Sept 26, 2014

Cryptographic Hardware and Embedded Systems

Bitline PUF:

Building Native Challenge-Response PUF Capability into Any SRAM

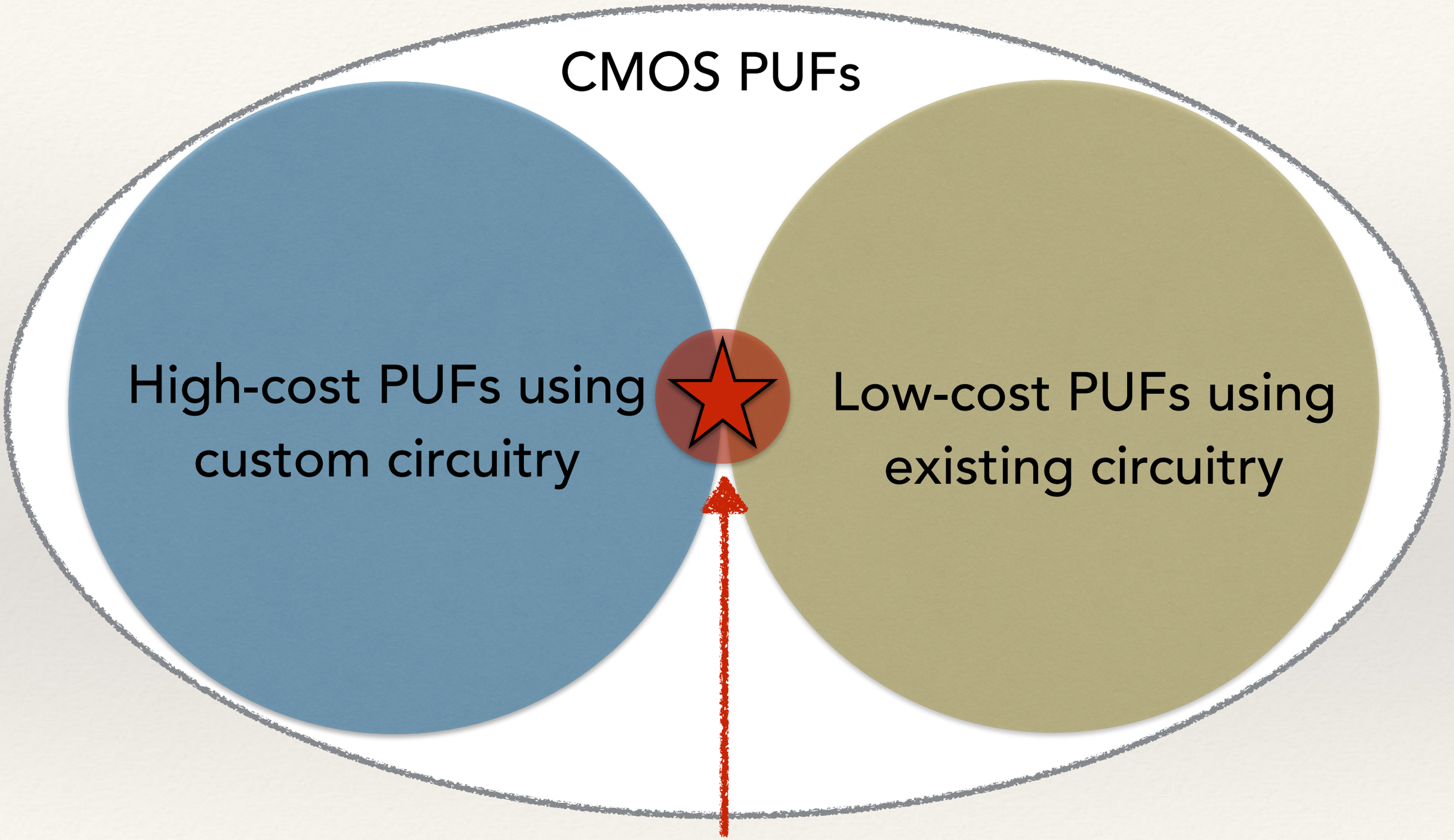
Daniel E. Holcomb

Kevin Fu

University of Michigan

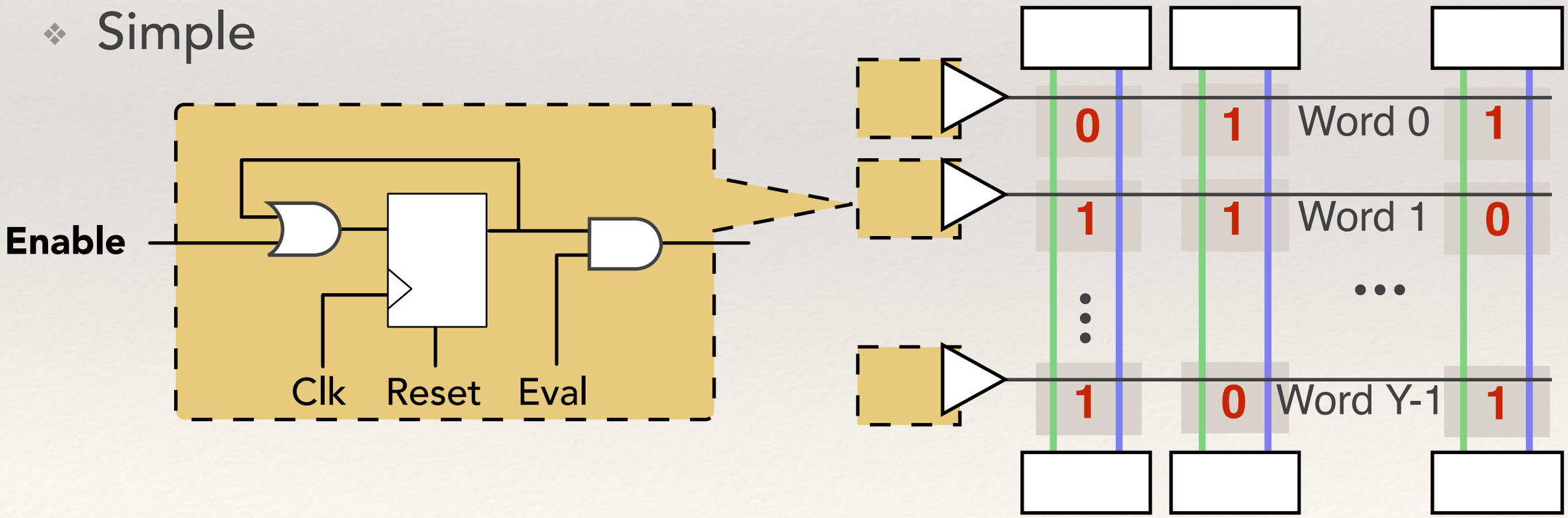
Acknowledgment: This work was supported in part by C-FAR, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and by NSF CNS-1331652. Any opinions, findings, conclusions, and recommendations expressed in these materials are those of the authors and do not necessarily reflect the views of the sponsors.

Context


CMOS PUFs
High-cost PUFs using custom circuitry
Low-cost PUFs using existing circuitry
This talk

Contributions

- **Adding a few gates to wordline drivers of SRAM creates a new PUF**
- Bitline PUF
  - Challenge-response operation
  - Low area overhead
  - Simple



Outline

1. **Introduction**
   - **PUFs**
   - **SRAM**
   - **Bitline PUF**
2. Evaluation
   - Uniqueness
   - Reliability
   - Modeling Attacks
3. Summary and Related work

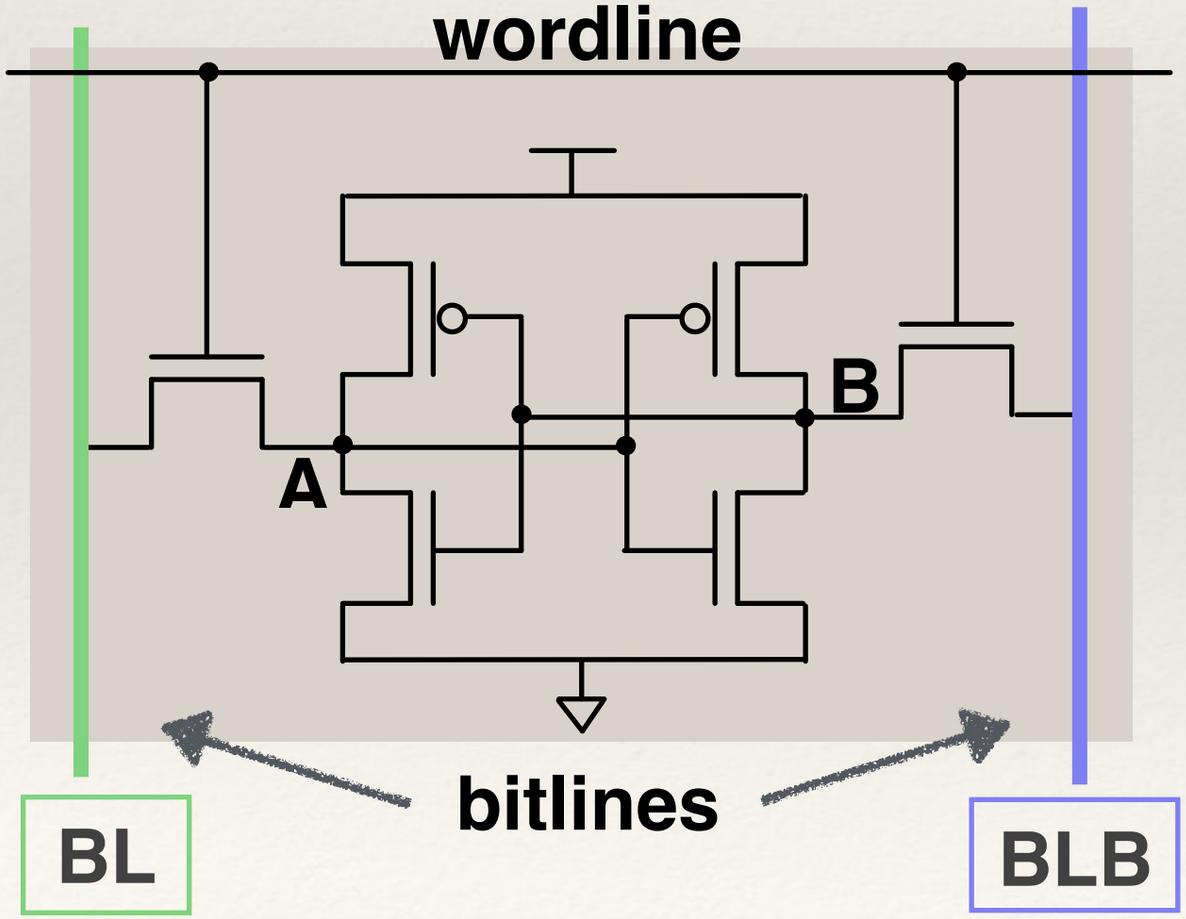
Physical Unclonable Functions (PUFs)

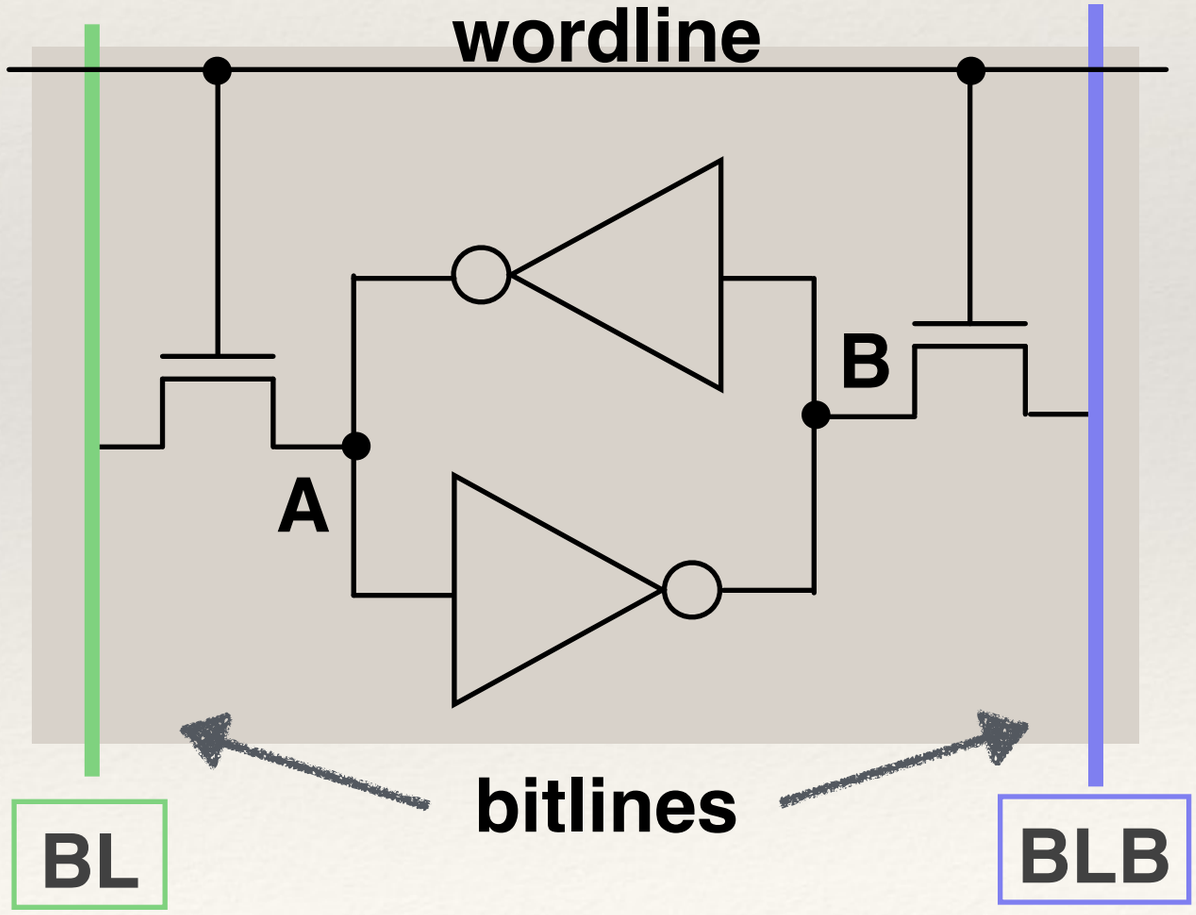
- Map challenges to responses according to uncontrollable physical variations
- Unique to each chip and persistent
  - Random dopant fluctuations and small devices
  - Balanced parasitics and wire lengths to avoid bias
- Applications include anti-counterfeiting and hardware metering

PUF Characterized by Challenge-Response Pairs (CRPs)

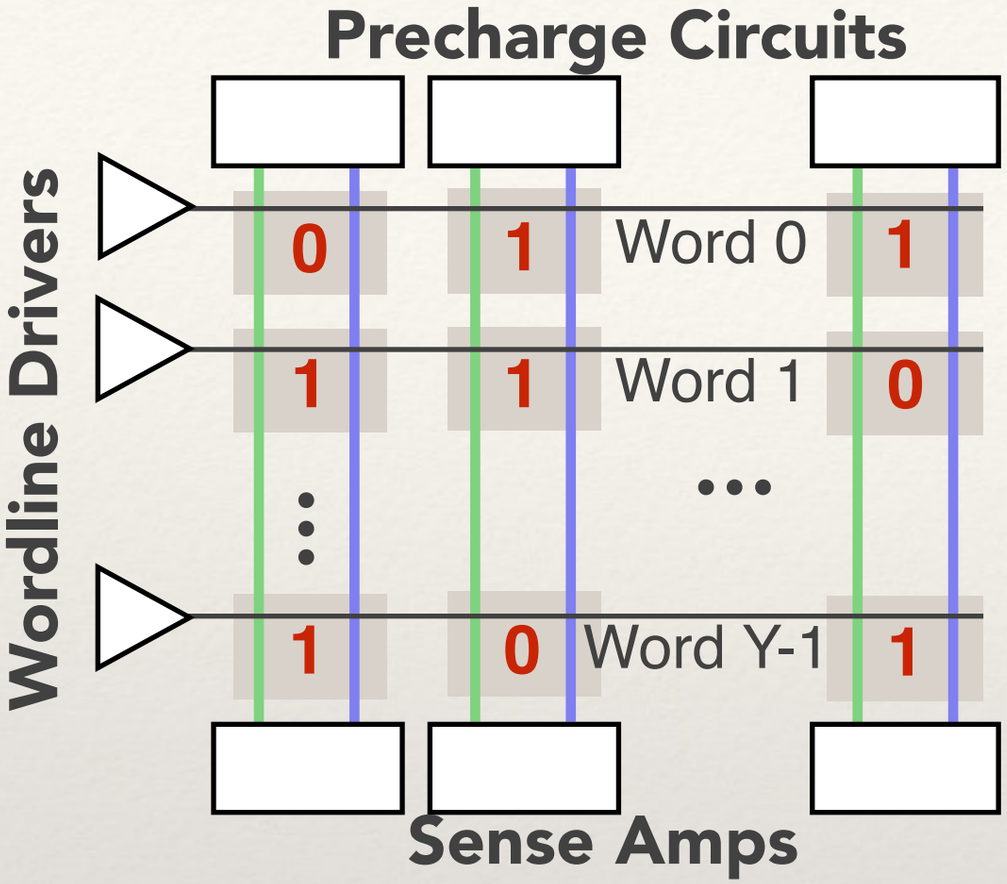
6-Transistor SRAM Cell

- Ubiquitous memory
- Two stable states: “0” (AB=01) “1” (AB=10)
- **Wordline** selects a cell for reading/writing
- Complementary **bitlines** read/write values to/from selected cells

