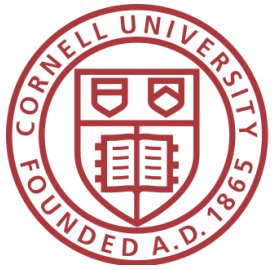
PPAC: A Versatile In-Memory Accelerator for Matrix-Vector-Product-Like Operations

Oscar Castañeda

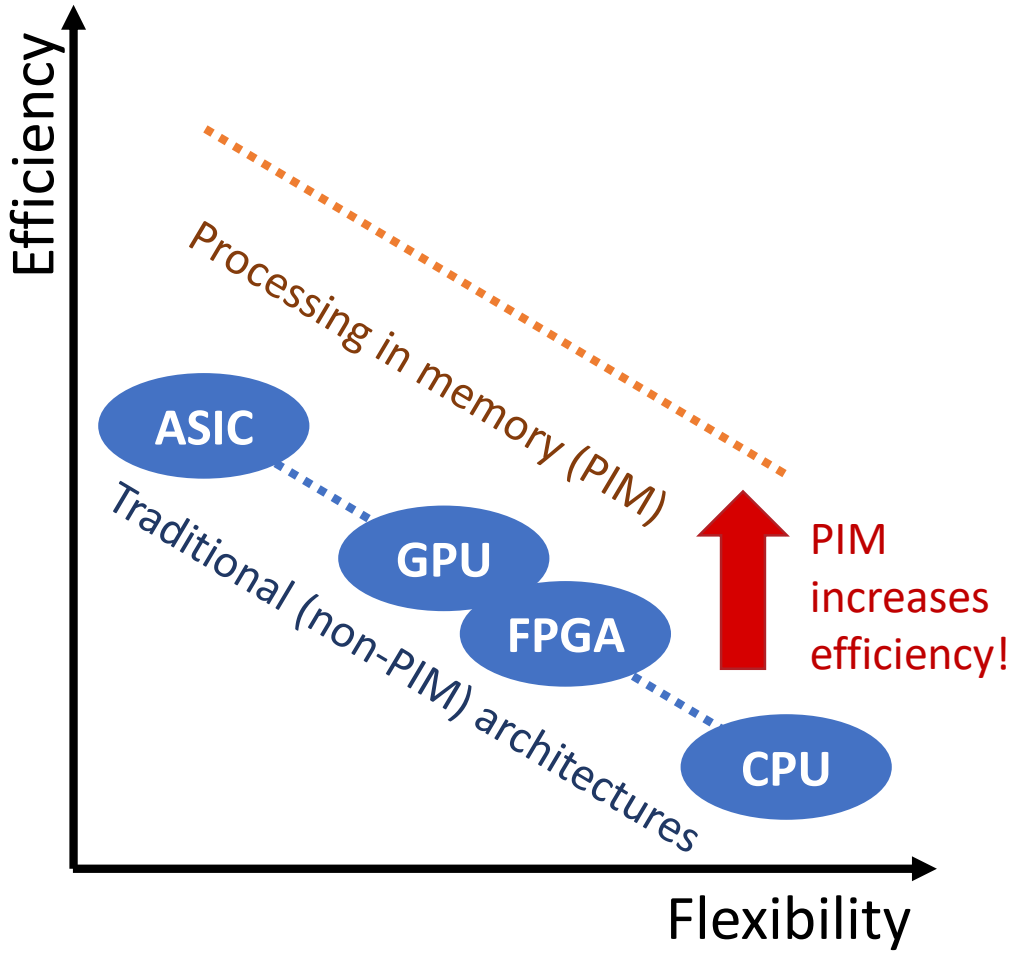
Joint work with Maria Bobbett,
Alexandra Gallyas-Sanhueza, and Christoph Studer

