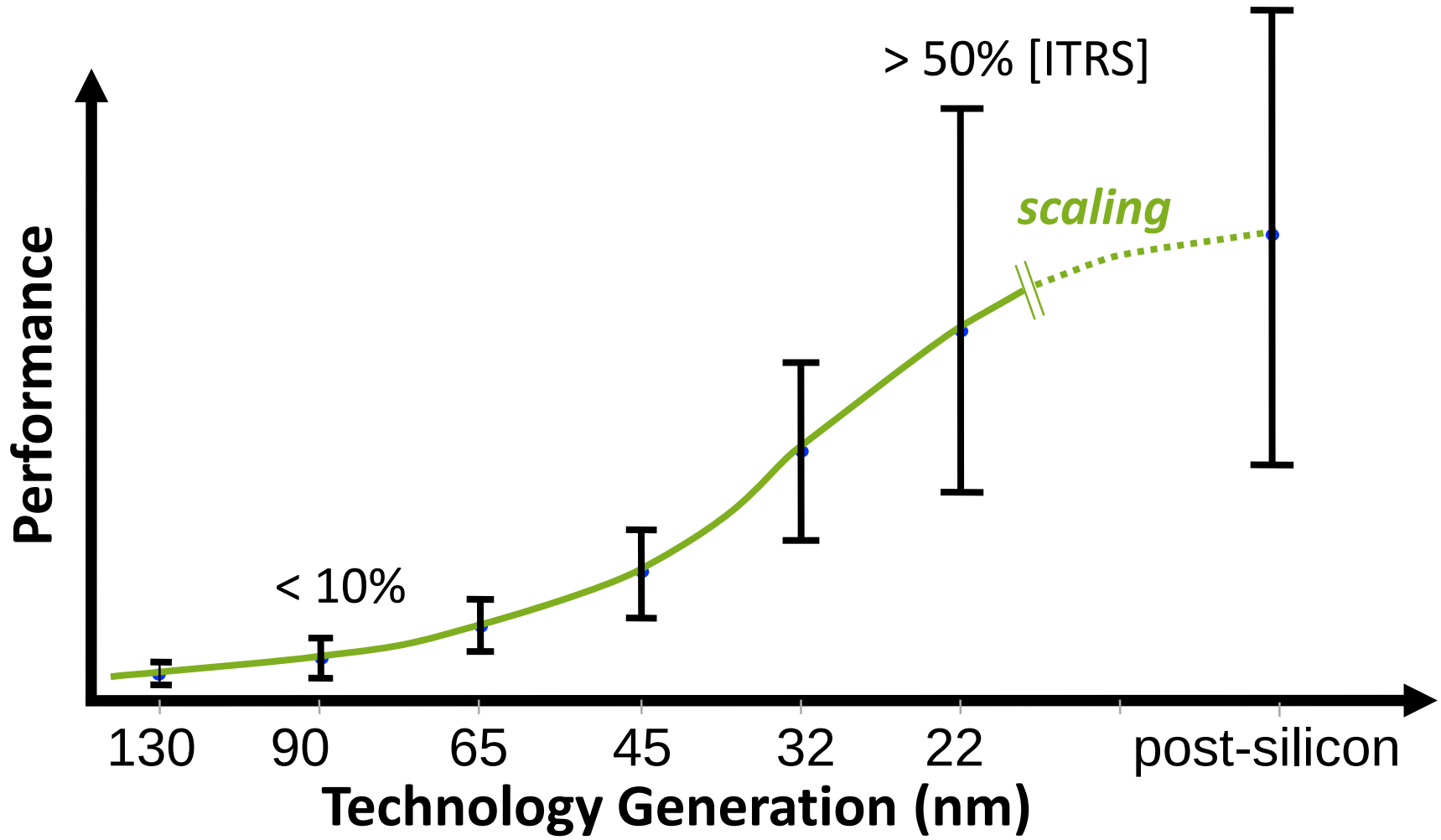
From Variability-Tolerance to Approximate Computing in Parallel Computing Architectures