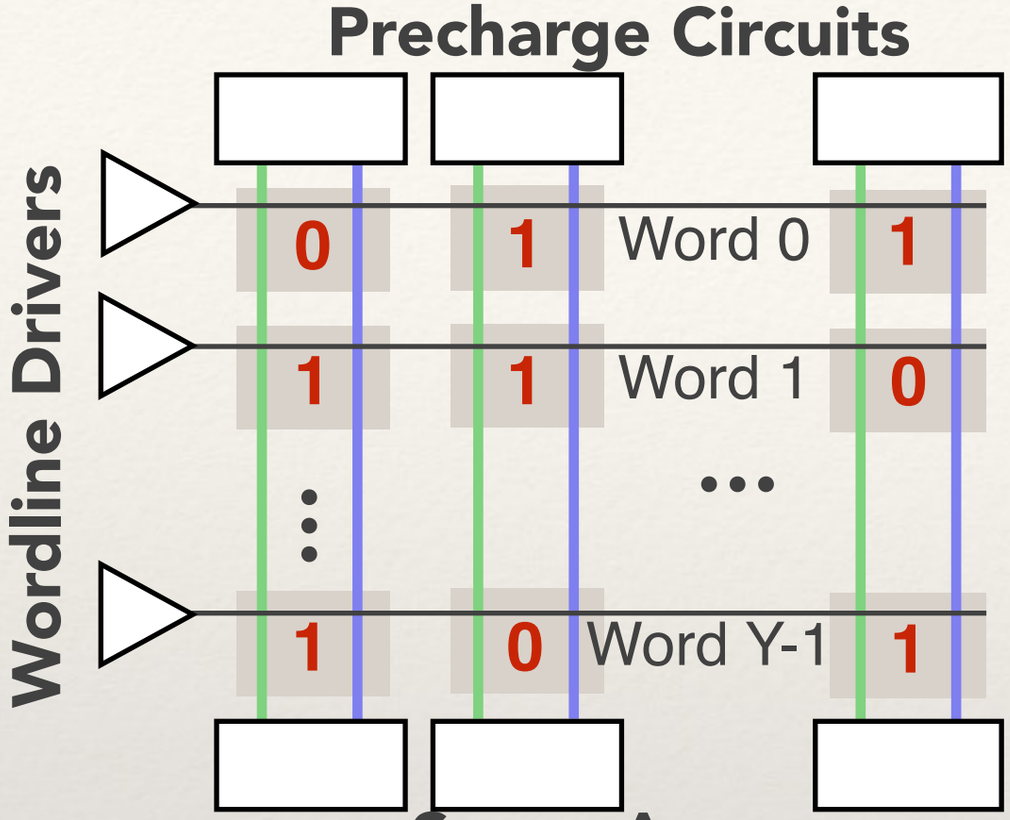


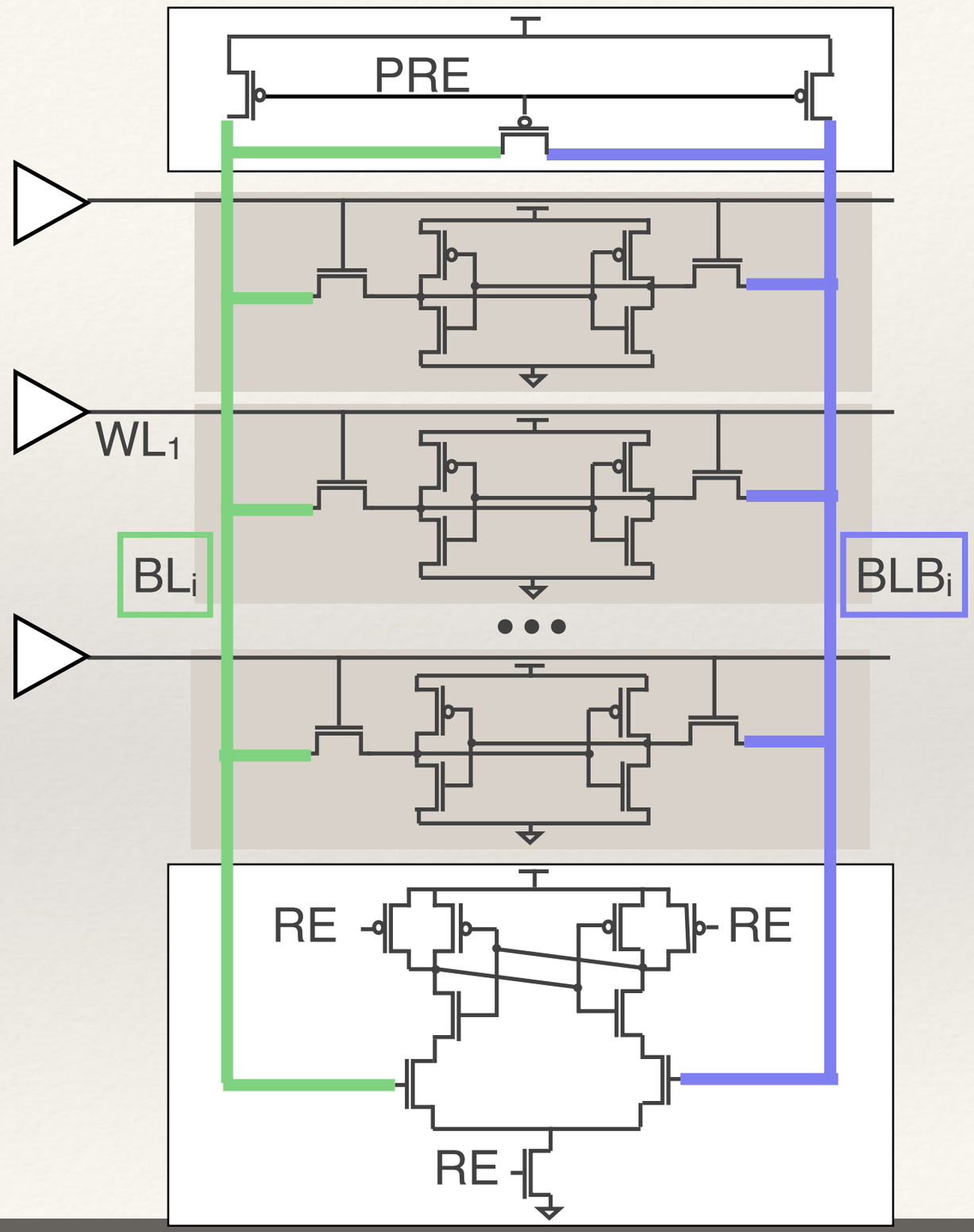


Reading an SRAM Cell

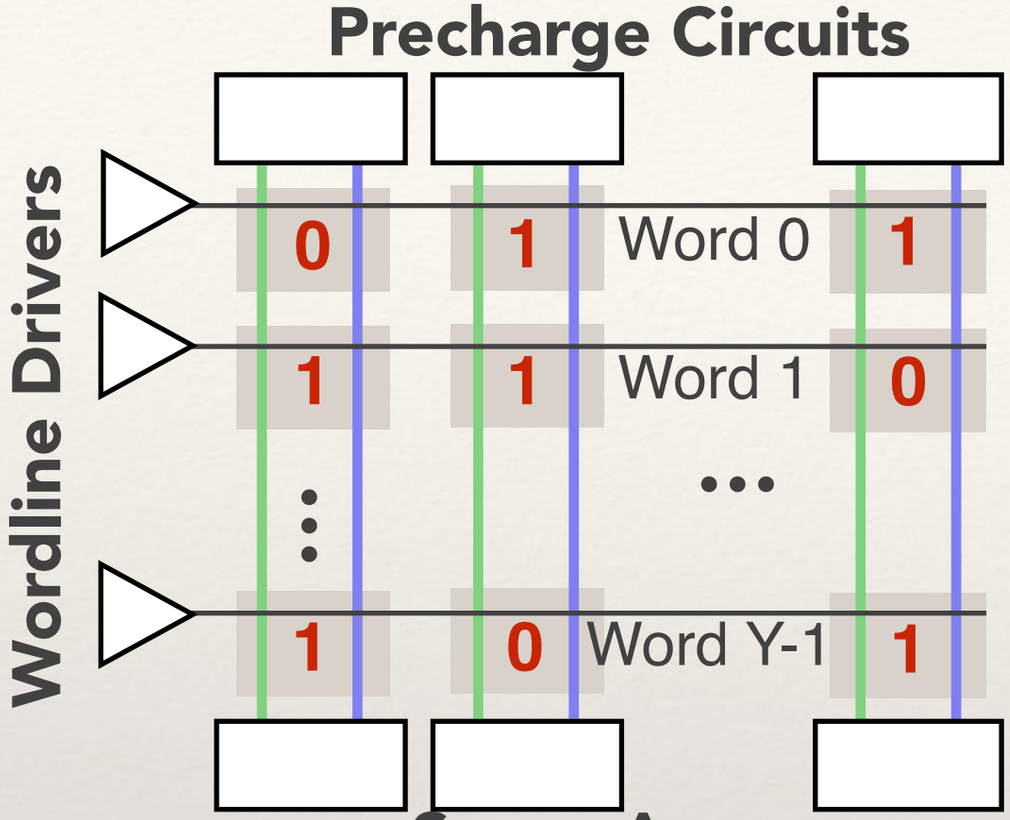


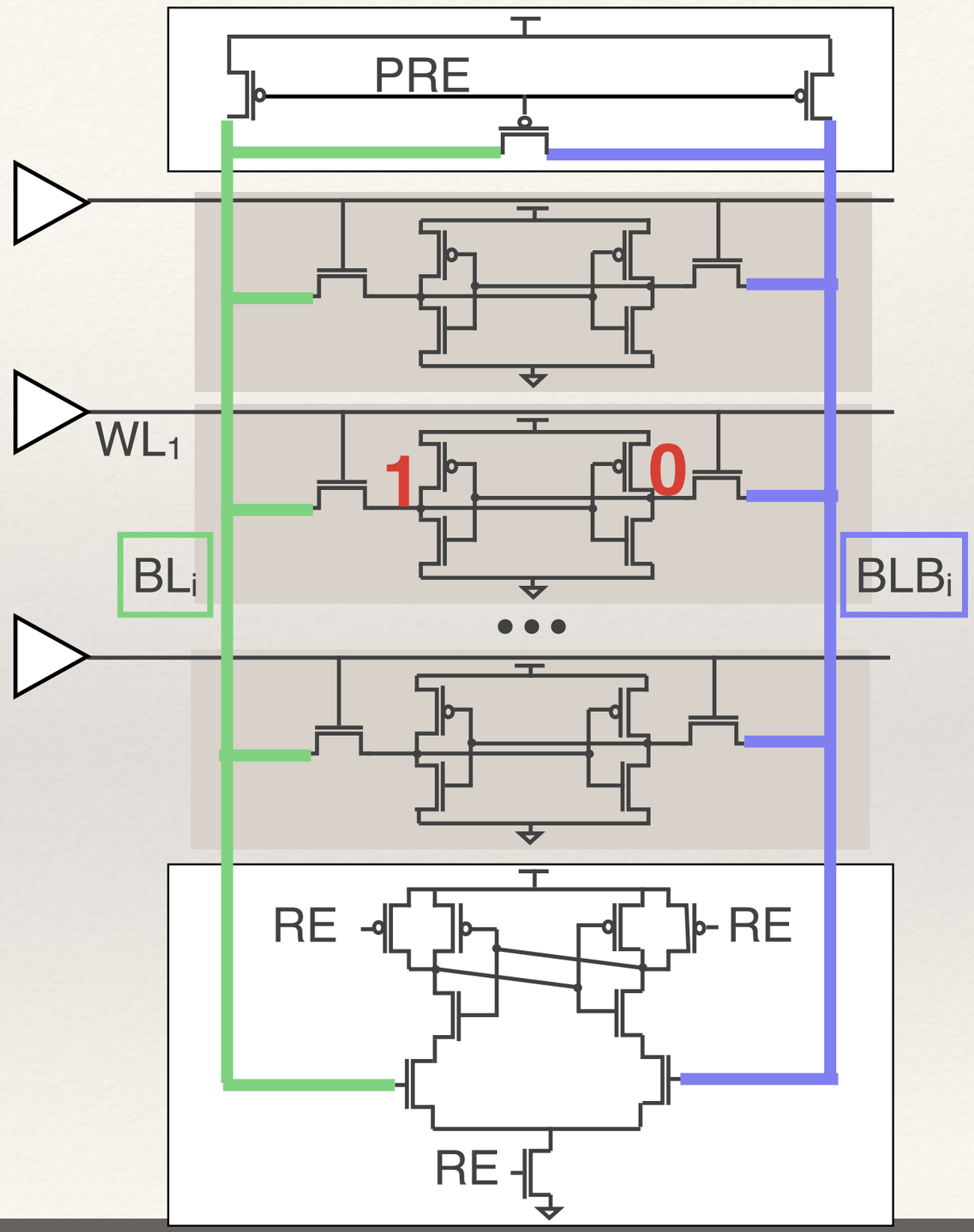
Reading an SRAM Cell



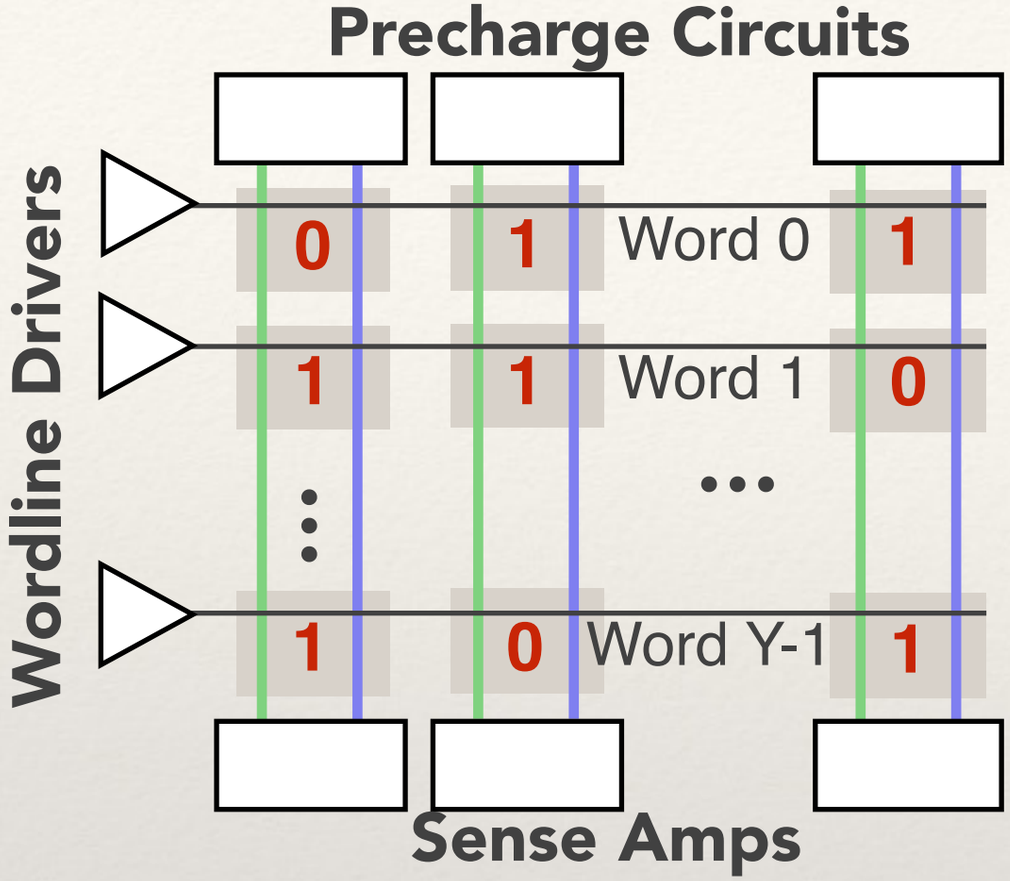


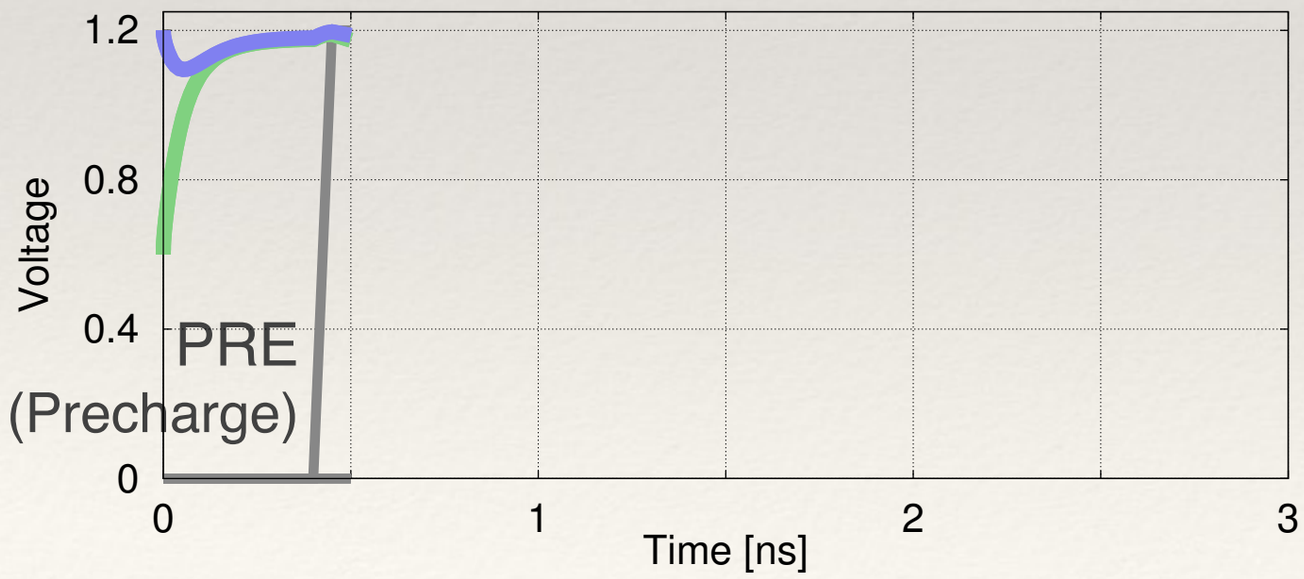
Reading an SRAM Cell

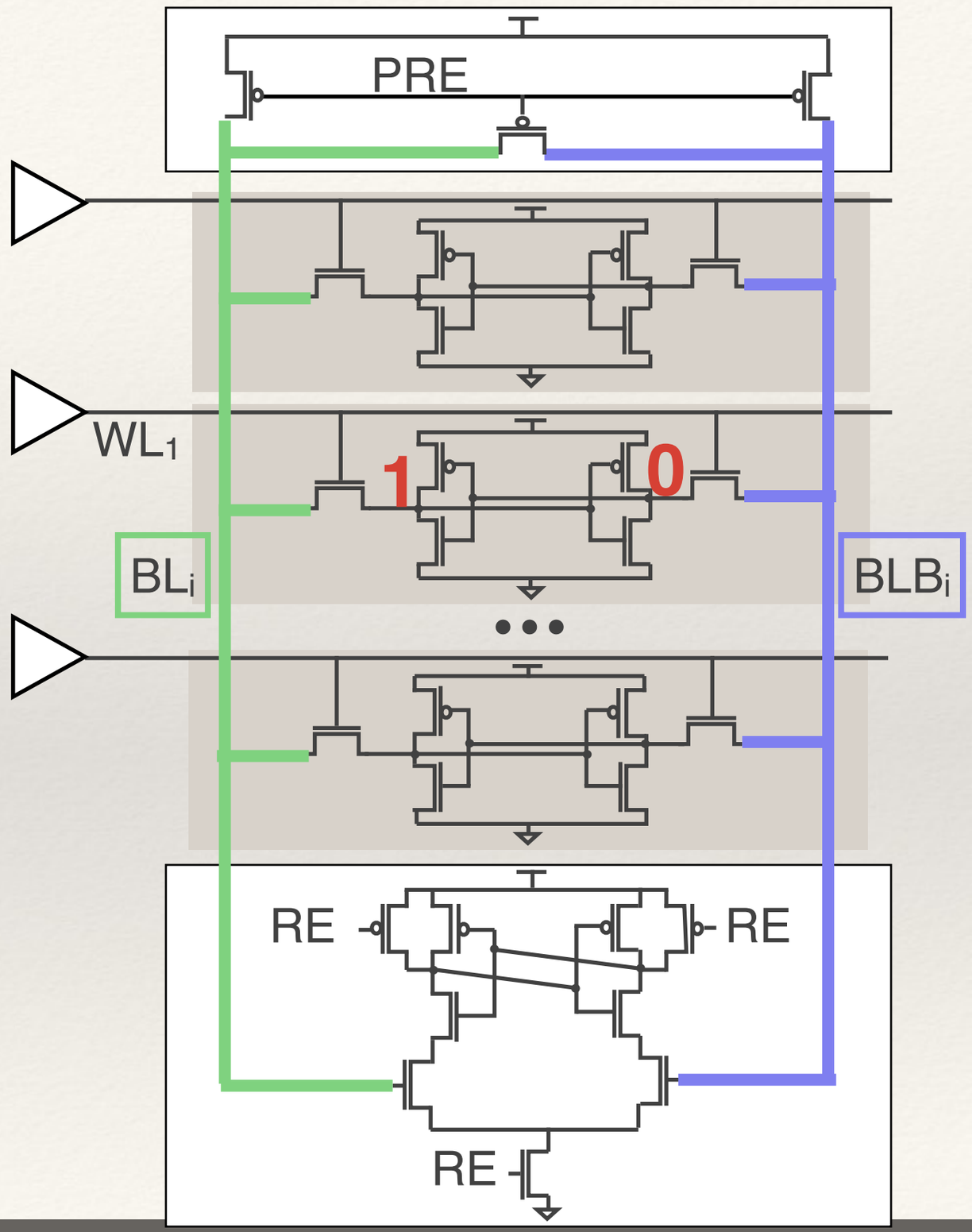




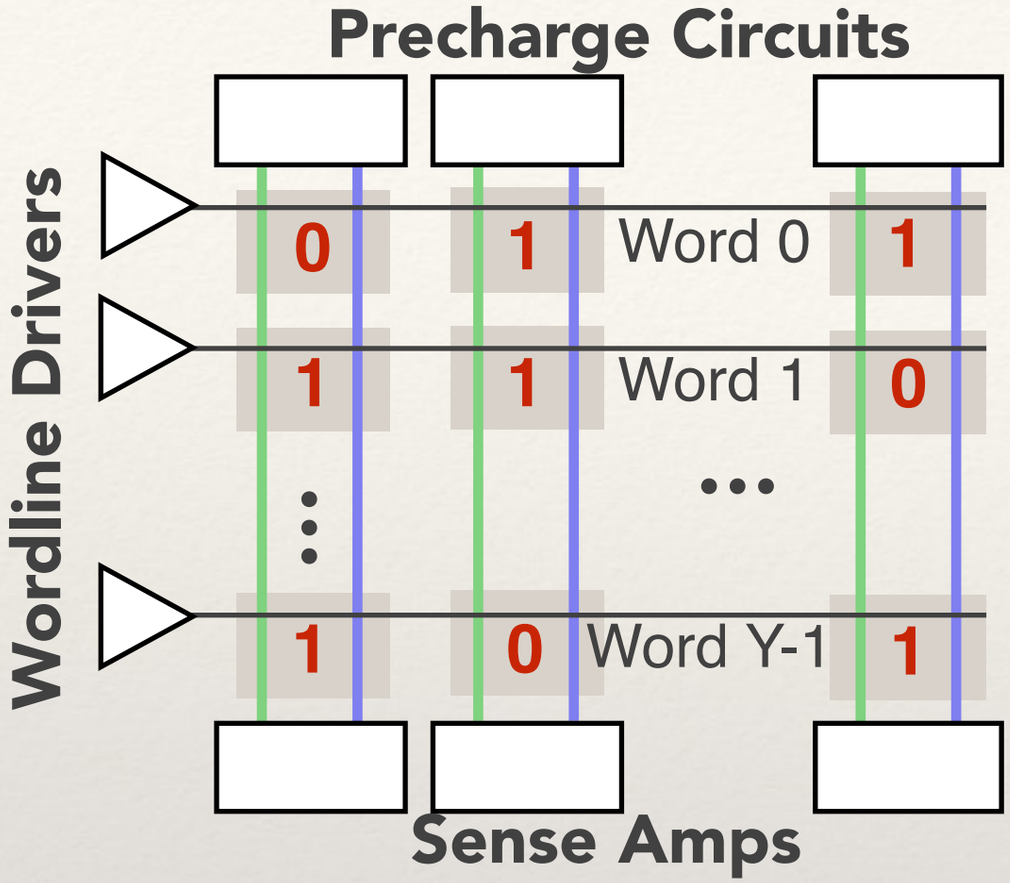
Reading an SRAM Cell

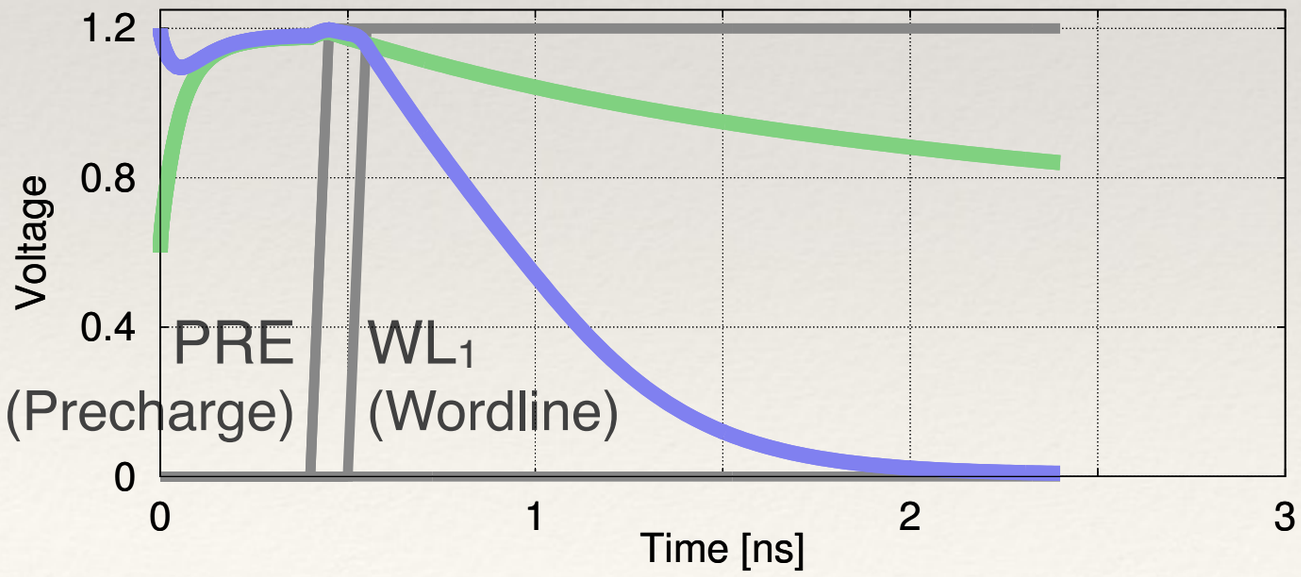


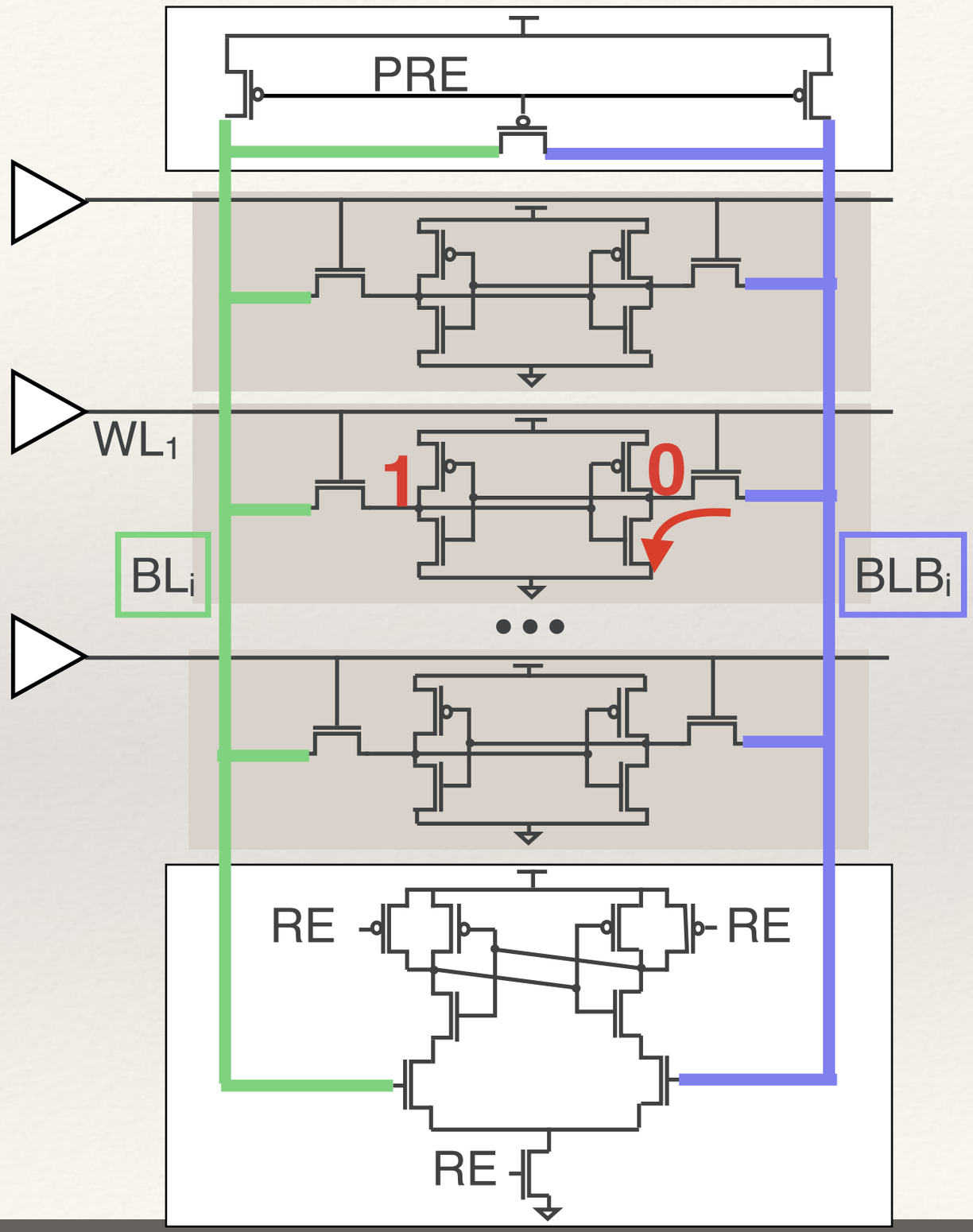




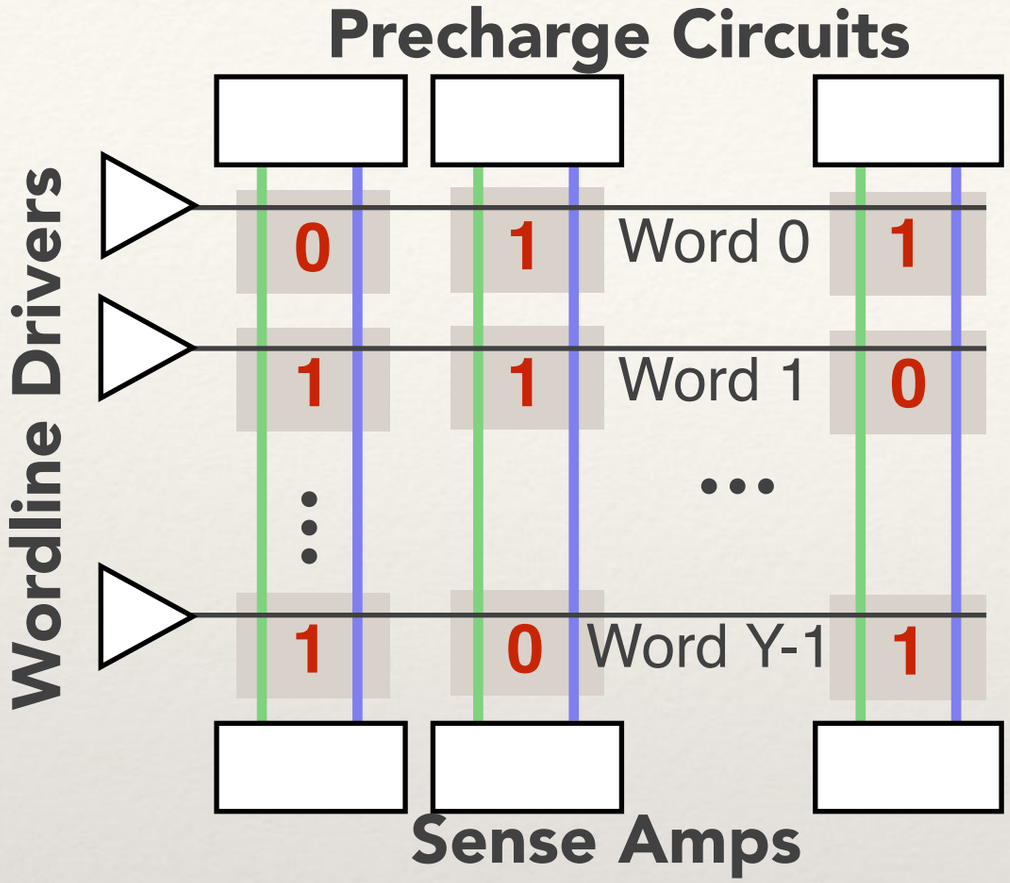
Reading an SRAM Cell

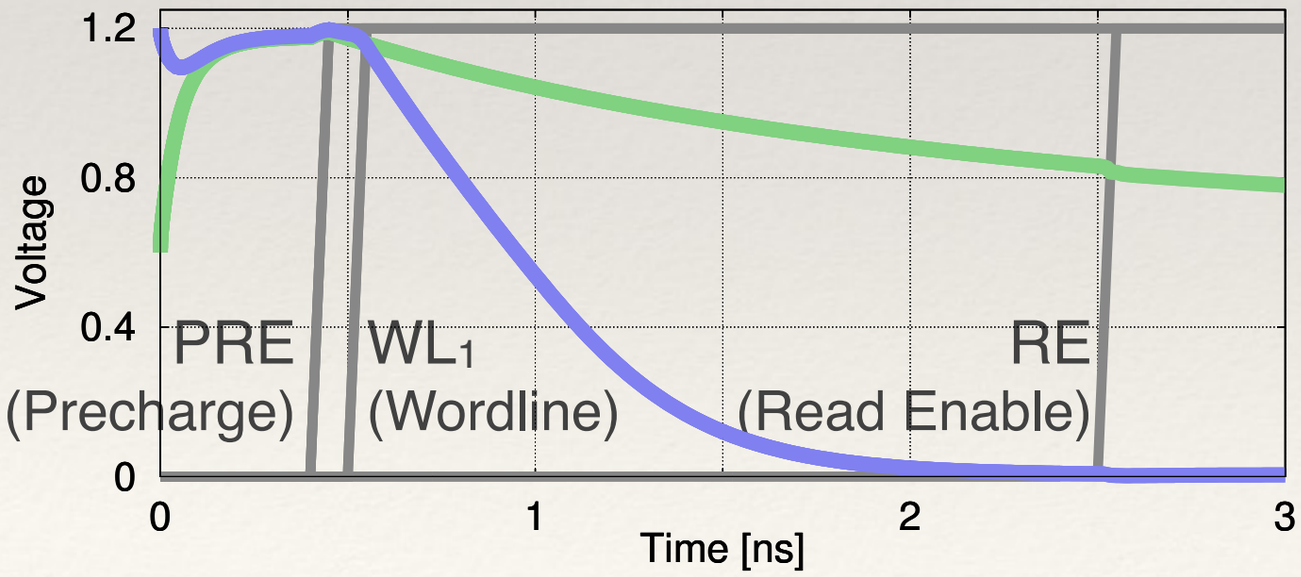


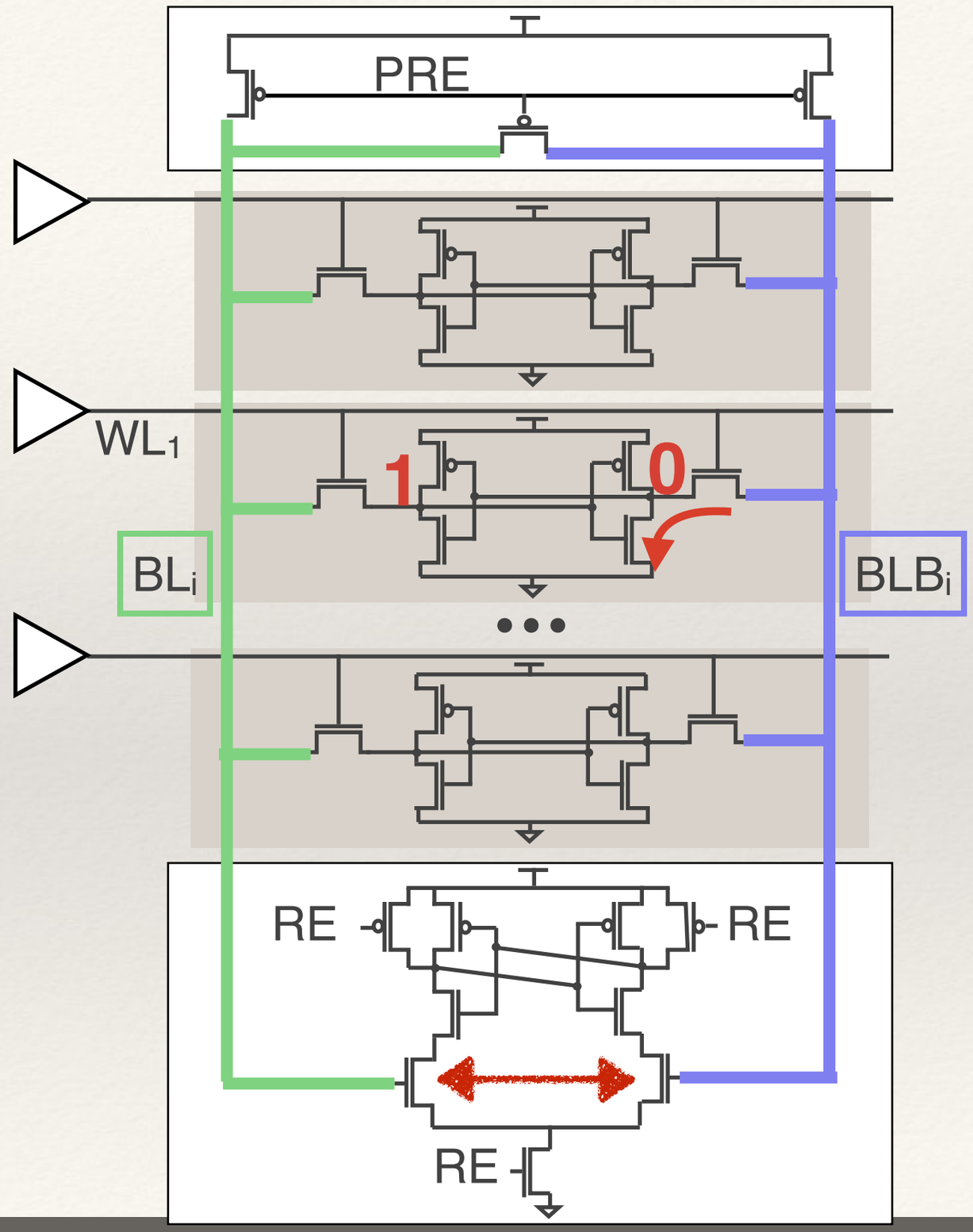




Reading an SRAM Cell

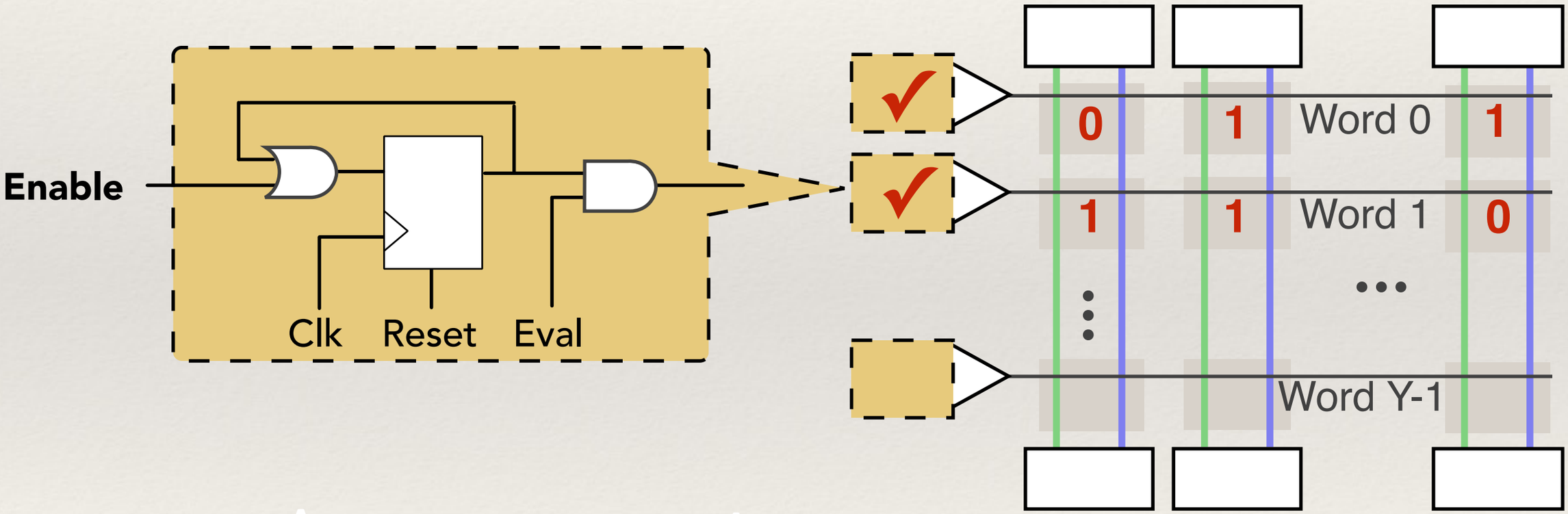


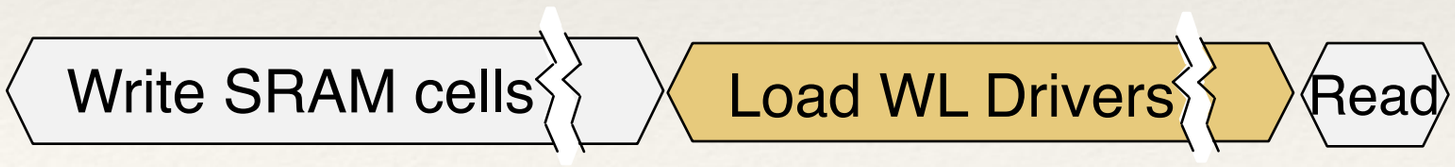




Bitline PUF

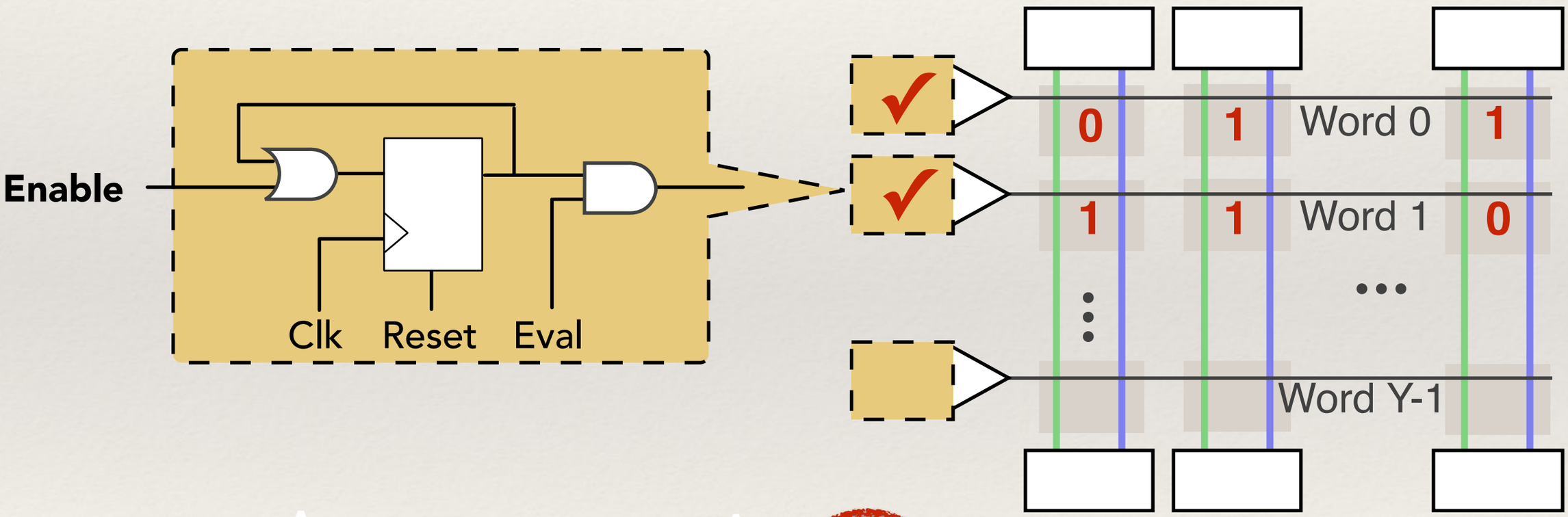