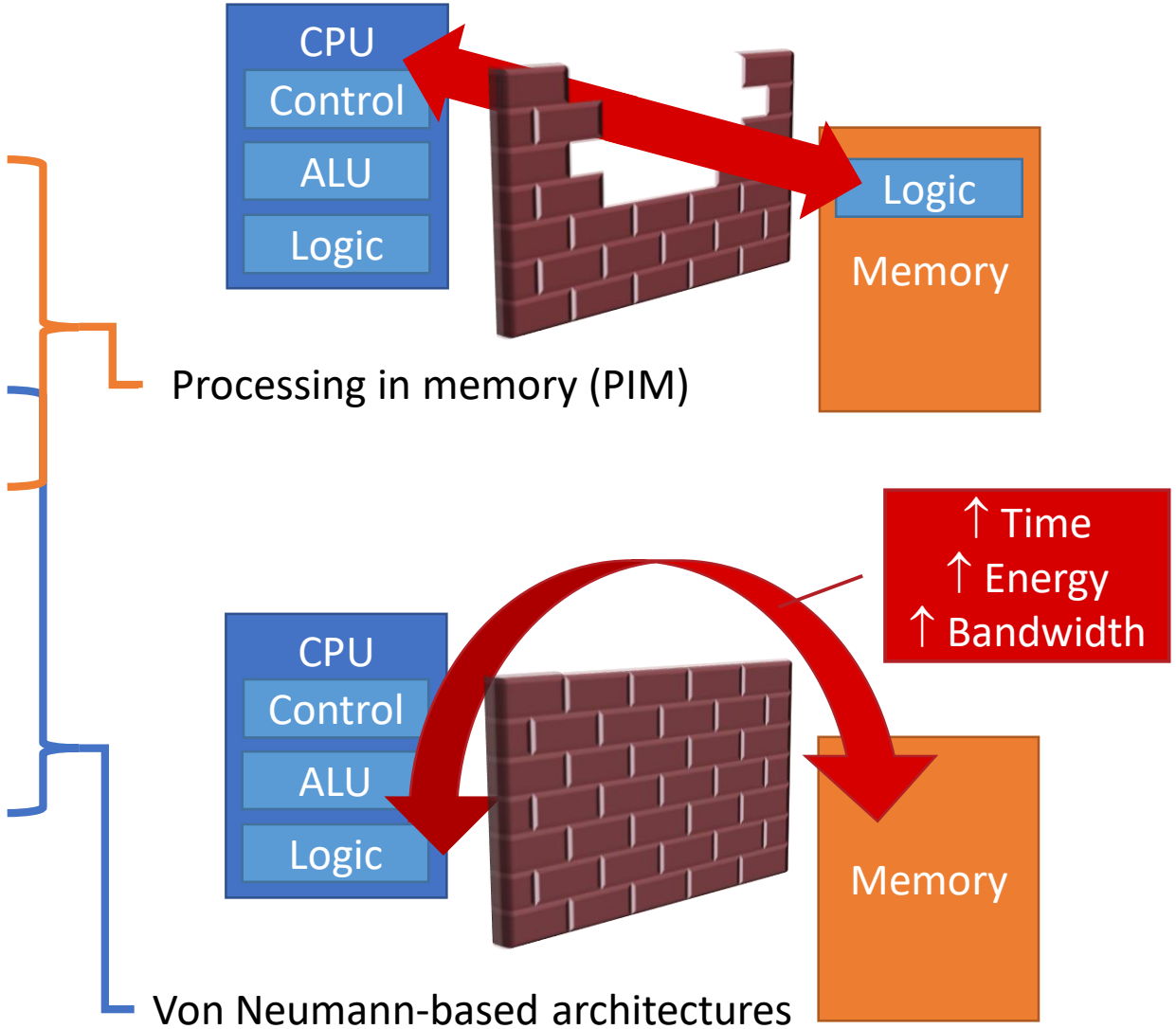


Cornell University

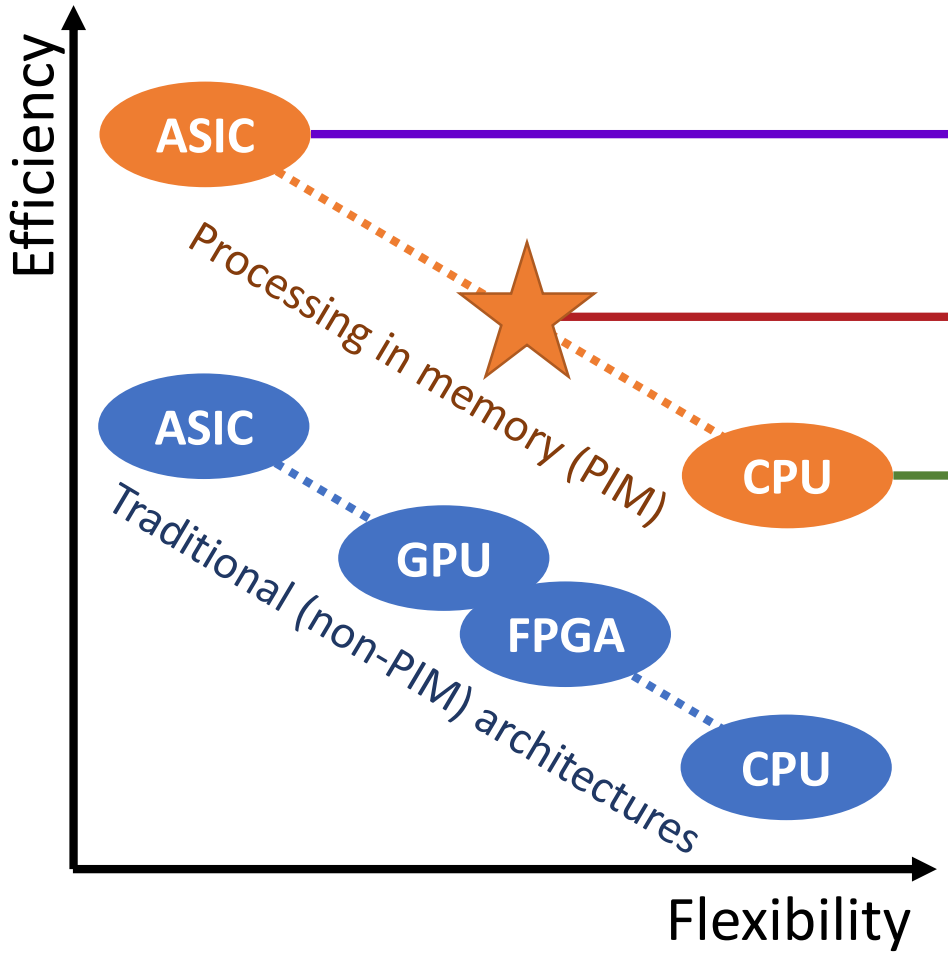
VLSI Information Processing

PIM breaks through the memory wall





Exploring PIM architectures



Application-specific PIM [3]

- ✓ High throughput and efficient architecture tailored for a given application
- × Often only support a single task: Flexibility ↓

This work: PPAC

- An **efficient** yet **flexible** PIM accelerator

General-purpose PIM [1,2]

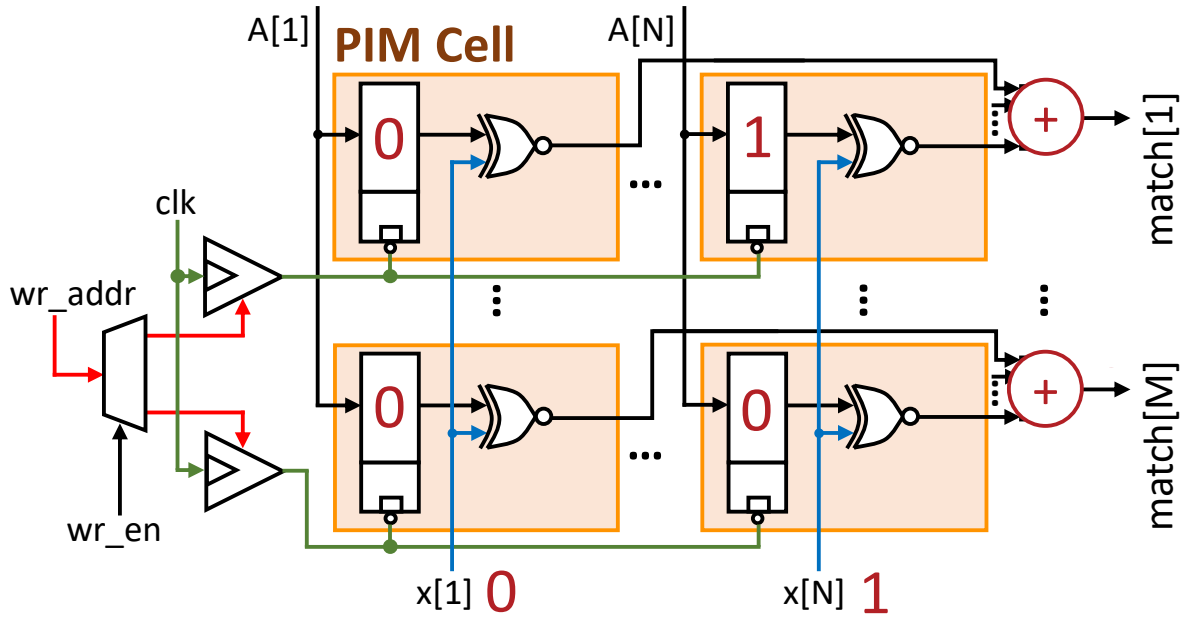
- ✓ Hardware support for atomistic operations (e.g., AND/NOR)
- × More complex tasks repeatedly use atomistic operations: Throughput ↓, Efficiency ↓

[1] S. Aga, S. Jeloka, A. Subramaniya, S. Narayanasamy, D. Blaauw, and R.Das, "Compute caches," in *HPCA*, Feb. 2017, pp. 481-492
[2] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural cache," in *ISCA*, Jun. 2018, pp. 383-396
[3] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing", in *arXiv*, Nov. 2018

PPAC builds upon CAMs

- **P**arallel **P**rocessor in **A**ssociative **C**ontent-addressable memory
- Each CAM row compares its stored word $\mathbf{a}_m$ with the input $\mathbf{x}$
- *Hamming similarity*: Number of matching bits between $\mathbf{a}_m$ and $\mathbf{x}$
  - $\bar{h}(\mathbf{a}_m, \mathbf{x}) = N - h(\mathbf{a}_m, \mathbf{x})$,

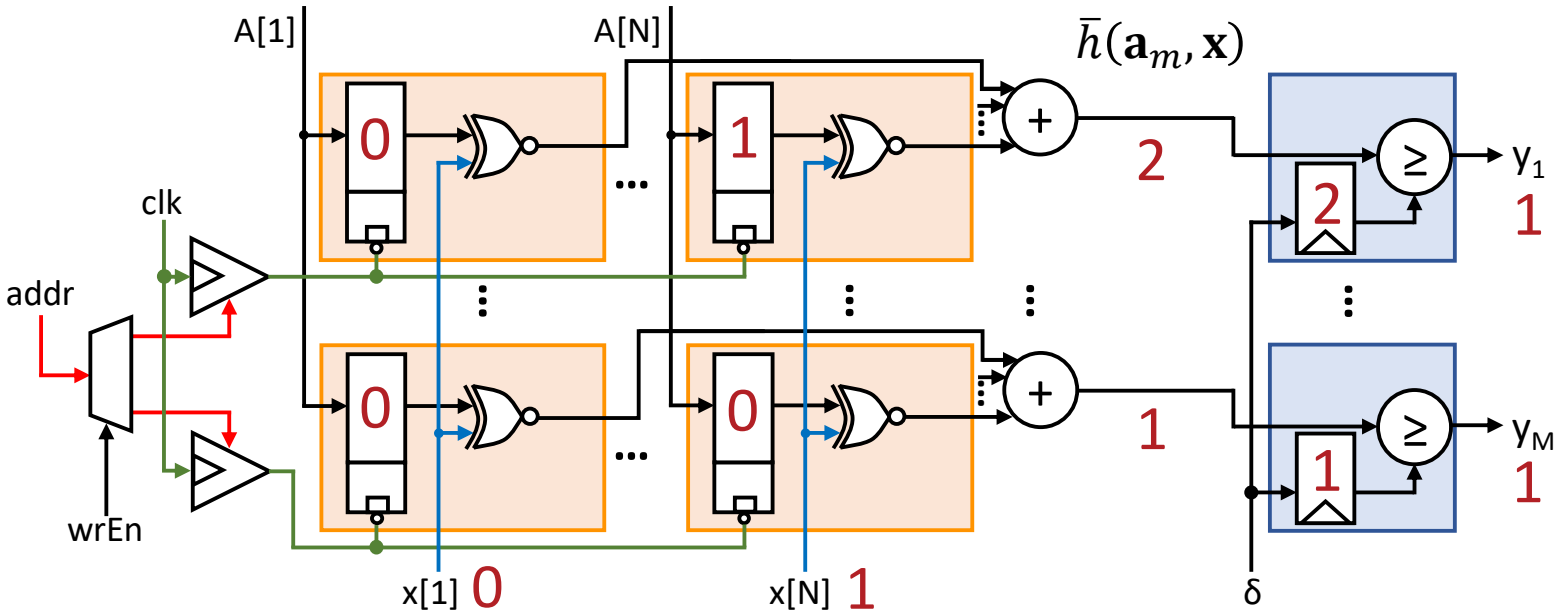
  where h is the Hamming distance and N the number of bits in $\mathbf{a}_m$ and $\mathbf{x}$
- A CAM declares a match if $\bar{h}(\mathbf{a}_m, \mathbf{x}) = N$



Content-Addressable Memory (CAM)

PPAC computes Hamming similarities

- Every row computes and processes $\bar{h}(\mathbf{a}_m, \mathbf{x})$
- Programmable threshold δ:
  Match if $\bar{h}(\mathbf{a}_m, \mathbf{x}) \geq \delta_m$
  - Complete match: $\delta_m = N$ (standard CAM)
  - Similarity match: $\delta_m < N$



Hamming similarity

- Single-cycle operation

✓ CAM Applications [1]
- Network switches and routers
- Computer caches

✓ Content Addressable Parallel Processor (CAPP) [2]
- Based on similarity matches

✓ Particle track reconstruction [3]

✓ Approx. nearest neighbor search
- Locality sensitive hashing
- Indoor localization via fingerprinting

[1] K. Pagiamtzis and A. Sheikholeslami, "Content-Addressable Memory (CAM) circuits and architectures: A tutorial and survey," *IEEE JSSC*, vol. 41, no.3, pp.712-727, Mar. 2006
[2] C. Foster, *Content Addressable Parallel Processors*. John Wiley and Sons, Inc., 1976
[3] A. Annovi, G. Calderini, S. Capra, et al., "Characterization of an associative memory chip in 28nm CMOS Technology," in *Proc. IEEE ISCAS*, May 2018.

