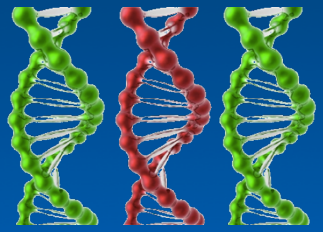
Robust Systems
From Today to the *N3XT 1,000X*

Subhasish Mitra

Department of EE & Department of CS
Stanford University

World Relies on Computing

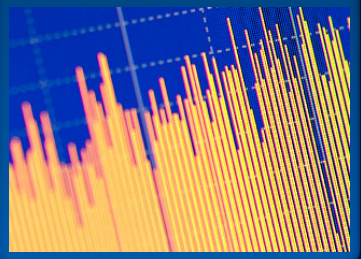
Genomics



Security

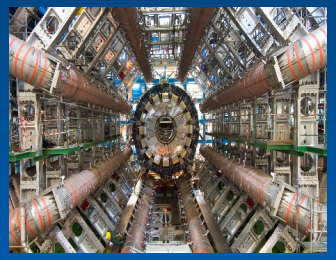
Smart Cities

Finance



Military

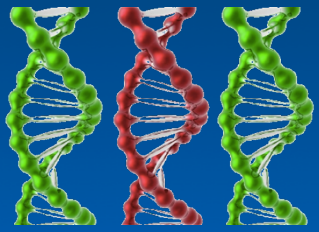
Science



Computational demands *exceed* Processing capability

World Relies on Computing

Genomics

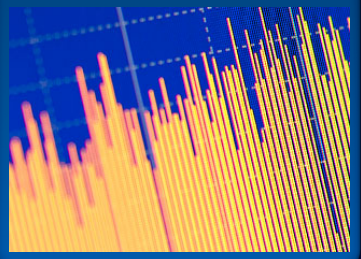


Smart Cities

- Ensure robust operation
- Meet computation demands
- New application horizon

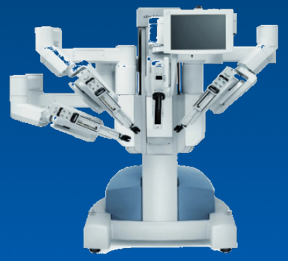
Security

Finance



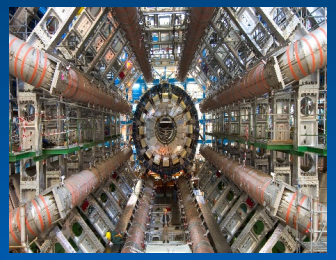
Military

Health Care



Government

Science



Research Topics

- Robust operation
  - Bugs, reliability, security
- Revolutionize nanosystems
  - 1,000X opportunity
- Program human brain
  - SNI Big Ideas in Neuroscience initiative

Outline

- Robust operation: silicon CMOS reliability
- Beyond silicon
- Conclusion

Silicon CMOS Reliability Challenges

- Radiation-induced soft errors
  - **Fatal** flip-flop errors
- Early-life failures (ELF)
  - Burn-in: difficult, expensive
- Variations: V_{dd}, thermal, circuit aging
  - Worst-case guardbands expensive

Definitions

- Malfunction (often referred to as failure)
  - Deviation from specified behavior
  - Underlying cause: failure
- Error: incorrect signal value
- Fault model
  - (Logic) representation of effect of failure

System Output Response to Failure

- Error on output: non-critical apps. (e.g., games ?)
- Fault-secure: correct outputs or error indication
  - Retry adequate (e.g., banks)
- Fault masked: correct outputs
  - Fault in specified class (e.g., spacecraft)
- Fail safe: correct or “safe” outputs

Definitions

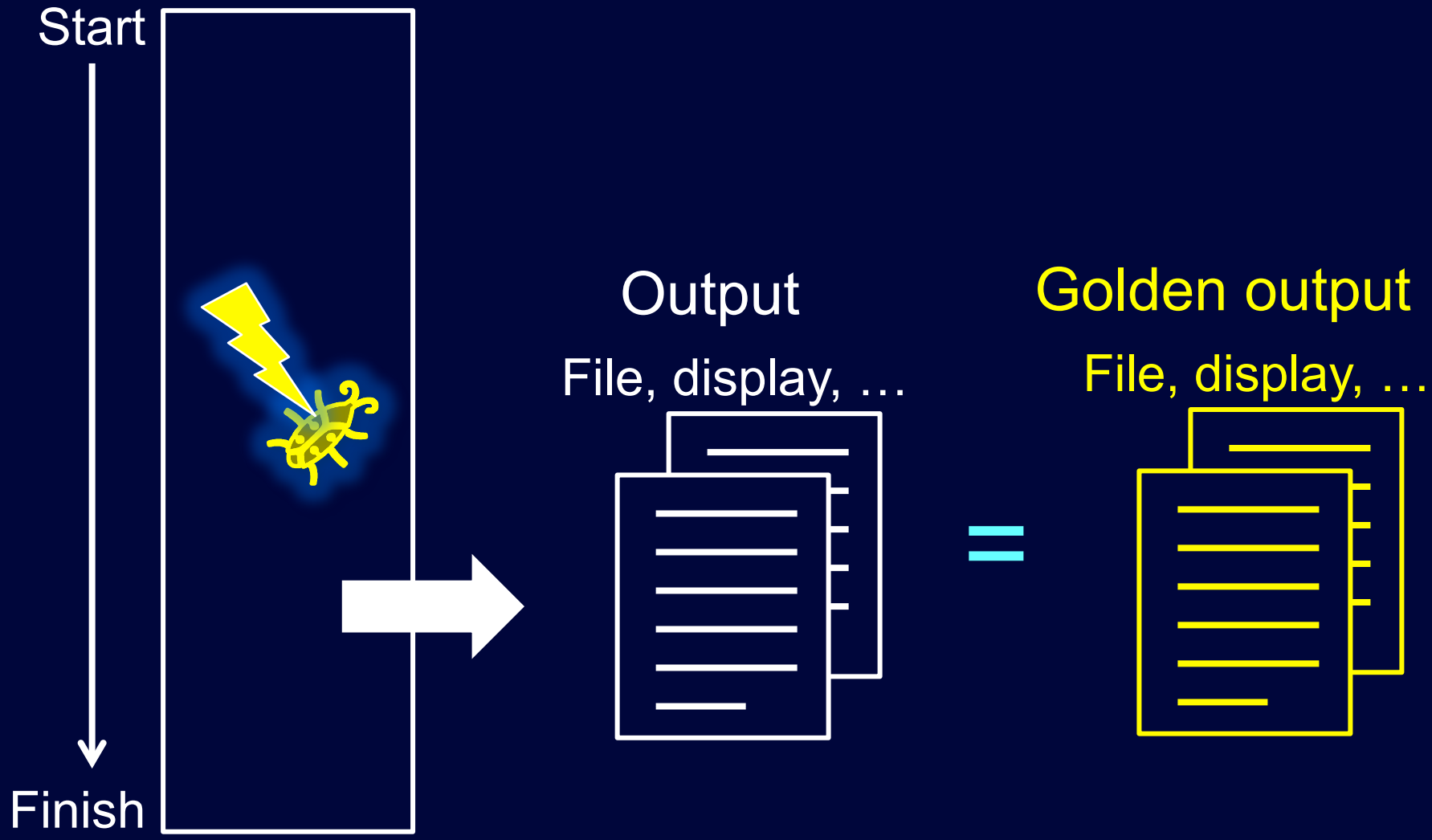
- Reliability: $R(t)$
  - Probability system works correctly up to time t
- Exponential model
  - $R(t) = e^{-\lambda t}$, λ = failure rate
- Mean Time to Failure (MTTF)

$$MTTF = \int_0^\infty t \times R(t)\,dt = \frac{1}{\lambda}$$

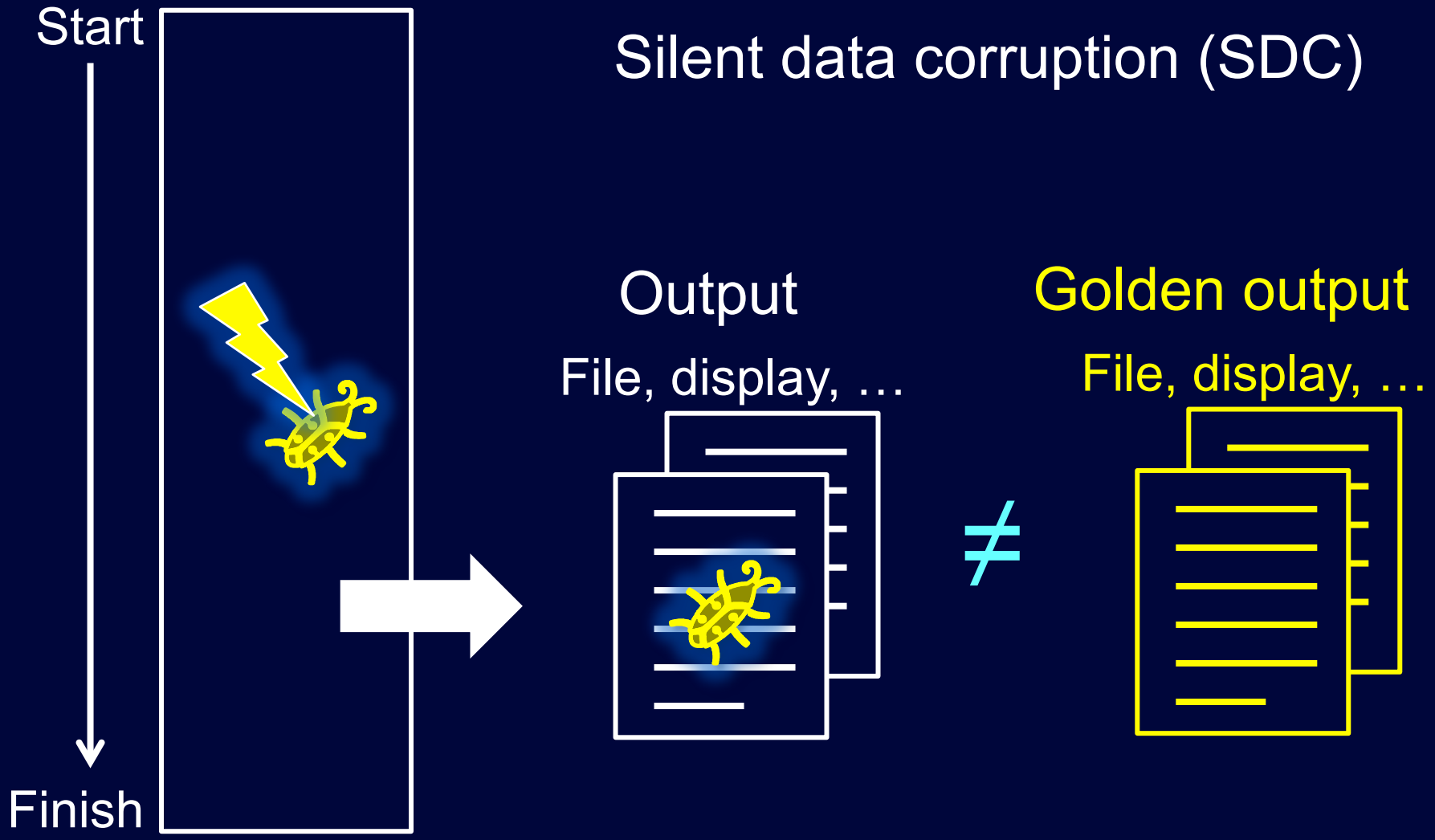
Definitions

- Availability: $A(t)$
  - Probability system works correctly **AT** time t
- Assume: system repaired after failure
  - Mean Time to Repair (MTTR)
- Steady-state availability: $\frac{MTTF}{MTTF + MTTR}$
- How to improve availability ?

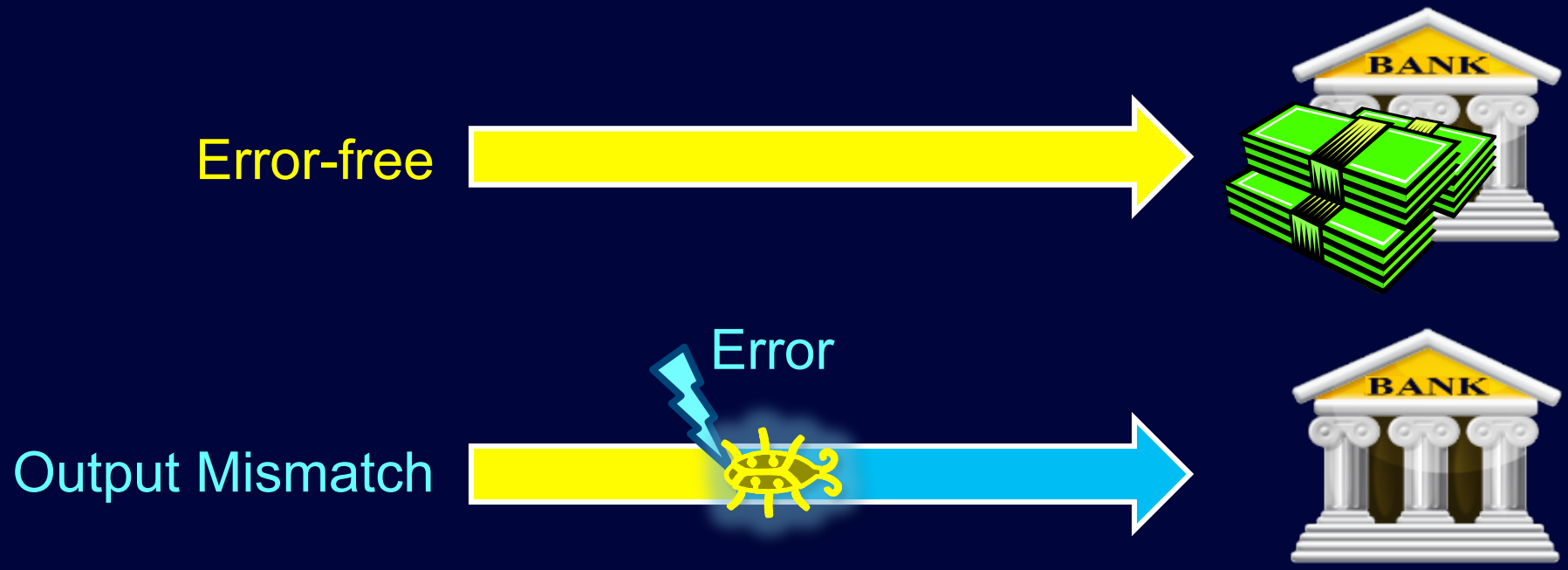
Error Effects: Vanished



Error Effects: Output Mismatch

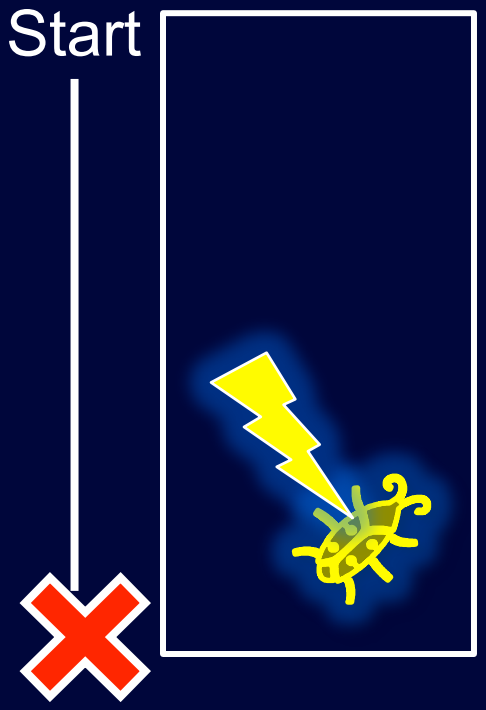


Silent Data Corruption (SDC)



- Output file incorrect
- No error indication

Error Effects: Unexpected Termination

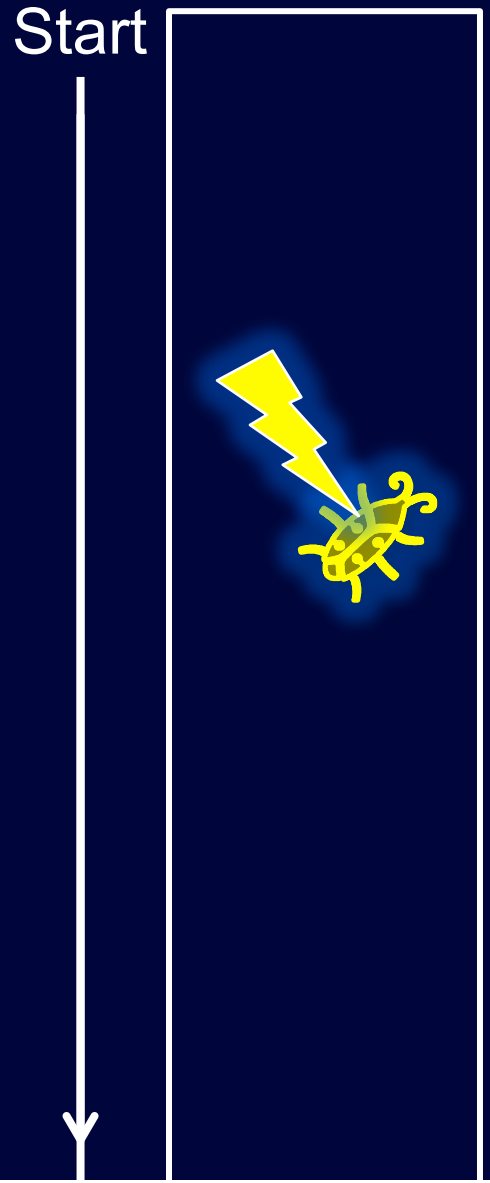


e.g.,

- Divide-by-zero
- Memory access violation
- Application-detected errors
- …

~~Finish~~

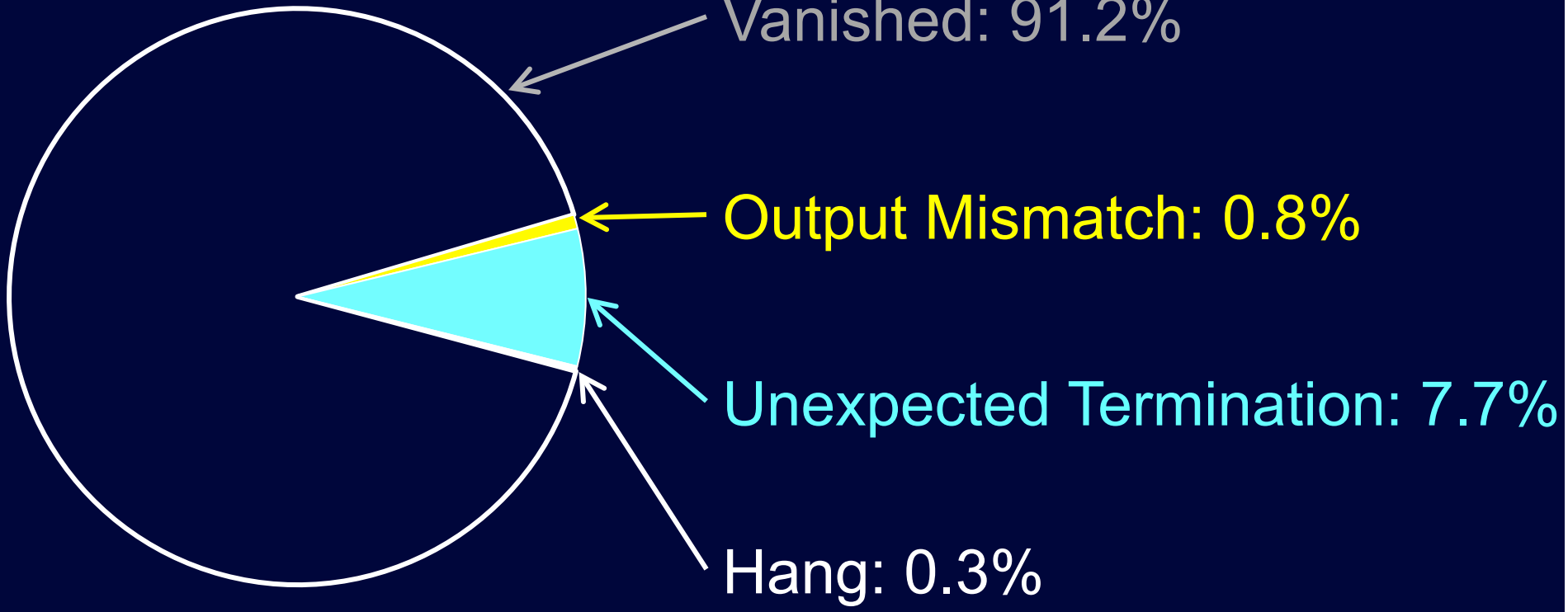
Hang



> 2 × error-free execution time

Does not finish / terminate

Soft Error Effects

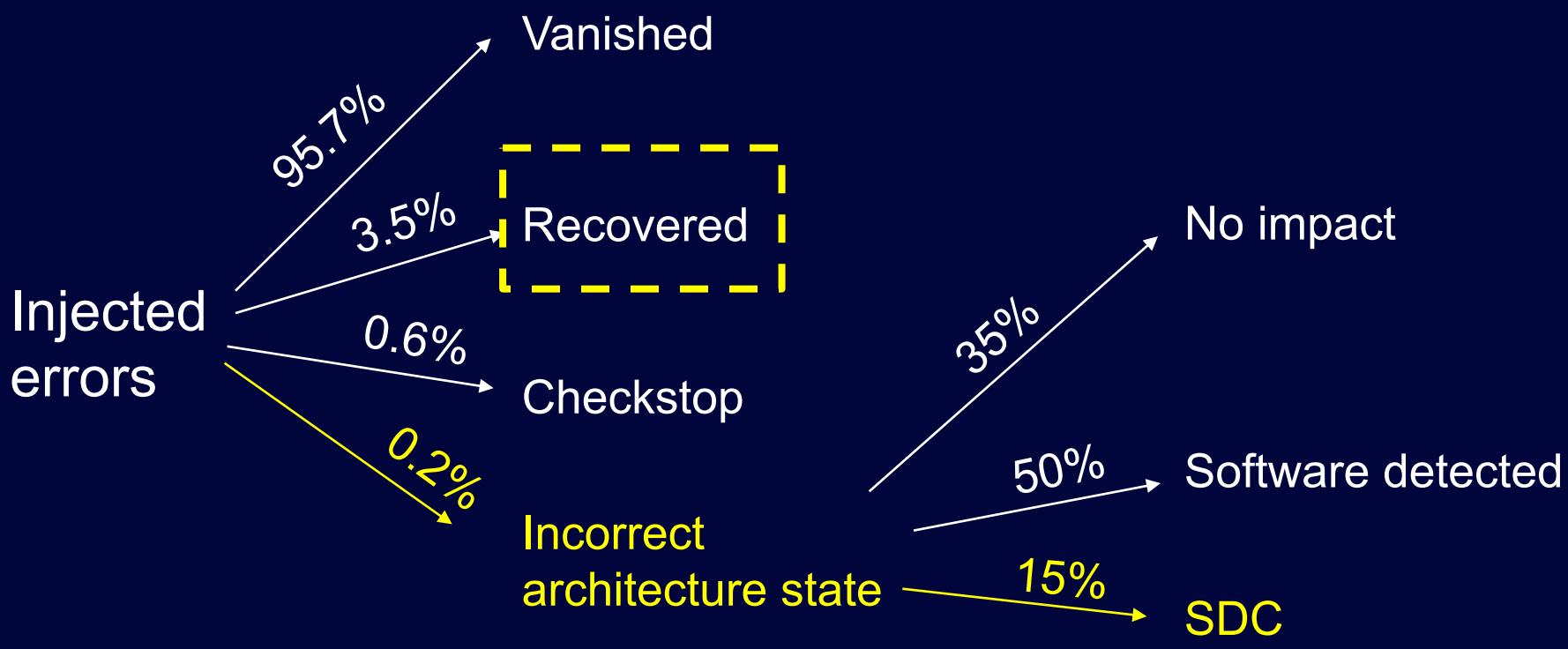


All injected flip-flop errors

Detected but Uncorrected Errors (DUE)

[Cho DAC 13]

Soft Error Effects: BZip2 on IBM Power6



[Sanda 08] (graphic contributed by Dr. Pia Sanda, ex-IBM)

Fault-Tolerance: Rich Literature

Expensive

Early pioneers

Early systems

How Low Cost ?

Applications

Approach

No

Output errors

Yes

Low-cost detection / correction

BISER, LEAP

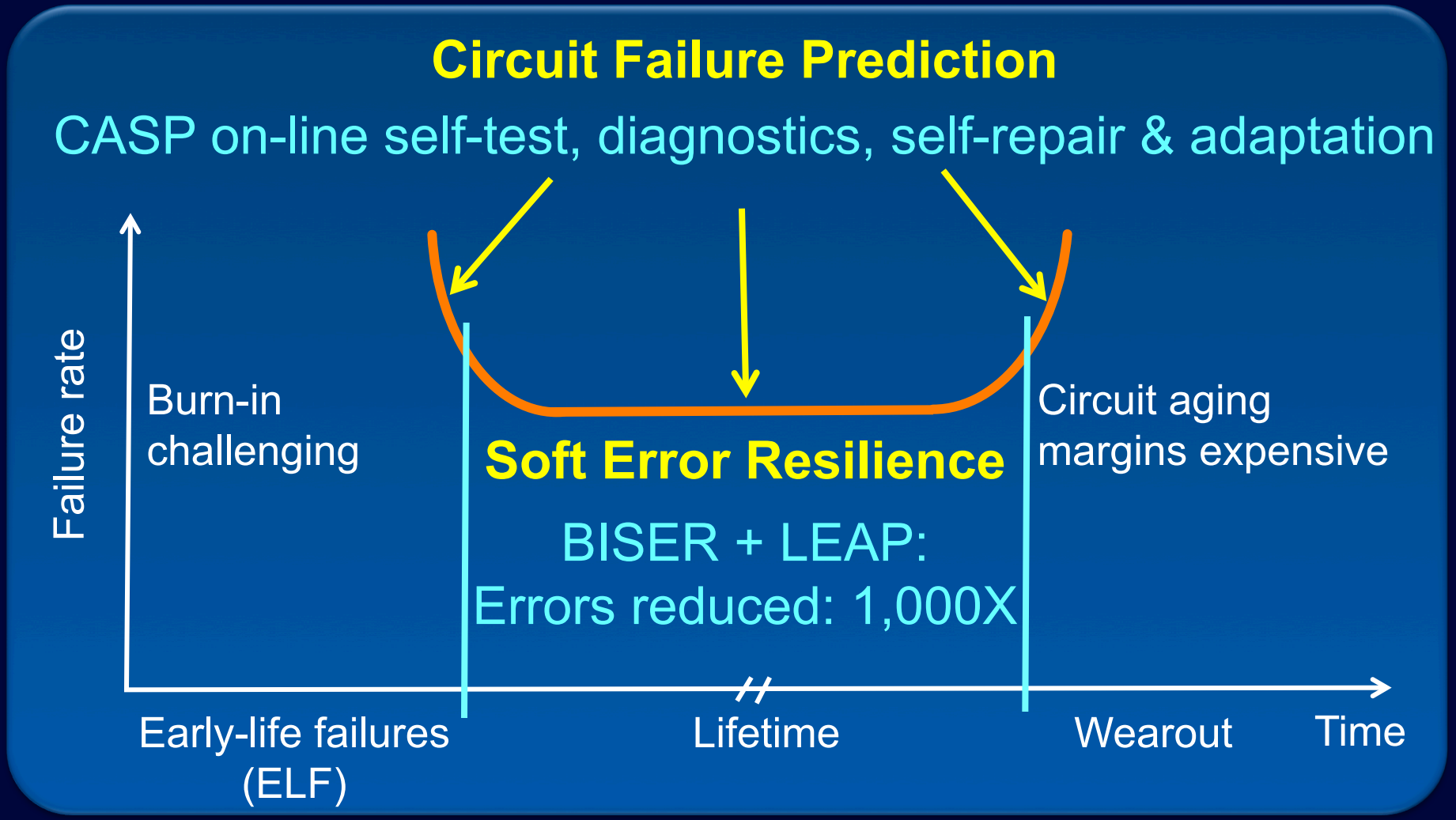
Circuit failure prediction

Light-weight correction

Error Resilient System Arch. (ERSA)

[Cho IEEE TCAD 12]

Low-Cost Techniques



How Low Cost?

Solution: cross-layer resilience?

"multiple error resilience techniques from different layers of the system stack cooperate to achieve cost-effective error resilience"

[DARPA, PERFECT BAA 12]

[Borkar Intel, IEEE Micro 05]

[Carter Intel, DATE 10]

[Pedram, NSF 12]

[Chandra ARM, DAC 14]

[Gupta IBM, IRPS 14]

[Henkel, DAC 14]

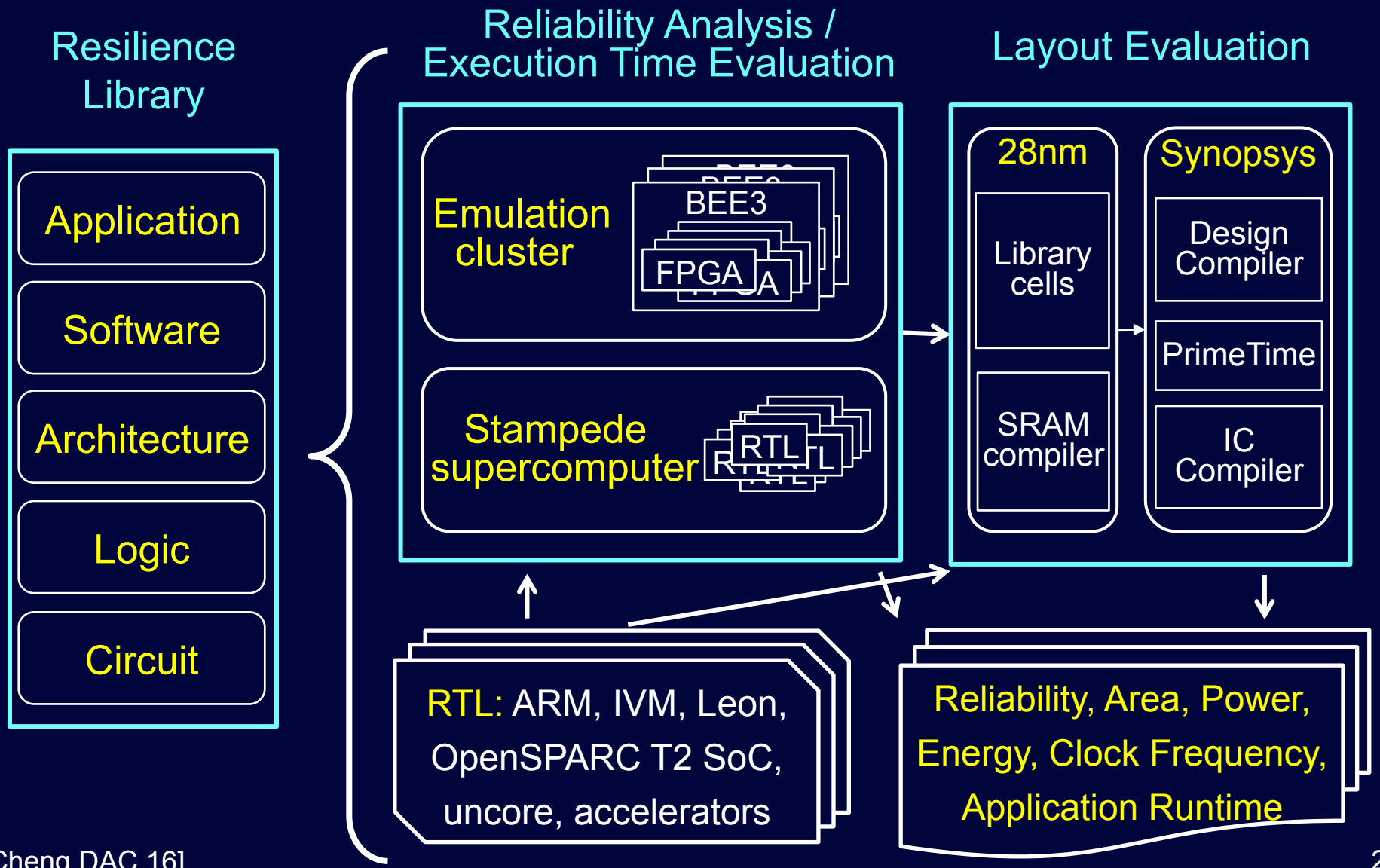
Existing Resilience Techniques

- Many point solutions
  - Some cross-layer, some single-layer
- Missing
  - End-to-end cross-layer resilience framework

ARGUS [Meixner, Micro 2007]
Parity checking [Spainhower, IBM Journal 1999]
BCDMR [Furuta, VLSI Circuits 2010]
EDDI [Oh, Trans. Reliability 2002]
DICE [Calin, Trans. Nuclear Science 1996]
Bose-Lin code [Bose, IEEE Trans. Computers 1985]
Reliability-driven transforms [Rehman, CADICS 2014]
Relyzer [Hari, DSN 2012]
Berger code [Berger, Information and Control 1961]
Fault screening [Racunas, HPCA 2007]
DIVA [Austin, Micro 1999]
z990 [Meaney, T-DMR 2005]
Resilience Actuators [Kleeberger, IEEE Micro 2013]
HLS-driven [Campbell, DAC 2015]
Multi-Layer [Henkel, DAC 2014]
Scalable Hardware [Chippa, DAC 2010]
ABFT for matrix [Huang, IEEE Trans. Computers 1984]
LEAP [Lee, IRPS 2010]
ANT [Hedge, ISLPED 1999]
Shoestring [Feng, Comp. Arch. News 2010]
ABFT HPC [Bosilca, Journal Parallel and Distributed Computing 2008]
Error detectors [Pattabiraman, DSC 2007]
Assertions [Sahoo, DSN, 2008]
RSE [Nakka, DSN 2004]
RCC [Seifert, IRPS 2010]
SSNoC [Varatkar, ISSCC 2007]
ABFT arrays [Nair, IEEE Trans. Computers 1990]
CFCSS [Oh, Trans. Reliability 2002]
Razor [Blaauw, ISSCC 2008]
BISER [Mitra, IEEE Computer 2005]
SWIFT [Reis, ISCG 2005]
RMT [Mukherjee, ISCA 2002]

CLEAR

Cross-Layer Exploration for Architecting Resilience



[Cheng DAC 16]

Today's Focus

- Radiation-induced soft errors in flip-flops
  - Single-Event Upsets (SEUs)
  - Single-Event Multiple Upsets (SEMUs)
- Combinational logic soft errors not critical

CLEAR: Extensive Study

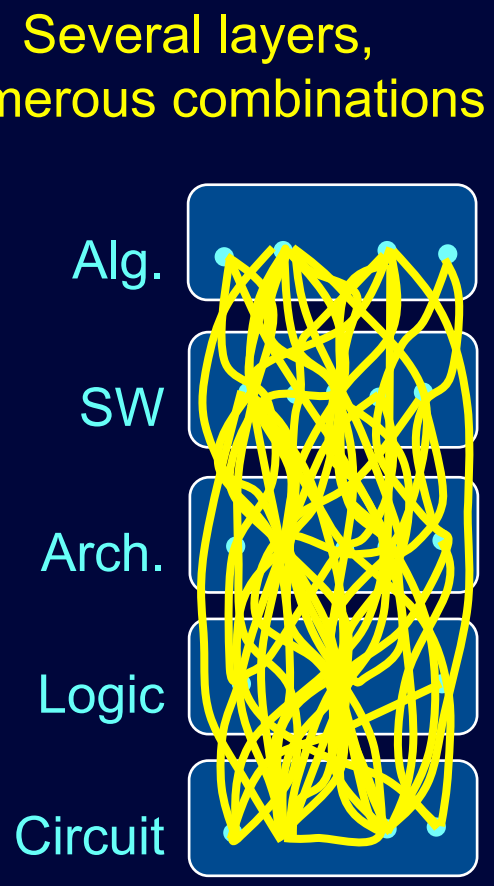
- Designs: wide variety
  - ARM, LEON3, Alpha, OpenSPARC multi-core SoC, accelerators
- Thorough flip-flop error injections
  - FPGA clusters, Stampede supercomputer (522,080 cores)
  - Full workloads (SPEC, PARSEC, PERFECT, proprietary)
- Detailed physical design
  - Wire routing, process / voltage / temperature corners

Many Cross-Layer Questions Answered

- Cross-layer always best?
- All cross-layer solutions equally good?
- Application constraints (e.g., soft real-time)?
- Benchmark dependence?
- Definitive guidelines for new resilience techniques

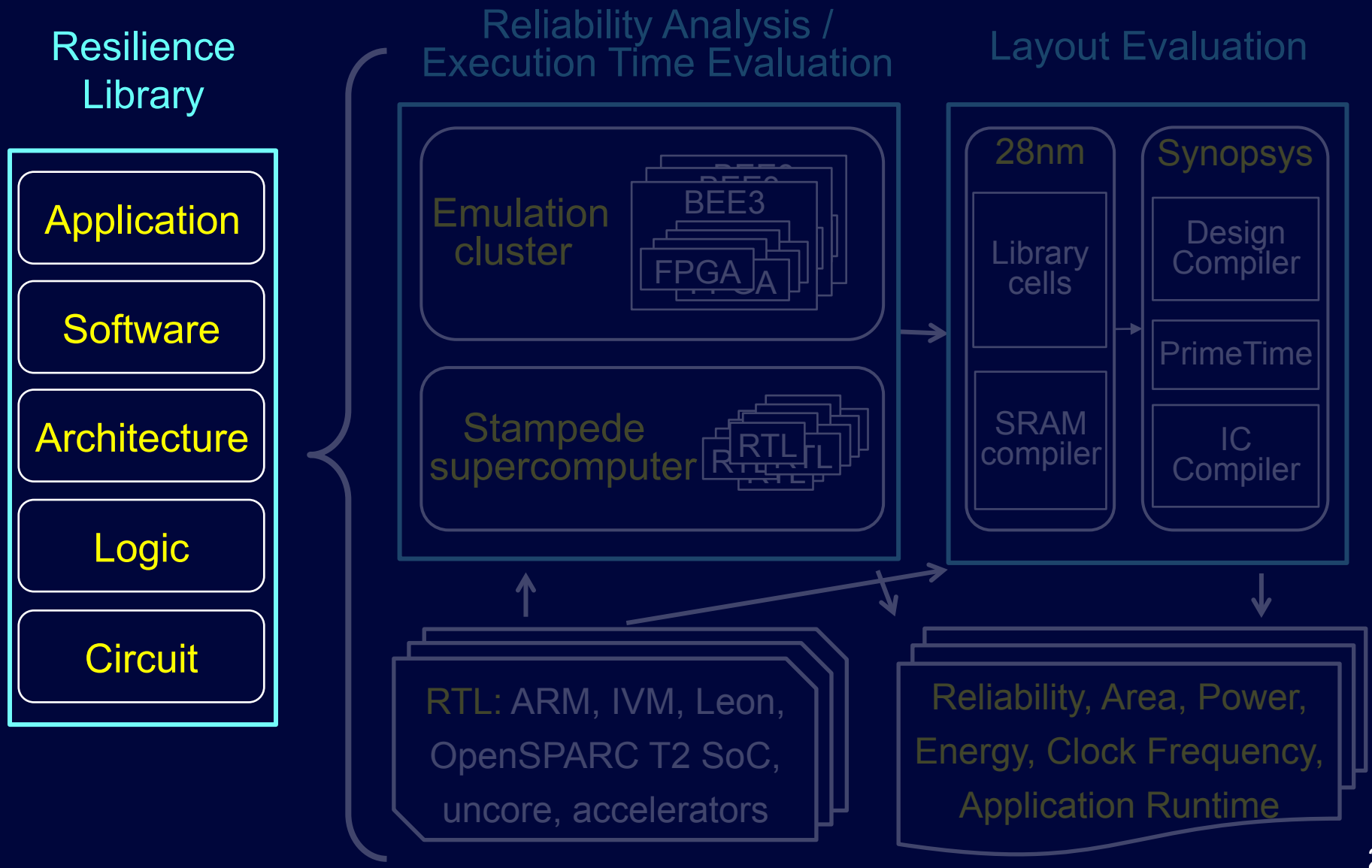
Key Message

- From black art to science
- 5-50x resilience, 0.2-6% energy cost
  - Circuit + logic + micro-arch. recovery
- Circuit alone (application-guided)
  - ~1% extra energy vs. best cross-layer