- Accumulate wordline enable signals for **concurrent read**
- Concurrent reading causes contention
- Contention resolves according to variations

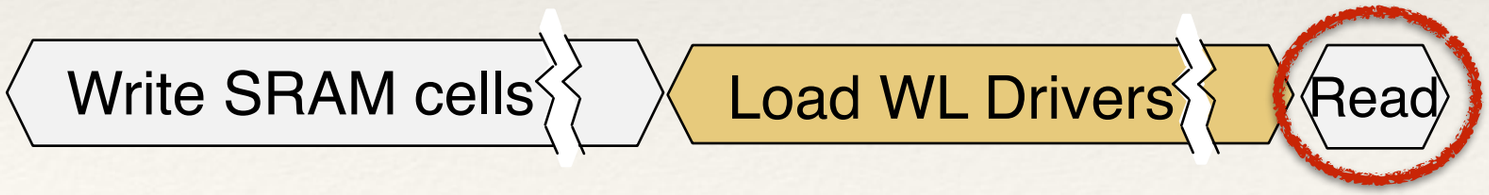




Bitline PUF

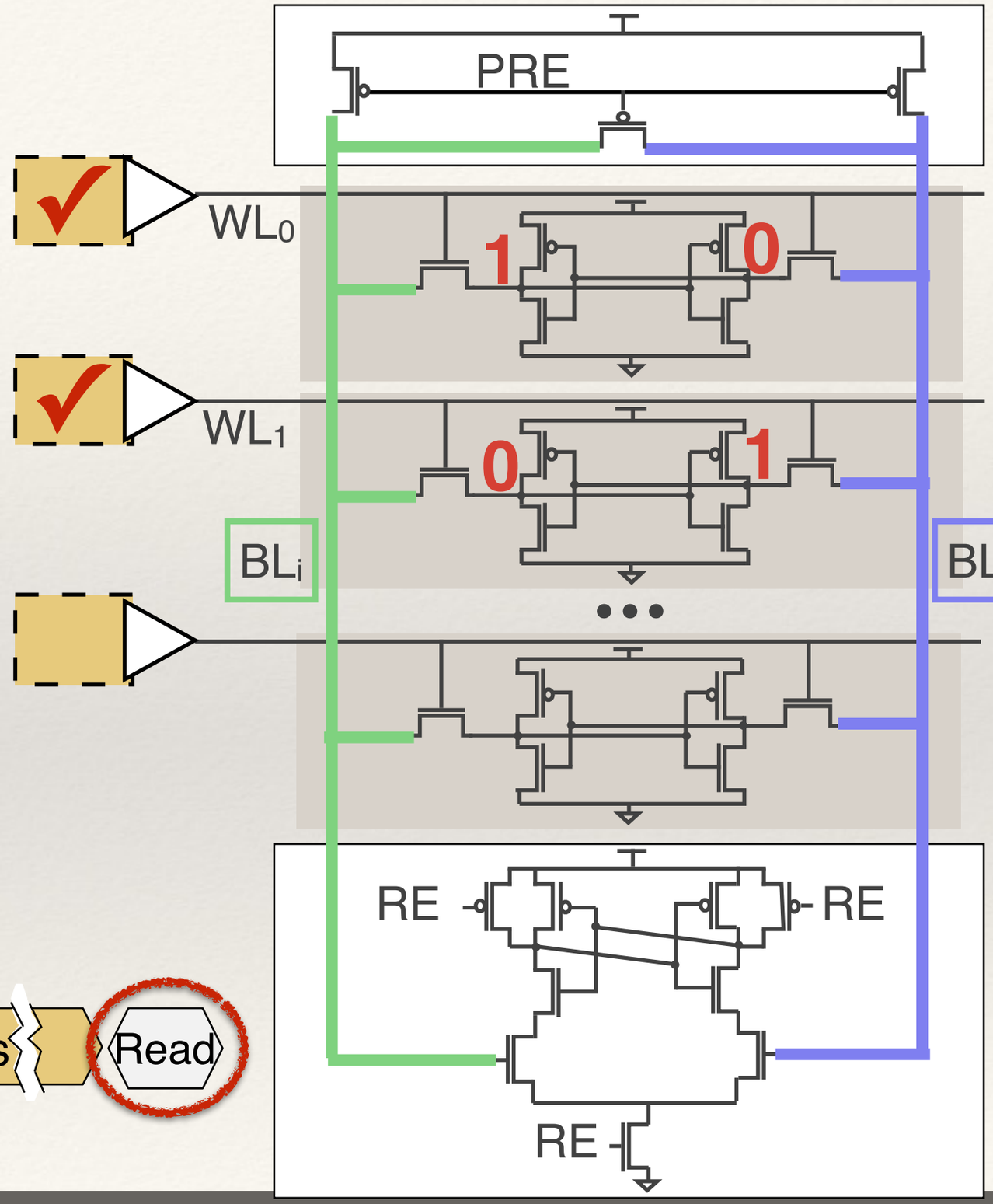
- Accumulate wordline enable signals for **concurrent read**
- Concurrent reading causes contention
- Contention resolves according to variations

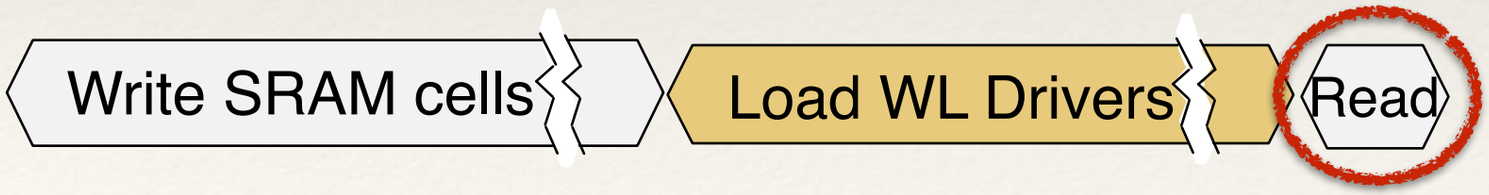




Reading a Bitline PUF

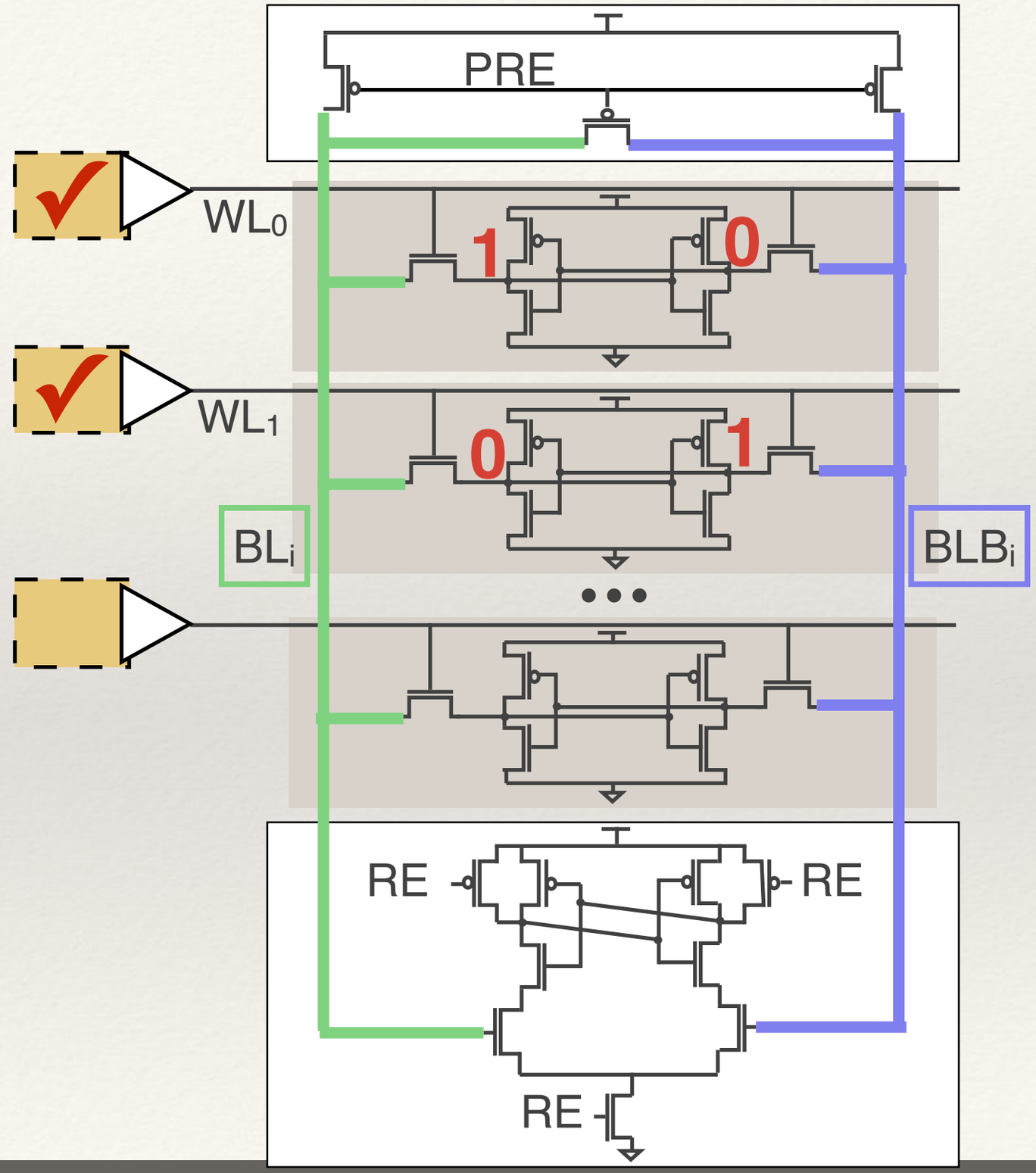
- Read with contention
- Contention resolves according to variation





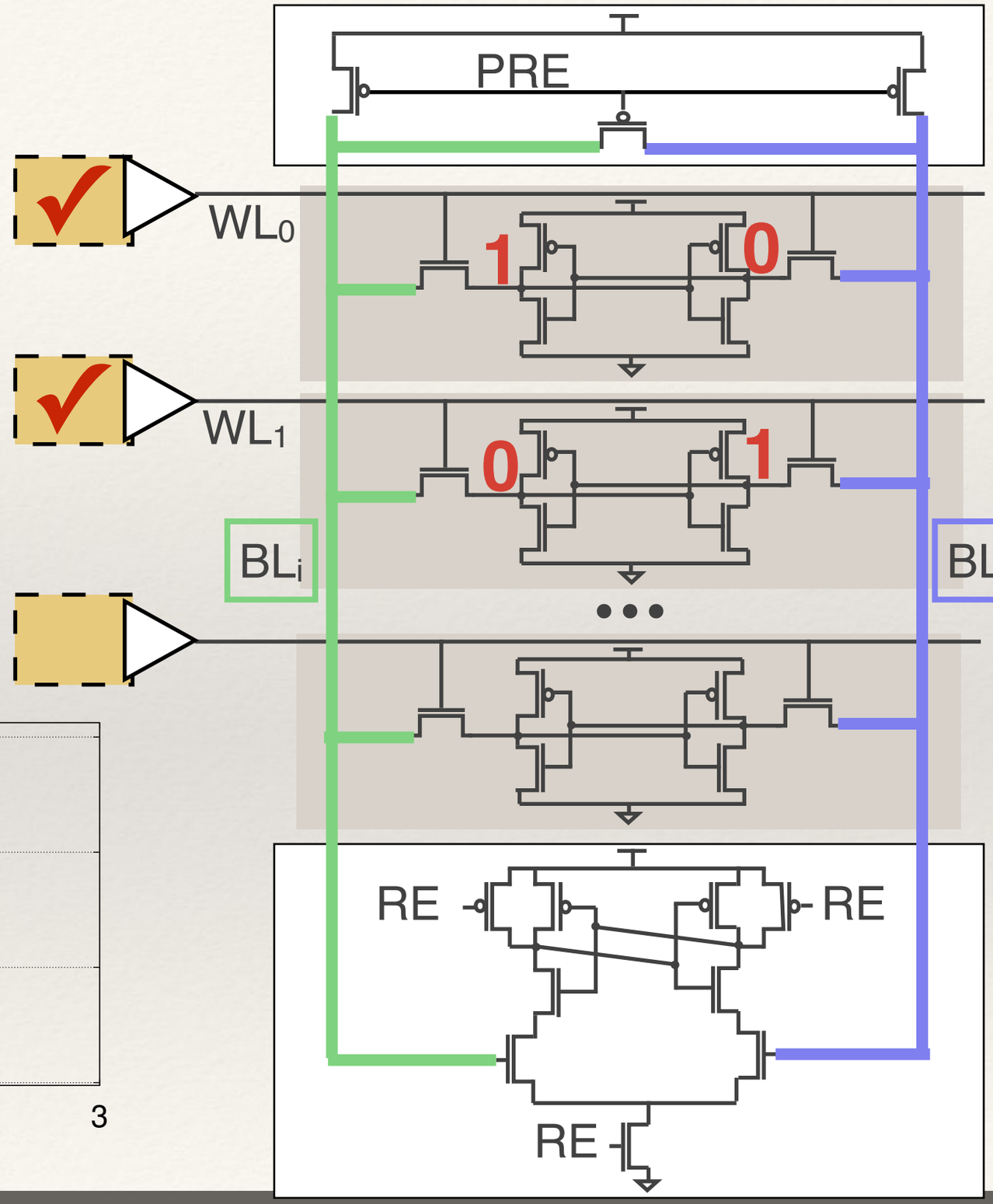
Reading a Bitline PUF

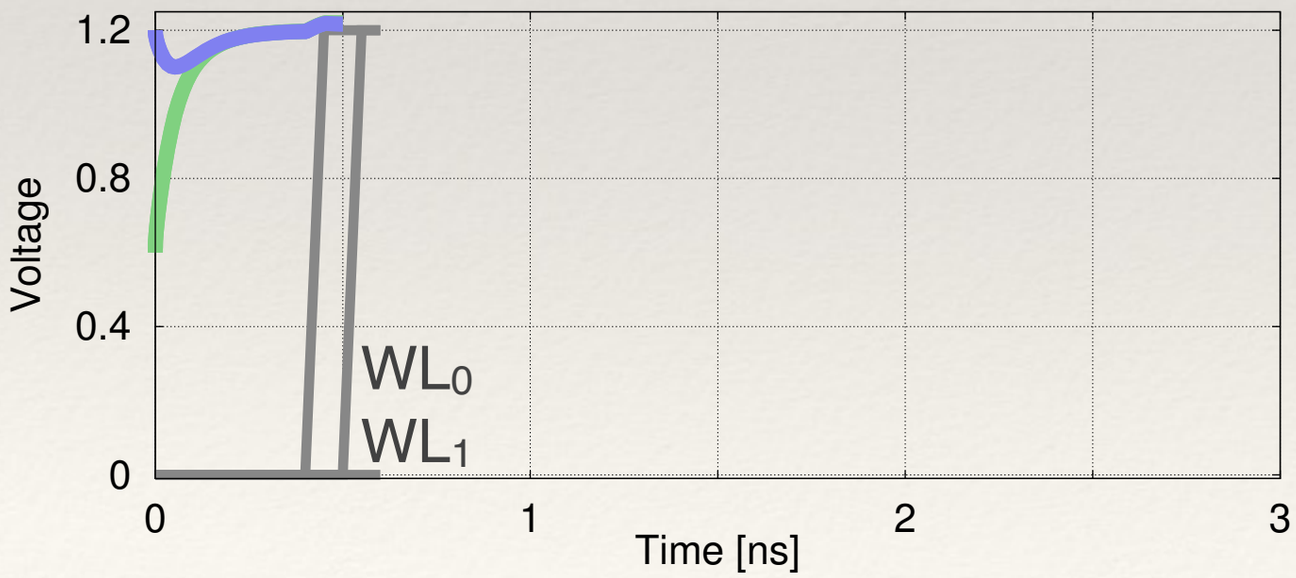
- Read with contention
- Contention resolves according to variation



Reading a Bitline PUF

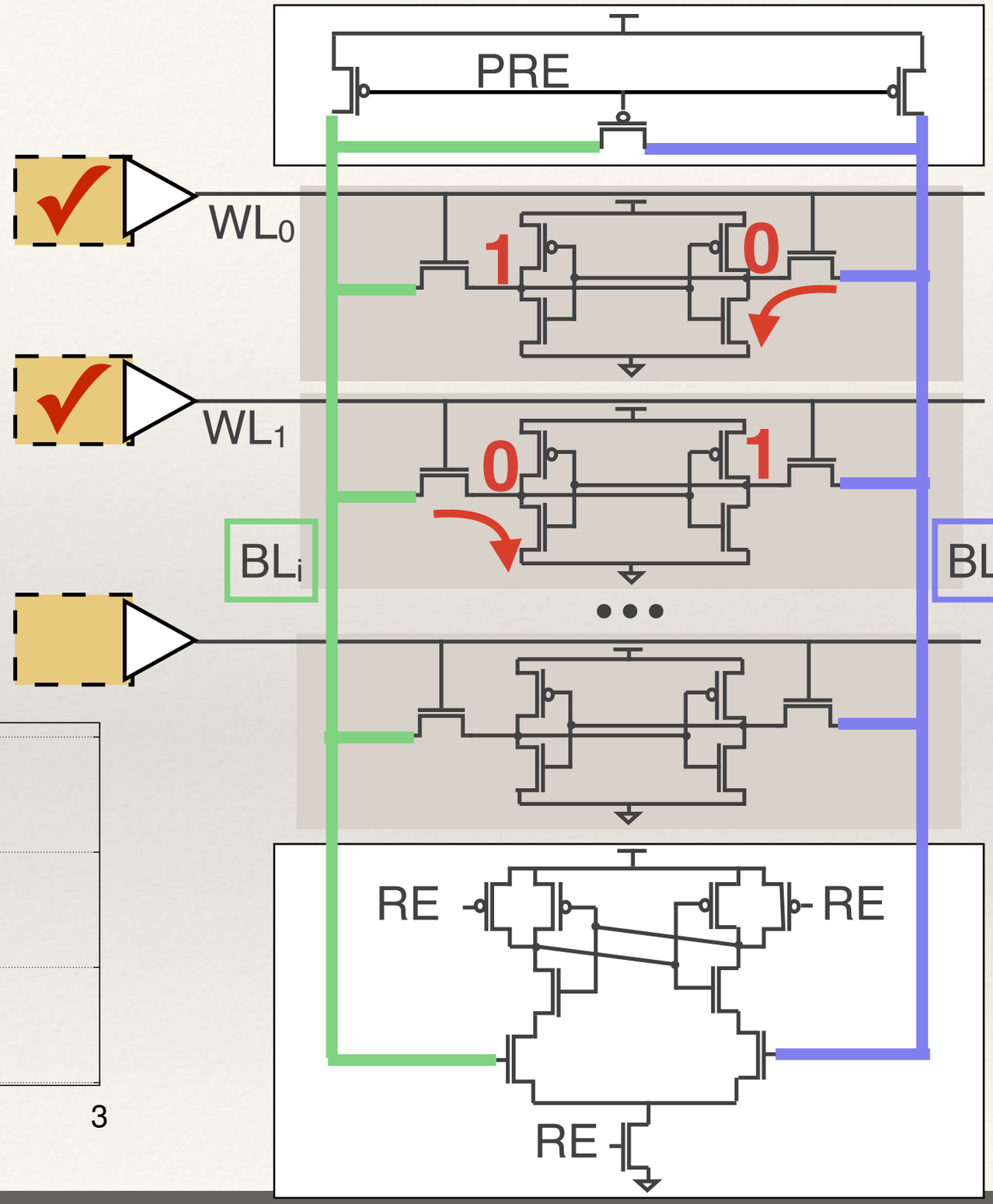
- Read with contention
- Contention resolves according to variation