Exploiting Hamming similarities

Binary-valued vector element $\mathbf{x}_n$

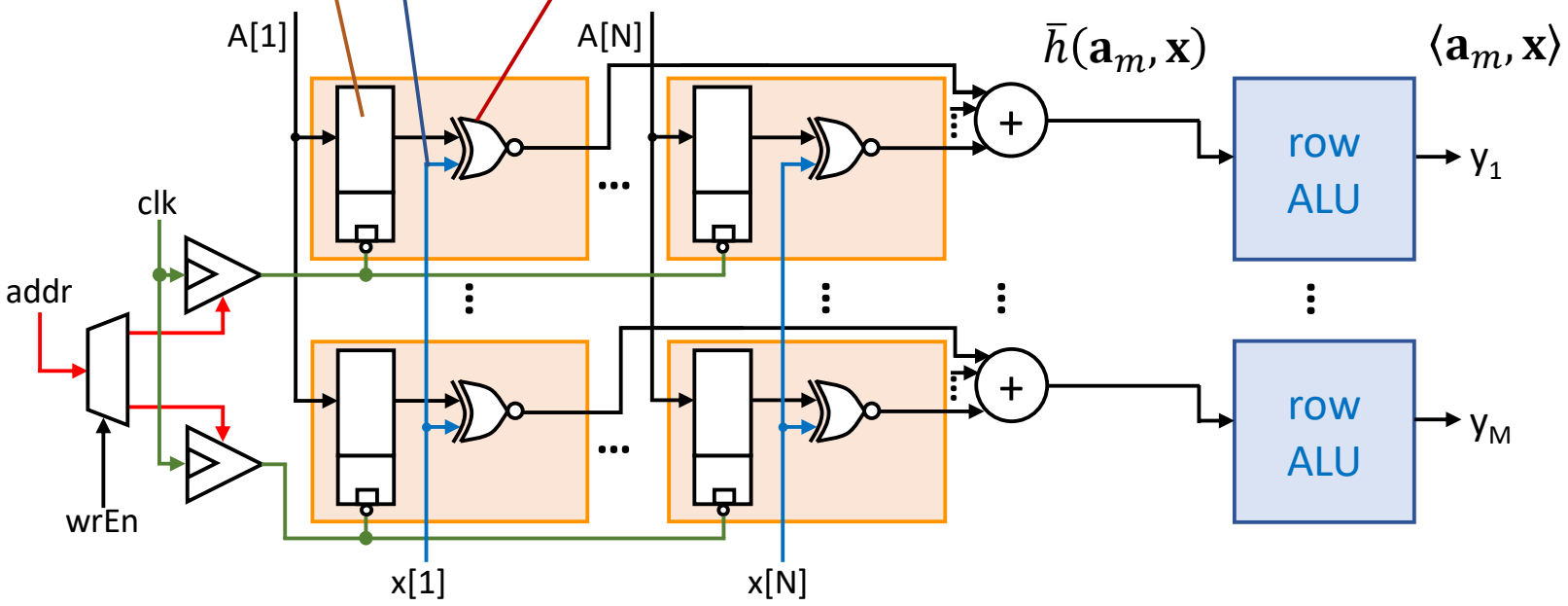
Stores binary-valued matrix entries $\mathbf{a}_{m,n}$

a	x	a XNOR x
HI	HI	HI
HI	LO	LO
LO	HI	LO
LO	LO	HI

HI→+1

LO→-1

a	x	a • x
+1	+1	+1
+1	-1	-1
-1	+1	-1
-1	-1	+1



$$\langle \mathbf{a}_m, \mathbf{x} \rangle = \sum_{n=1}^{N} \mathbf{a}_{m,n} \mathbf{x}_n$$

$$\langle \mathbf{a}_m, \mathbf{x} \rangle = -N + 2\bar{h}(\mathbf{a}_m, \mathbf{x})$$

$$\begin{bmatrix} \langle \mathbf{a}_1, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{a}_M, \mathbf{x} \rangle \end{bmatrix} = \mathbf{A} \cdot \mathbf{x}$$

1-bit matrix-vector product (MVP)

- Single-cycle operation

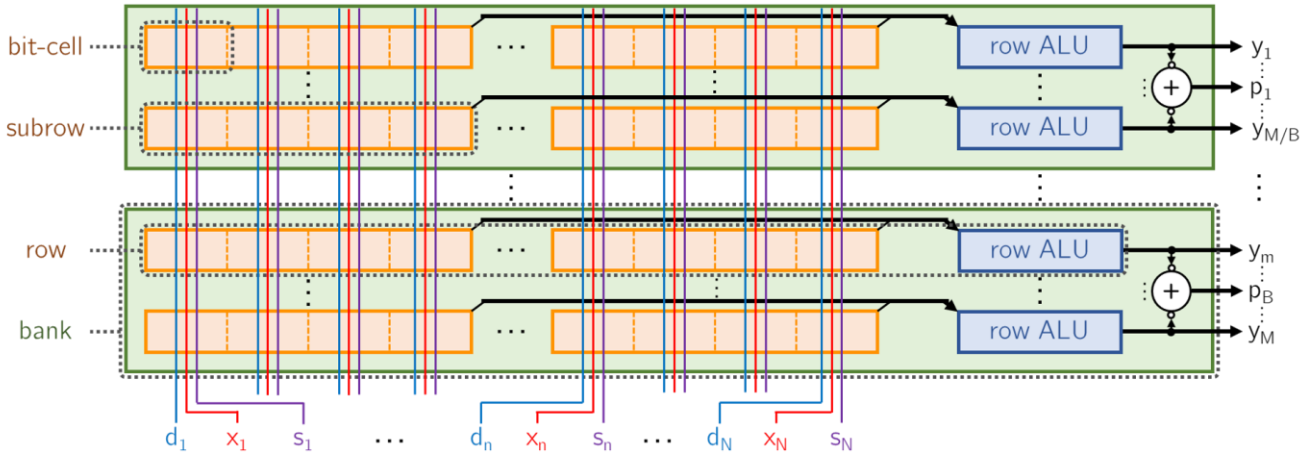
✓ Binarized neural networks [1]

✓ mmWave/THz equalization

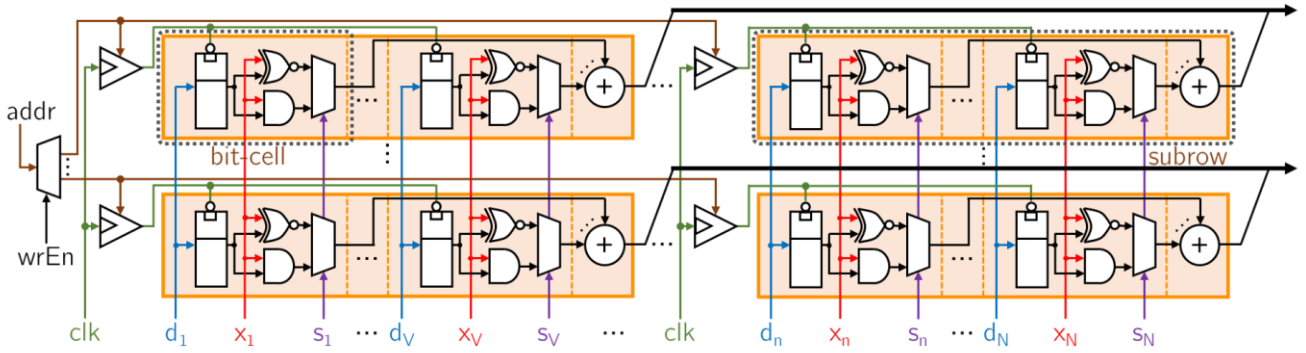
[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in *Proc. NeurIPS*, Dec. 2016, pp. 4114-4122.