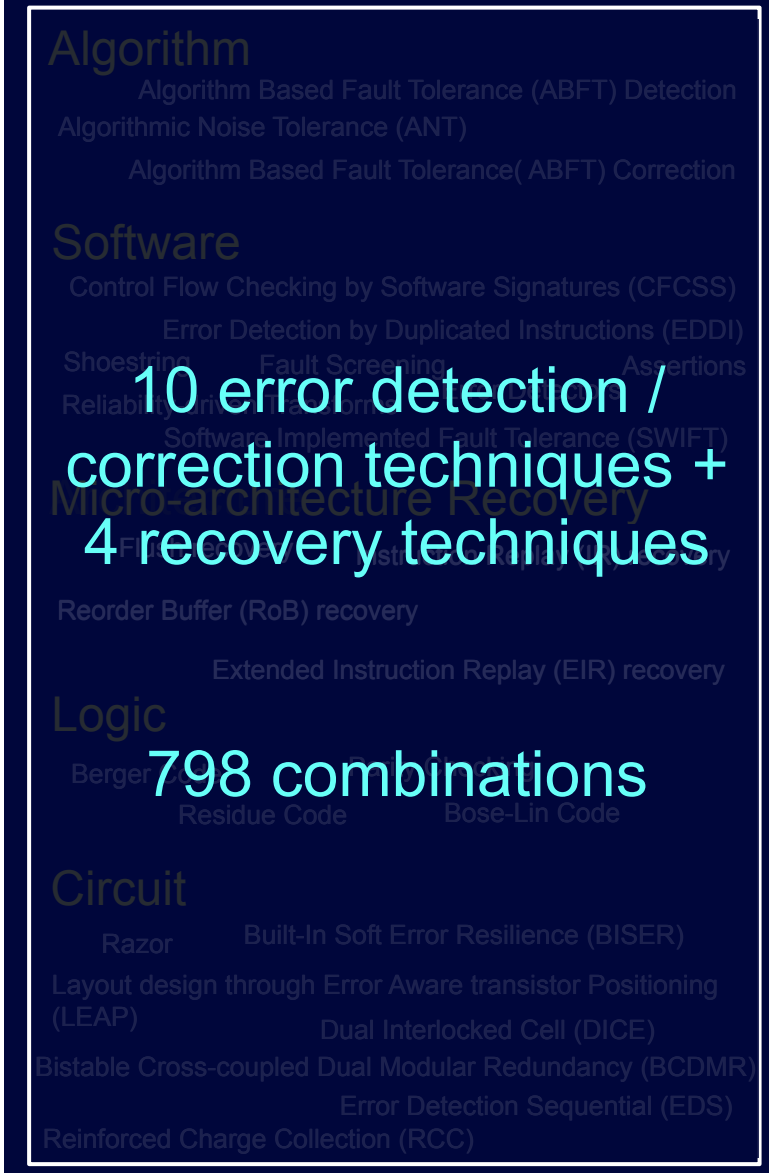


CLEAR

Cross-Layer Exploration for Architecting Resilience



Representative Resilience Techniques



1. Algorithm Based Fault Tolerance (ABFT) Correction
2. ABFT Detection

3. Software Assertions
4. Control Flow Checking by Software Signatures (CFCSS)
5. Error Detection by Duplicated Instructions (EDDI)

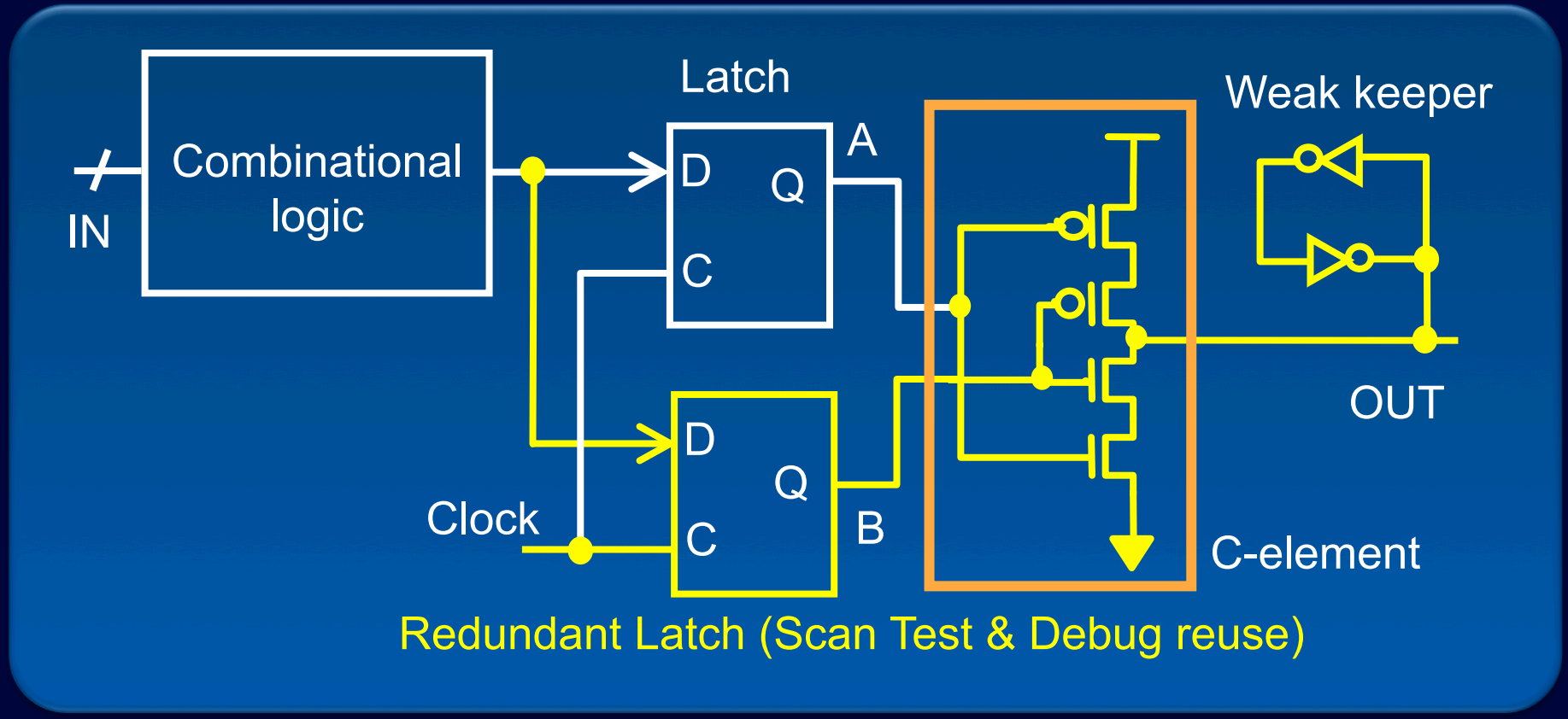
6. Data Flow Checking (DFC)
7. Monitor Cores

8. Logic Parity

Micro-arch. Recovery
1. Flush
2. Reorder Buffer (RoB)
3. Instruction Replay (IR)
4. Extended IR (EIR)

9. Layout design through Error-Aware transistor Positioning (LEAP)
10. Error Detection Sequential (EDS)

BISER: Built-In Soft Error Resilience



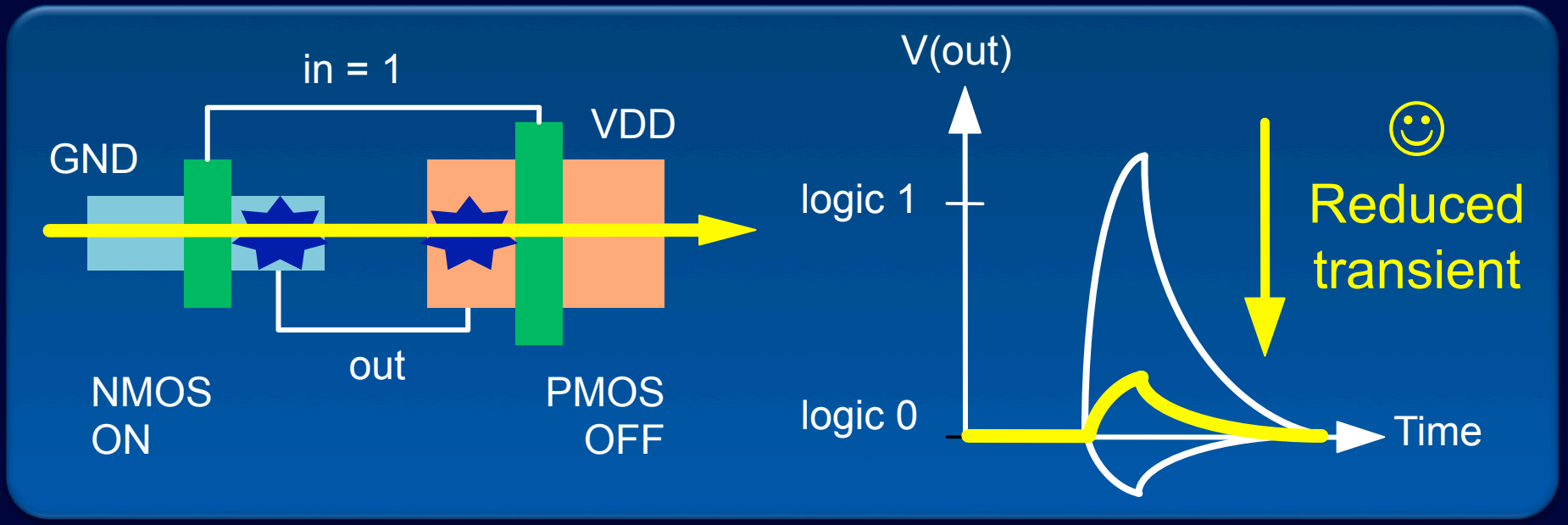
45nm: up to 1,000X benefits

[Mitra IEEE Computer 05, TVLSI 06]

Single Error Assumption Inadequate

- Single-event **multiple** upsets (SEMUs)

LEAP: Layout by Error Aware transistor Positioning



Errors **Corrected**: SEUs and SEMUs

[Lee IRPS 10, Lilja IEEE TNS 13]

Extensive LEAP Characterization

- Radiation beam experiments
  - 40nm, 28nm, 20nm, 14nm
  - Bulk, SOI
- VDD: nominal, near-threshold

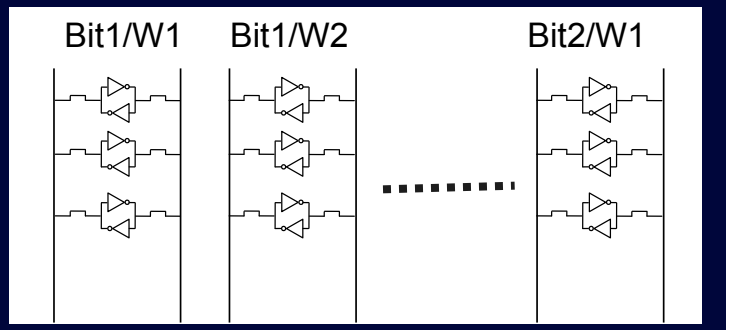
Flip-flop	Soft Error Rate (SER)	Area	Power	Delay	Energy
Baseline	1	1	1	1	1
LEAP-DICE	**2×10^{-4}**	2	1.8	1	1.8

[Lee IRPS 10; Lilja IEEE T. Nucl. Sci. 13, SEE 16; Quinn NSREC 15, REDW 15; Turowski SEE 15]

Memory ECC and SEMUs

- Don’t implement multiple error correction blindly
- Multiple **physically adjacent** errors

Memory ECC and SEMUs



Option 1

- Memory interleaving
  - 2 physically adjacent errors
    - Single errors in 2 separate words
  - ☹ Cost, difficult for smaller geometries

Option 2

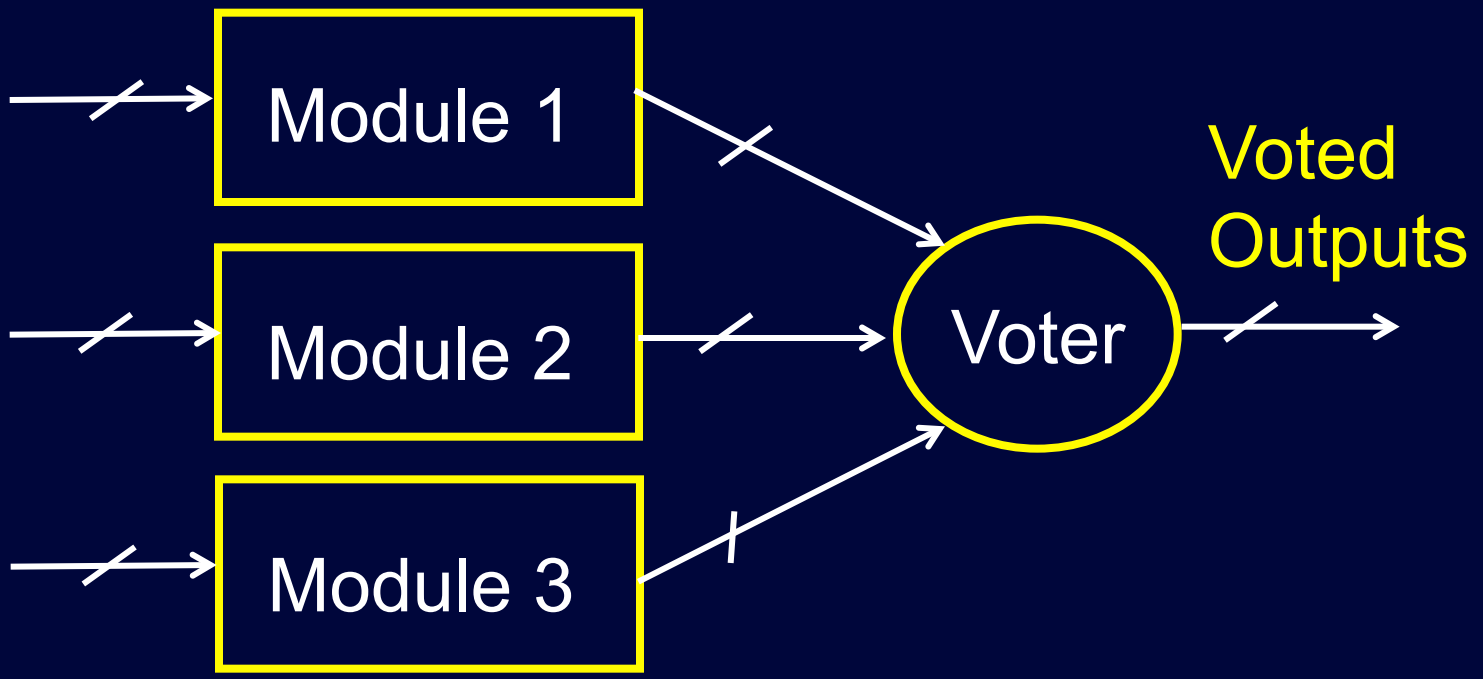
- Adjacent bit error correction [Dutta ITC 07]

Memory ECC Challenges

- Performance overhead
  - Pipelining
    - Additional latency, verification effort
  - Detection followed by correction
    - Variable latency, verification effort
- Small distributed memories

Error Masking

- No error on outputs
  - **T**riple **M**odular **R**edundancy (TMR)

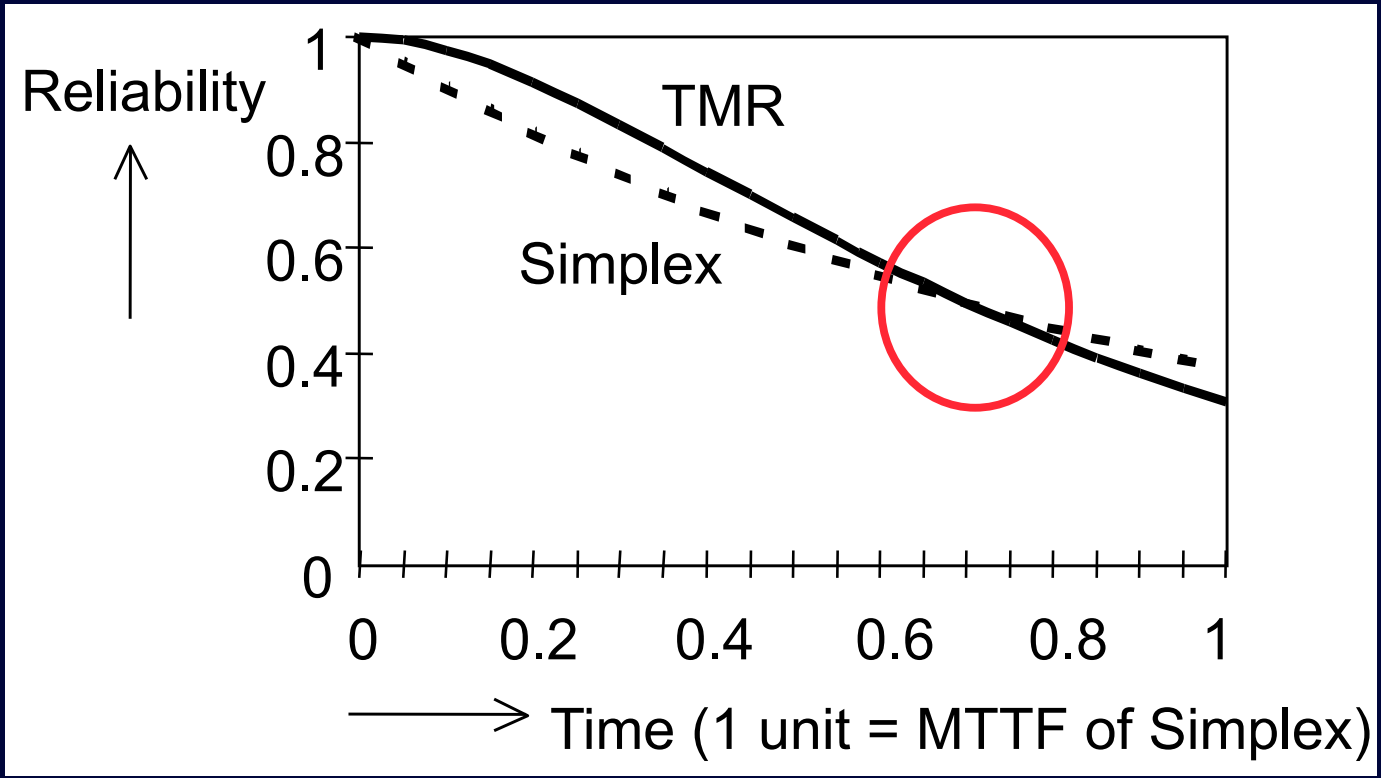


TMR Reliability

$$R_{TMR} = R_{voter} \times [R_m^3 + R_m^2 \times (1 - R_m)]$$

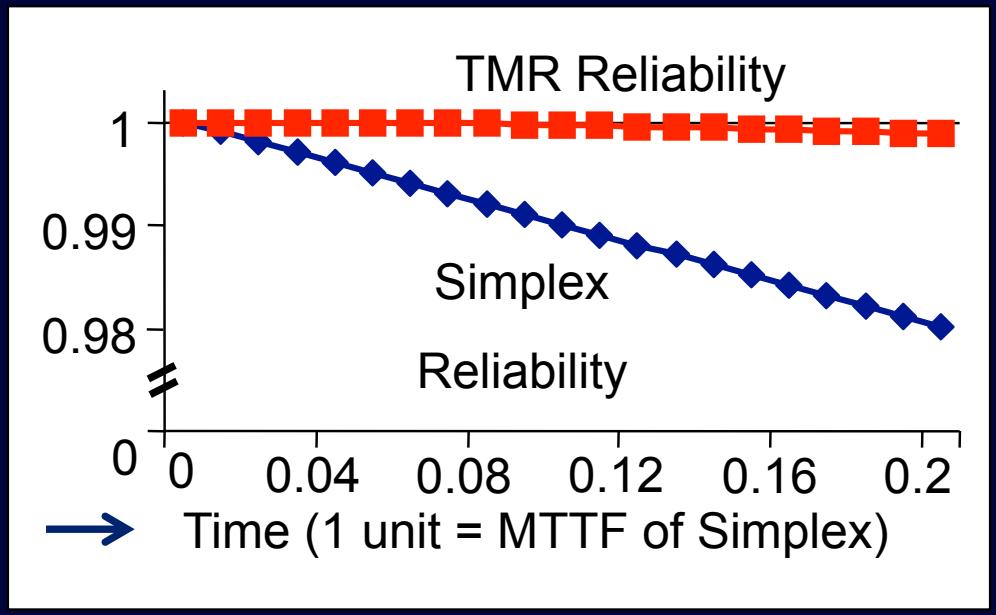
- R_m: individual module reliability
- Pessimistic: non-overlapping errors
- Optimistic: correlated / common-mode failures
- TMR MTTF < Simplex MTTF

TMR Reliability



- TMR reliability = simplex reliability
  - Time = $\log_e 2 \times$ Simplex MTTF (perfect voter)

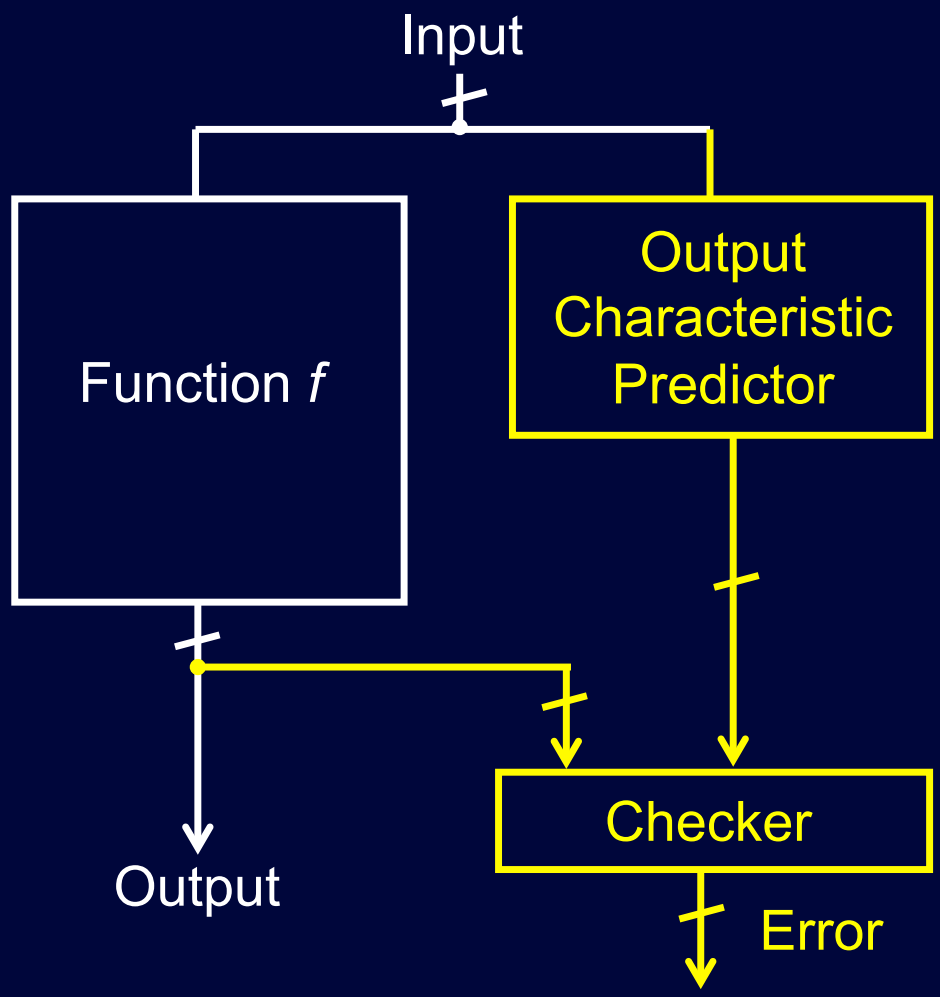
TMR Reliability vs. Mission Time



- TMR effective for “short” mission times
- Other options: TMR-Simplex, TMR + Duplex-Repair

Concurrent Error Detection (CED)

- Normal system operation
- Preserve data integrity
  - Correct outputs or
  - Error indicated
    - Incorrect outputs
  - aka fault-secure



Output “Characteristics”

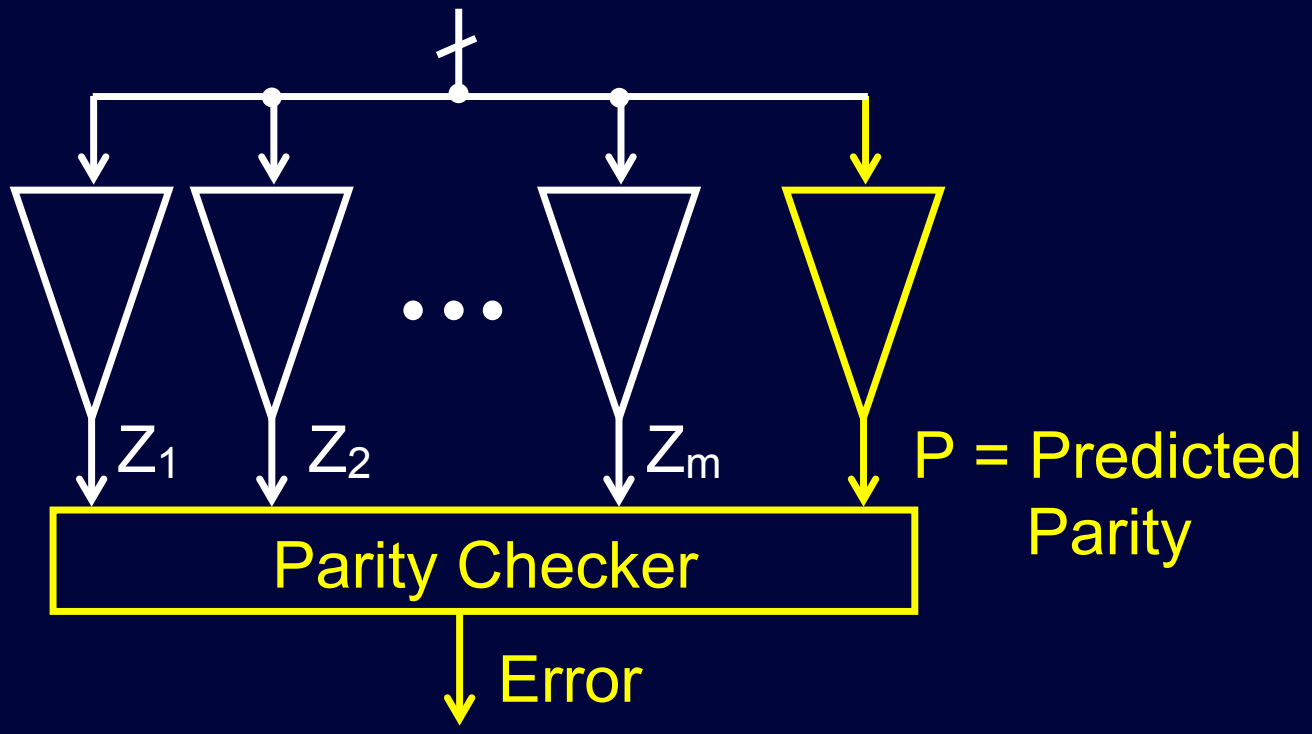
- Output itself: duplication
  - Major challenges if not “fine-grained”
- Output parity
- Output residue
- 1s or 0s count in output word
- Many others (extensive literature)
- Self-checking checkers

Processor Duplication Challenges

- Synchronization !!
- False DUEs when out of sync
  - e.g., error correction event in one processor
  - Mismatch when output pins compared

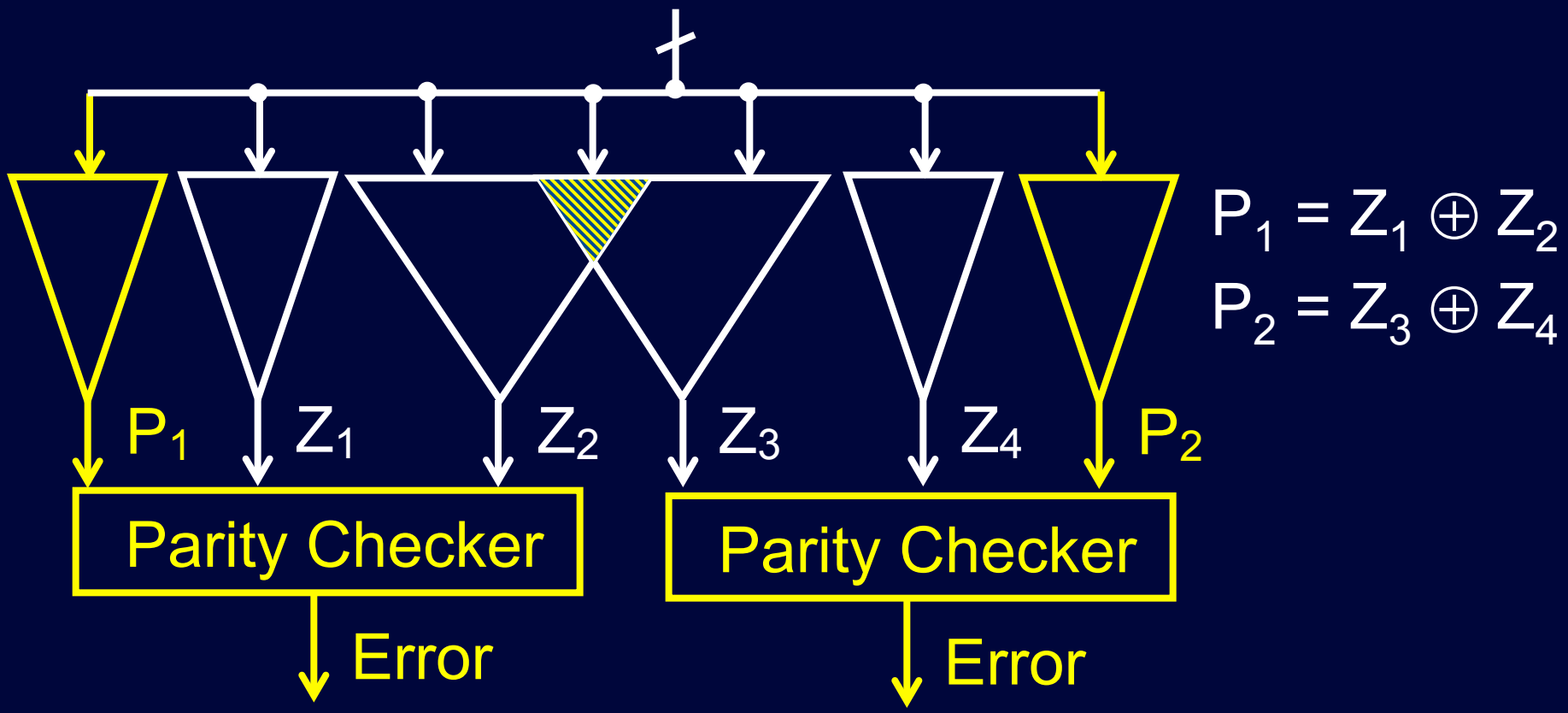
Single-Bit Logic Parity Prediction

- $P = Z_1 \oplus Z_2 \oplus \ \ldots \oplus Z_m$
- Disjoint output logic (no logic sharing)
  - Only for combinational logic errors



Multiple-Bit Logic Parity Prediction

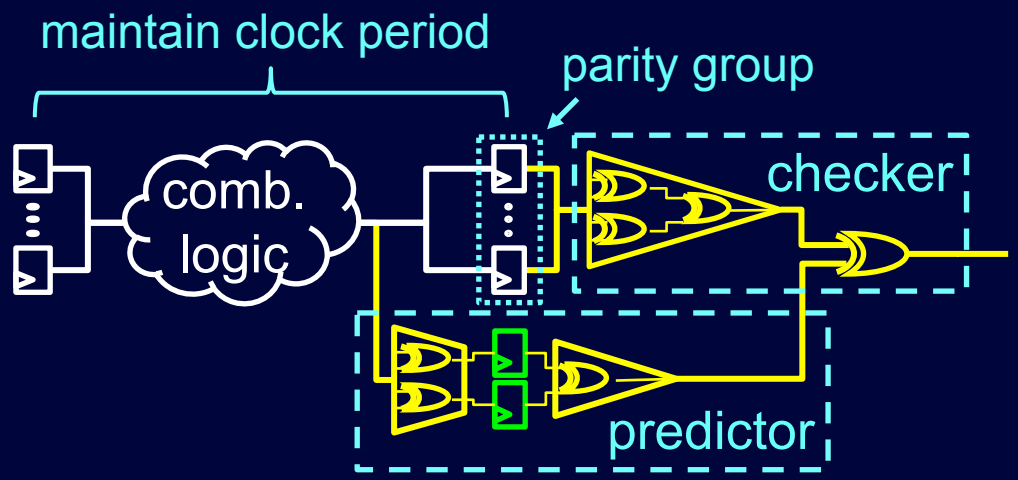
- Main purpose: cost reduction (sharing, routing, logic)
  - Can be still expensive



Logic Parity Checking: Flip-Flop Errors

Logic parity

- Original design
- Parity logic
- Pipeline flip-flops



Logic parity: Naïve	200 MHz clock speed impact
Logic parity: Incorrect heuristic	80% additional energy impact
Logic parity: CLEAR heuristic	No clock speed impact Minimal energy impact

Parameters: parity size, flip-flop vulnerability, floorplan location, timing path slack, etc.

Parity Prediction for Datapath Circuits

- $S = A + B$ (n-bit operation)
- Parity (S) = $S_1 \oplus S_2 \oplus S_3 \oplus \ldots S_n$

  = $(A_1 \oplus B_1 \oplus C_1) \oplus (A_2 \oplus B_2 \oplus C_2) \ldots (A_n \oplus B_n \oplus C_n)$

  = $(A_1 \oplus A_2 \ldots A_n) \oplus (B_1 \oplus B_2 \ldots B_n) \oplus (C_1 \oplus C_2 \ldots C_n)$

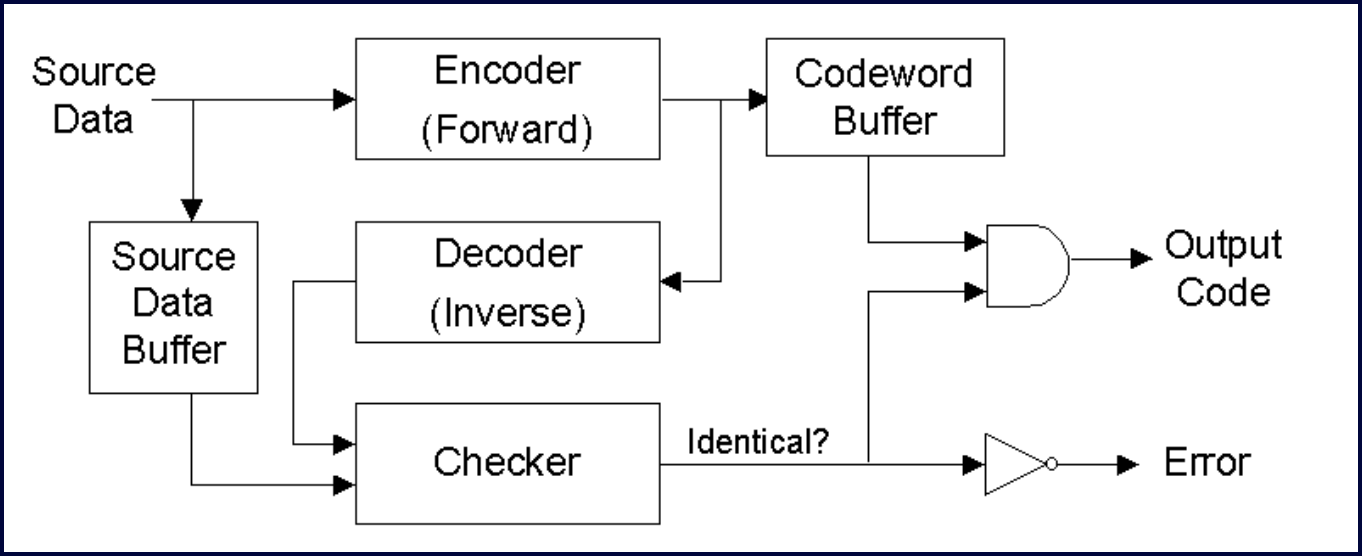
  = Parity (A) $\oplus$ Parity (B) $\oplus$ Parity (internal carries)
- Parity (internal carries) expensive
  - Several strategies for high-performance adders

Residue Codes for Datapath

- $y = x \bmod b$: y is residue of x (modulo b)
- Residue $(A + B)$ = Residue (A) + Residue (B)
- Residue $(A \times B)$ = Residue $(A) \times$ Residue (B)
- Choice of b: Mersenne prime (form 2^m-1)
  - Coverage, checker complexity
- Issues: bit-wise logic, operand residue, checker cost
  - Often used for multipliers

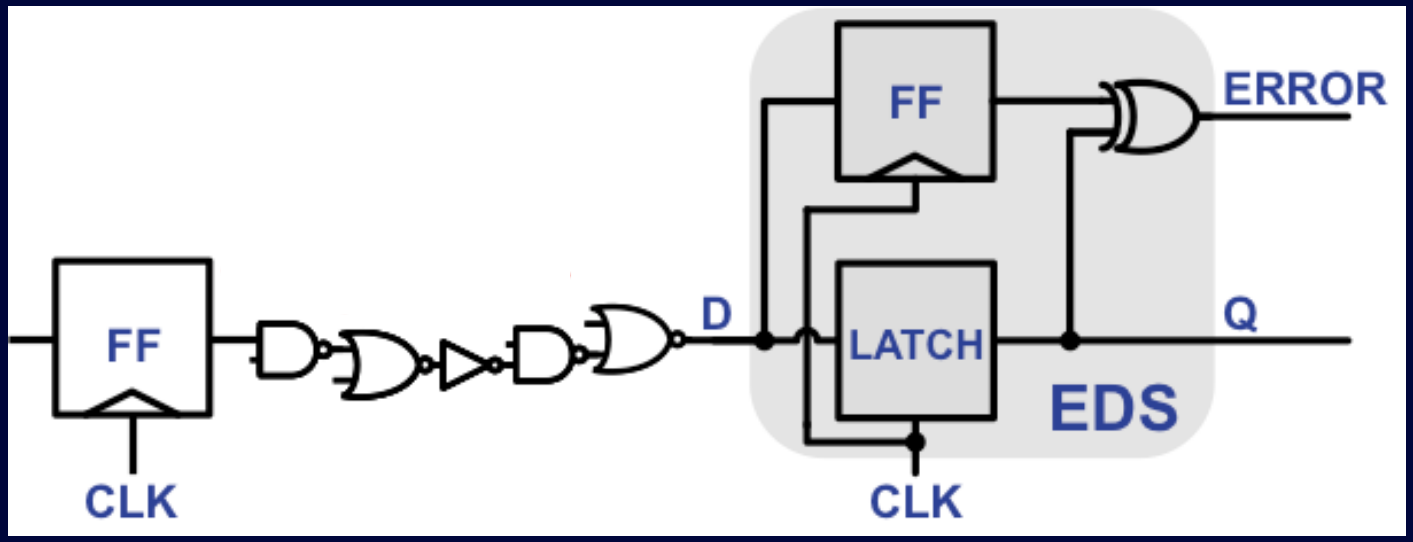
Application-Specific CED

- LZ compression: loss-less, invertible
  - Compression: complex
  - Decompression: simple