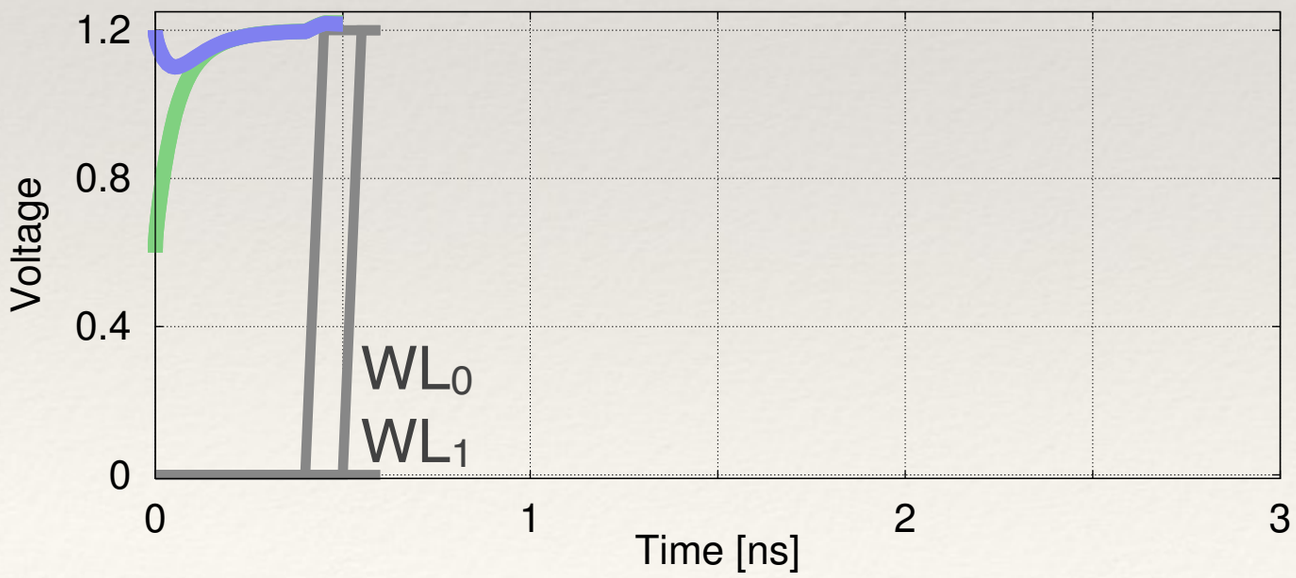




Reading a Bitline PUF

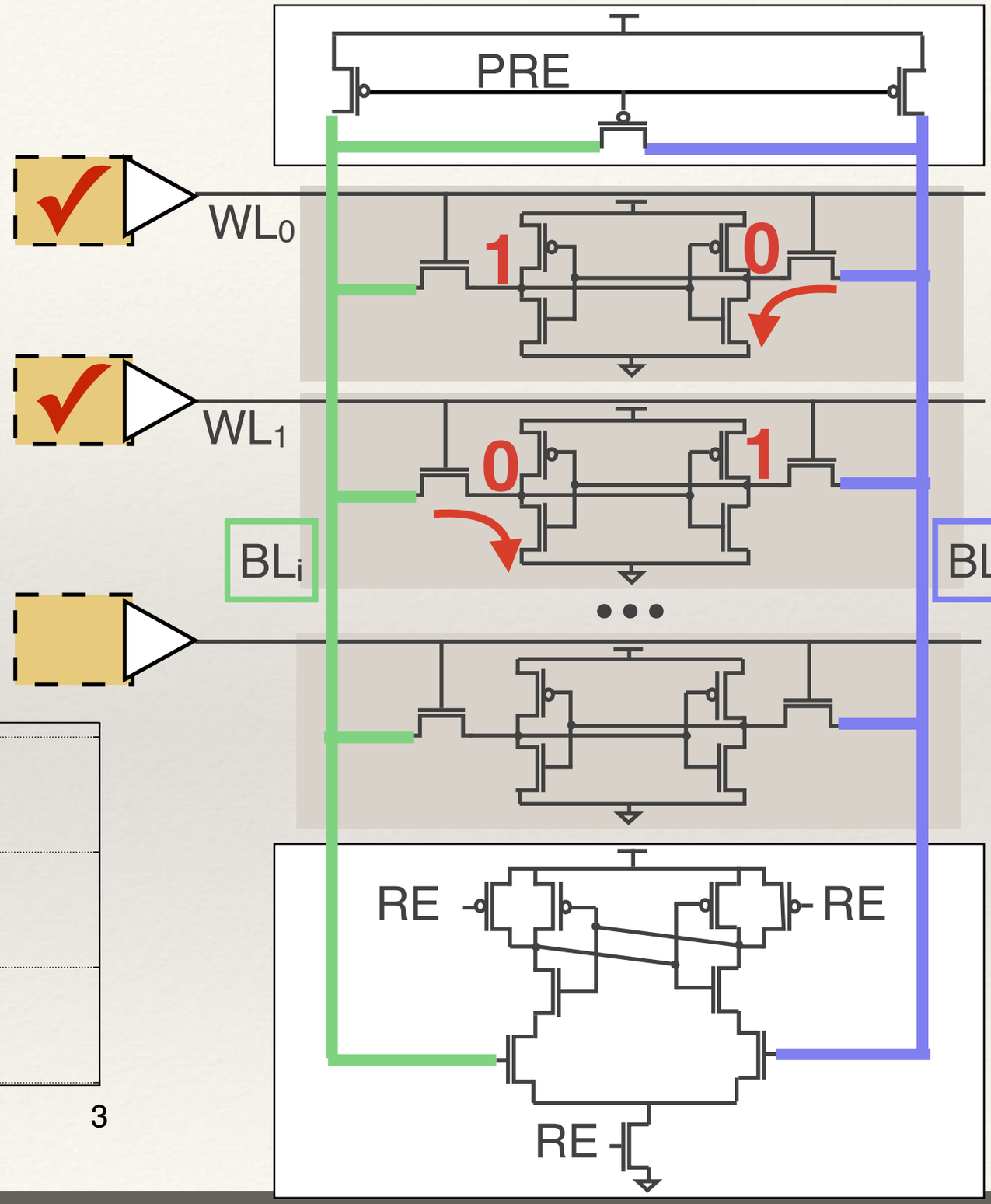
- Read with contention
- Contention resolves according to variation

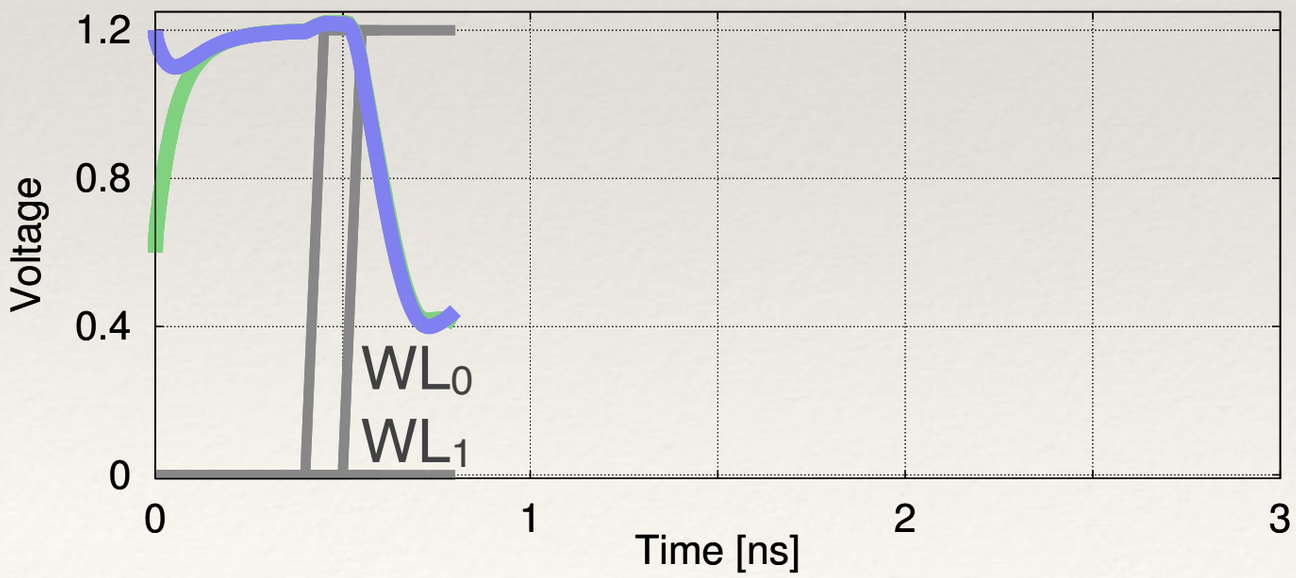




Reading a Bitline PUF

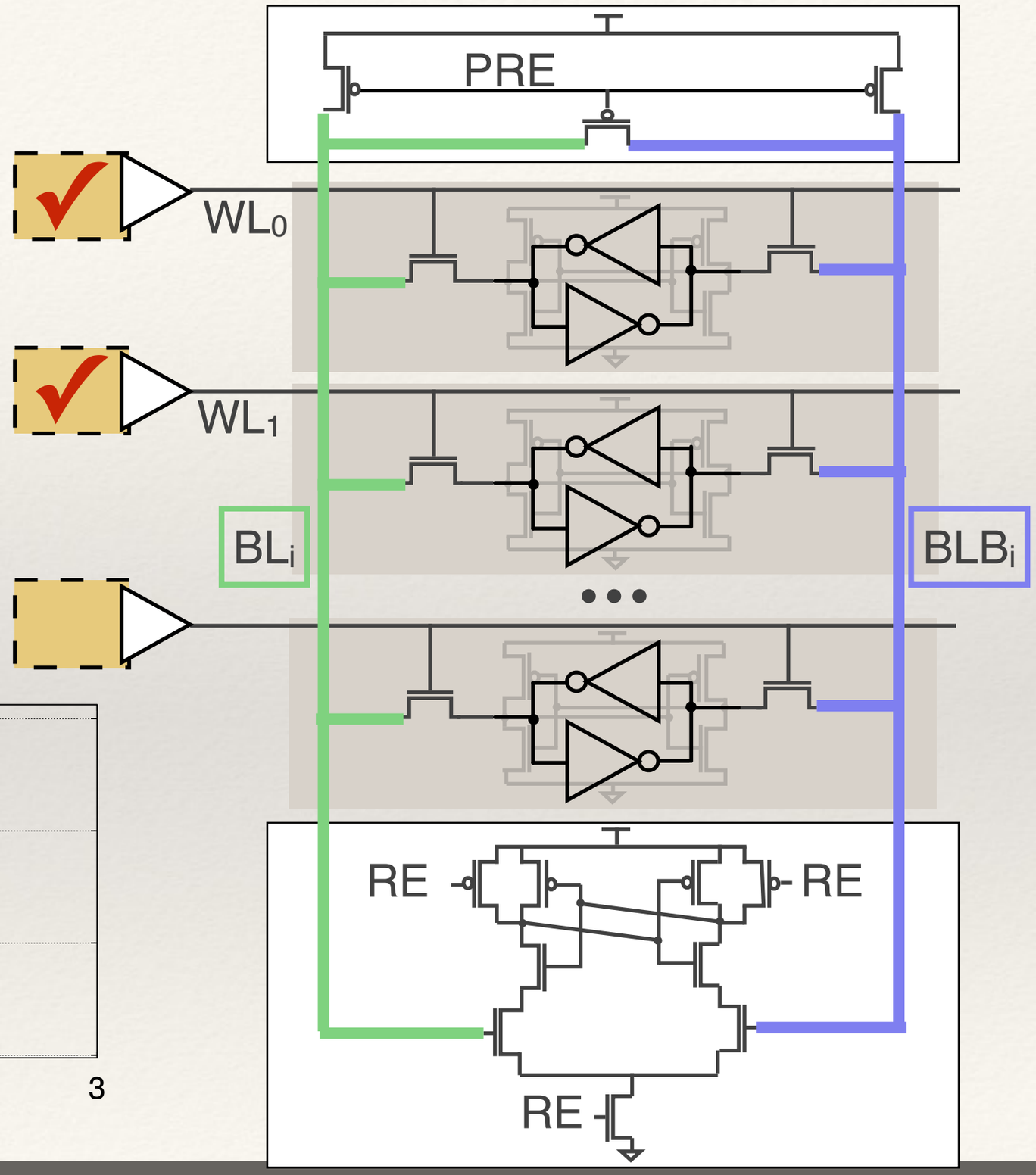
- Read with contention
- Contention resolves according to variation

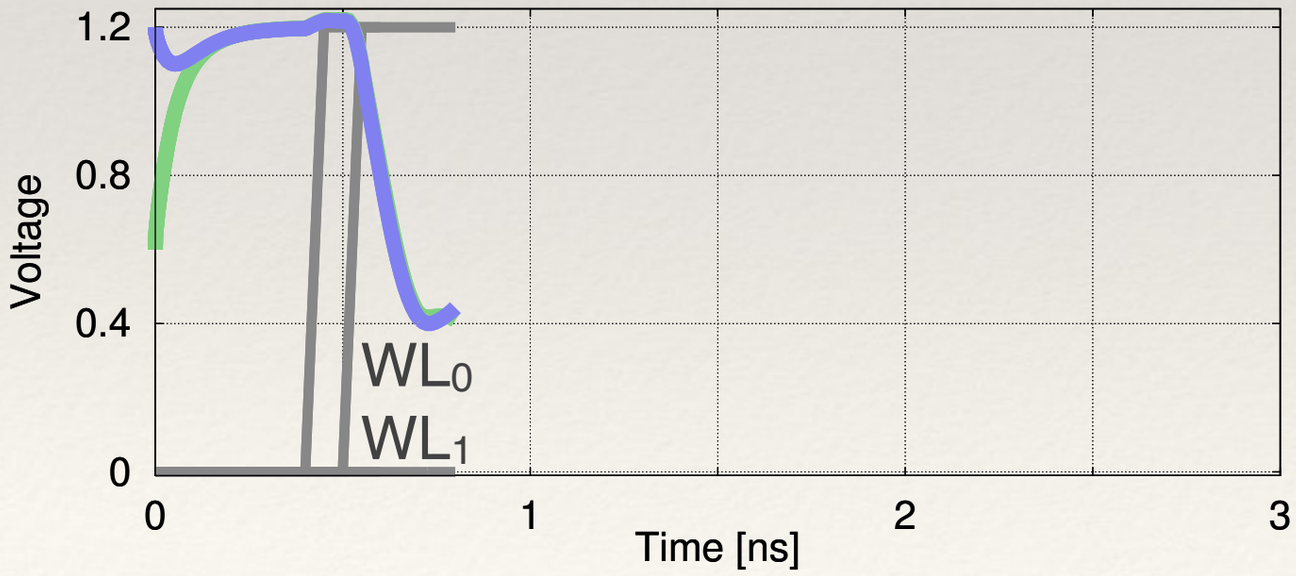




Reading a Bitline PUF

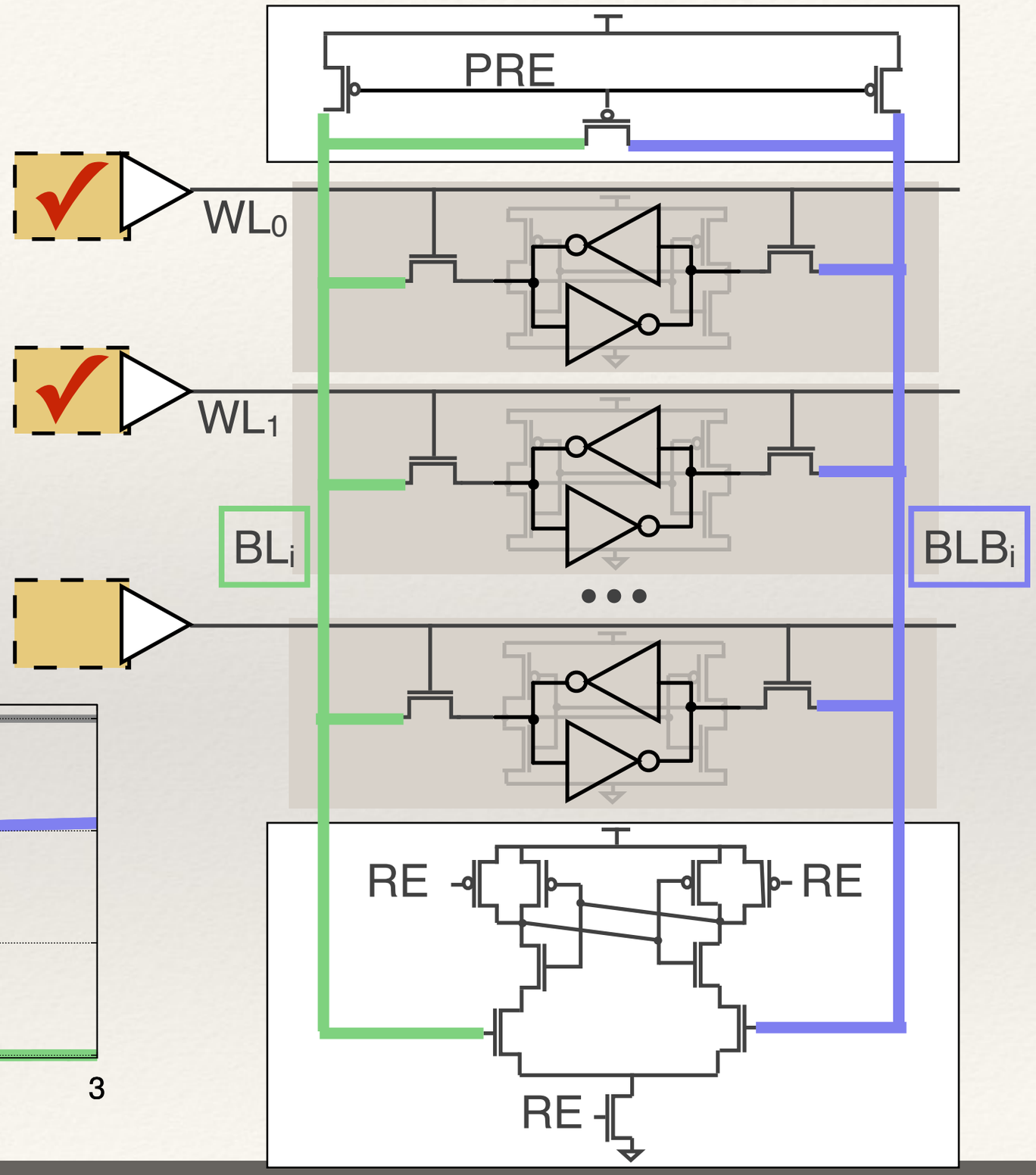
- Read with contention
- Contention resolves according to variation

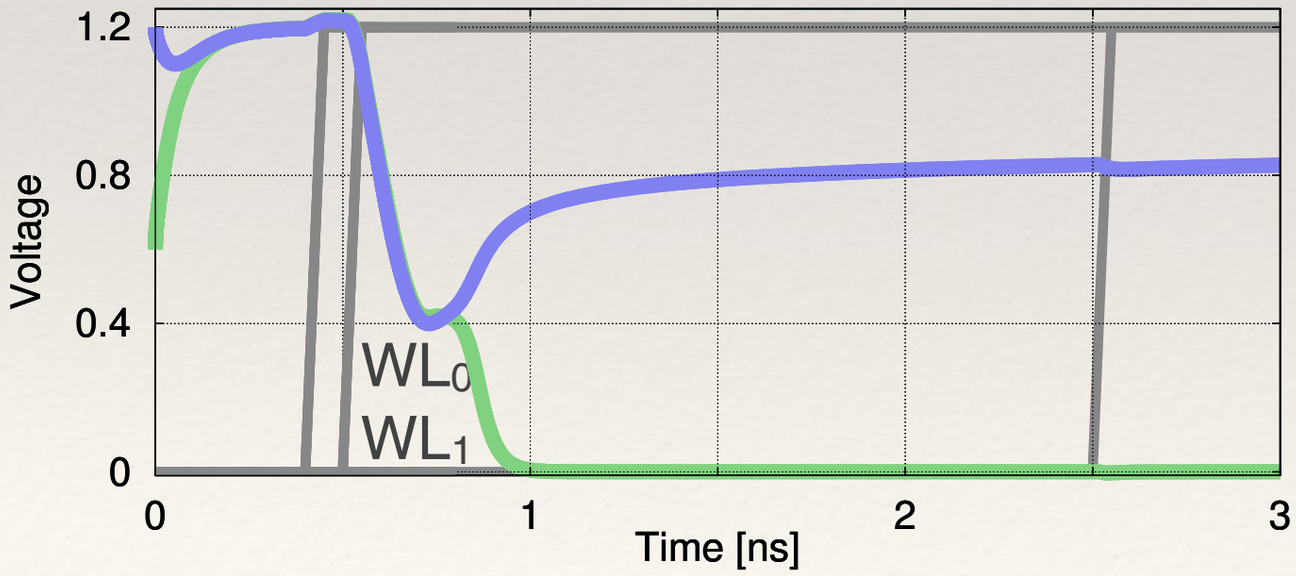




Reading a Bitline PUF

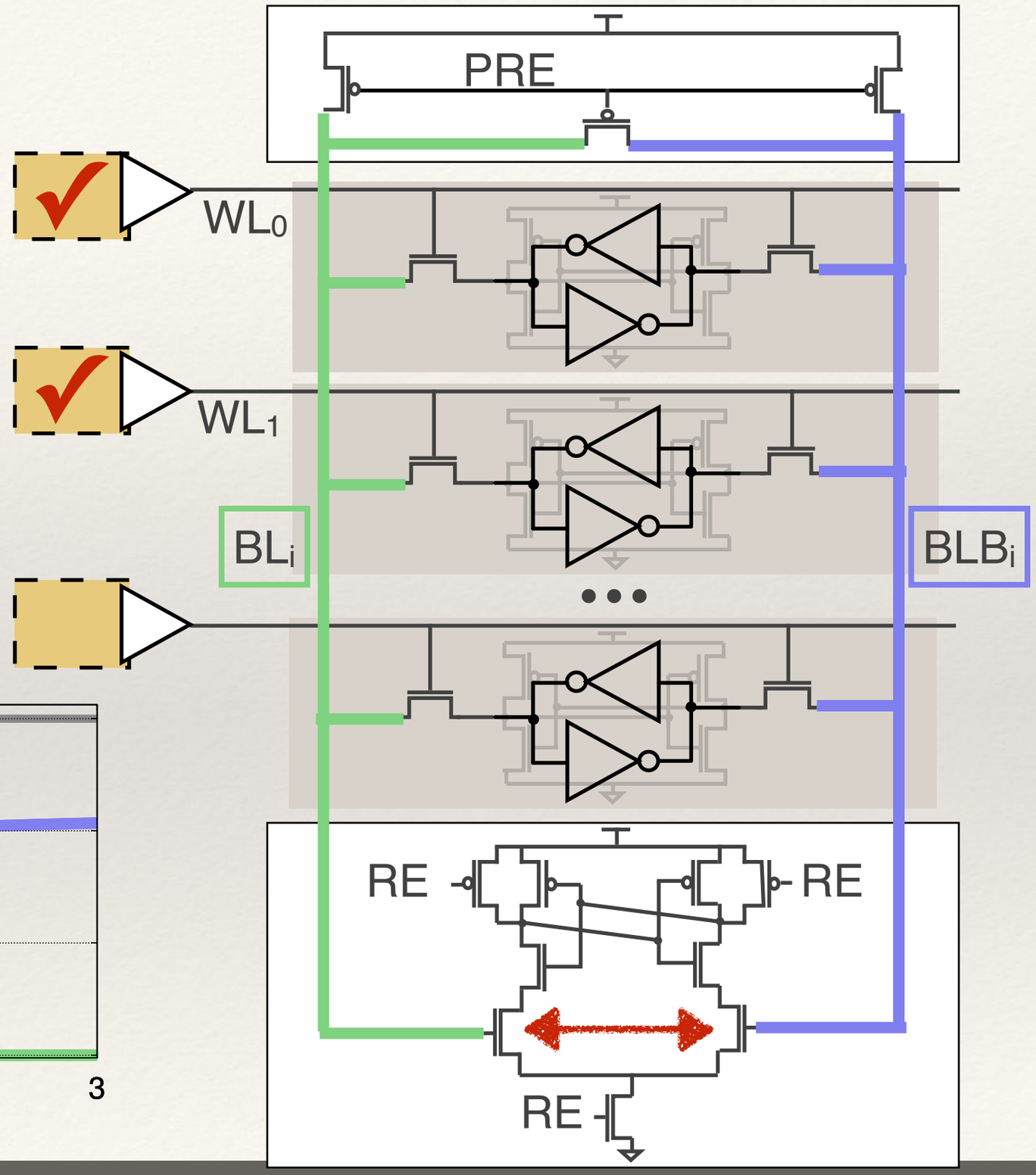
- Read with contention
- Contention resolves according to variation

