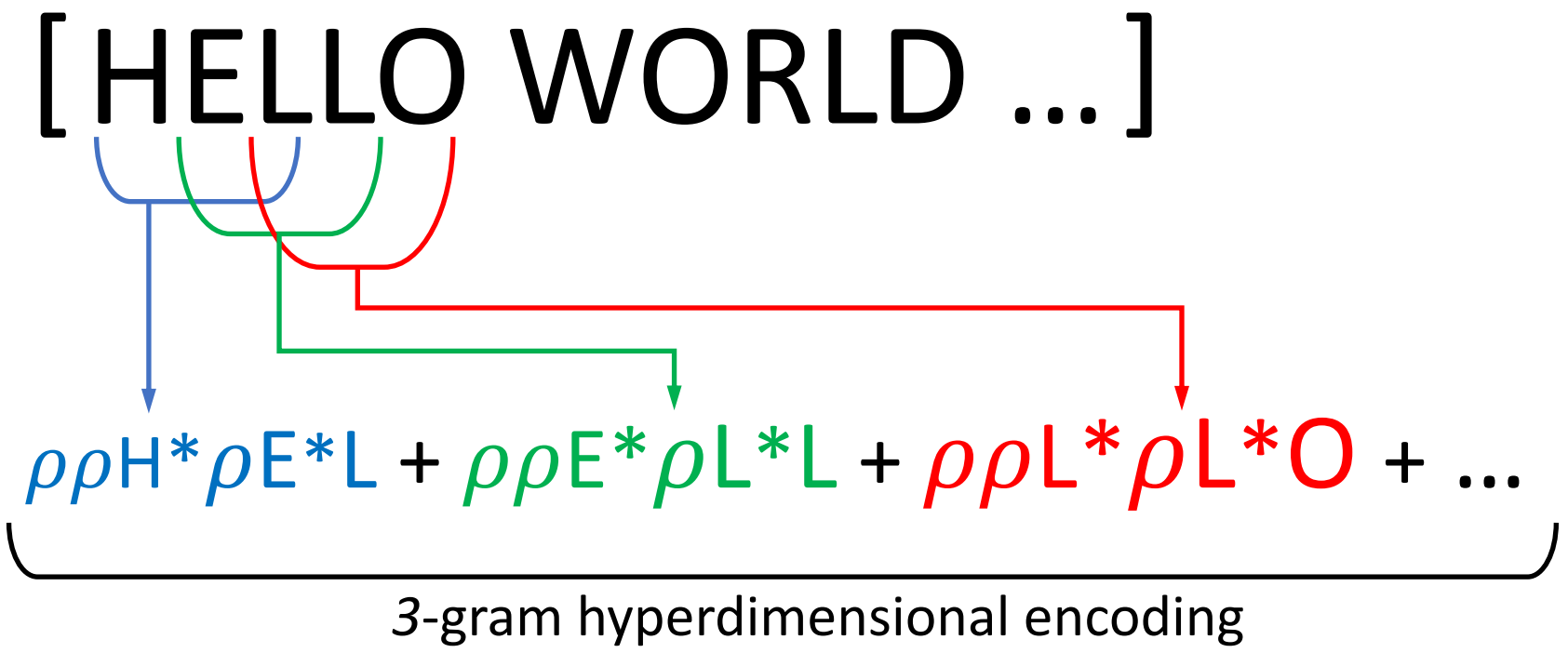
Specialized HD Architectures

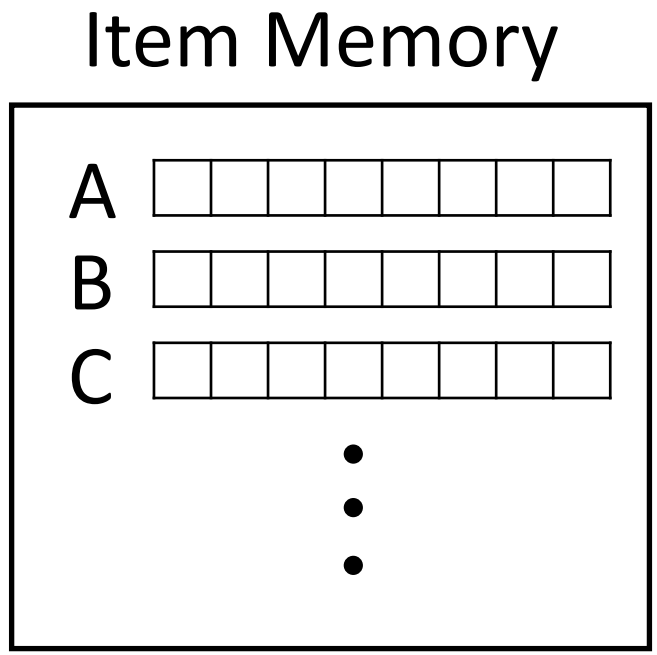
[Slides Courtesy of Abbas Rahimi, IBM Research; Alisha Menon, UC Berkeley]

Dense Hyperdimensional Encoder for Language Recognition

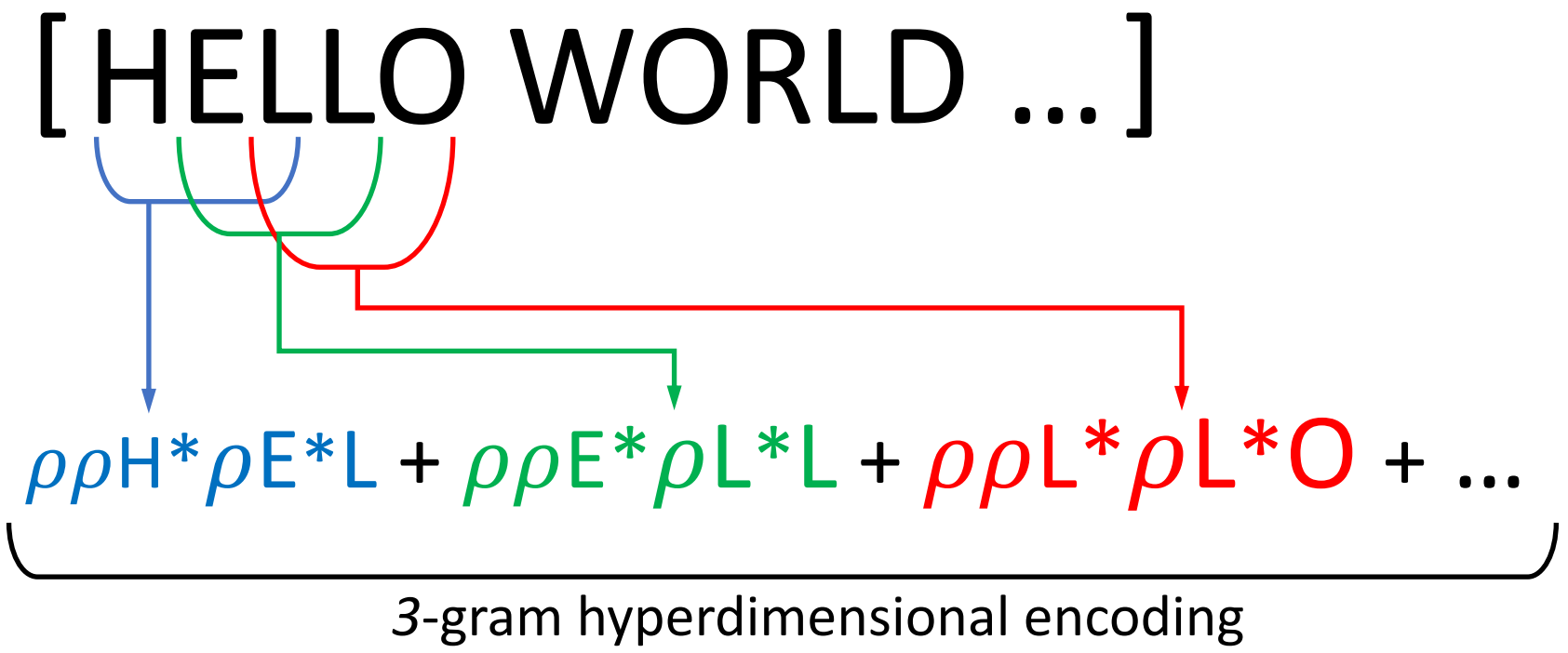
[HELLO WORLD …]

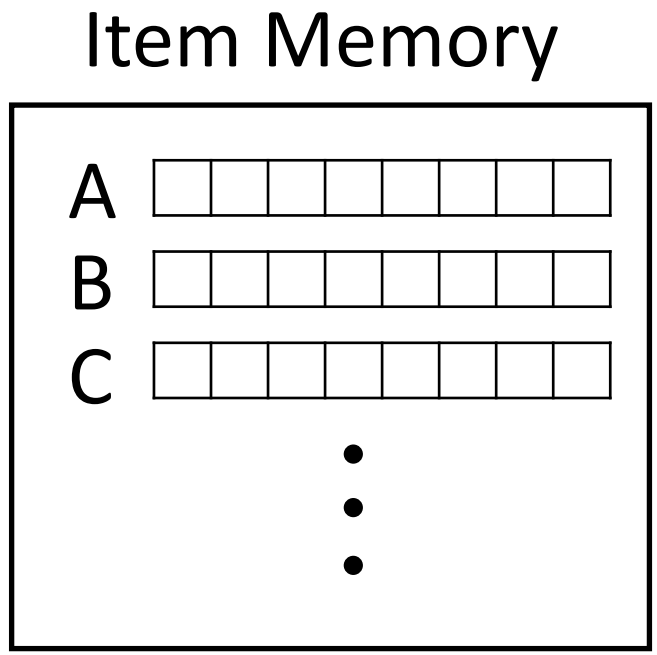
Dense Hyperdimensional Encoder for Language Recognition





Dense Hyperdimensional Encoder for Language Recognition

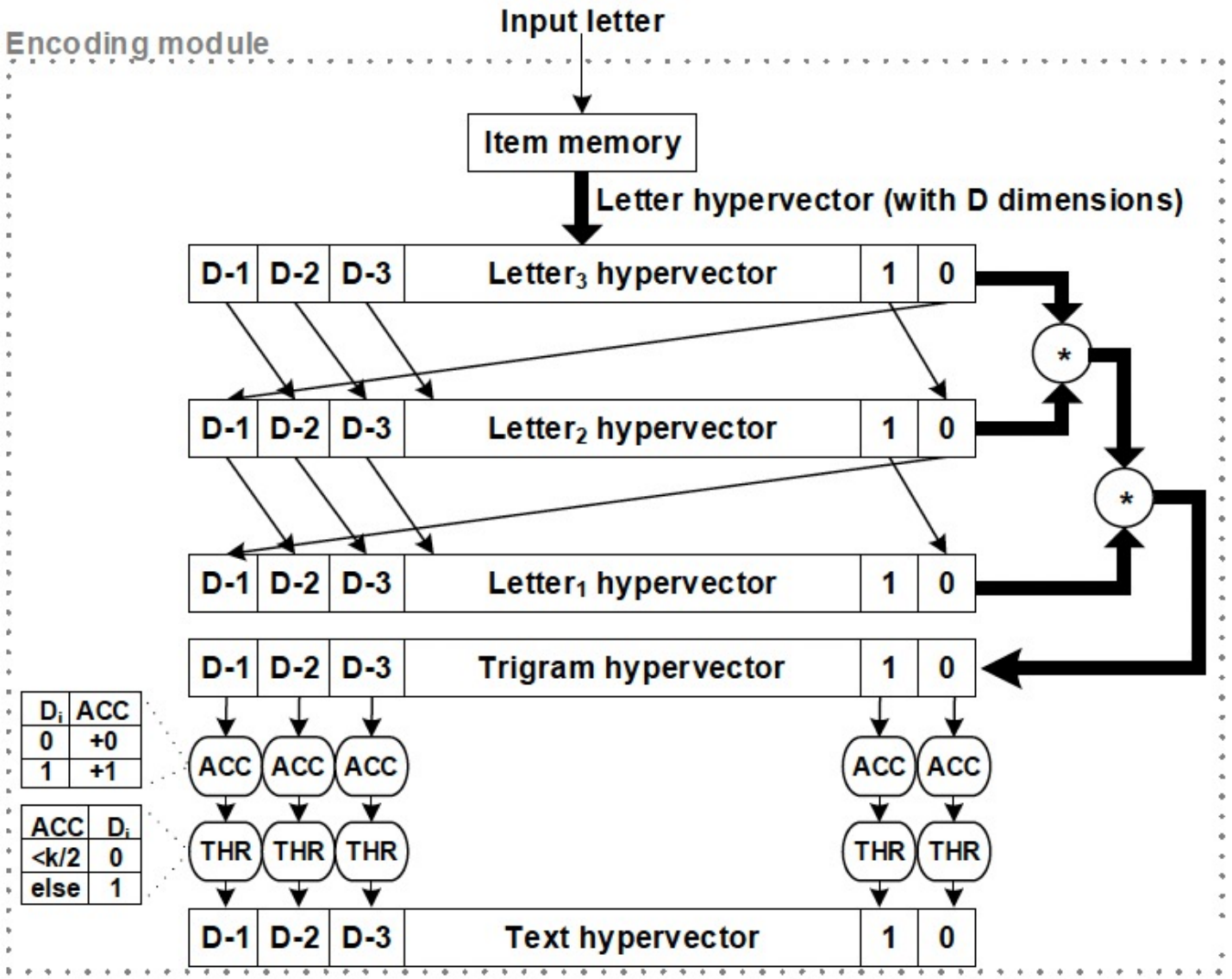




Three observations:

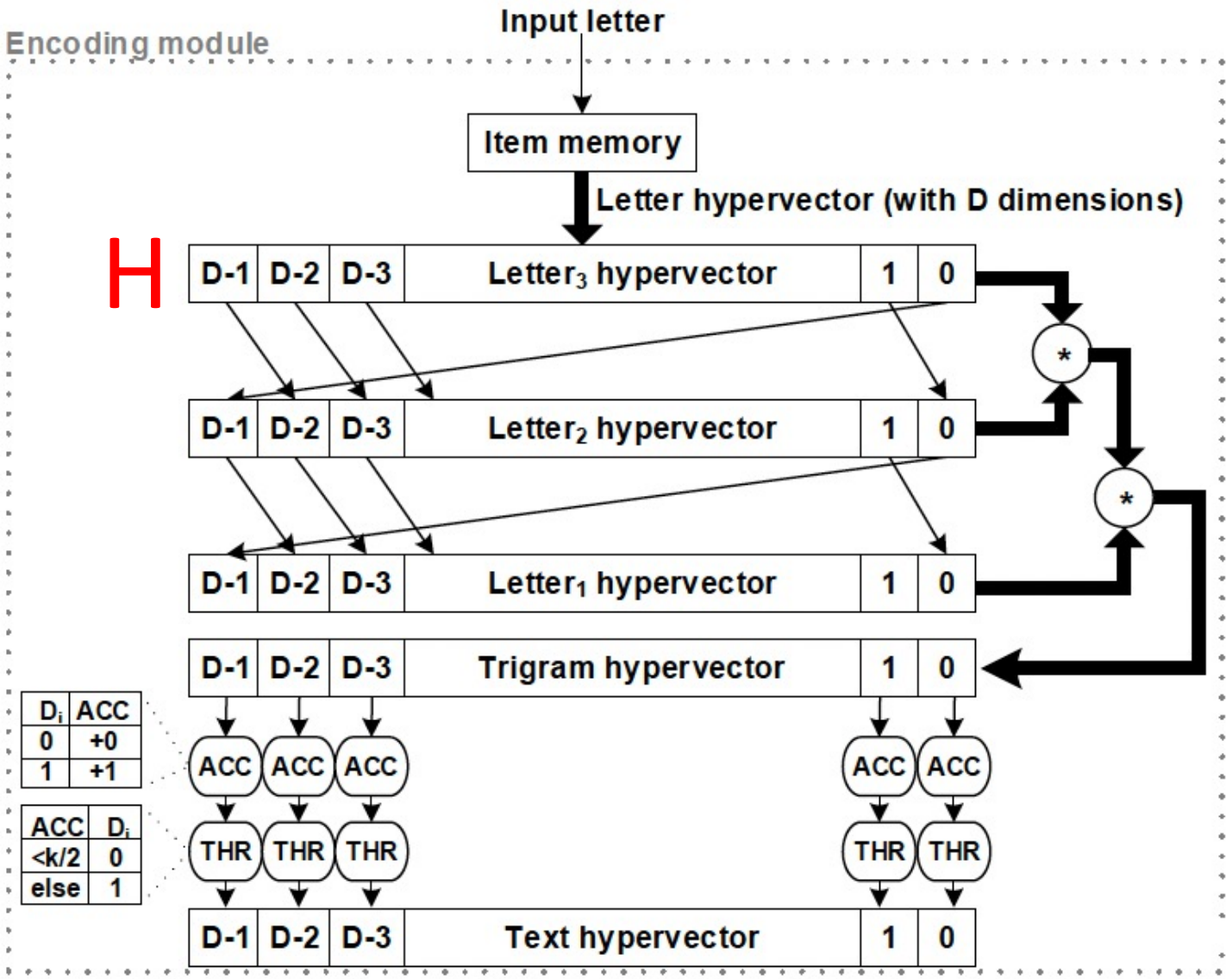
- Each *letter hypervector* is permuted twice times
- Pointwise multiplication (XOR binding) requires three registers to generate *trigram hypervector*
- Pointwise addition produces *text hypervector*

Dense Hyperdimensional Encoder for Language Recognition



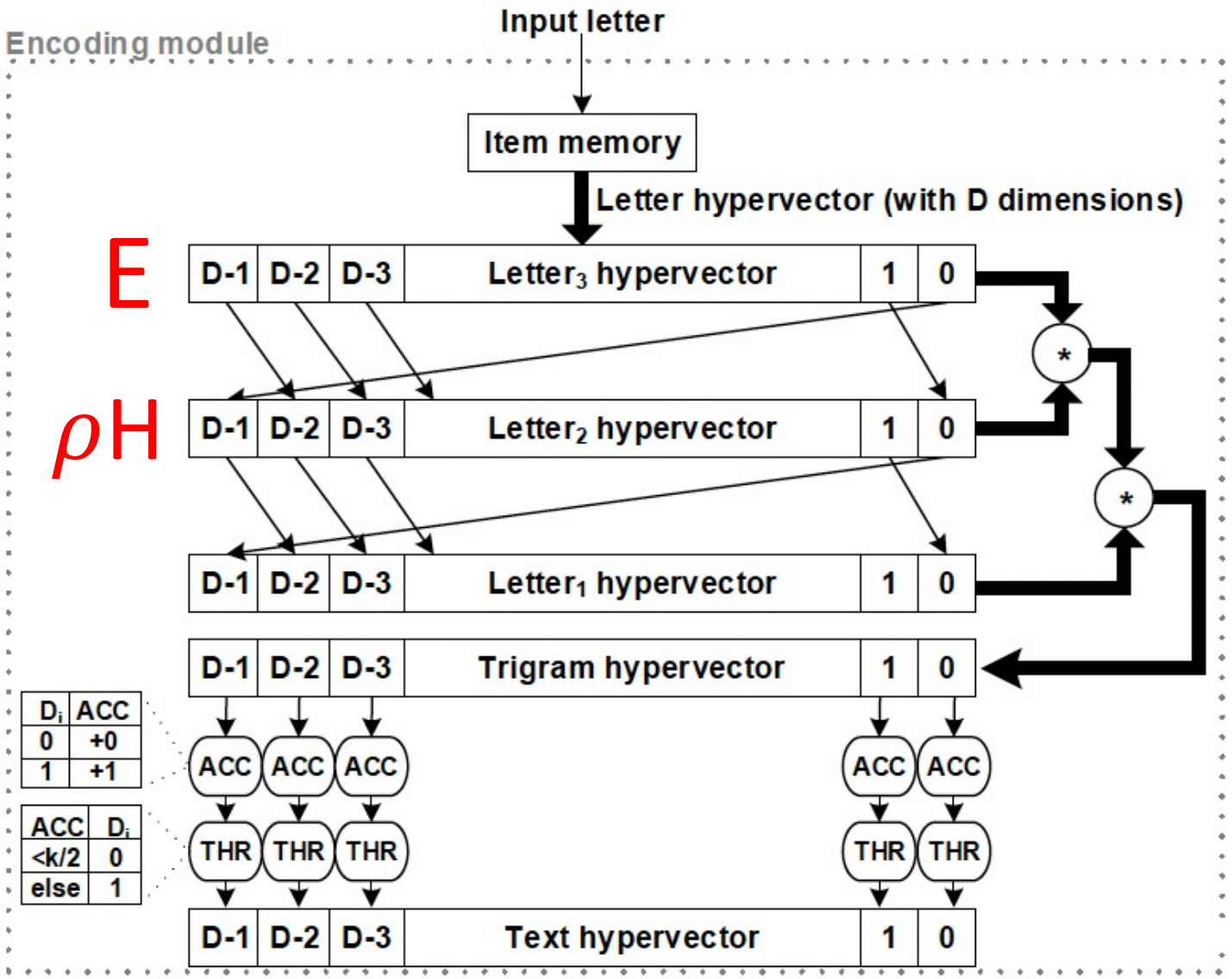
[HELLO WORLD ...]

Dense Hyperdimensional Encoder for Language Recognition



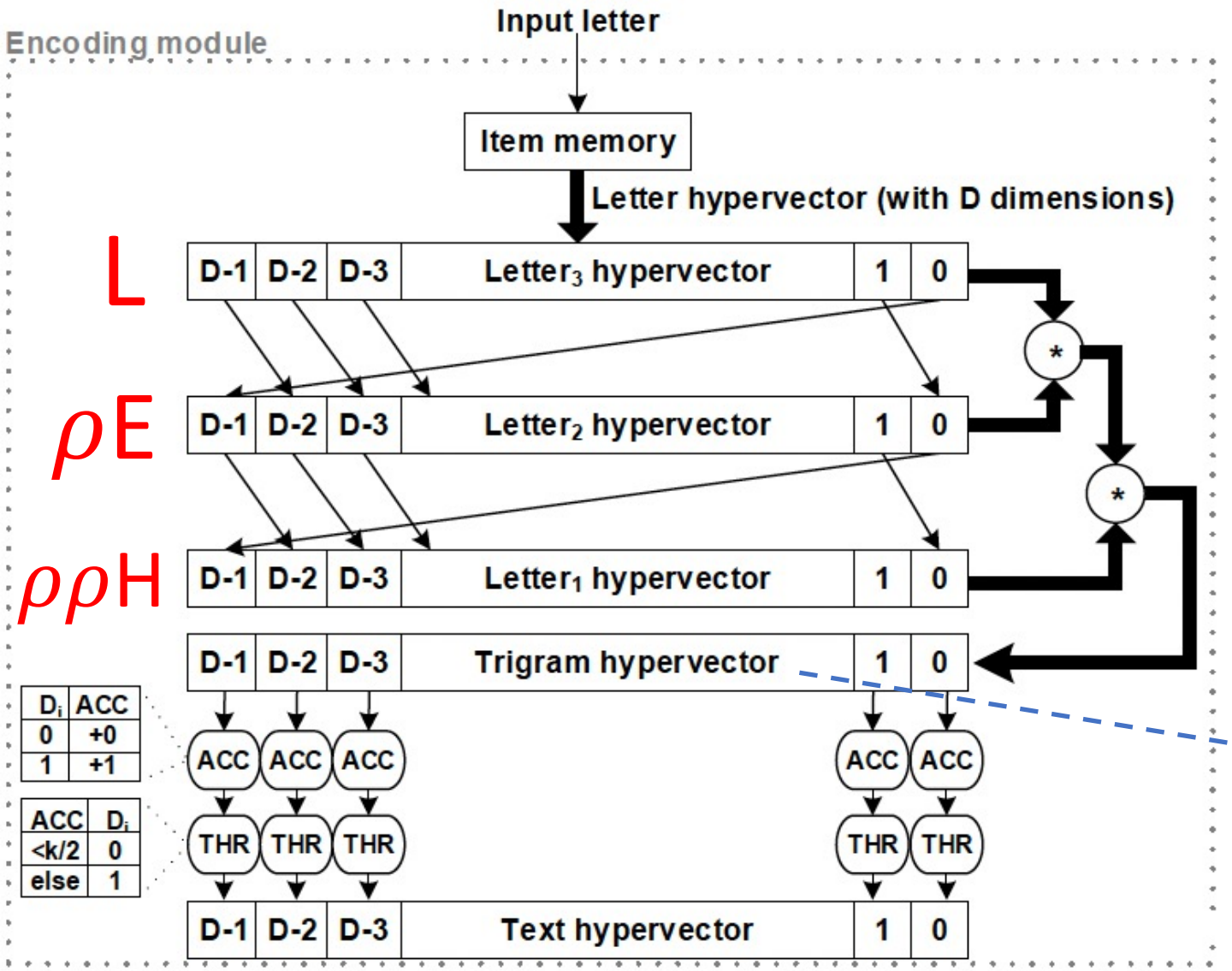
[HELLO WORLD ...]

Dense Hyperdimensional Encoder for Language Recognition



[HELLO WORLD …]

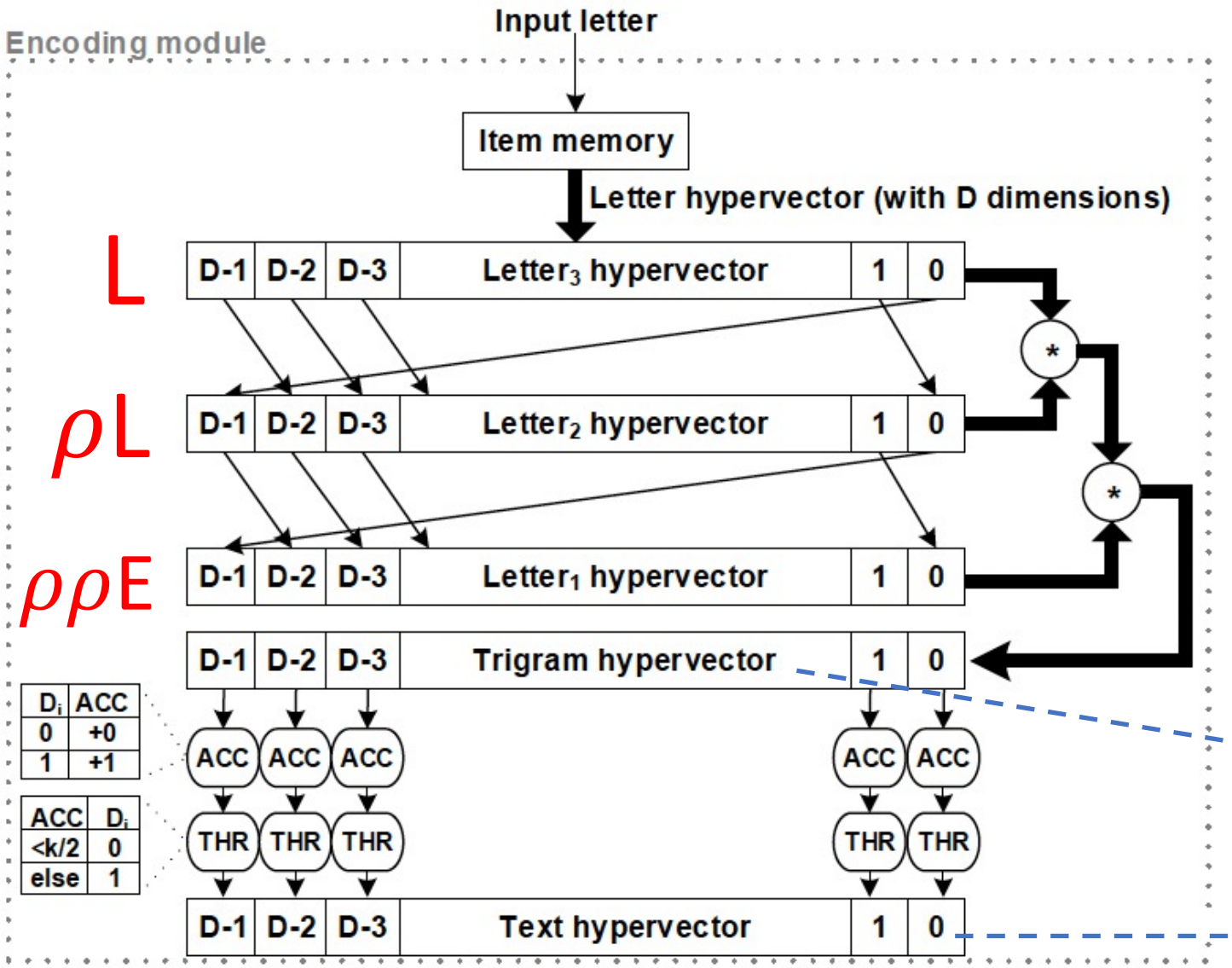
Dense Hyperdimensional Encoder for Language Recognition



[HELLO WORLD …]

ρρH*ρE*L

Dense Hyperdimensional Encoder for Language Recognition

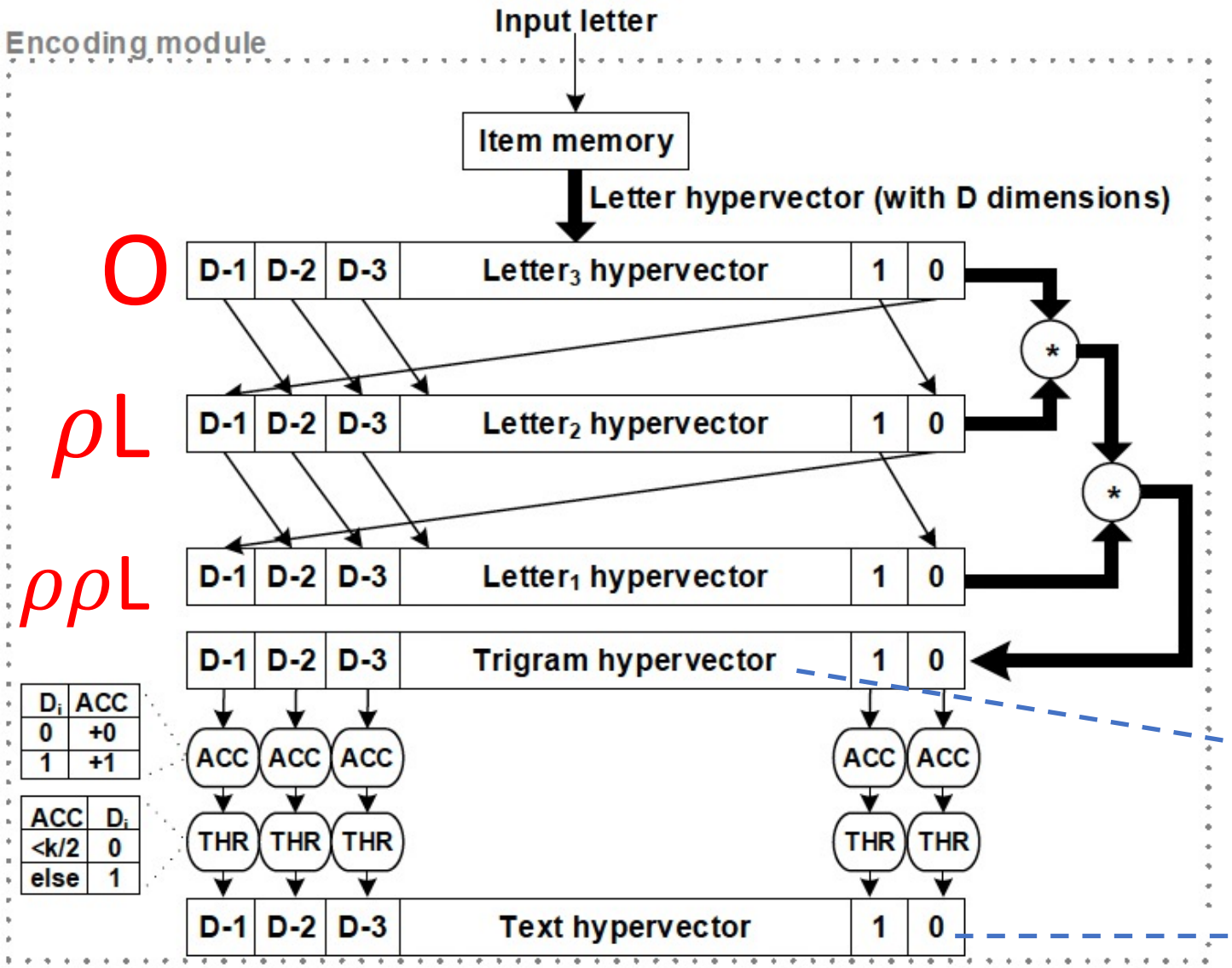


[HELLO WORLD ...]