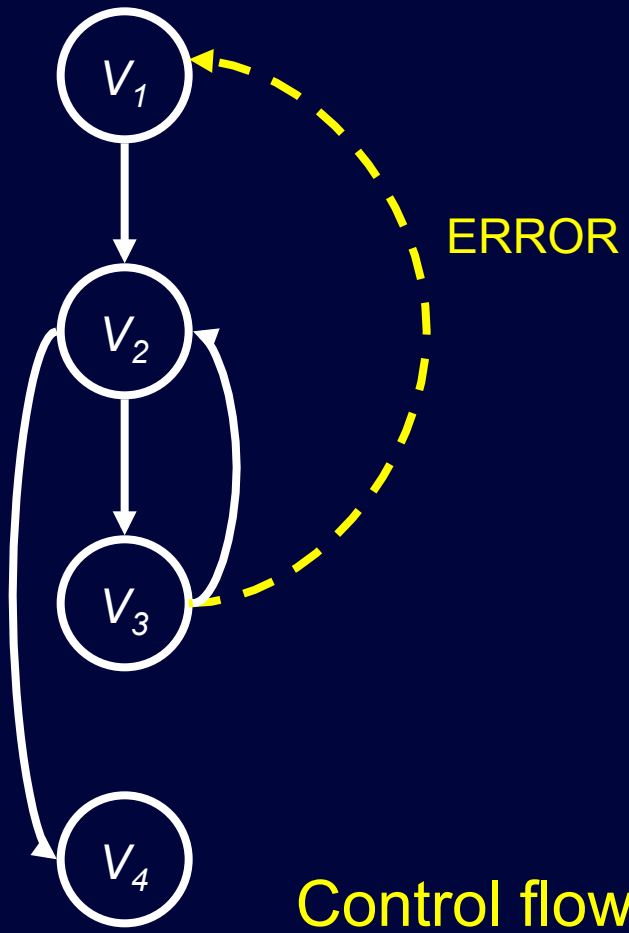
9% area overhead, 0.5% delay overhead [Huang 00]

Error Detection Sequentials (EDS)



Slide obtained from K. Bowman, J. Tschanz, et al., Intel

Errors in Processors



Control flow error

Incorrect instruction sequence

Computational error

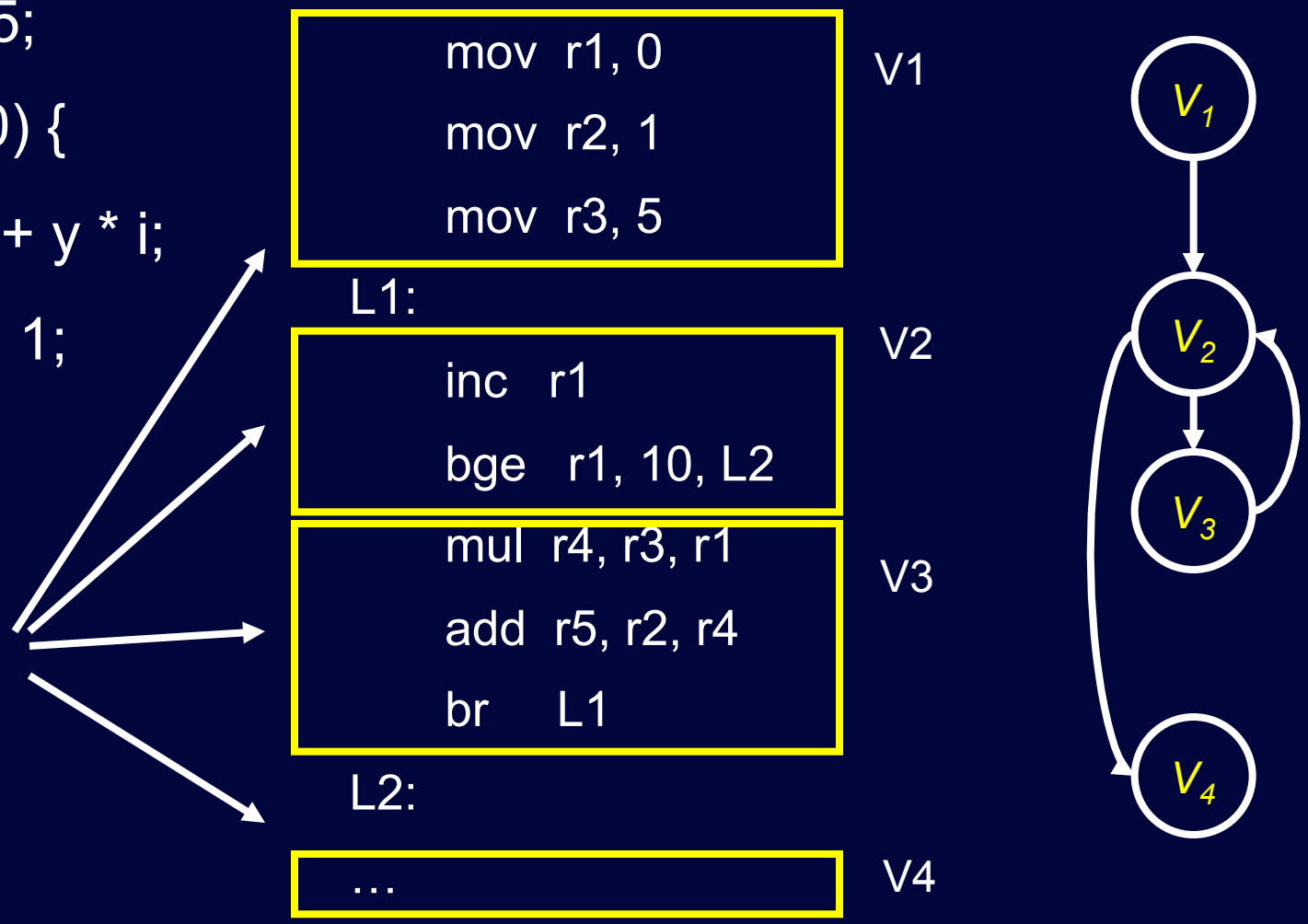
Incorrect computation

Memory error

Incorrect value or address

Program Representation: Control Flow Graph

```
i=0; x=1; y=5;
While (i < 10) {
        z = x + y * i;
        i = i + 1;
}
```



SIHFT

- Software Implemented Hardware Fault Tolerance
  - Automated by compiler
  - EDDI [Oh, IEEE Trans. Reliability 02]
  - CFCSS [Oh IEEE Trans. Reliability 02b]
  - ED^{4}I [Oh IEEE Trans. Computers 02]
  - Lots of recent publications

EDDI

- Error Detection using Duplicated Instructions
- Duplicate instructions inside basic blocks
  - Different registers
- Duplicate data structures
- Comparison before memory stores
- Performance penalty 13% - 111%
  - Reduced by Instruction Level Parallelism (ILP)

EDDI Example

```
ADD R3, R1, R2          ; R3 ← R1 + R2
MUL R4, R3, R5          ; R4 ← R3 * R5
ST  0(SP), R4           ; store R4 in location pointed by SP
```

```
ADD R3, R1, R2          ; R3 ← R1 + R2 master
ADD R23, R21, R22       ; R23 ← R21 + R2 shadow
MUL R4, R3, R5          ; R4 ← R3 * R5 master
MUL R24, R23, R25       ; R24 ← R23 * R25 shadow
BNE R4, R24, Error_Handler  ; compare
ST  0(SP), R4           ; store master result
ST  offset(SP), R24     ; store shadow result
```

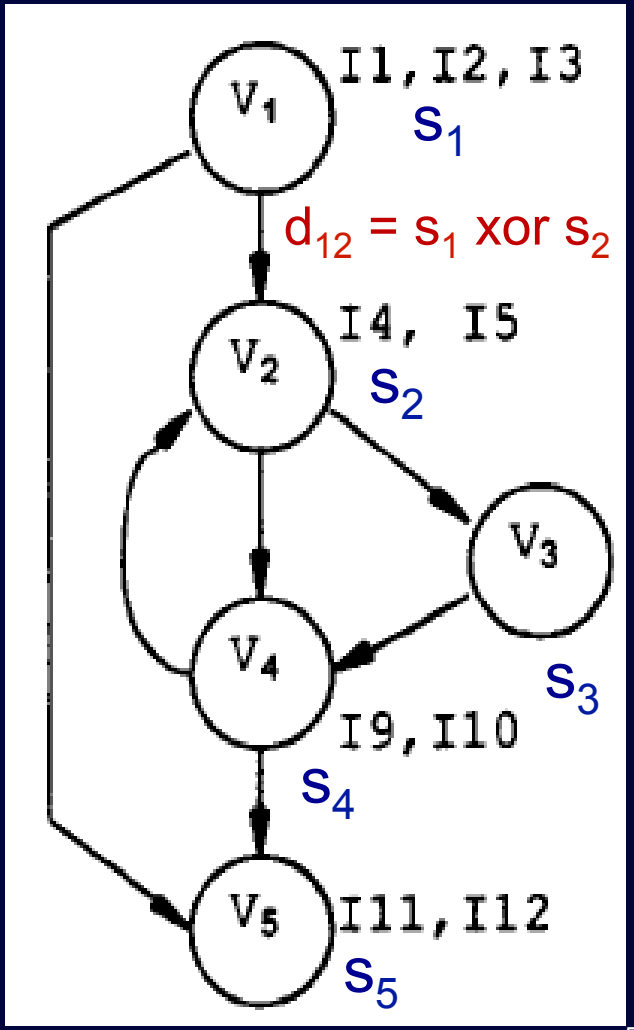
EDDI Design Choices

- Check after each instruction ?
- Storeless basic blocks (SBB) ?
  - No branch or store except final instruction
- Why SBB ?
  - Correctness defined by program output
  - Erroneous branches: stores skipped ?
    - Check at branches too

CFCSS

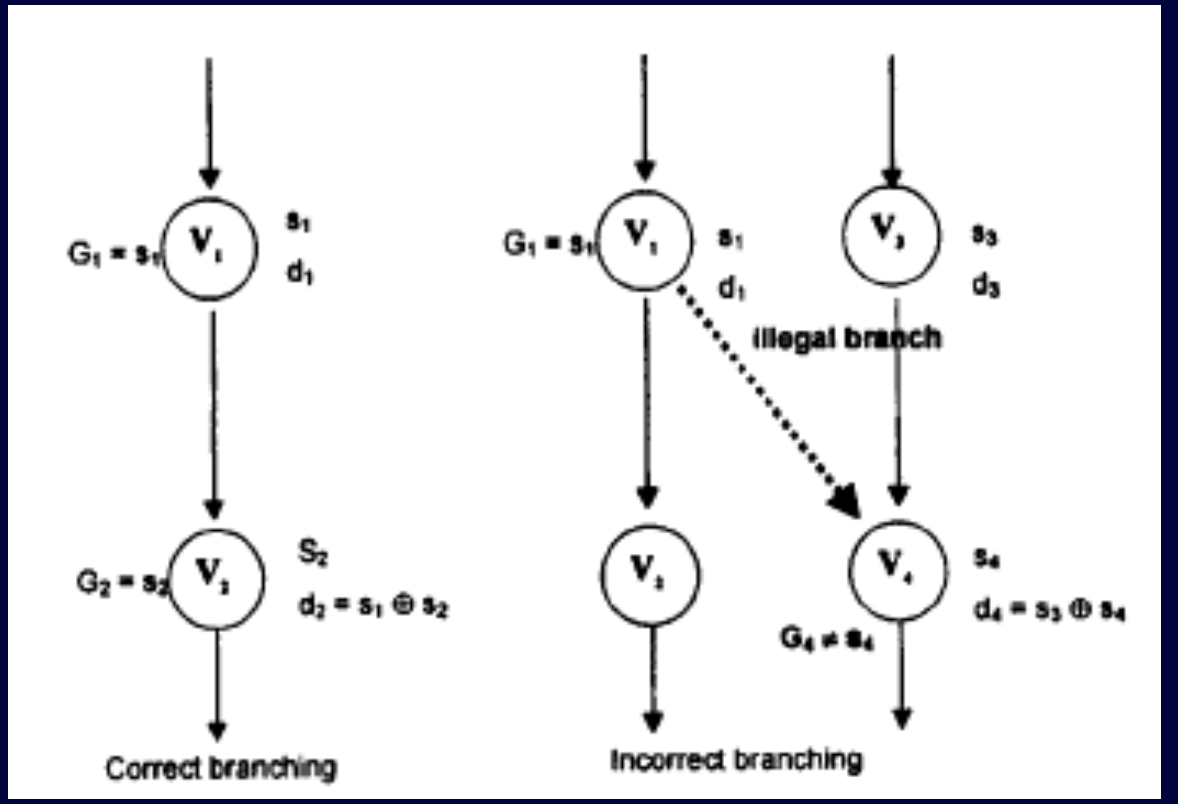
- Control Flow Checking using Software Signatures

  - Each node
    - Unique signature
  - Each edge
    - Transition between 2 signatures
    - Difference function: XOR



CFCSS

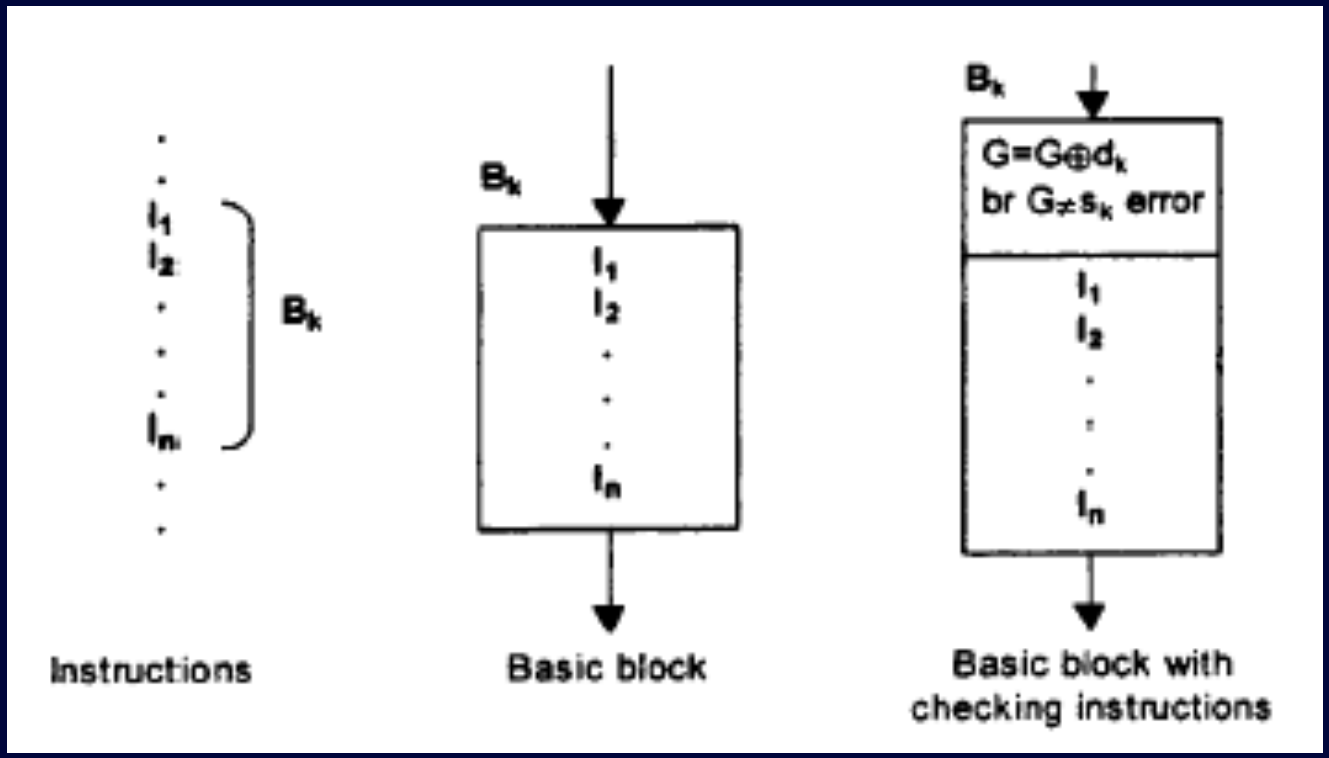
- Runtime signature *G*
- Basic block *i* to *j*
  - $G = s_i$ XOR $d_{i,j}$
  - Check $G = s_j$



G_n	run-time signature at node V_n
s_n	signature assigned to node V_n
d_n	signature difference

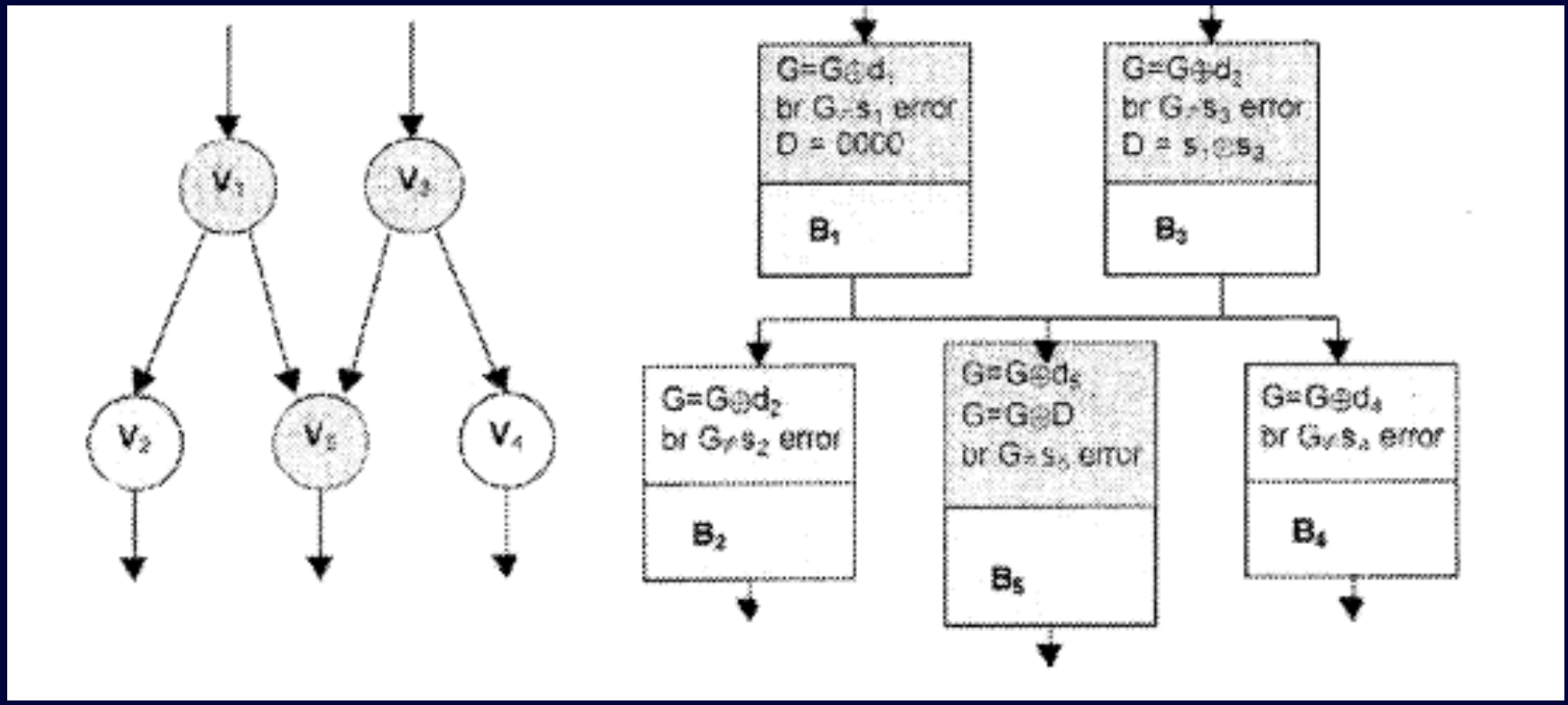
CFCSS Implementation

- Global variable *G* holds run-time signature
- Compute & check signature: start of each basic block



CFCSS: Branch Fan-in

- Basic block with multiple predecessors
- Run-time adjusting signature *D* differentiates fan-in



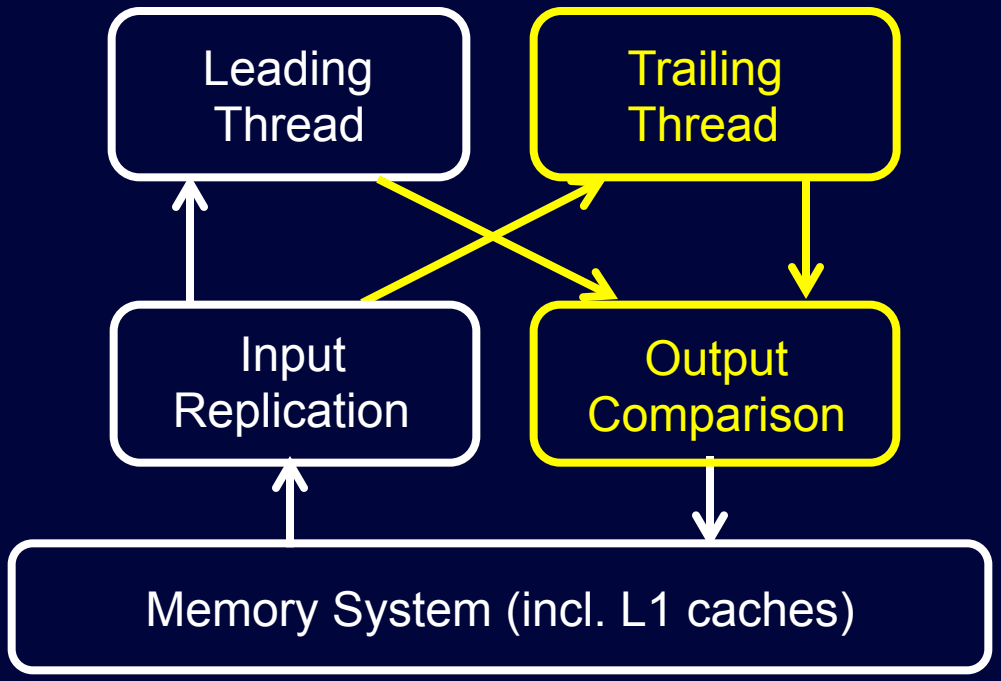
ED^{4}I

- Error Detection using Diverse Data & Duplicated Instructions
- Duplicated instructions, data diversity
  - Expressions in shadows multiply by *k* (-1, -2, …)
- *k* = -2: good choice
- Transient errors & most permanent faults detected
- Issues: floating point, pointers

SIHFT Results [Lovelette 02]

- COTS in space: no hardware redundancy
- ARGOS satellite experiment
  - Compare rad-hard processor vs. COTS
- Undetected errors in rad-hard processor
- COTS: 5.55 SEUs / Mbyte / day, 99.7% coverage
- 98.8% successful recovery: software ECC + restart
- COTS + SIHFT: faster than rad-hard

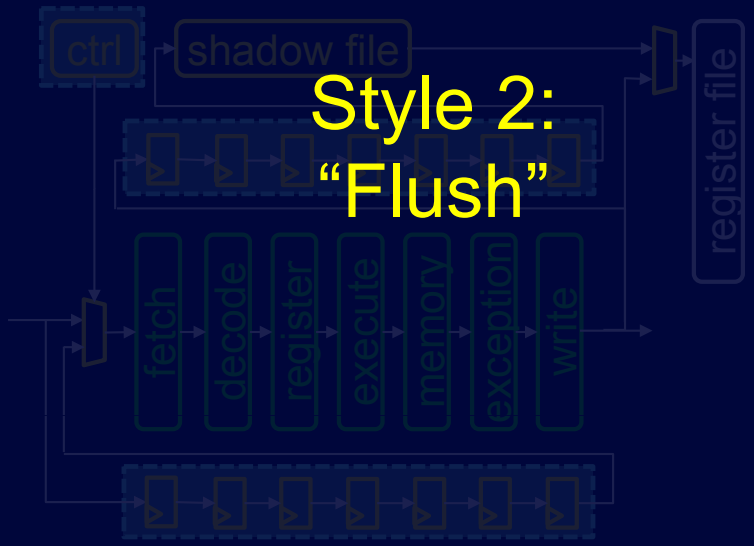
Multi-Threading for CED



- Same application computed by two threads
  - [Rotenberg 99, Saxena 00, Mukherjee 02]

What About Recovery?

Instruction Replay (IR)



Style 2:
"Flush"

Flush

Style 1:
"Instruction Replay"

Cross-layer protected

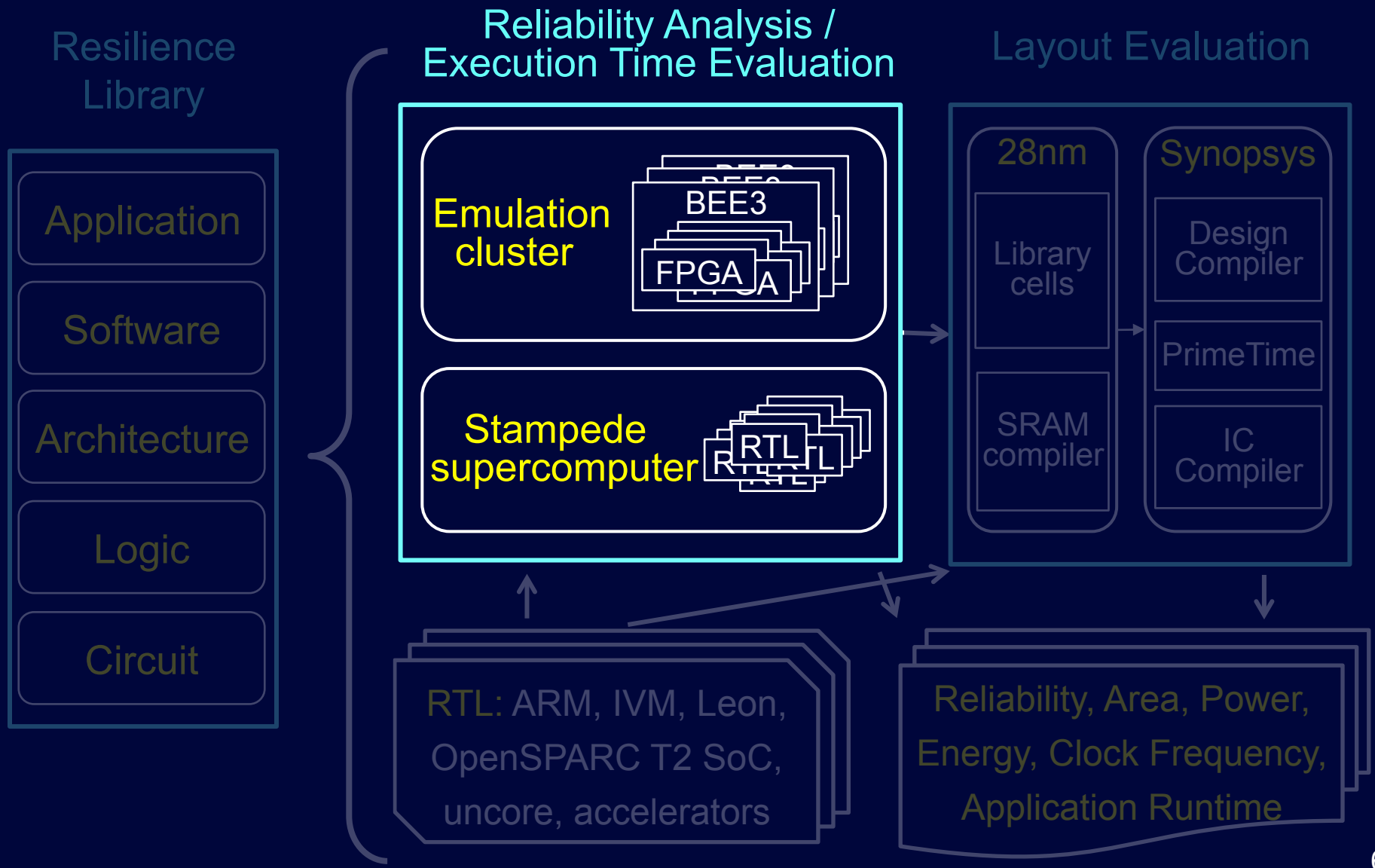
LEAP-DICE protected

Recovery logic

	Instruction Replay	Flush recovery
Overhead for recovery hardware	16% area, 21% energy	0.6% area, 0.9% energy
Recovery latency	47 cycles	7 cycles

CLEAR

Cross-Layer Exploration for Architecting Resilience

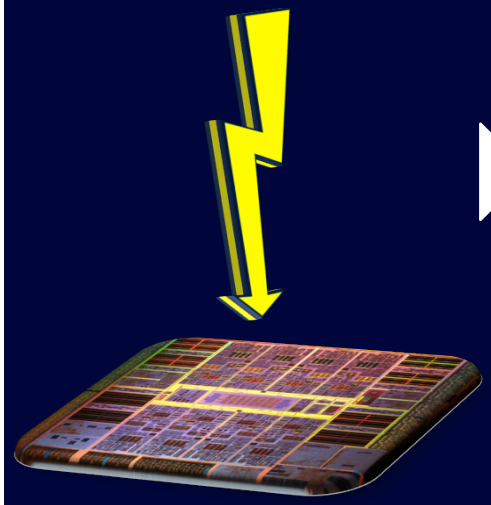


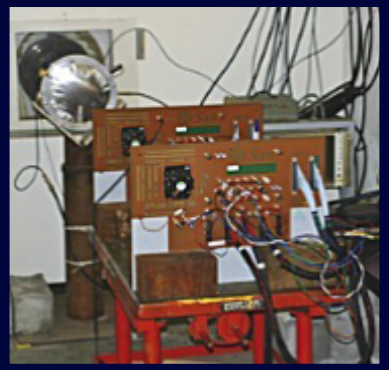
Radiation-Induced Soft Errors

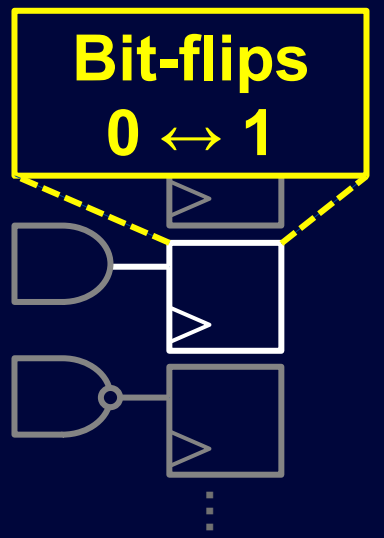
Soft errors

Radiation
beam testing

Flip-flop
error injection





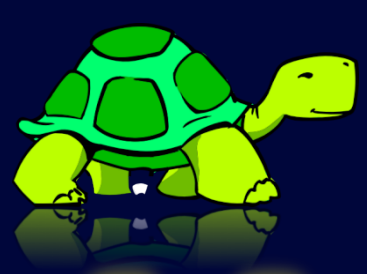


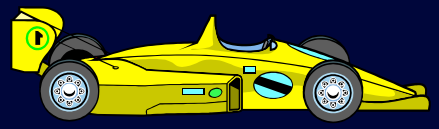
The Los Alamos
Neutron Science Center

Simulation /
Emulation

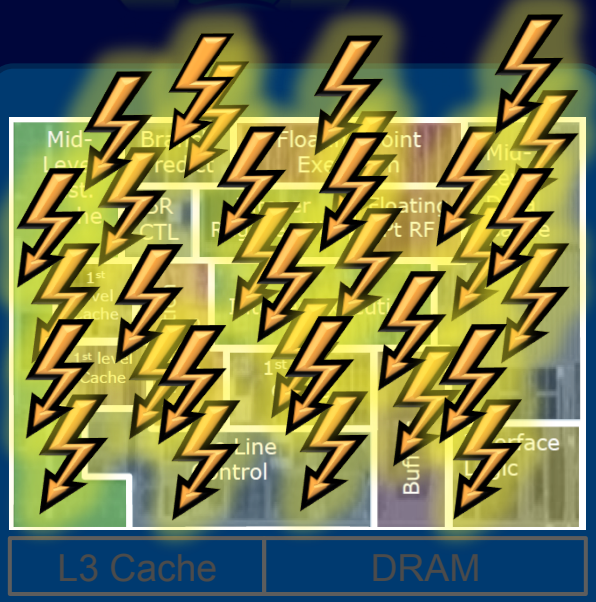
Soft Error Injection

Simulation speed



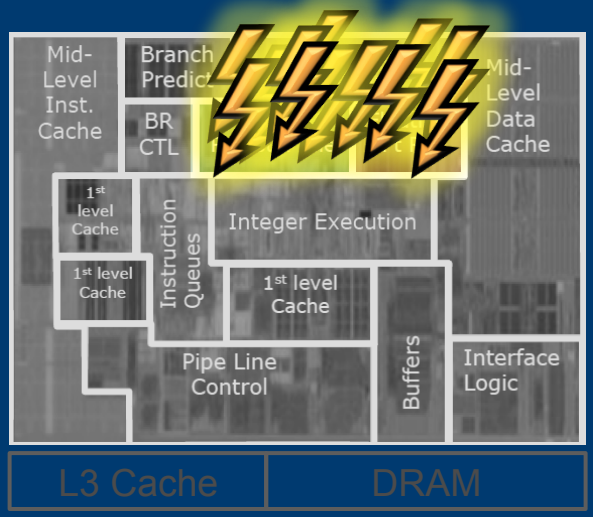


High-level error injection



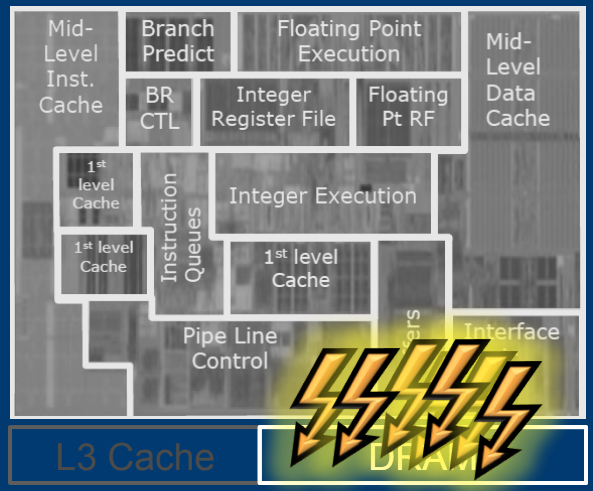
Flip-flop

10^2 cycles / sec



Architectural register

10^7 cycles / sec



Program variable

10^9 cycles / sec

Soft Error Injection

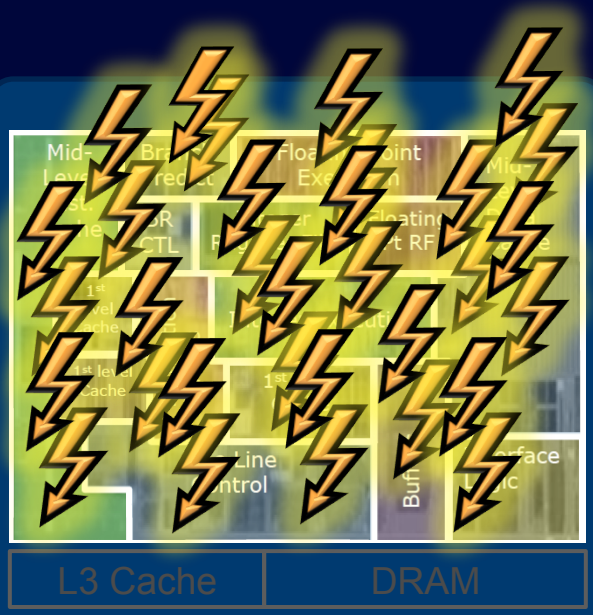
Accuracy

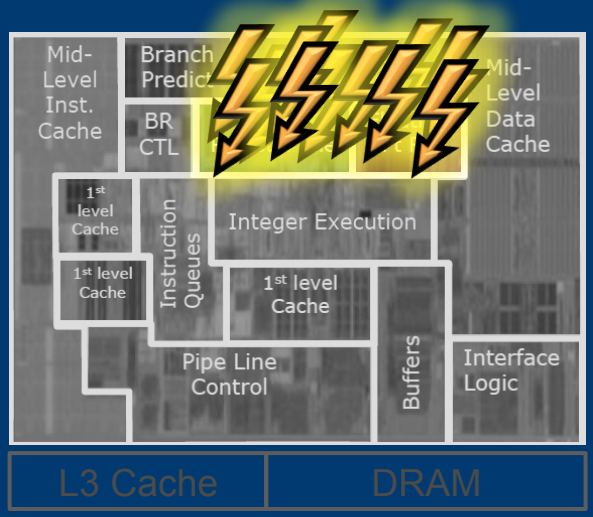
?

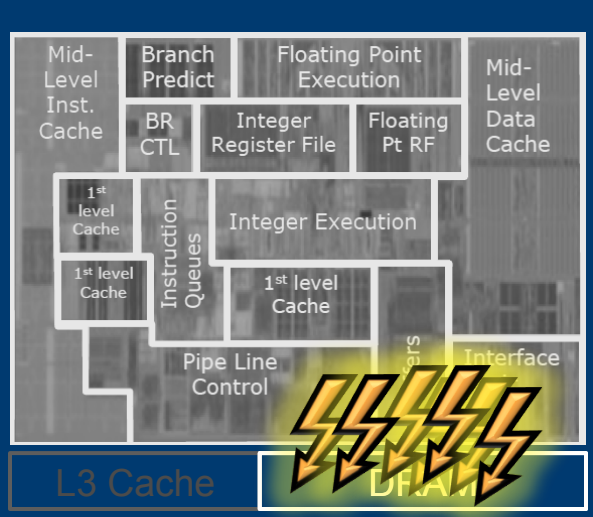
?

Ground truth

High-level error injection







Flip-flop

10^2 cycles / sec

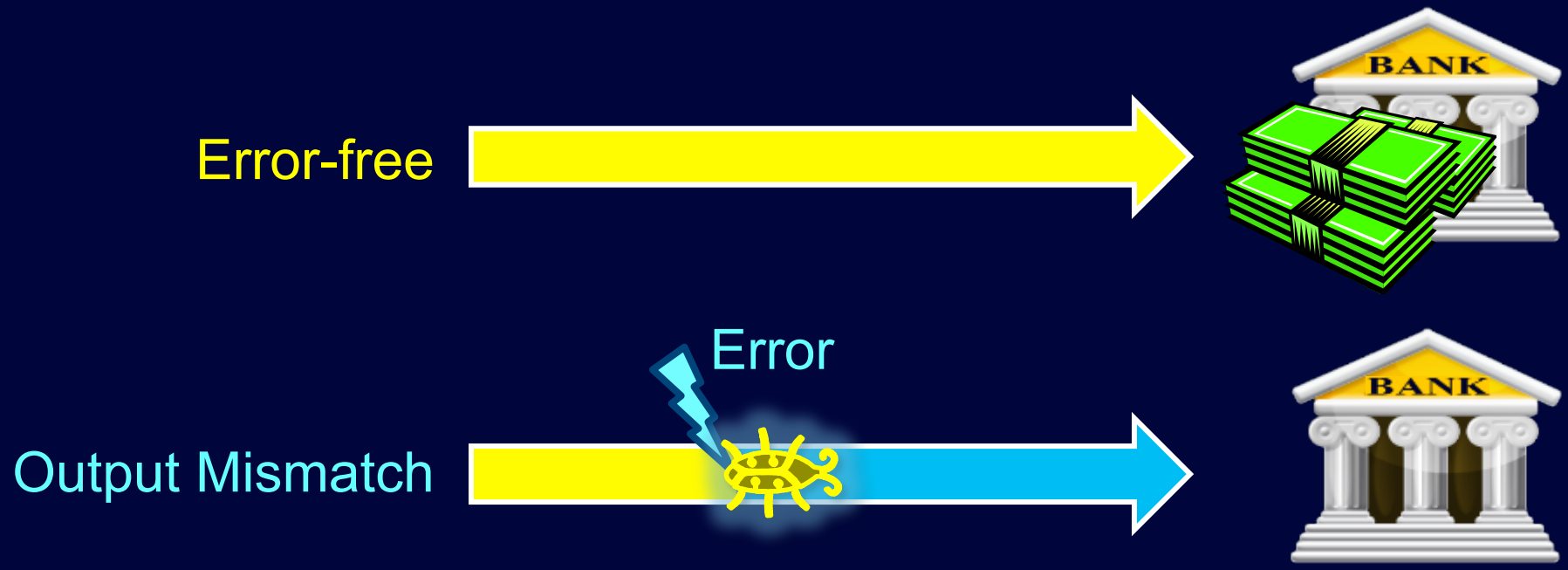
Architecture register

10^7 cycles / sec

Program variable

10^9 cycles / sec

Silent Data Corruption (SDC)



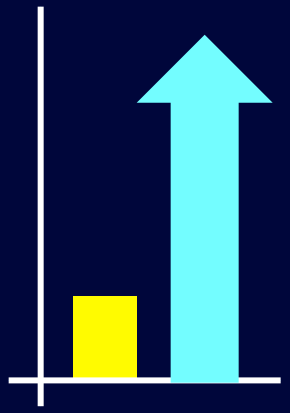
- Output file incorrect
- No error indication

Detected but Uncorrected Error (DUE) also considered

Perils of Inaccurate Estimation

Overestimation

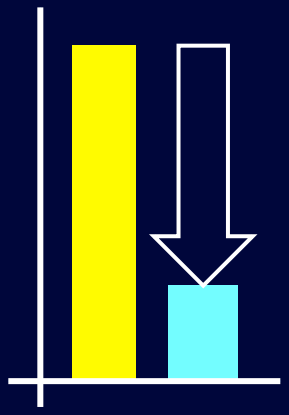
SDC



Overprotection

Underestimation

SDC



Unhappy customers

Little Prior Work

Error injection studies

Quantified comparison

[Rimen FTCS 94]

[Rebaudengo IOLTW 02]

What We Found

- Naïve high-level injections **highly** inaccurate

- How inaccurate?