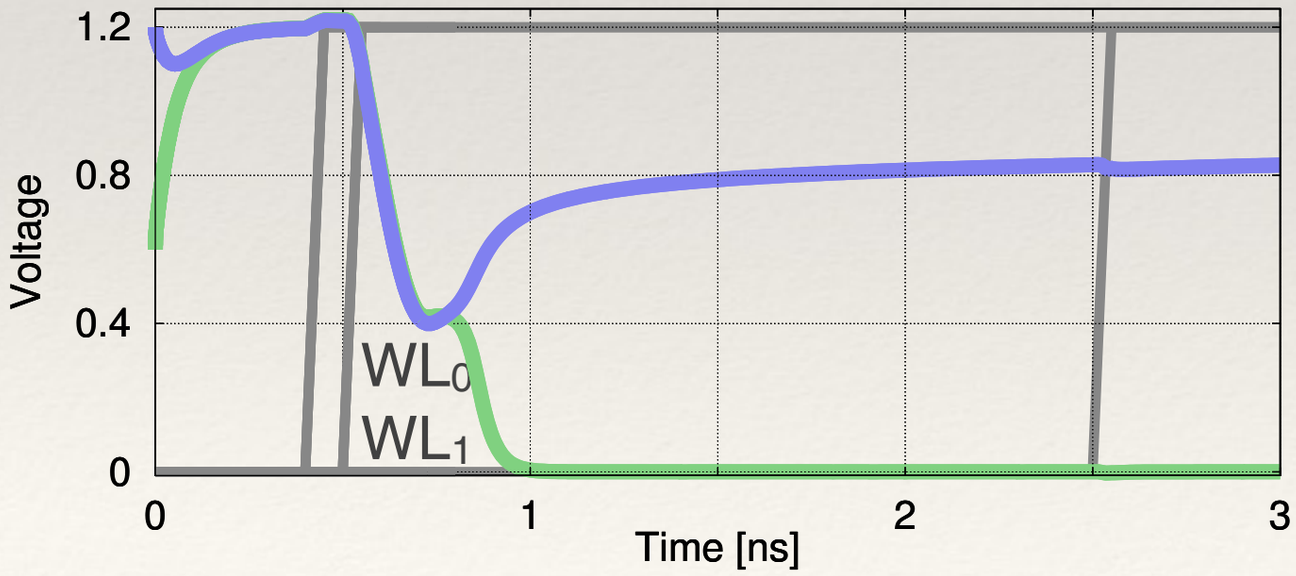




Reading a Bitline PUF

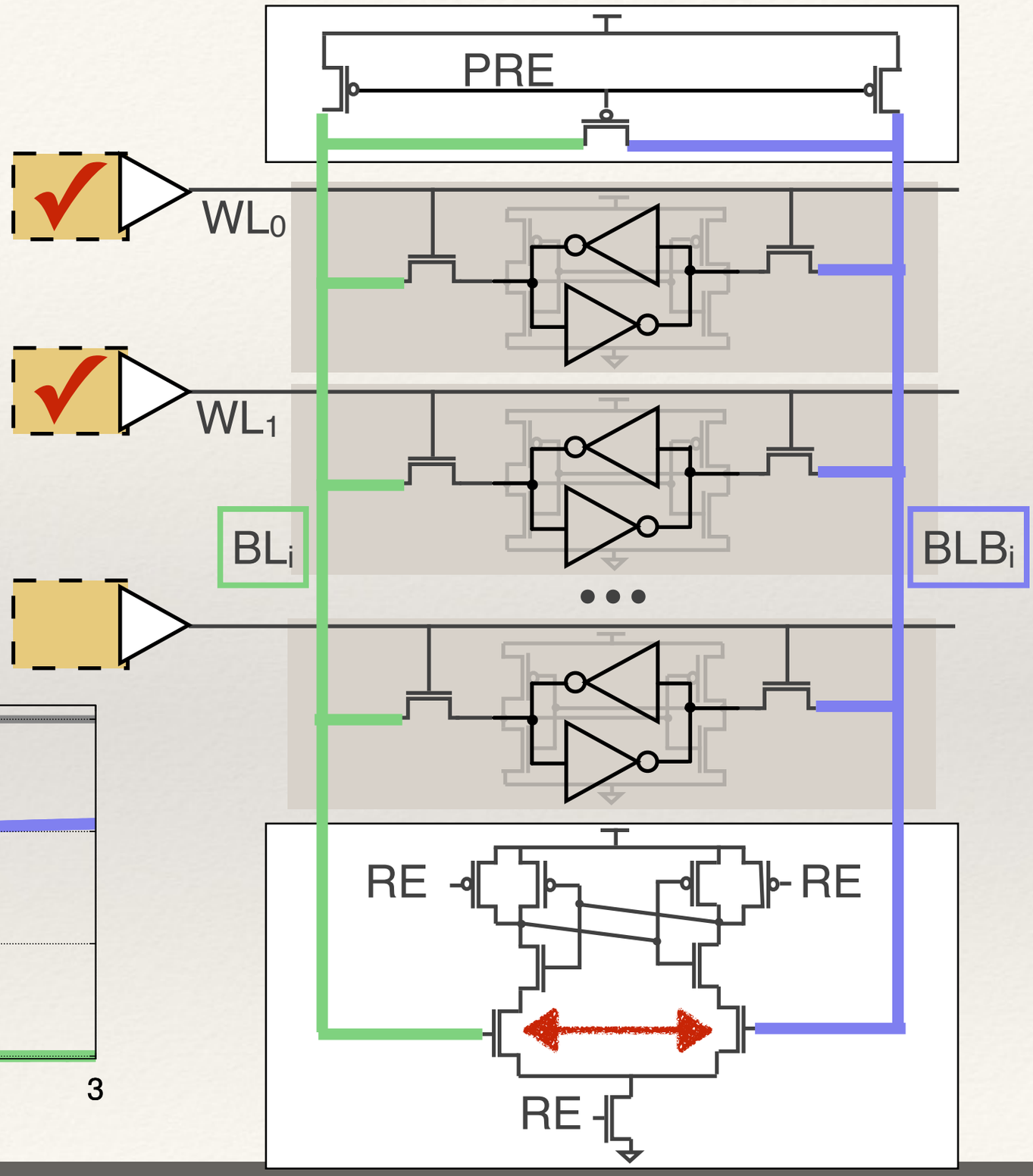
- Read with contention
- Contention resolves according to variation

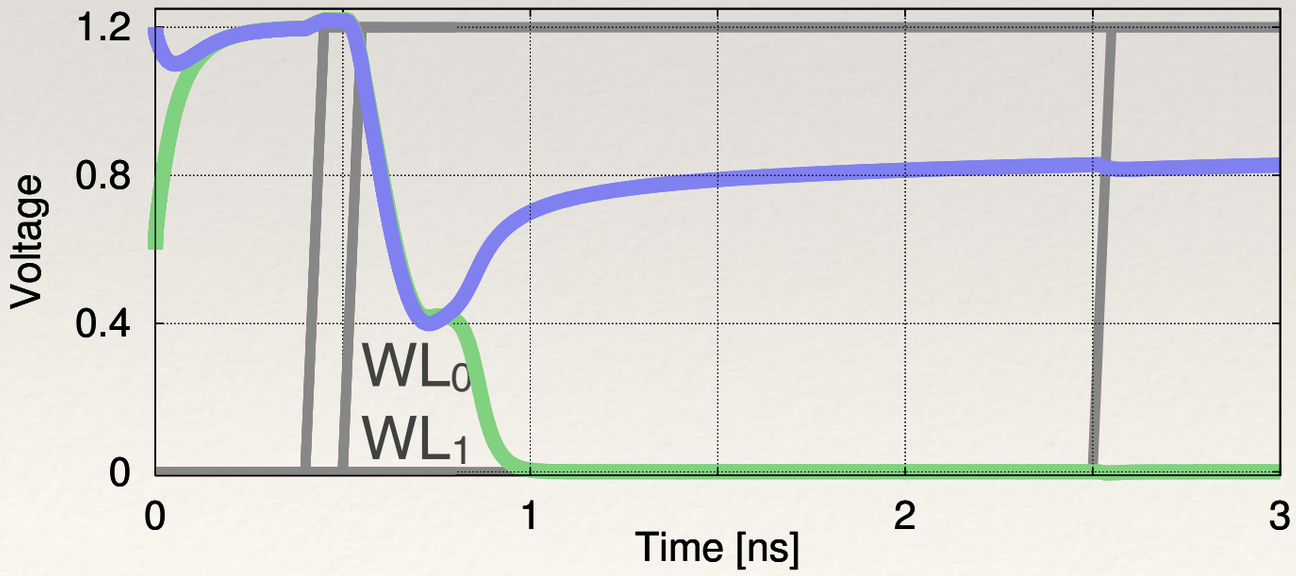




Reading a Bitline PUF

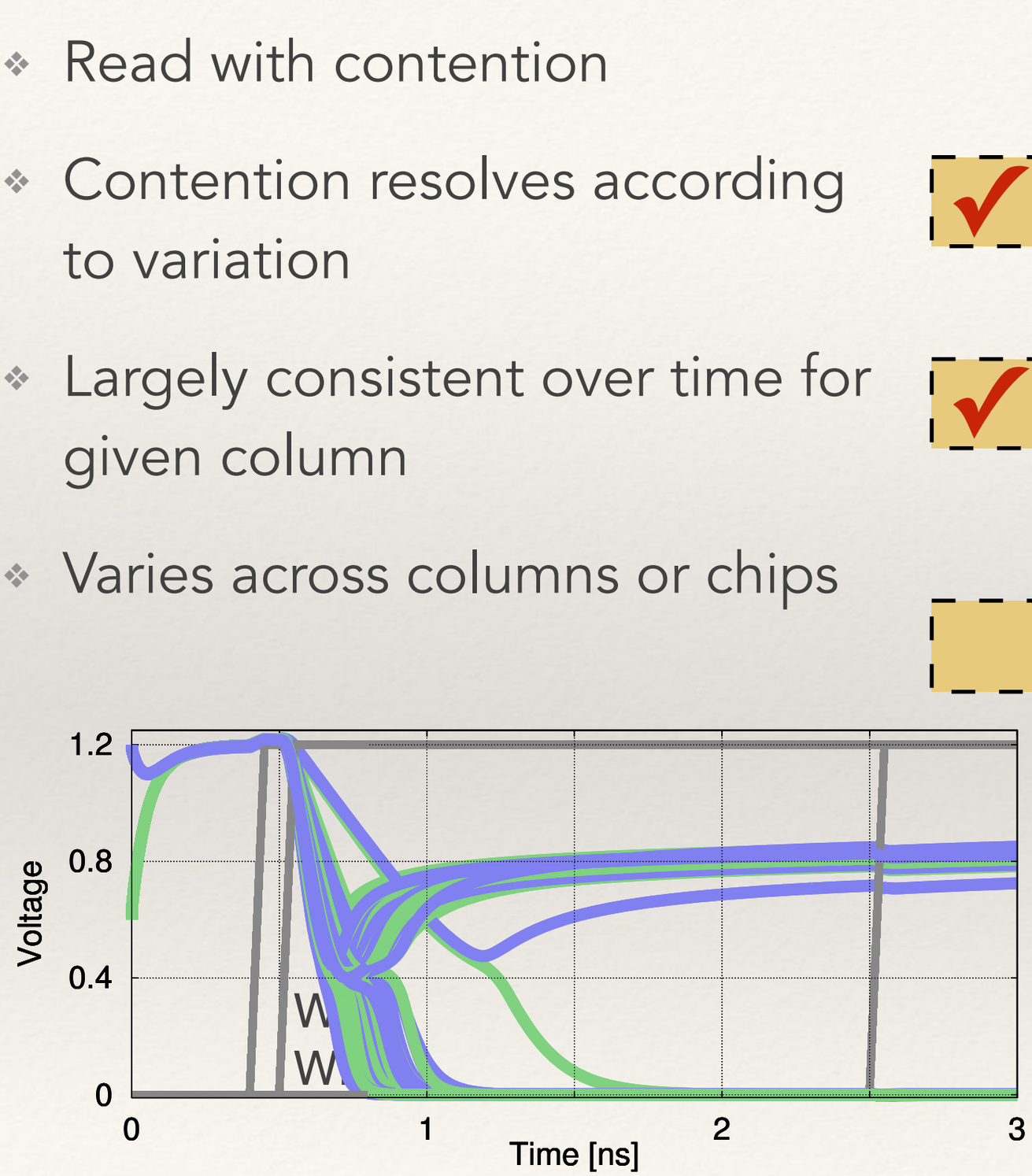
- Read with contention
- Contention resolves according to variation
- Largely consistent over time for given column

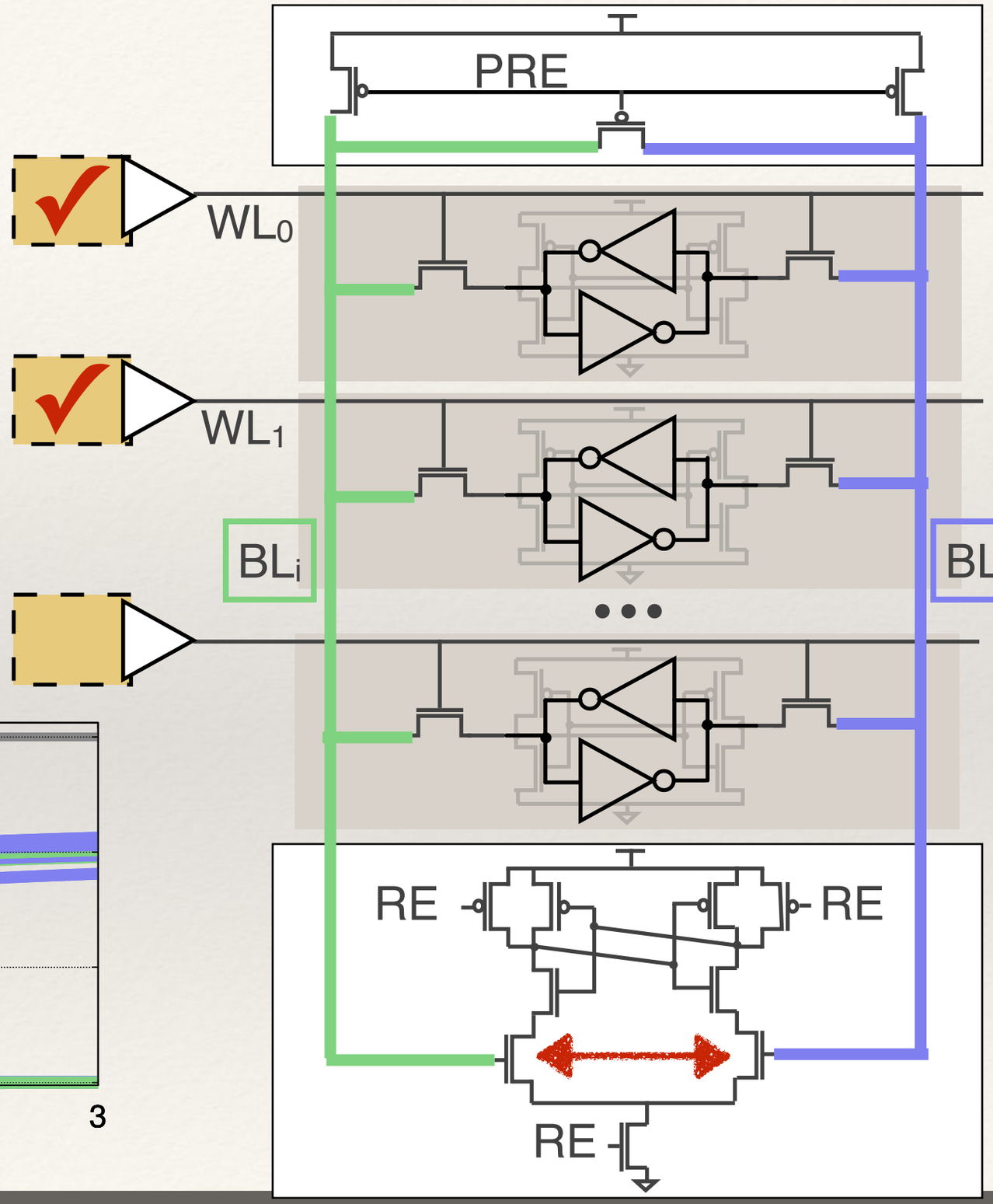




Reading a Bitline PUF

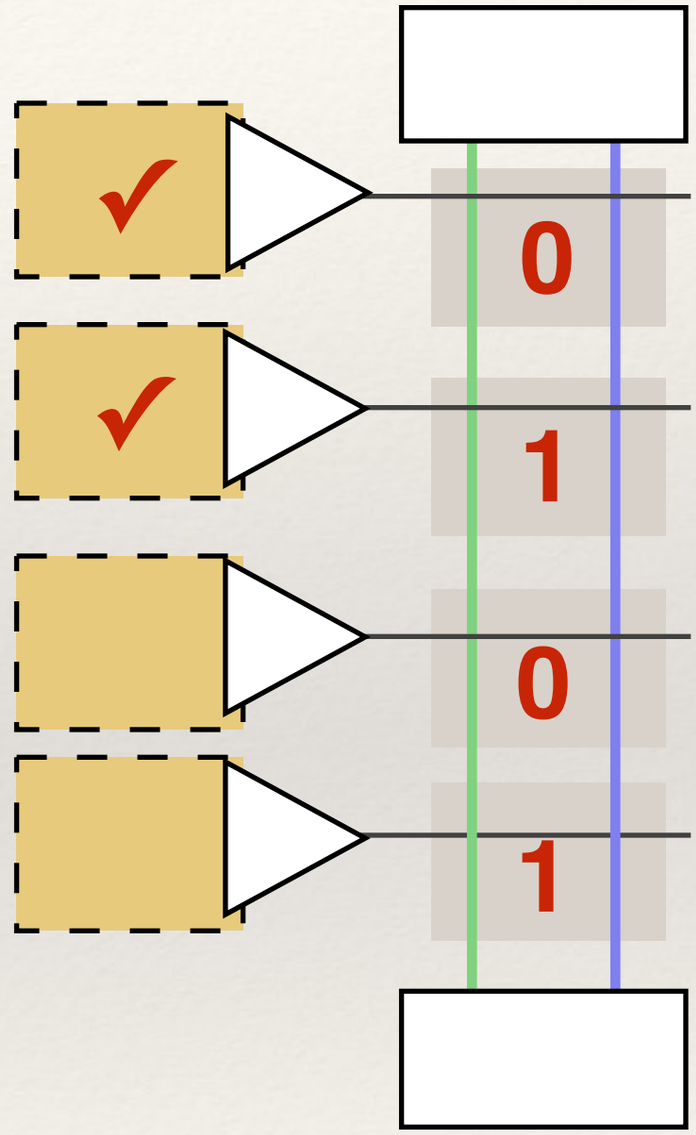
- Read with contention
- Contention resolves according to variation
- Largely consistent over time for given column
- Varies across columns or chips



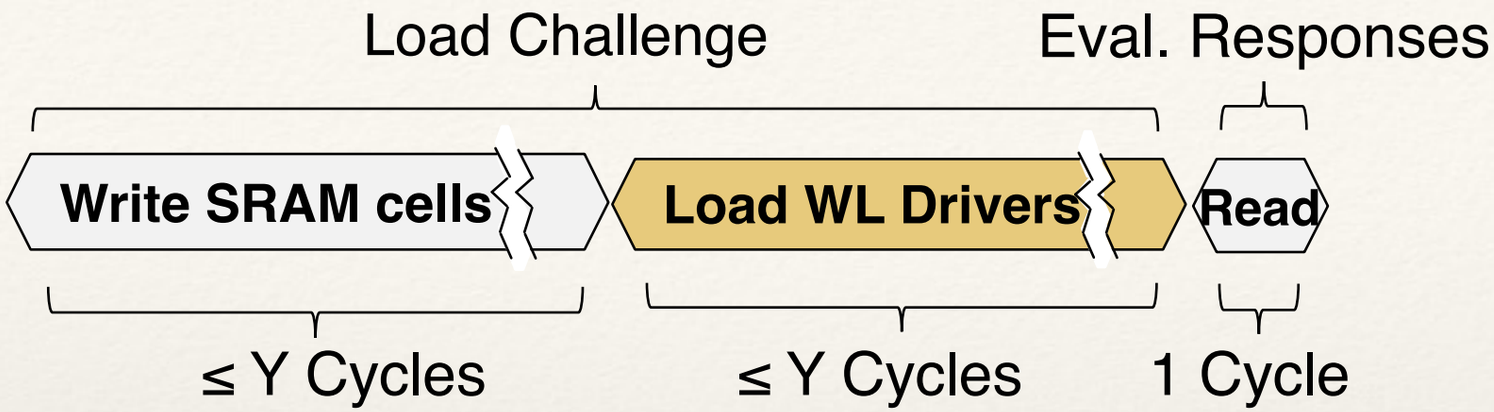


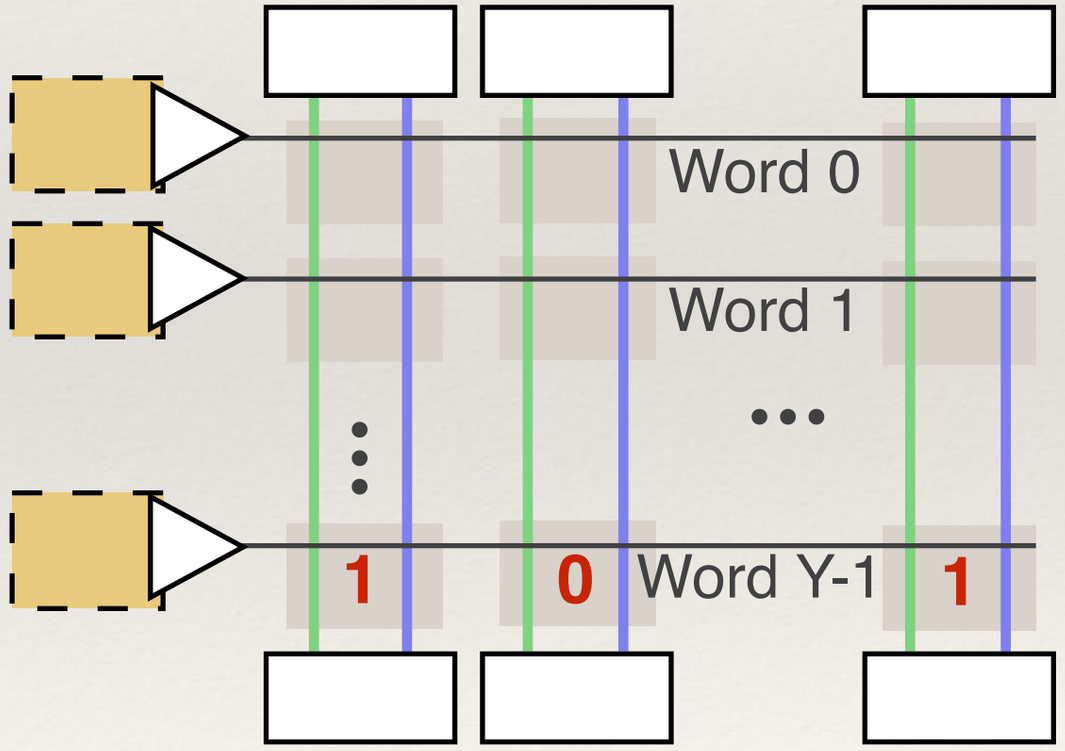
Challenge Response Pairs

- PUF Challenge:
  - 4^Y possible challenges (Y = num. rows)
  - For each cell in column:
    1. wordline on, cell value 0
    2. wordline on, cell value 1
    3. wordline off, cell value 0
    4. wordline off, cell value 1
- PUF Response:
  - Value read by sense amp of column(s)