ρρE*ρL*L

(ρρH*ρE*L)

Dense Hyperdimensional Encoder for Language Recognition



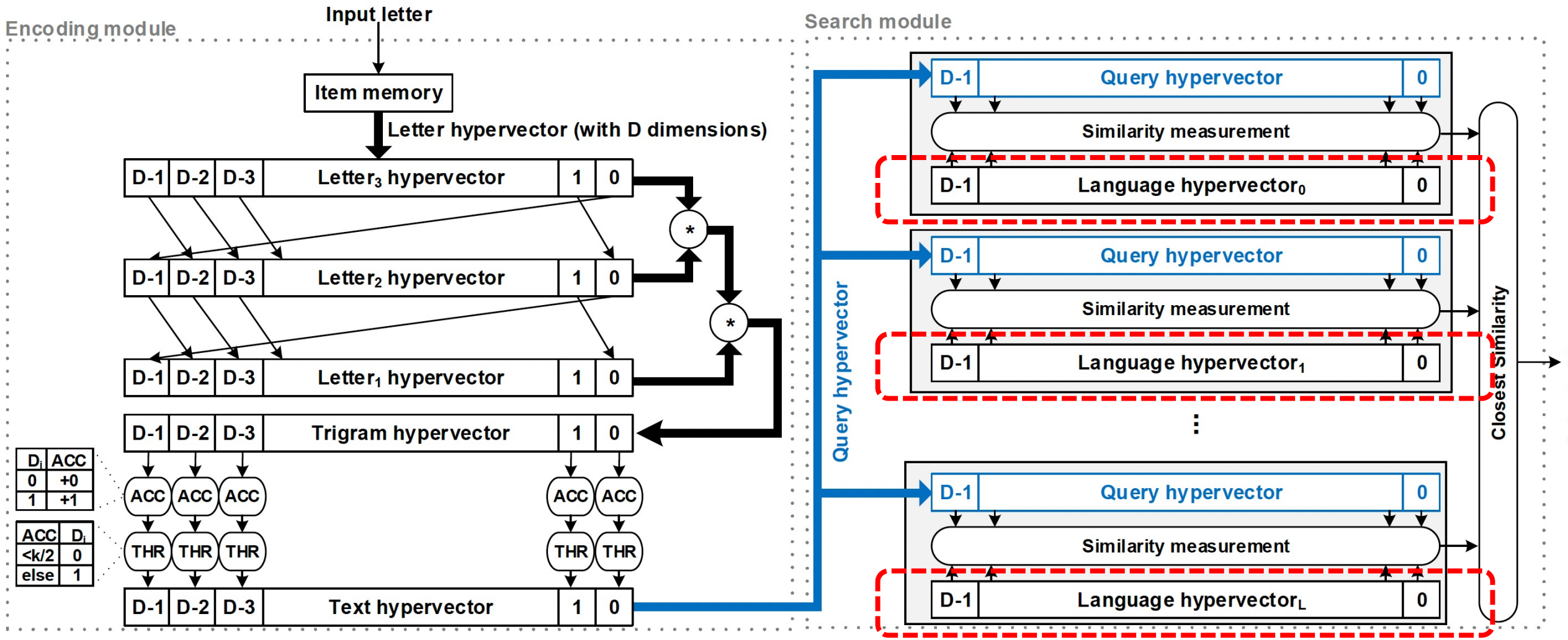
[HELLO WORLD …]

ρρL*ρL*O

(ρρH*ρE*L) + (ρρE*ρL*L)

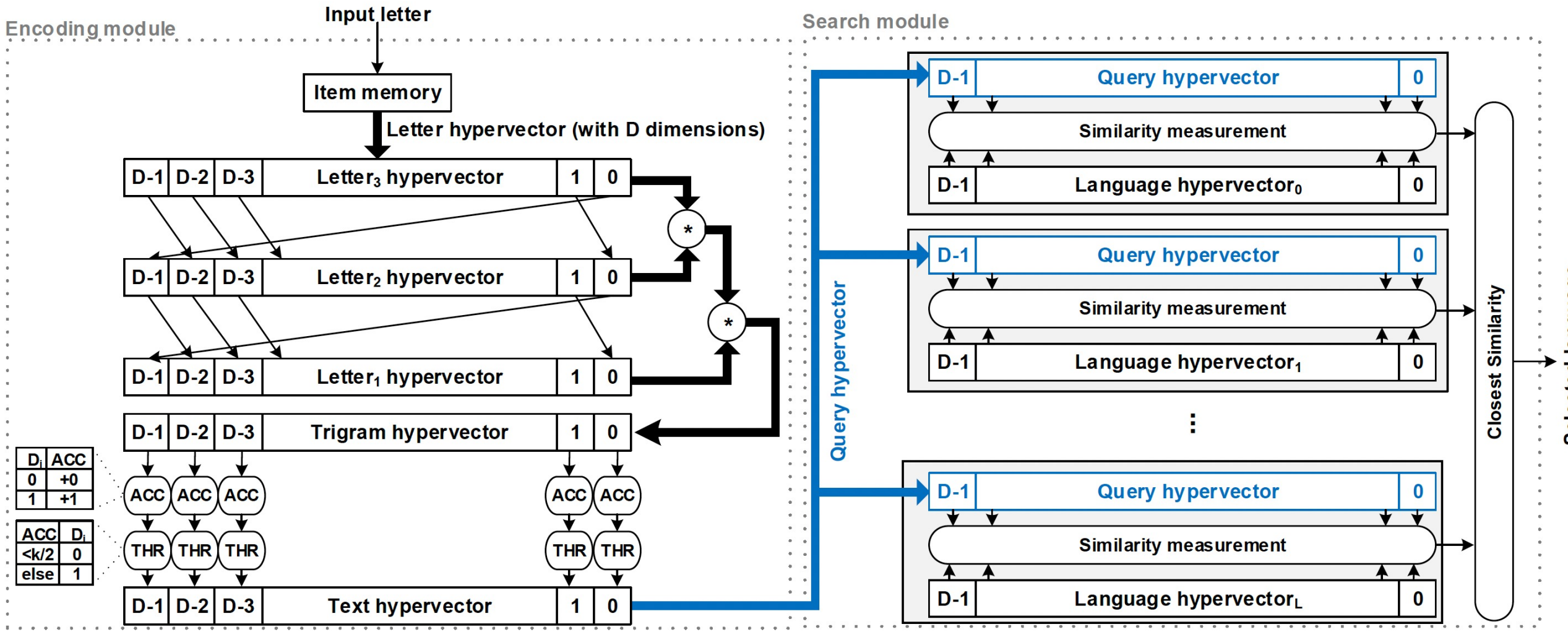
Dense Hyperdimensional Encoder for Language Recognition

Training: Store encoded language hypervectors in the associative memory

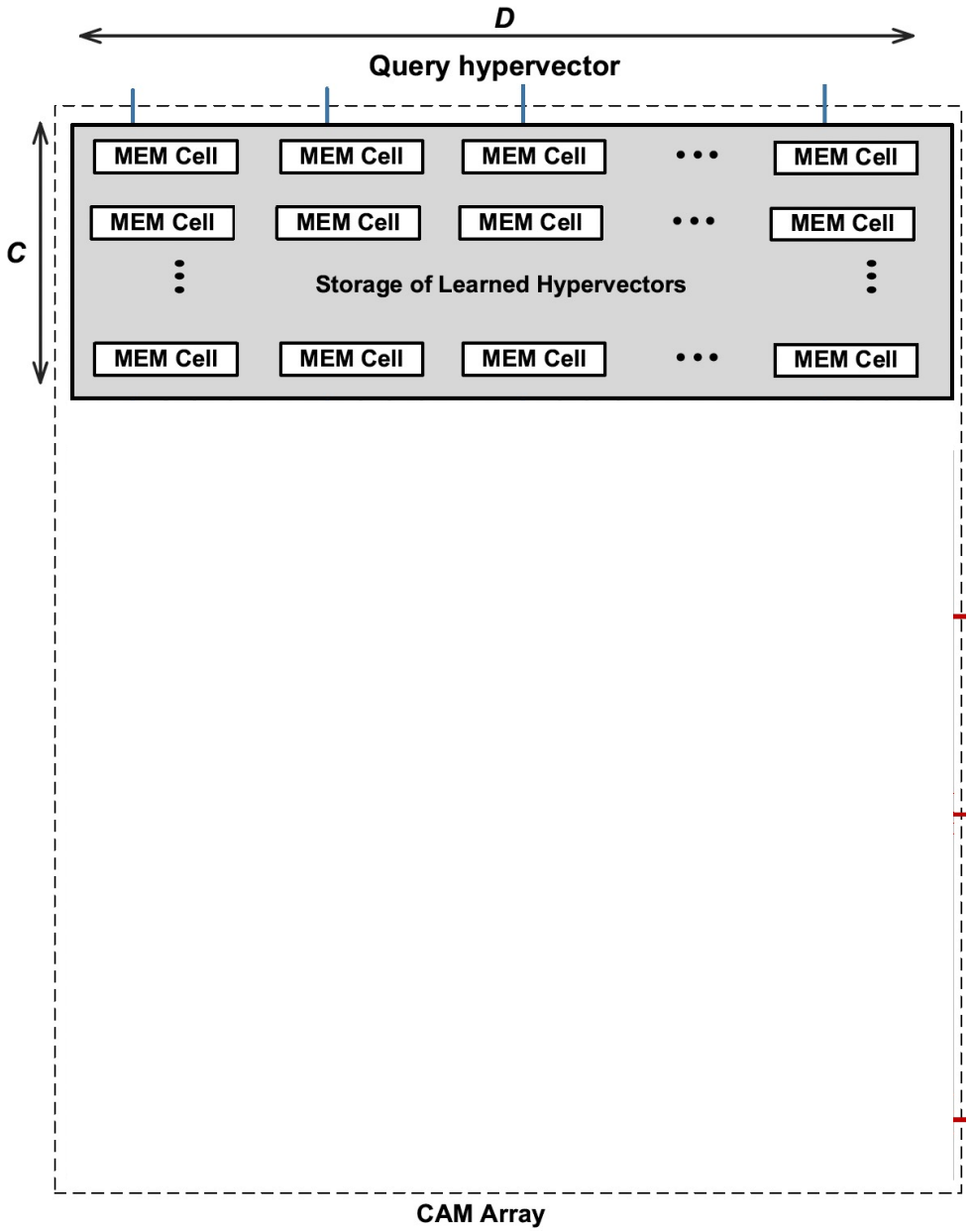


Dense Hyperdimensional Encoder for Language Recognition

Testing/Query: Encode text of an unknown language (Query hypervector)



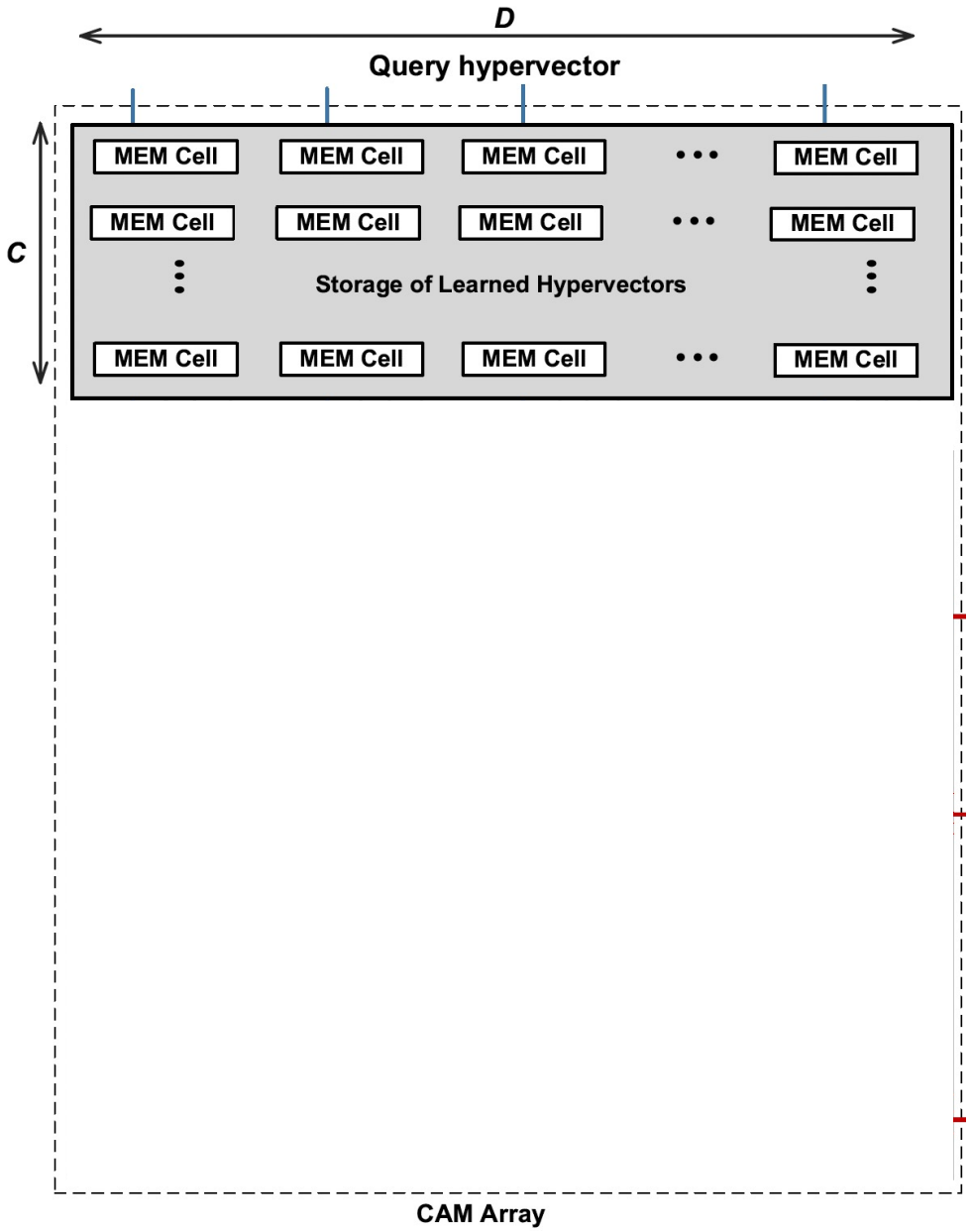
Associative Search



Digital Hyperdimensional Associative Memory (D-HAM)

- SoA associative search uses *content-addressable memories* (CAMs)
- Distance Computation: *Hamming distance* is hardware friendly (avoid normalization in Cosine)

Associative Search



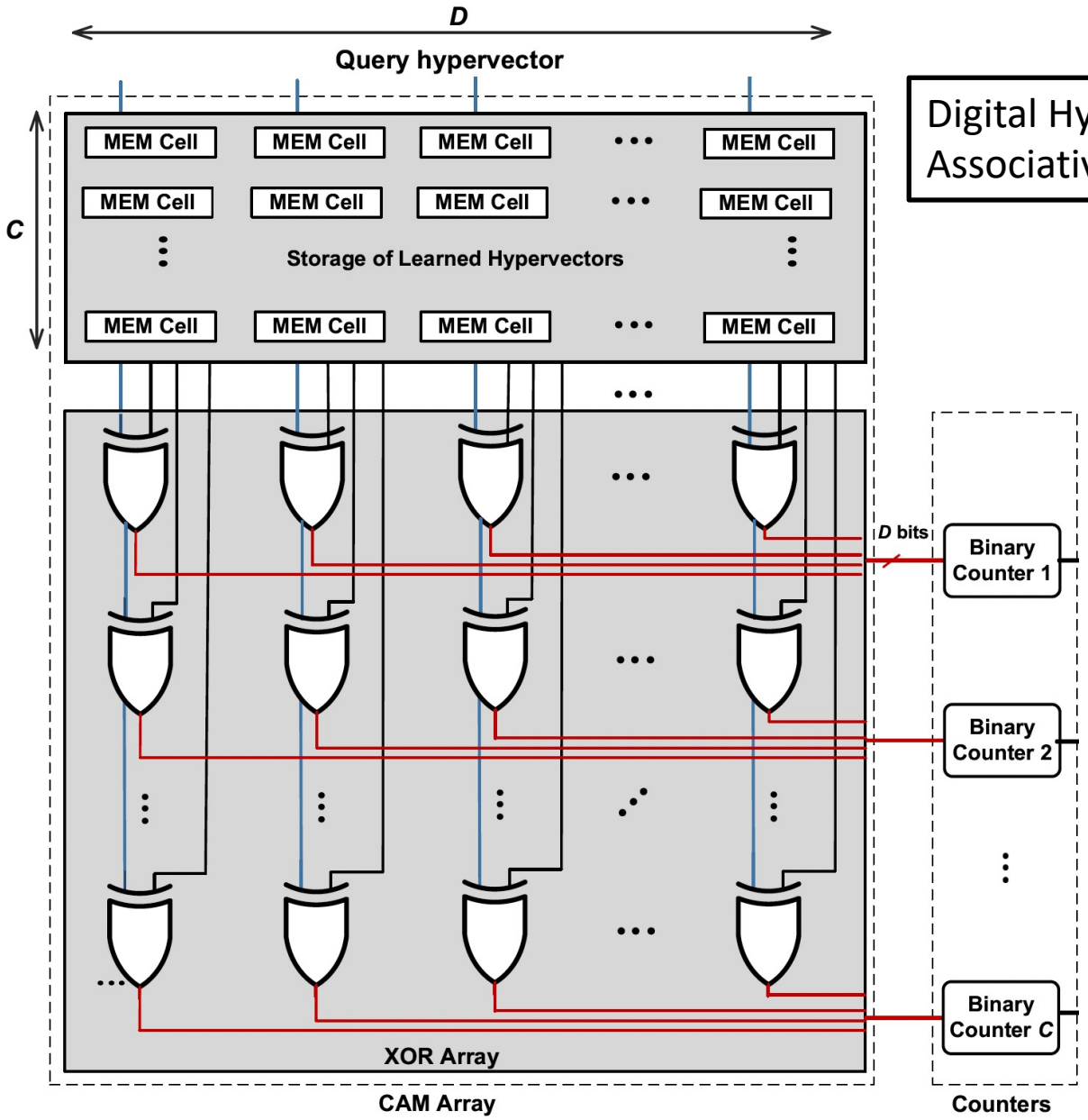
Digital Hyperdimensional Associative Memory (D-HAM)

- SoA associative search uses *content-addressable memories* (CAMs)
- Distance Computation: *Hamming distance* is hardware friendly (avoid normalization in Cosine)

	1	0	1	0	0	0
XOR	1	1	1	0	1	0
	0	1	0	0	1	0

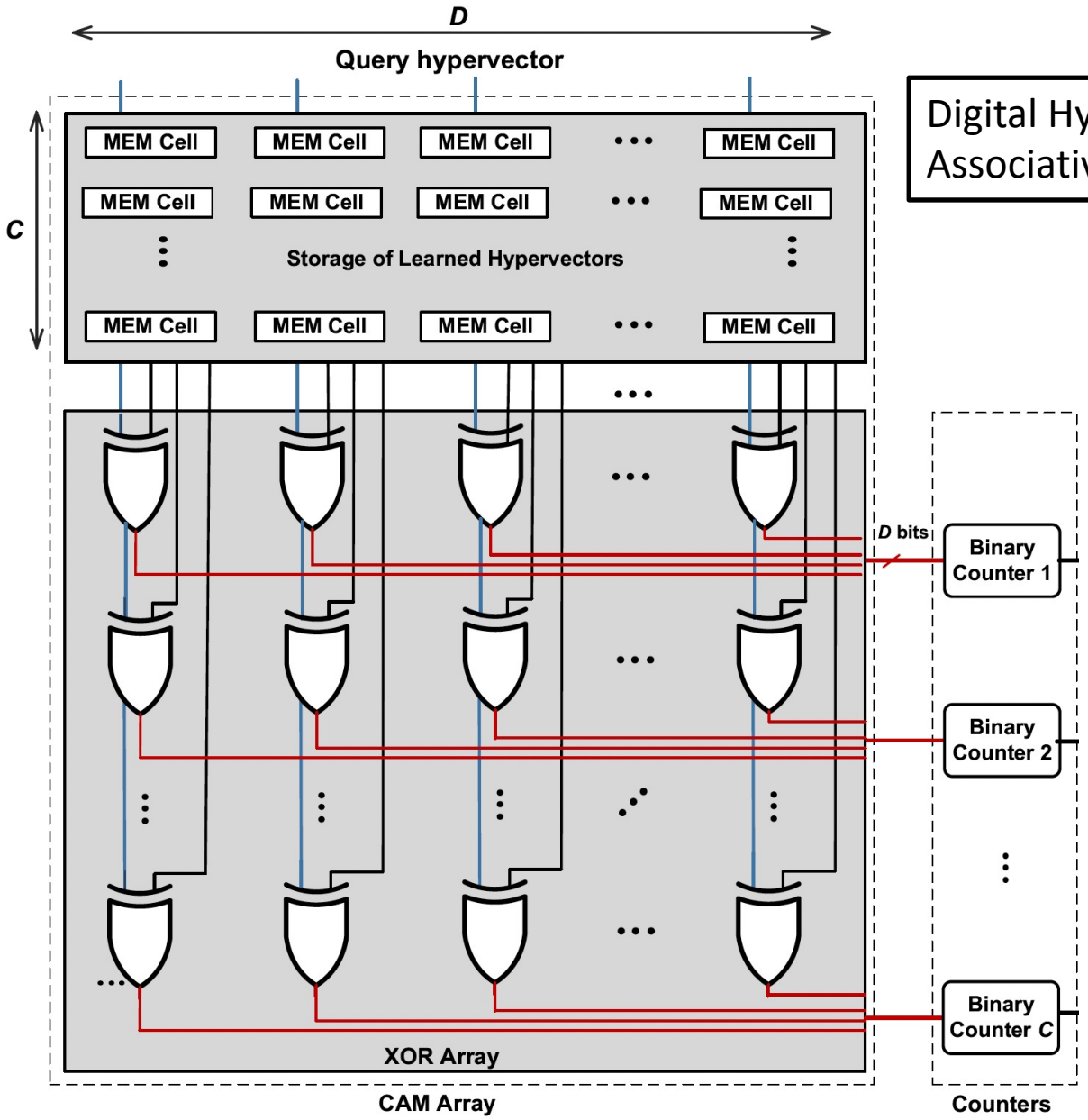
Hamming distance = Count of 1's = 2

Associative Search



Digital Hyperdimensional Associative Memory (D-HAM)

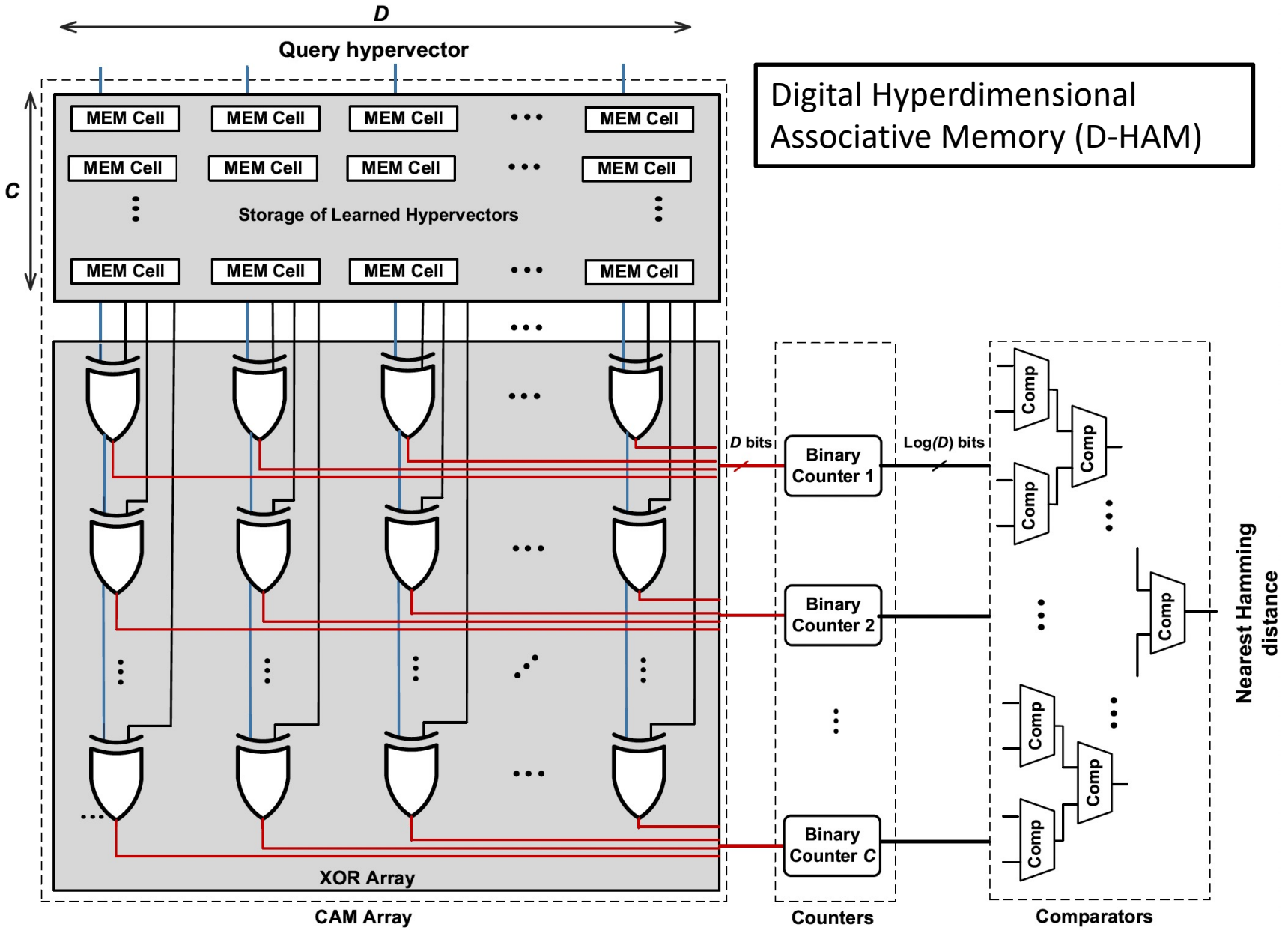
Associative Search



Digital Hyperdimensional Associative Memory (D-HAM)