What We Found

- Naïve high-level injections **highly** inaccurate

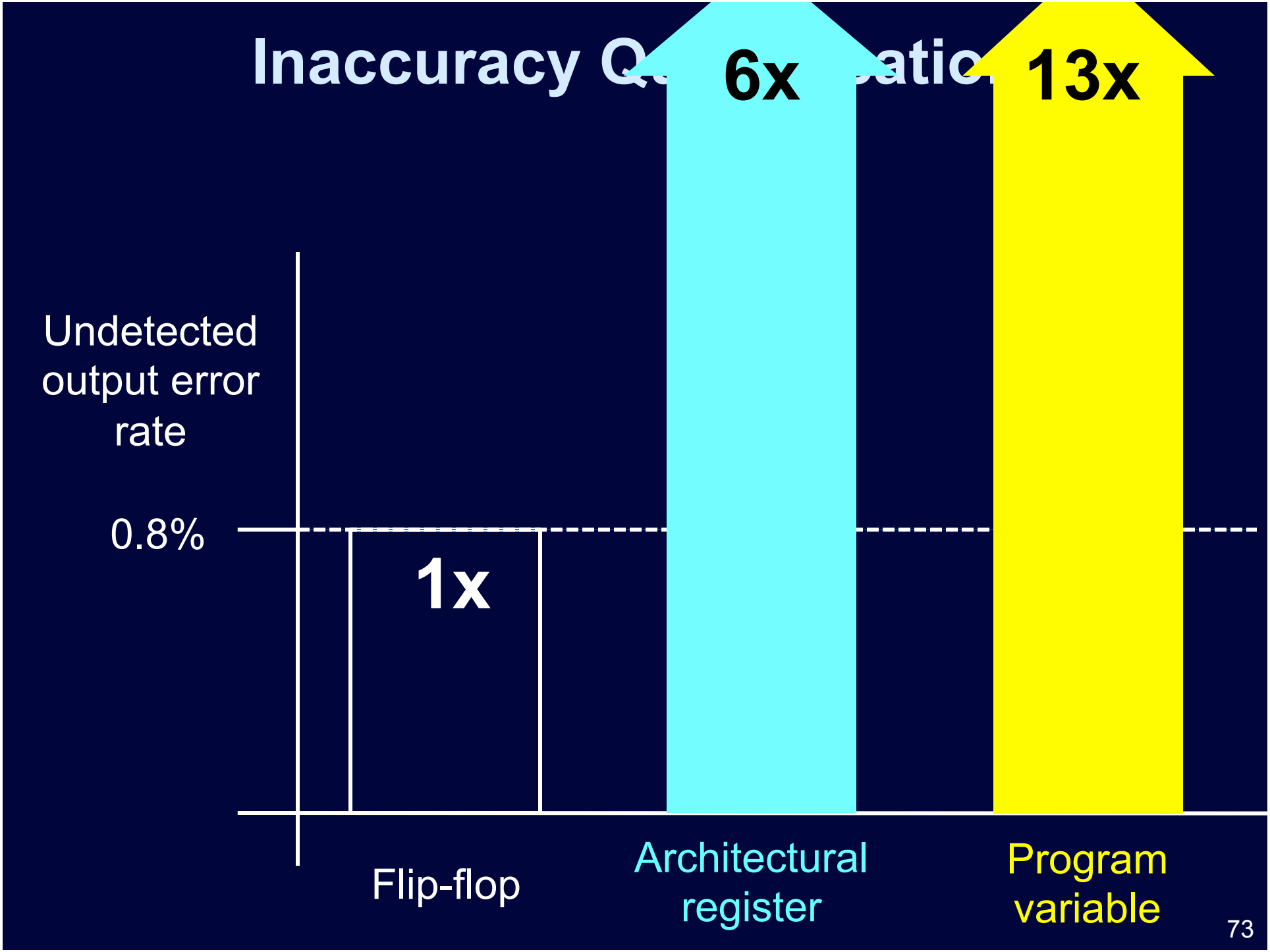
- How inaccurate?

Designs:

LEON3 (in-order, single-issue), ALPHA (out-of-order, superscalar)

Applications: SPEC 2000

Error injection samples: 6 million


Inaccuracy Q atic
6x
13x
Undetected output error rate
0.8%
1x
Flip-flop
Architectural register
Program variable

What We Found

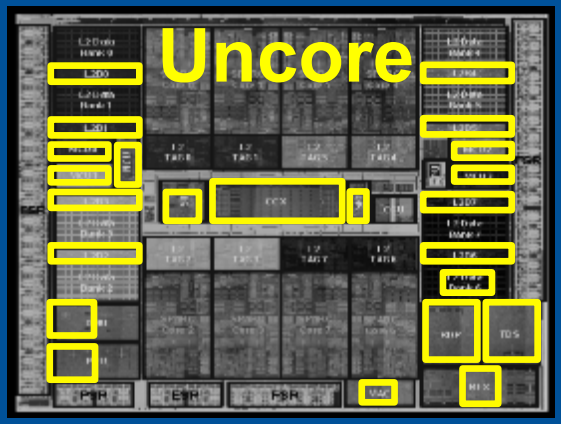
- Naïve high-level injections **highly** inaccurate
- How inaccurate?
  - Up to 45X
  - Neither optimistic nor pessimistic

What We Found

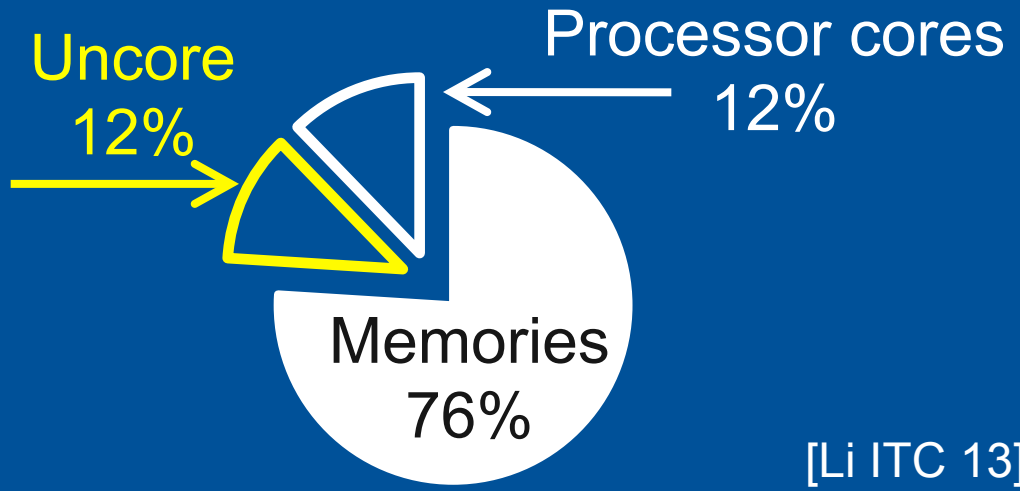
- Naïve high-level injections **highly** inaccurate
- How inaccurate?
  - Up to 45X
  - Neither optimistic nor pessimistic
- Why inaccurate?
  - Only 3% flip-flop error propagations modeled

Uncore Components

OpenSPARC T2 SoC

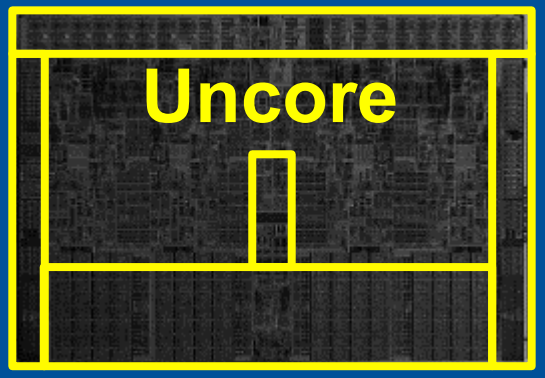


Area



[Li ITC 13]

Intel i7 quad-core SoC



Power

Processor cores 60.2%	Uncore 39.8%

[Gupta USENIX 12]

Existing Work

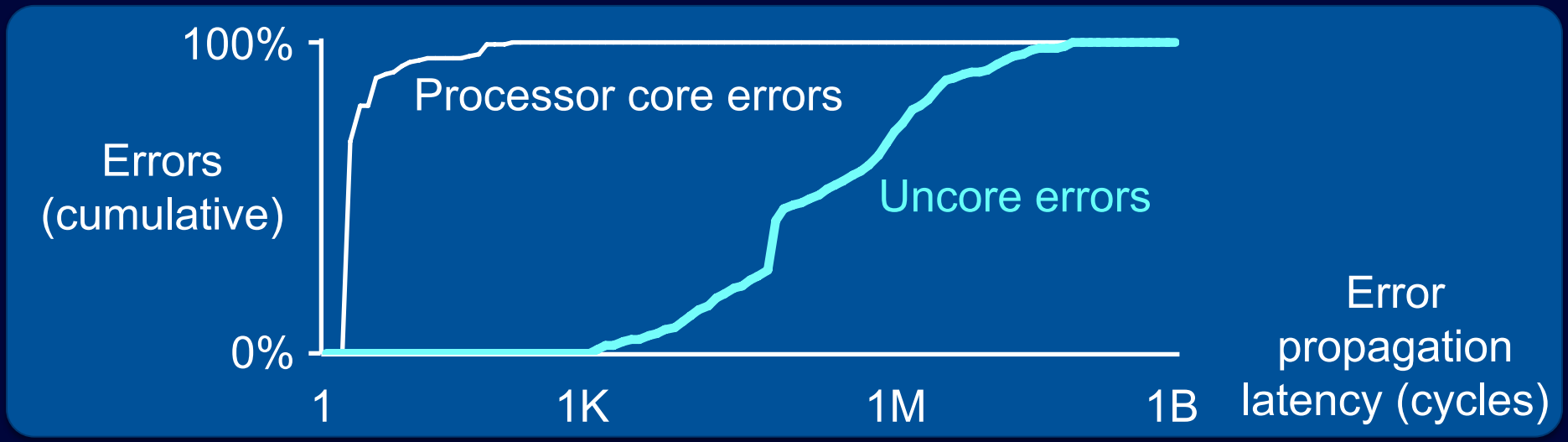
Errors in processor cores

HUNDREDs of publications

Errors in uncore?

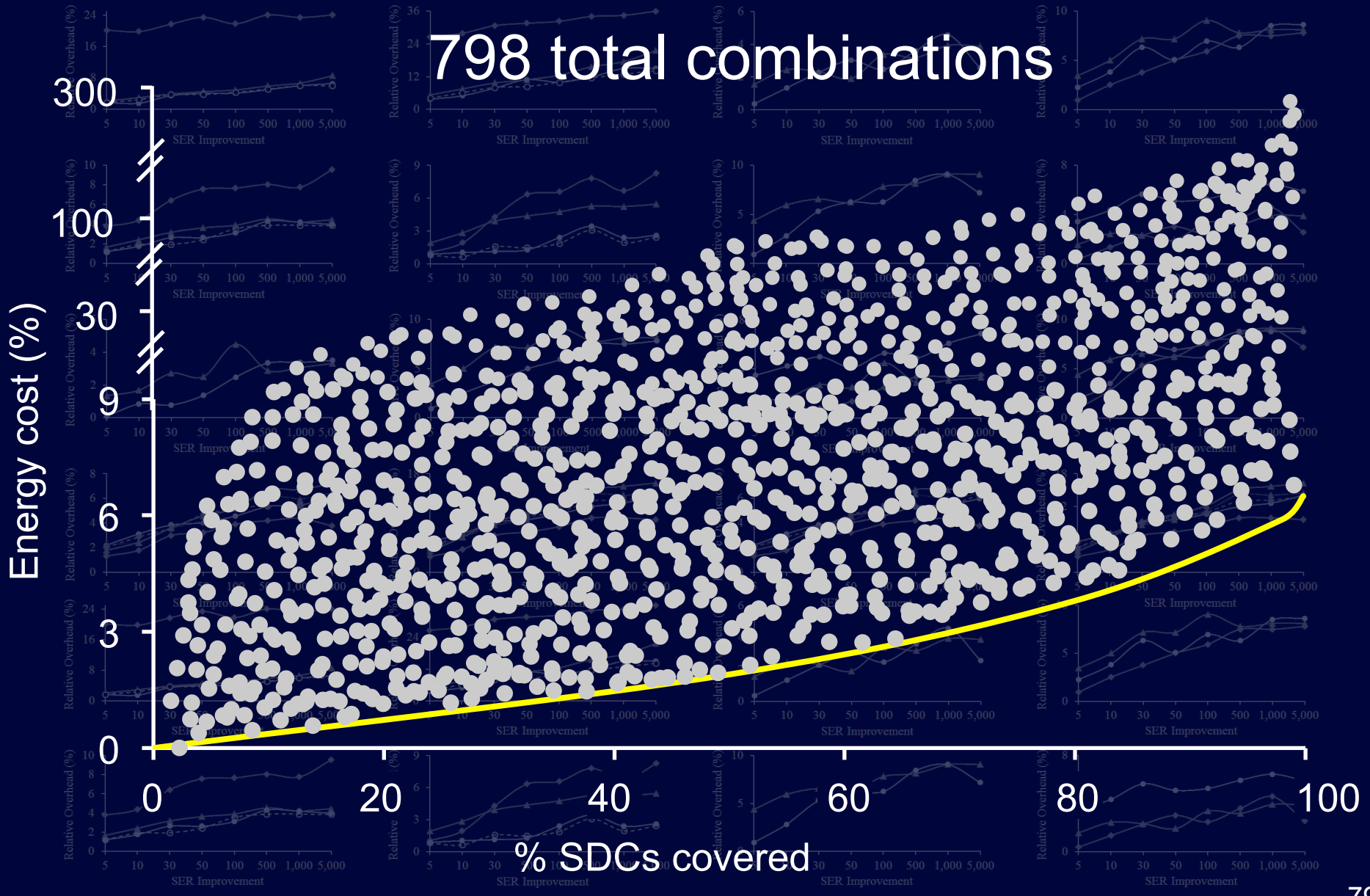
Uncore Soft Errors: First Extensive Study

- New error injection: fast **&** accurate
  - 20,000x speedup vs. RTL
- Reliability impact: uncore ≈ processor cores
  - BUT, **long** error propagation latency

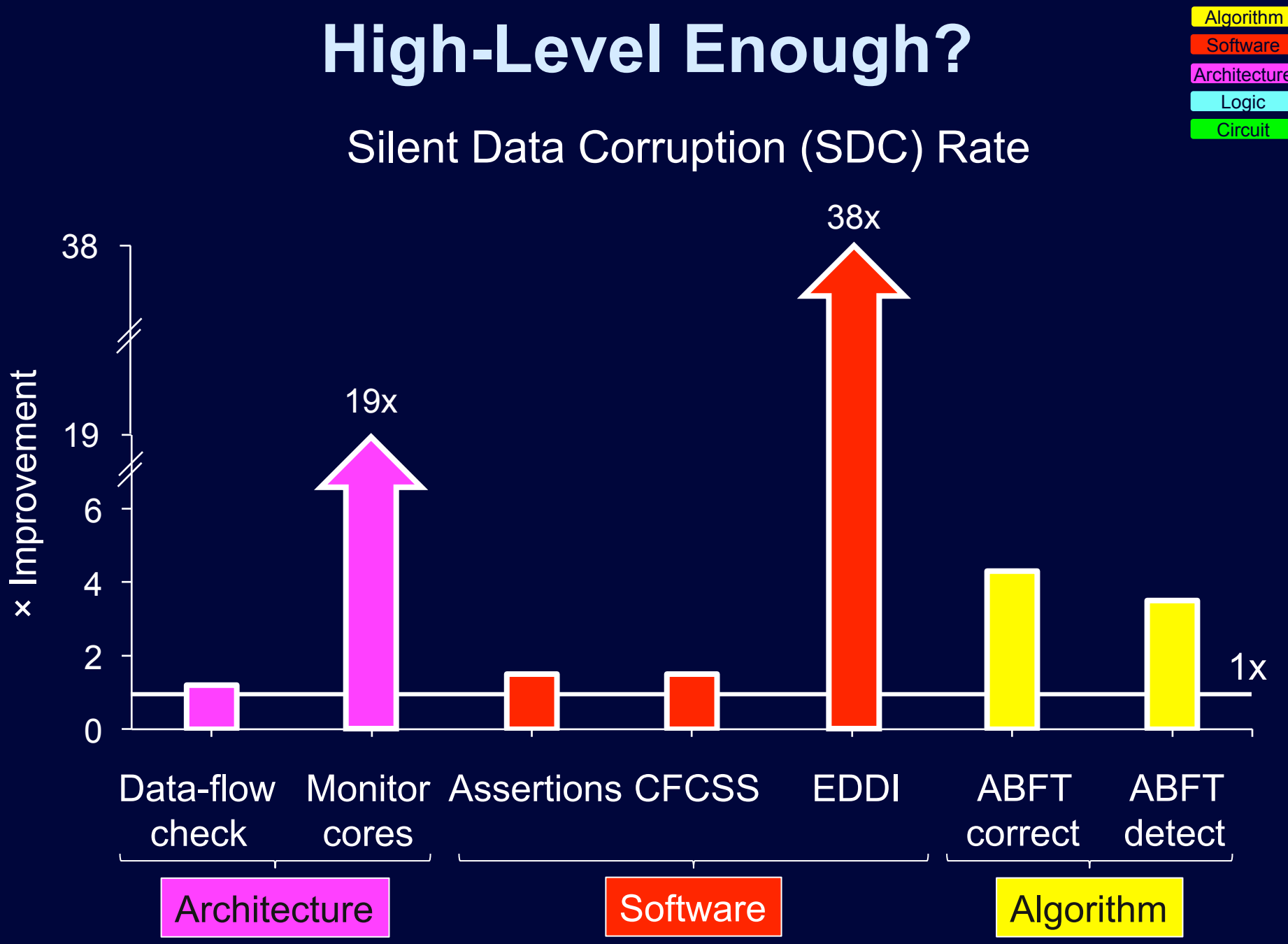


[Cho DAC 15]

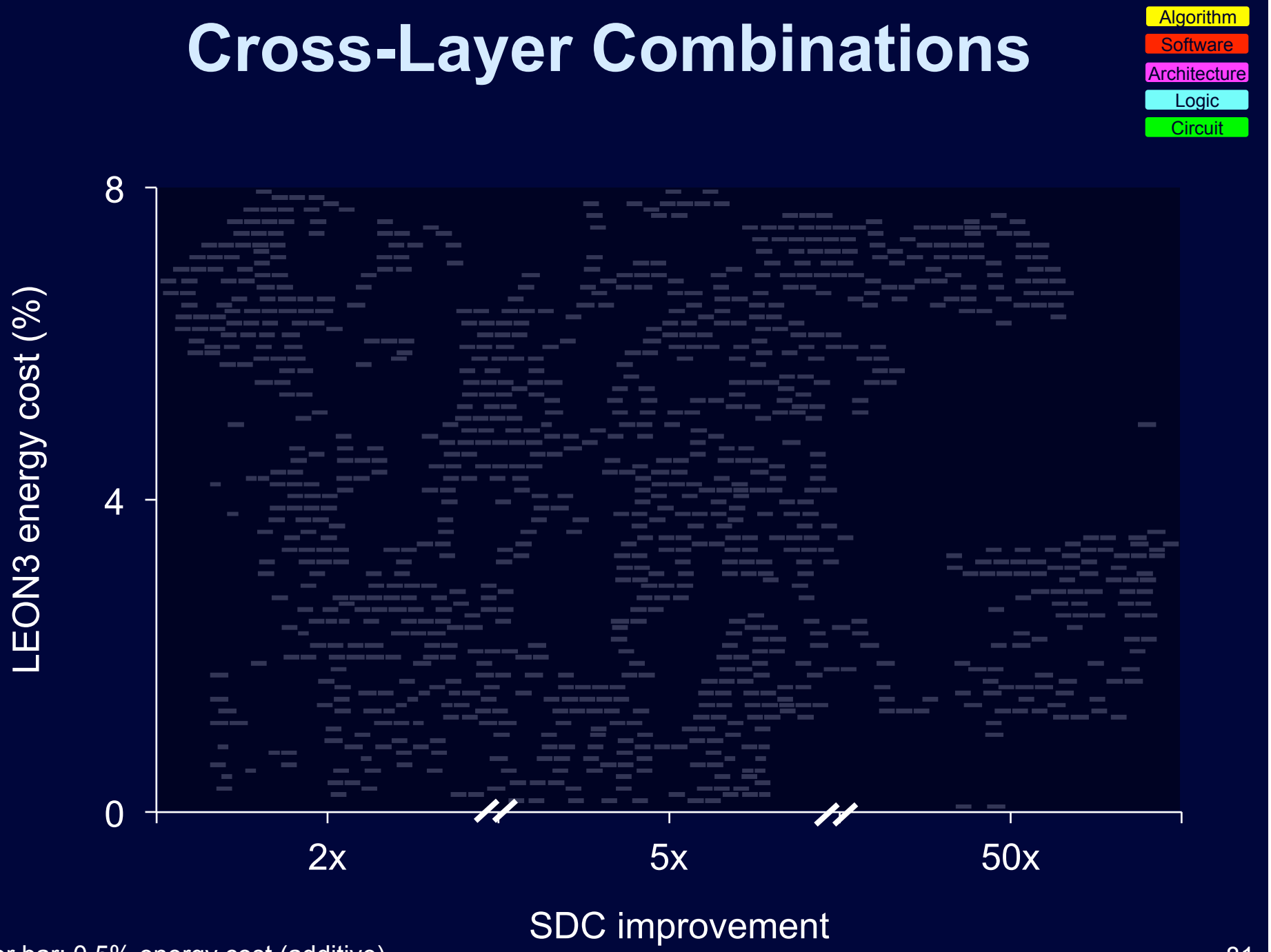
Lots of CLEAR Results



High-Level Enough?

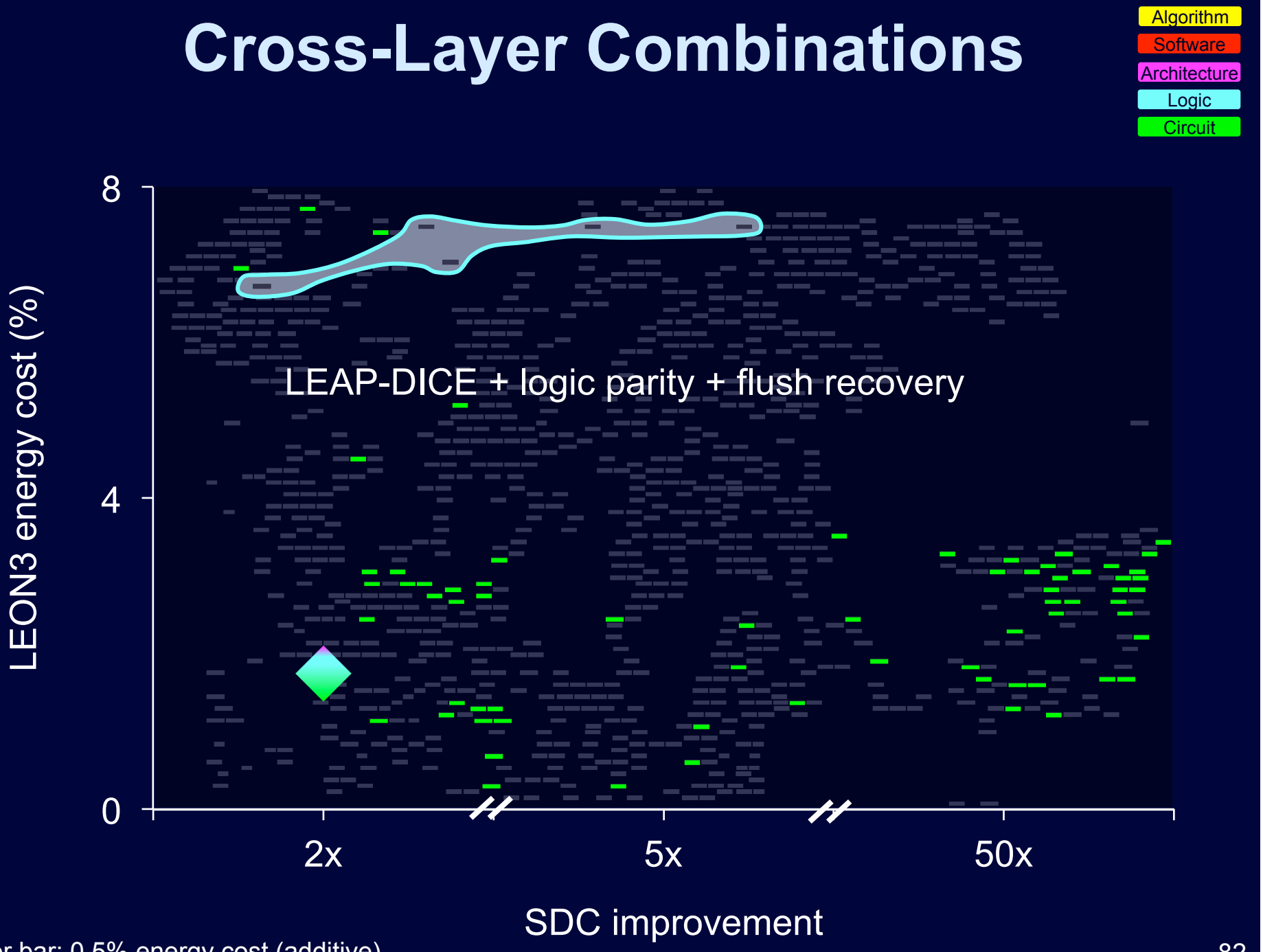


Cross-Layer Combinations



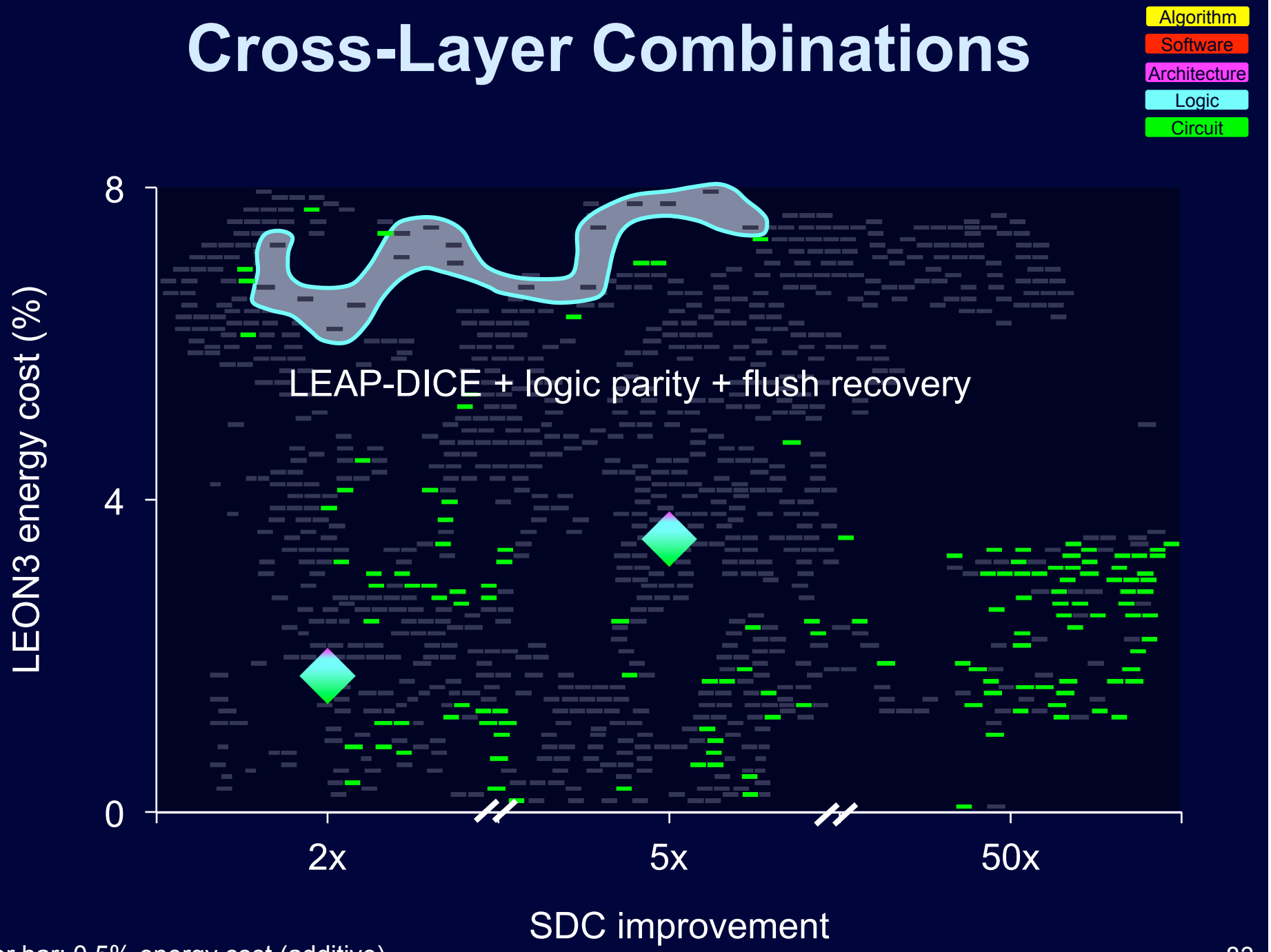
Error bar: 0.5% energy cost (additive)

Cross-Layer Combinations



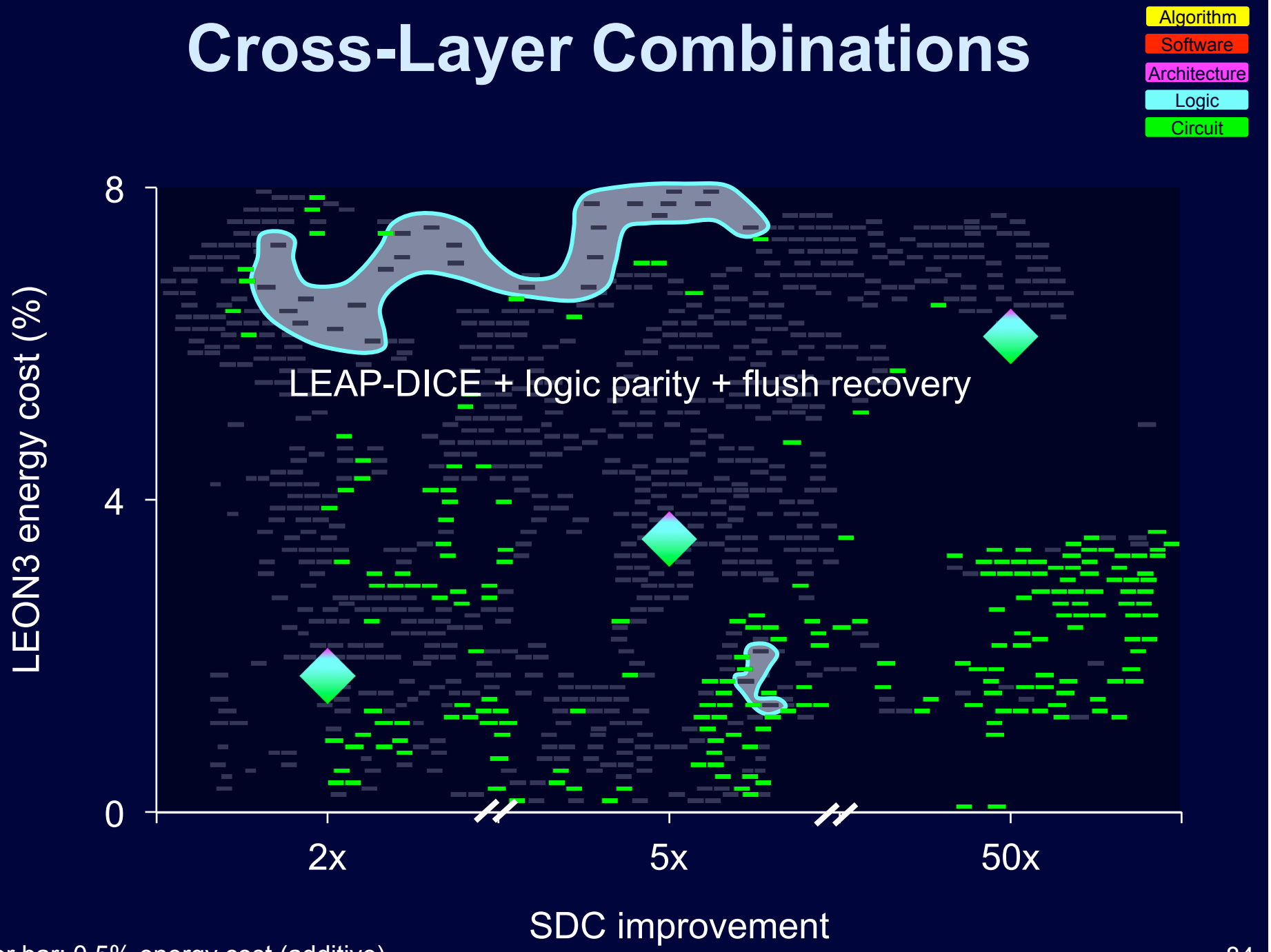
Error bar: 0.5% energy cost (additive)

Cross-Layer Combinations



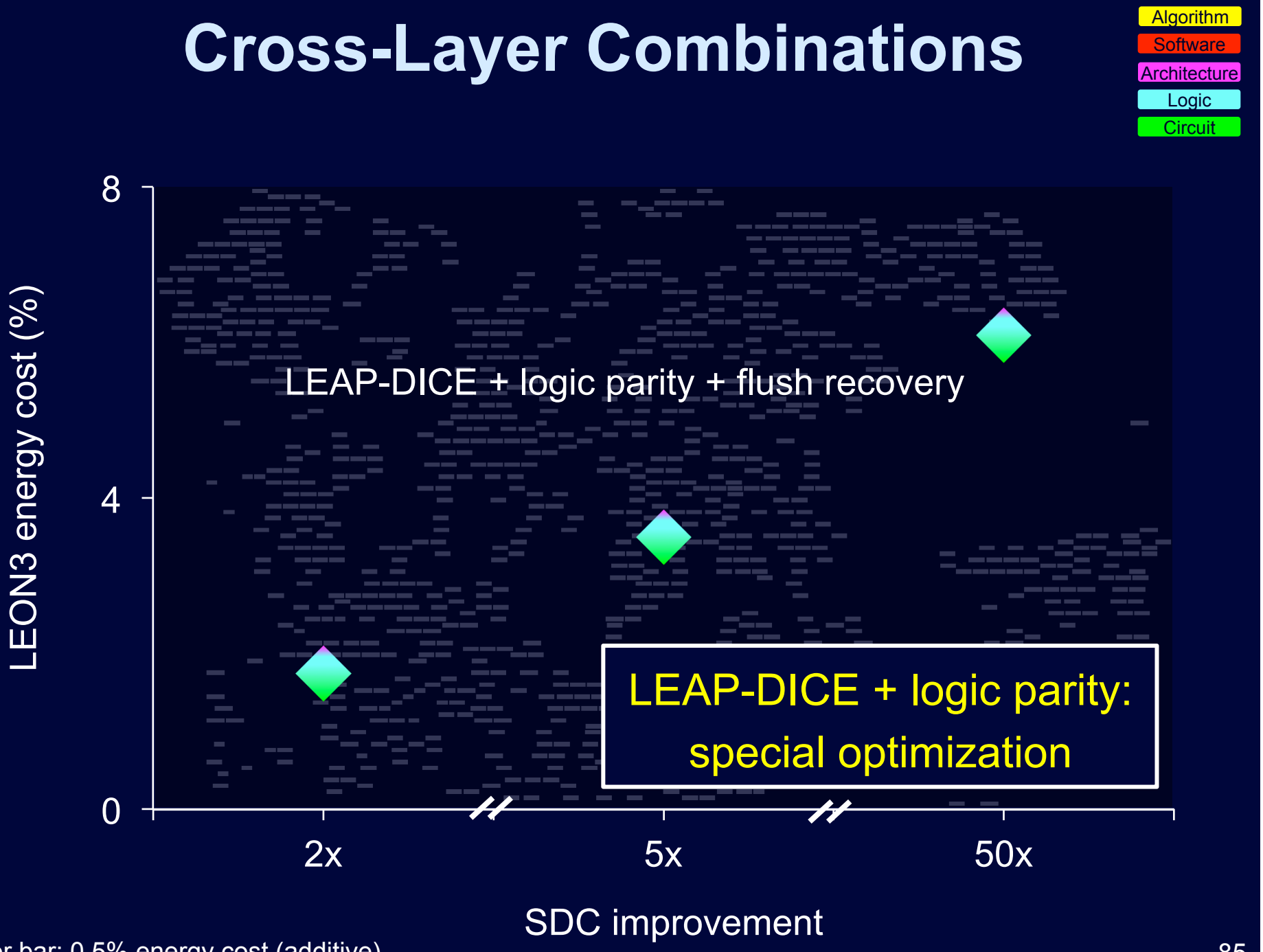
Error bar: 0.5% energy cost (additive)

Cross-Layer Combinations



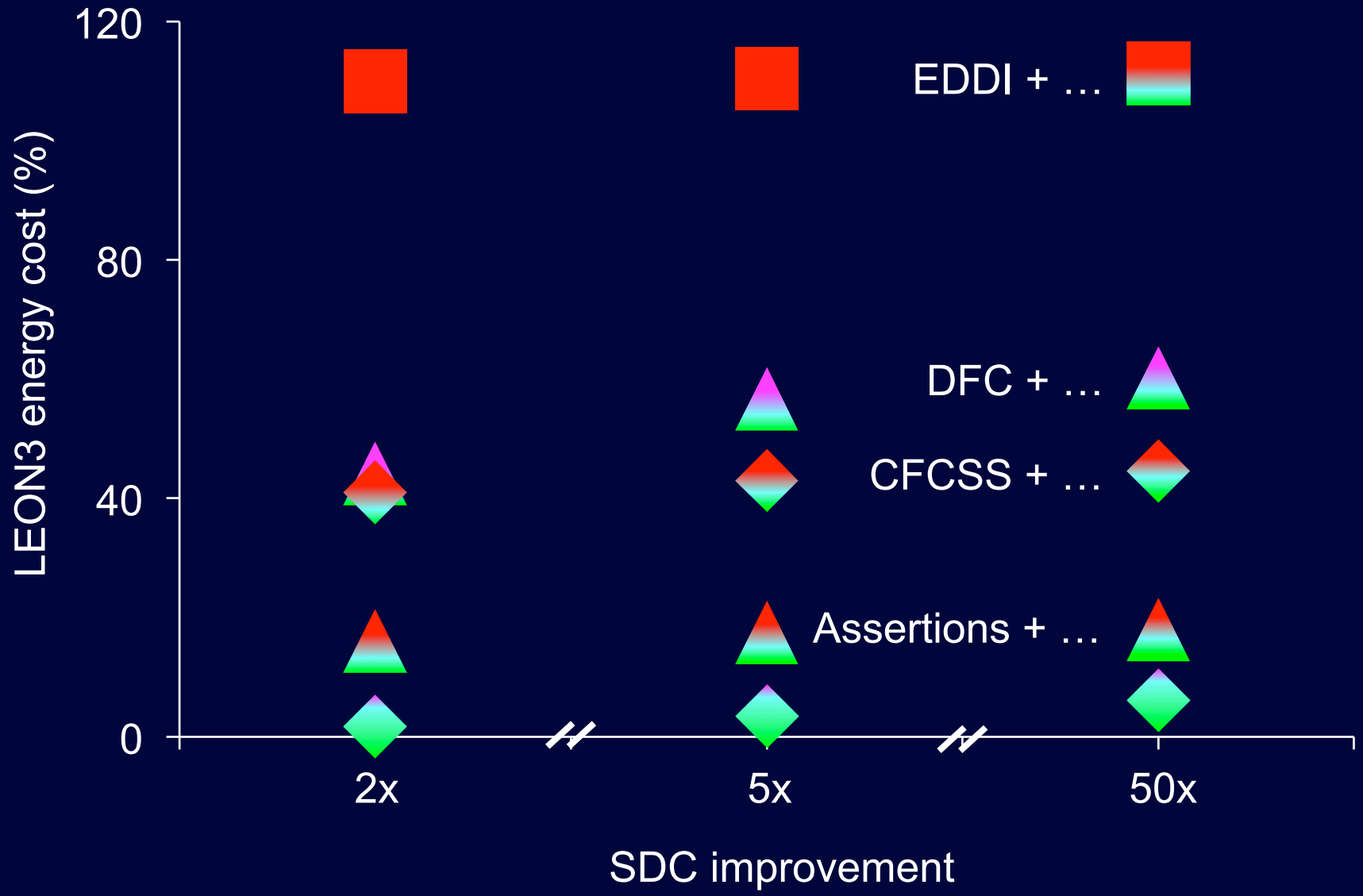
Error bar: 0.5% energy cost (additive)