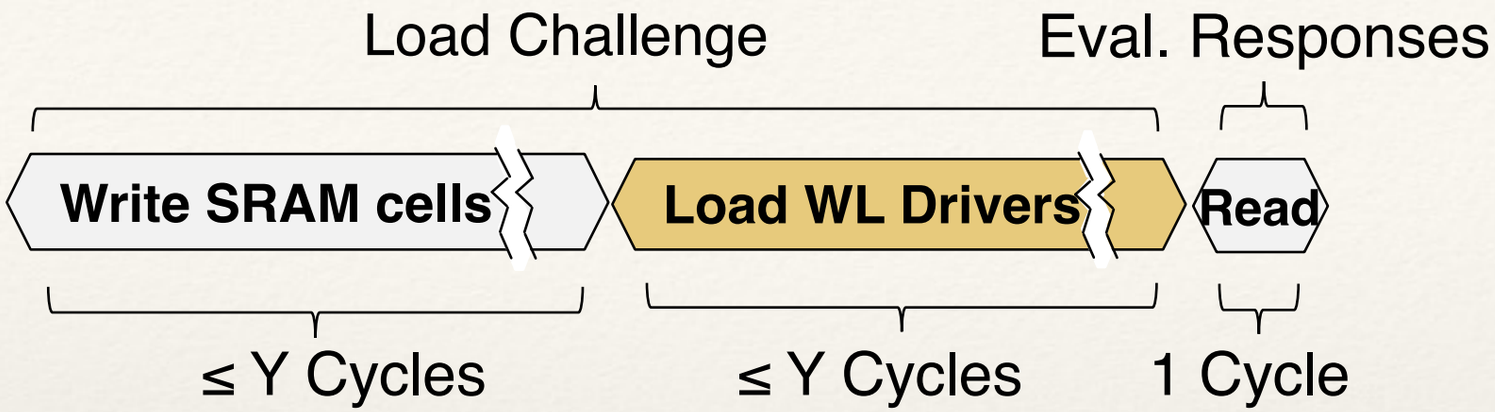


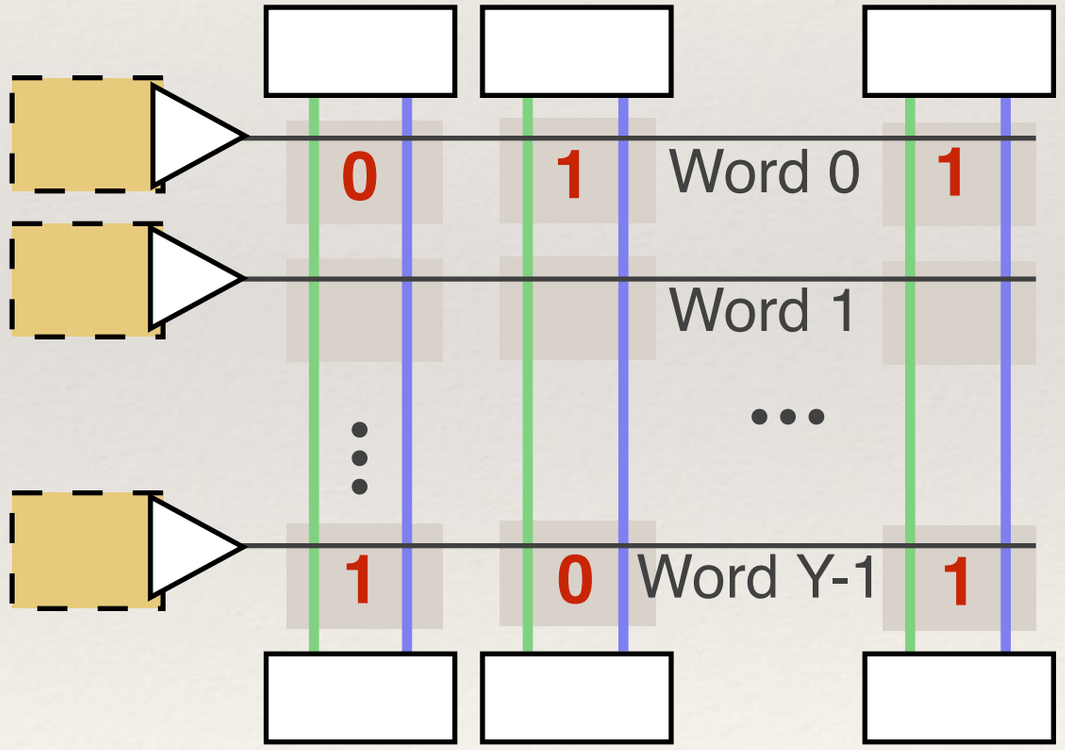
Performance and Overhead



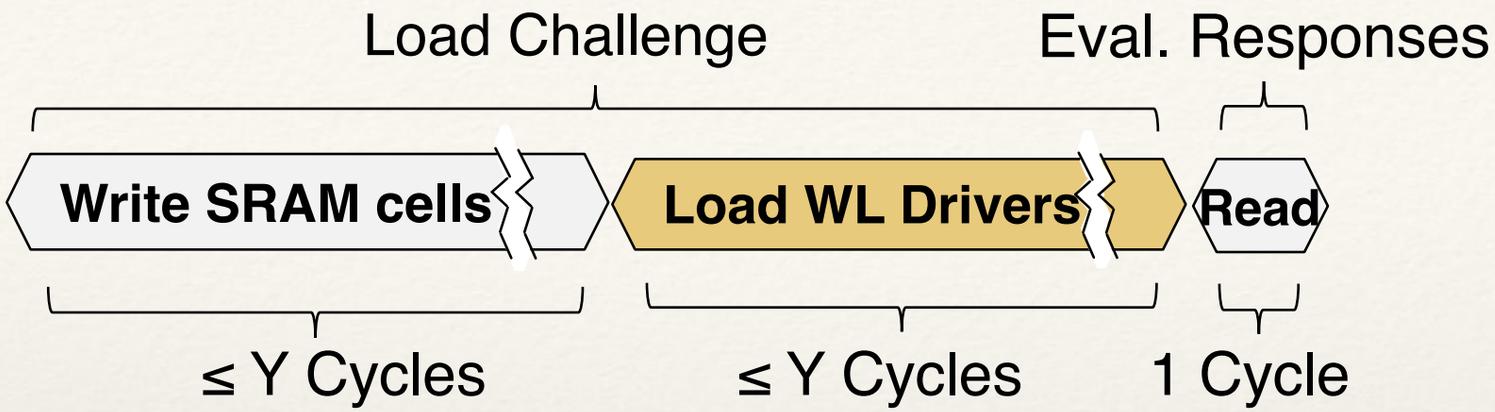


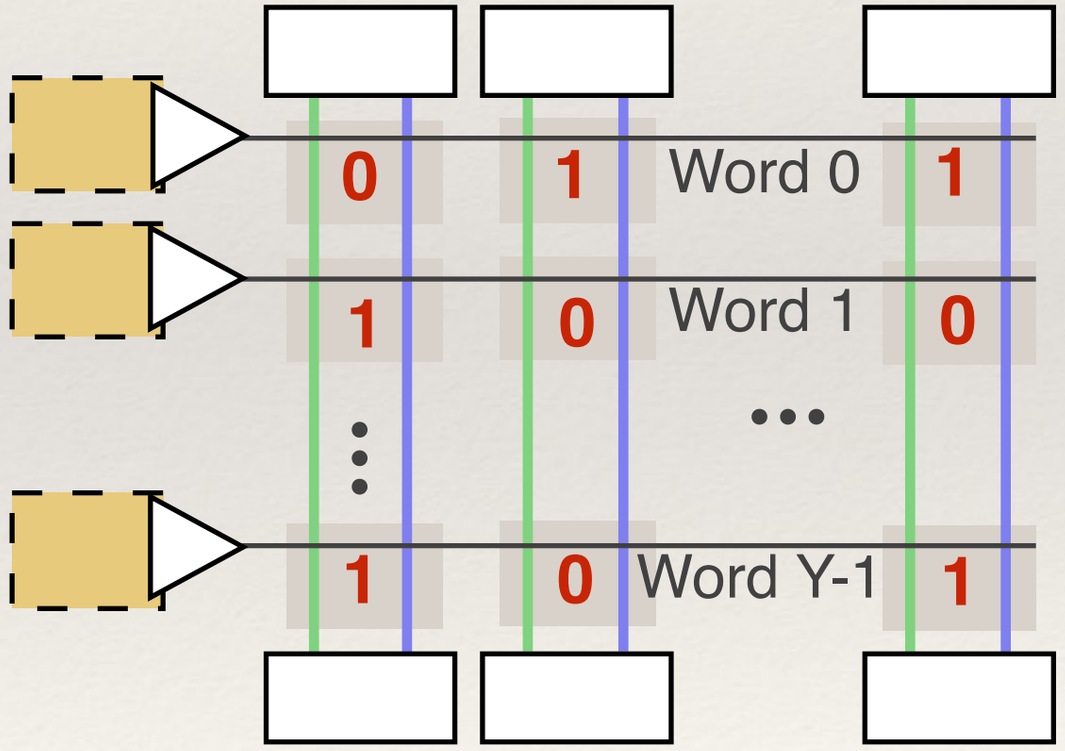
Performance and Overhead



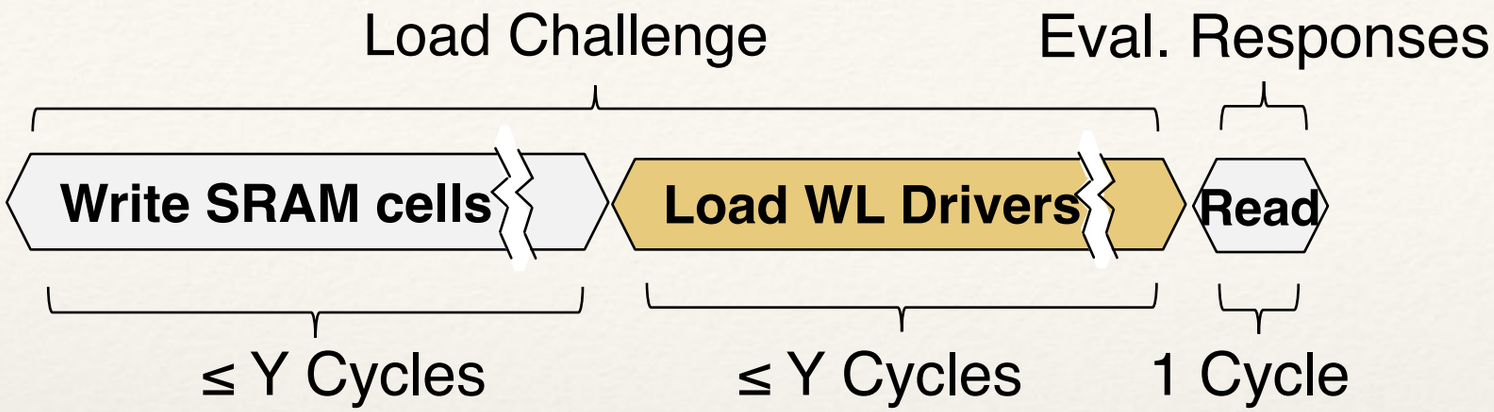


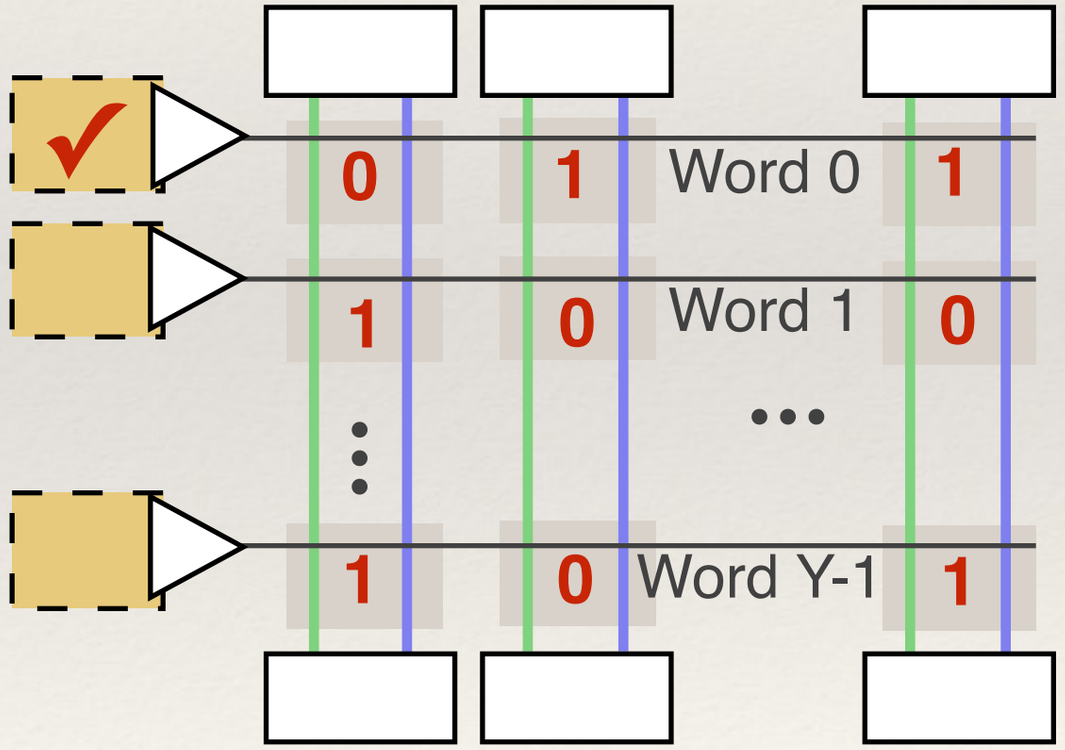
Performance and Overhead



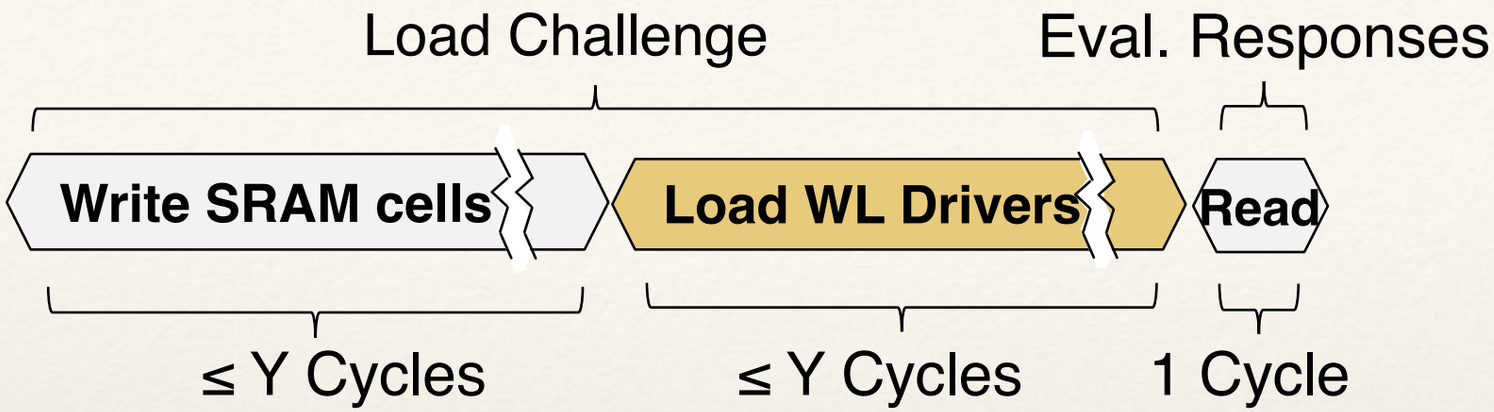


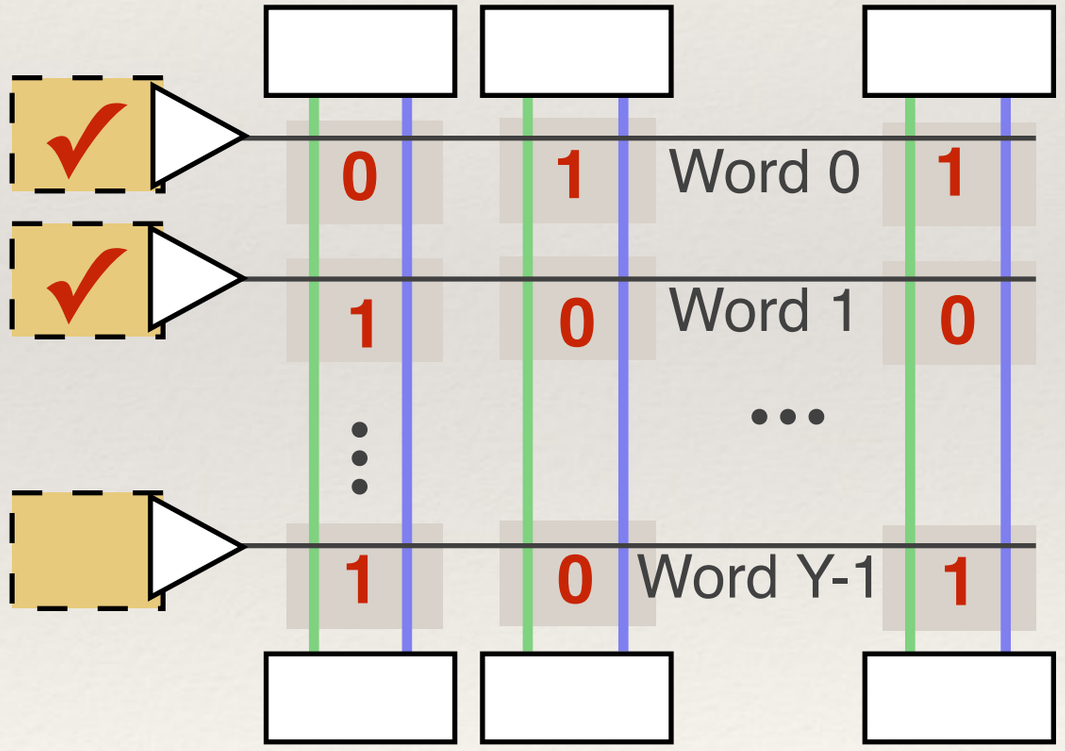
Performance and Overhead



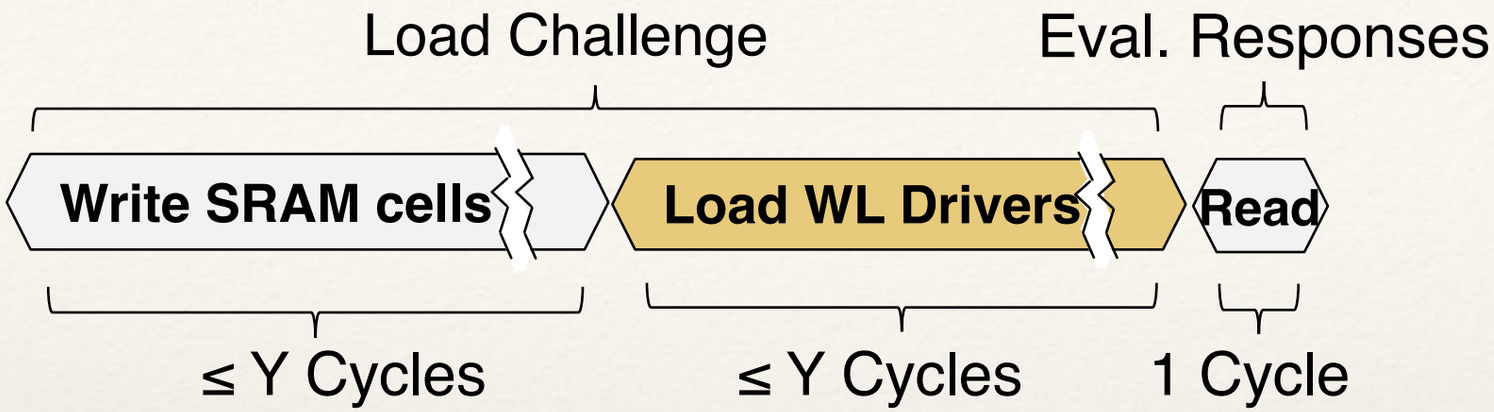


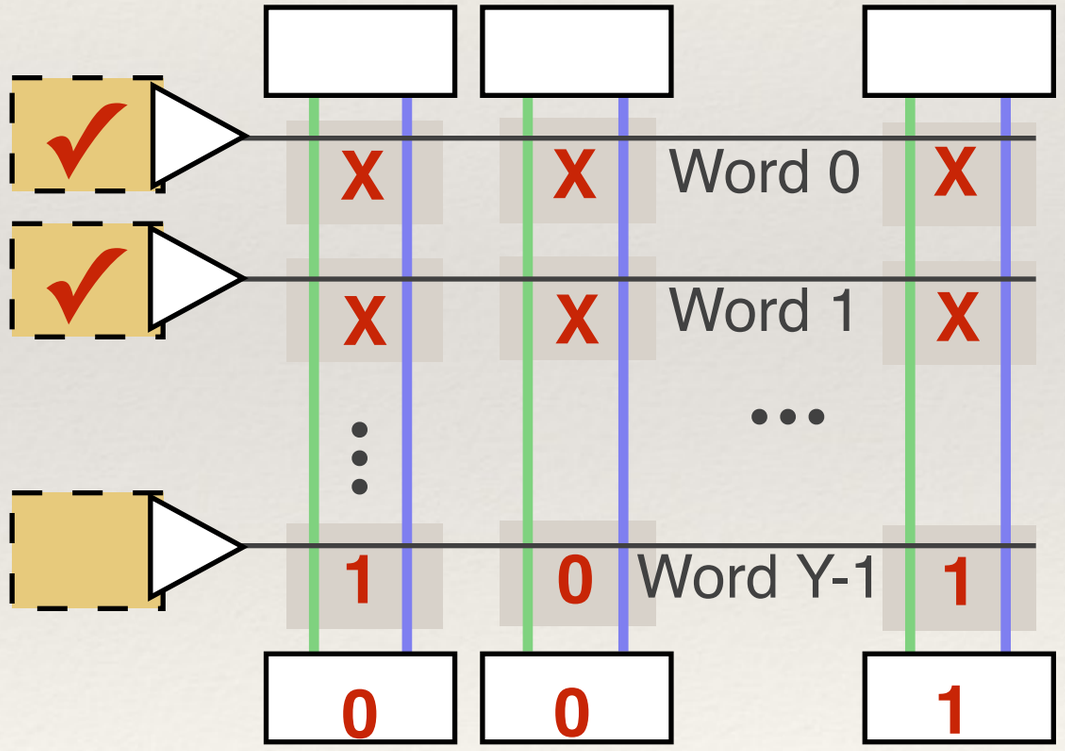
Performance and Overhead



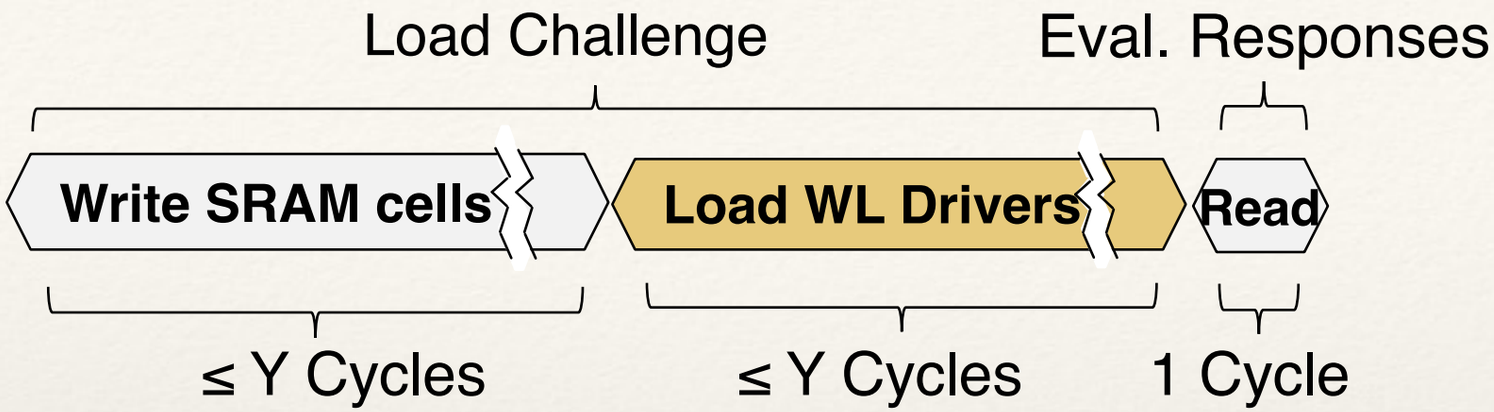


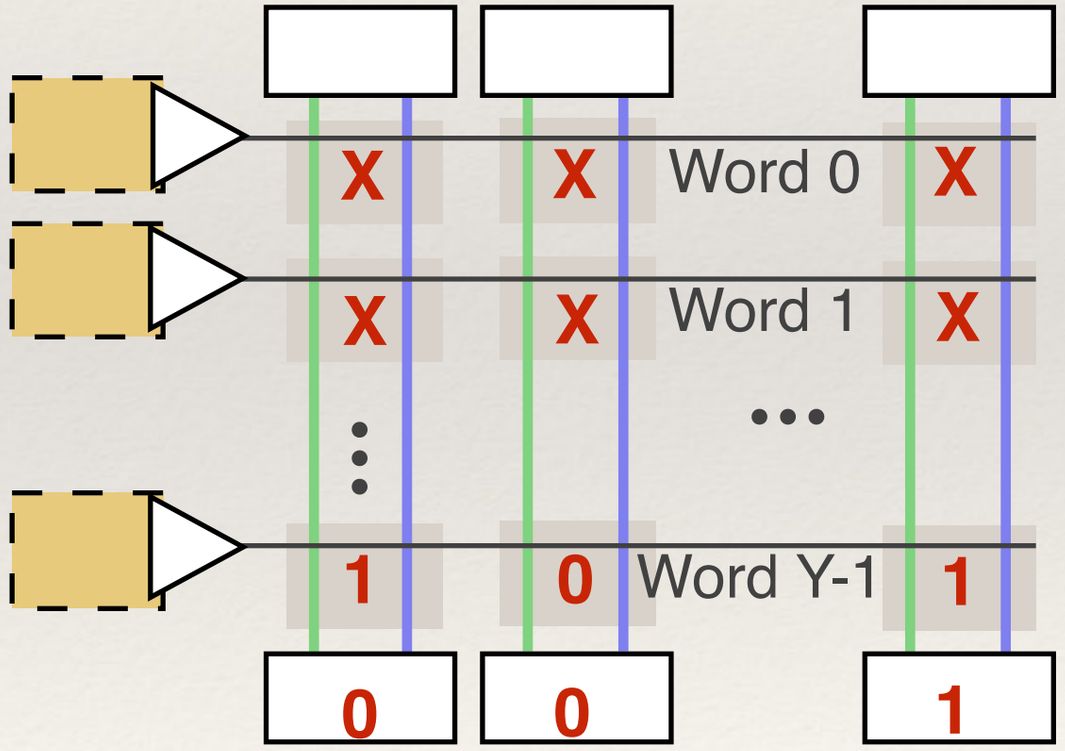
Performance and Overhead



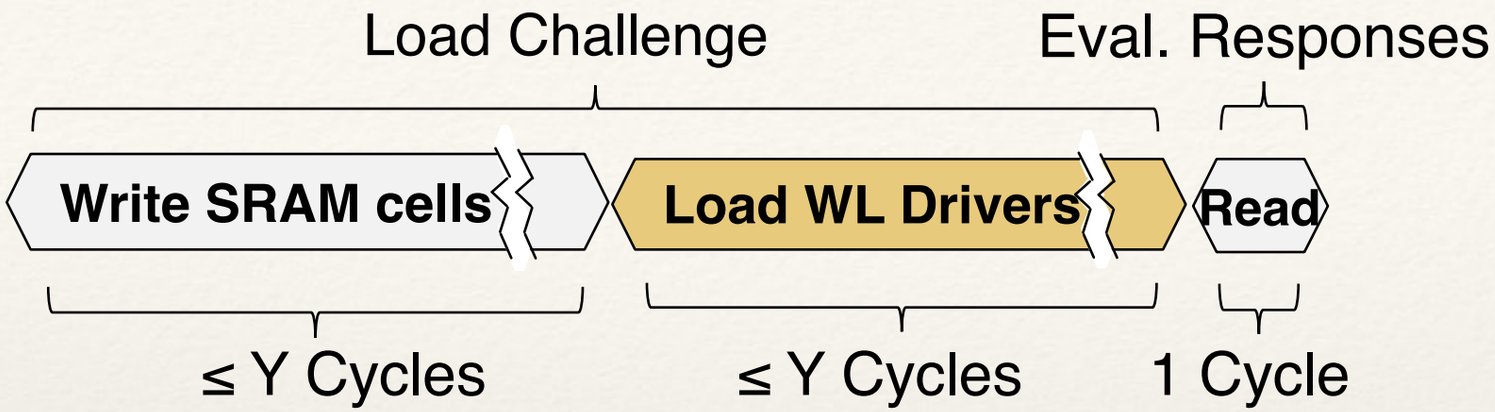


Performance and Overhead

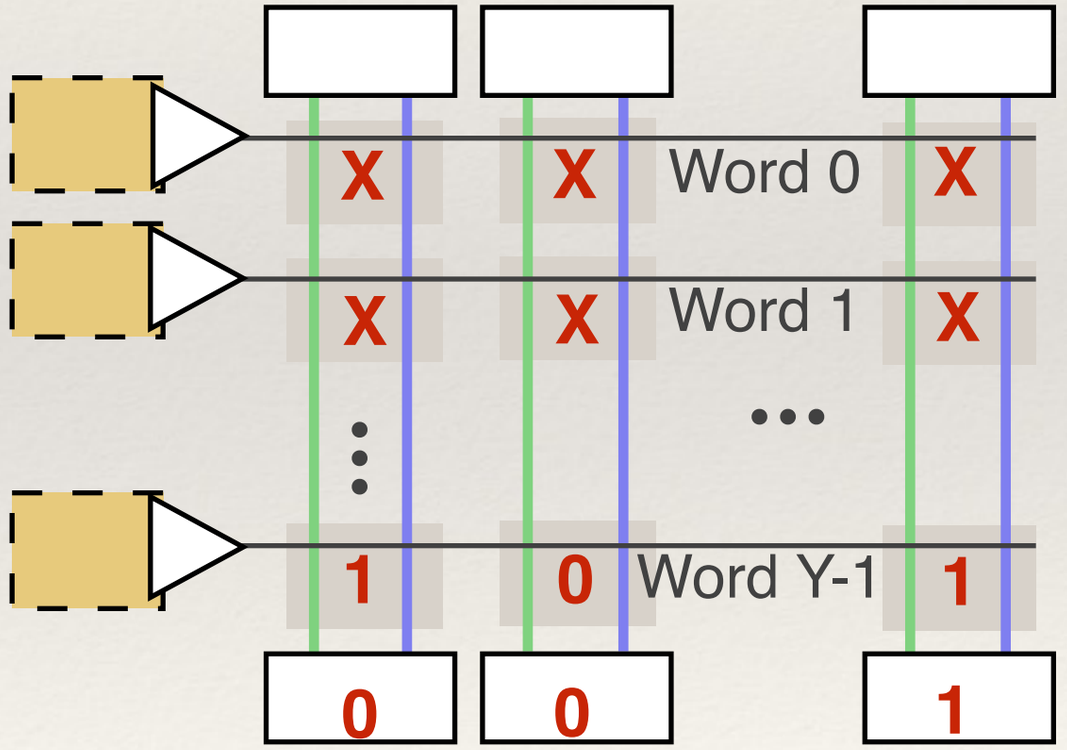




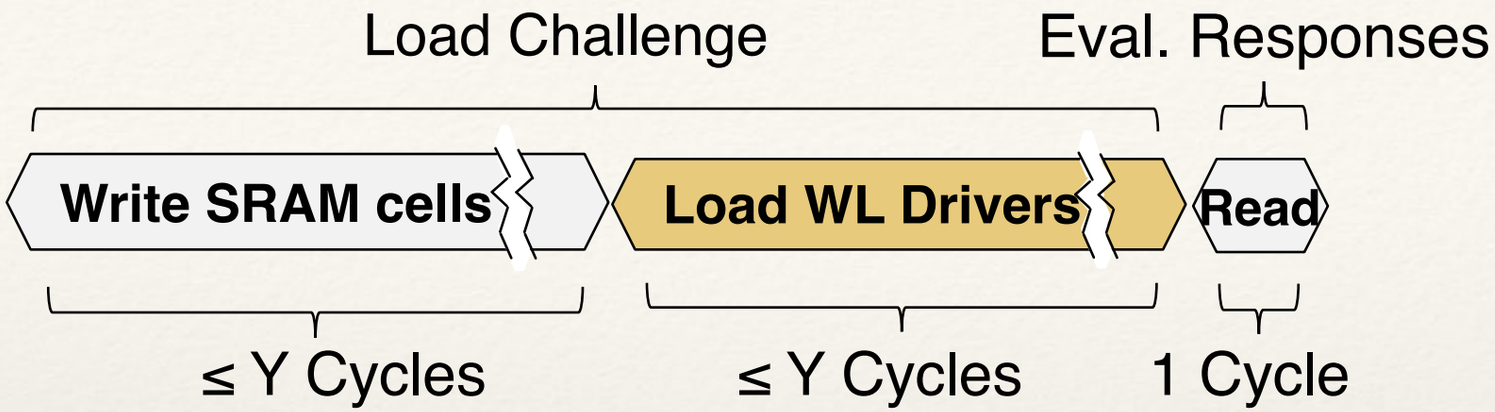
Performance and Overhead



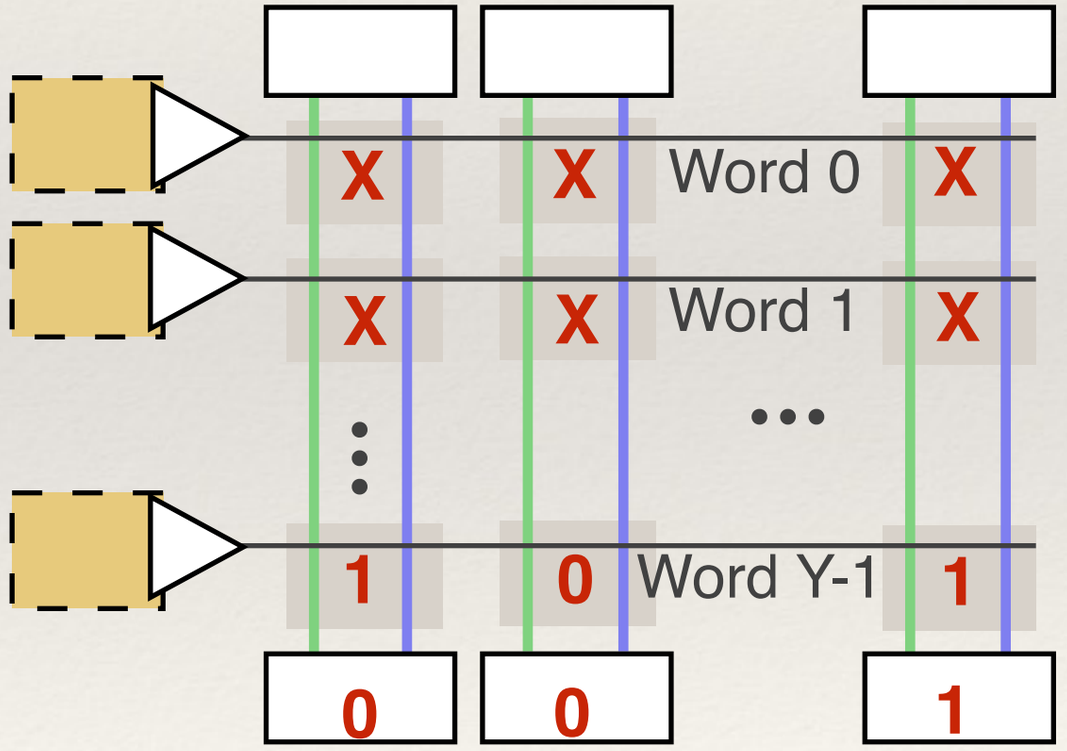
- Word-parallel (e.g. 256 columns)
- Response latency
  - 6 cycles for 256-bit response as shown
  - Depends on number of enabled rows