Abbas Rahimi

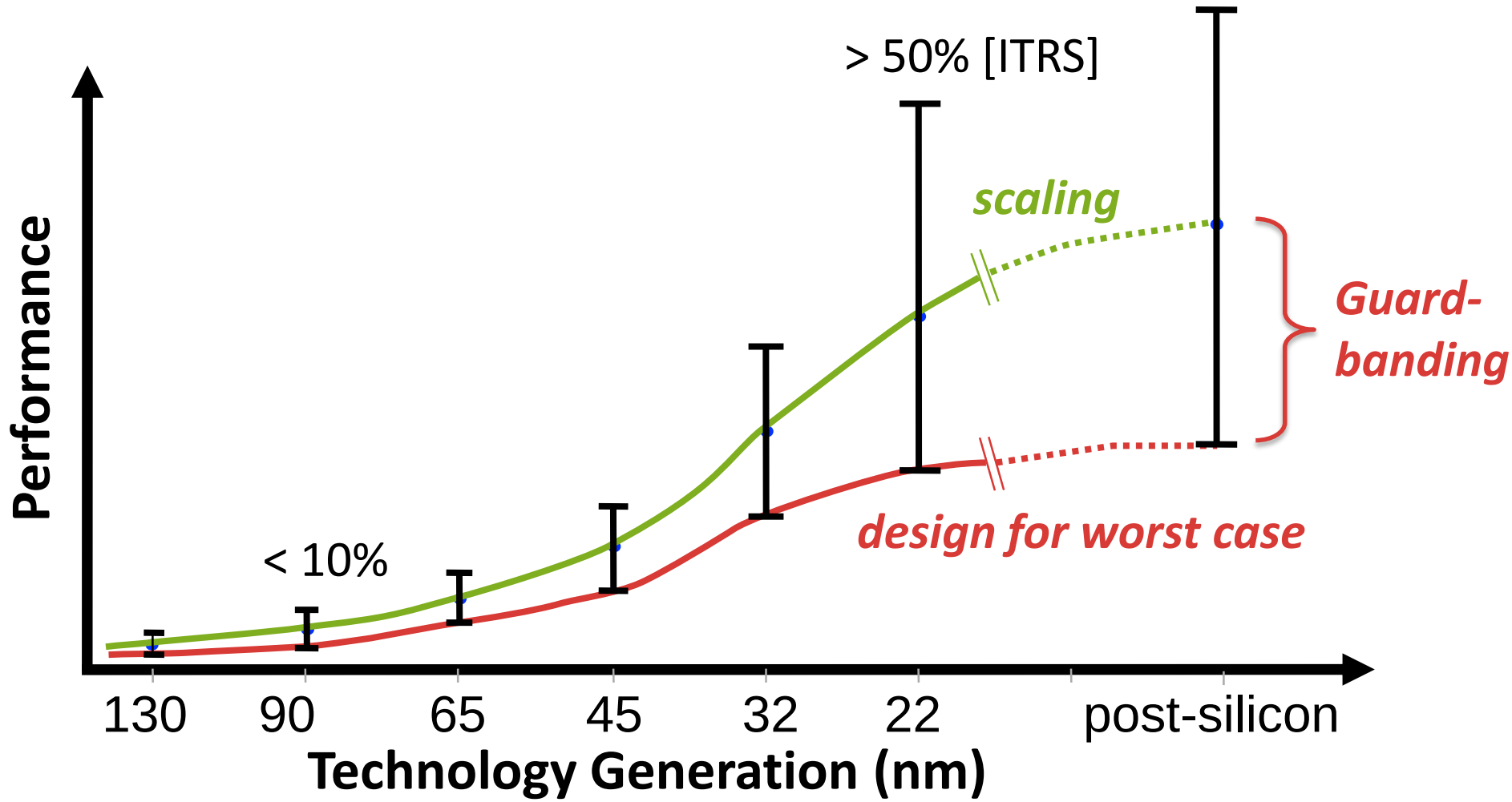
Microelectronic Embedded Systems Laboratory