Cross-Layer Combinations

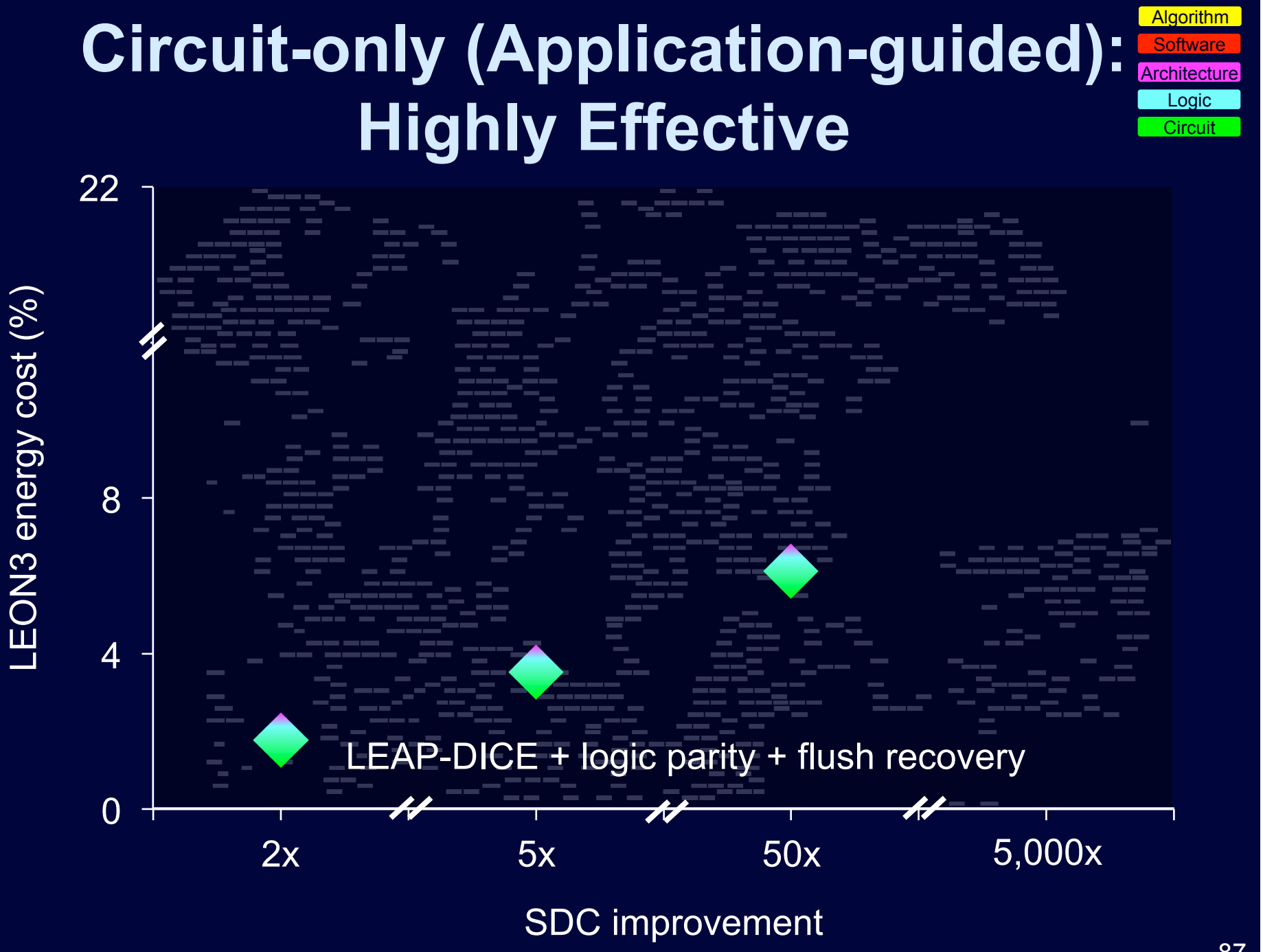


Error bar: 0.5% energy cost (additive)

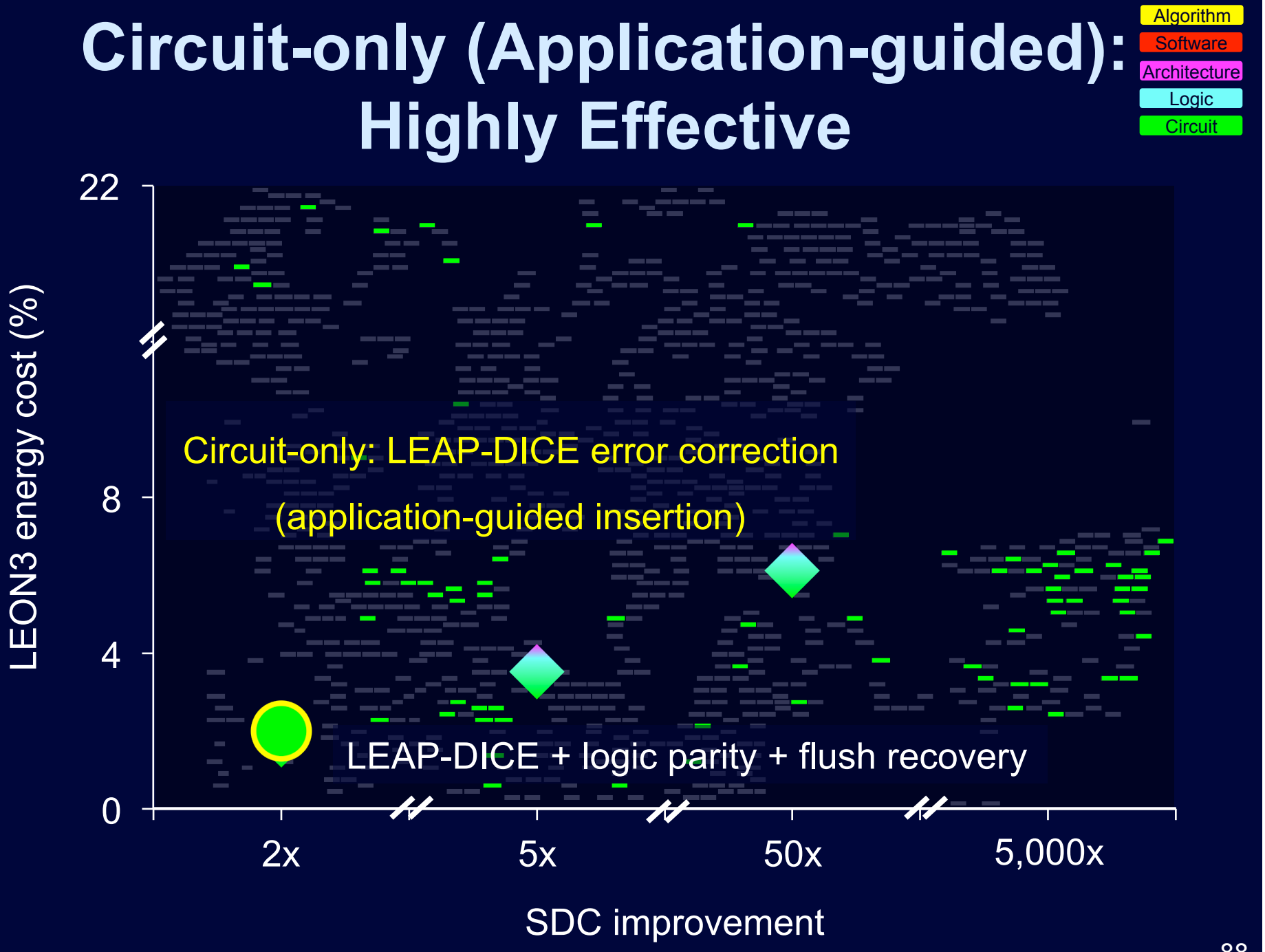
Architecture & Software: Too Expensive



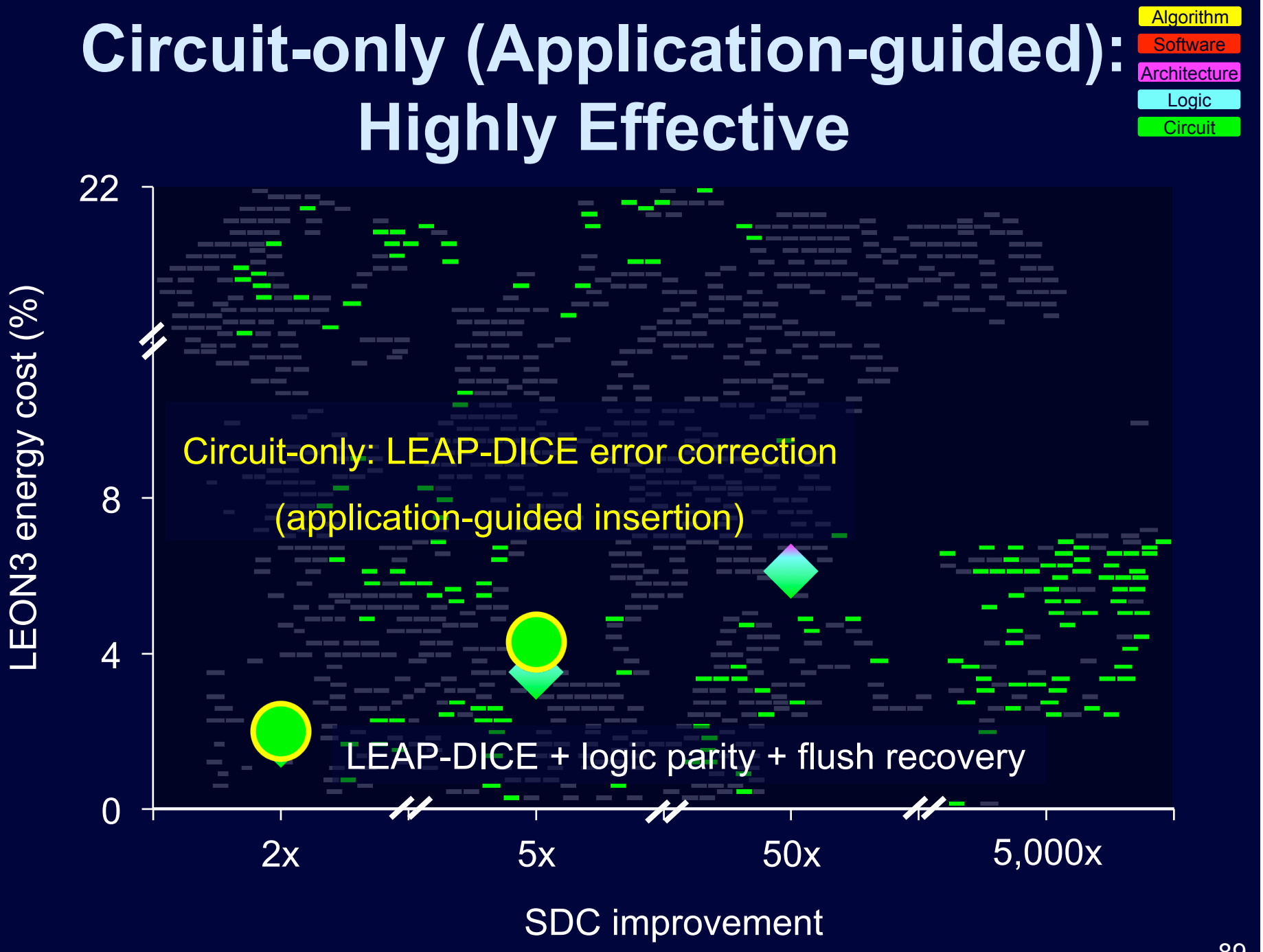
Circuit-only (Application-guided): Highly Effective



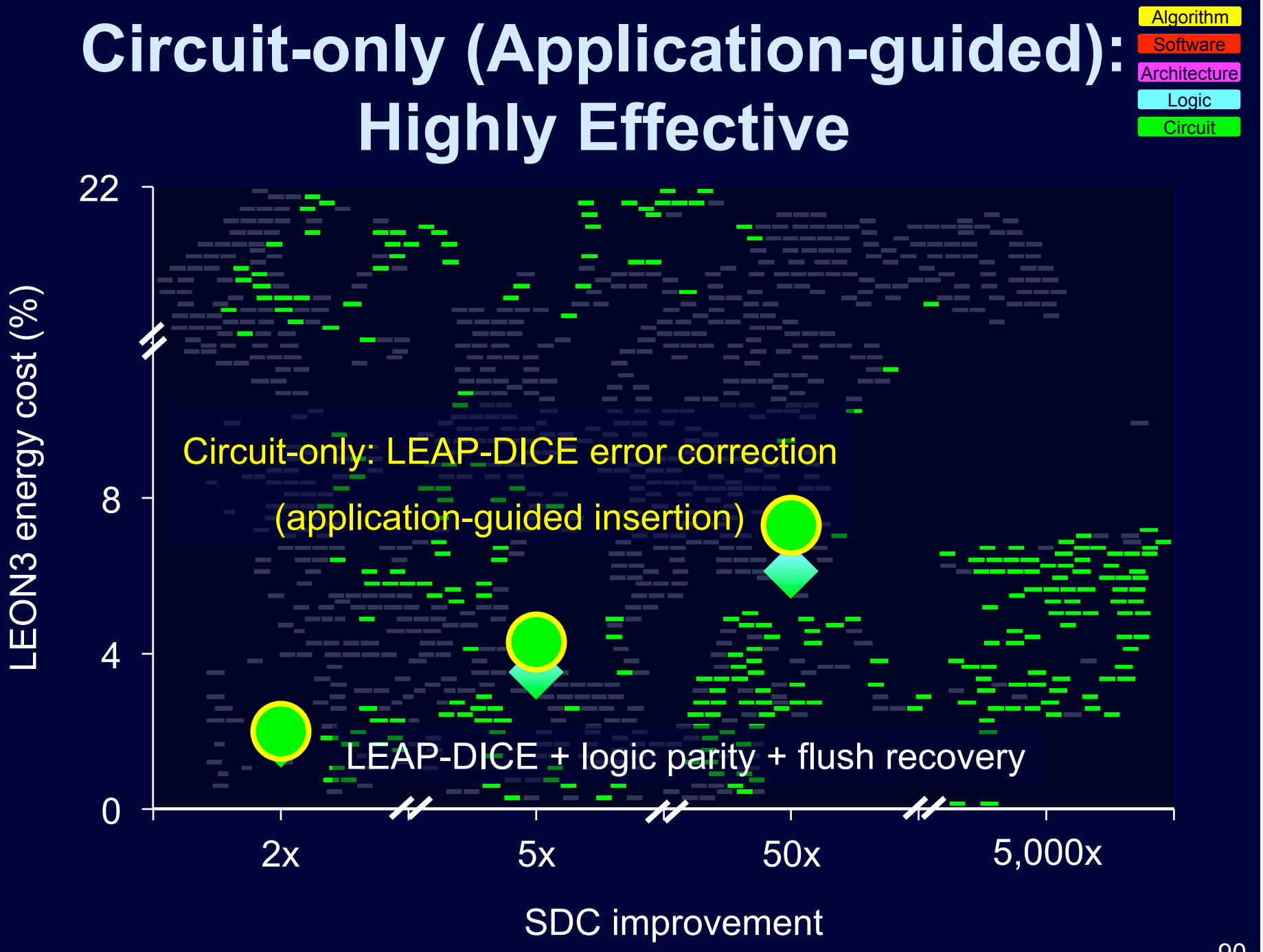
Circuit-only (Application-guided): Highly Effective



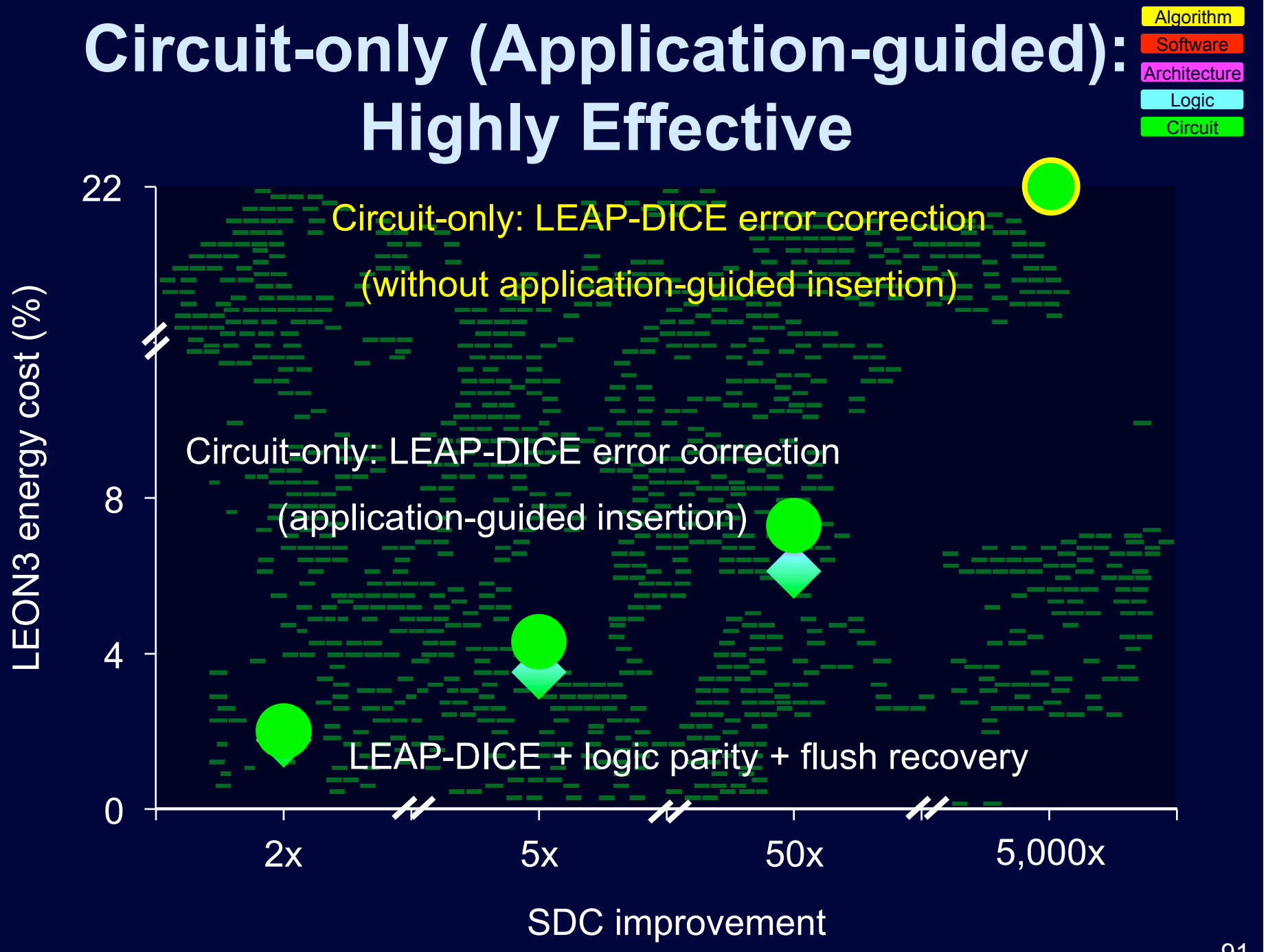
Circuit-only (Application-guided): Highly Effective



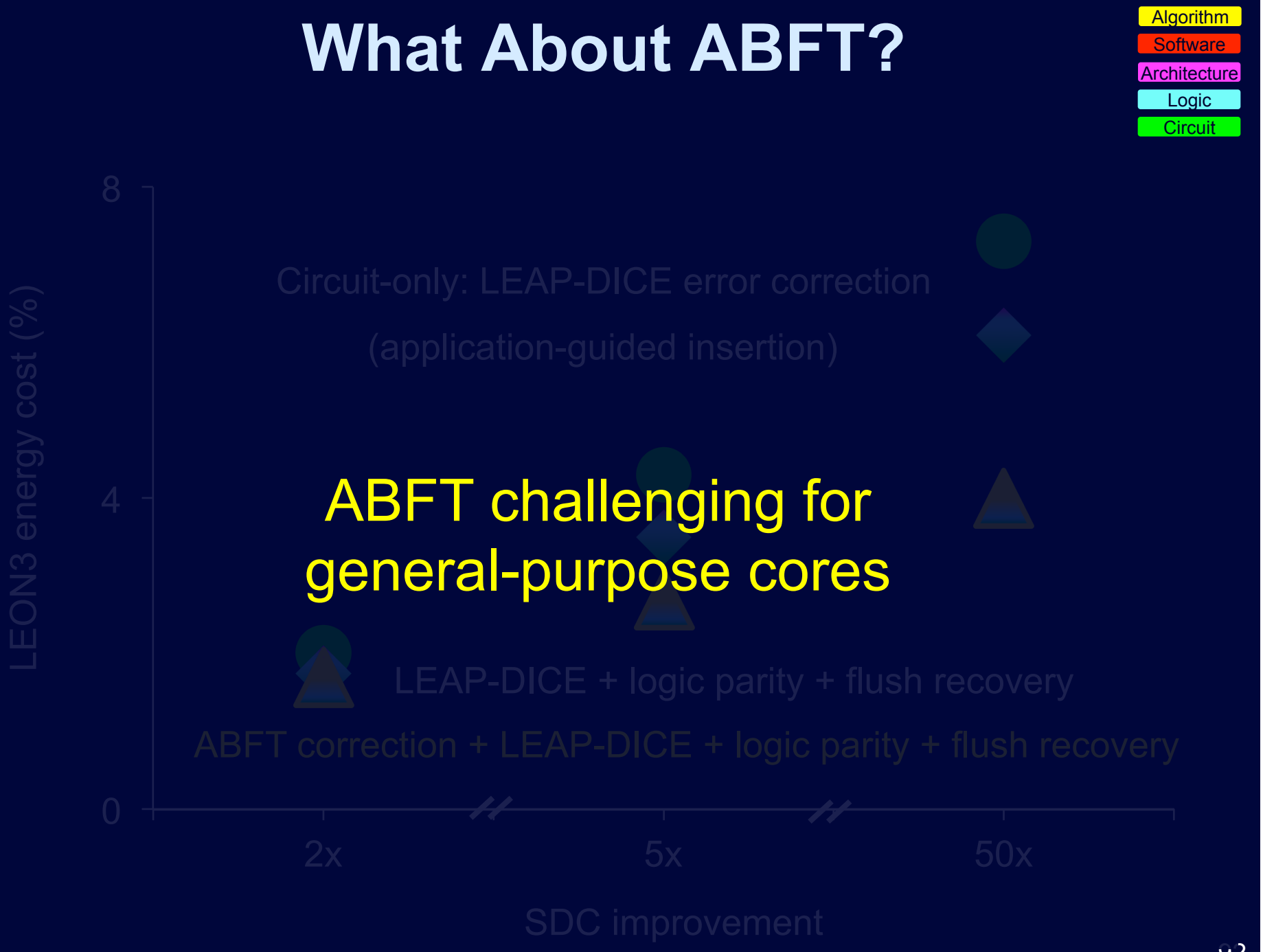
Circuit-only (Application-guided): Highly Effective



Circuit-only (Application-guided): Highly Effective



What About ABFT?

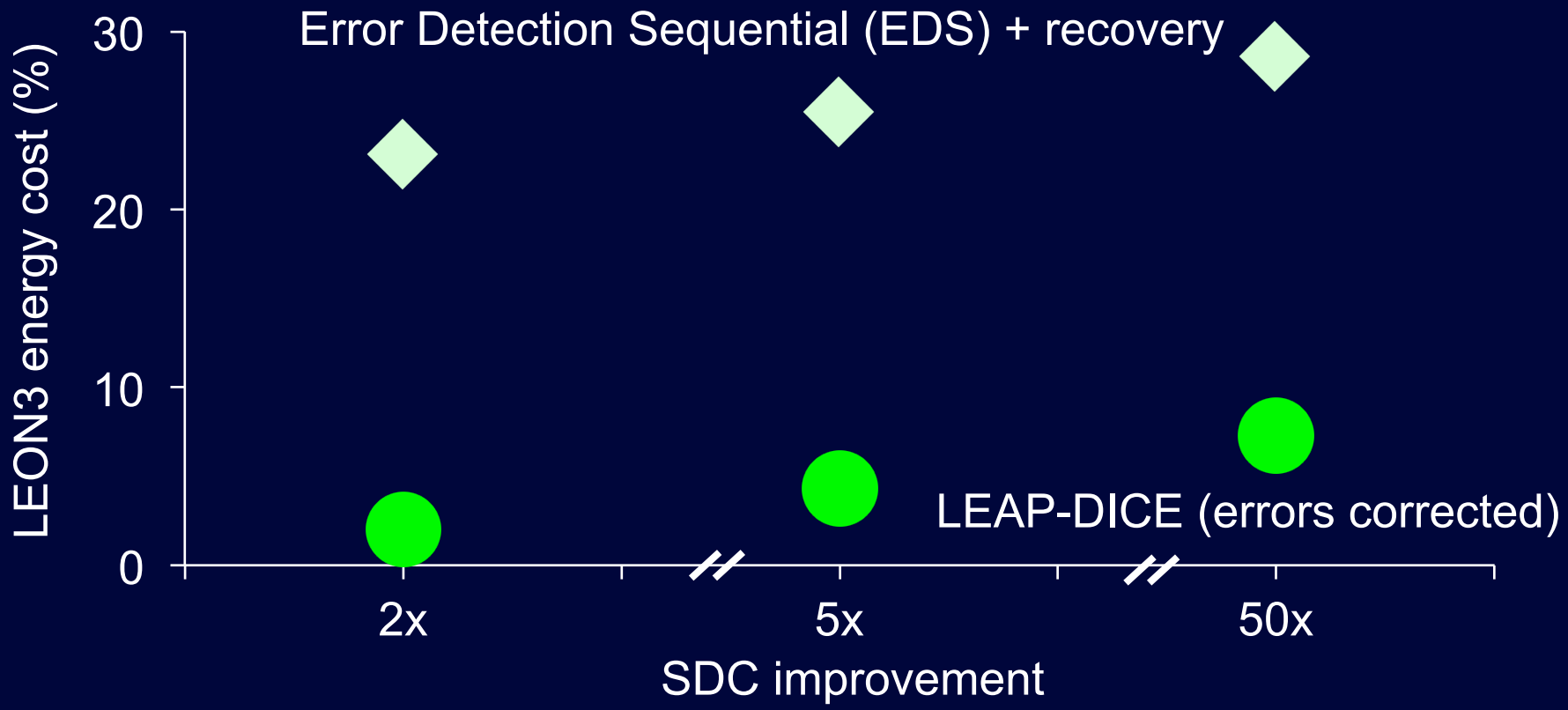


ABFT challenging for general-purpose cores

CLEAR Insights

- Hidden costs & inefficiencies
- Implementation matters
- Inaccurate analysis

Example: "Hidden" Costs



Flip-flop	Area	Energy
Nominal	1	1
LEAP-DICE	2	1.8
EDS	1.5	1.4

Not just flip-flop overhead
Routing, recovery impact

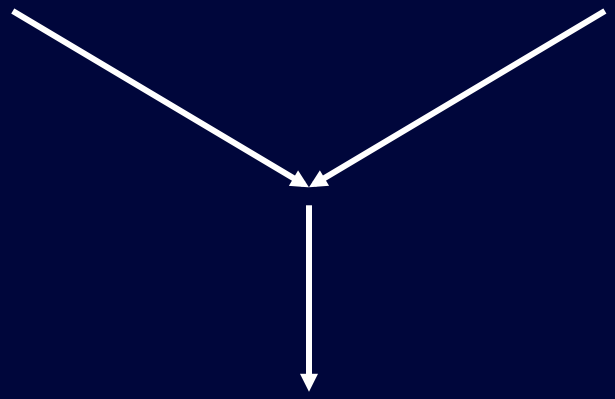
Example: Inefficiencies

Few Flip-flops Protected

Data Flow Checking (DFC)
57%

Low SDC Coverage per Flip-flop

Data Flow Checking (DFC)
30%



Result: Low SDC Improvement

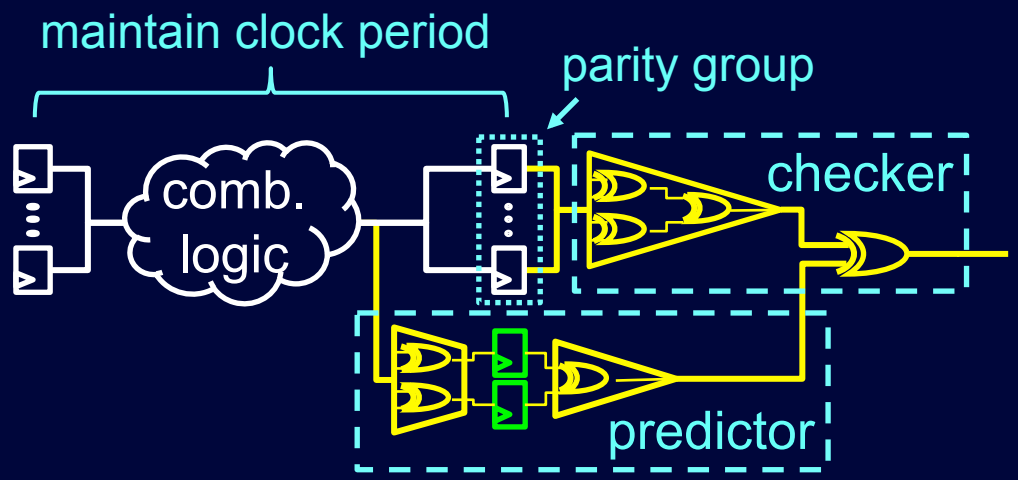
Data Flow Checking (DFC)
1.2x

[Meixner, MICRO 07]

Example: Implementation Matters

Logic parity

- Original design
- Parity logic
- Pipeline flip-flops



Logic parity: Naïve	200 MHz clock speed impact
Logic parity: Incorrect heuristic	80% additional energy impact
Logic parity: CLEAR heuristic	No clock speed impact Minimal energy impact

Parameters: parity size, flip-flop vulnerability, floorplan location, timing path slack, etc.

Example: Inaccurate Analysis

- Software assertions: SDC improvement
  - Prior publications: 3.9x
    - Inaccurate error injection
  - Accurate analysis: 1.5x

	Flip-Flop error injection	Register Uniform error injection
SDC improvement	1.5x	4.8x

[Sahoo DSN 08, Hari DSN 12]

How About Benchmark Dependence?

- 50 <*training, evaluation*> pairs
  - Training: 4 SPEC, Evaluation: 7 SPEC

Trained SDC improvement	5x	50x	500x
Evaluated SDC improvement	4.8x	39x	433x

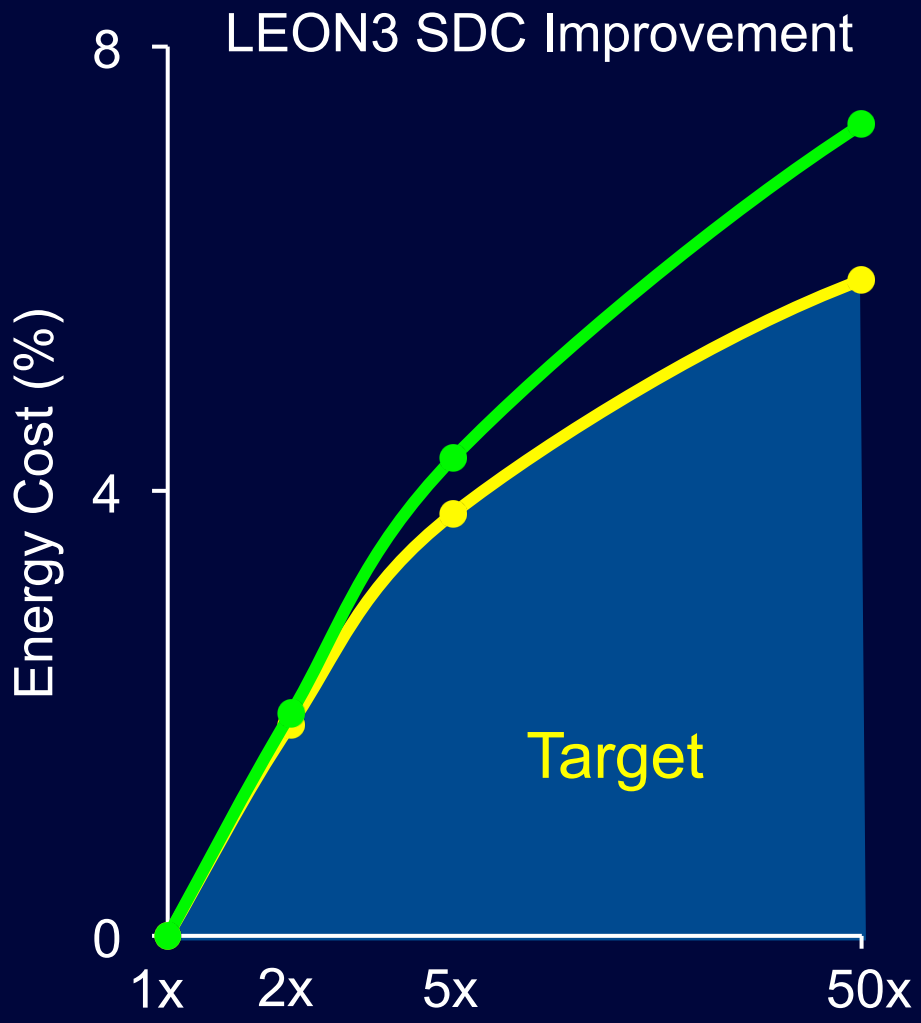
Add “lightweight” hardening (e.g., LHL)

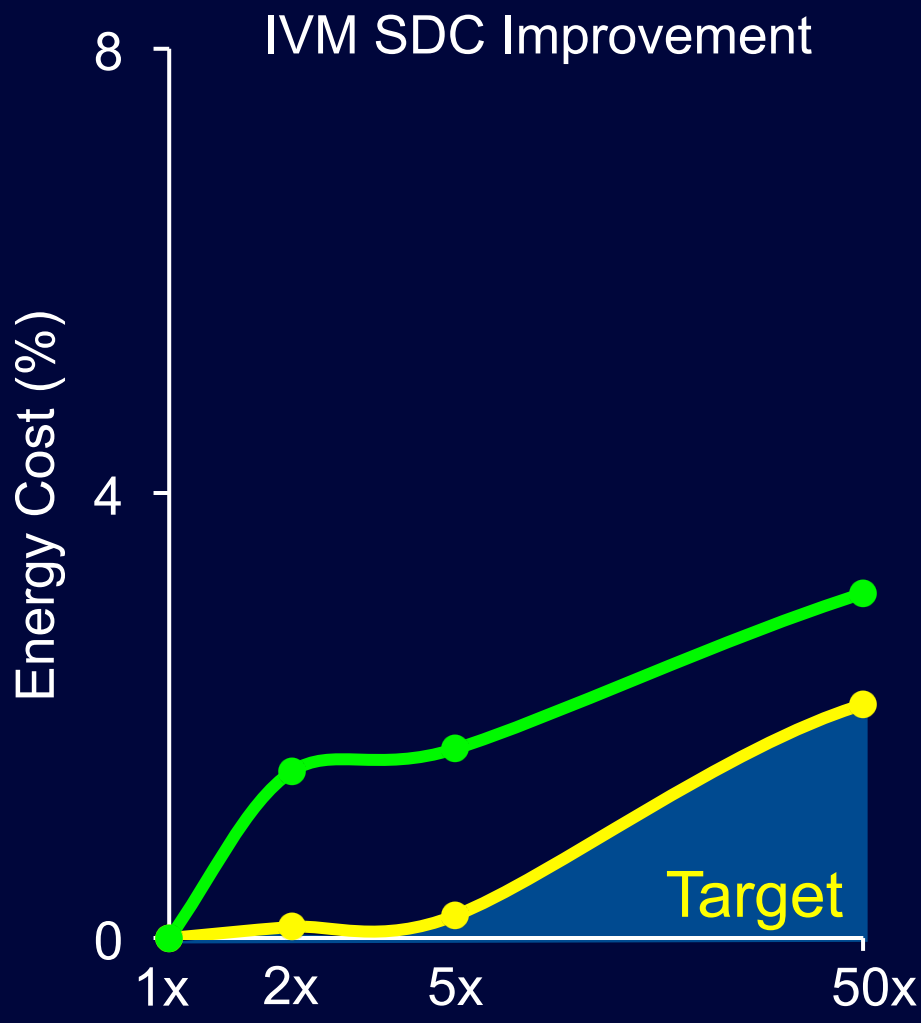
Extra energy cost (additive)	2%	1%	0.8%
Final SDC improvement	19x	152x	1,326x

Light-Hardened LEAP (LHL)

Flip-Flop	Soft Error Rate (SER)	Area	Power	Delay	Energy
Baseline	1	1	1	1	1
LHL	2.5×10^{-1}	1.2	1.1	1.2	1.3
LEAP-DICE	2×10^{-4}	2	1.8	1	1.8