The PPAC architecture

- 2D array of latch-based bit-cells
- Each of the M rows stores an N-bit word and has a row ALU
- Multiple rows are grouped in a bank



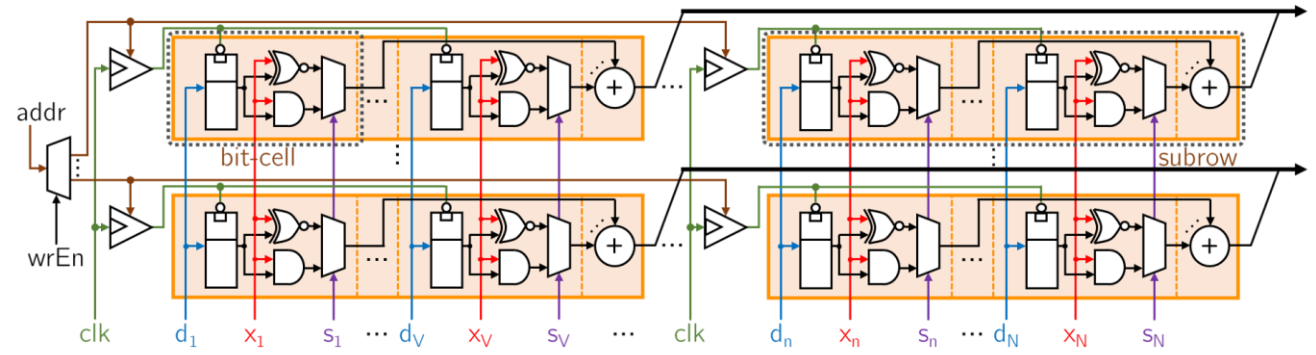
- Bit-cell with two operators
  - XNOR/AND
- All bit-cells on the same column share input and control signals
- A row is divided into subrows with local pop counts for scalability

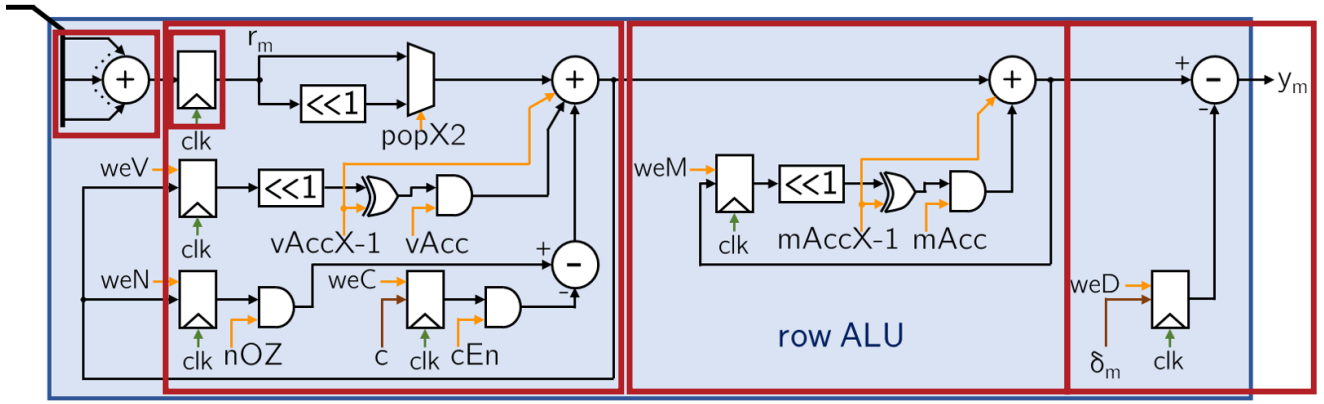


A population count (pop count for short) corresponds to counting how many ones are there in a set of 1-bit numbers.

The PPAC row ALU

- Adds local pop counts to compute the row population count r_m
- Two accumulators:
  - First one for multi-bit vectors; includes offsets to correctly interpret r_m
  - Second for multi-bit matrixes
- A programmable threshold δ_m is subtracted from the second's accumulator output to obtain y_m
  - Can be used as a comparator for declaring exact/similarity matches
- Pipeline stage after row population count to increase throughput





Operating MVPs in different number formats

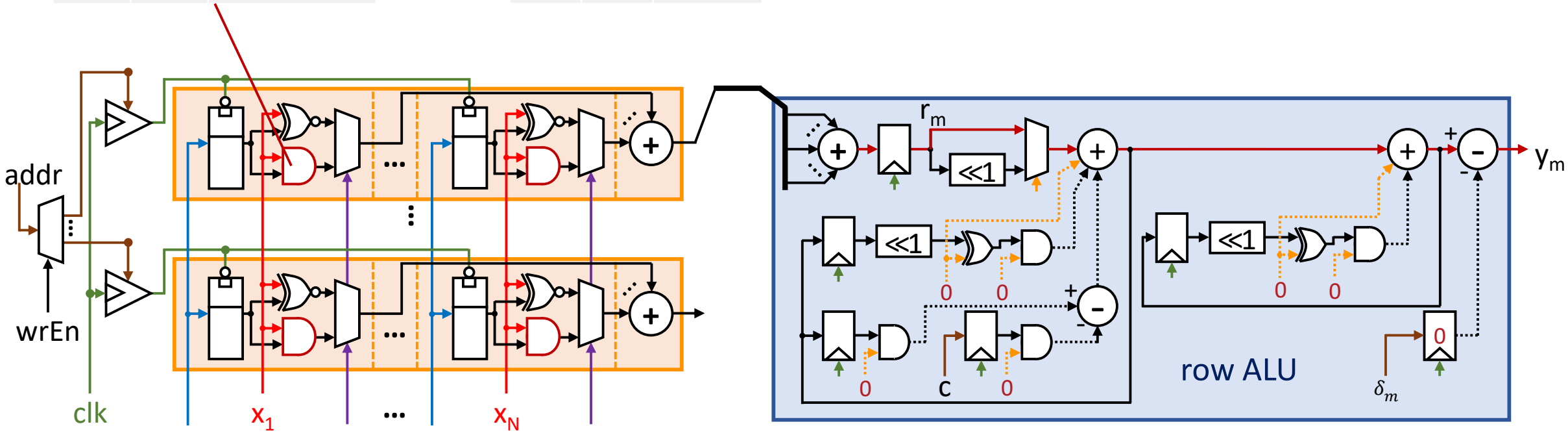
a	x	a AND x
HI	HI	HI
HI	LO	LO
LO	HI	LO
LO	LO	LO

HI→1

LO→0

a	x	a • x
1	1	1
1	0	0
0	1	0
0	0	0

- Key for building standard (multi-bit) unsigned and 2's complement signed arithmetic, and other operations!

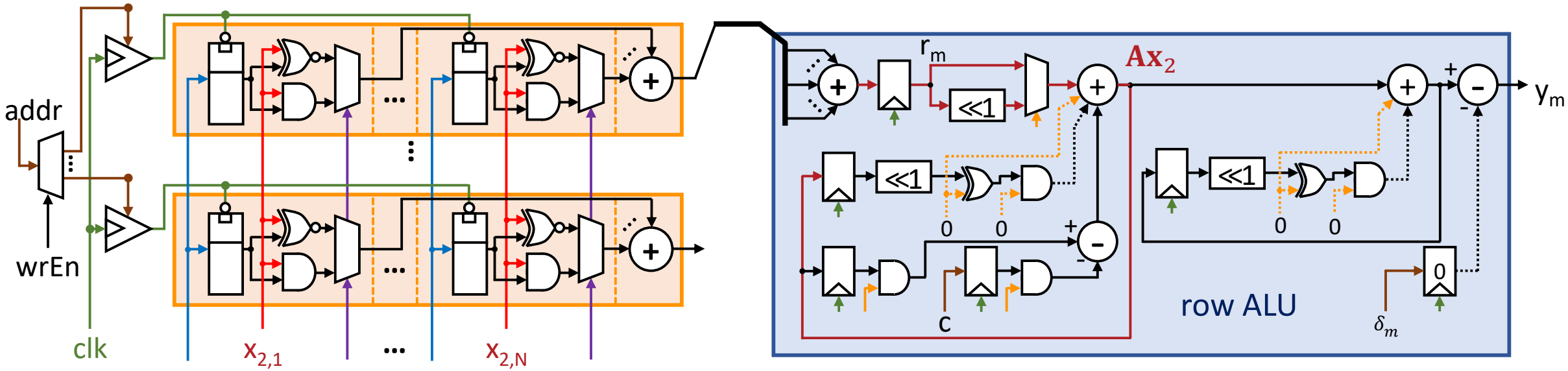


Executing an MVP with a multi-bit vector

- Assume $\mathbf{A}$ has 1-bit entries, while $\mathbf{x}$ has L-bit entries
- Bit-serial execution in L clock cycles
  - Operate MSBs of $\mathbf{x}$ first ($\mathbf{x}_L$), LSBs last ($\mathbf{x}_1$)
  - Result of $\mathbf{A}\mathbf{x}_L$ can be negated for 2's complement arithmetic