July 22, 2015

Ever-Increasing Variability in Semiconductor



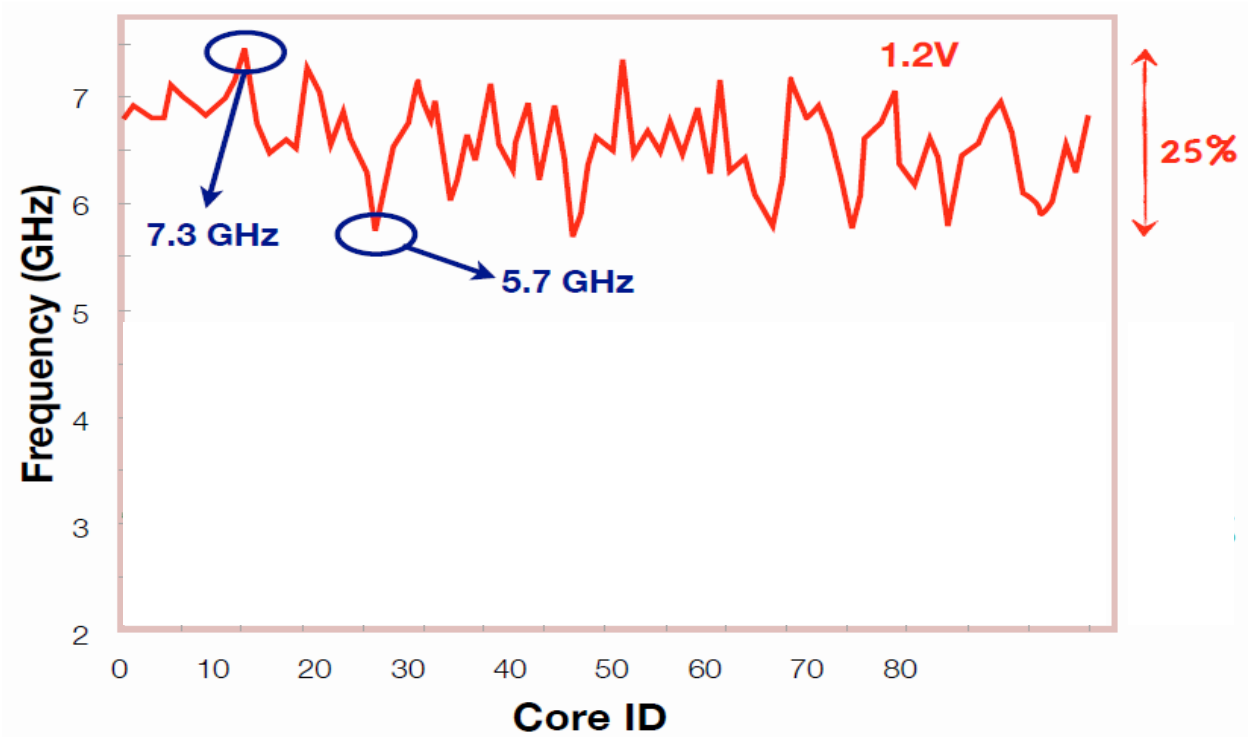
Ever-Increasing Variability in Semiconductor



Guardbanding leads to loss of operational efficiency!