- **Nearest neighbor** = Class HV with Minimum Hamming distance to Query HV

Associative Search



Digital Hyperdimensional Associative Memory (D-HAM)

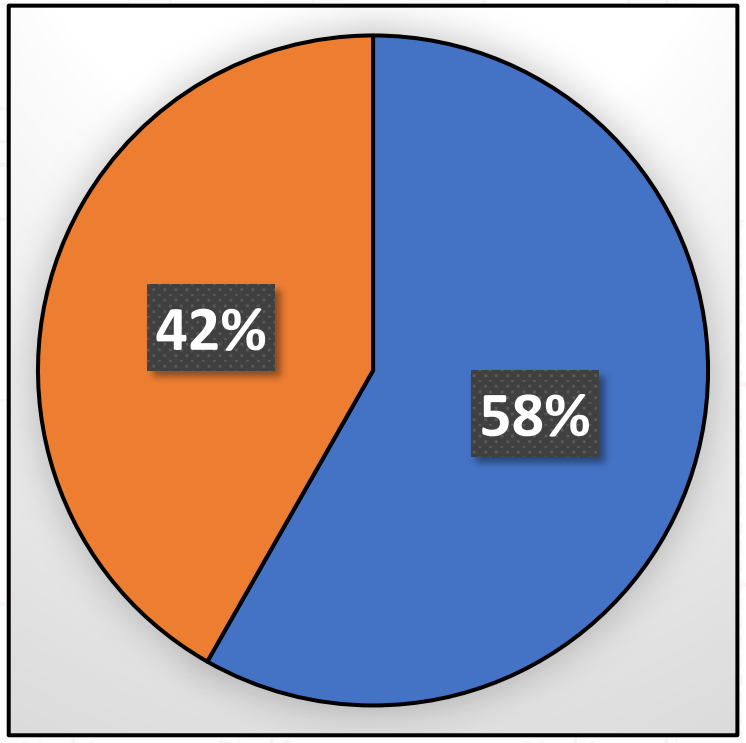
- **Nearest neighbor** = Class HV with Minimum Hamming distance to Query HV
- Use a binary tree of comparators (height = log *C*)

Associative Search

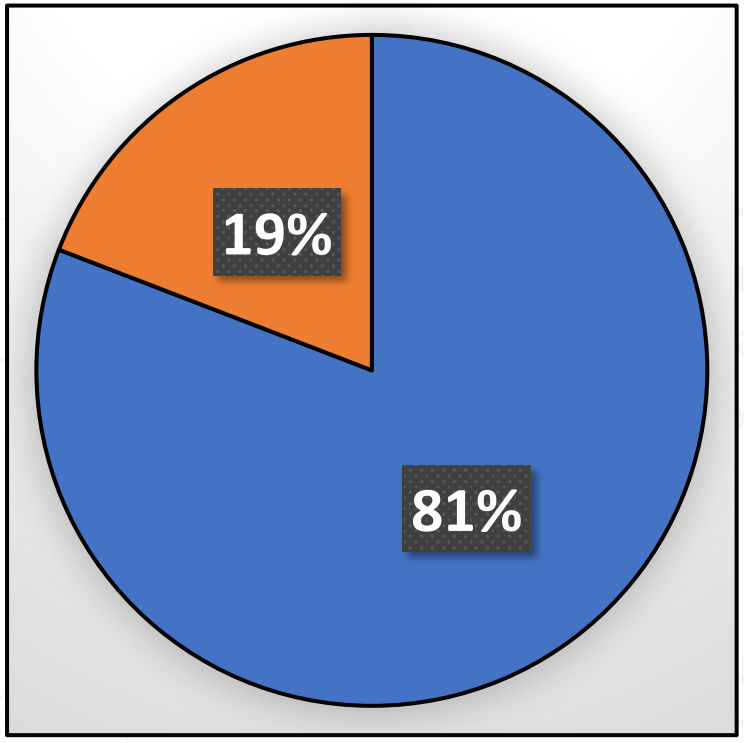
$$D = 10{,}000$$
$$C = 100$$

CAM array

Counters and Comparators



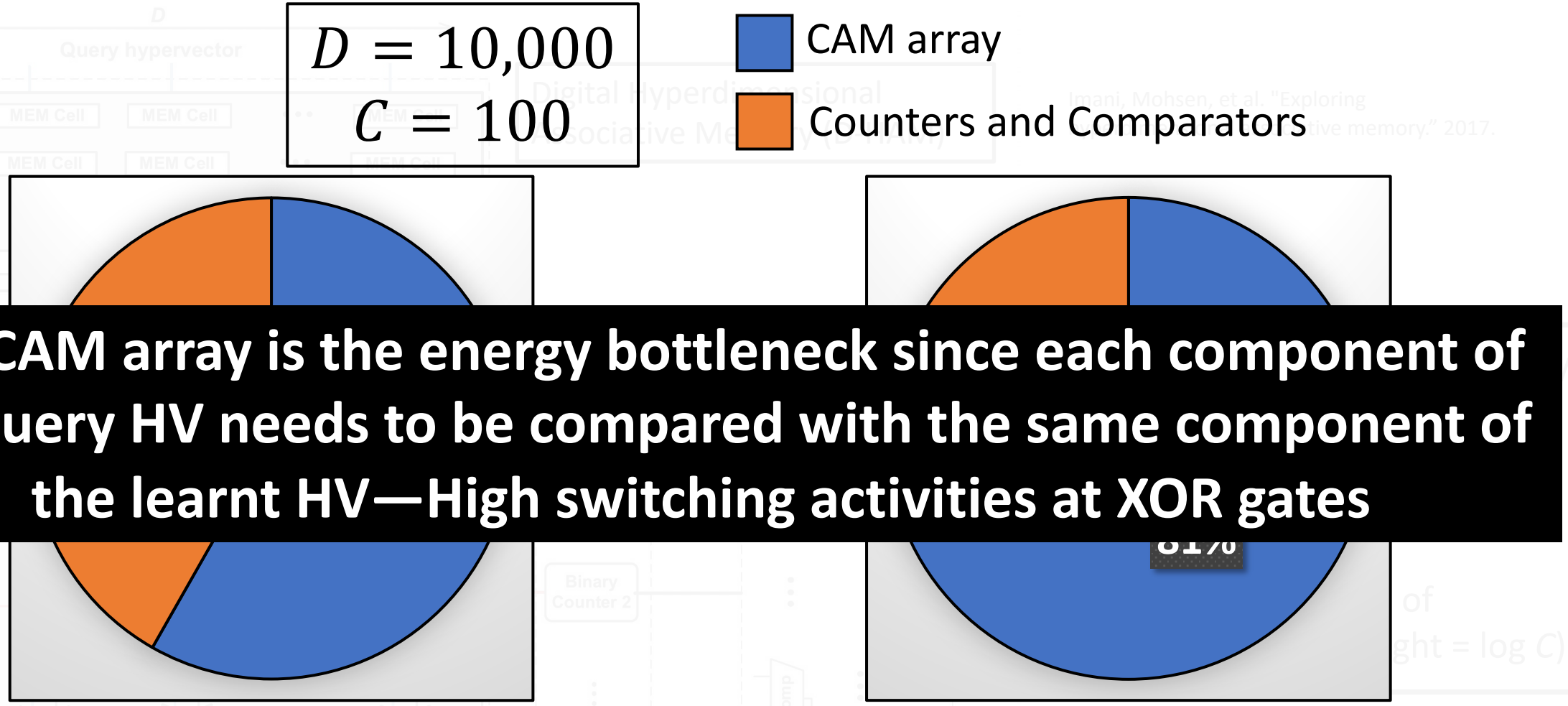
Total area
(26 mm^2)



Total Energy Consumed Per Query
(6155 pJ)

Associative Search

$D = 10{,}000$
$C = 100$



The CAM array is the energy bottleneck since each component of the query HV needs to be compared with the same component of the learnt HV—High switching activities at XOR gates

Total area
(26 mm^2)

Total Energy Consumed Per Query
(6155 pJ)

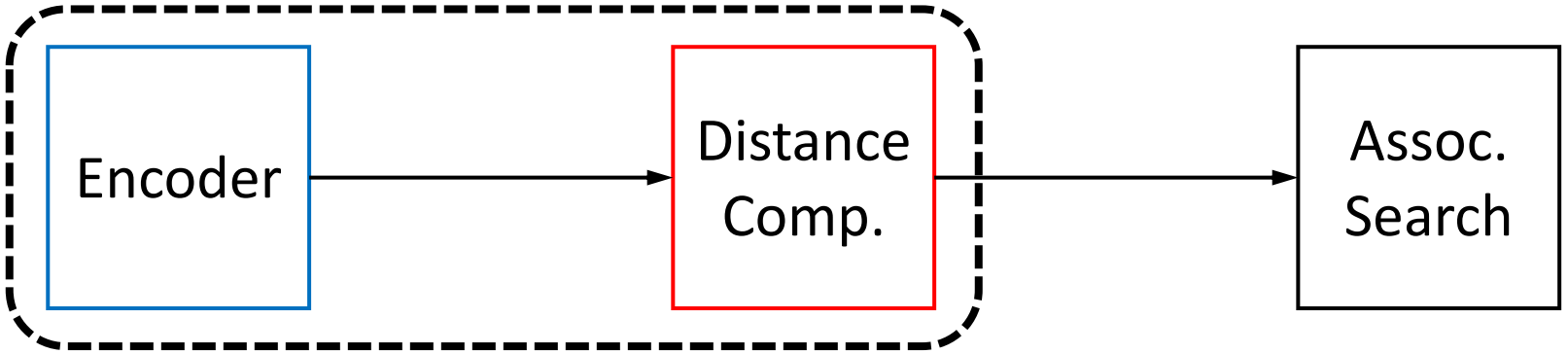
Sparse Hyperdimensional Encoder for Language Recognition

- An attempt to reduce switching activity…
- Number of “1” elements ≪ Number of “0” elements
  - Example: If *D = 100 000*, number of “1” elements could be 1000

Sparse Hyperdimensional Encoder for Language Recognition

- An attempt to reduce switching activity…
- Number of “1” elements ≪ Number of “0” elements
  - Example: If *D = 100 000*, number of “1” elements could be 1000

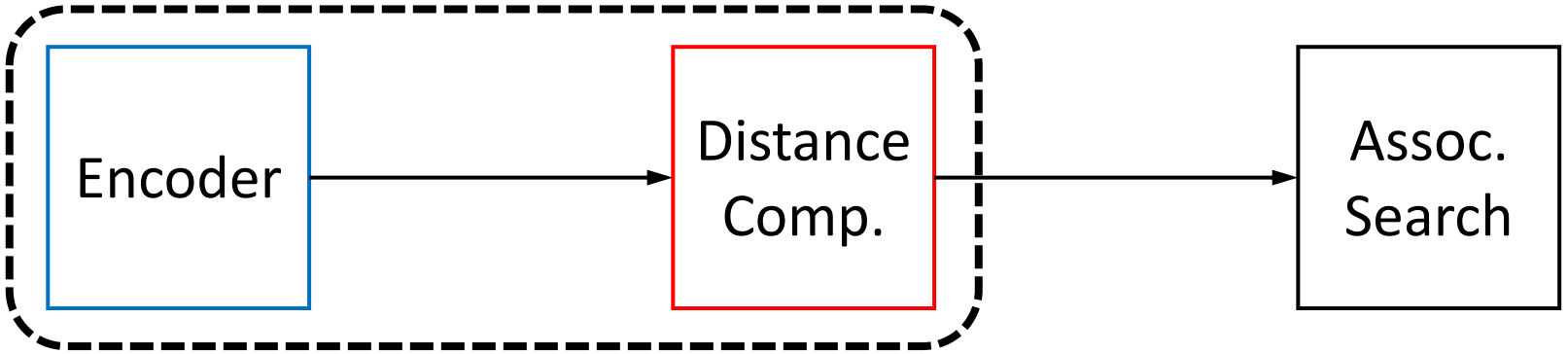
How can we use sparseness to reduce energy?



Sparse Hyperdimensional Encoder for Language Recognition

- An attempt to reduce switching activity…
- Number of "1" elements ≪ Number of "0" elements
  - Example: If *D = 100 000*, number of "1" elements could be 1000

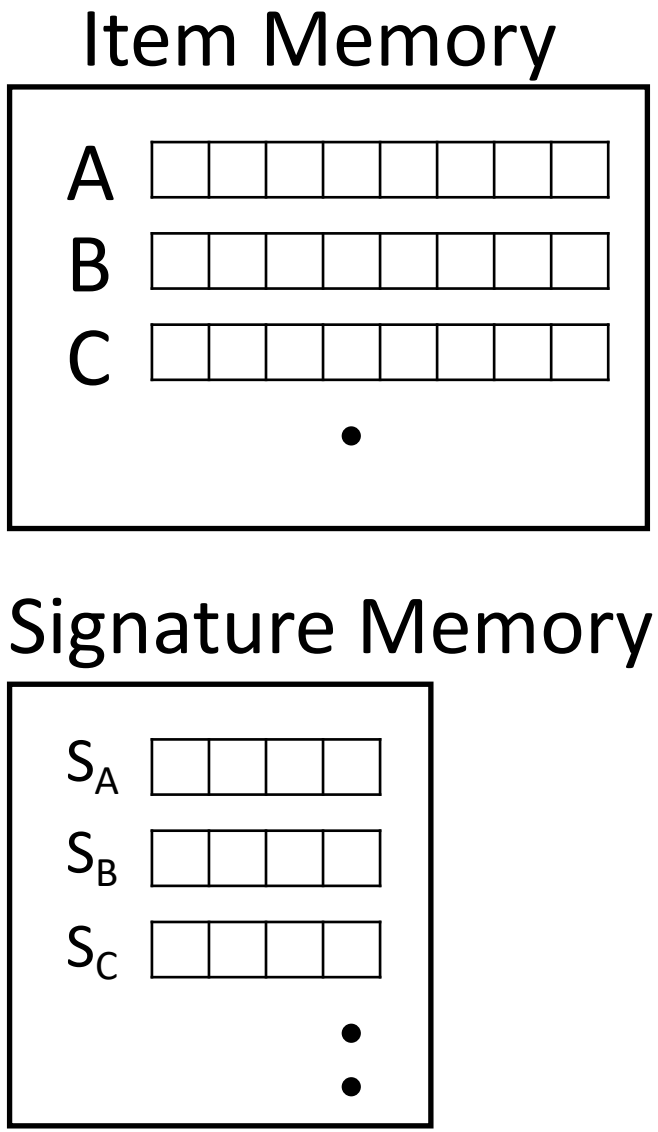
How can we use sparseness to reduce energy?



XOR Binding does not work with sparse representation! Why?

The alignment of 1's in the compared sparse HVs is more important now—Use dot product as a metric…

Sparse Hyperdimensional Encoder for Language Recognition



Sparse Hyperdimensional Encoder for Language Recognition

Introduce permutation-based binding/encoding…

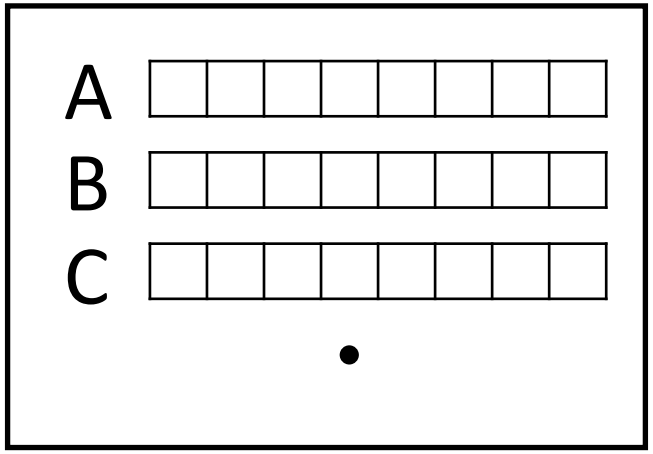
[HELLO WORLD …]

$\rho_L = S_H \oplus S_E$

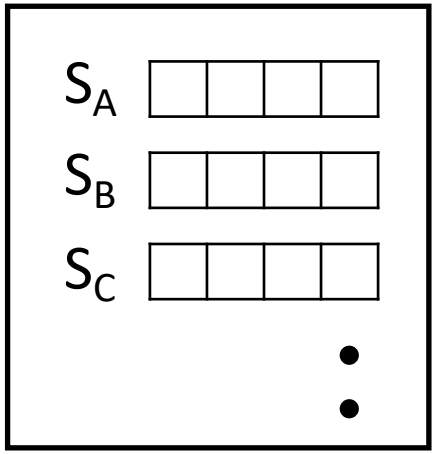
$\rho_H = S_E \oplus S_L$

$\rho_E = S_H \oplus S_L$

Item Memory



Signature Memory



Sparse Hyperdimensional Encoder for Language Recognition

Introduce permutation-based binding/encoding…

[HELLO WORLD …]

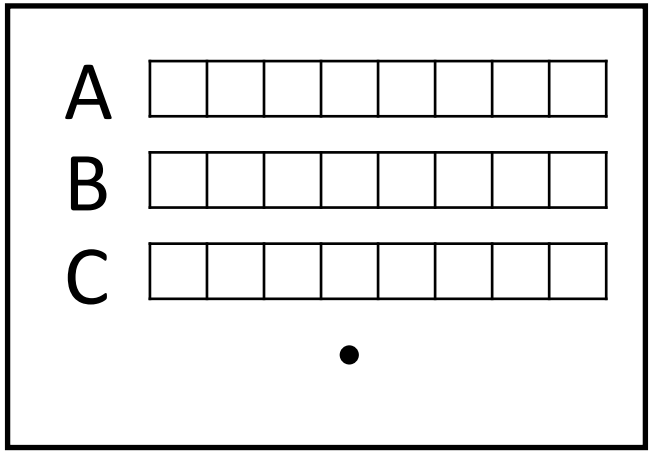
$\rho_H = S_E \oplus S_L$

$\rho_E = S_H \oplus S_L$

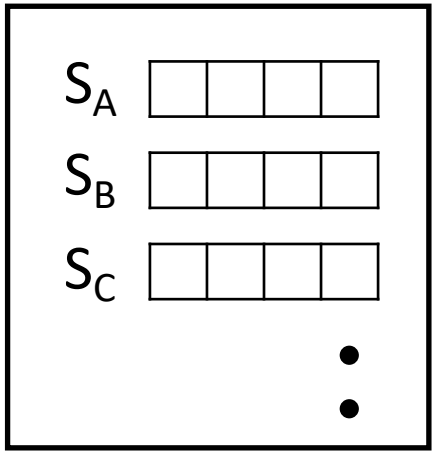
$\rho_L = S_H \oplus S_E$

Use them for bundling-based *3-gram* encoding

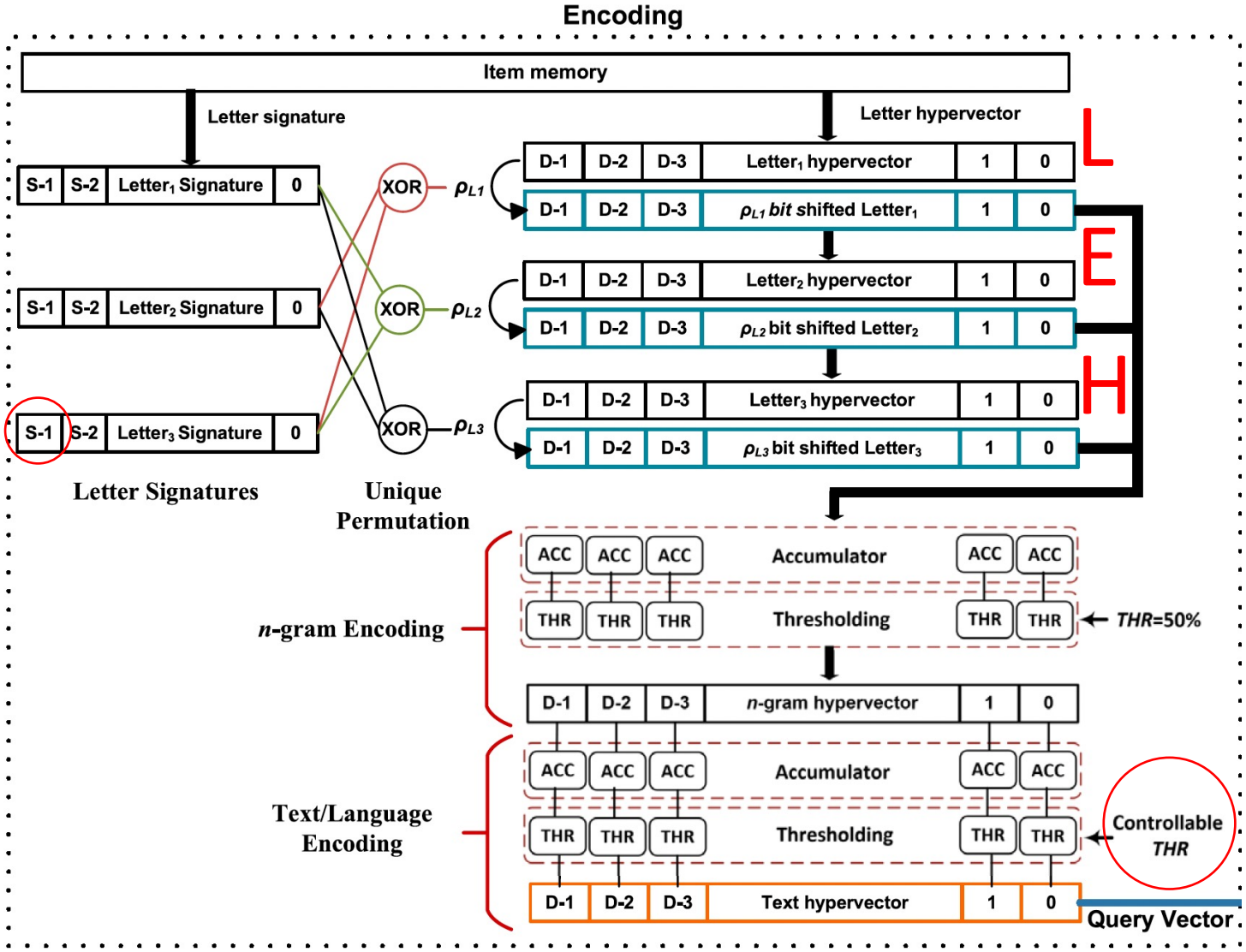
Item Memory



Signature Memory



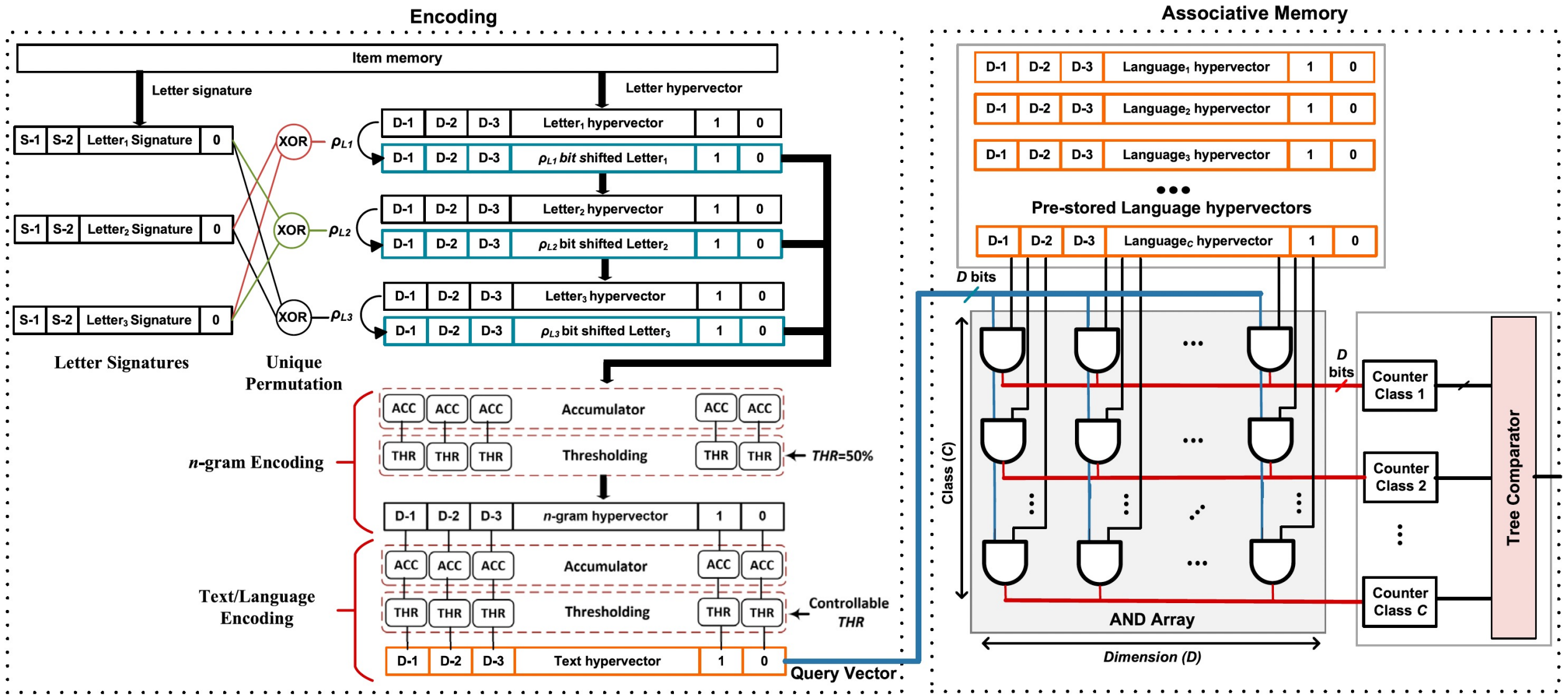
Sparse Hyperdimensional Encoder for Language Recognition



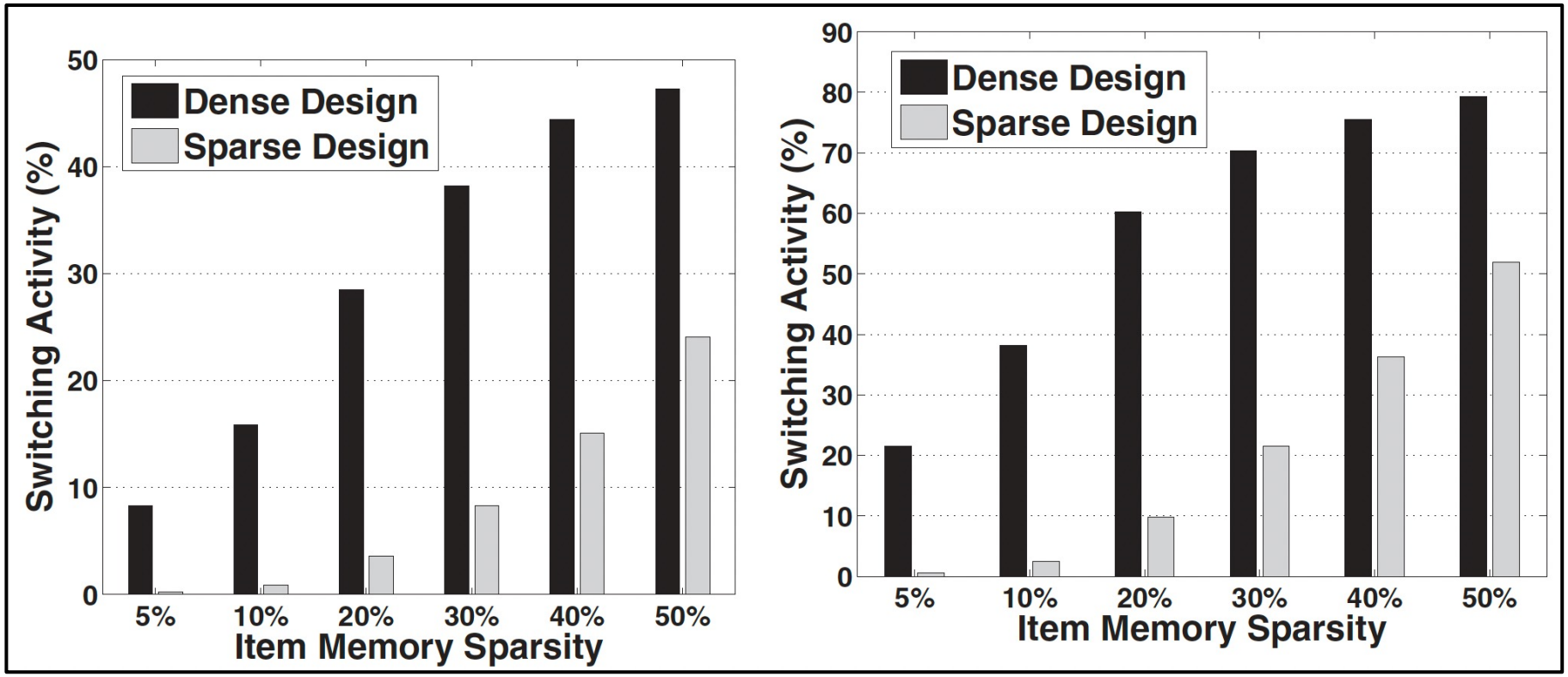
Permutation-based encoding…

A form of bundling and context-dependent thinning to maintain appropriate density…