Target for Future Resilience Techniques





LEAP-DICE
error correction

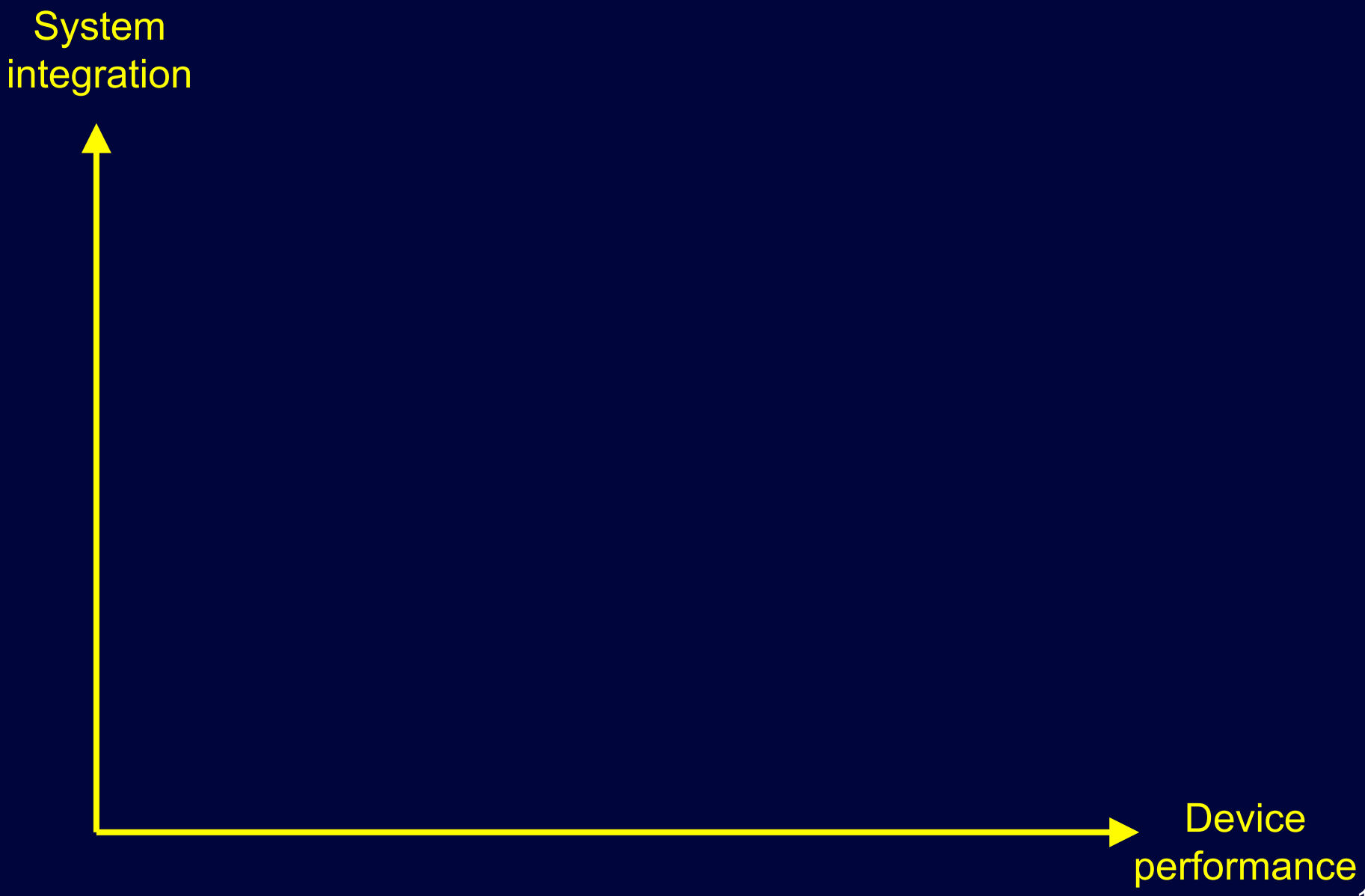
LEAP-DICE + logic parity
+ flush recovery

Outline

- Robust operation: silicon CMOS reliability
- Beyond silicon
- Conclusion

US National Academy of Sciences (2011)

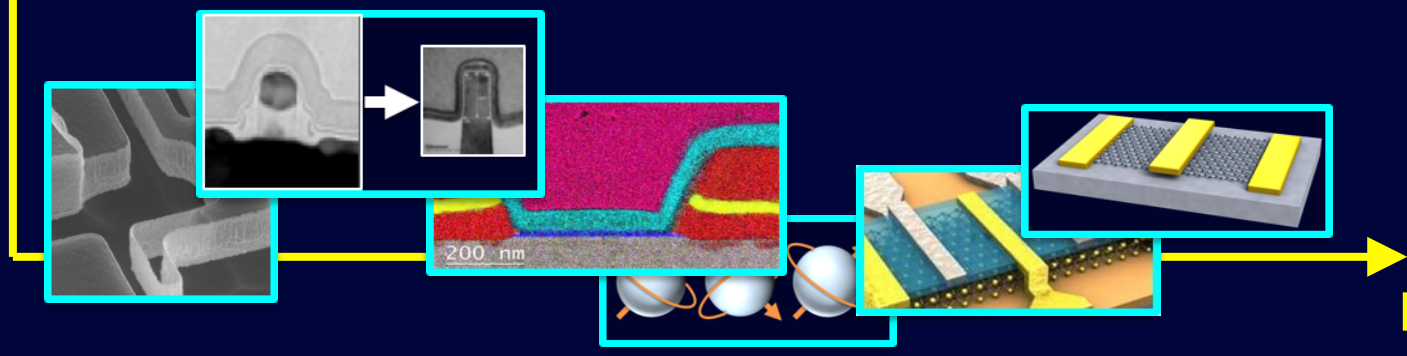
Improve Computing Performance



Option 1: Better Transistors

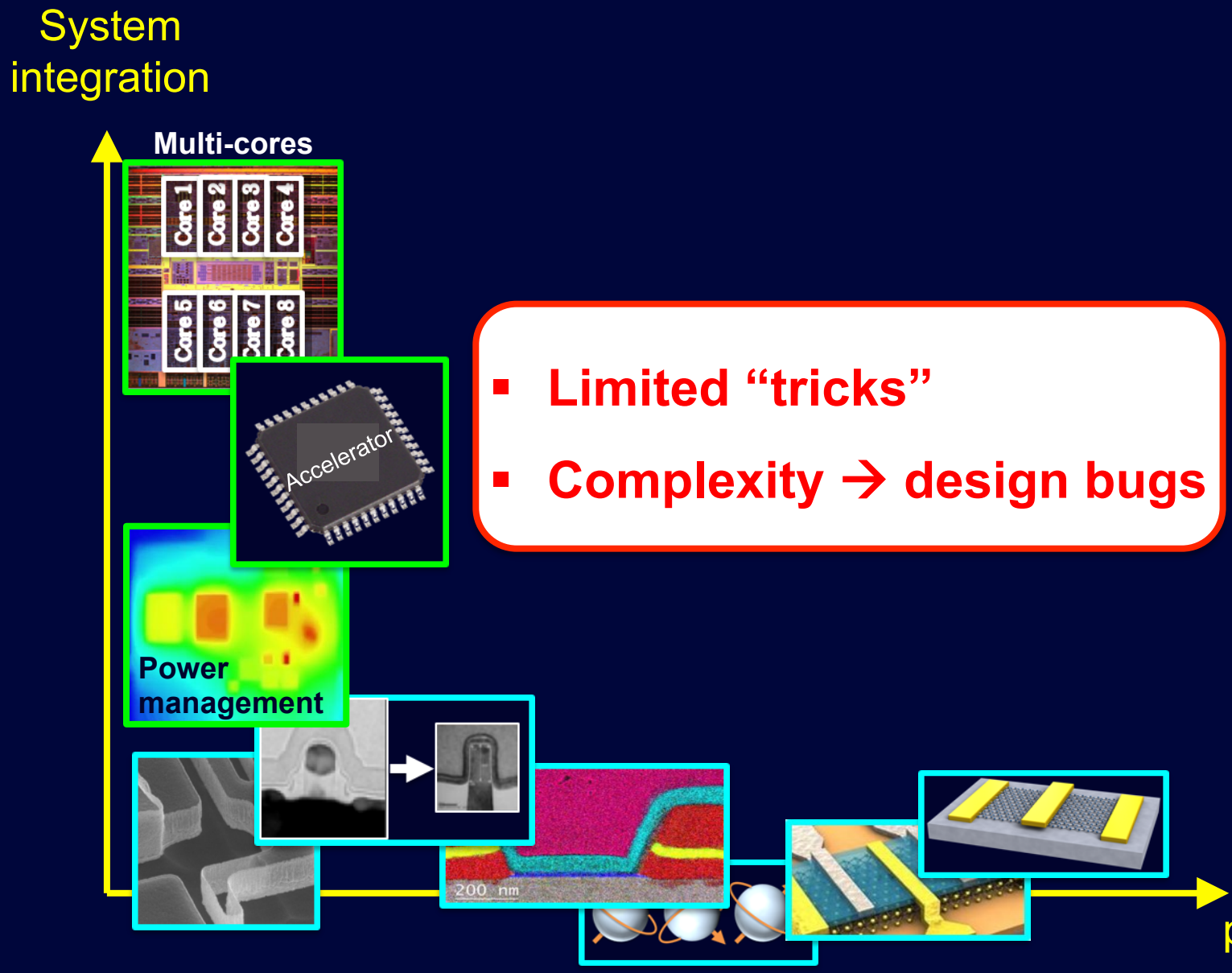
System integration

- **Few experimental demos**
- **Transistors ≠ system**



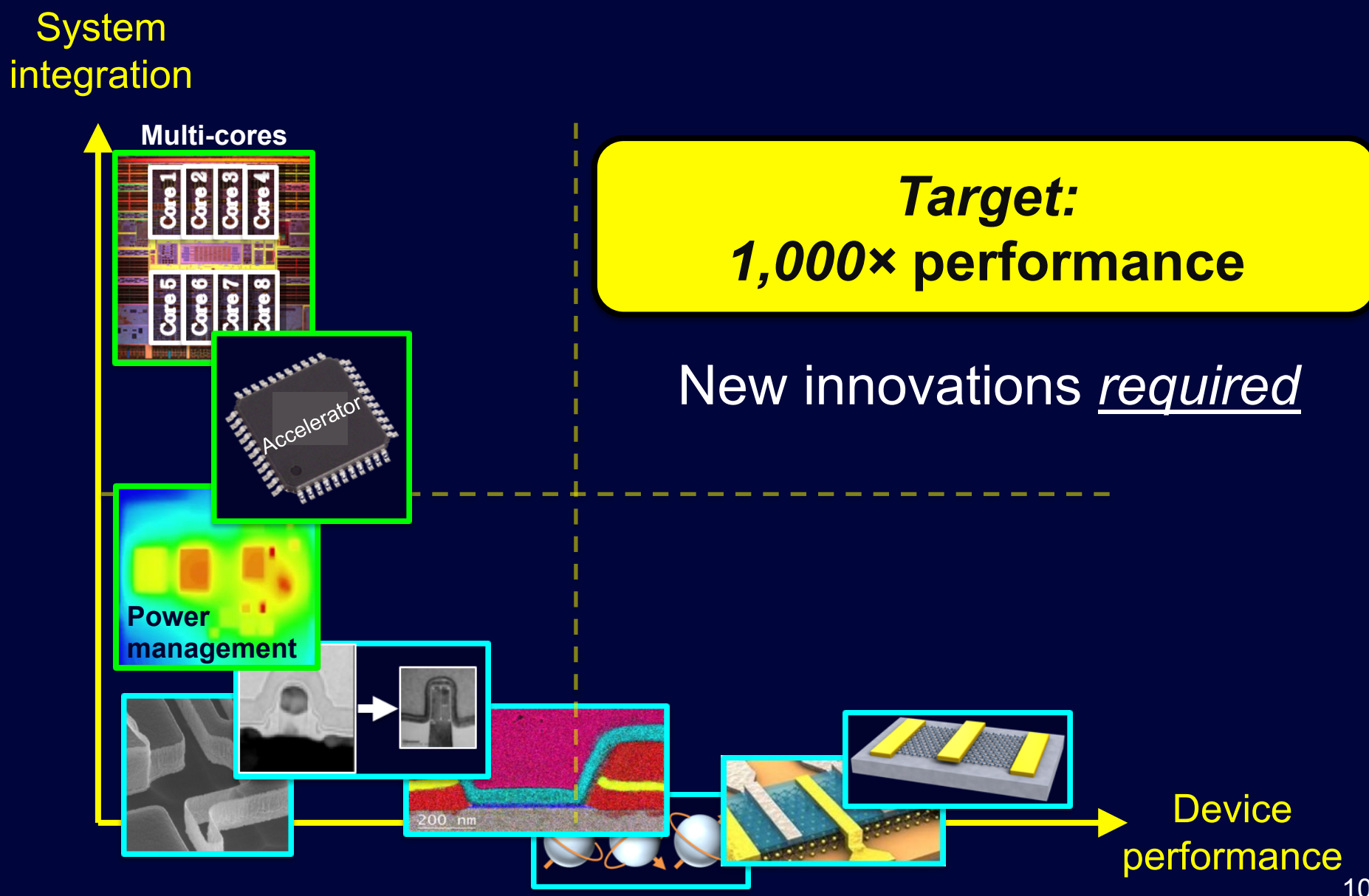
Device performance

Option 2: Design Tricks



- **Limited "tricks"**
- **Complexity → design bugs**

Improve Computing Performance



Target:
1,000× performance

New innovations *required*

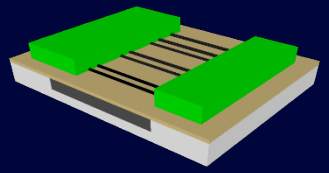
Solution: Nano*Systems*

Transform new nanotech

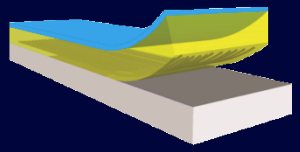
into new systems

enable new applications

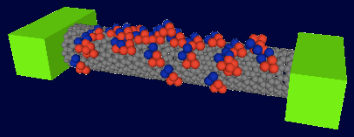
New devices



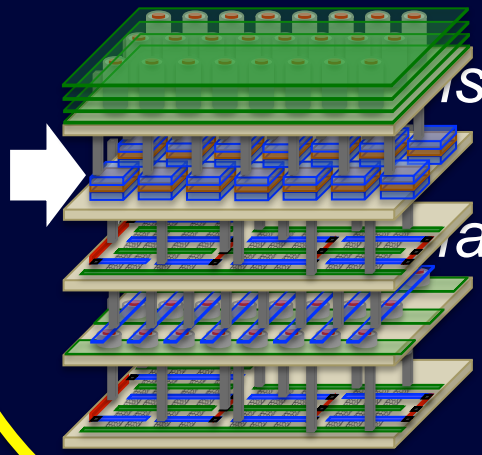
New fabrication



New sensors



New Architectures

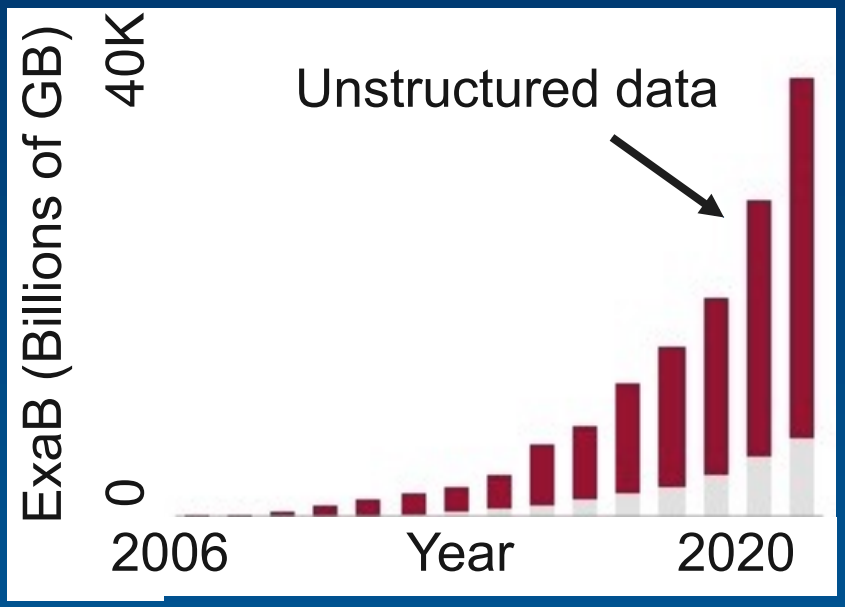


ns?

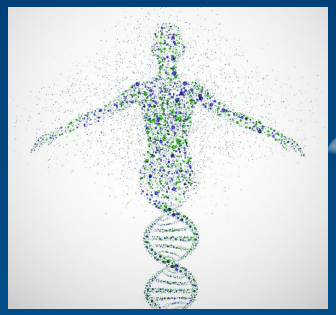
fabricatio

Abundant-Data Explosion

"Swimming in sensors, drowning in data"



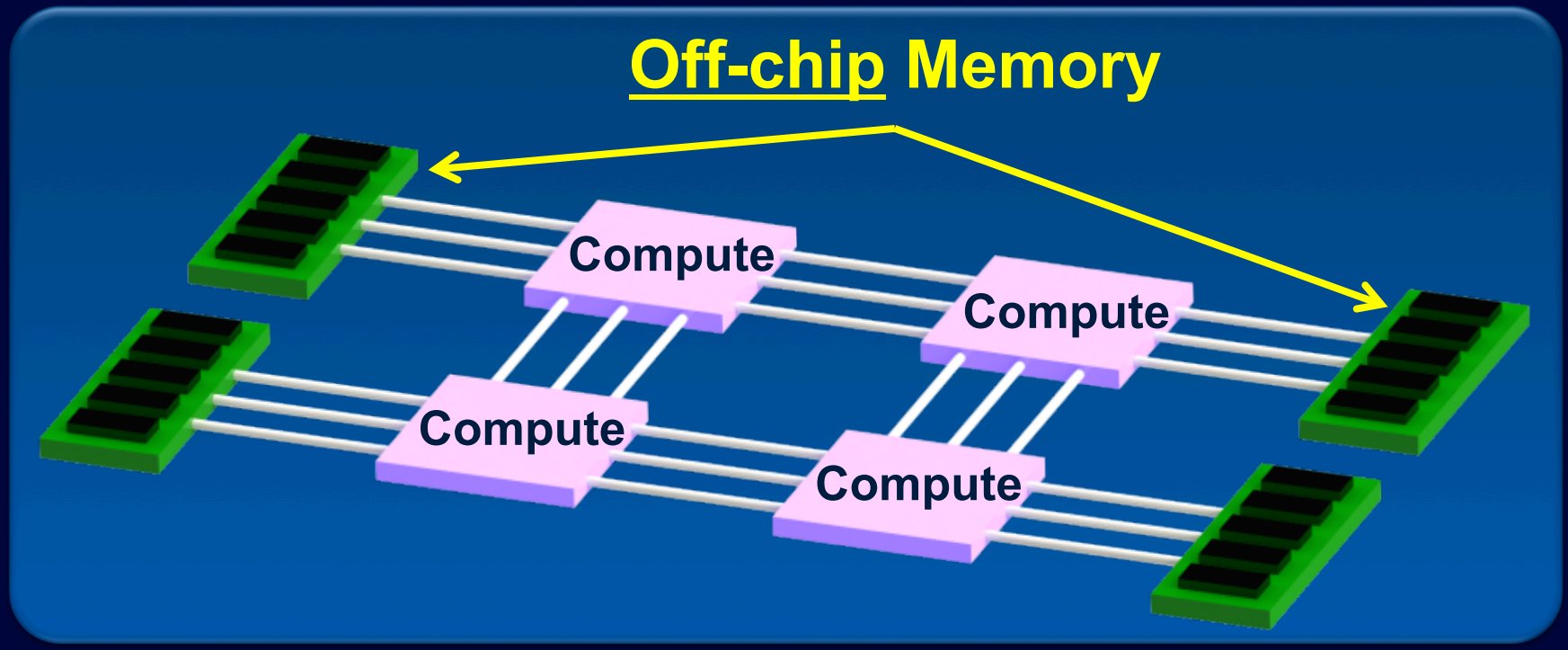
Wide variety & complexity



- Mine, search, analyze: near real-time
  - Data centers, mobile phones, robots

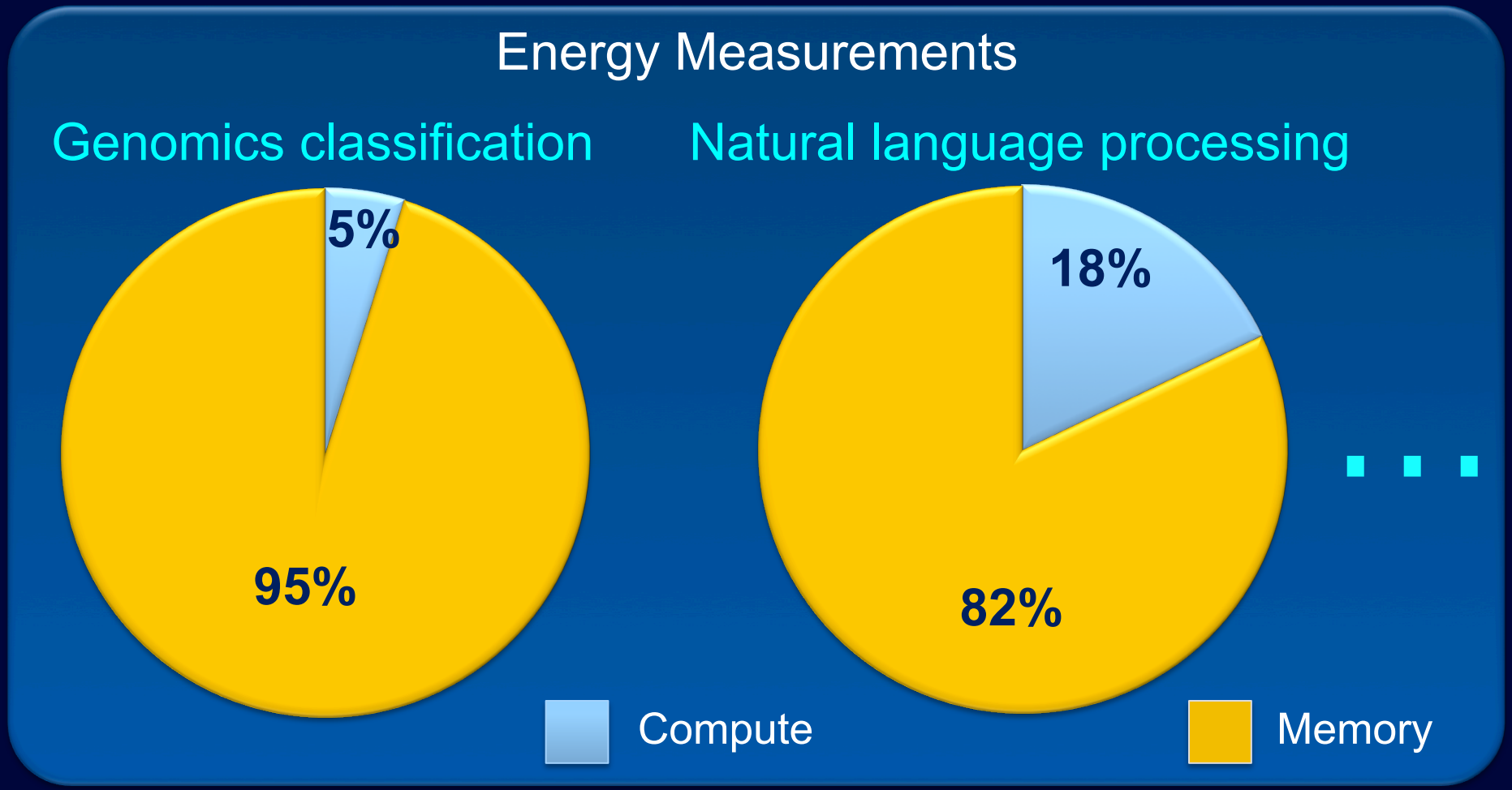
Today's System Bottlenecks

- **Separate** compute & memory chips
- Not enough on-chip memory
- Capacity **&** bandwidth critical



Abundant-Data Applications

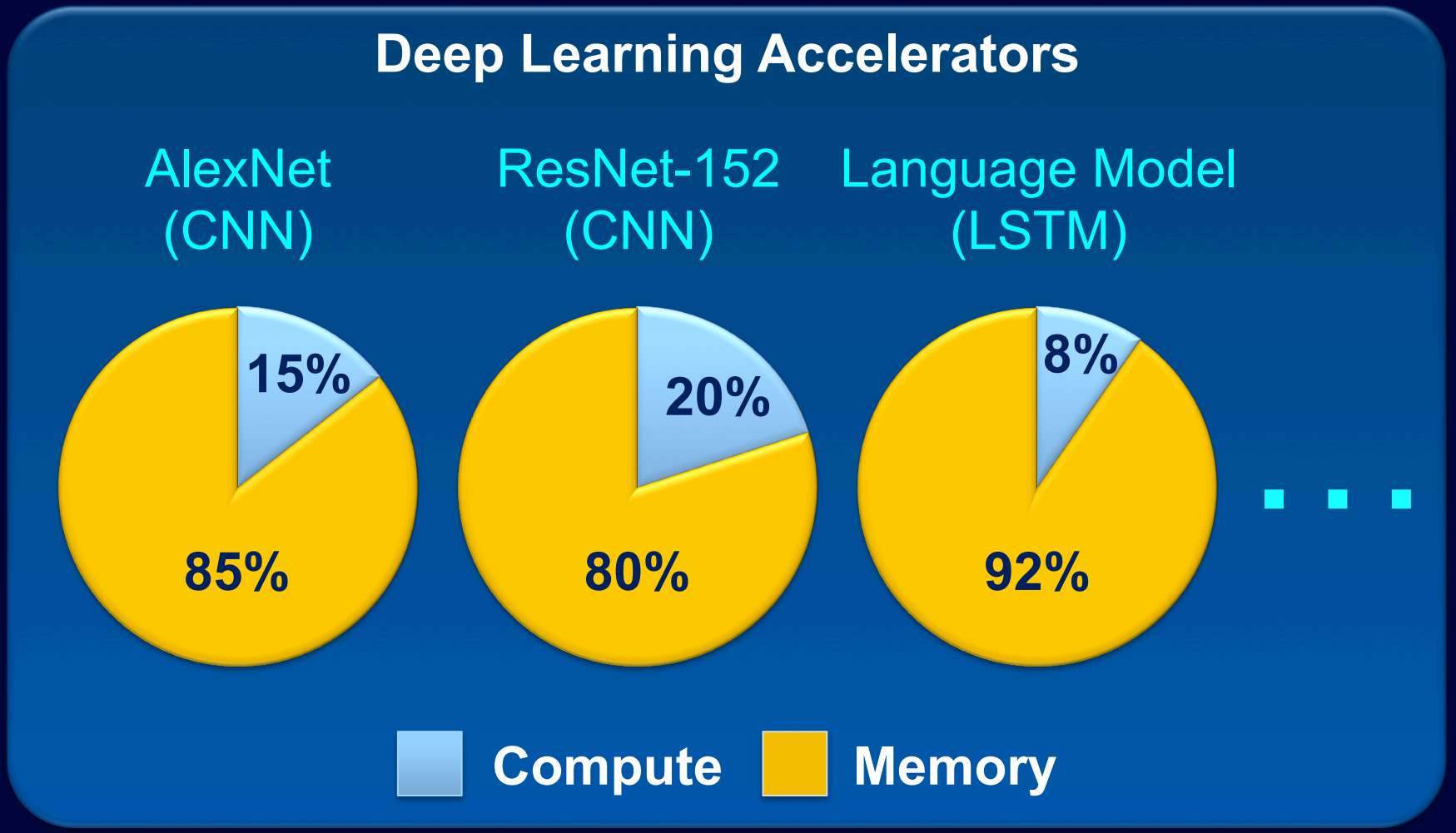
Huge memory wall: processors, accelerators



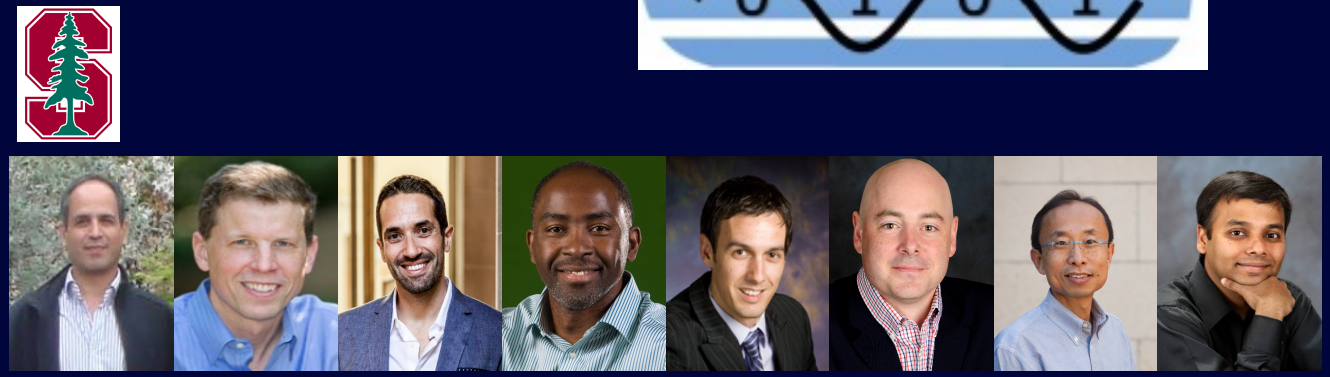
Intel performance counter monitors 2 CPUs, 8-cores/CPU + 128GB DRAM

Abundant-Data Applications

Huge memory wall: processors, accelerators



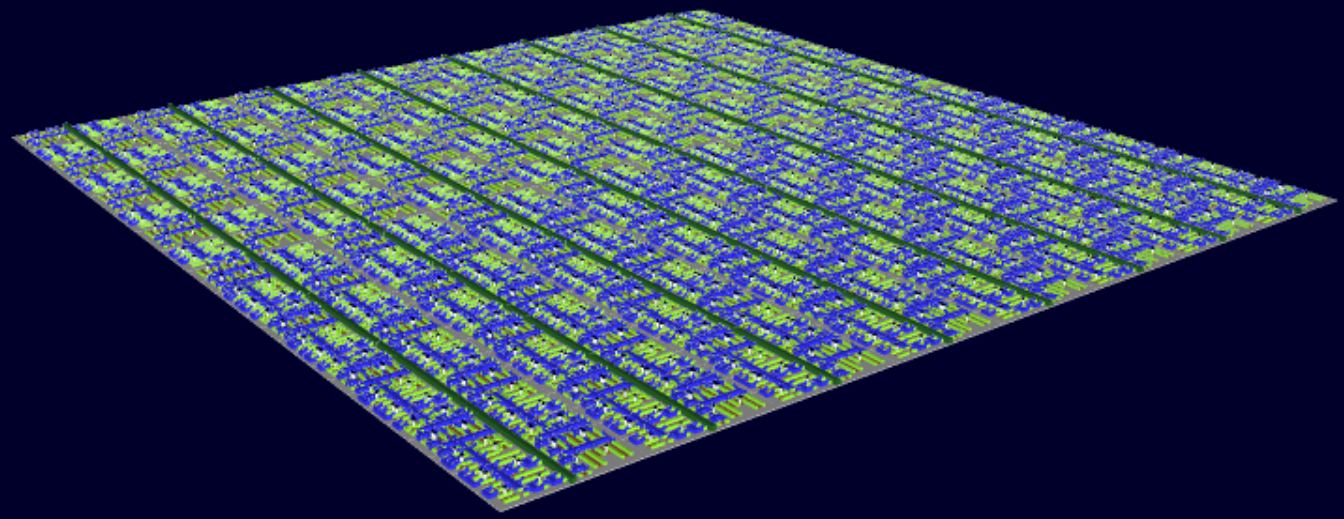
Nano-Engineered Computing Systems Technology



[Aly IEEE Computer 15]

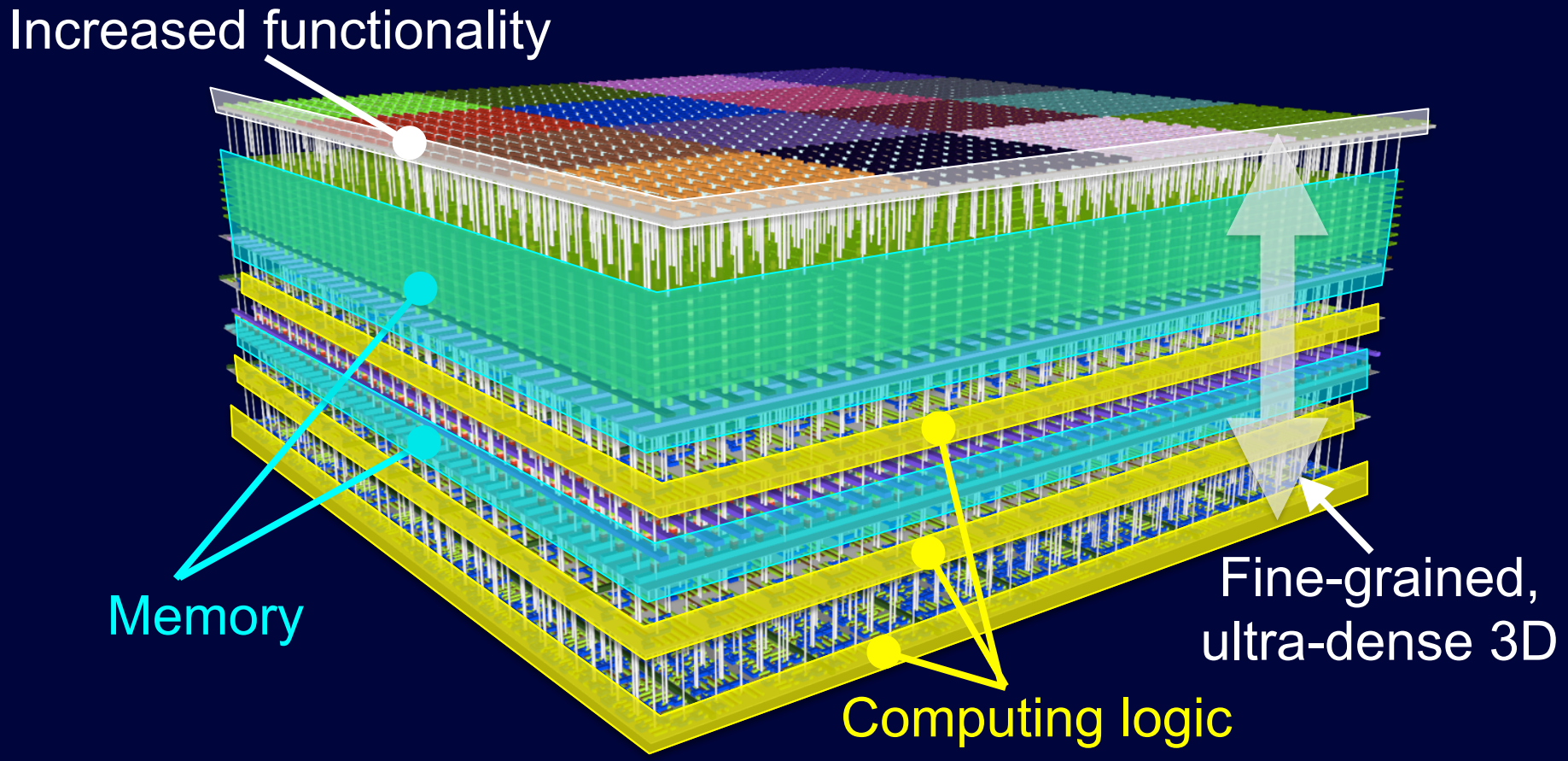
N3XT NanoSystems

Computation immersed in memory



N3XT NanoSystems

Computation immersed in memory



Impossible with today's technologies

N3XT Computation Immersed in Memory

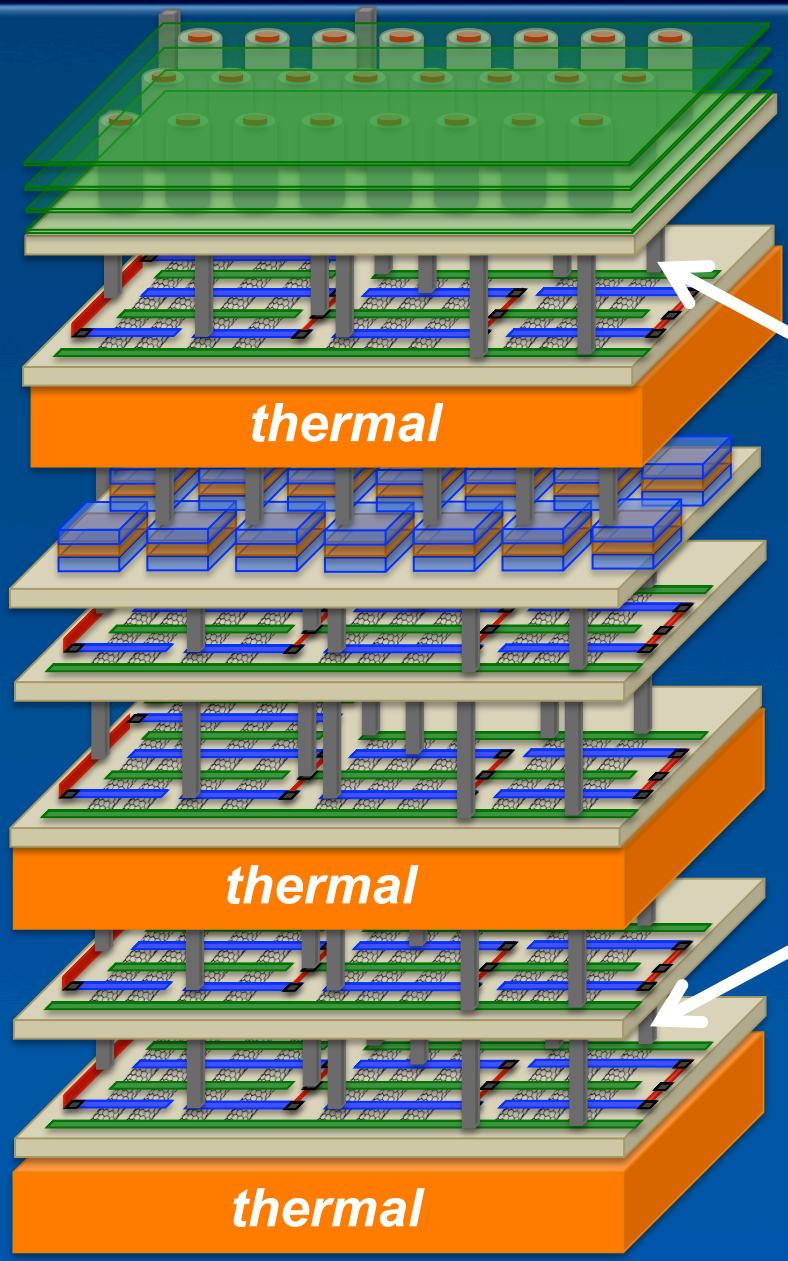
3D Resistive RAM
Massive storage

1D CNFET, 2D FET
Compute, RAM access

MRAM
Quick access

1D CNFET, 2D FET
Compute, RAM access

1D CNFET, 2D FET
Compute, Power, Clock

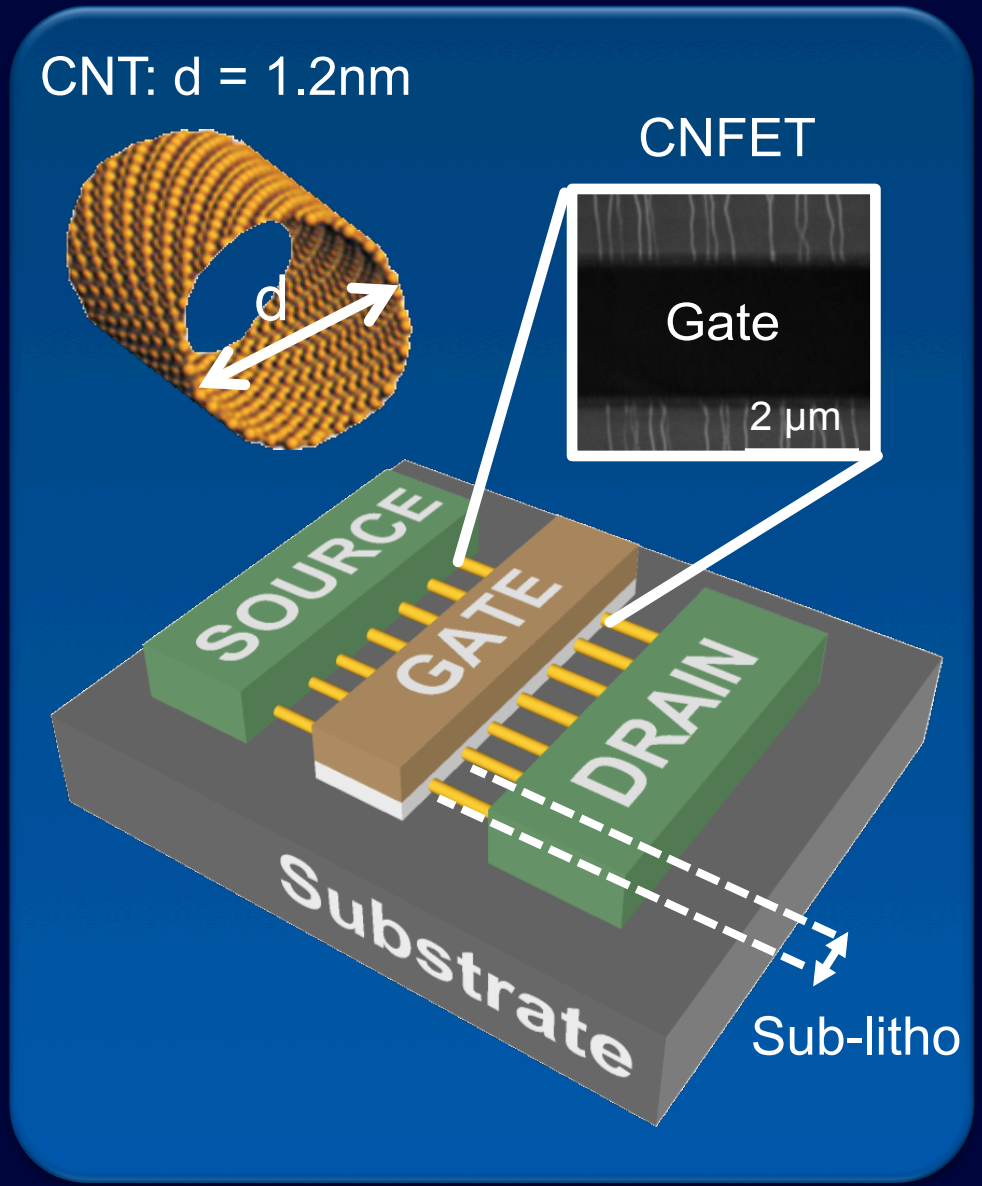


No TSV

Ultra-dense, fine-grained vias

Silicon compatible

Carbon Nanotube FET (CNFET)