Sources of Variability

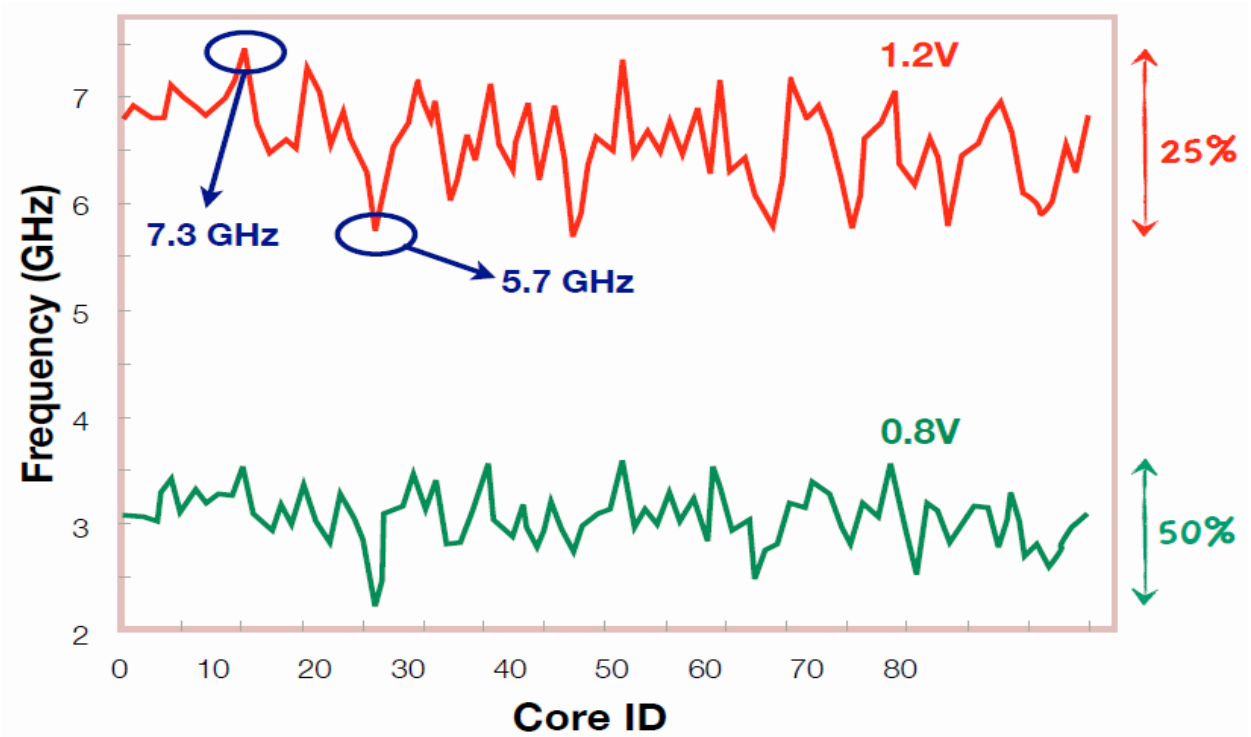
I. Static: Manufacturing process



An Intel 80-core processor within a single die in 65nm, *JSSC'11*

Sources of Variability

I. Static: Manufacturing process



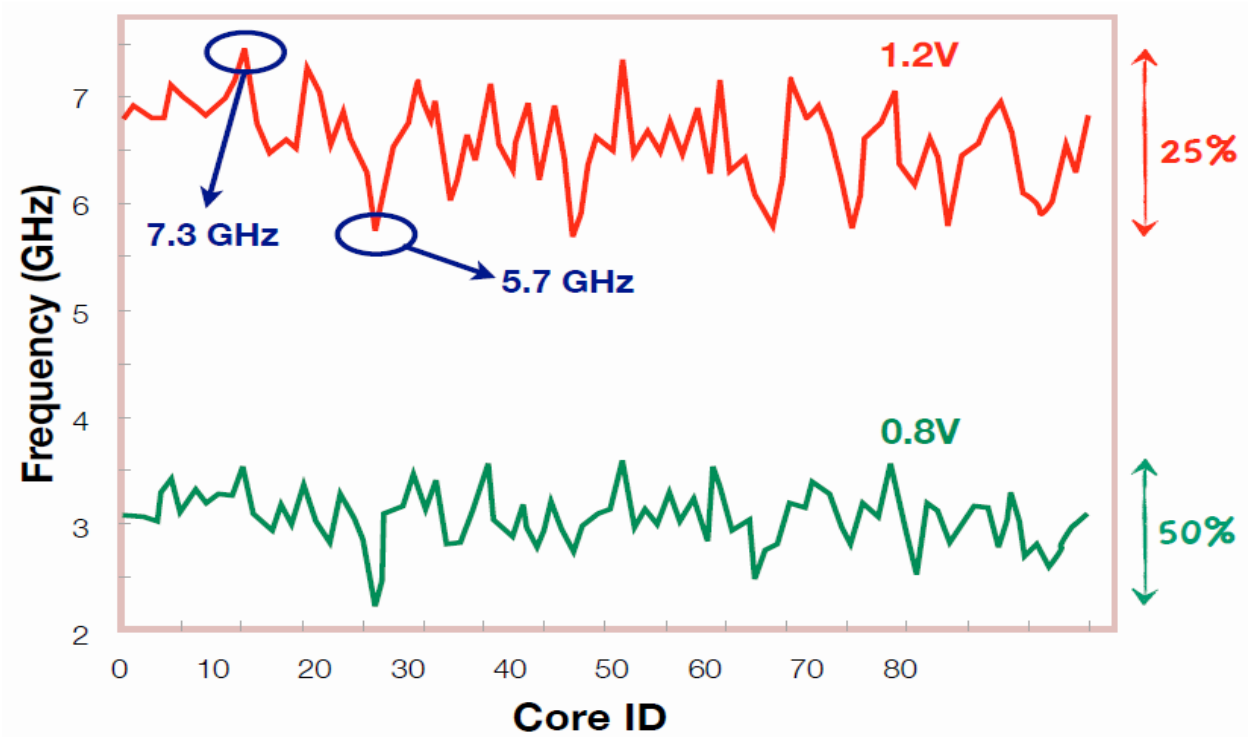
An Intel 80-core processor within a single die in 65nm, *JSSC'11*