Associative Search (AND Gates)



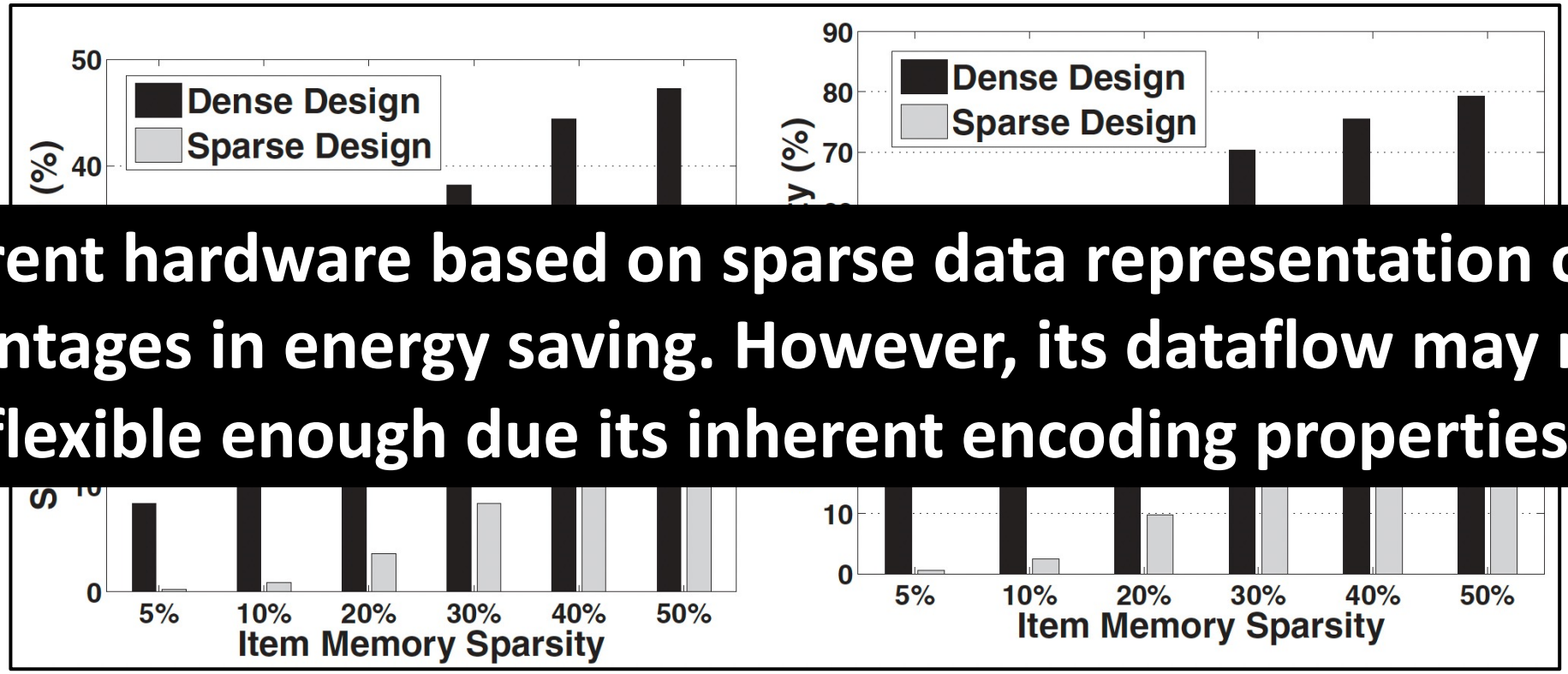
Sparse vs. Dense Data Representation



Encoding

Associative Memory

Sparse vs. Dense Data Representation



Current hardware based on sparse data representation offers advantages in energy saving. However, its dataflow may not be flexible enough due its inherent encoding properties…

Encoding

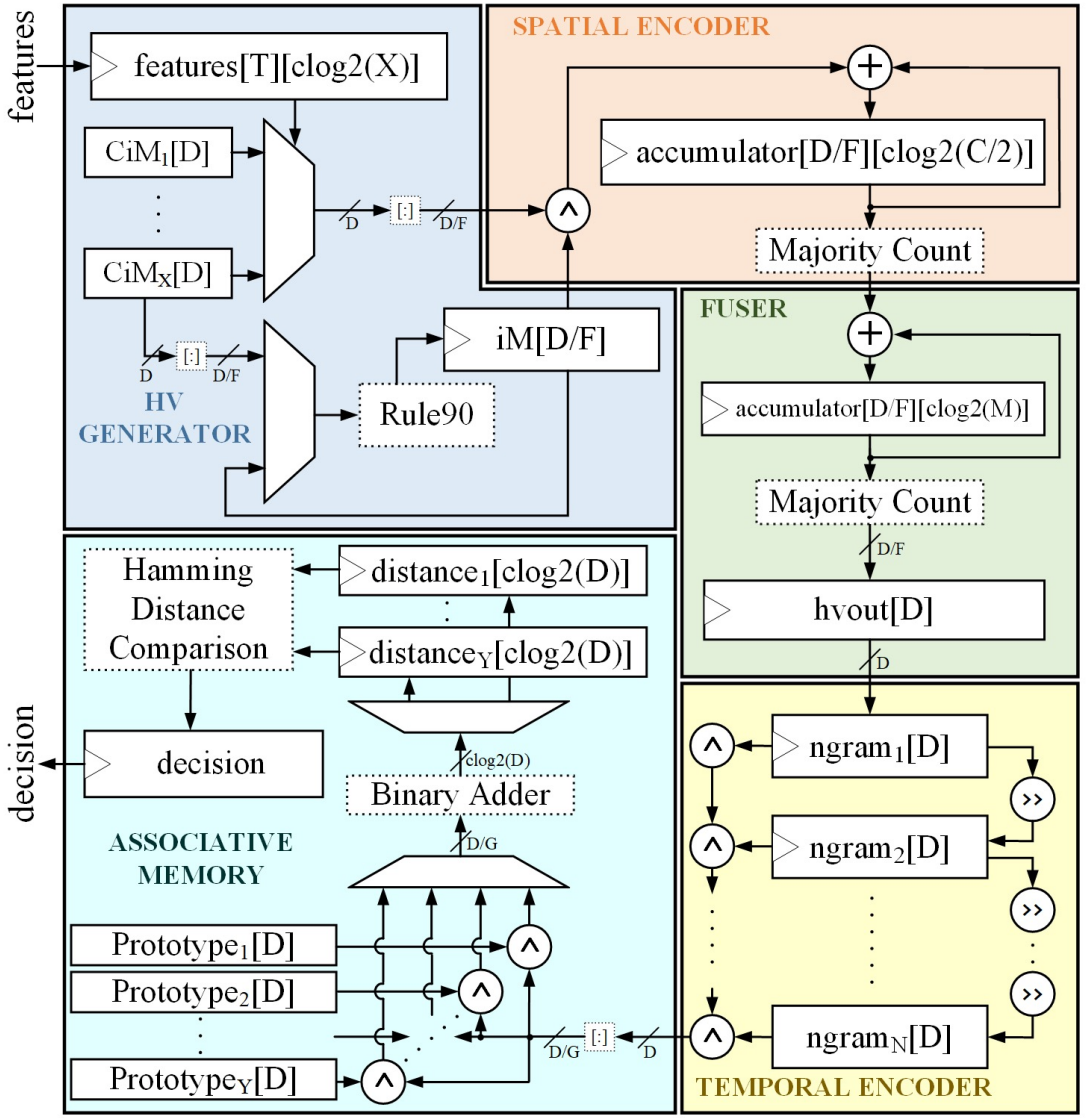
Associative Memory

Hyperdimensional Processor Architecture for Emotion Recognition

D: HV Dimension
F: # Folds in HV Generator, Spatial Encoder, Fuser
G: # Folds in Associative Memory
N: # Ngrams
X: # Discrete Input Feature Values
Y: # Output Classes

M: # Modalities
C: # Channels in the Modality with the most Channels
T: Total # Channels across all Modalities
clog2: ceiling of the base 2 logarithm

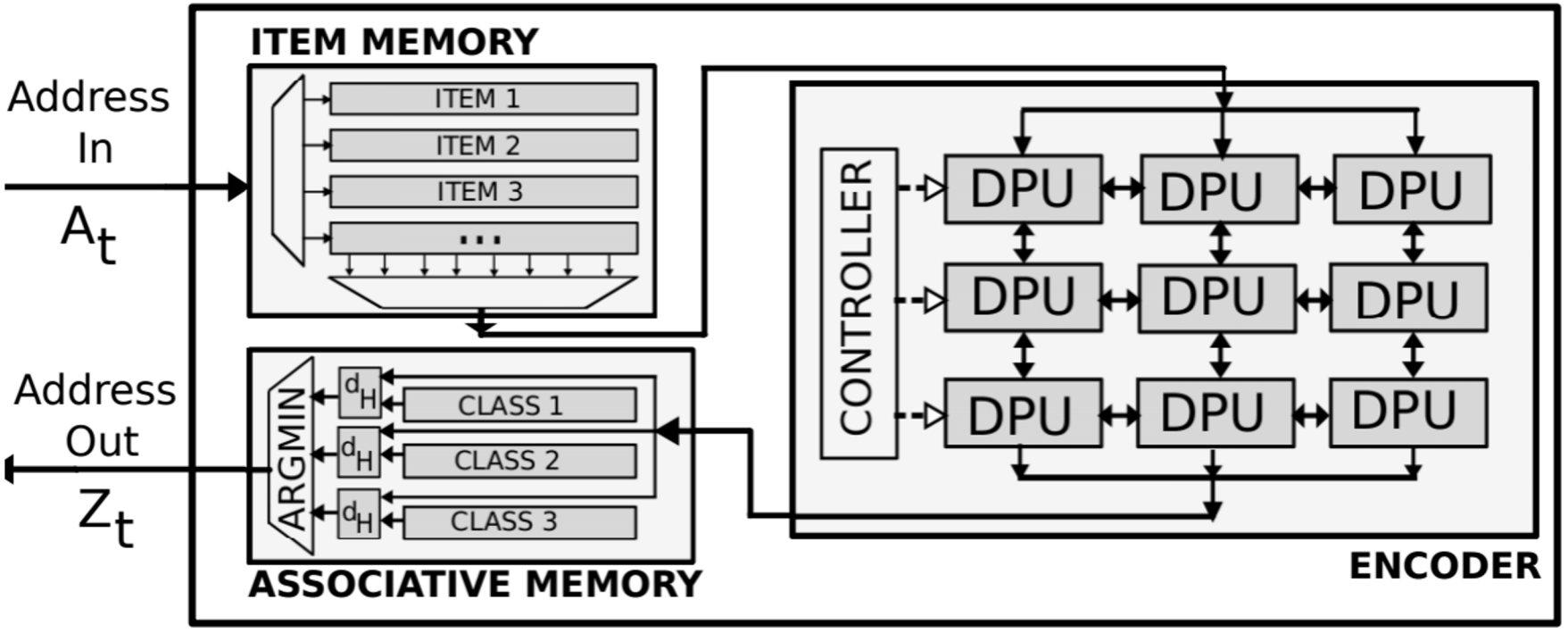
XOR ADD RIGHT SHIFT



Nearly General-Purpose HD Encoding Architectures

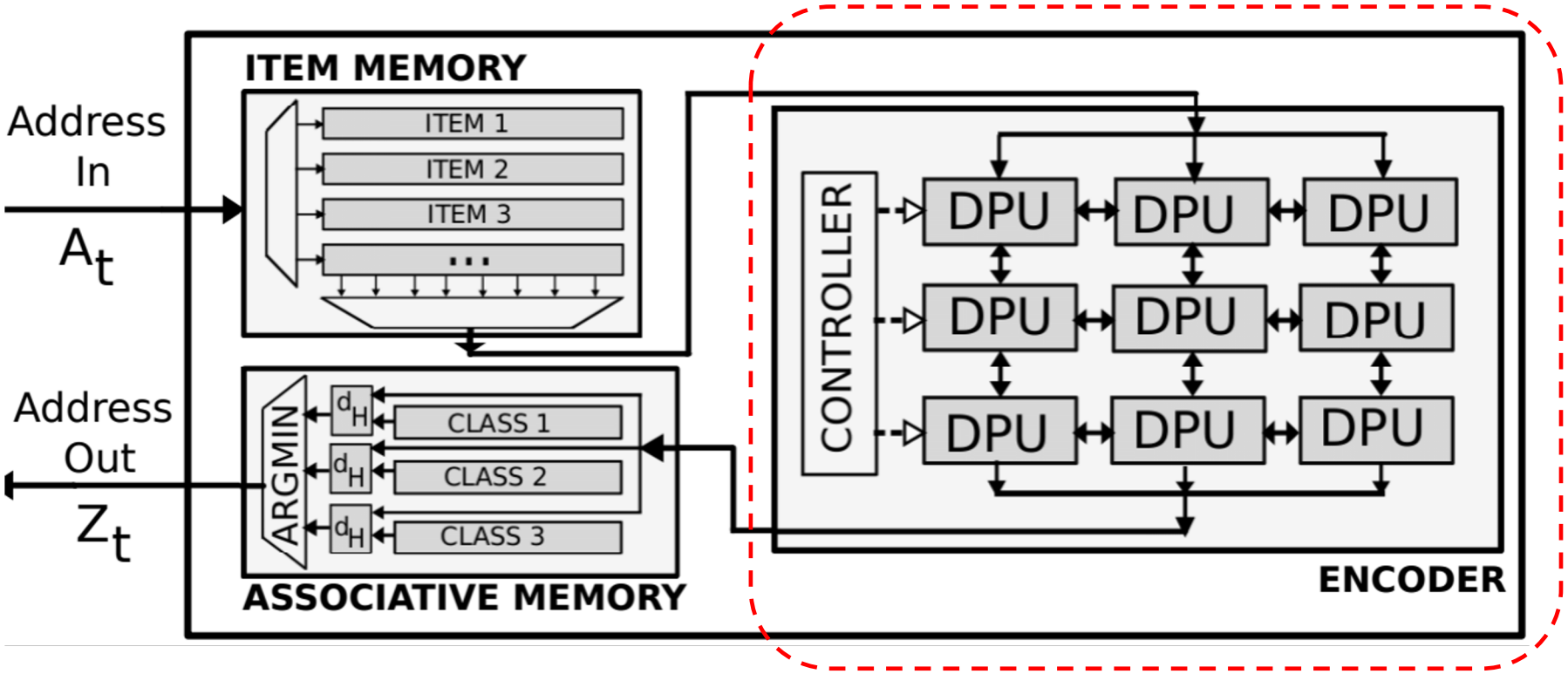
[Slides Courtesy of Sohum Datta, UC Berkeley]

Binary HD Computing Processor Architecture



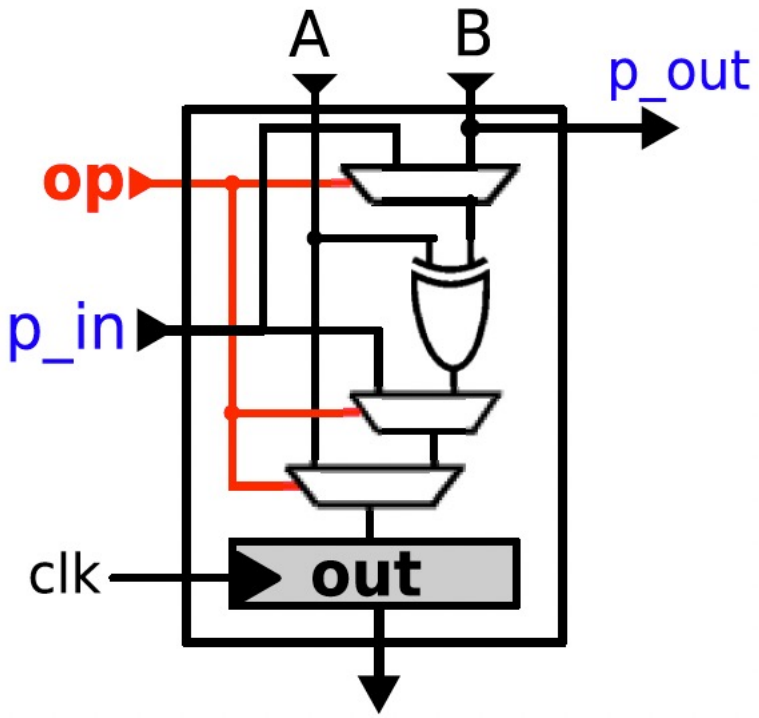
- **Item Memory:** Stores the randomly generated high-dimensional vectors.
- **Encoder:** Performs computations on high-dimensional vectors to produce learned representations.
- **Associative Memory:** Stores learned representations and computes distance between them and test vector representations for prediction.

Binary HD Computing Processor Architecture



- **Unidirectional Dataflow Architecture:** No reconfiguration in the Item Memory or Associative Memory
- **Generic Encoding:** Reconfiguration is possible in the Encoder, through DPUs and control signals

Programmable HD Encoder

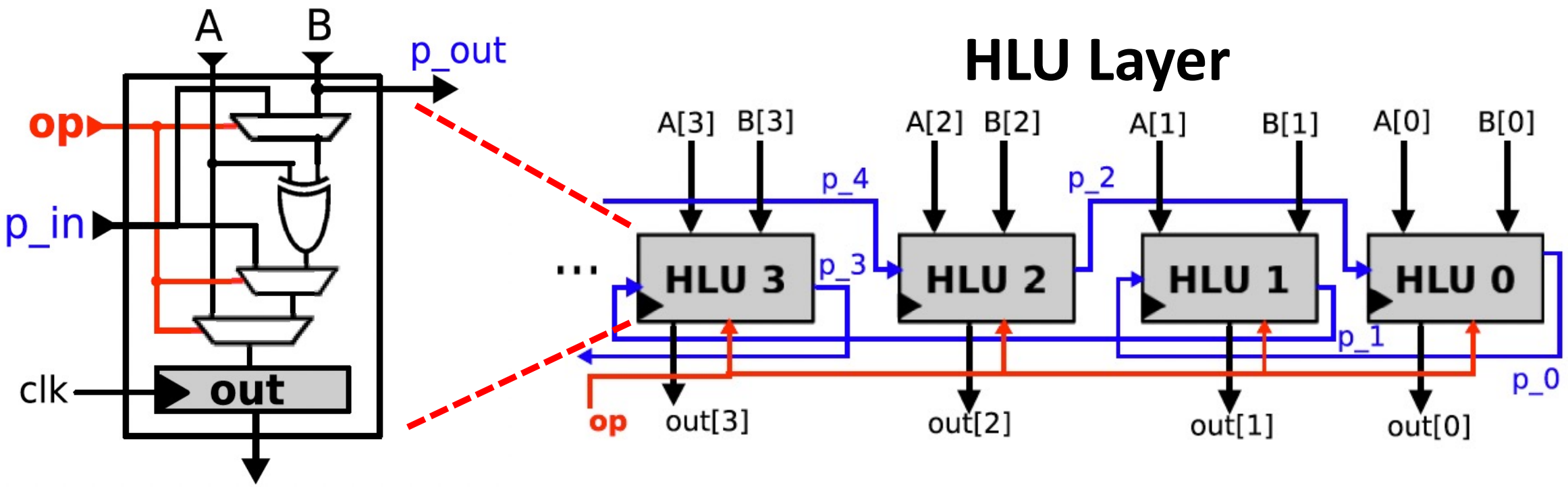


Hyperdimensional Logic Unit (HLU)

"The simplest (single-bit) DPU"

- **Multiply** $(C = A \oplus B)$
- **Permute** $(C = \rho(A))$
- **Delay** $(C = A)$
- **Permute-and-multiply** $(C = A \oplus \rho(B))$

Programmable HD Encoder



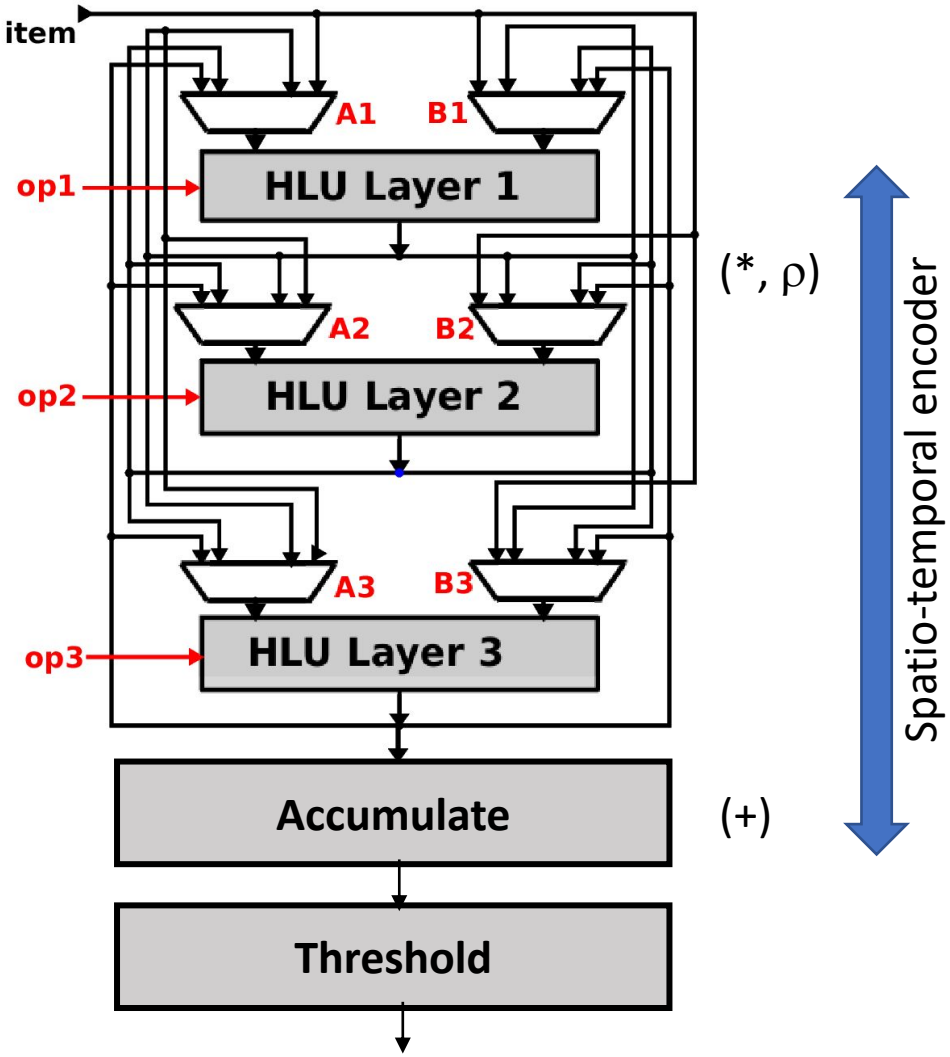
Hyperdimensional Logic Unit (HLU)

"The simplest (single-bit) DPU"

- **Multiply** $(C = A \oplus B)$
- **Permute** $(C = \rho(A))$
- **Delay** $(C = A)$
- **Permute-and-multiply** $(C = A \oplus \rho(B))$

Permute: Any Hamiltonian path connection through p_in and p_out is valid… **But we need to take physical routing into consideration (Minimize wire length and routing congestion) …**

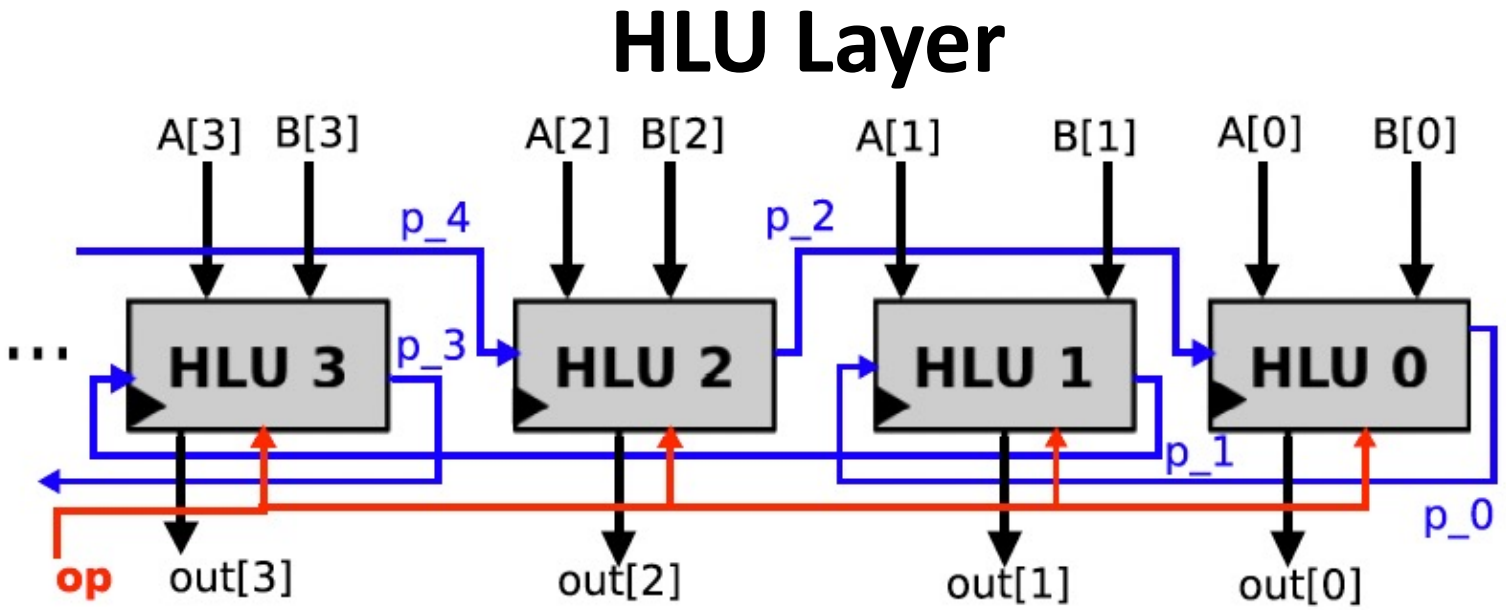
Programmable HD Encoder



(*, ρ)

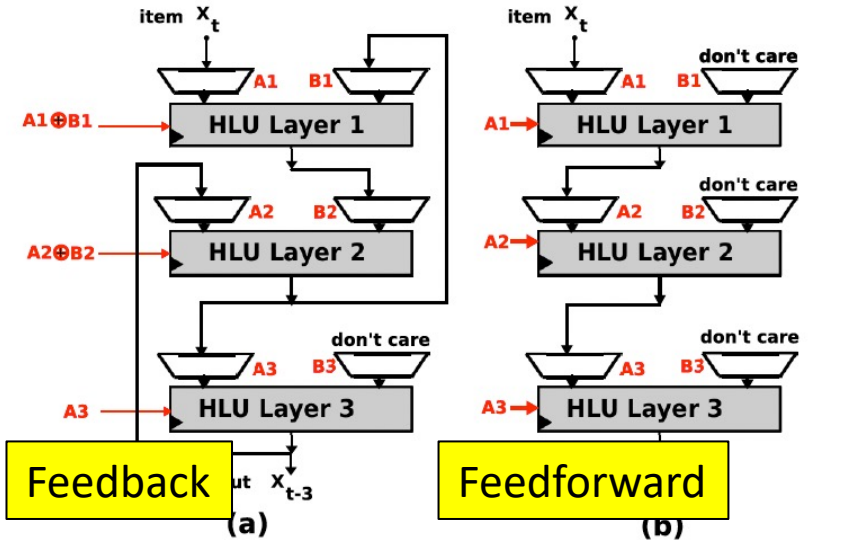
(+)

Spatio-temporal encoder



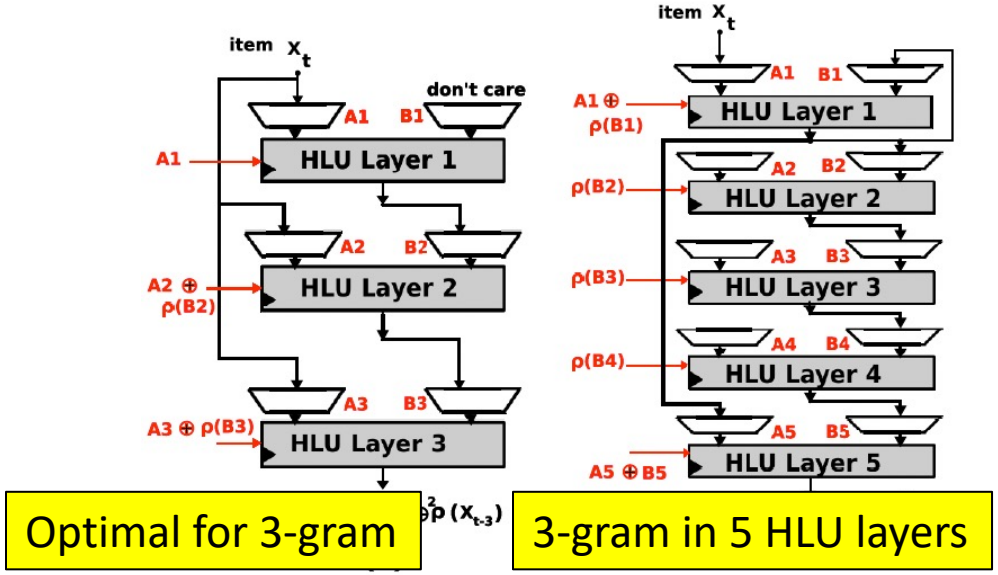
Accumulator: Perform superposition at each component
Threshold: Perform binarization (MSB of counters are used)