$$\mathbf{x} = \sum_{\ell=1}^{L} 2^{\ell-1}\mathbf{x}_\ell$$

$$\mathbf{A}\mathbf{x} = \sum_{\ell=1}^{L} 2^{\ell-1}\mathbf{A}\mathbf{x}_\ell$$

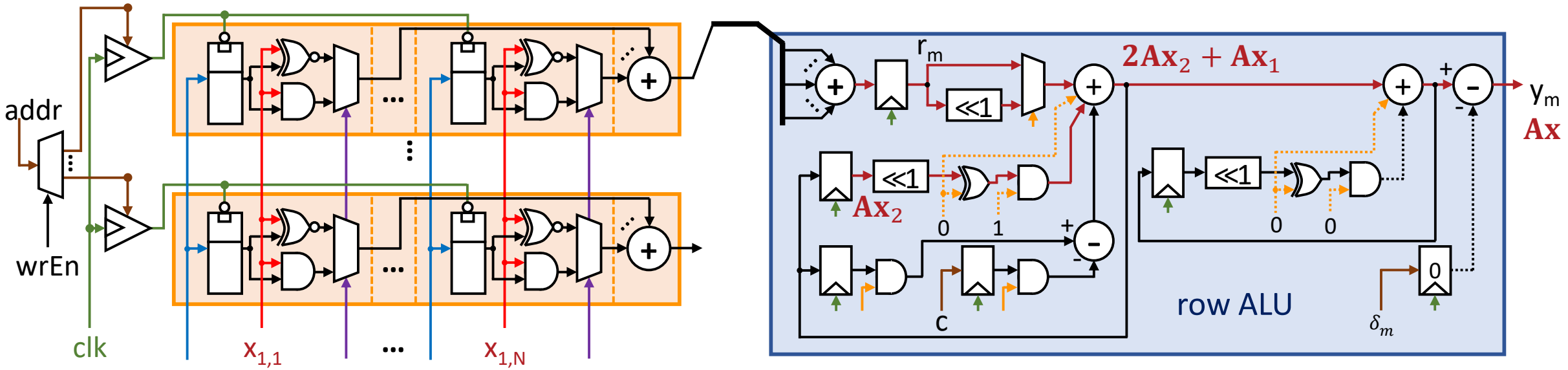


Executing an MVP with a multi-bit vector

- Assume $\mathbf{A}$ has 1-bit entries, while $\mathbf{x}$ has L-bit entries
- Bit-serial execution in L clock cycles
  - Operate MSBs of $\mathbf{x}$ first ($\mathbf{x}_L$), LSBs last ($\mathbf{x}_1$)
  - Result of $\mathbf{A}\mathbf{x}_L$ can be negated for 2's complement arithmetic

$$\mathbf{x} = \sum_{\ell=1}^{L} 2^{\ell-1}\mathbf{x}_\ell$$

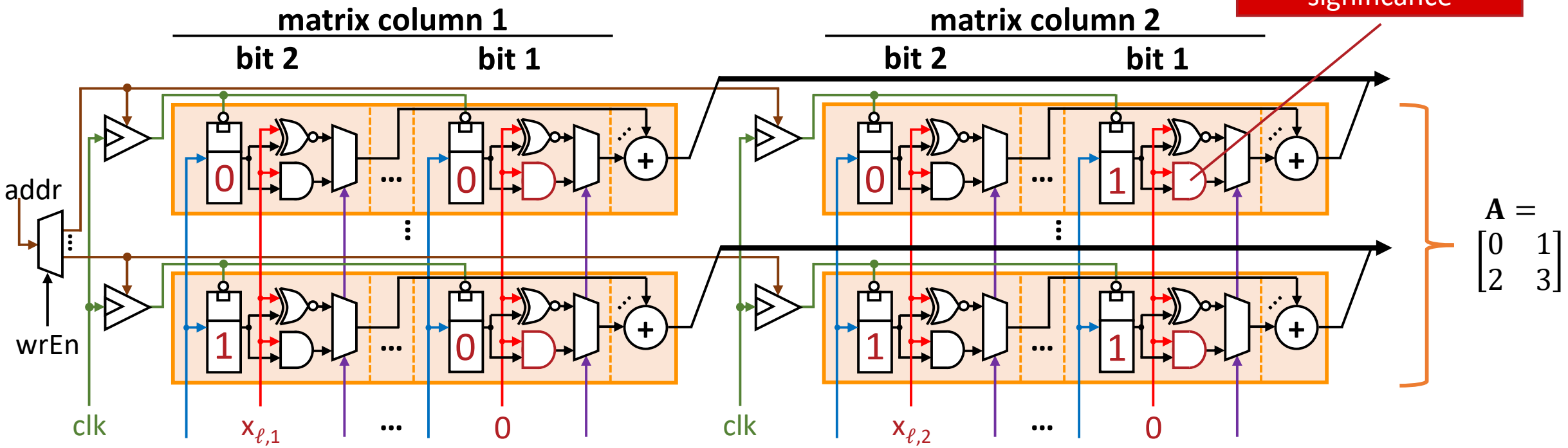
$$\mathbf{A}\mathbf{x} = \sum_{\ell=1}^{L} 2^{\ell-1}\mathbf{A}\mathbf{x}_\ell$$



Executing an MVP with a multi-bit matrix

- Assume $\mathbf{A}$ has K-bit entries and $\mathbf{x}$ has L-bit entries
- Bit-serial execution in KL clock cycles
  - Same idea as multi-bit vectors, but memory cannot be changed fast enough [1]
  - Operate MSBs of $\mathbf{A}$ first ($\mathbf{A}_K\mathbf{x}$), LSBs last ($\mathbf{A}_1\mathbf{x}$)
  - Result of $\mathbf{A}_k\mathbf{x}$ can be negated for 2's complement arithmetic

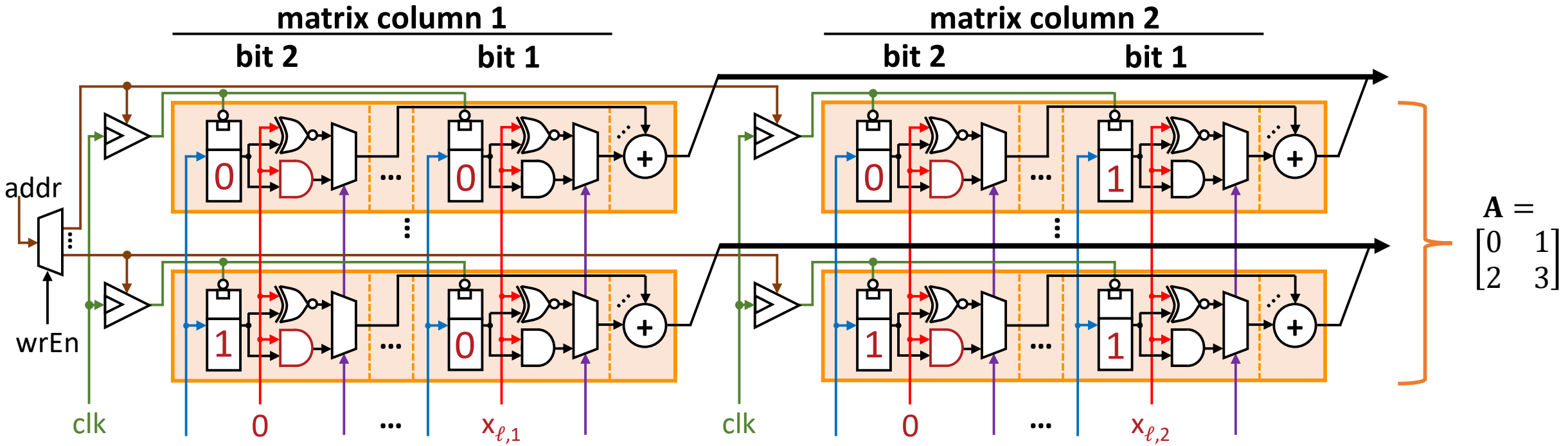
Nulls contribution of bits with different significance



[1] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," arXiv preprint, Nov. 2018.