Performance and Overhead

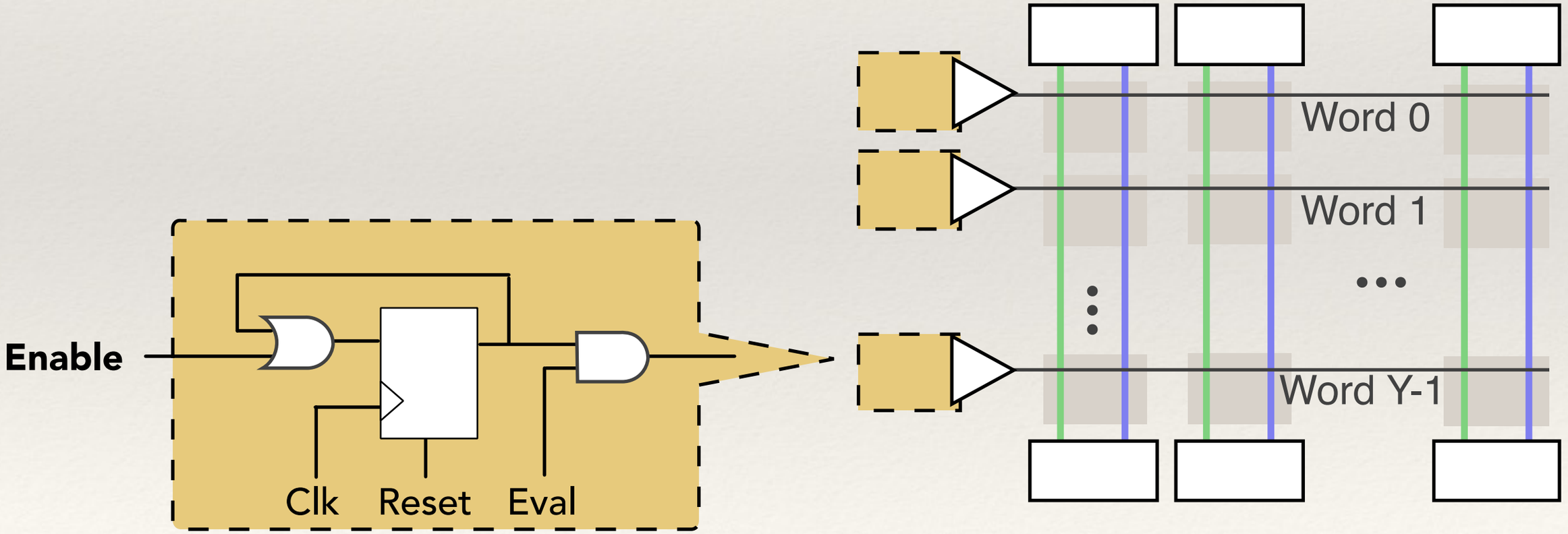


- Word-parallel (e.g. 256 columns)
- Response latency
  - 6 cycles for 256-bit response as shown
  - Depends on number of enabled rows
- Area overhead
  - A few extra gates per SRAM row
  - Don’t need to add circuitry on all rows



Integration

- Simple digital interface
- No power-cycling required
- Non-exclusive, SRAM rows still usable as memory when not used for PUF
- Does not upset stored data in non-used rows

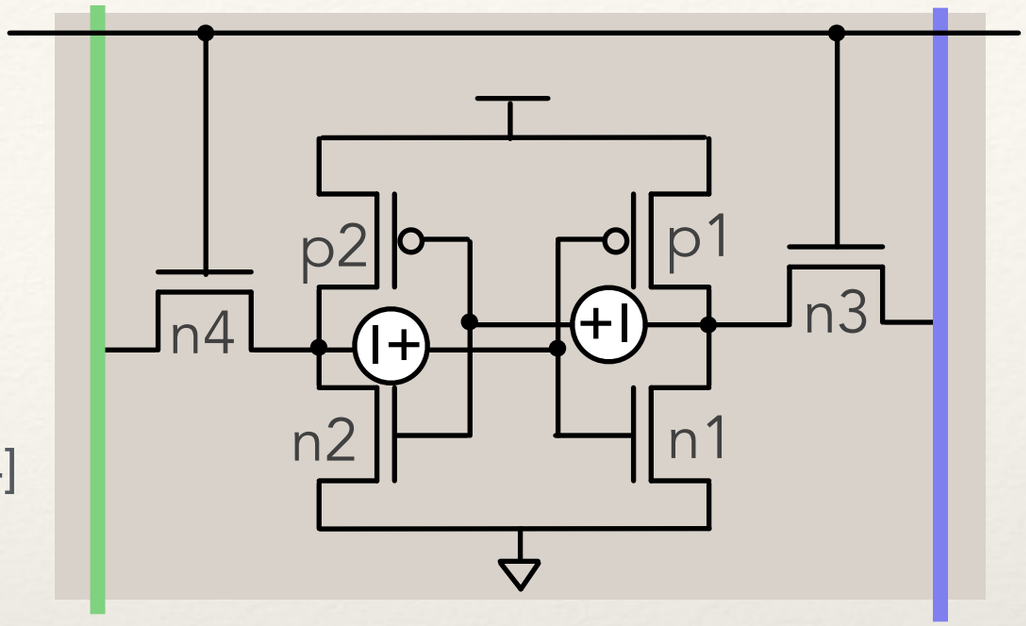


Outline

1. Introduction
   - PUFs
   - SRAM
   - Bitline PUF
2. **Evaluation**
   - **Uniqueness**
   - **Reliability**
   - **Modeling Attacks**
3. Summary and Related work

Methodology

- Circuit simulation using Ngspice
- Devices are 90nm Predictive Technology Model [1]
- Sizing according to Nii et al. [2]
- Variation: threshold voltage and channel length [3,4]
- Noise: between cross-coupled nodes [5]

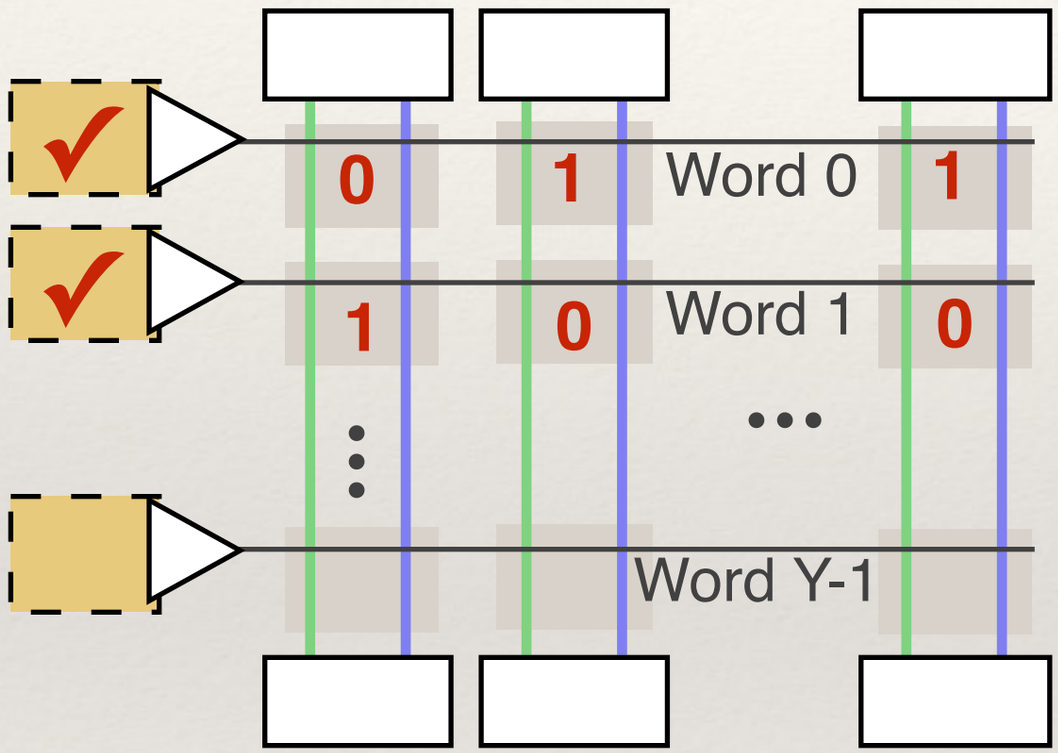


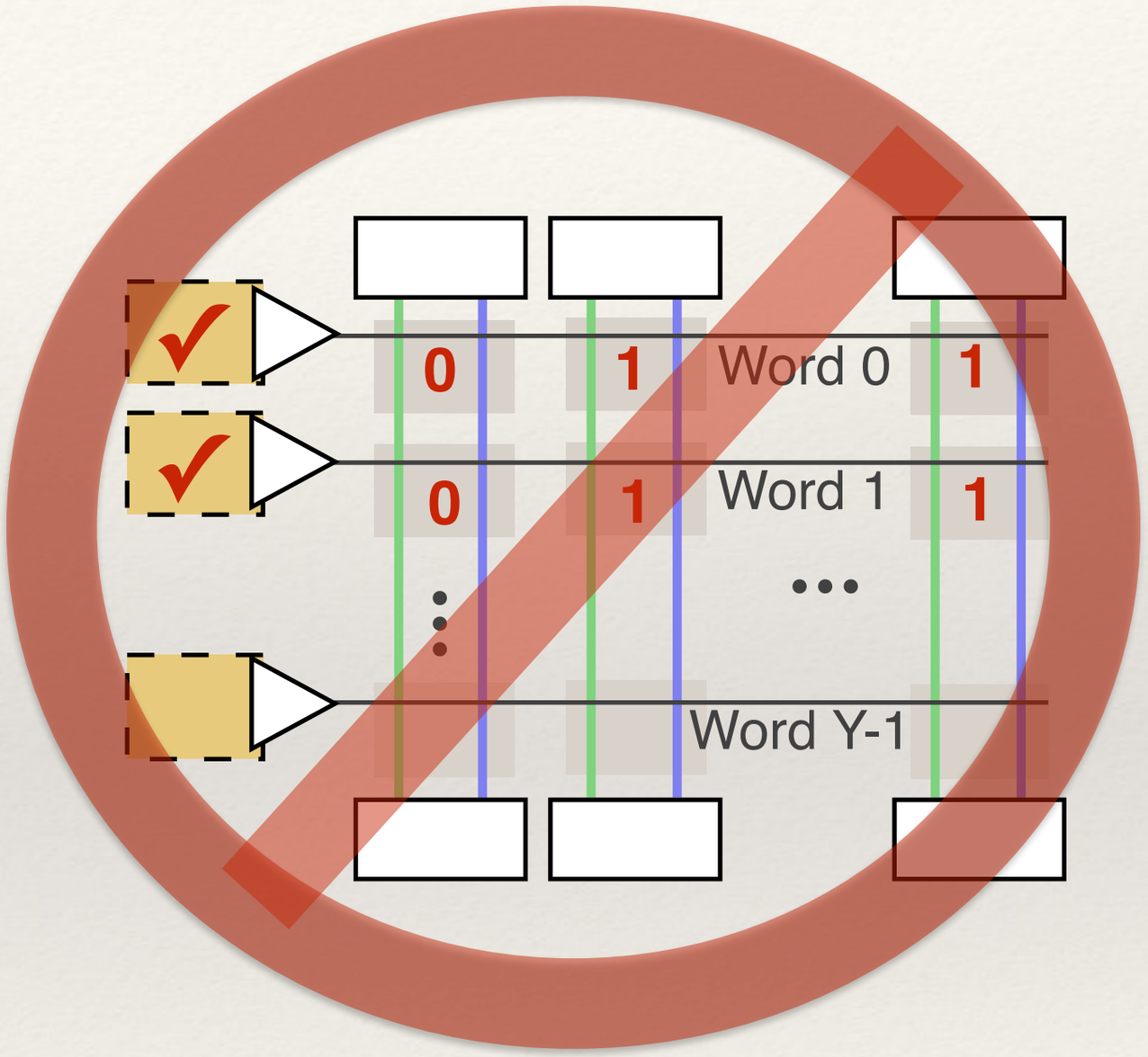
		Sizing		Process Variation			
				vth0 [mV]		lint [nm]	
		W [nm]	L [nm]	μ	σ	μ	σ
SRAM cell	n1,n2	200	90	397	13.4	7.5	3
	n3,n4	140	90	397	16.0	7.5	3
	p1,p2	140	90	-339	16.0	7.5	3
Sense Amp & Precharge	NMOS	1000	90	397	6.0	7.5	3
	PMOS	1000	90	-339	6.0	7.5	3

experiment code available online: https://github.com/danholcomb/bitline-puf

[1] Predictive Technology Model. 90nm NMOS and PMOS BSIM4 Models
[2] Nii et al., IEEE Journal of Solid State Circuits, 2004
[3] Pelgrom et al. IEEE Journal of Solid State Circuits, 1989
[4] Seevinck et al. IEEE Journal of Solid State Circuits, 1987
[5] Anis et al. Workshop on System-on-Chip for Real-Time Applications, 2005

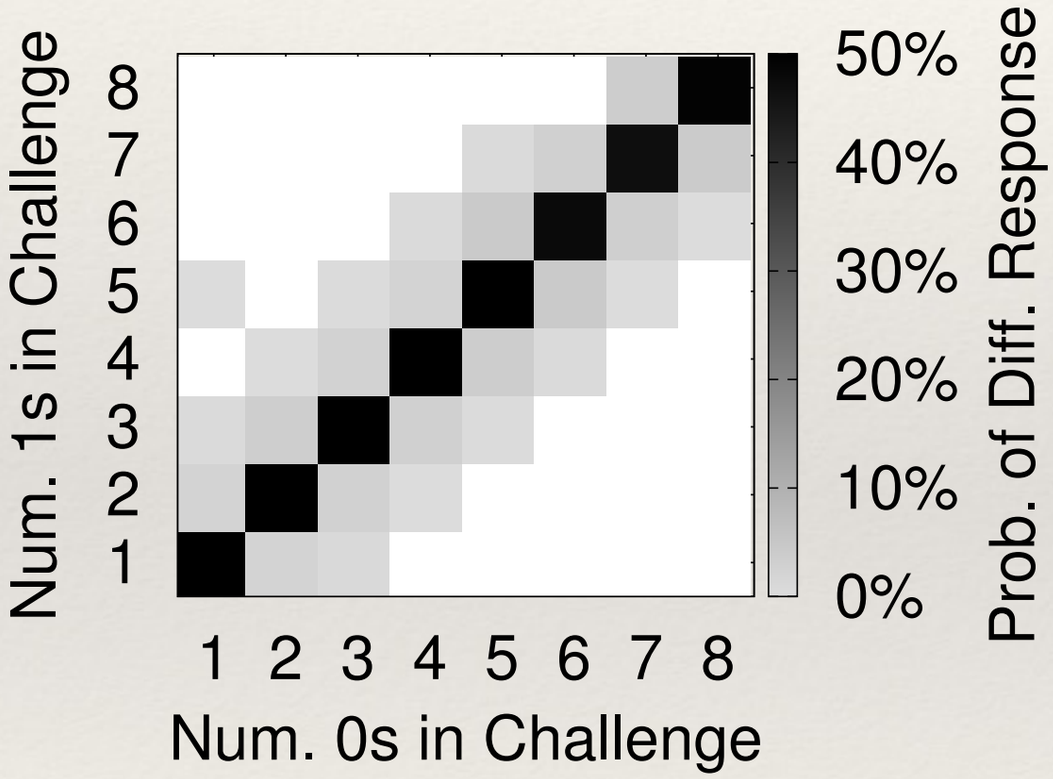
Choosing Useful Challenges

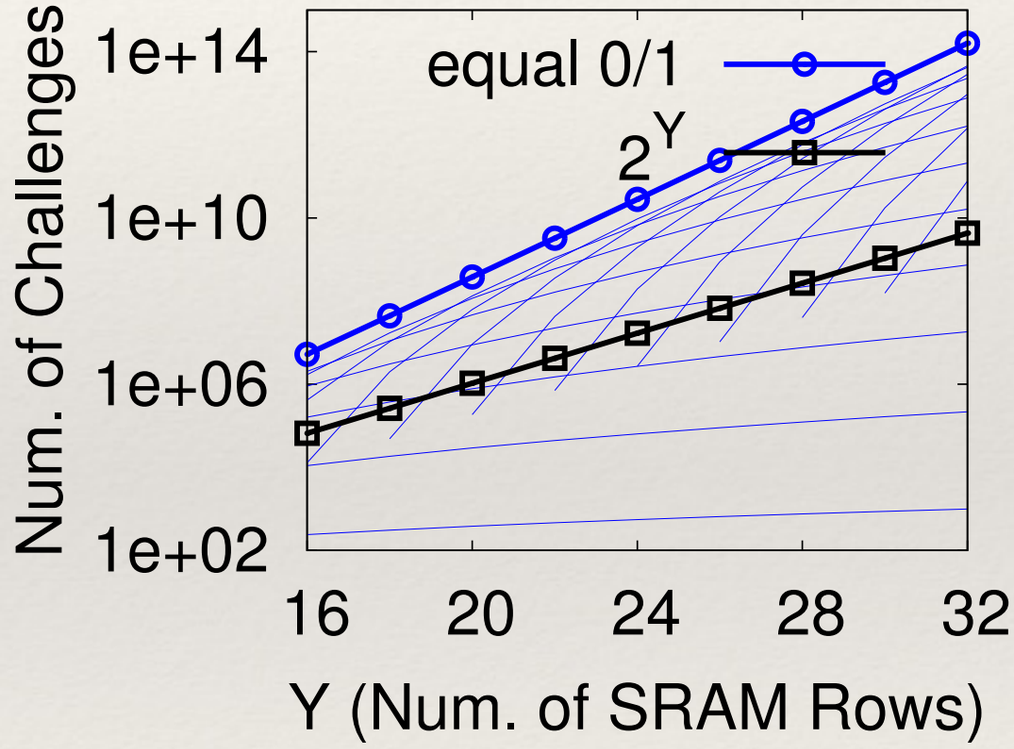




Choosing Useful Challenges

- Useful challenges have equal number of 0s and 1s
- Exponential subset of the 4^Y possible challenges

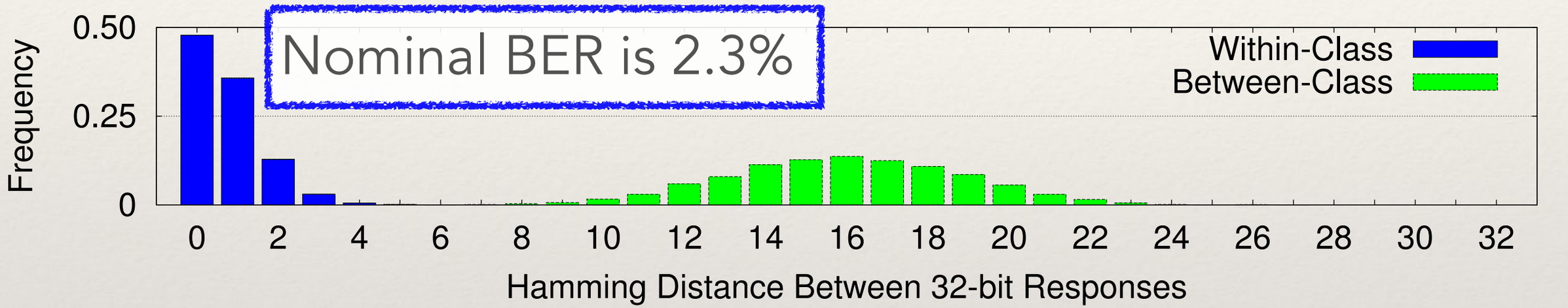




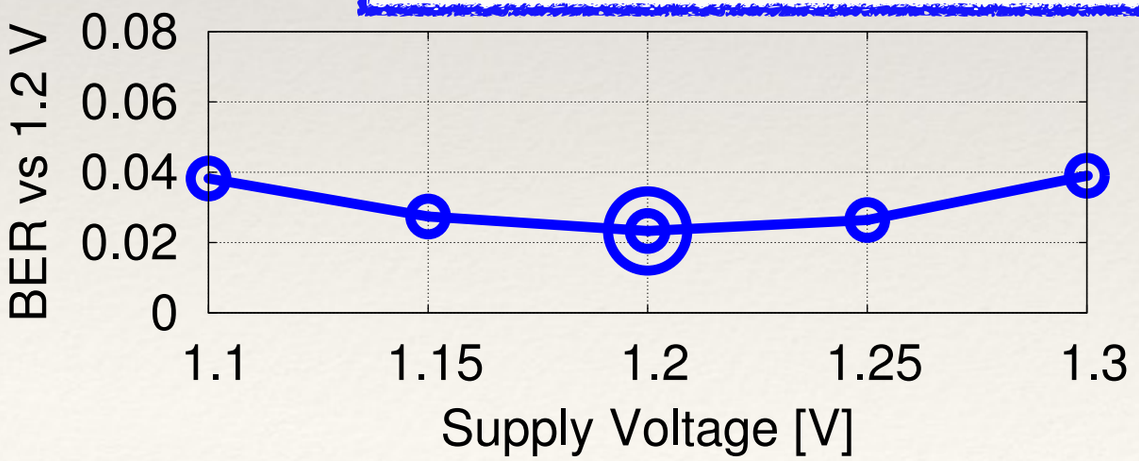
(Asymmetric designs may have different useful challenges)

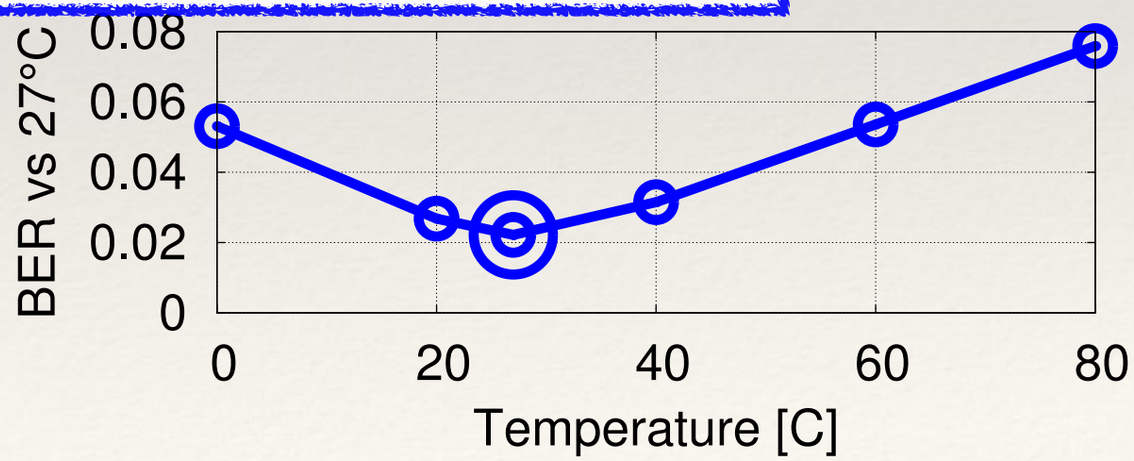
Uniqueness and Reliability

- Applying random challenges with equal number 0s and 1s
- Nominal conditions: 1.2V and 27°C



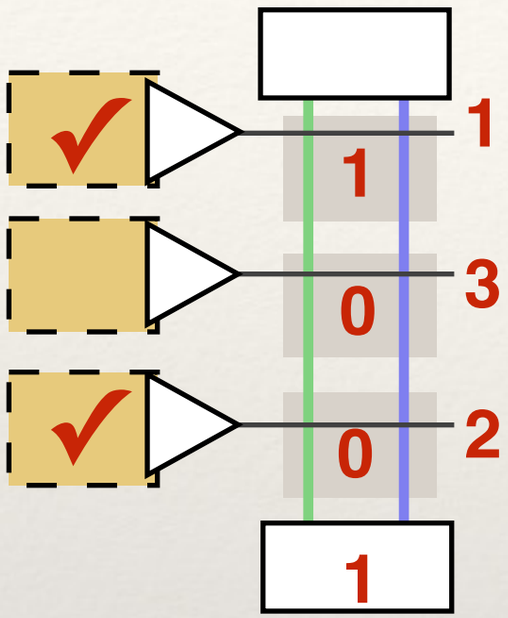
BER ≤ 7.6% across voltage and temperature