Encoder Parameters: HLU Network, Depth, Width



Feedback

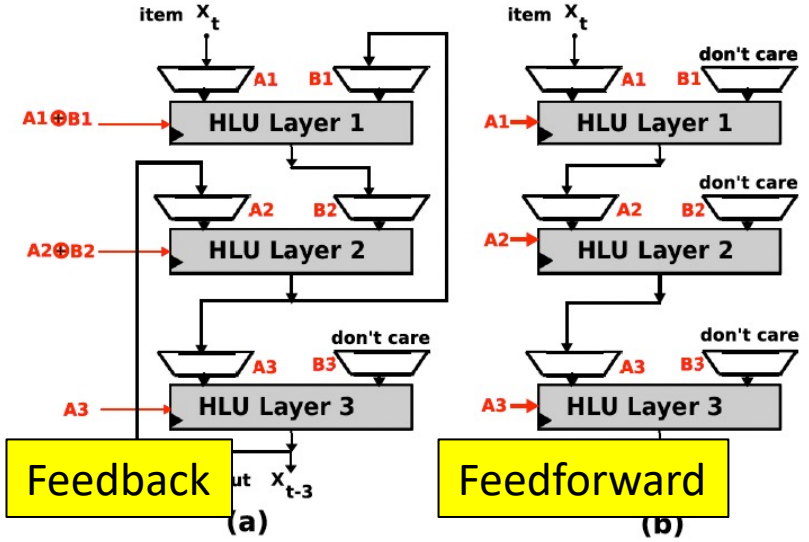
Feedforward

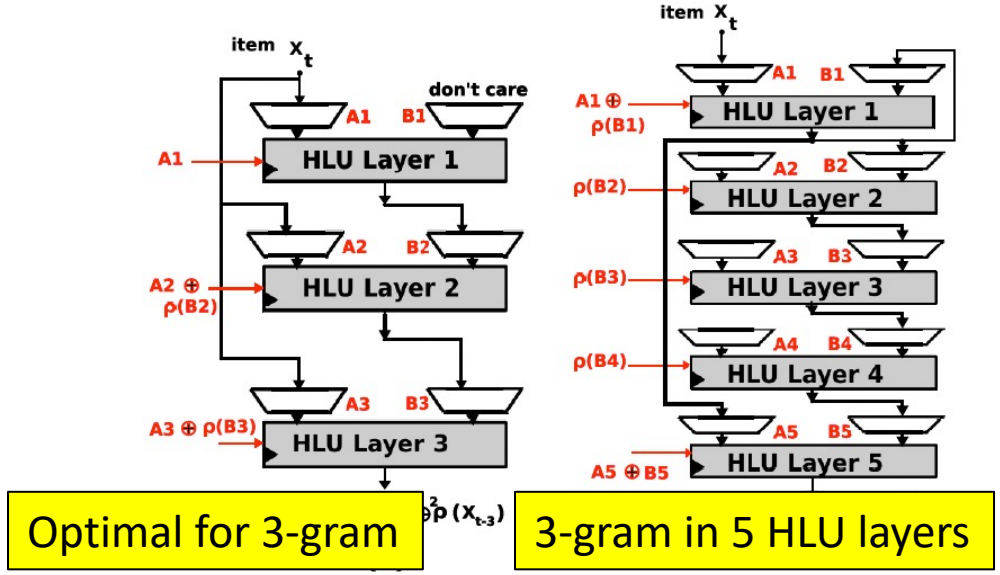


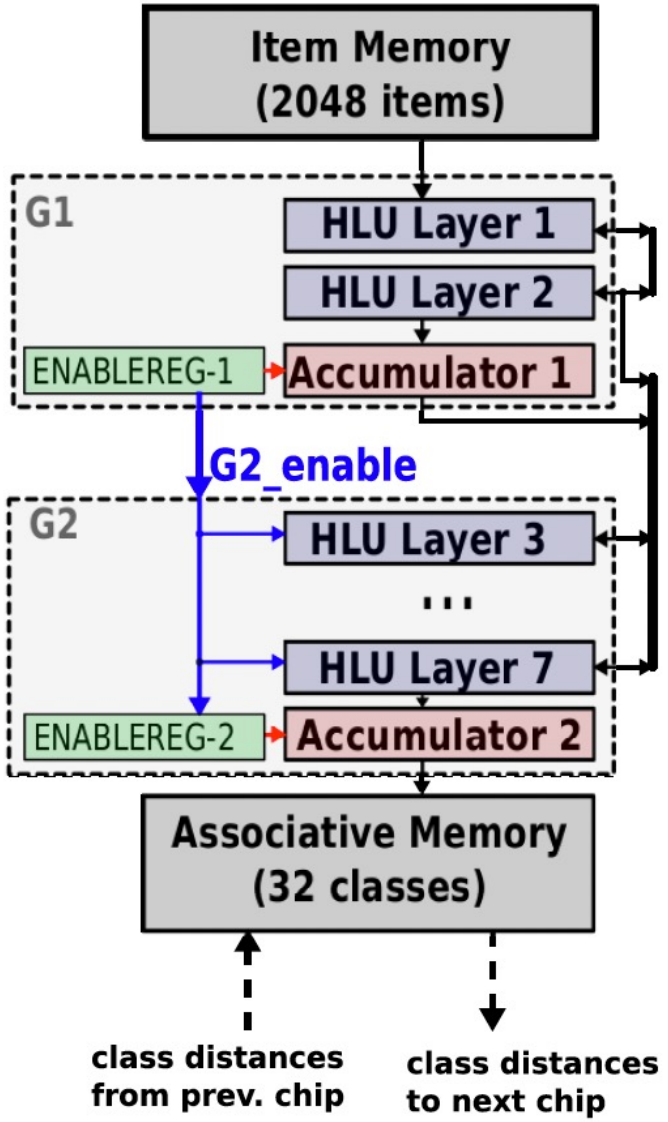
Optimal for 3-gram

3-gram in 5 HLU layers

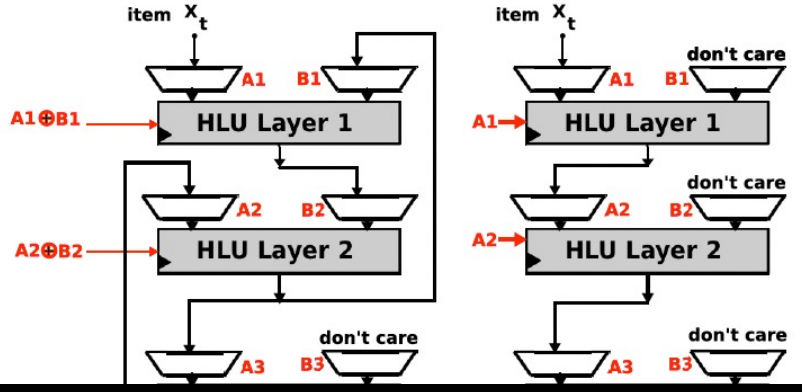
Encoder Parameters: HLU Network, Depth, Width

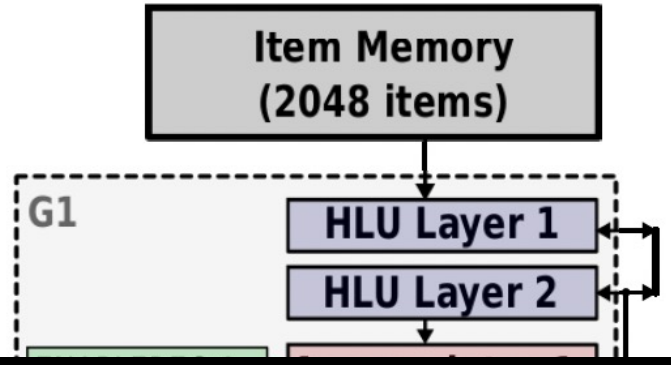






Encoder Parameters: HLU Network, Depth, Width



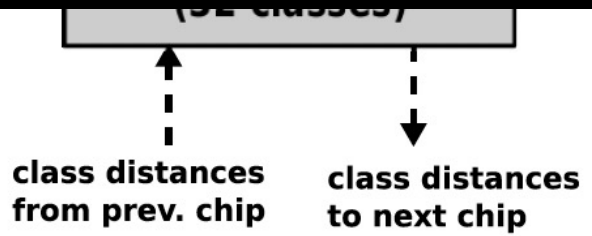


7-layer configurable pipelined data-path to efficiently implement space-time encoders encountered in streaming-based learning and classification tasks

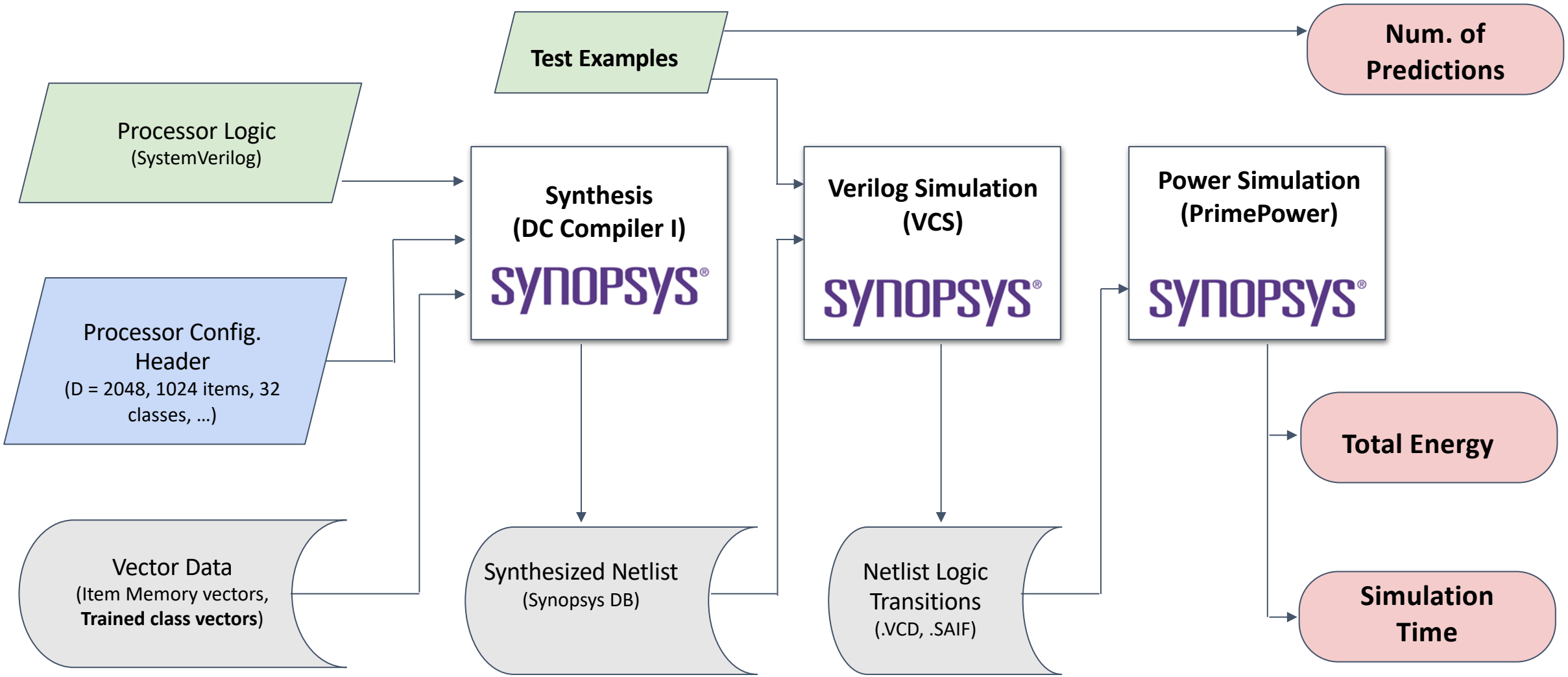
Width = 2048 bits

Optimal for 3-gram

3-gram in 5 HLU layers



CMOS Chip Design Flow

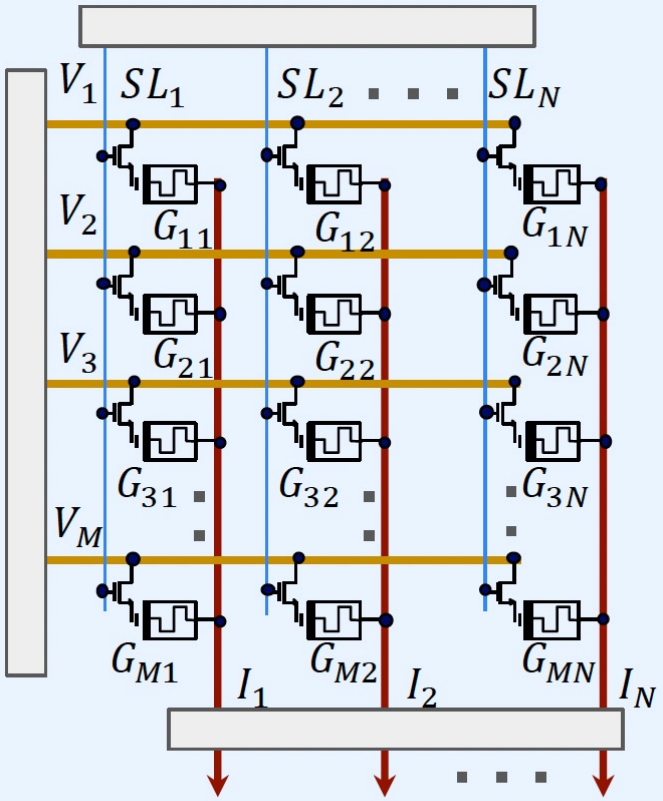


Energy/prediction = Total Energy / Num. of Predictions; Latency = Simulation Time / Num. of Predictions

Introducing Analog Computing and Non-Volatile Memories

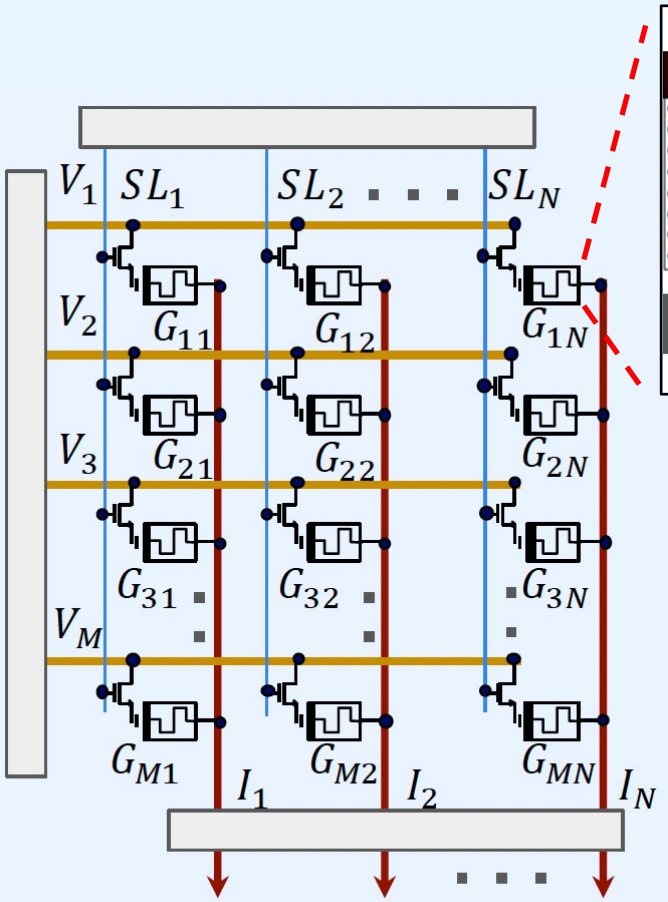
[Slides Courtesy of G. Karunaratne, IBM Research]

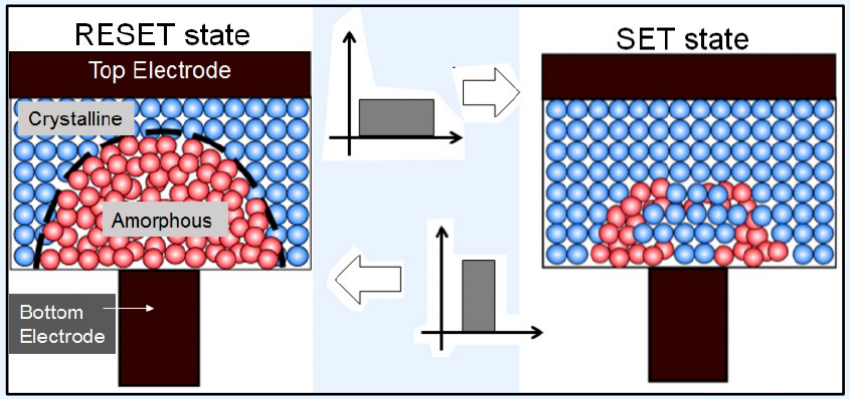
In-Memory HD Computing



$$I = G * V$$

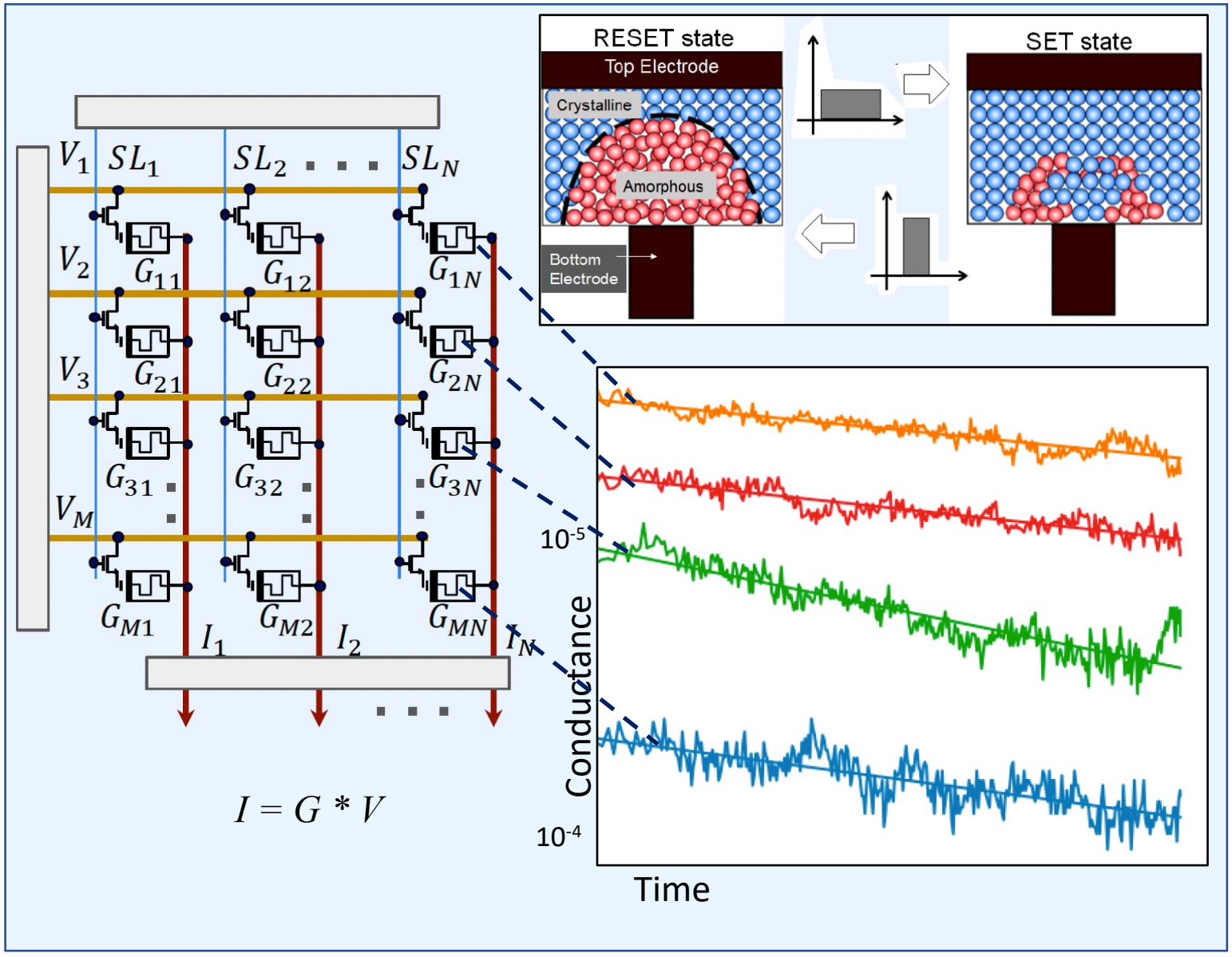
In-Memory HD Computing



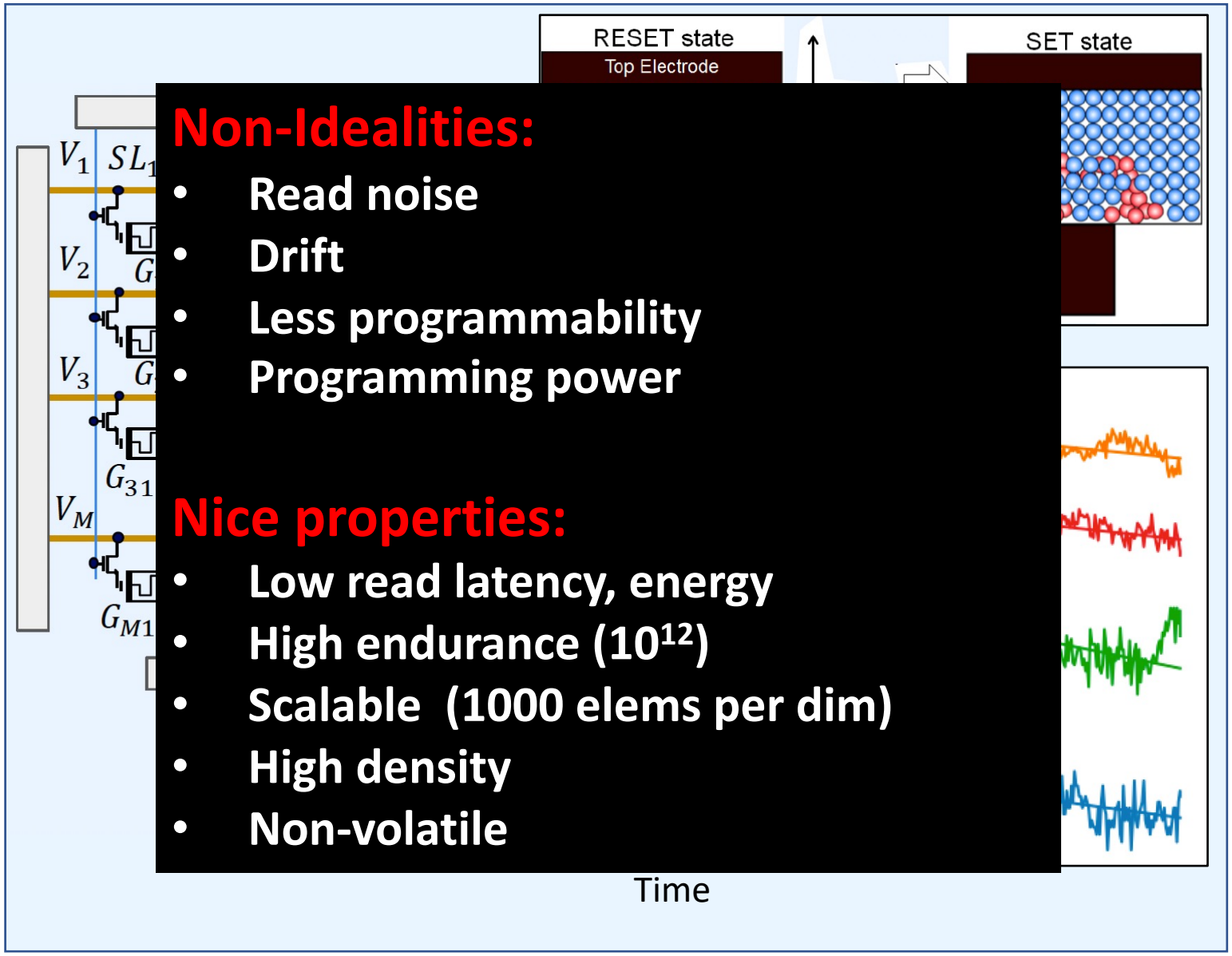


$$I = G * V$$

In-Memory HD Computing



In-Memory HD Computing



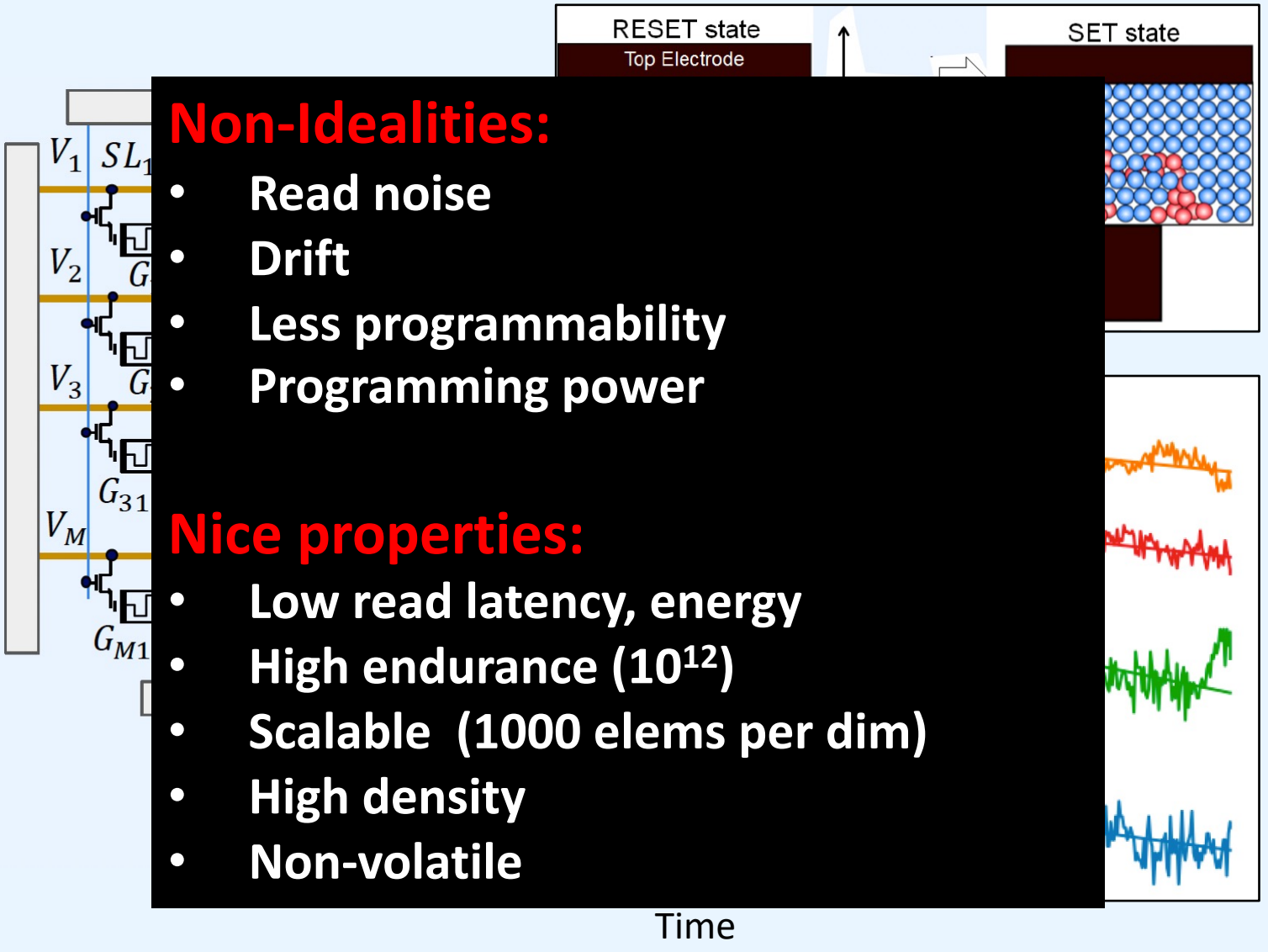
Non-Idealities:

- Read noise
- Drift
- Less programmability
- Programming power

Nice properties:

- Low read latency, energy
- High endurance (10^{12})
- Scalable (1000 elems per dim)
- High density
- Non-volatile

In-Memory HD Computing



Non-Idealities:

- Read noise
- Drift
- Less programmability
- Programming power

Nice properties:

- Low read latency, energy
- High endurance (10^{12})
- Scalable (1000 elems per dim)
- High density
- Non-volatile