Executing an MVP with a multi-bit matrix

- Assume $\mathbf{A}$ has K-bit entries and $\mathbf{x}$ has L-bit entries
- Bit-serial execution in KL clock cycles
  - Same idea as multi-bit vectors, but memory cannot be changed fast enough [1]
  - Operate MSBs of $\mathbf{A}$ first ($\mathbf{A}_K\mathbf{x}$), LSBs last ($\mathbf{A}_1\mathbf{x}$)
  - Result of $\mathbf{A}_k\mathbf{x}$ can be negated for 2's complement arithmetic



[1] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," arXiv preprint, Nov. 2018.

PPAC does MVPs in different number formats

L-bit number formats supported by PPAC

Name	uint	int	oddint
LO level	0	0	-1
HI level	1	1	1
Signed?	No	Yes	Yes
Min. value	0	-2^{L-1}	$-2^{L}+1$
Max. value	$2^{L}-1$	$2^{L-1}-1$	$2^{L}-1$
E.g., $L=2$	{0,1,2,3}	{-2,-1,0,1}	{-3,-1,1,3}

$$x = \sum_{\ell=1}^{L} 2^{\ell-1} x_{\ell}$$

Multi-bit matrix-vector product (MVP)

- For a matrix with K-bit entries and a vector with L-bit entries, an MVP is computed in KL cycles
- ✓ Neural network inference using low-precision `int`/`uint` numbers
- ✓ Hadamard transform using a 1-bit `oddint` matrix and a multi-bit `int` vector
  - Signal processing, imaging and communications applications

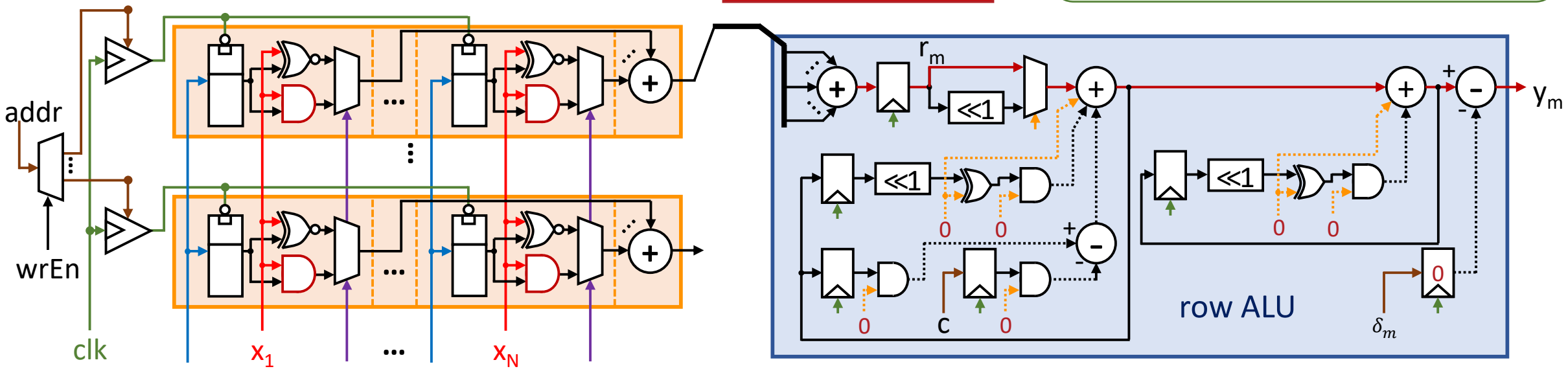
MVPs in GF(2)

- Galois Field of Two Elements, GF(2):
  - × : AND
  - + : Mod-2 +
- Mod-2 + is the LSB of y_m

Cannot implement reliably in analog

GF(2) arithmetic

- Single-cycle operation
- ✓ Forward-error correction
- ✓ Decoding
  - LDPC codes
  - Polar codes
- ✓ Cryptography
  - Secure hashing
  - AES S-box computation



Computing Boolean functions with PPAC

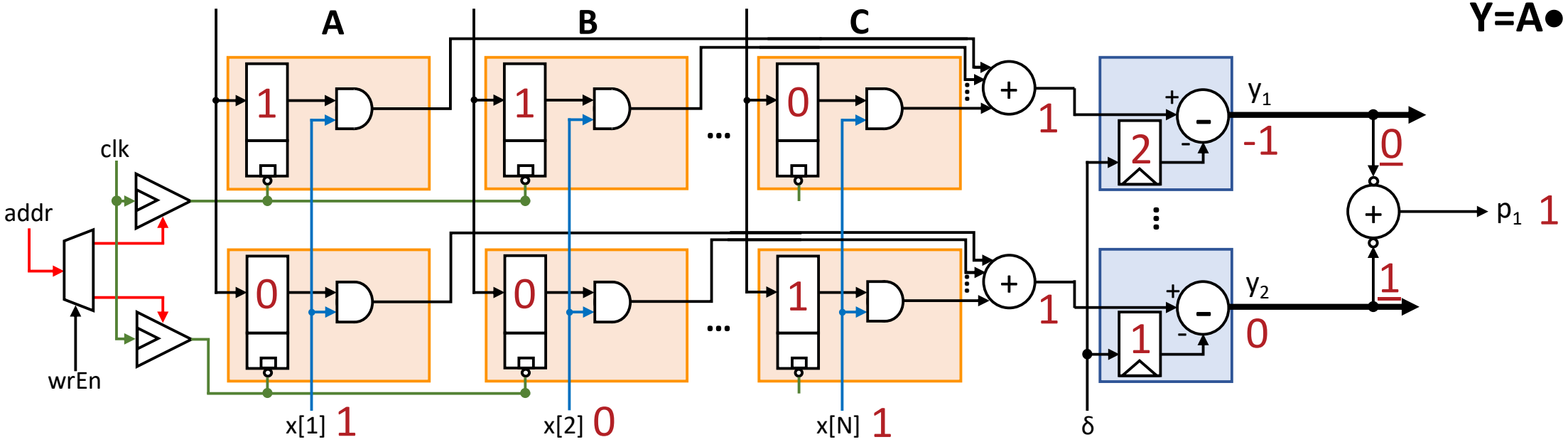
- Sum of min-terms
  - Column → Boolean variable (complement in different column)
  - Row → Min-term
  - Bank → Logic function
- $\mathbf{A}_{m,n} = 1$ if mth minterm contains nth variable
- PPAC can compute Boolean functions with two AND/OR/MAJ levels

Logic function computation

- Single-cycle operation

✓ Look-up table

✓ Programmable Logic Array (PLA)

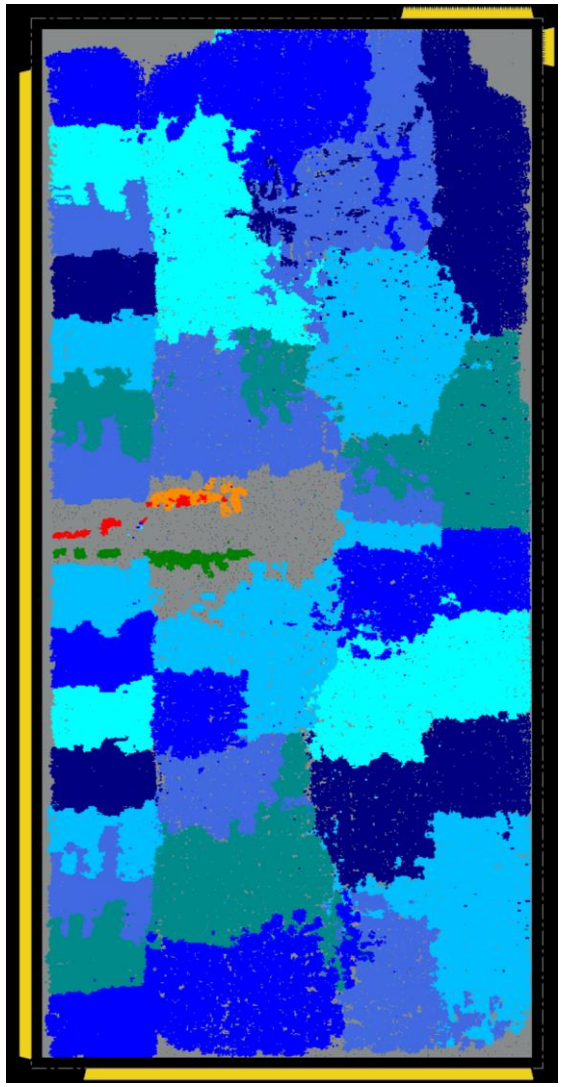


Implementing PPAC

Post-layout 28nm CMOS implementation results; $L, K \leq 4$

Words M	16	16	256	256
Word-length N	16	256	16	256
Area [µm^2]	14 161	72 590	185 283	783 240
Density [%]	75.77	70.45	75.52	72.13
Cell area [kGE]	17	81	213	897
Max. clock freq. [GHz]	1.116	0.979	0.824	0.703
Power (@0.9V,25°C) [mW]	6.64	45.60	78.65	381.43
Peak throughput [TOP/s]	0.55	8.01	6.54	91.99
Energy-eff. [fJ/OP]	12.00	5.69	12.03	4.15