Modeling Attacks

- Can a model predict Bitline PUF's responses? **(Yes)**



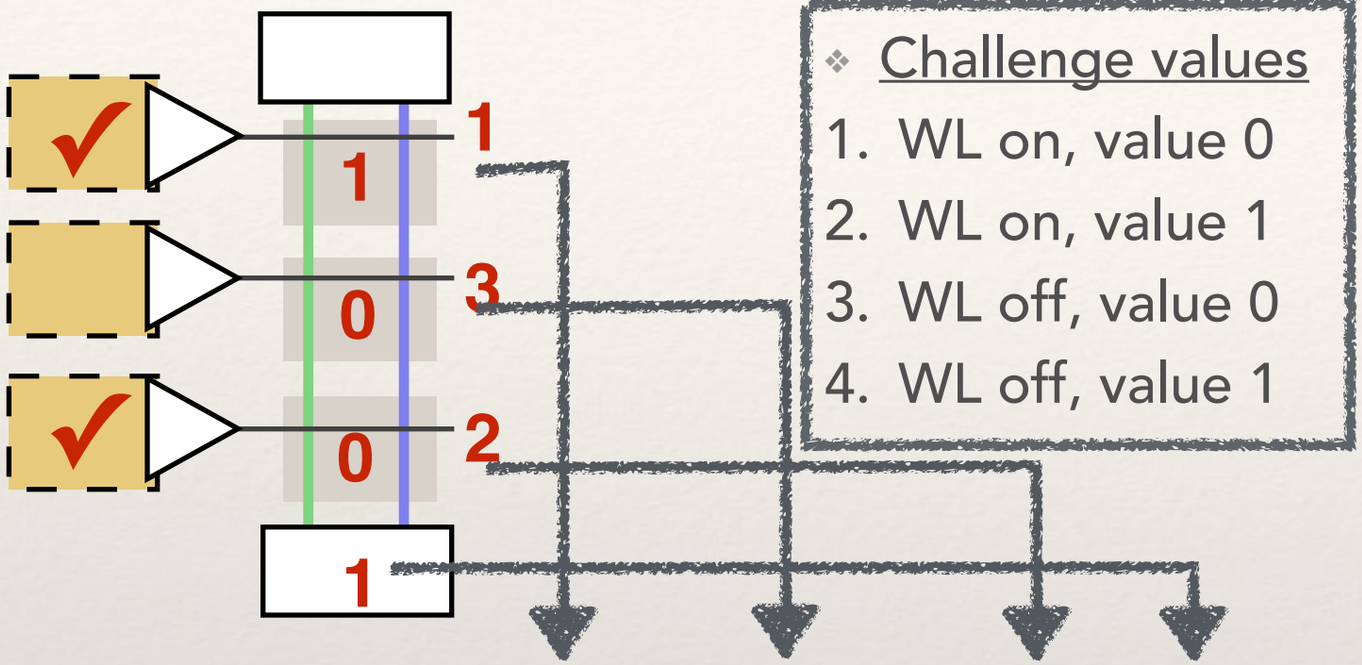
- Challenge values

1. WL on, value 0
2. WL on, value 1
3. WL off, value 0
4. WL off, value 1

[1] Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, 1999

Modeling Attacks

- Can a model predict Bitline PUF's responses? **(Yes)**



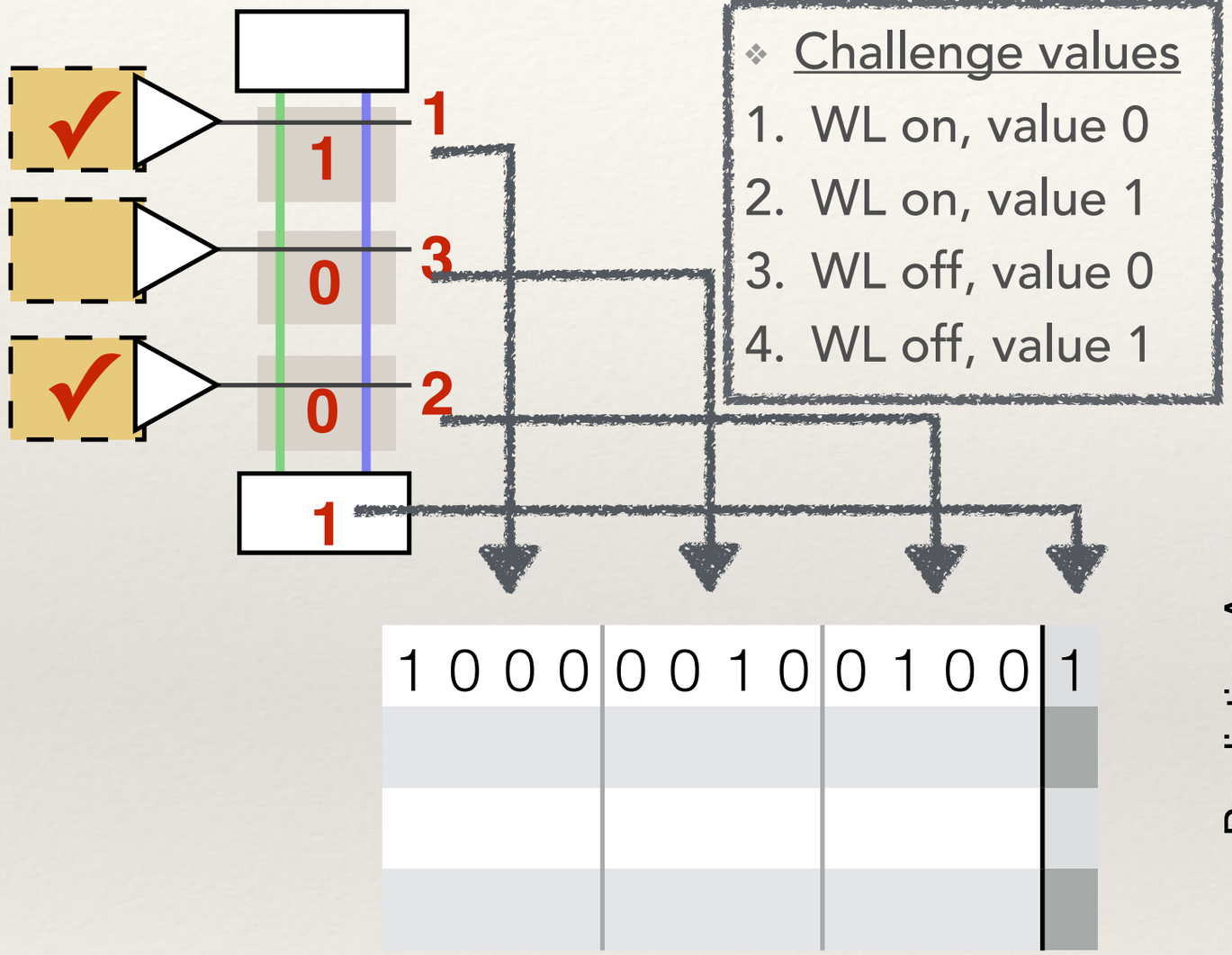
- Challenge values
1. WL on, value 0
2. WL on, value 1
3. WL off, value 0
4. WL off, value 1

1	0	0	0	0	0	1	0	0	1	0	0	1

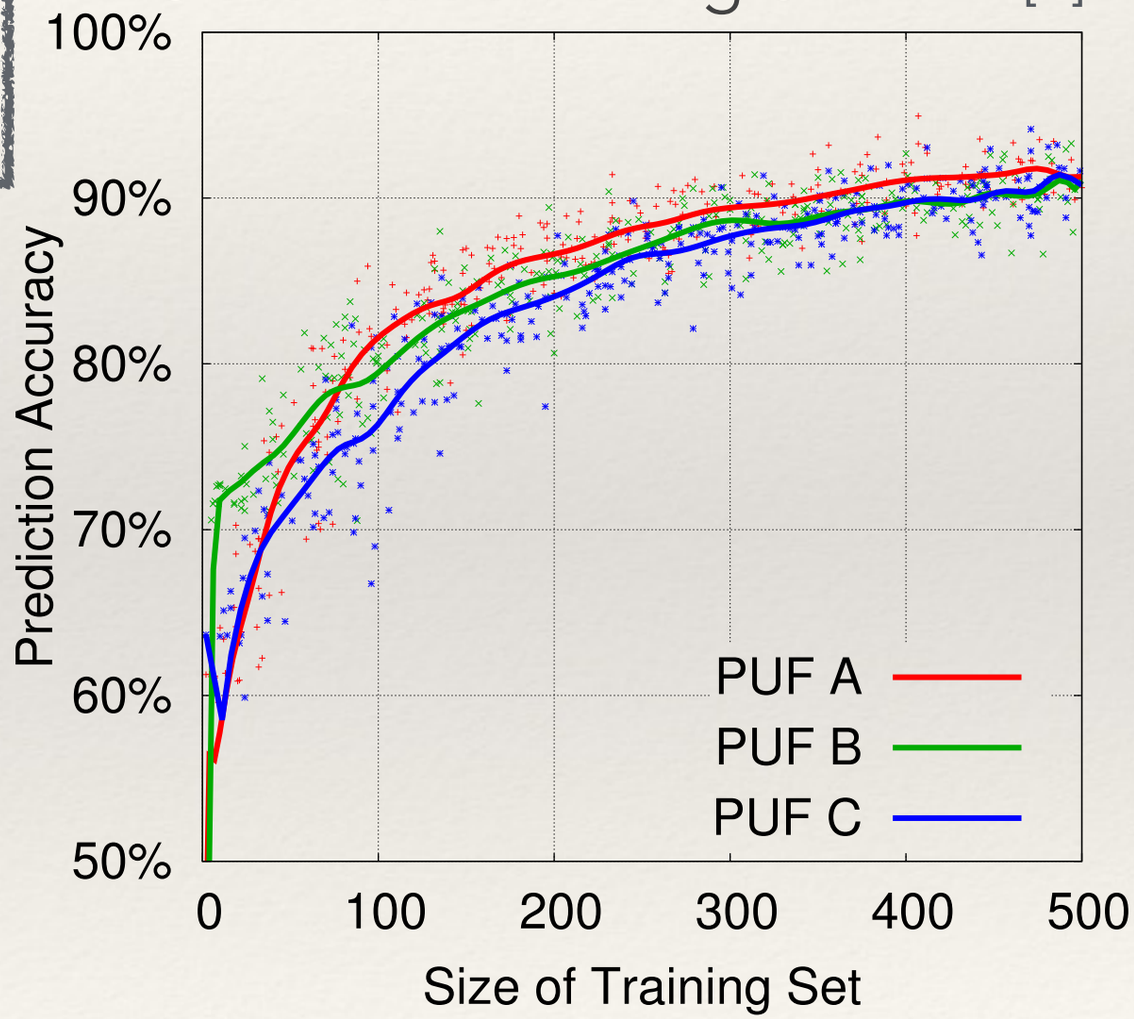
[1] Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, 1999

Modeling Attacks

- Can a model predict Bitline PUF's responses? **(Yes)**



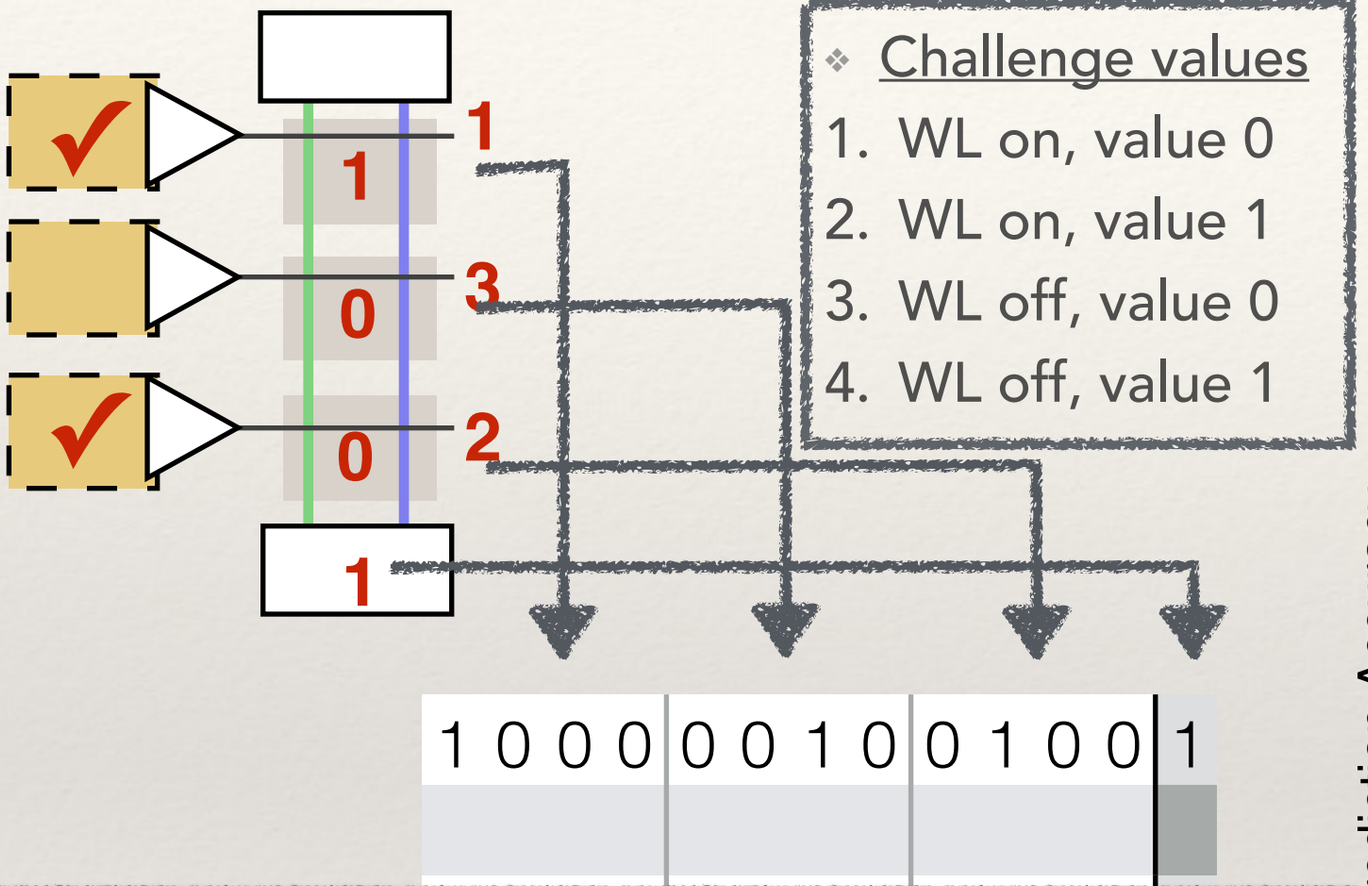
Classification using SVM^{light} [1]



[1] Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, 1999

Modeling Attacks

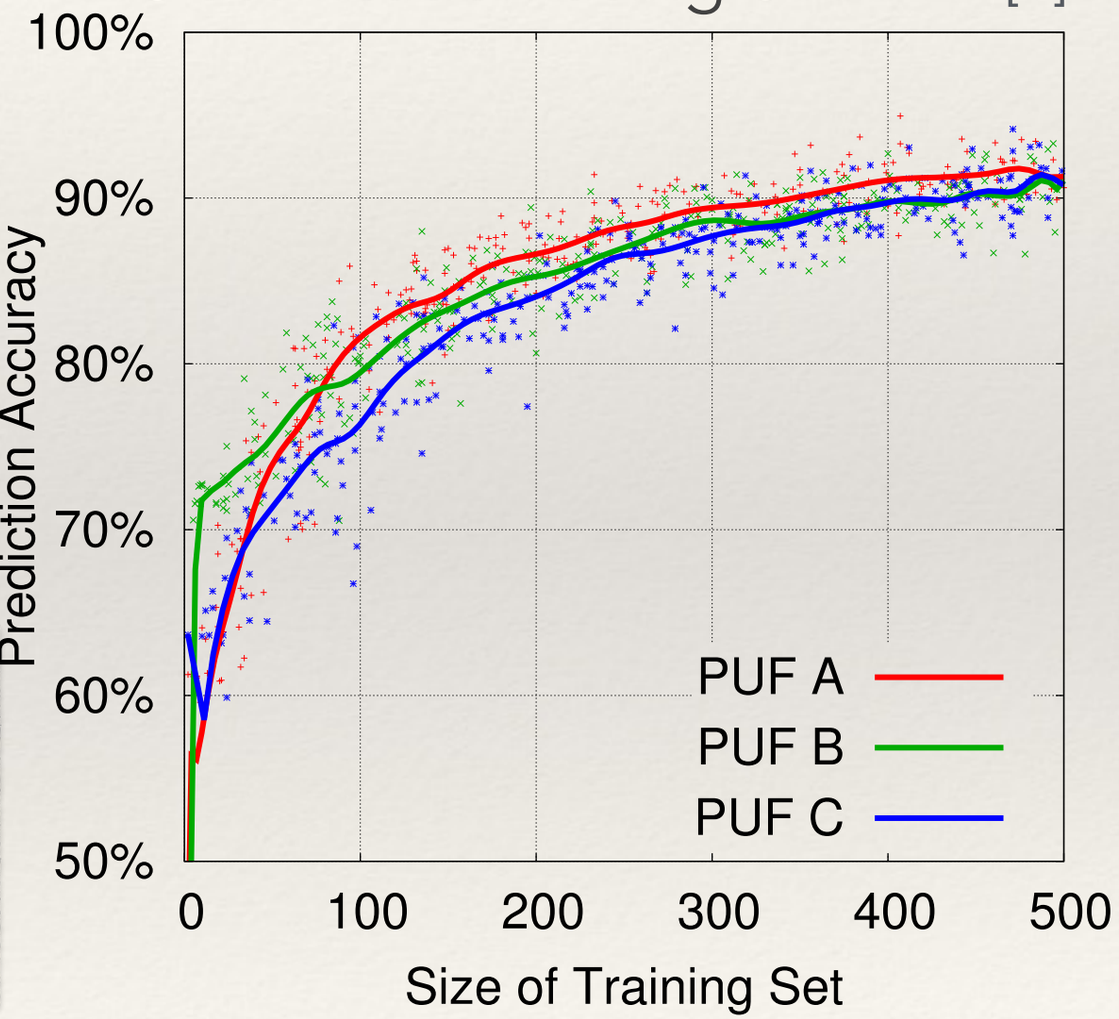
- Can a model predict Bitline PUF's responses? **(Yes)**



- Challenge values
  1. WL on, value 0
  2. WL on, value 1
  3. WL off, value 0
  4. WL off, value 1

- CRPs must be obfuscated as usual (Mission Impossible?)

Classification using SVM^{light} [1]



[1] Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, 1999

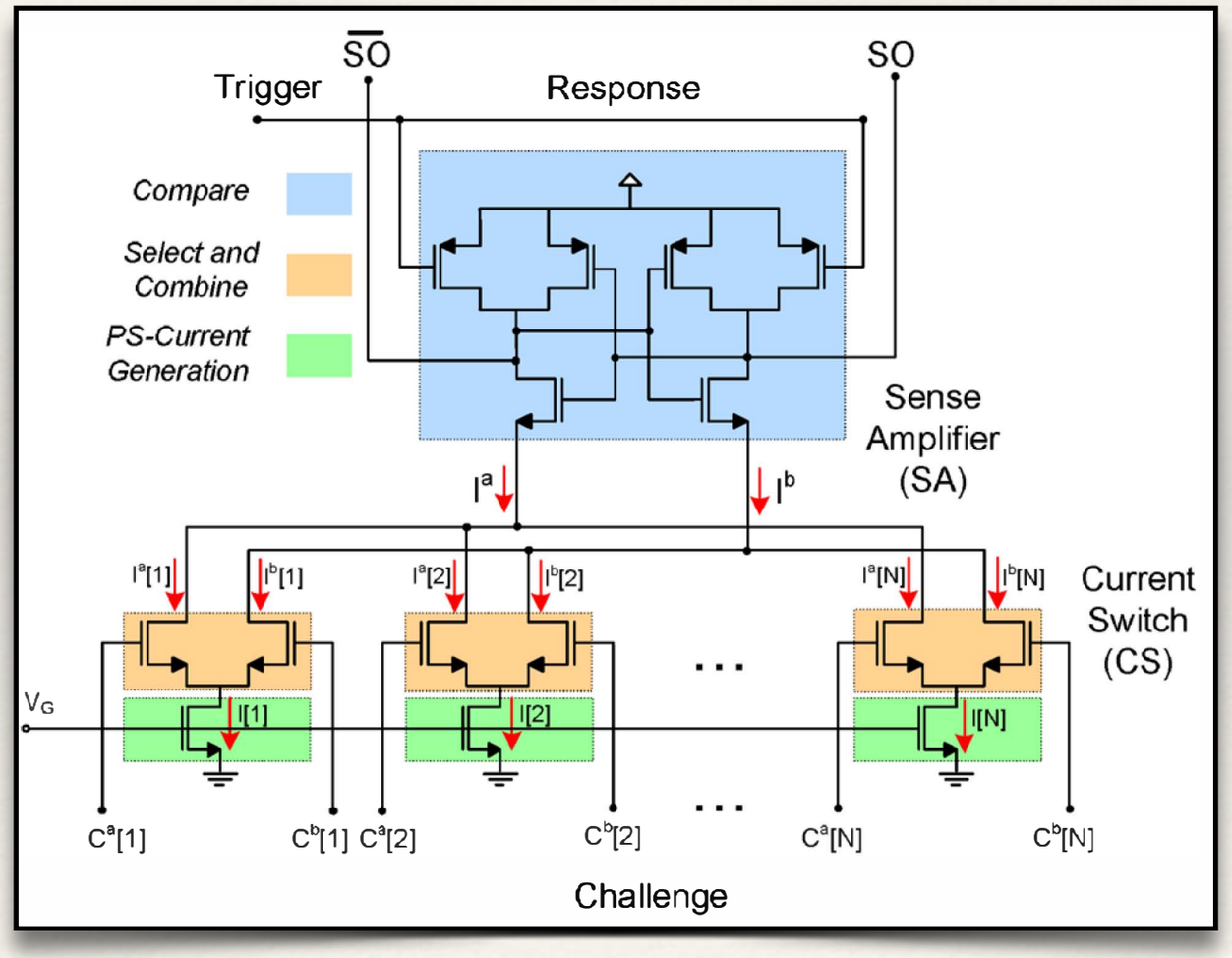
Outline

1. Introduction
   - PUFs
   - SRAM
   - Bitline PUF
2. Evaluation
   - Uniqueness
   - Reliability
   - Modeling Attacks
3. **Summary and Related work**

Related Work

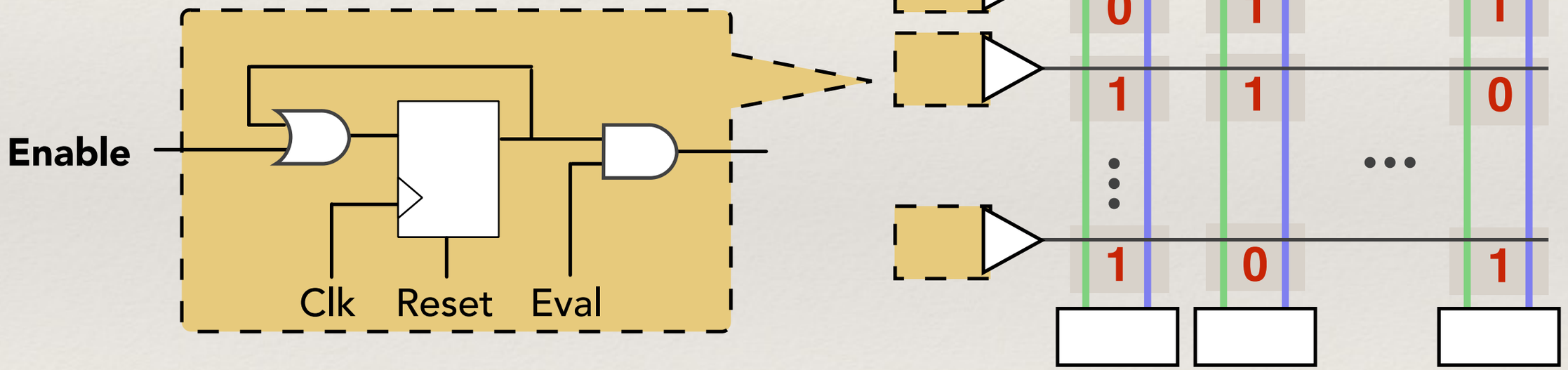
- SRAM PUFs [1,2,3]
- Bistable-ring PUF [4]
- Low-power current PUF [5]

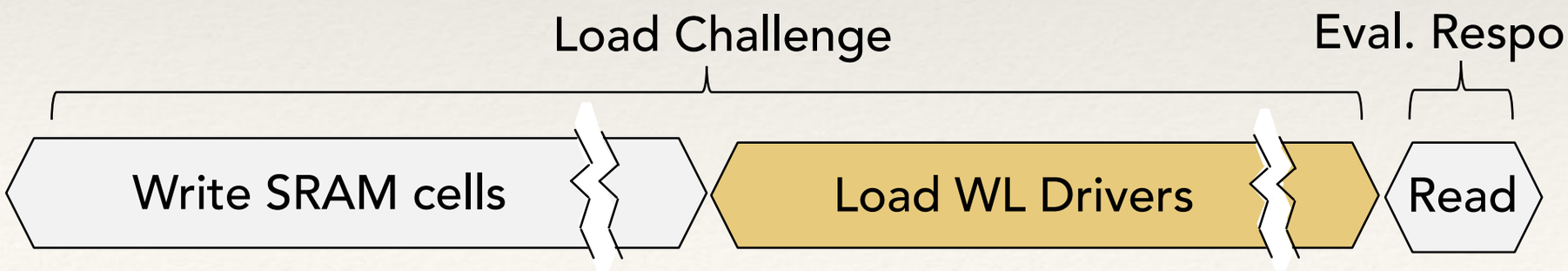
[1] Guajardo et al. CHES 2007
[2] Holcomb et al. T-Comp 2009
[3] Zheng et al. DAC 2013
[4] Chen et al. HOST 2011
[5] Majzoobi et al. ISCAS 2011



Bitline PUF: Summary

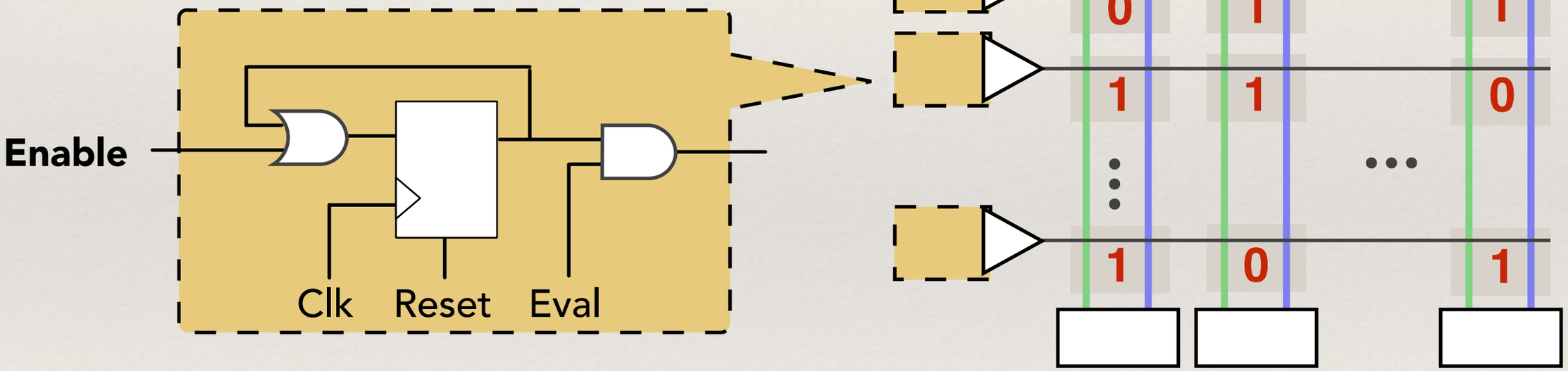
- **Modifying wordline drivers of SRAM array creates a new PUF with desirable properties**
  - Challenge-response operation
  - Low area overhead
  - Simple





Bitline PUF: Questions?

- **Modifying wordline drivers of SRAM array creates a new PUF with desirable properties**
  - Challenge-response operation
  - Low area overhead
  - Simple



Backup: Power Consumption