HD Computing

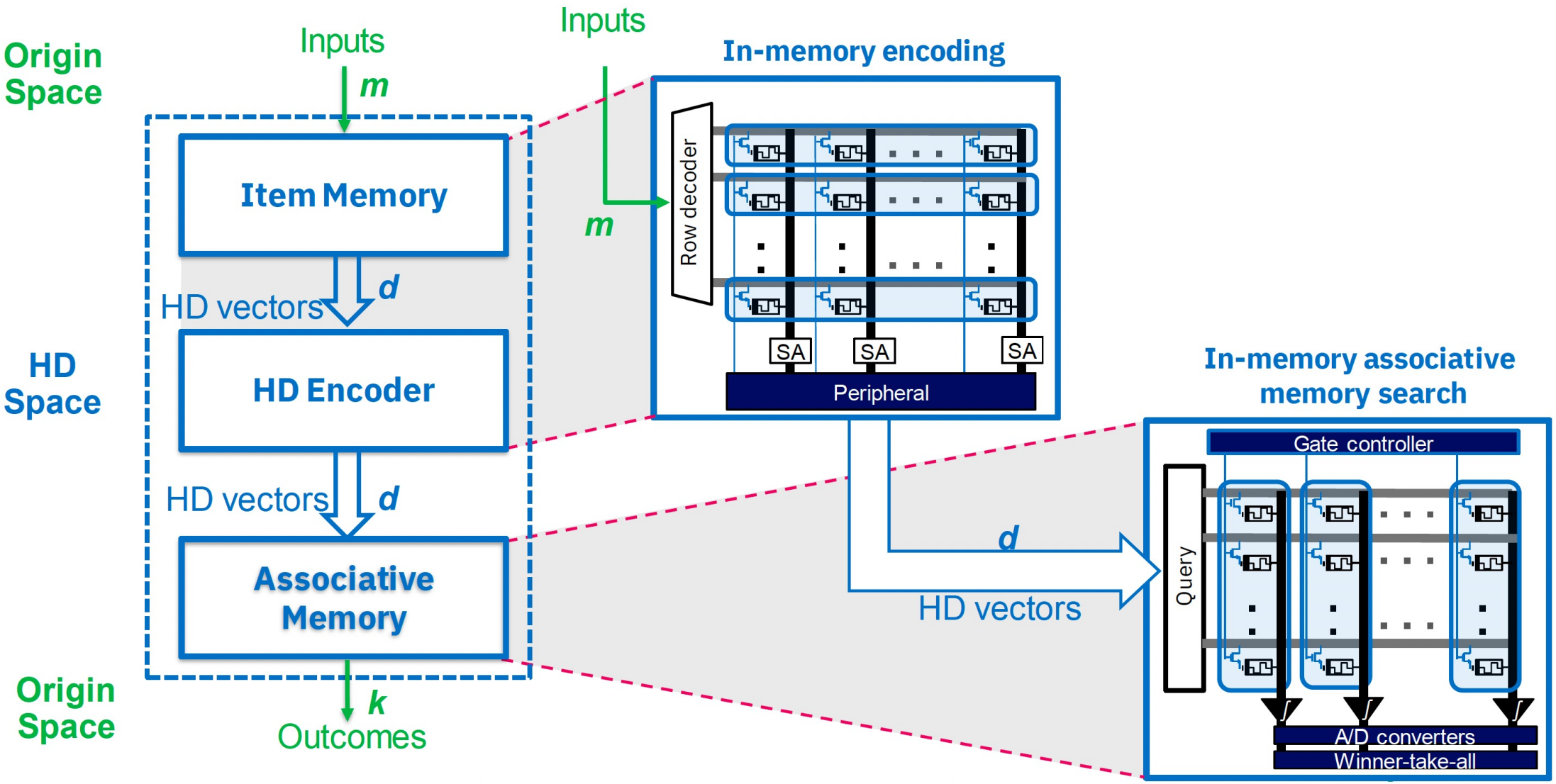
Challenges:

- Von Neumann bottleneck
- Massively parallel

Nice properties:

- Well-defined arithmetic operations
- Robust

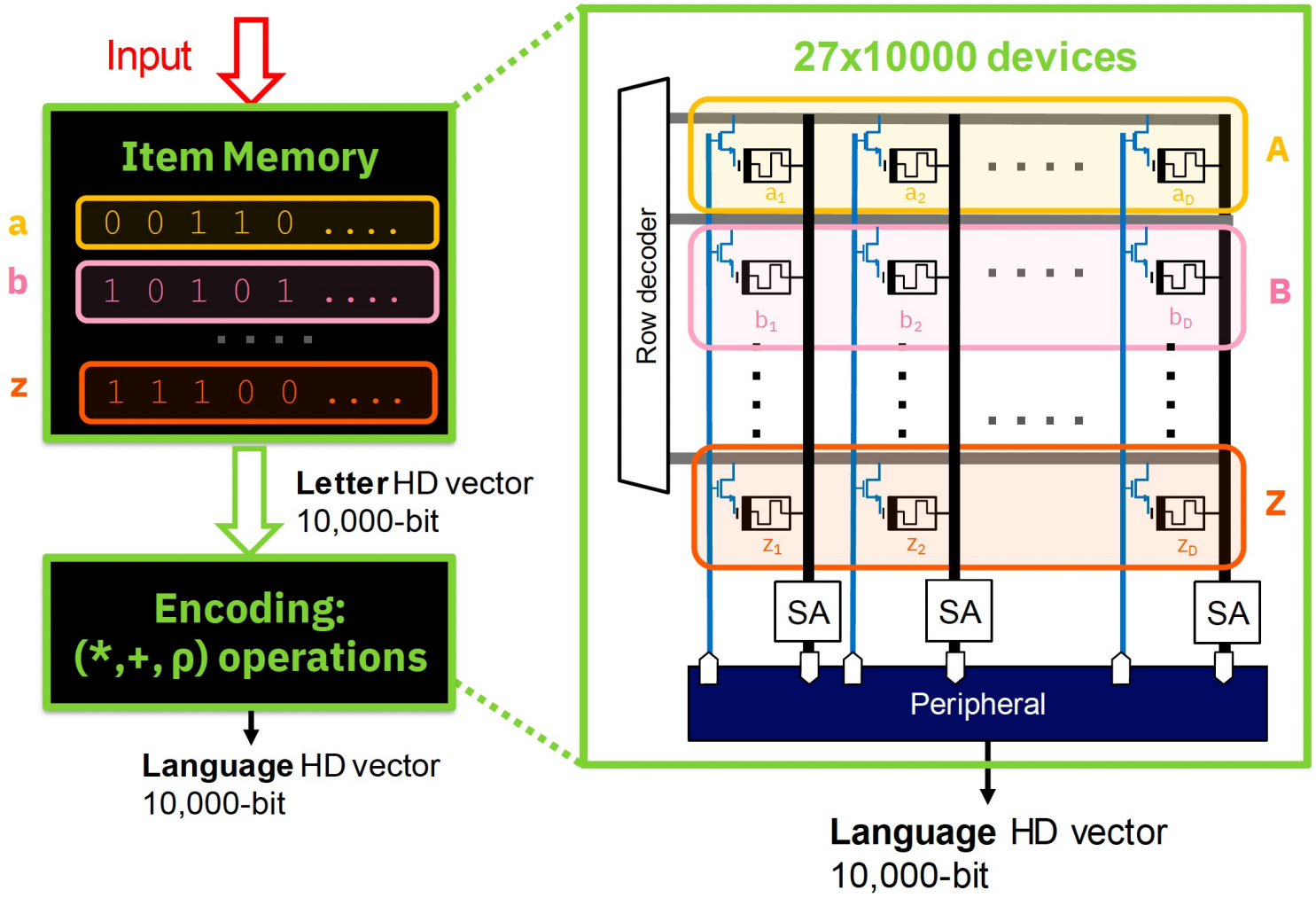
In-Memory HD Computing



In-Memory HD Encoding

- **Goal**: Combine the basis (**letter**) HD vectors from the Item Memory to create:
  - The prototype (**language**) HD vectors representing each class, **or**
  - The **query** HD vector (from unknown language) for **inference**

- Item Memory HD vectors encoded in conductance states of memristive devices

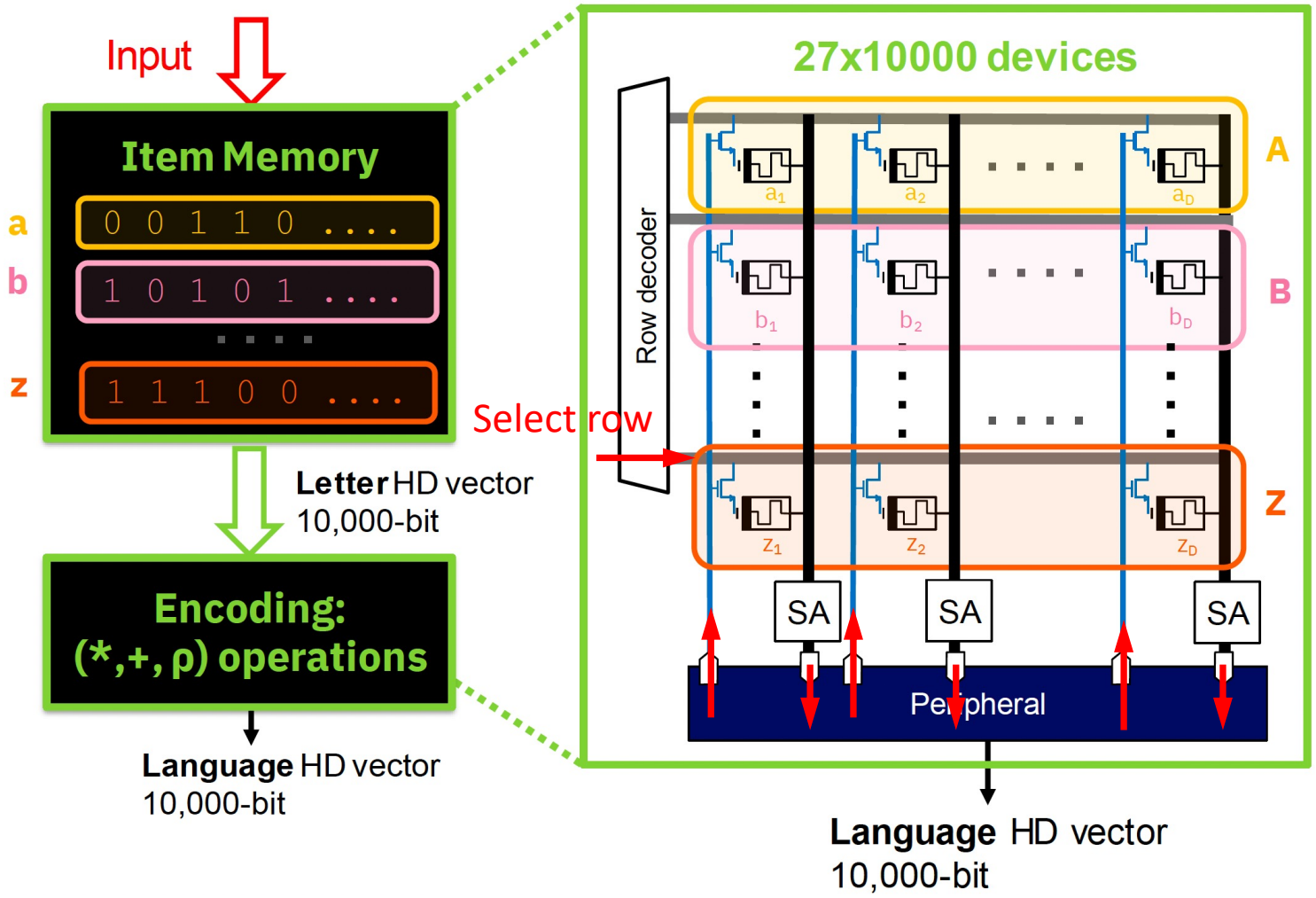


In-Memory HD Encoding

- **Goal**: Combine the basis (**letter**) HD vectors from the Item Memory to create:
  - The prototype (**language**) HD vectors representing each class, **or**
  - The **query** HD vector (from unknown language) for **inference**

- Item Memory HD vectors encoded in conductance states of memristive devices

- Encoding operations performed using **in-memory read logic**



HD Encoding Based on 2-minterms

- How to find an encoding solution that suits in-memory read logic?
- **Minterm expansion** of XNOR function:

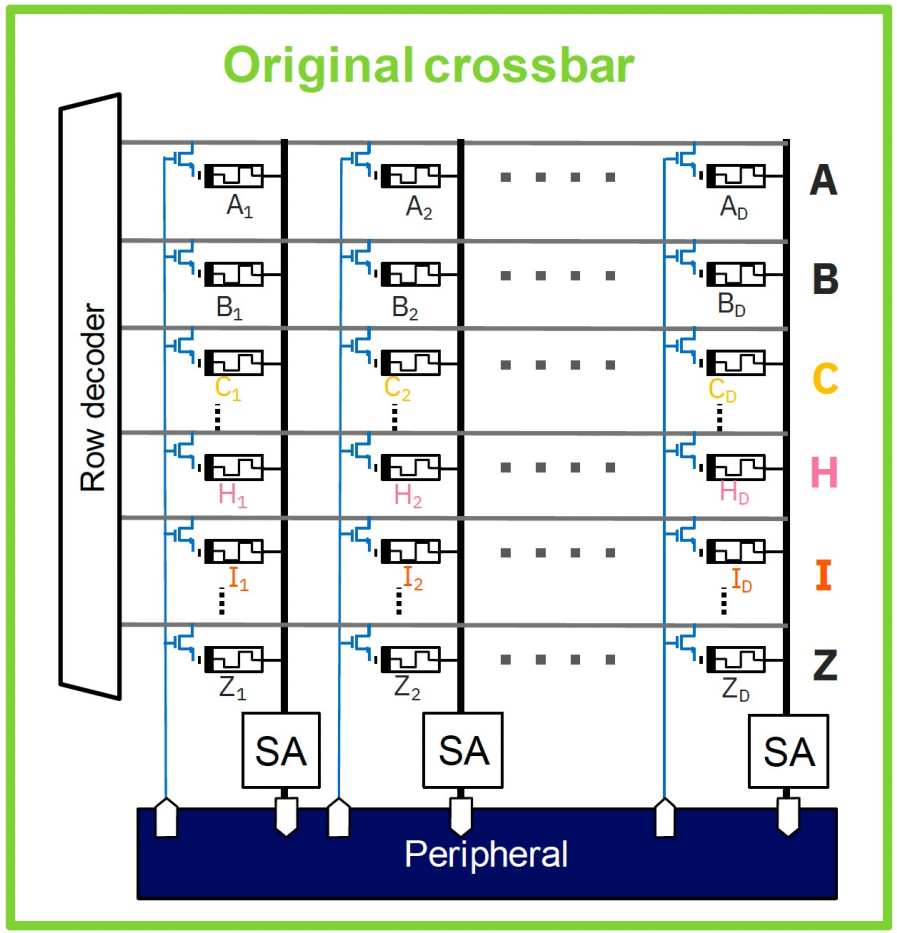
$$A\overline{\oplus}B = (A \wedge B) \vee (\bar{A} \wedge \bar{B})$$

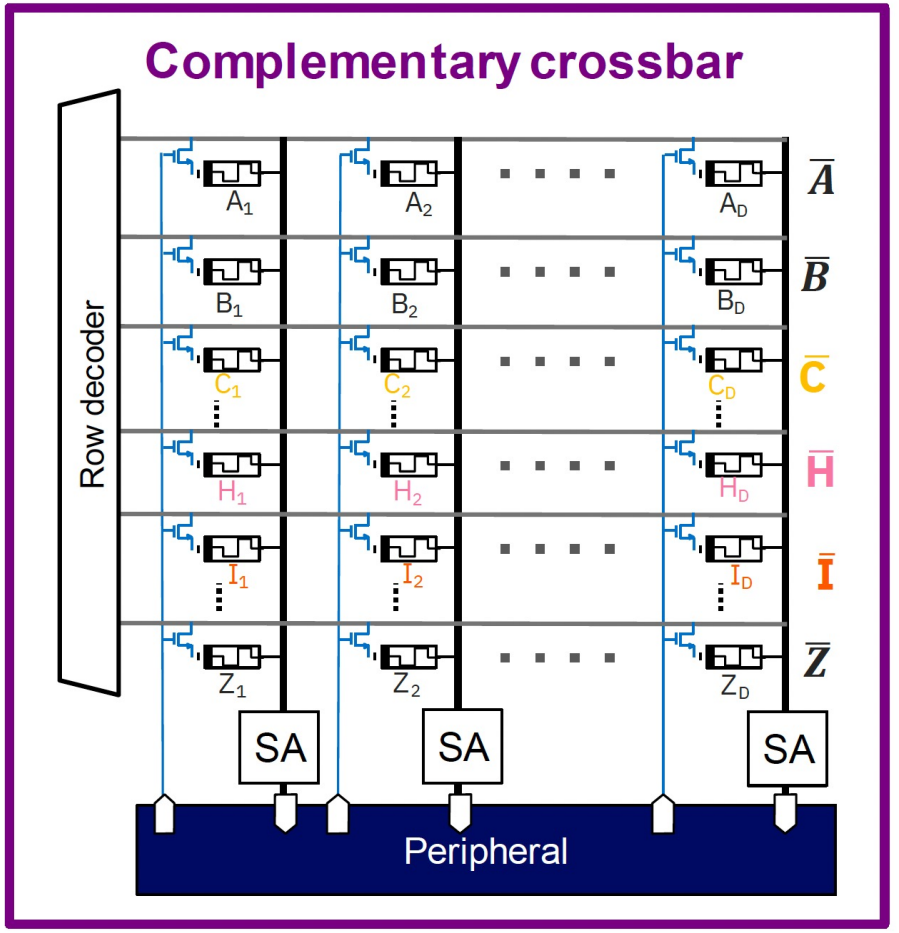
All-positive minterm

All-negative minterm

HD Encoding Based on 2-minterms

- Use two parallel crossbars to approximate the encoding dynamics
- Original and complementary item memories in two crossbars to produce 2-minterms