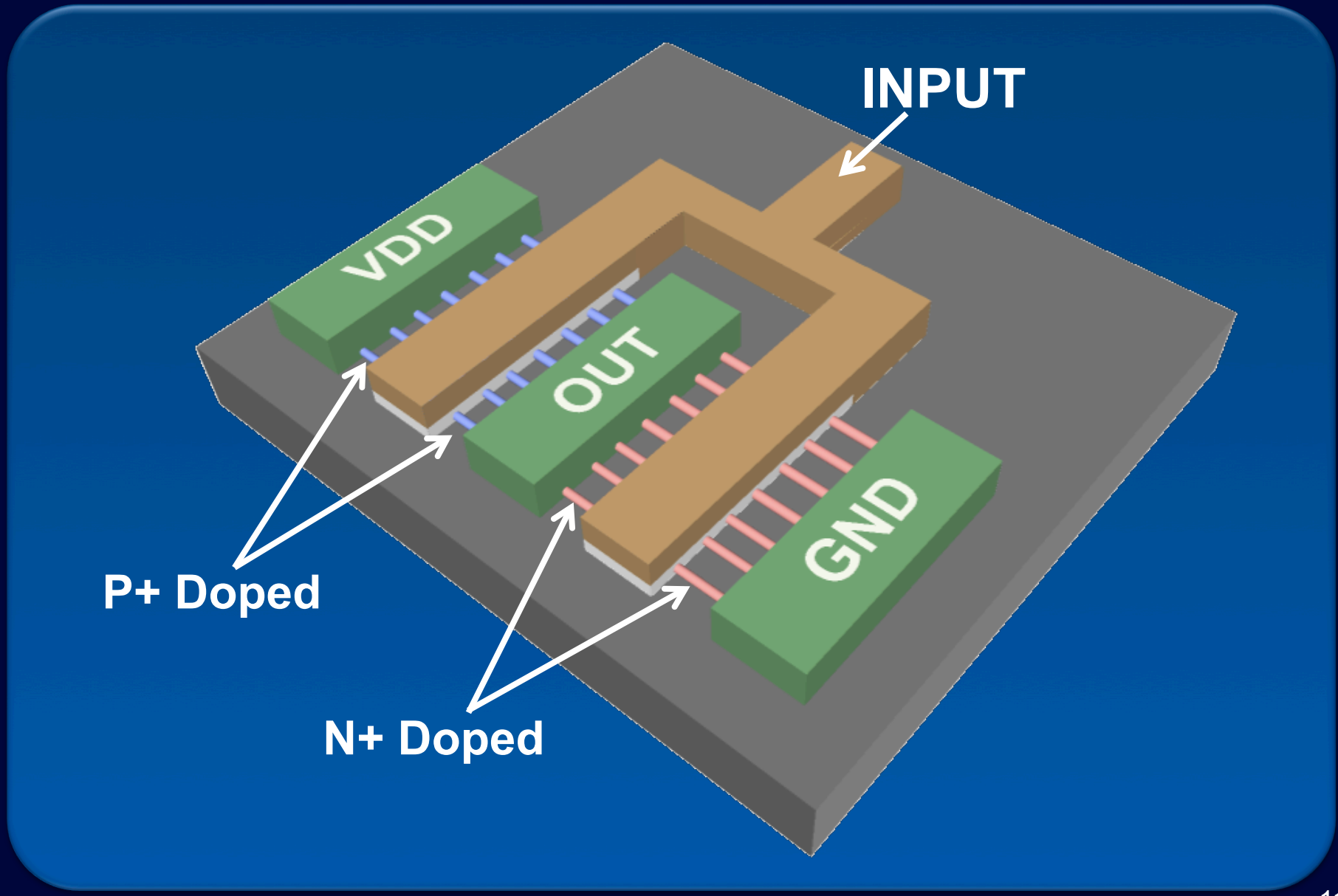
Energy **D**elay **P**roduct

- ~ 10× benefit

Full-chip case studies

[IBM, IMEC, Stanford, other commercial]

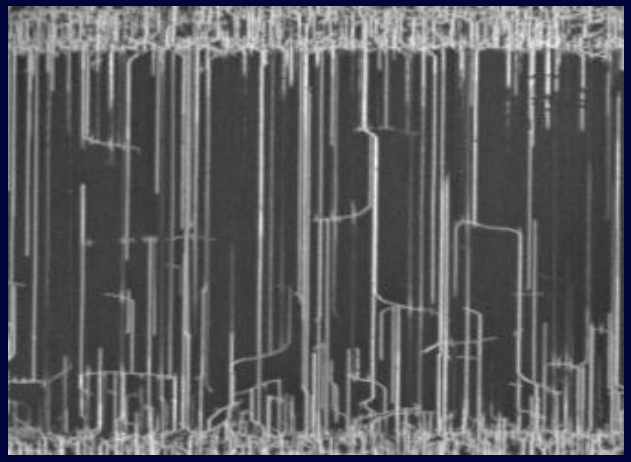
CNFET Inverter



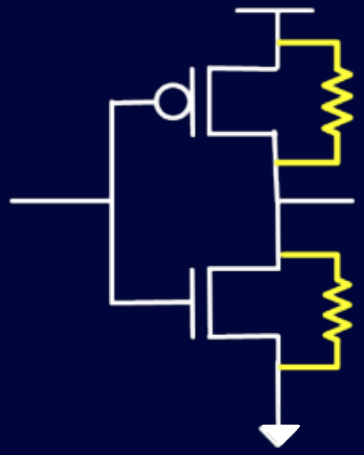
Big Promise, Major Obstacles

- Process advances alone inadequate

Mis-positioned CNTs



Metallic CNTs

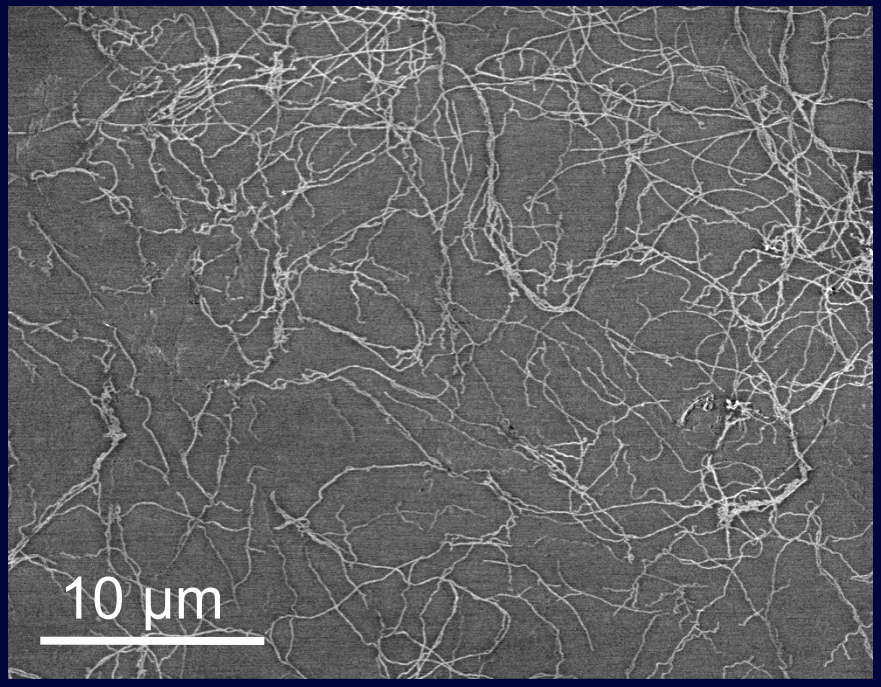


Imperfection-immune paradigm

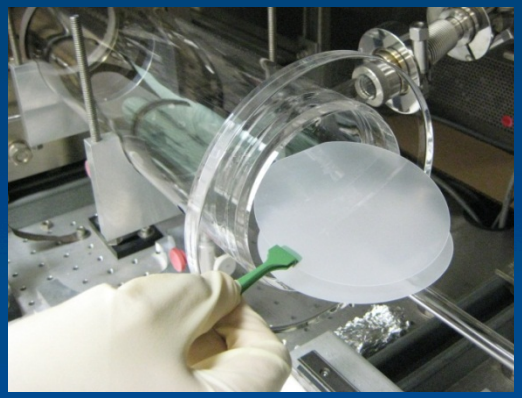
[Zhang IEEE TCAD 12]

CNT Growth *circa* 2005

- Highly mis-positioned

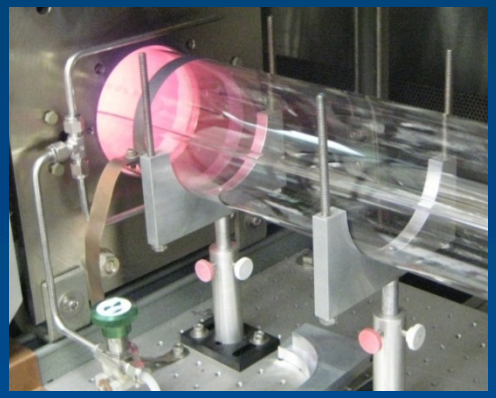


First Wafer-Scale Aligned CNT Growth

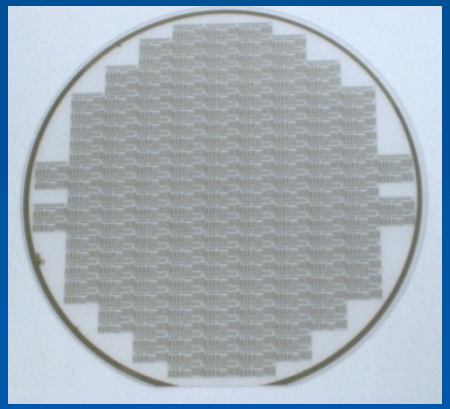


Quartz wafer
with catalyst

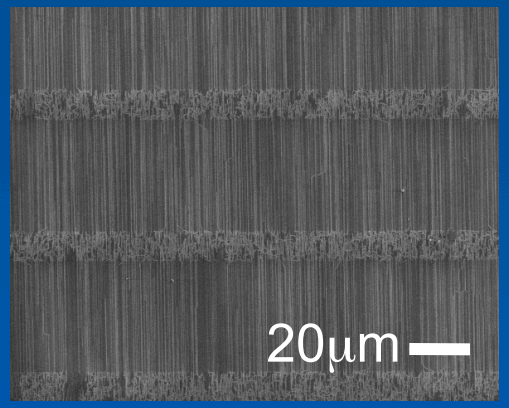
Aligned
CNT growth



Quartz wafer with CNTs



99.5% aligned CNTs



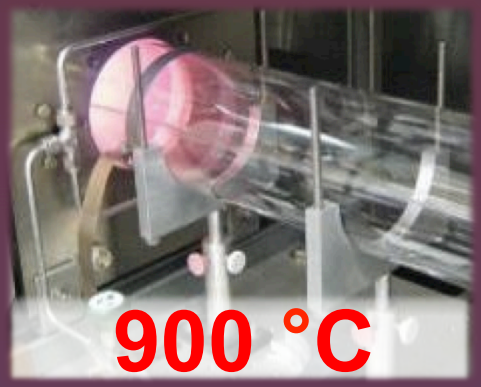
Stanford Nanofabrication Facility

[Patil VLSI Tech. 08, IEEE TNANO 09]

Wafer-Scale CNT Transfer

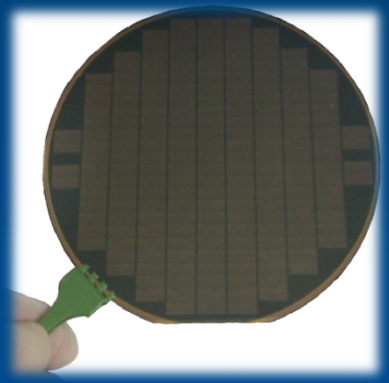
High-temperature CNT growth

Low-temperature circuit fabrication



CNT transfer

120 °C

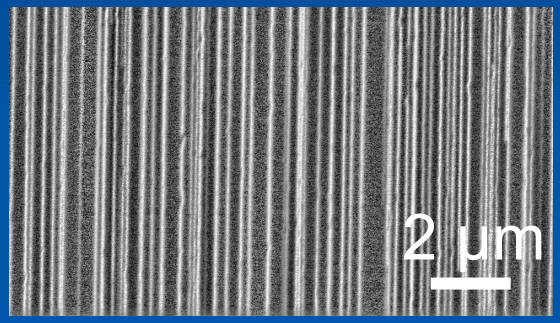


Before transfer

Quartz

After transfer

SiO_2/Si



[Patil VLSI Tech. 08, IEEE TNANO 09]

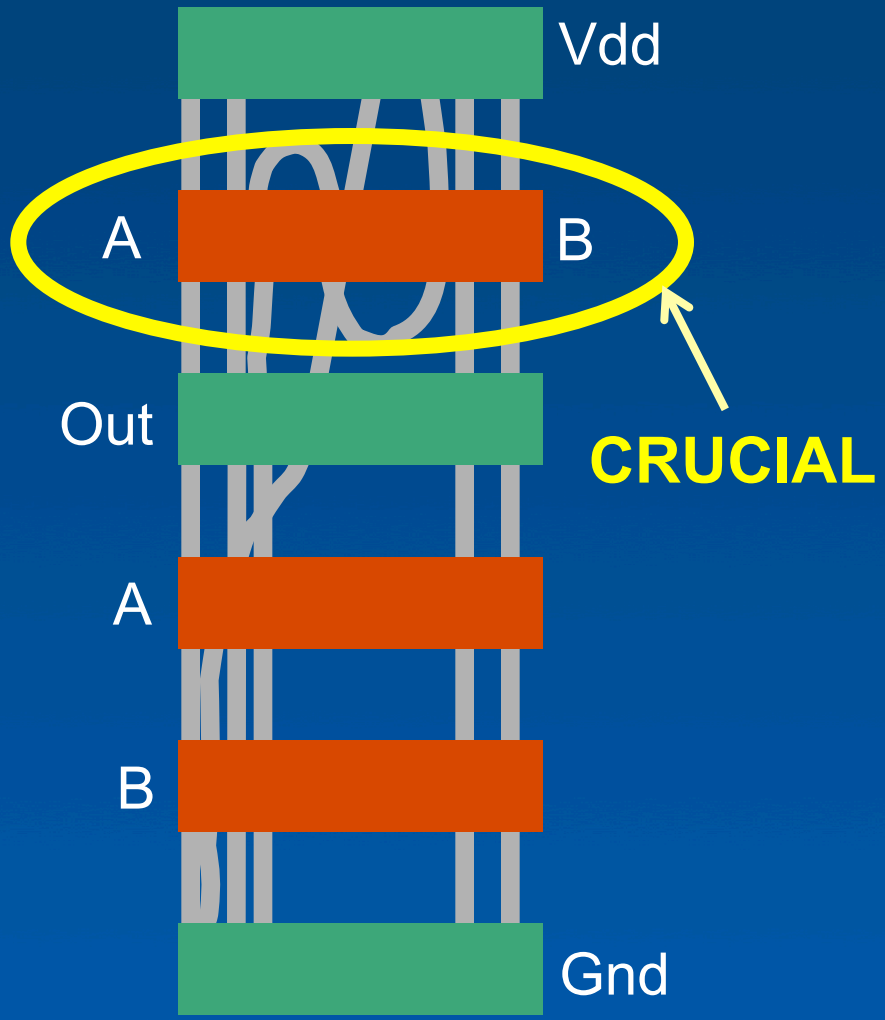
Mis-Positioned CNT-Immune NAND

1. Grow CNTs

[Patil IEEE TCAD 09]

Mis-Positioned CNT-Immune NAND

1. Grow CNTs
2. Extended gate, contacts

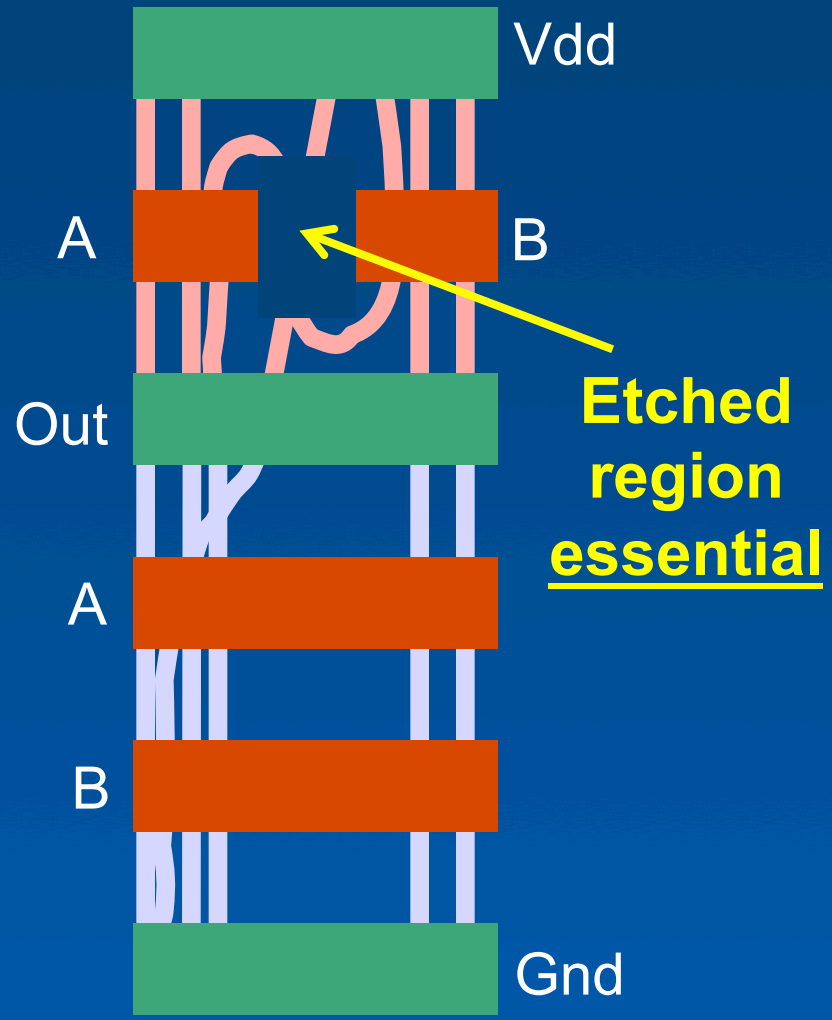


[Patil IEEE TCAD 09]

Mis-Positioned CNT-Immune NAND

1. Grow CNTs
2. Extended gate, contacts
3. Etch gate & CNTs
4. Dope P & N regions

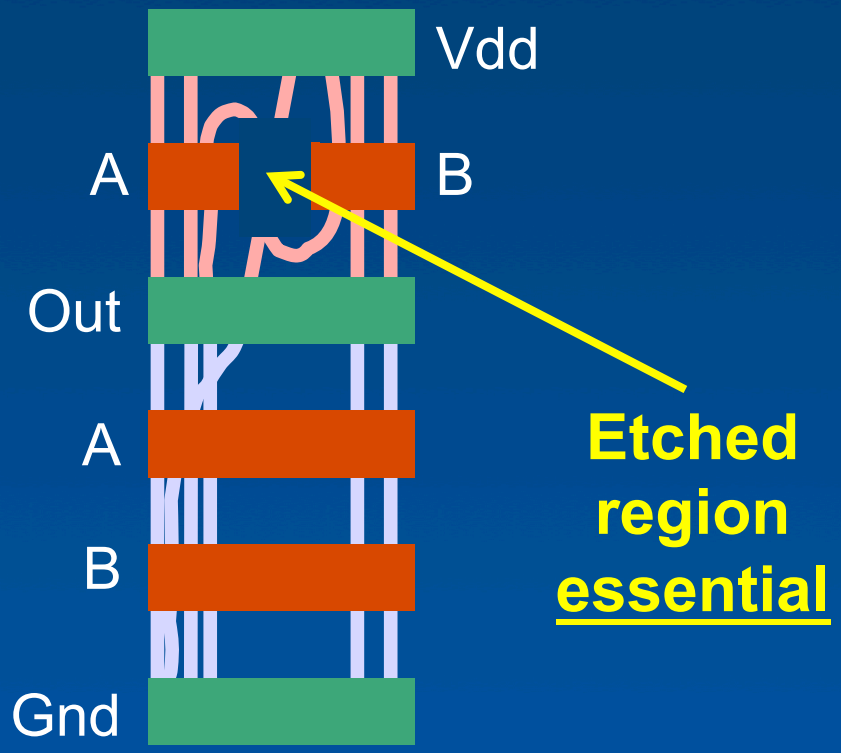
- Arbitrary logic functions
  - Graph algorithms



[Patil IEEE TCAD 09]

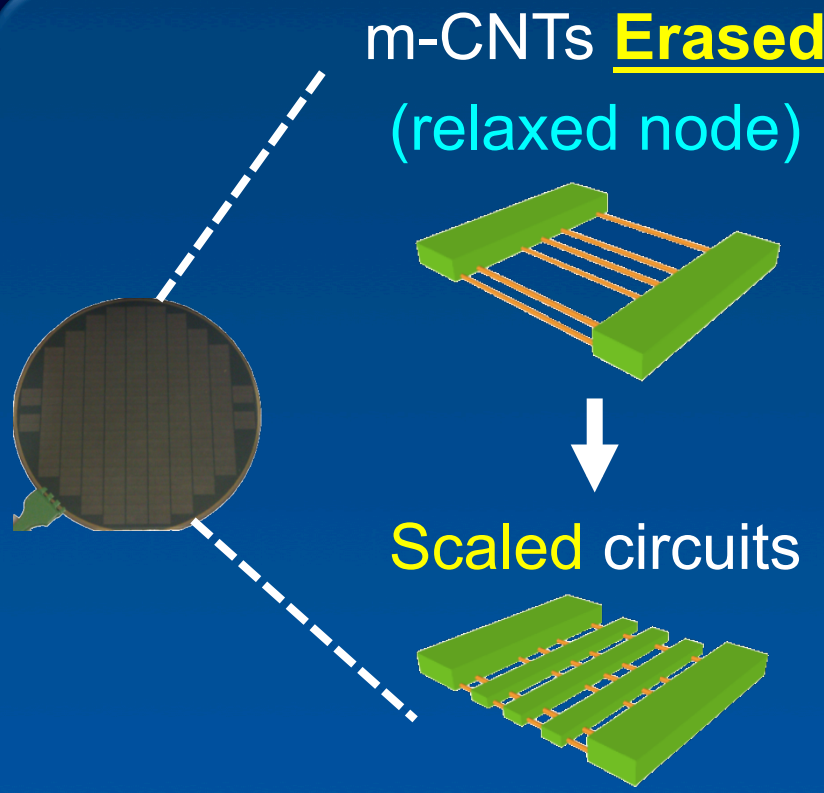
Imperfection-Immune VLSI

Mis-positioned CNTs



- Arbitrary logic functions

Metallic CNTs



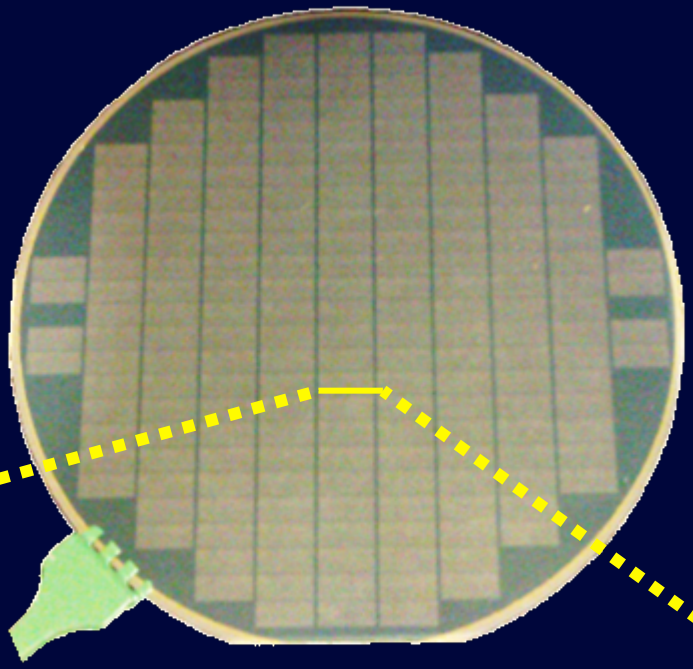
- Scalable m-CNT Removal

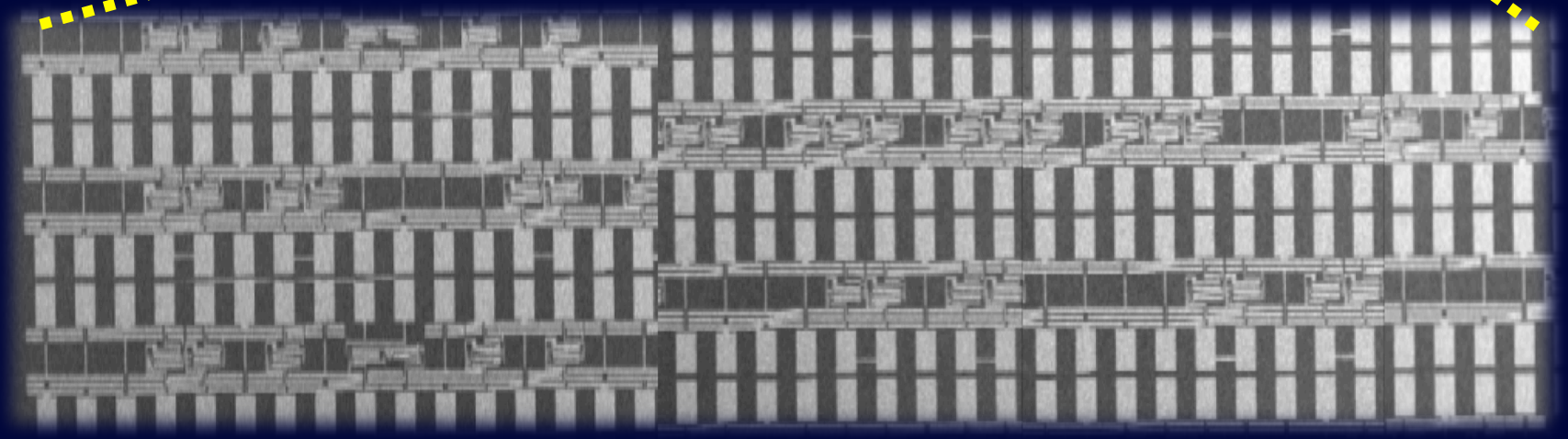
[Patil Symp. VLSI Tech. 08, TCAD 09, Shulaker IEDM 15]

Most Importantly

- VLSI processing
  - No per-unit customization
- VLSI design
  - Immune CNT library

CNT Computer

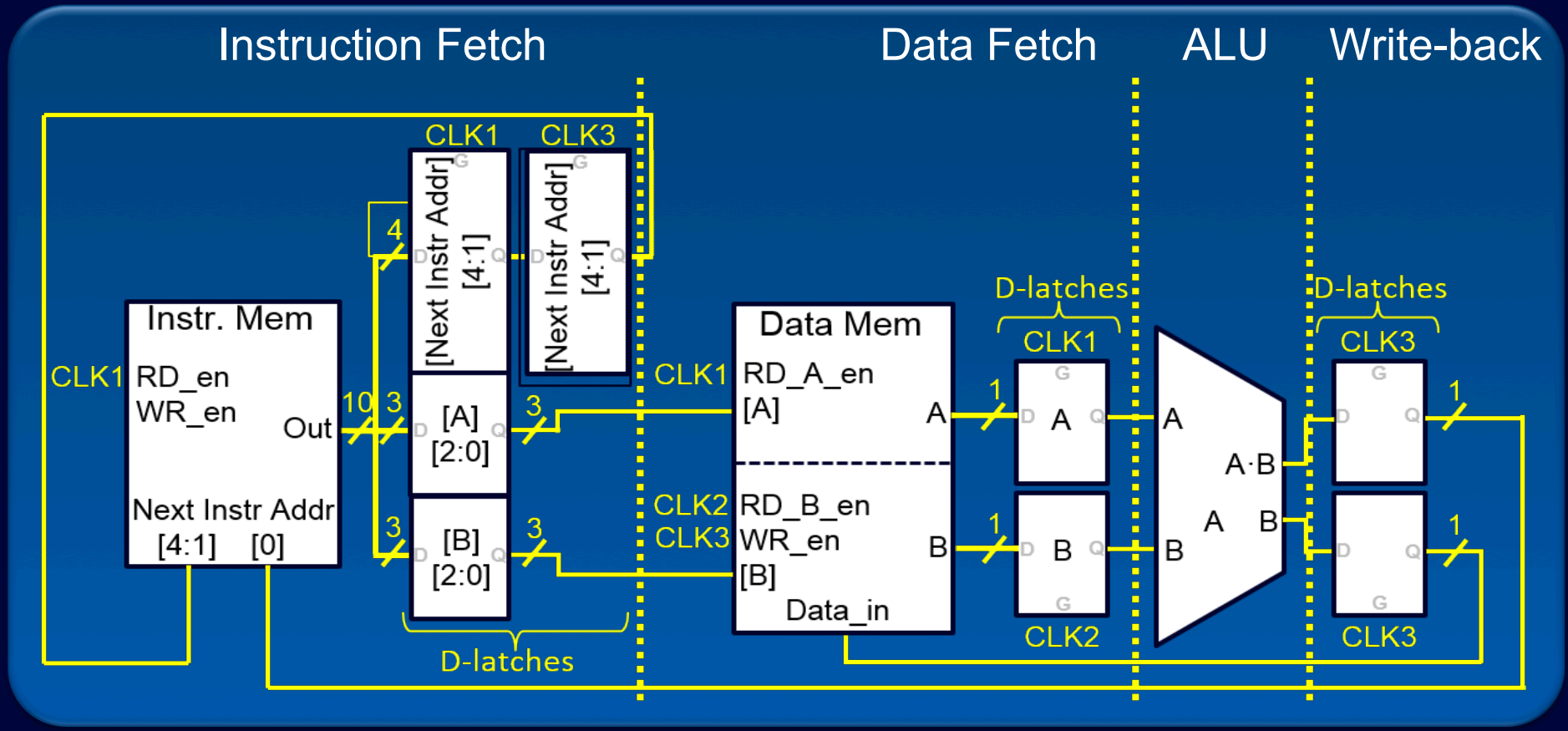




[Shulaker Nature 13]

CNT Computer

- Turing-complete processor: entirely CNFETs



[Shulaker Nature 13]

10× EDP, BUT…

How can we do better ?

N3XT Computation Immersed in Memory

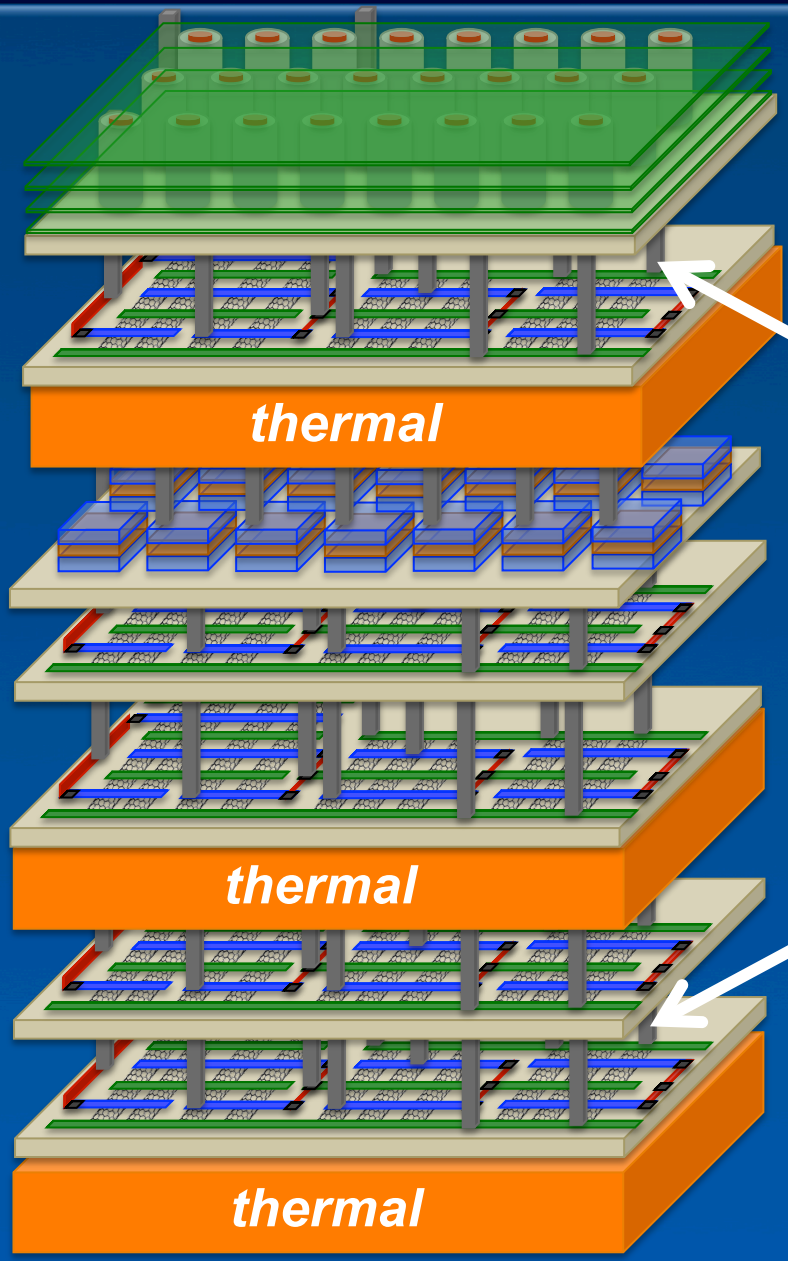
3D Resistive RAM
Massive storage

1D CNFET, 2D FET
Compute, RAM access

MRAM
Quick access

1D CNFET, 2D FET
Compute, RAM access

1D CNFET, 2D FET
Compute, Power, Clock



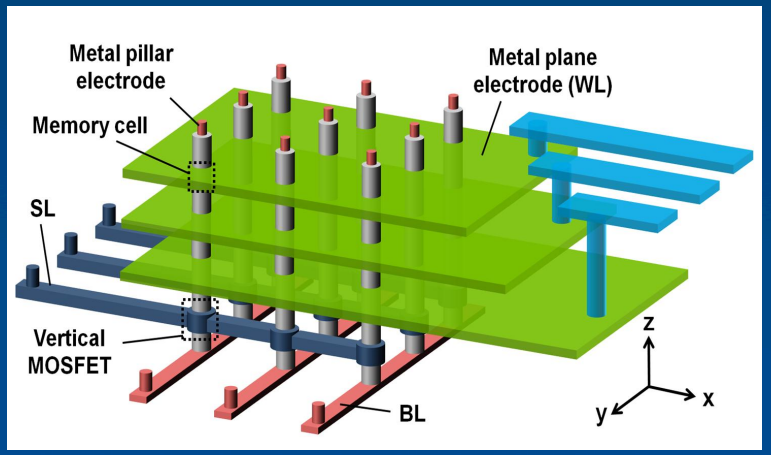
No TSV

Ultra-dense, fine-grained vias

Silicon compatible

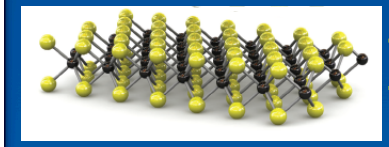
Many Nano-scale Innovations

Memory & logic devices

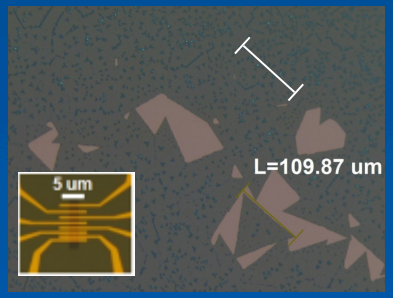


3D Resistive RAM (RRAM)

MoS_2

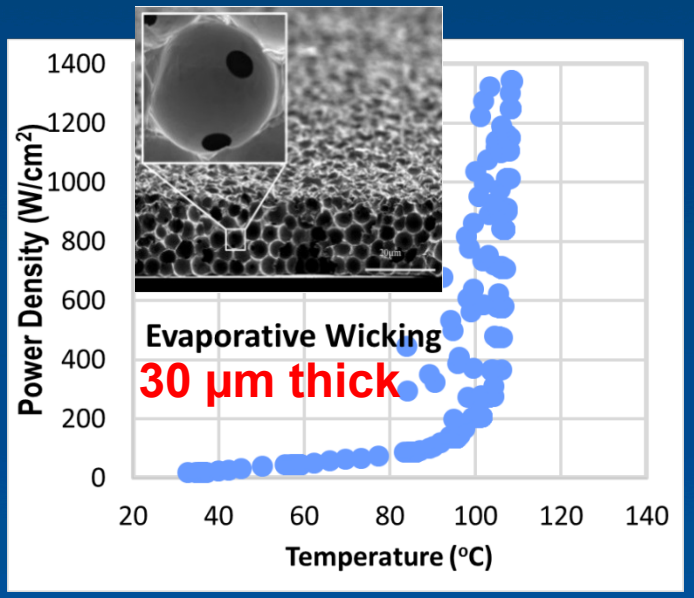


≤1 nm



2D FETs: large-area monolayer MoS_2

Embedded cooling



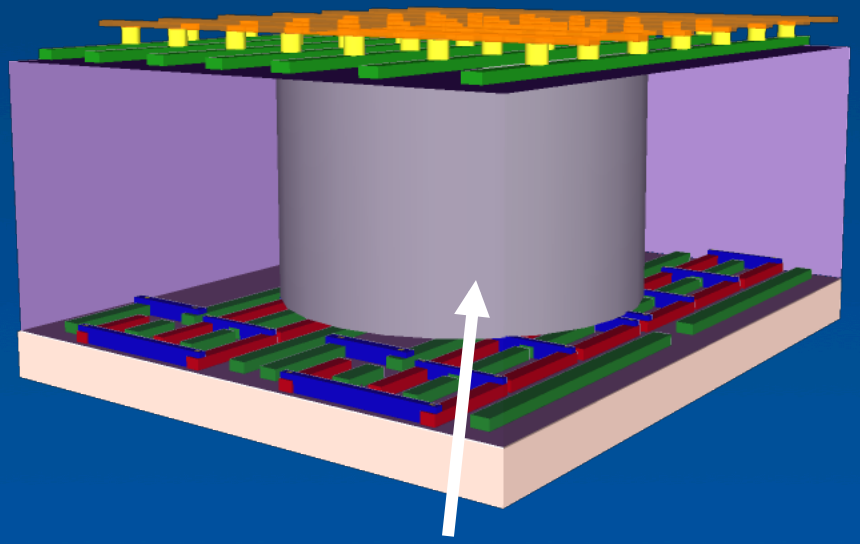
Phase change: hotspots suppressed

Vertical metal nanowire arrays

3D Integration

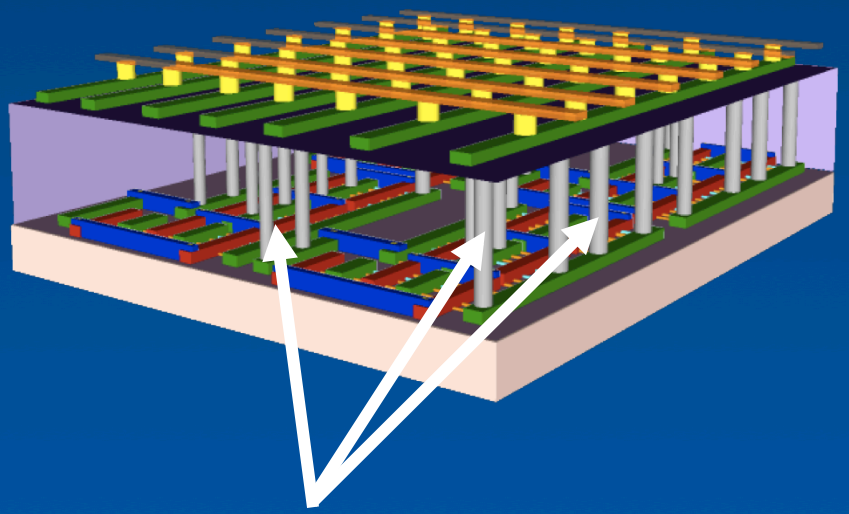
- Massive ILV density >> TSV density

TSV (chip stacking)