- All-digital CMOS design
  - RTL design: Parameterizable and portable
  - Automated CAD tool implementation
  - Robust to process variations and easy to test



256x256 PPAC Layout

One OP is either a 1-bit multiplication or a 1-bit addition

Comparison with existing accelerators

BNN Accelerator Design	PIM?	Mixed signal?	Impl.	Tech. [nm]	Supply [V]	Area [mm^2]	Peak TP [GOP/s]	Energy-eff. [TOP/s/W]	Norm. Peak TP [GOP/s]	Norm. Energy-eff. [TOP/s/W]
PPAC	yes	no	layout	28	0.9	0.78	91 994	184	91 994	184
CIMA [1]	yes	yes	silicon	65	1.2	8.56	4 720	152	10 957	1 456
Bankman *et al*. [2]	no	yes	silicon	28	0.8	5.95	-	532	-	420
Brein [3]	yes	no	silicon	65	1.0	3.9	1.38	2.3	3.2	15
UNPU [4]	no	no	silicon	65	1.1	16	7 372	46.7	17 114	376
XNE [5]	no	no	layout	22	0.8	0.016	108	112	84.7	54.6

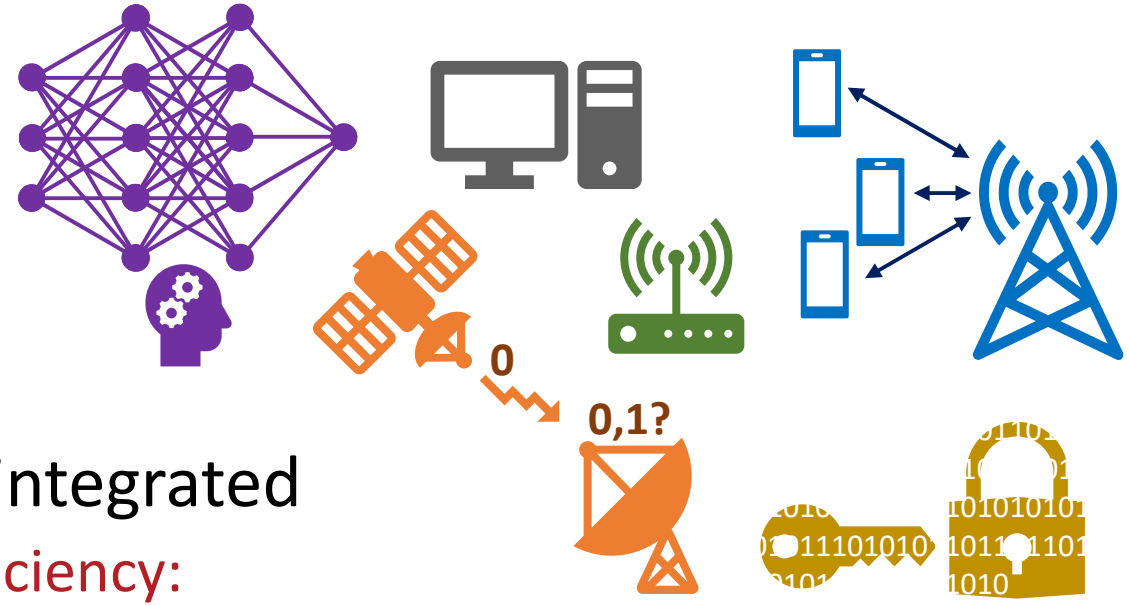
- Energy-efficiency 7.9x and 2.3x lower than mixed-signal designs in [1] and [2]
- PPAC can compute a 4-bit 256-dim. inner product in **16 clock cycles**; Neural Cache [6] requires at least **98 clock cycles**

But PPAC is flexible and synthesizable!

[1] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," arXiv preprint, Nov. 2018.
[2] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," in IEEE ISSCC, Feb. 2018, pp. 222-224.
[3] K. Ando, K. Ueyoshi, K. Orimo, et al., "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE JSSC*, vol. 53, pp. 983-994, Apr. 2018.
[4] J. Lee, C. Kin, S. Kang, D. Shin, S. Kim, and H. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE JSSC,* vol. 54, no. 1, pp. 173-185, Jan. 2019.
[5] F. Conti, P. Schiavone, and L. Benini, "XNOR neural engine: A hardware accelerator IP for 21.6-fJ/op binary neural network inference," *IEEE Trans. Comp.-Aided Design Integrated Circ. Syst,* vol. 37, Nov. 2018.
[6] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," *ACM/IEEE ISCA*, Jun. 2018, pp. 189-200.

PPAC: A versatile in-memory accelerator

- Massively-parallel engine for matrix-vector-product-like operations
  - Hamming similarity
  - Matrix-vector product
    - 1-bit or multi-bit
    - {LO,HI} →{-1,+1} or {LO,HI} →{0,+1}
    - GF(2)
  - Boolean function computation



- Memory and data-path seamlessly integrated
  - High throughput, and high energy-efficiency:

    A 28nm CMOS 256x256 PPAC achieves 92 TOP/s @ 184 TOP/s/W
- Competitive performance with high flexibility

For more information, please visit vip.ece.cornell.edu

Backup Slides

Exploiting Hamming similarities

Binary-valued vector element $\mathbf{x}_n$

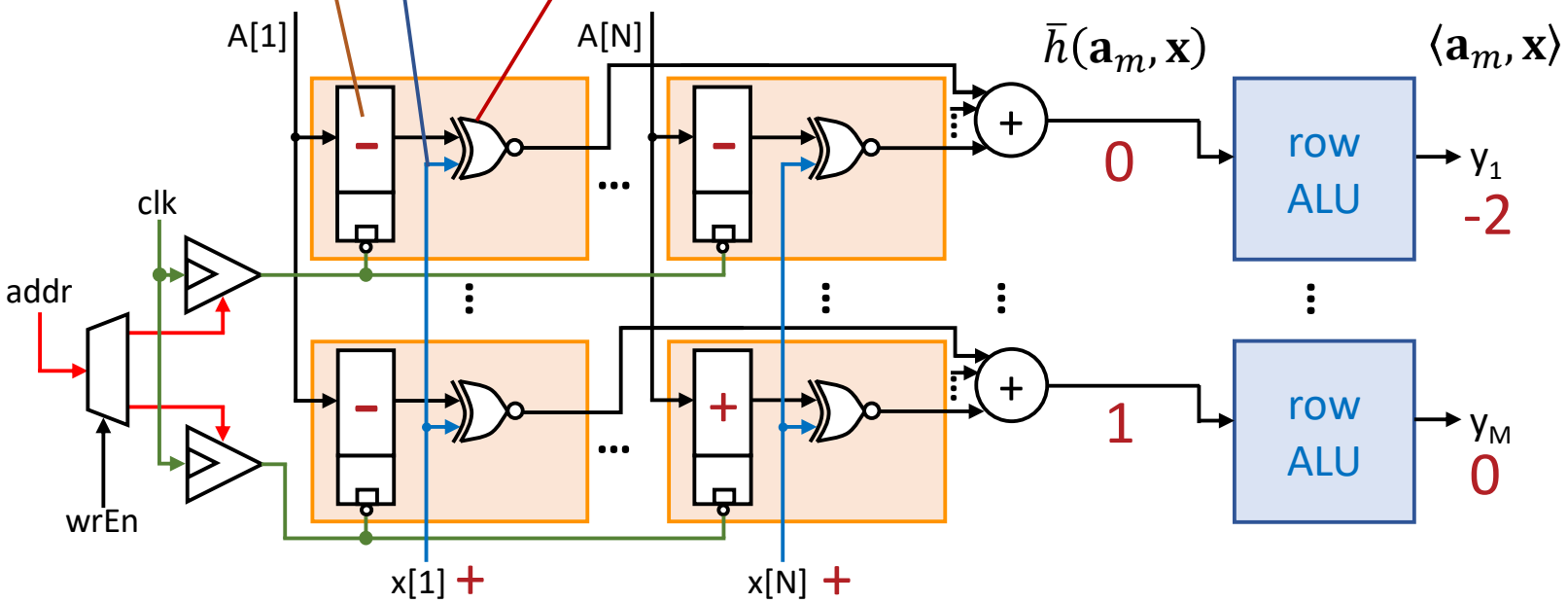
Stores binary-valued matrix entries $\mathbf{a}_{m,n}$

a	x	a XNOR x
HI	HI	HI
HI	LO	LO
LO	HI	LO
LO	LO	HI

HI→+1

LO→-1

a	x	a • x
+1	+1	+1
+1	-1	-1
-1	+1	-1
-1	-1	+1



$$\langle \mathbf{a}_m, \mathbf{x} \rangle = \sum_{n=1}^{N} \mathbf{a}_{m,n} \mathbf{x}_n$$

$$\langle \mathbf{a}_m, \mathbf{x} \rangle = -N + 2\bar{h}(\mathbf{a}_m, \mathbf{x})$$

$$\begin{bmatrix} \langle \mathbf{a}_1, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{a}_M, \mathbf{x} \rangle \end{bmatrix} = \mathbf{A} \cdot \mathbf{x}$$

1-bit matrix-vector product (MVP)

- Single-cycle operation

✓ Binarized neural networks [1]

✓ mmWave/THz equalization

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in *Proc. NeurIPS*, Dec. 2016, pp. 4114-4122.