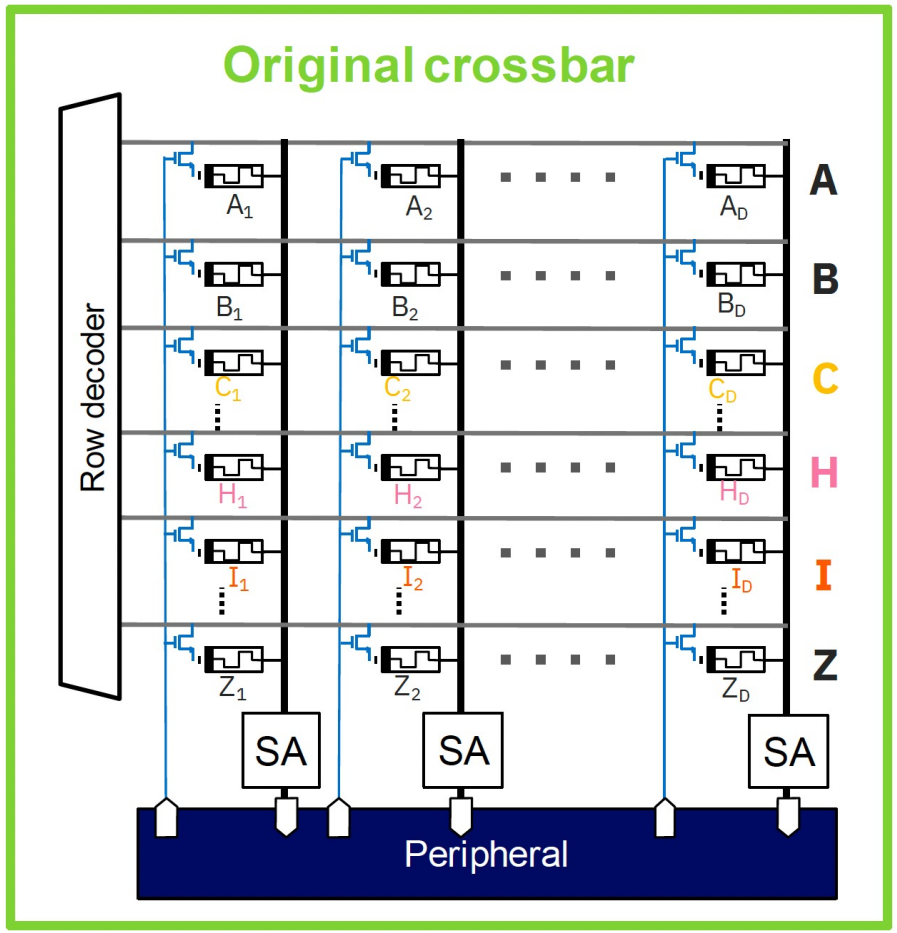


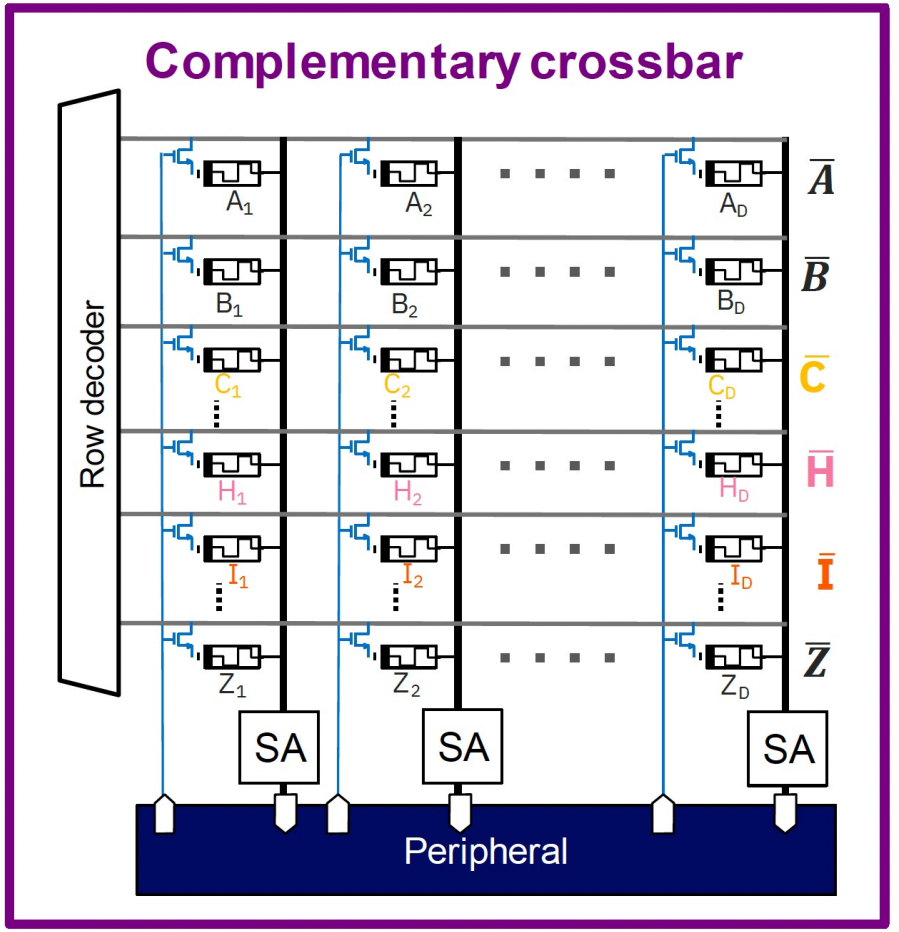


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

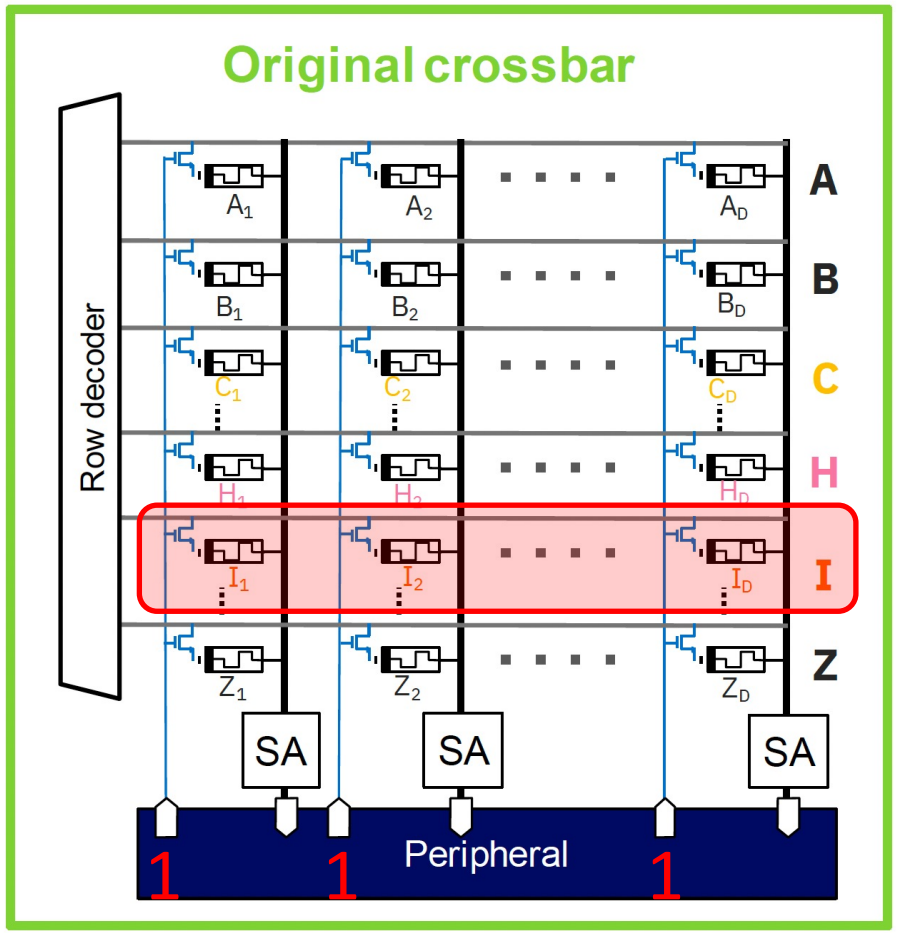


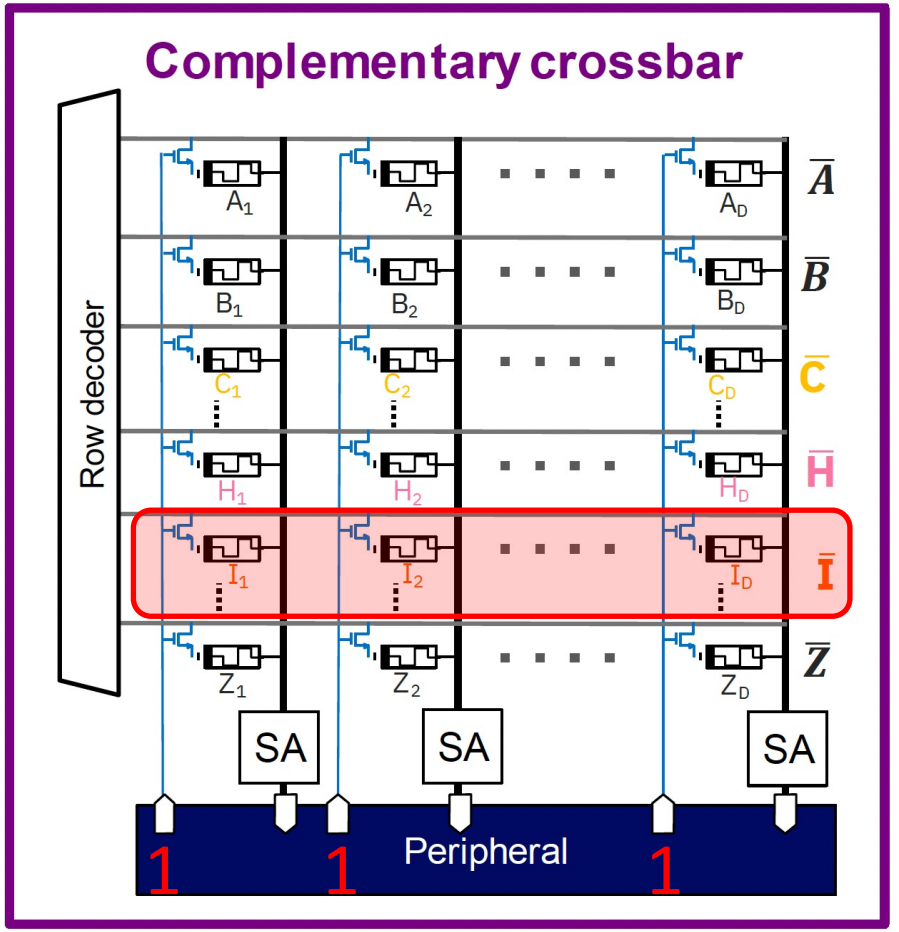


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

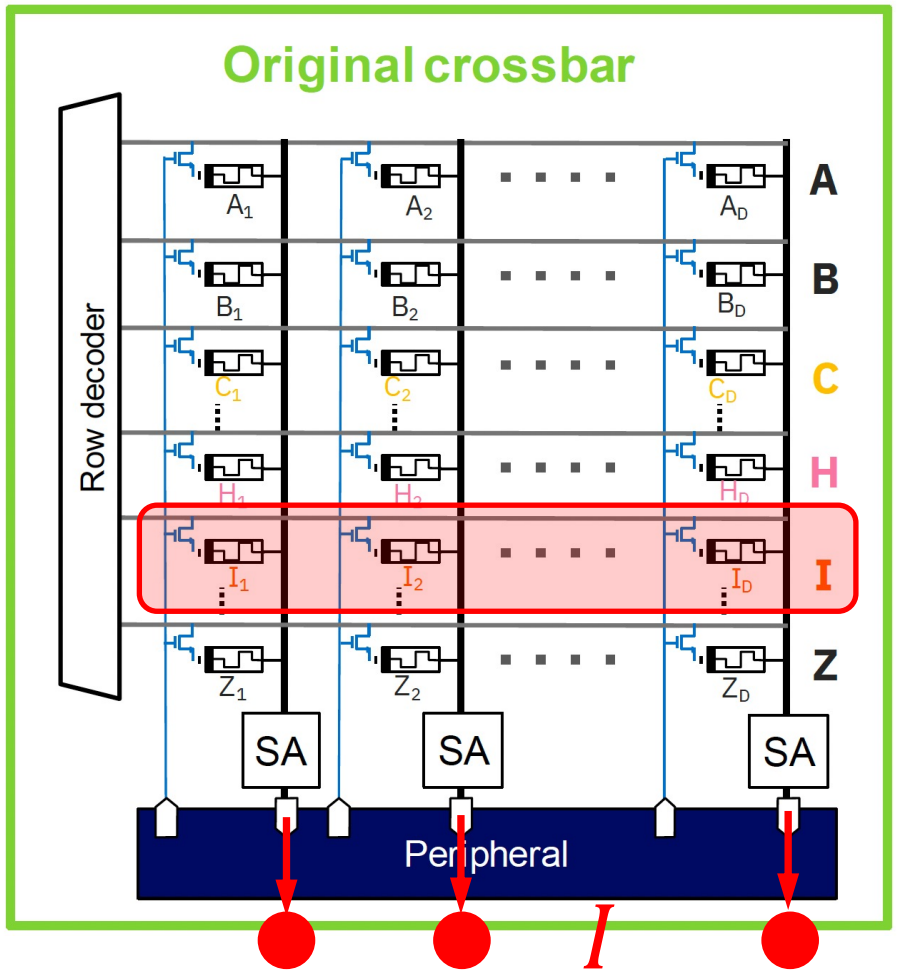


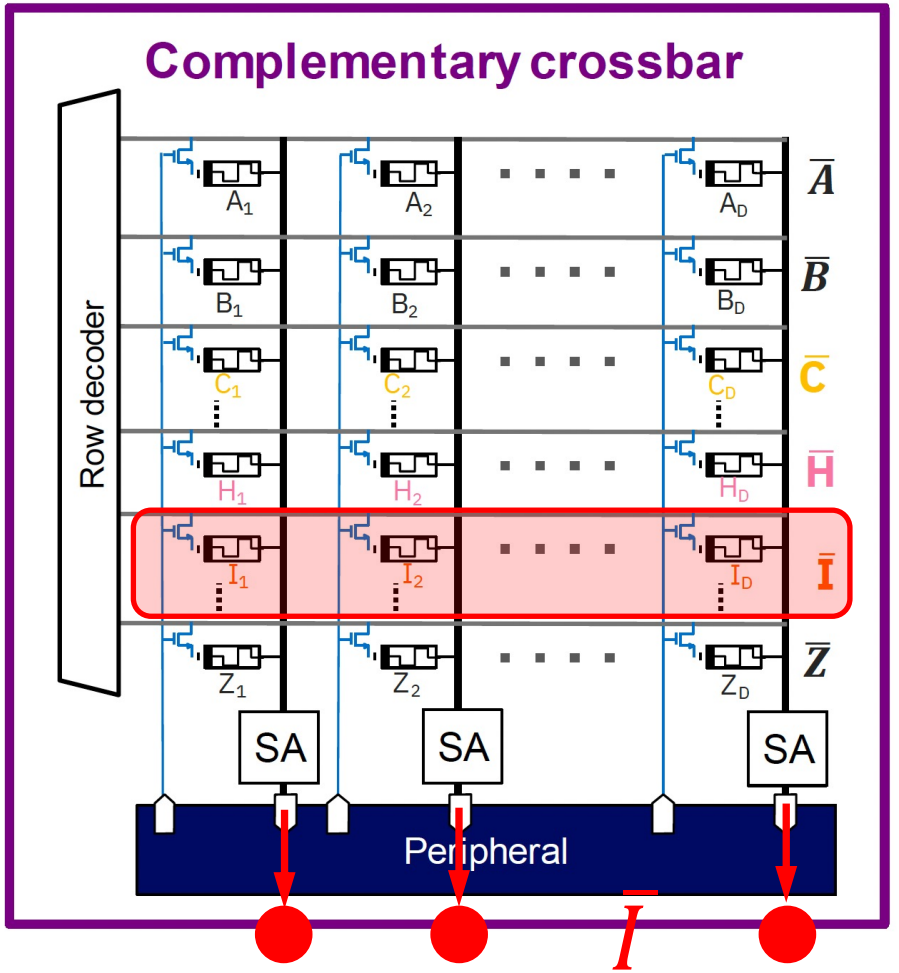


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

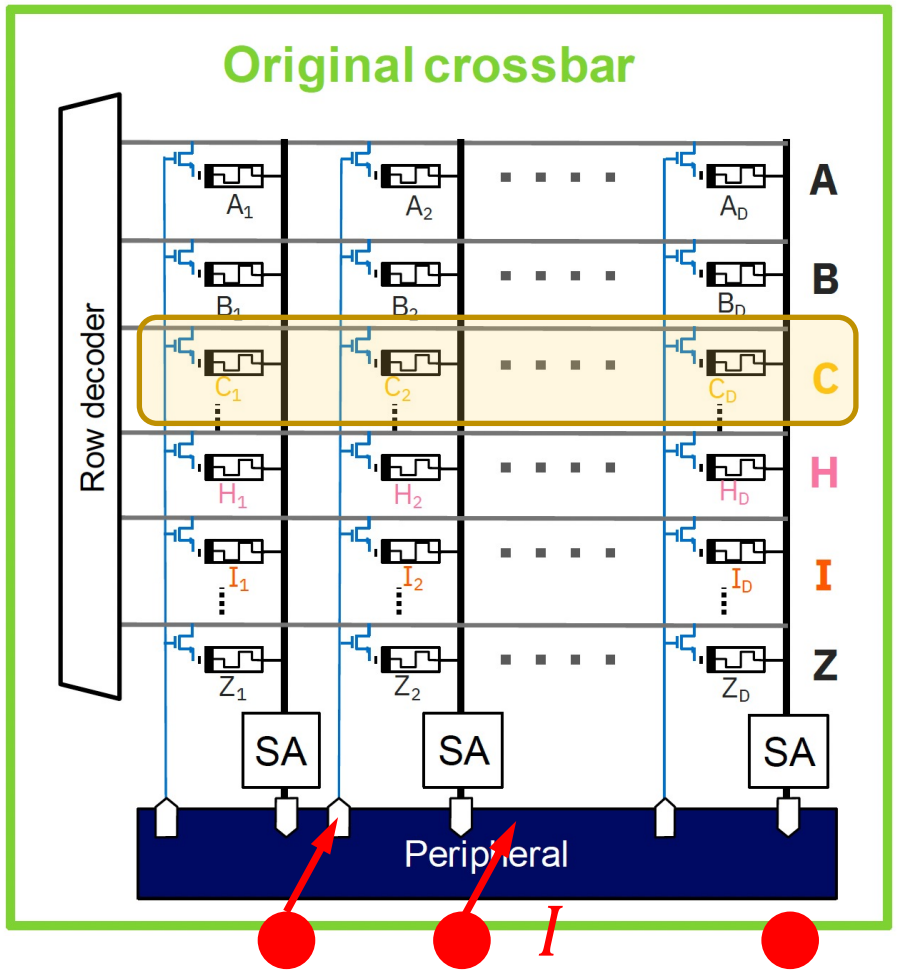


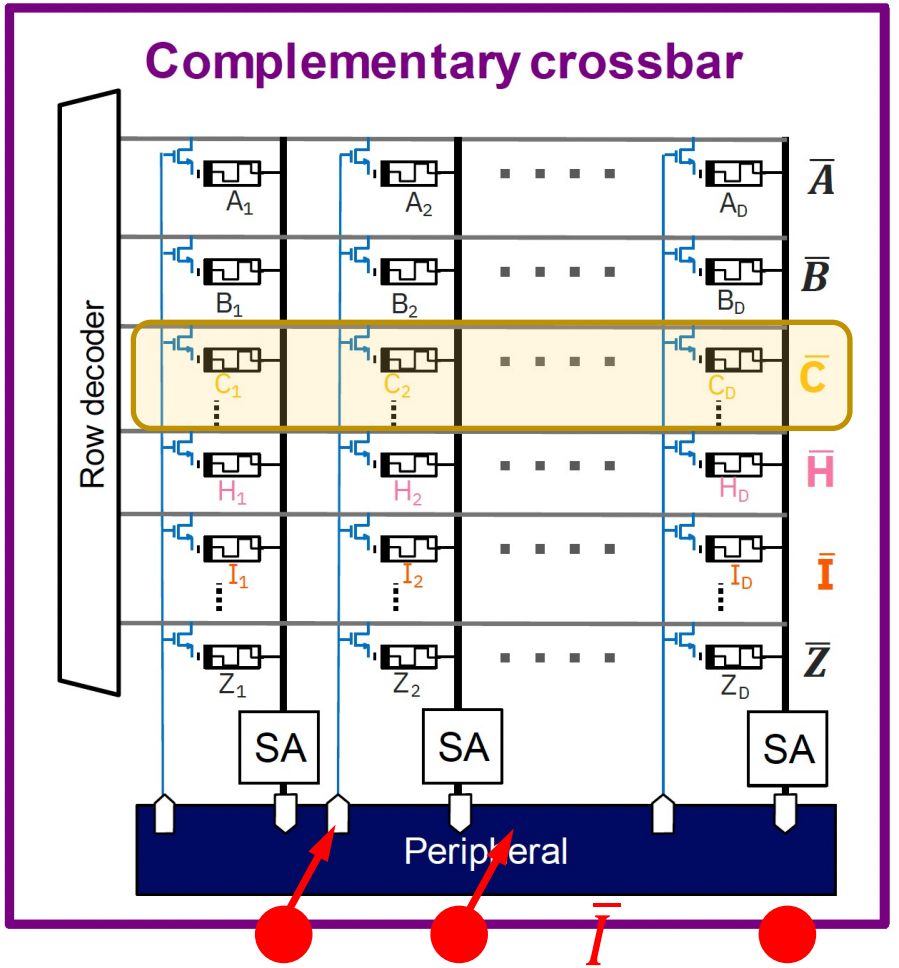


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

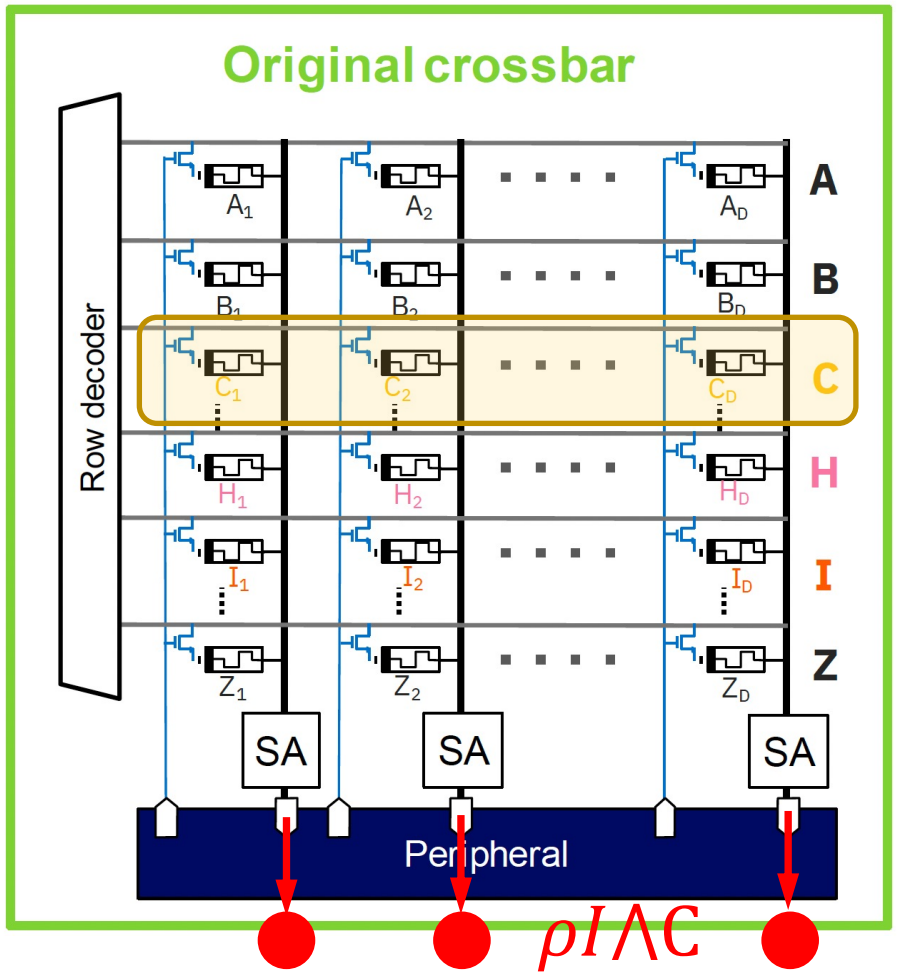


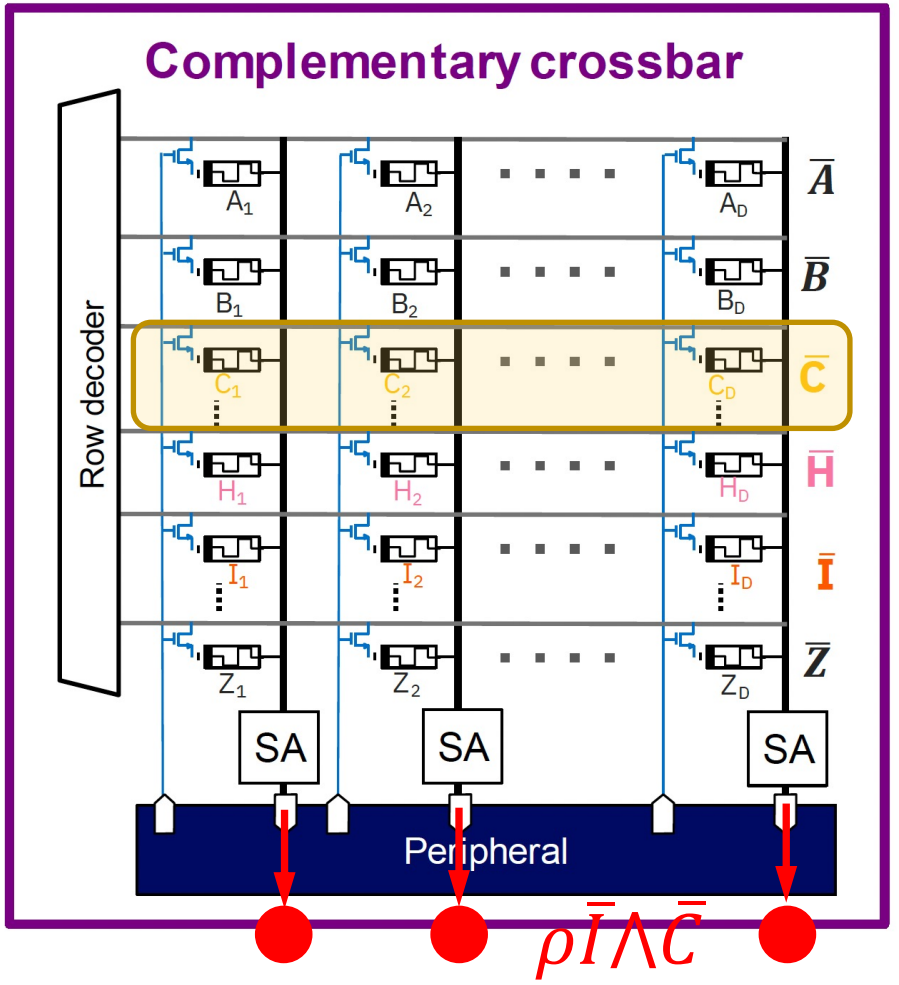


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

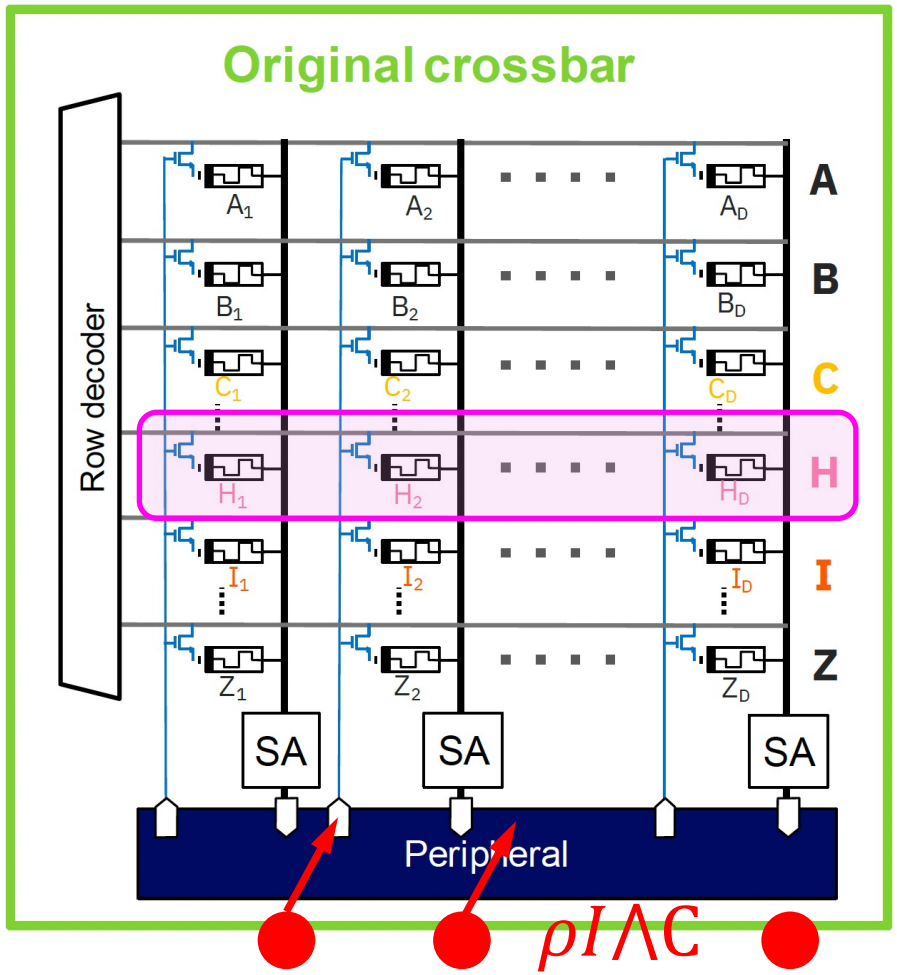


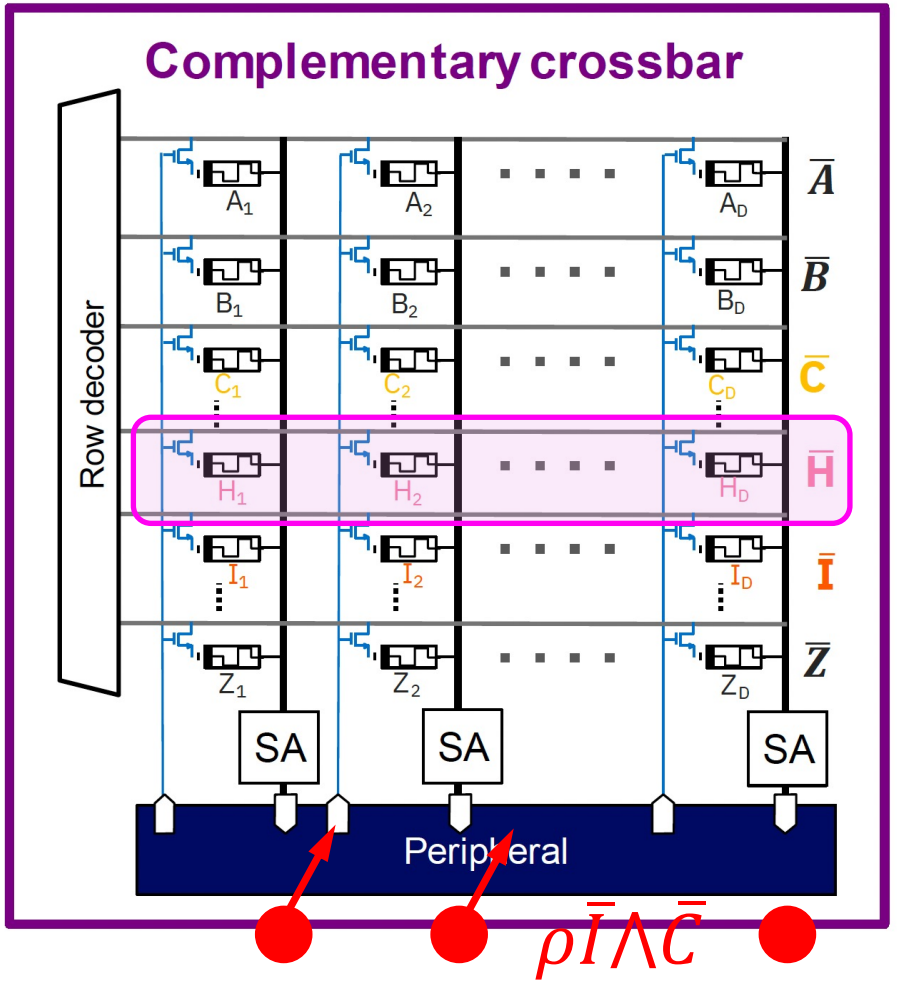


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

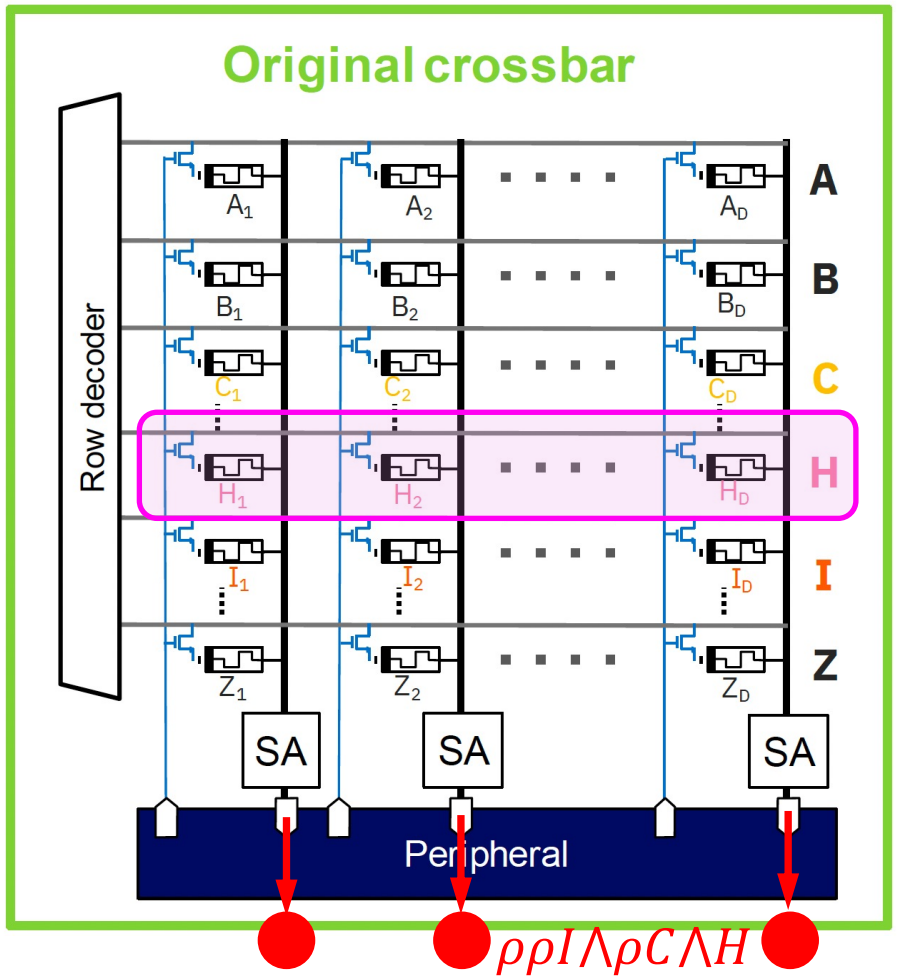


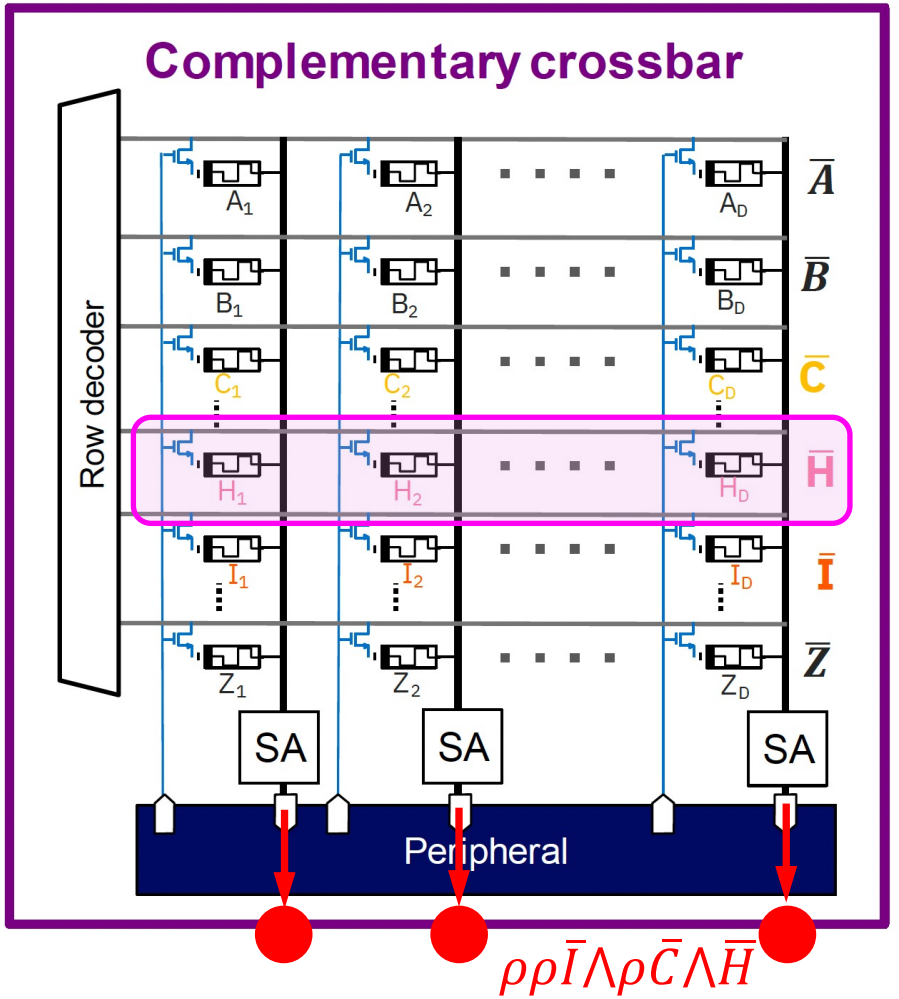


HD Encoding Based on 2-minterms

- Example:

$$ICH = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$

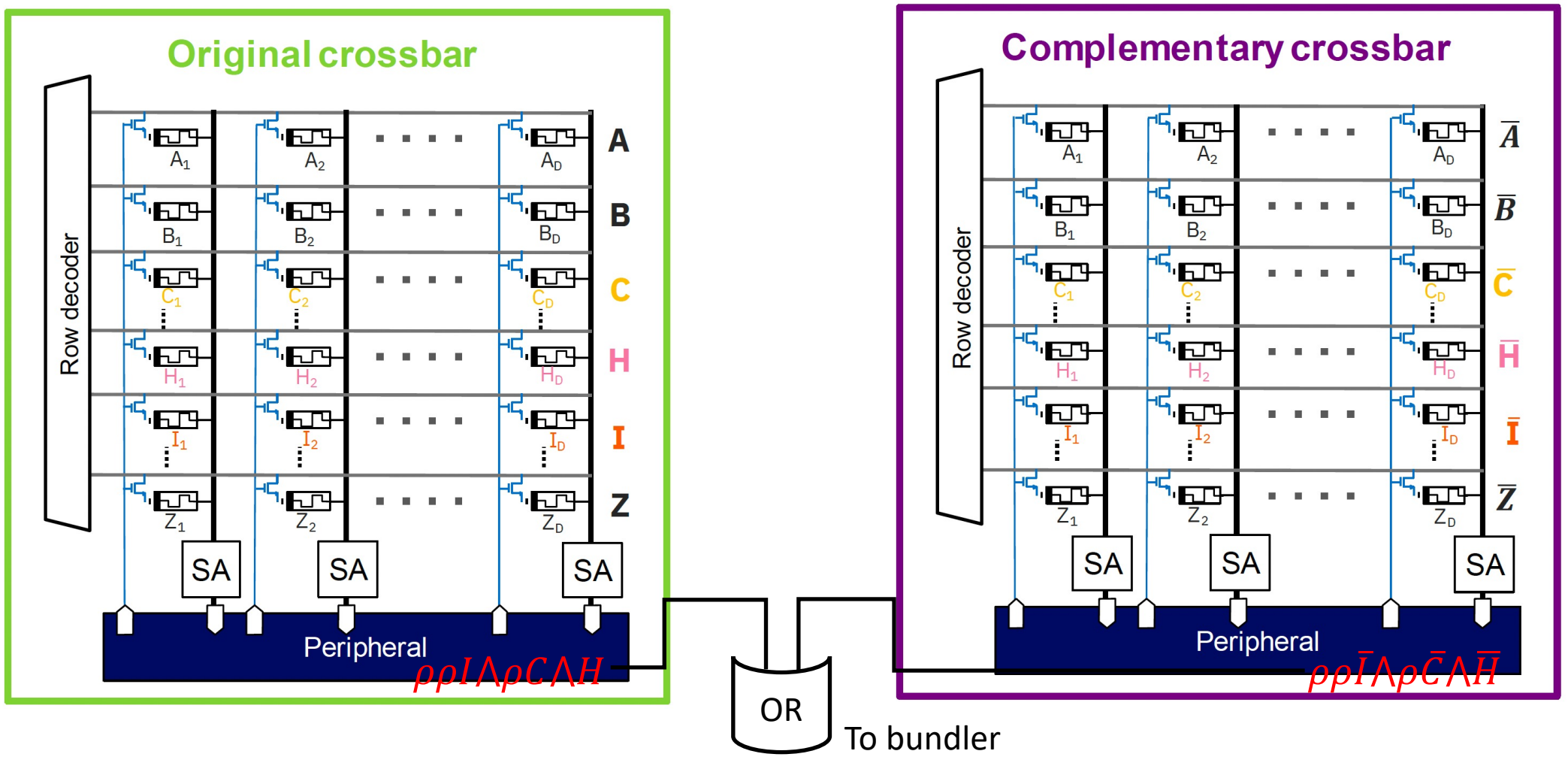




HD Encoding Based on 2-minterms

- Example:

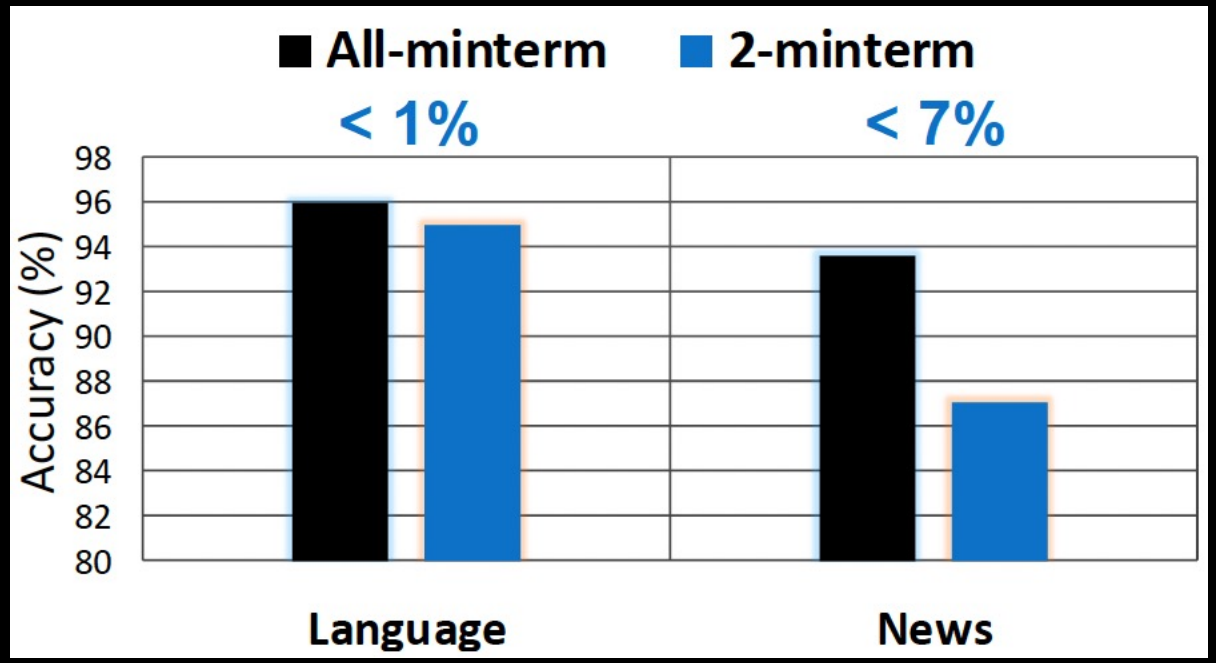
$$\boxed{ICH} = \rho\rho I * \rho C * H \approx (\rho\rho I \wedge \rho C \wedge H) \vee (\rho\rho \bar{I} \wedge \rho \bar{C} \wedge \bar{H})$$



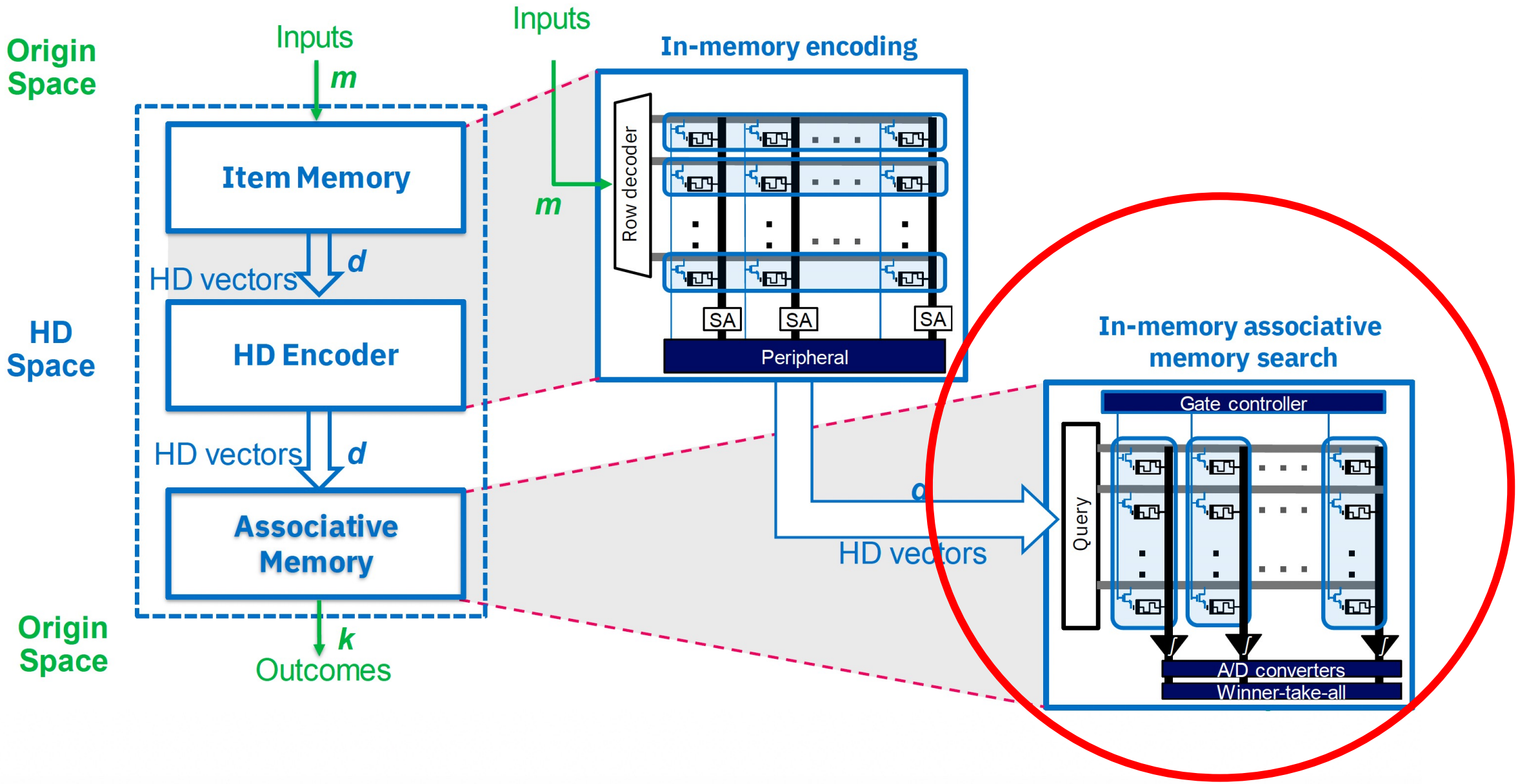
HD Encoding Based on 2-minterms

- Example:

2-minterm provides accuracy that is close to all-minterm:



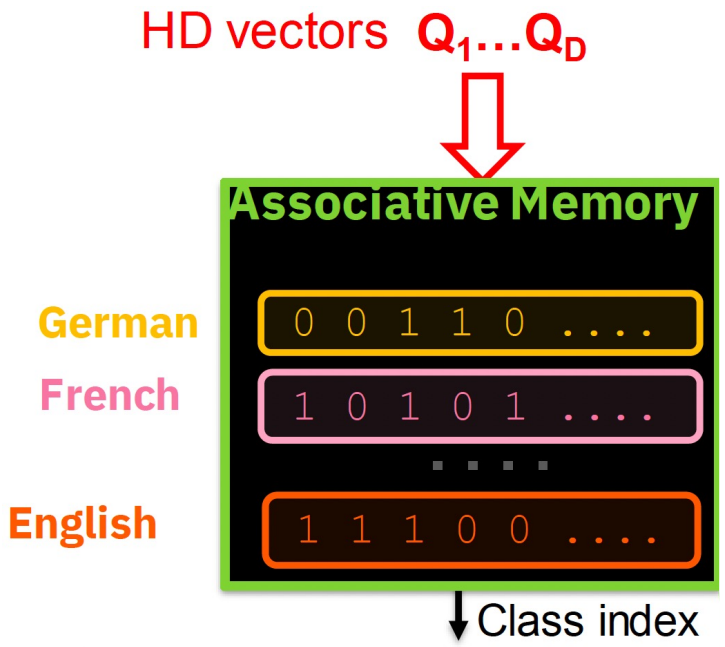
In-Memory Associative Search



In-Memory Associative Search

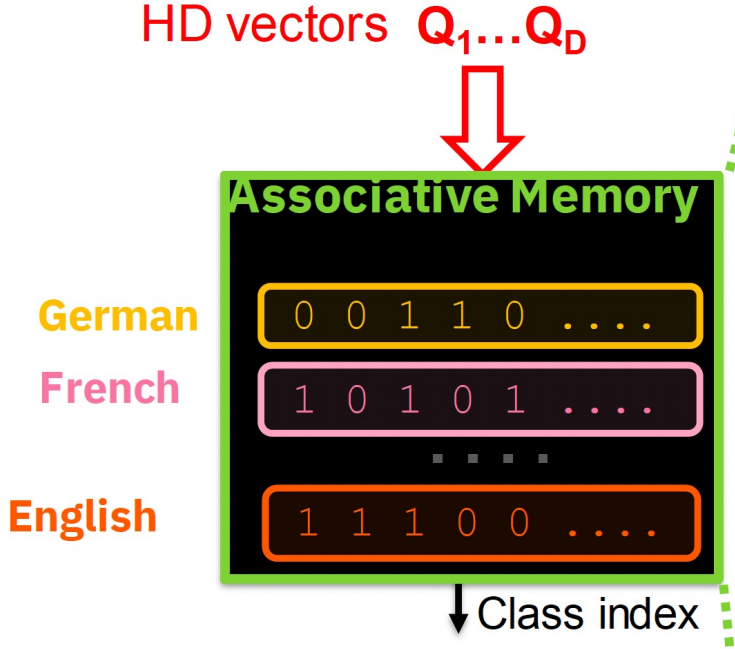
- **Goal**: Find which class (**language**) a **query** vector $\boldsymbol{Q}$ belongs to

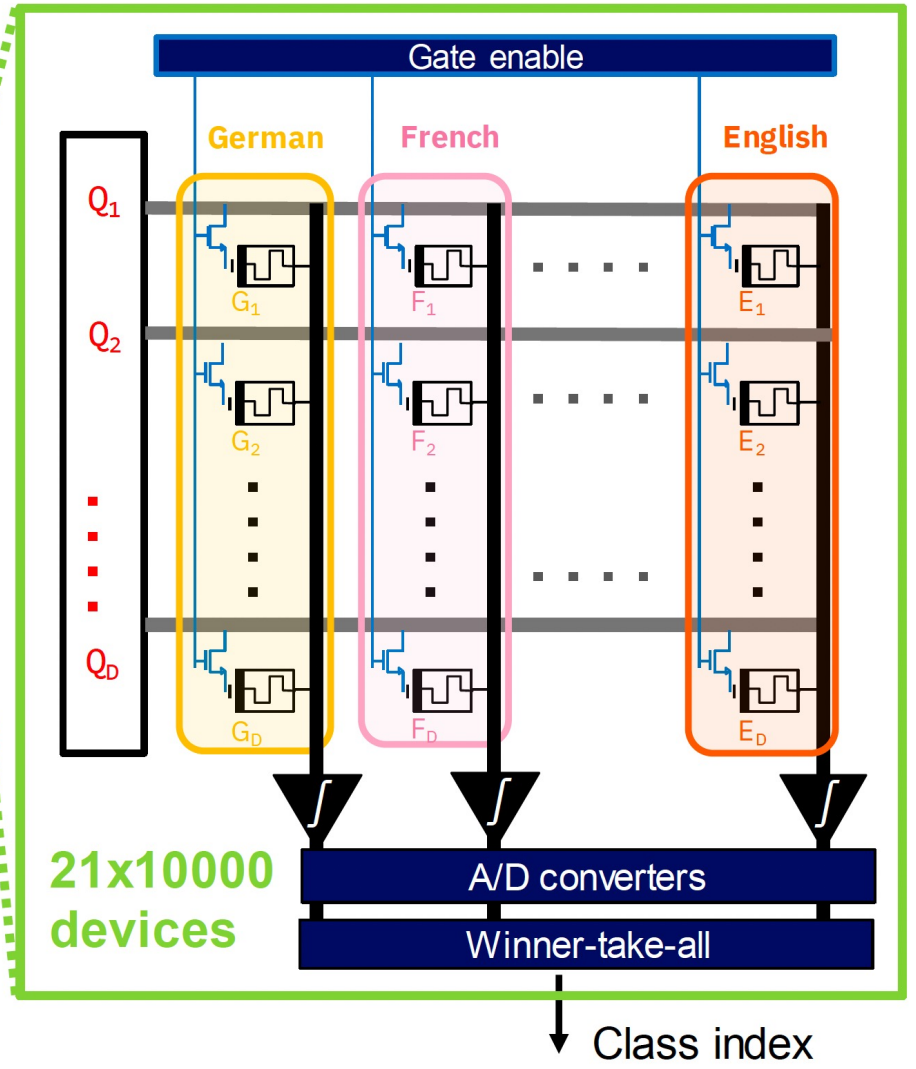
- Learned prototype vectors $\boldsymbol{P_i}$ encoded in conductance states of memristive devices



In-Memory Associative Search

- **Goal**: Find which class (**language**) a **query** vector $\boldsymbol{Q}$ belongs to
- Learned prototype vectors $\boldsymbol{P_i}$ encoded in conductance states of memristive devices
- Approximate Hamming distance by dot-product for hardware-friendly implementation:



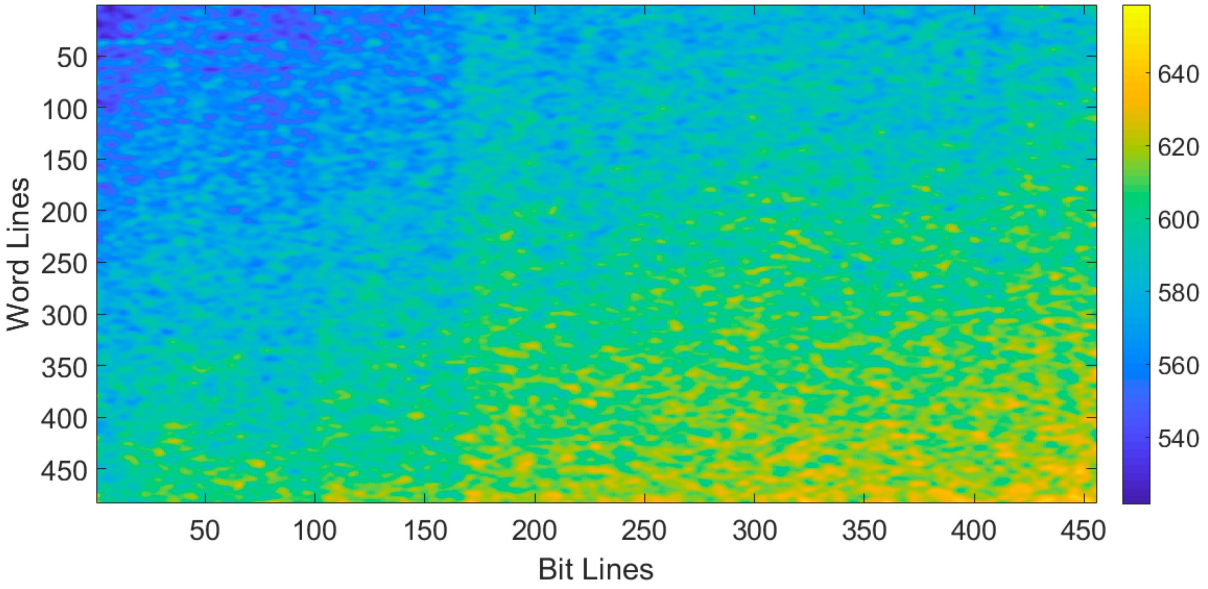


$$Class = \underset{i}{\operatorname{argmax}}\, P_i \cdot Q + \overline{P}_i \cdot \overline{Q} \approx \underset{i}{\operatorname{argmax}}\, P_i \cdot Q$$

Dataset	Hamming Accuracy	Dot Product Accuracy
Language Classification	97.00%	96.20%
News Classification	91.90%	91.00%
EMG Classification	98.50%	97.96%

Mitigating Array-Level Variability

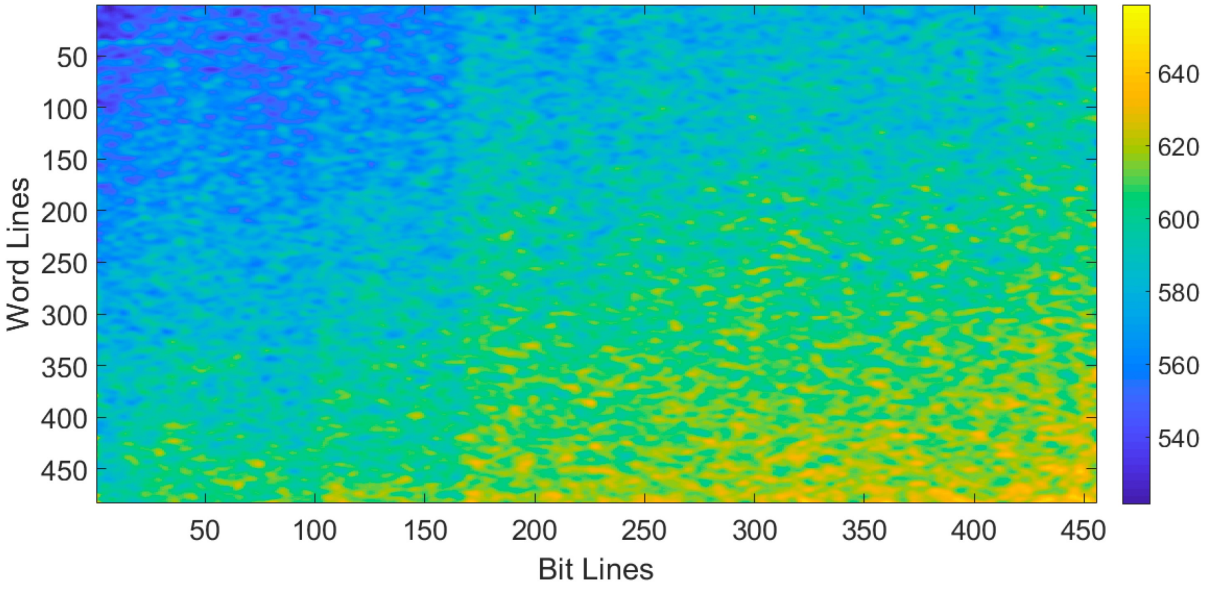
- Coarse-grained randomization of HD vector programming across chip to mitigate array-level variability



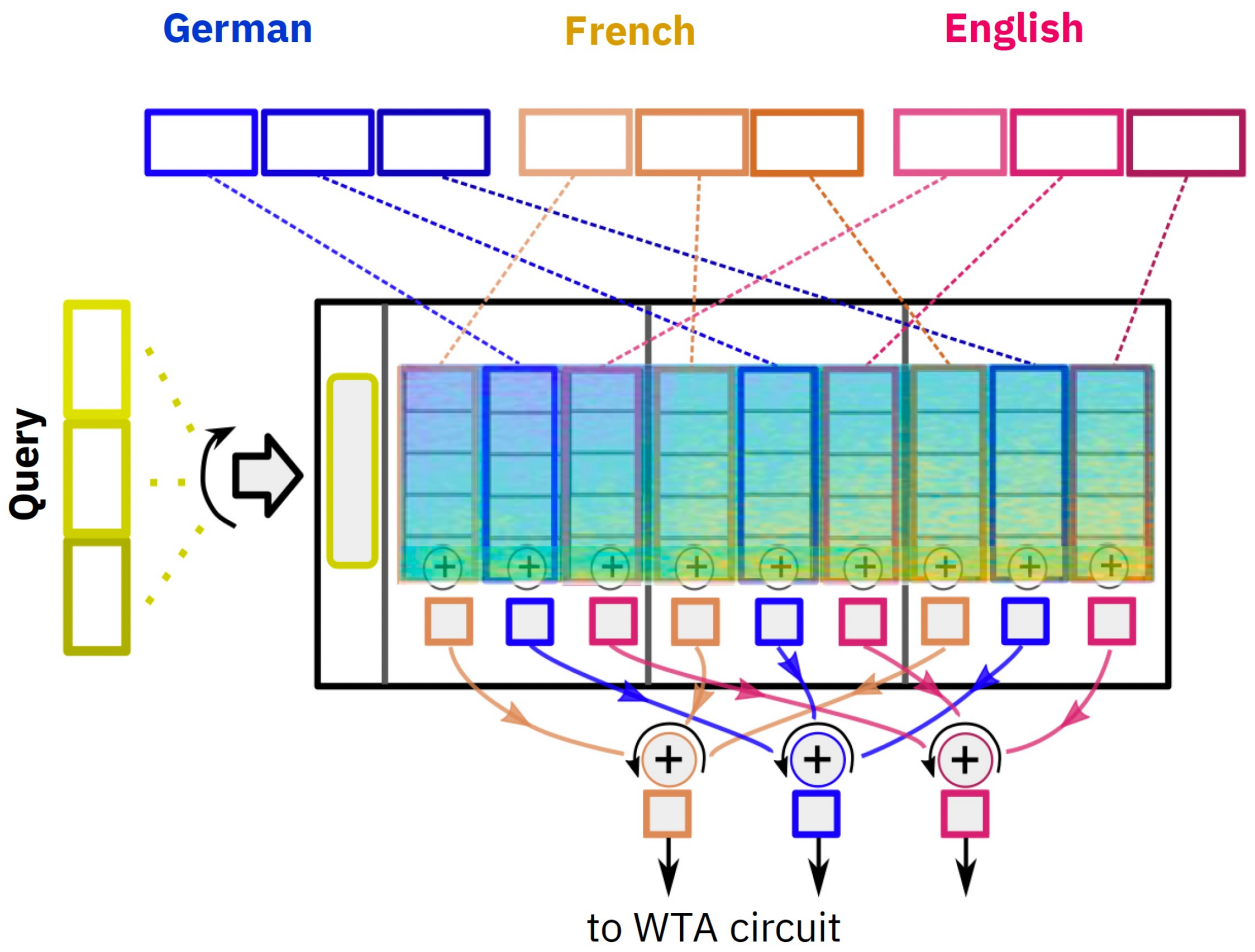
Naïve placement: Accuracy drop 15%

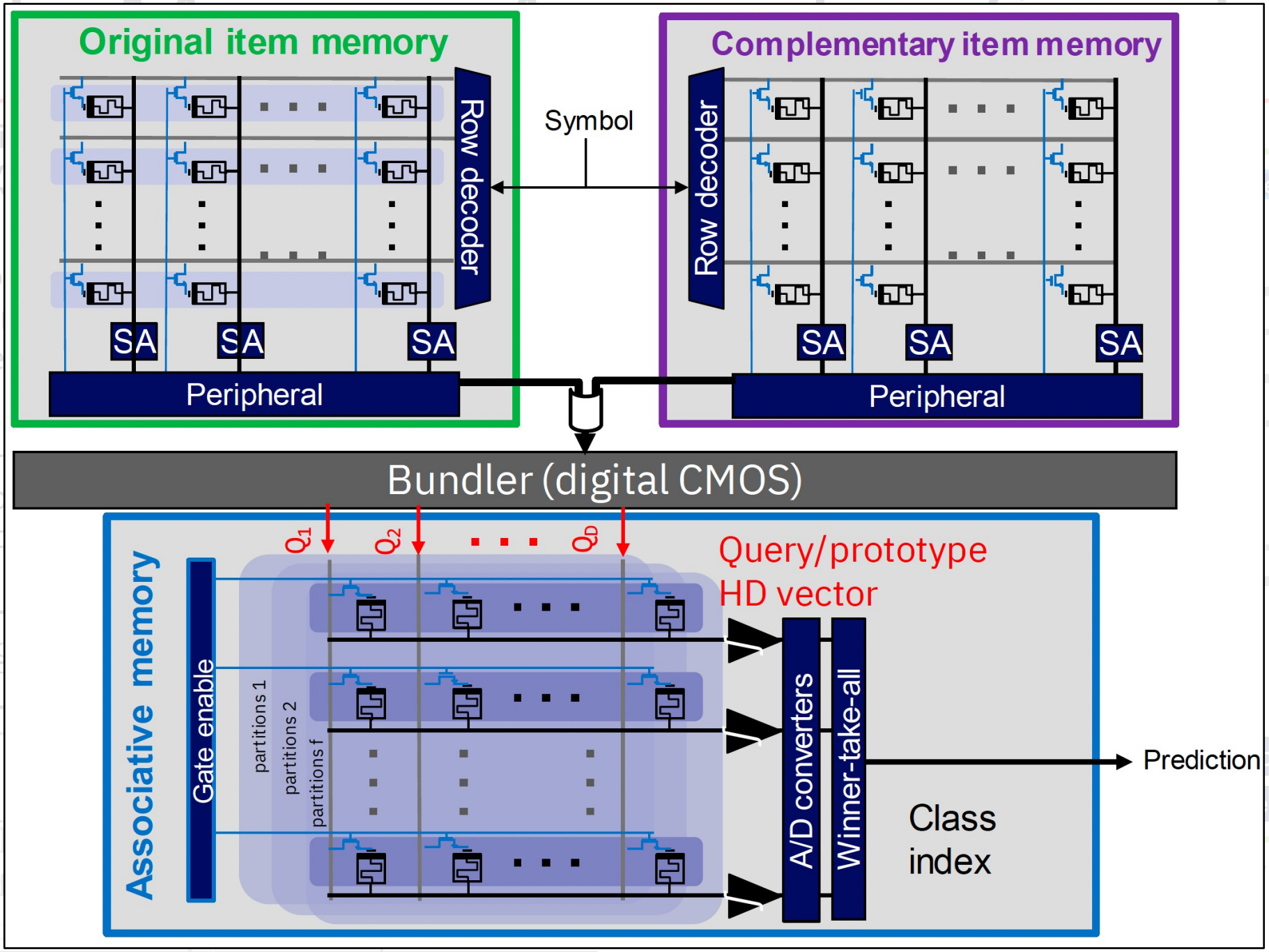
Mitigating Array-Level Variability

- Coarse-grained randomization of HD vector programming across chip to mitigate array-level variability



Naïve placement: Accuracy drop 15%




Original item memory
Row decoder
SA
SA
SA
Peripheral
Symbol
Complementary item memory
Row decoder
SA
SA
SA
Peripheral
Bundler (digital CMOS)
Associative memory
Gate enable
partitions 1
partitions 2
partitions f
Q_1
Q_2
Q_D
Query/prototype
HD vector
A/D converters
Winner-take-all
Prediction
Class
index

Mitigating Array-Level Variability

- **In-memory computing with PCM + 65nm CMOS peripherals leads to 6.61x energy reduction and 3.81x area reduction, compared to full 65nm CMOS design**
- **Room for improvement in CMOS peripheral circuits**
- **Same concepts can be applied to other memristive technologies (RRAM, MRAM, …)**

Bit Lines

to WTA circuit

Naïve placement: Accuracy drop 15%

Takeaways

- HD computing is realized today in CMOS – but true opportunity lays in integrating memory, logic, and sensing
- In-memory HD computing shows a great potential in reducing energy – but there is still a room for improving the digital peripherals
- Hardware design is all about resolving tradeoffs: performance, flexibility, energy efficiency