Through silicon via (TSV)

Dense, e.g., monolithic



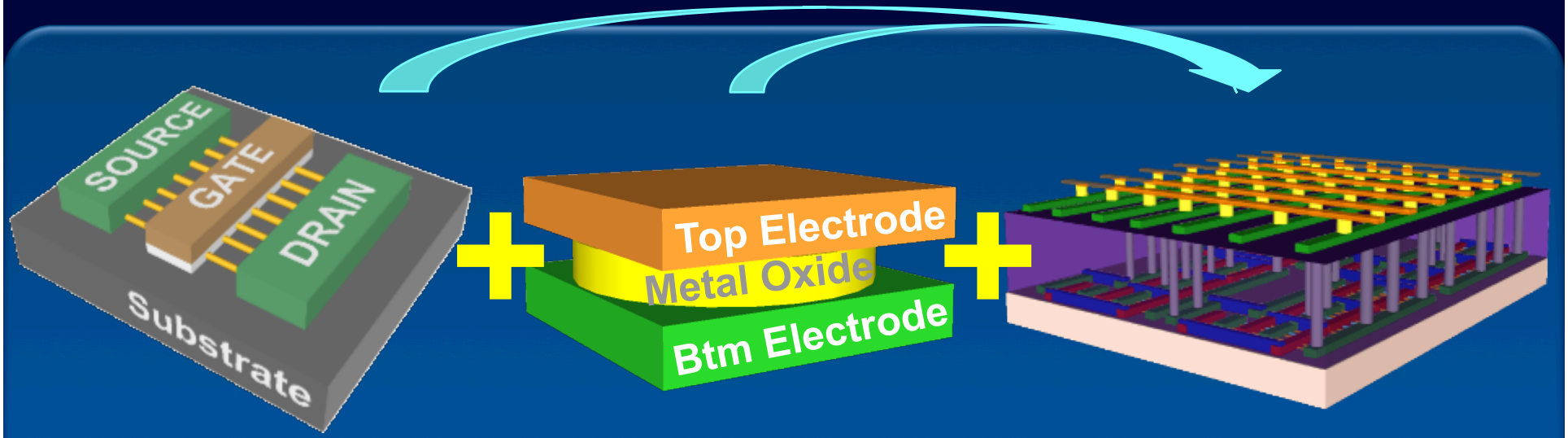
Nano-scale inter-layer vias (ILVs)

Realizing Monolithic 3D

- Low-temperature fabrication: **< 400 °C**

Device + Architecture Benefits

Naturally enabled



Emerging logic

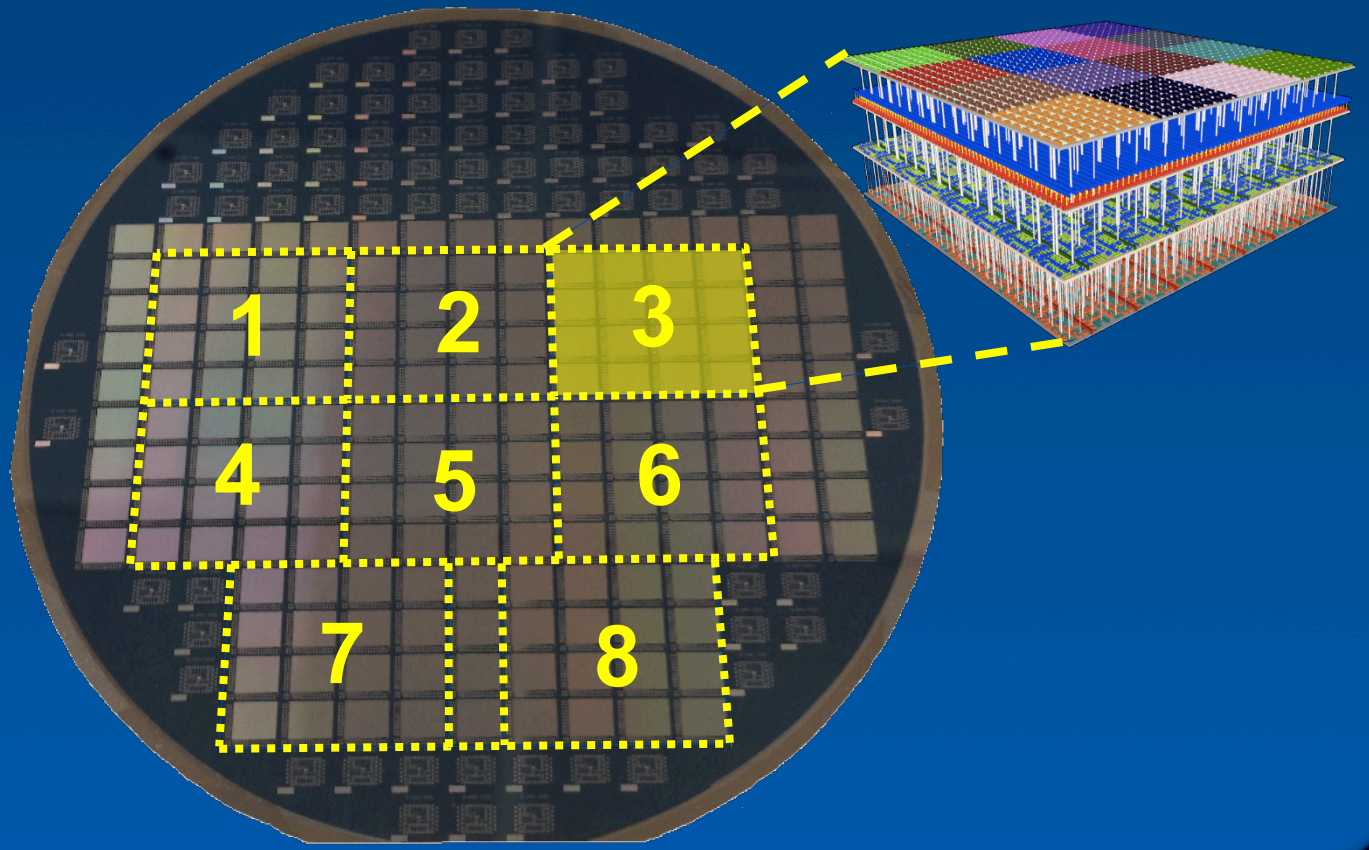
Emerging memory

Monolithic 3D integration

[Wei IEDM 09, 13, Shulaker VLSI Tech 14]

3D NanoSystem

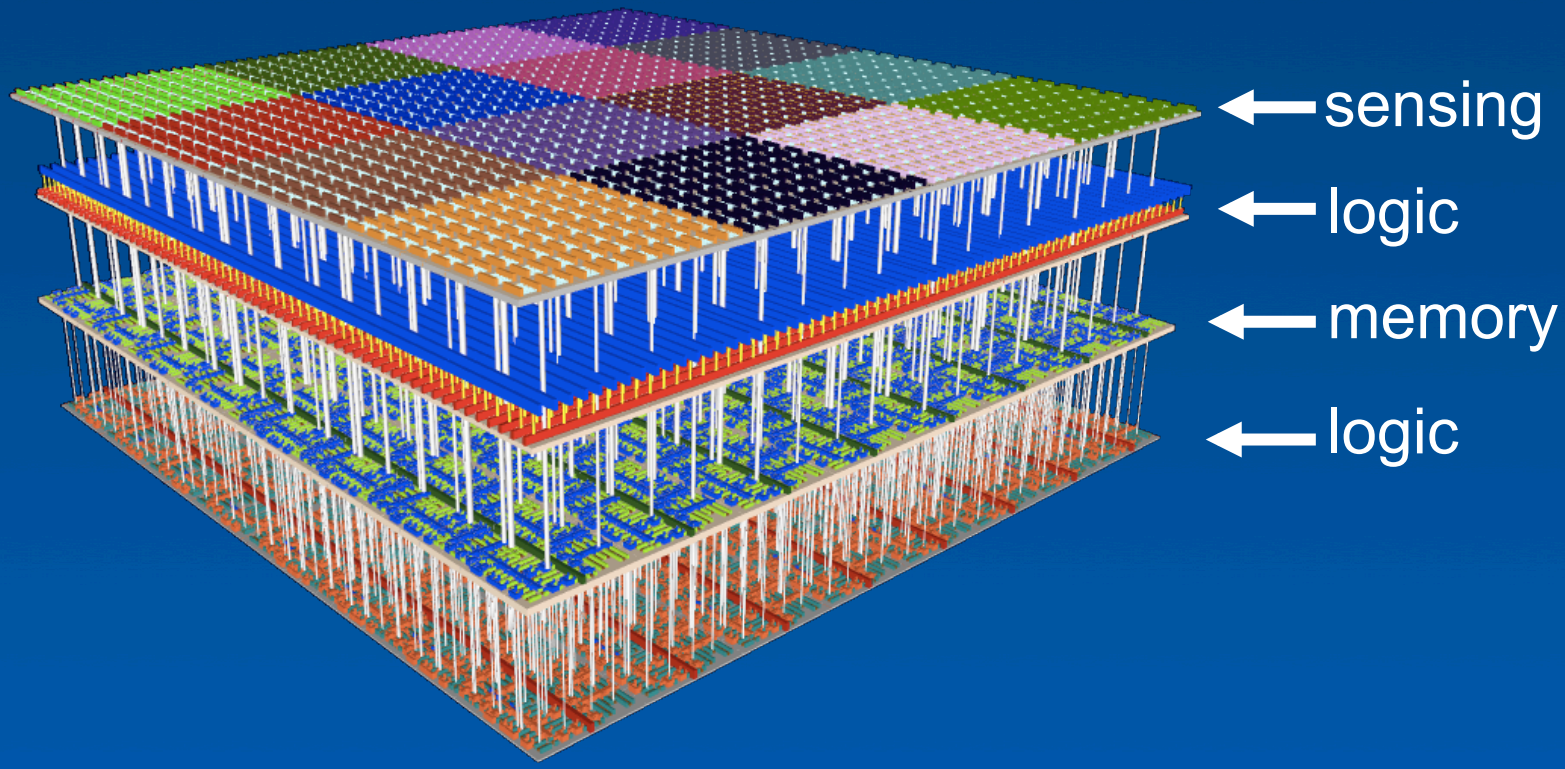
Wafer-scale design + fabrication



[Shulaker Nature 17]

3D NanoSystem

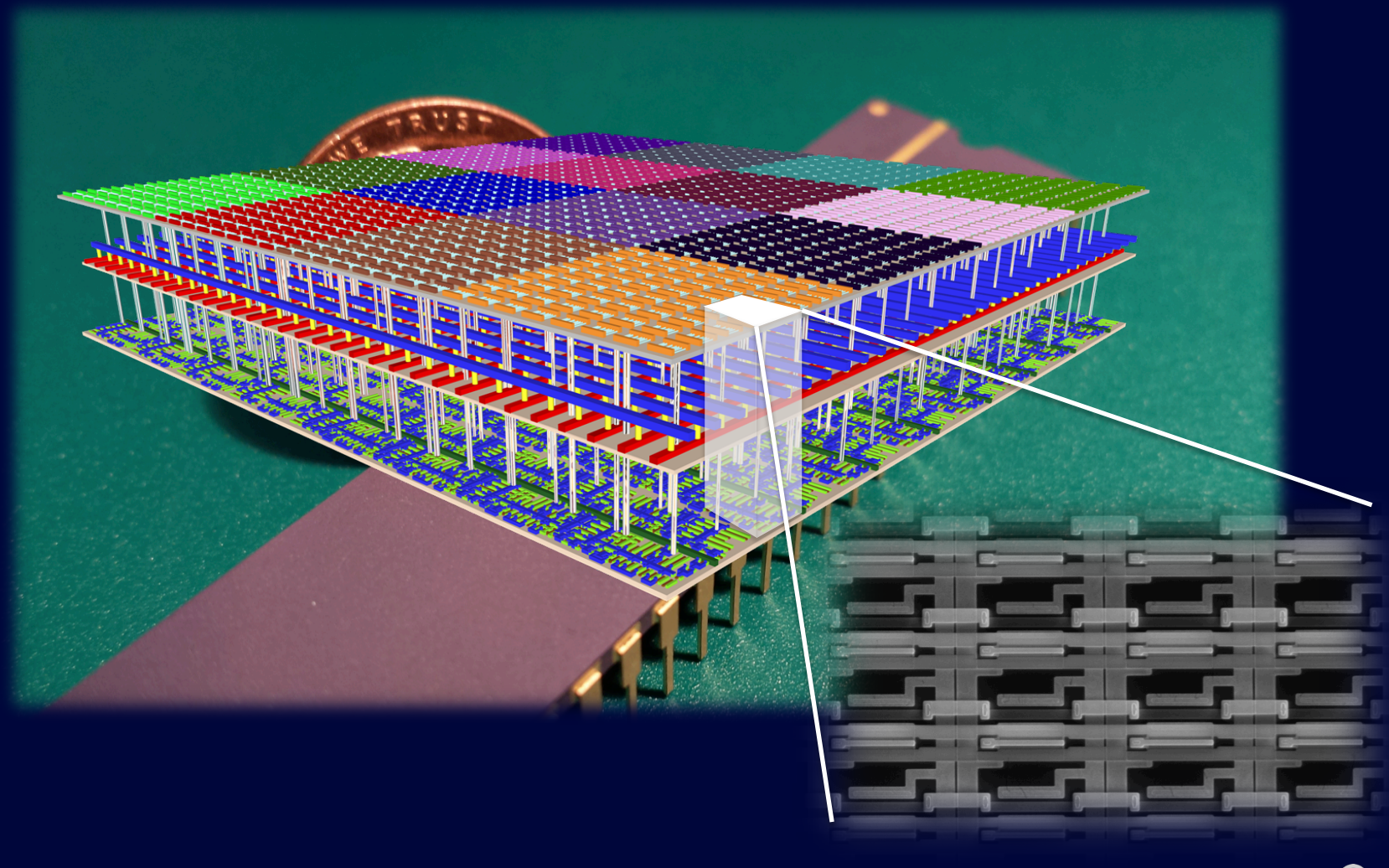
>2 Million CNFETs, 1 Mbit Resistive RAM



[Shulaker Nature 17]

3D NanoSystem

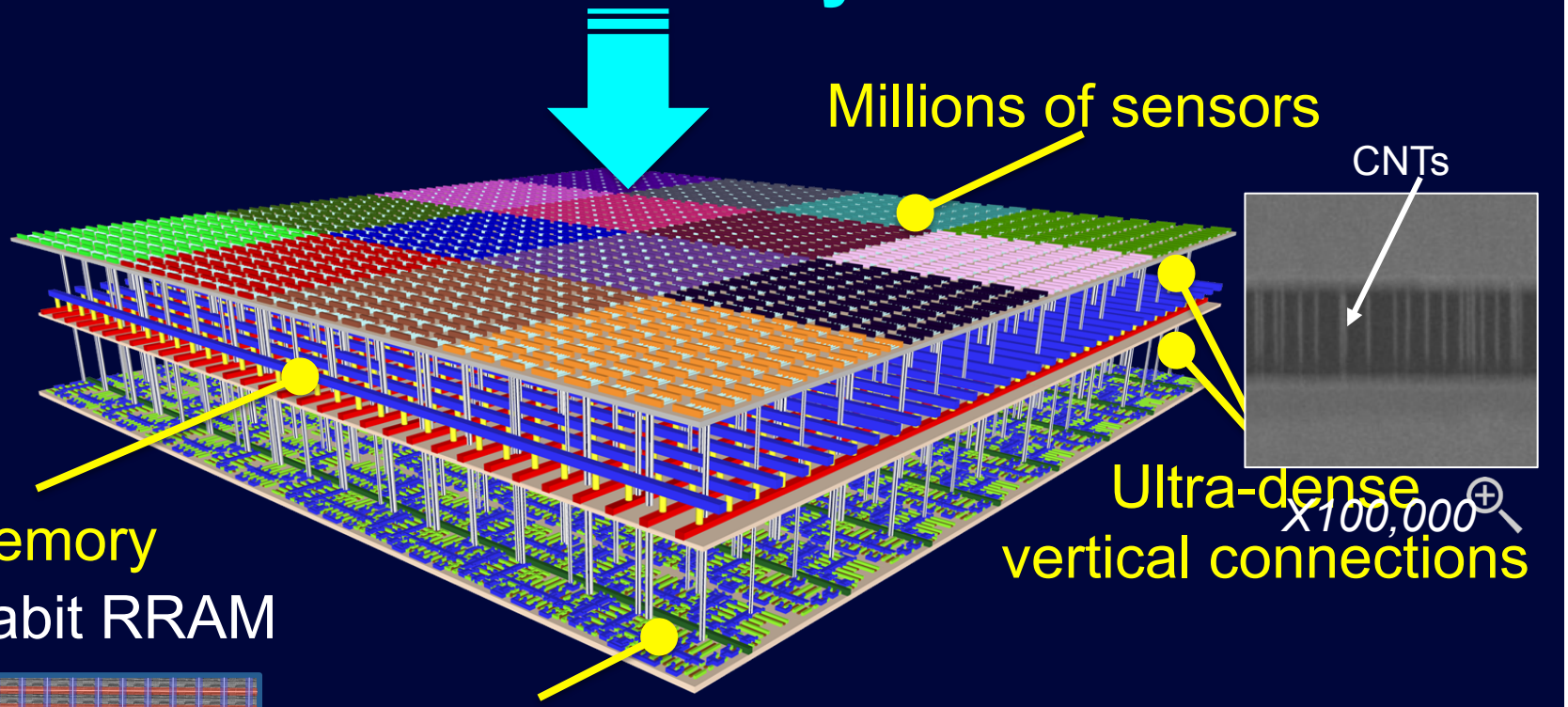
- Interwoven compute + memory + sensing



[Shulaker Nature 17]

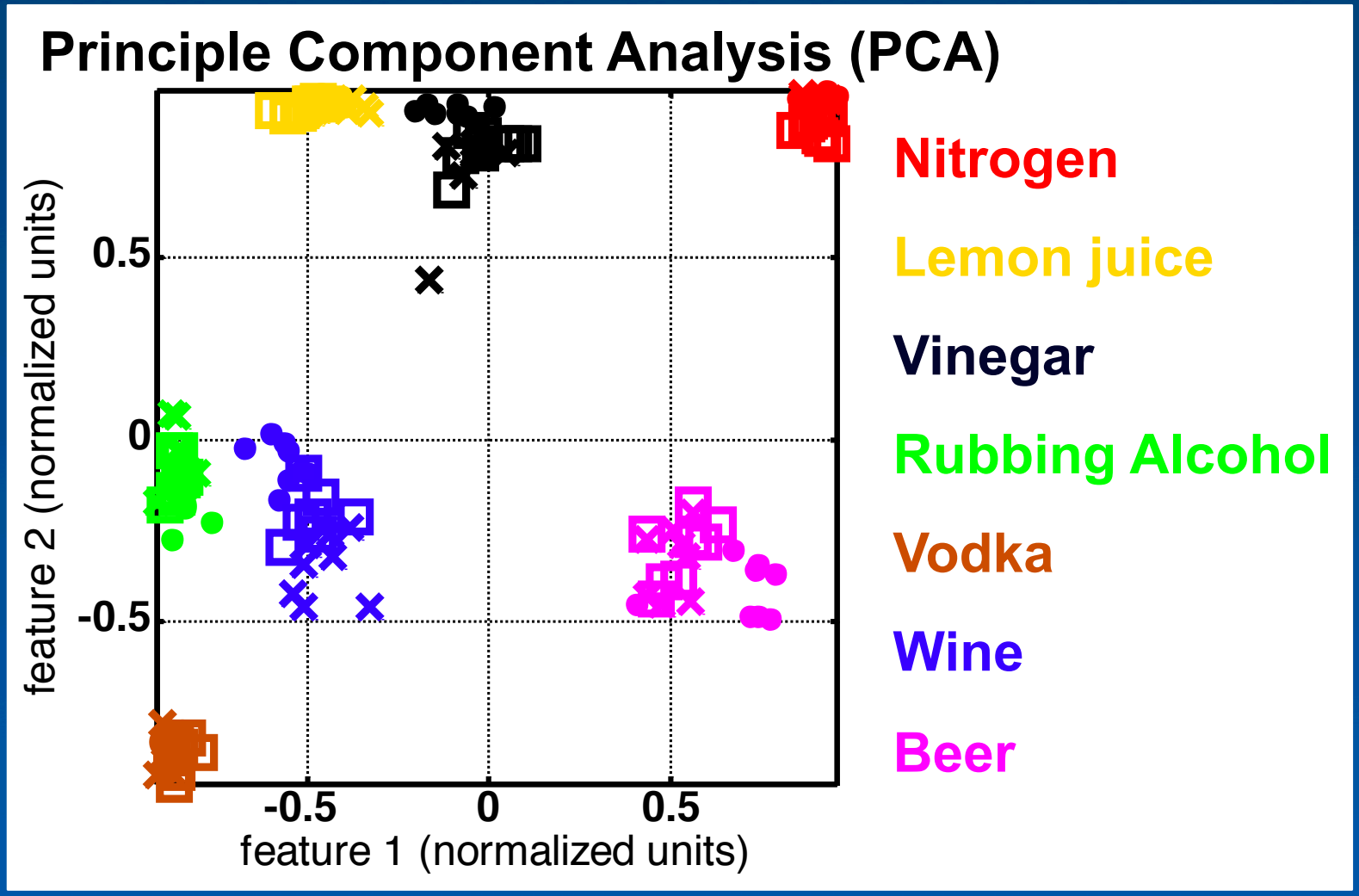
3D NanoSystem

Abundant data: Terabytes / second



[Shulaker Nature 17]

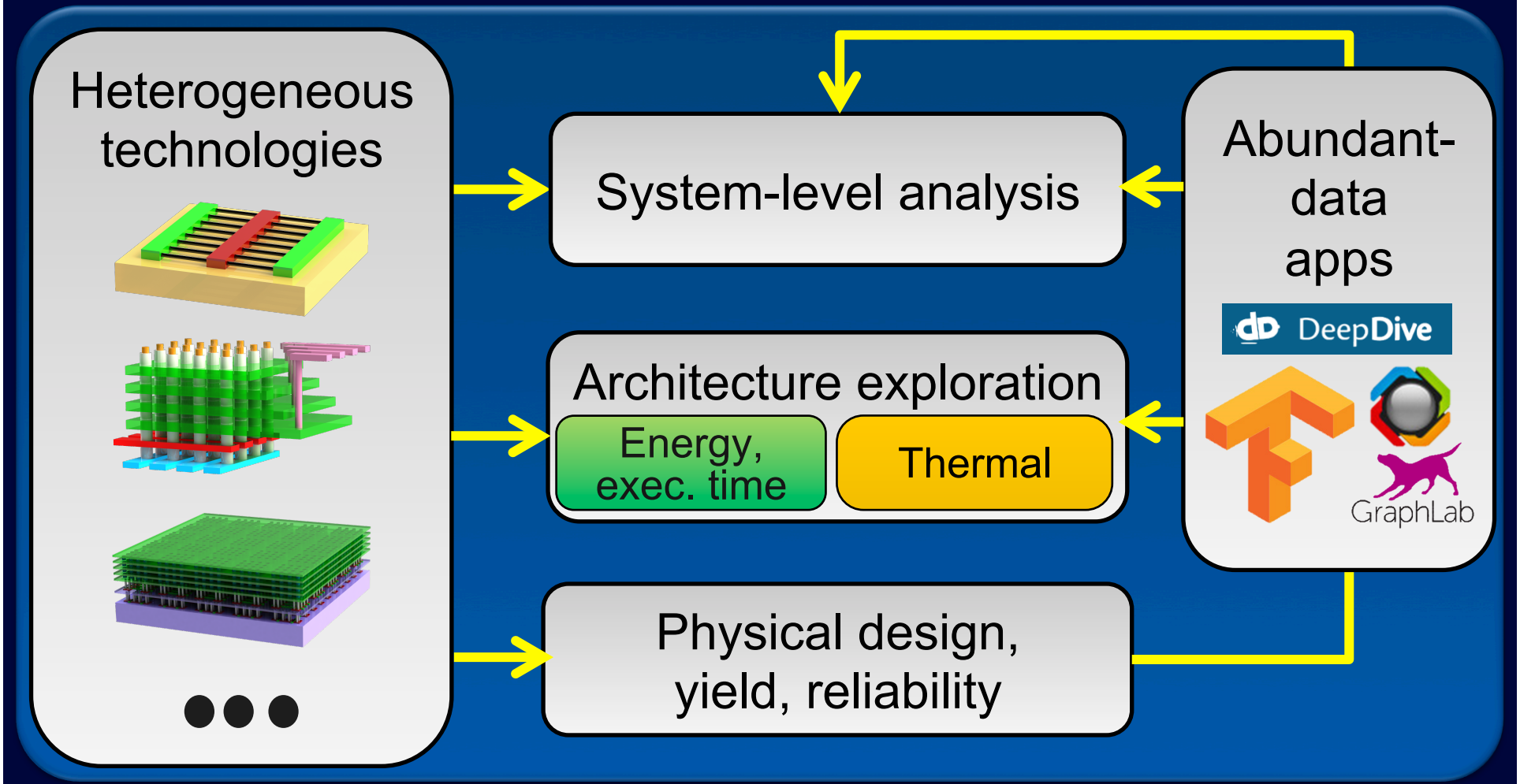
3D NanoSystem Results



[Shulaker Nature 17]

N3XT Simulation Framework

Joint technology, design & app. exploration



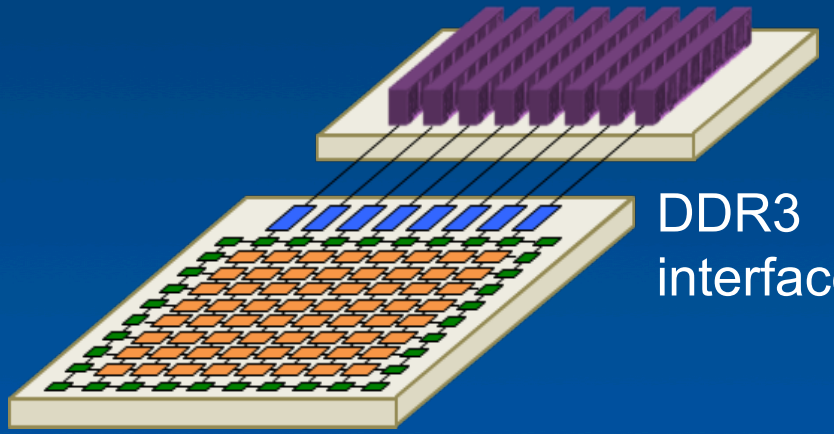
[Aly IEEE Computer 15, Hwang CODES/ISSS 17]

Massive Benefits: Deep Learning, Graph Analytics, …

IBM graph analytics | DeepDive | BigDataBench | TensorFlow | intel

2D

64 GB off-chip DRAM



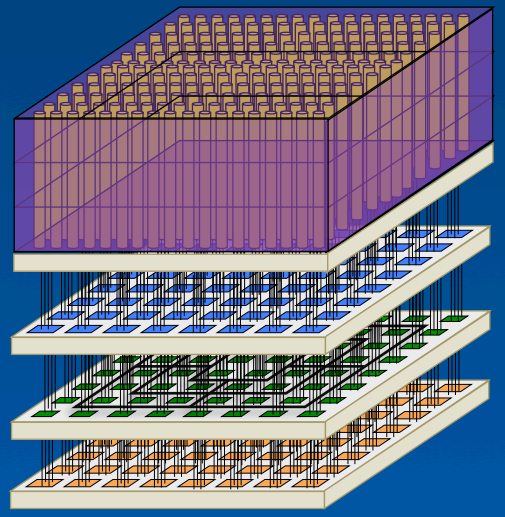
DDR3 interface

64 processor cores

SRAM cache

Single-chip N3XT

64 GB on-chip 3D RRAM



"Simple" interface

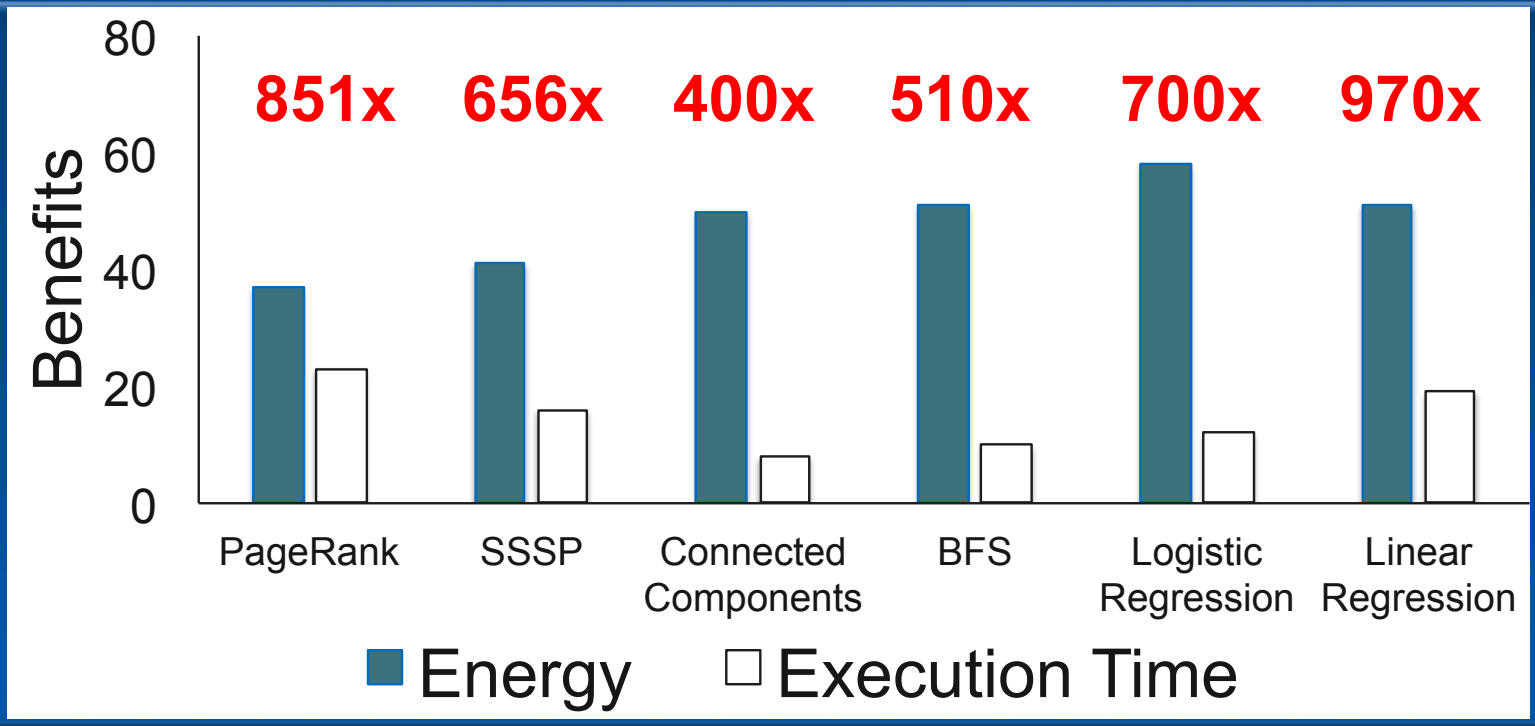
STTRAM cache

64 processor cores

Massive Benefits: Deep Learning, Graph Analytics, …

IBM graph analytics | DeepDive | BigDataBench | TensorFlow | intel

~1,000× benefits, existing software

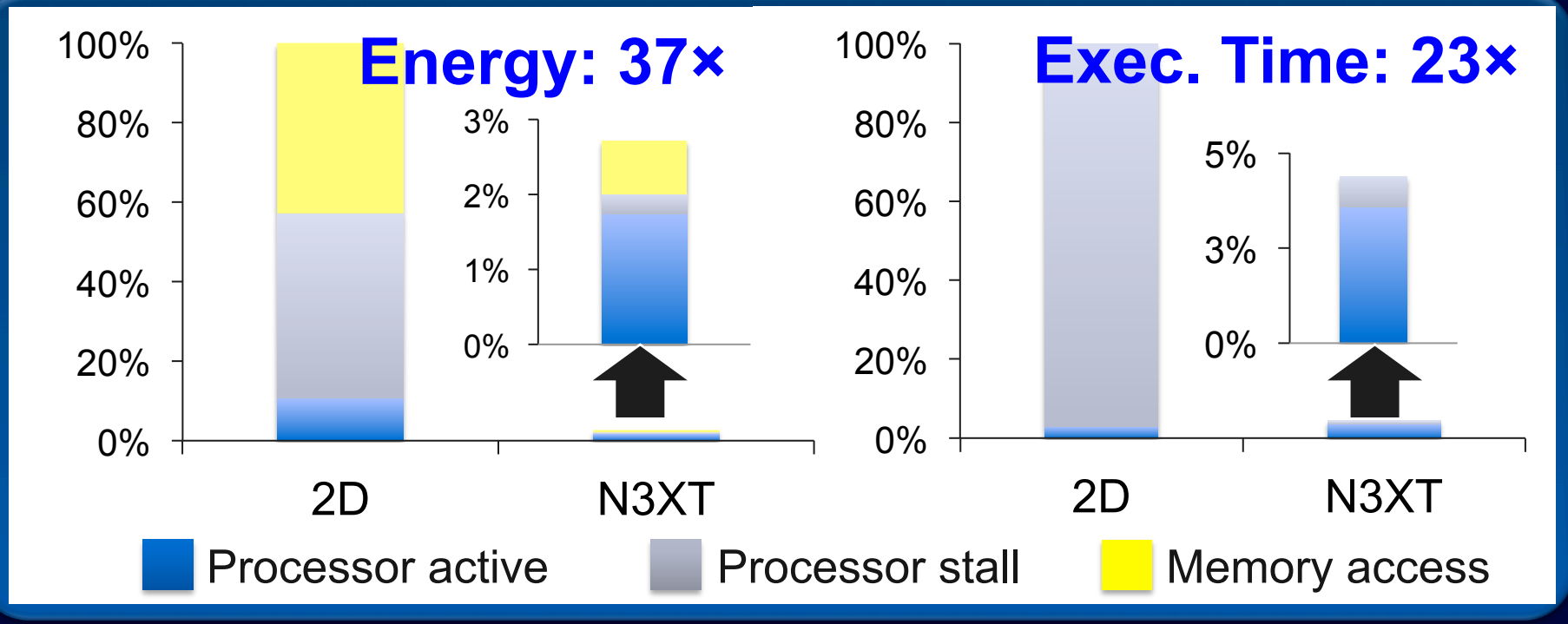


[Aly IEEE Computer 15]

Massive Benefits: Deep Learning, Graph Analytics, …

IBM graph analytics | DeepDive | BigDataBench | TensorFlow | intel

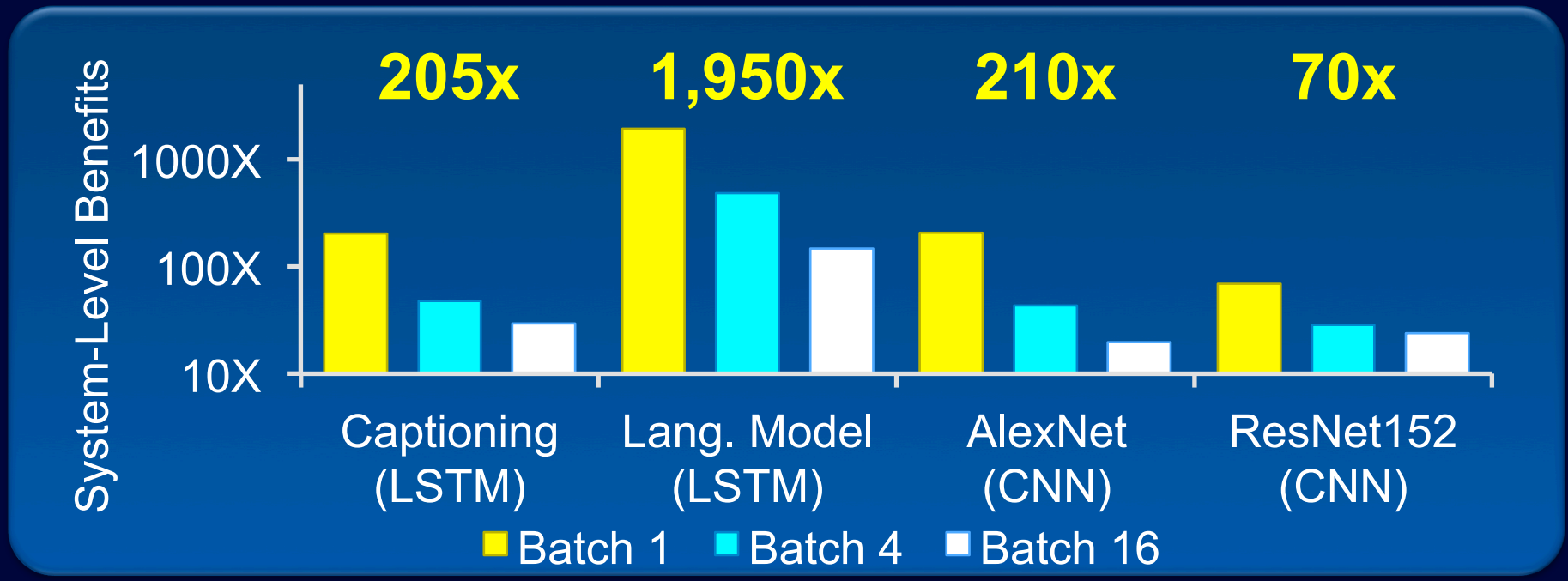
851× benefits



PageRank app.

Massive Benefits: Deep Learning, Graph Analytics, …

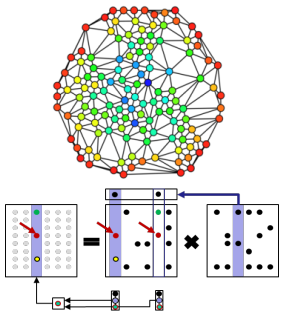
- 100× – 1,000× benefits (energy × execution time)
  - Deep learning accelerators



Chip stacking: 2 - 4× benefits

[Hwang CODES/ISSS 17] LSTM: Long Short Term Memory, CNN: Convolutional Neural Network

Complement with Software Solutions



DSL compiler

Learning: key architectural concept

Cross-Layer Resilience

Co-optimized s/w + h/w

Runtime optimization

Yield, reliability

DSL = Domain-Specific Language

More Opportunities

Co-optimized hardware + software

Brain-inspired

Technology innovations

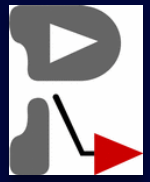
"Brain-Inspired Computing Exploiting Carbon Nanotube FETs and Resistive RAM: Hyperdimensional Computing Case Study," *ISSCC* 2018.

Outline

- Robust operation: silicon CMOS reliability
- Beyond silicon
- Conclusion

Thanks to Research Group

Thanks to Sponsors & Collaborators




Conclusion

- Robust systems
  - New solutions: elegantly simple, effective

Silicon CMOS Reliability	Beyond silicon
BISER, LEAP	CNFET nanosystems
Failure prediction	*N3XT* monolithic 3D
CLEAR cross-layer	1,000X opportunity