Operating MVPs in different number formats

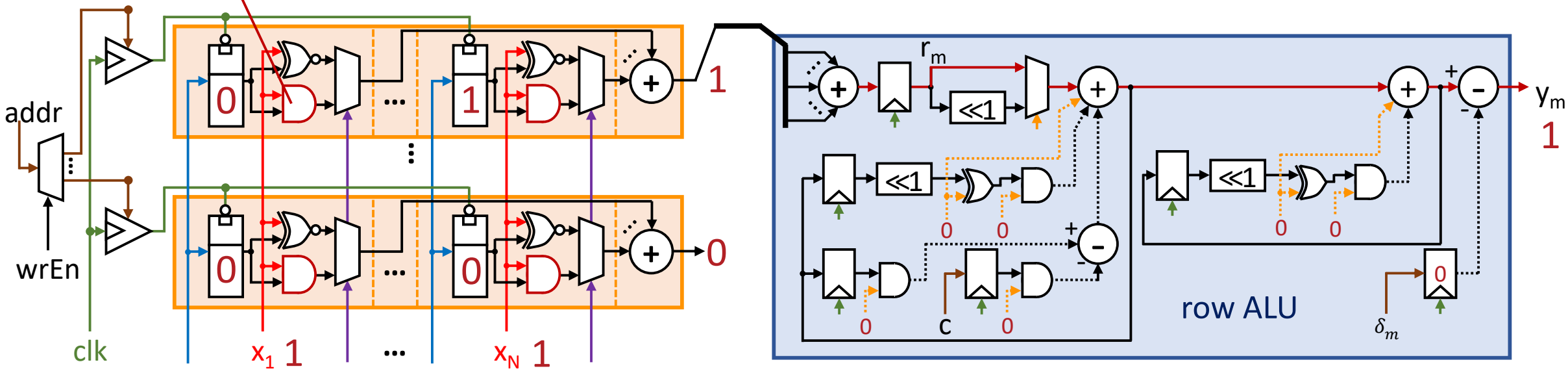
a	x	a AND x
HI	HI	HI
HI	LO	LO
LO	HI	LO
LO	LO	LO

HI→1

LO→0

a	x	a • x
1	1	1
1	0	0
0	1	0
0	0	0

- Key for building standard (multi-bit) unsigned and 2's complement signed arithmetic, and other operations!

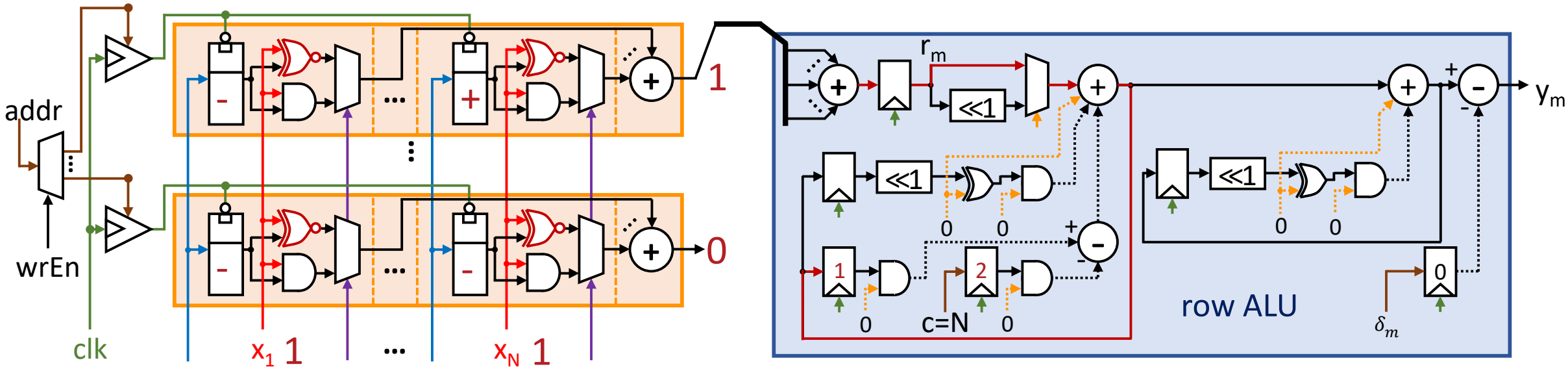


Operating across different number formats

- So far, we have operated matrix-vector-products where both matrix and vector have $\{\pm 1\}$ or $\{0,1\}$ entries
- Operations between different number representations are possible too
  - For example, if the matrix has $\{\pm 1\}$ entries, but the vector has $\{0,1\}$ entries, then:

$$\langle \mathbf{a}_m, \mathbf{x} \rangle = \bar{h}(\mathbf{a}_m, \mathbf{x}) + \bar{h}(\mathbf{a}_m, 1) - N$$

  - $\bar{h}(\mathbf{a}_m, 1)$ is stored in the row ALU and only calculated if the matrix $\mathbf{A}$ changes

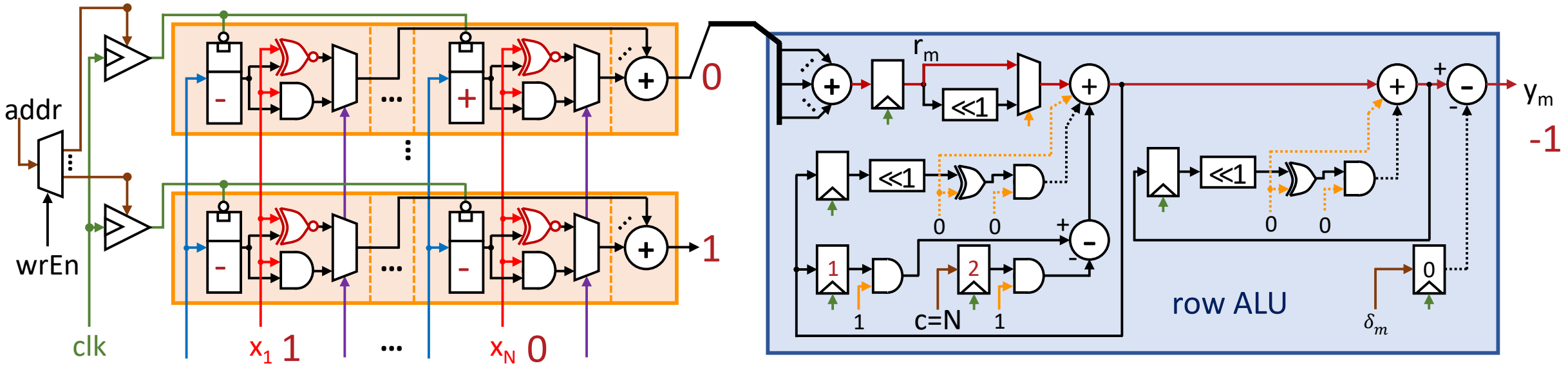


Operating across different number formats

- So far, we have operated matrix-vector-products where both matrix and vector have $\{\pm 1\}$ or $\{0,1\}$ entries
- Operations between different number representations are possible too
  - For example, if the matrix has $\{\pm 1\}$ entries, but the vector has $\{0,1\}$ entries, then:

$$\langle \mathbf{a}_m, \mathbf{x} \rangle = \bar{h}(\mathbf{a}_m, \mathbf{x}) + \bar{h}(\mathbf{a}_m, 1) - N$$

  - $\bar{h}(\mathbf{a}_m, 1)$ is stored in the row ALU and only calculated if the matrix $\mathbf{A}$ changes



Computing Boolean functions with PPAC

- Sum of min-terms
  - Column → Boolean variable (complement in different column)
  - Row → Min-term
  - Bank → Logic function
- $\mathbf{A}_{m,n} = 1$ if mth minterm contains nth variable
- PPAC can compute Boolean functions with two AND/OR/MAJ levels

Logic function computation

- Single-cycle operation

✓ Look-up table

✓ Programmable Logic Array (PLA)