Sources of Variability

I. Static: Manufacturing process

II. Dynamic: Voltage and temperature

III. Temporal: Aging (wearout)



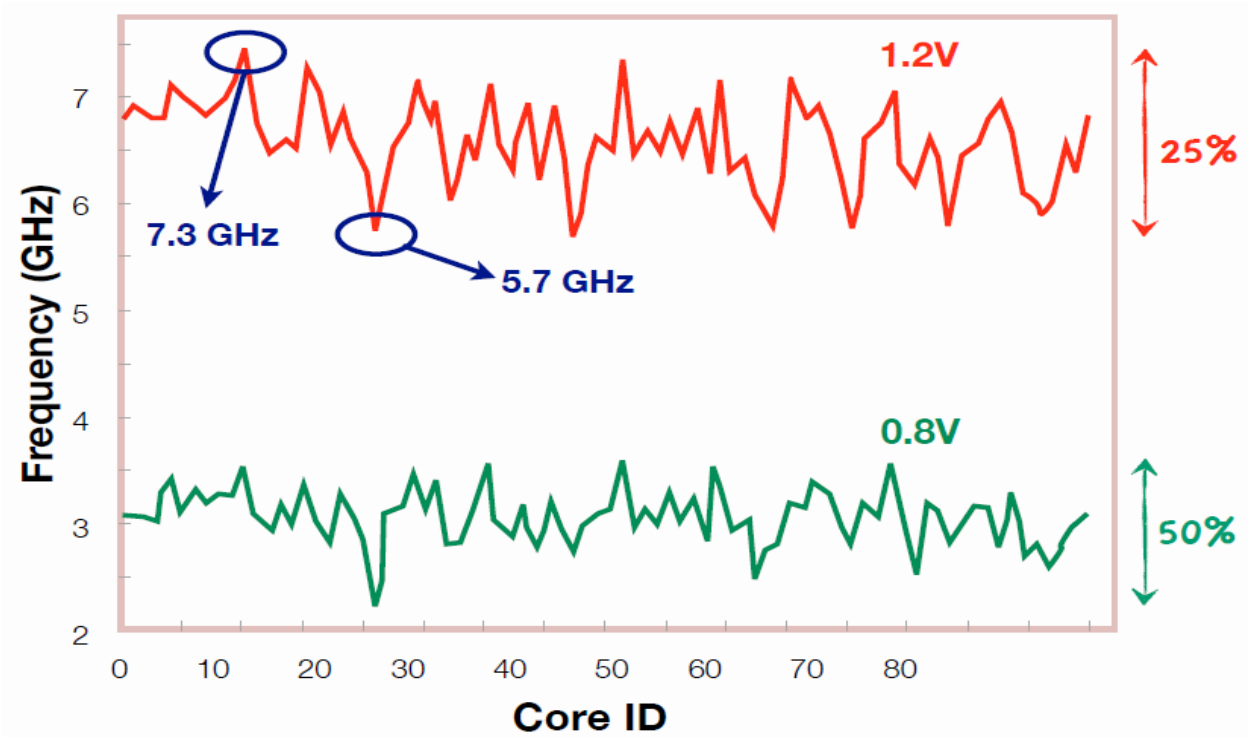
An Intel 80-core processor within a single die in 65nm, *JSSC'11*

Sources of Variability

I. Static: Manufacturing process
II. Dynamic: Voltage and temperature
III. Temporal: Aging (wearout)

~20x performance variation

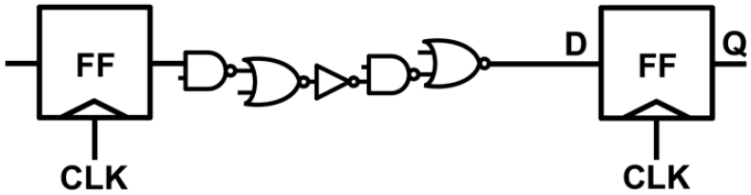
Dreslinski, *Proc. IEEE'10*



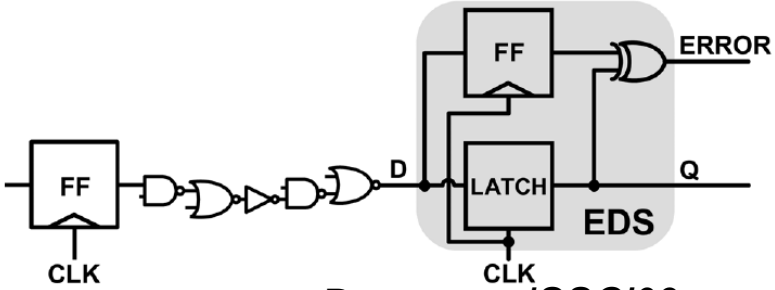
An Intel 80-core processor within a single die in 65nm, *JSSC'11*

Reduced Guardband Causes Timing Error

Reducing guardband



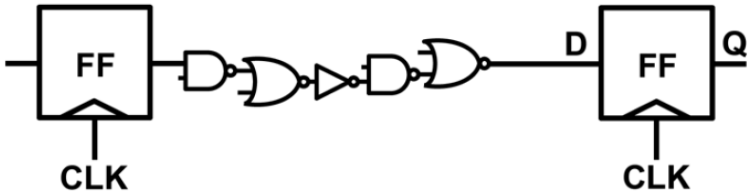
Timing error



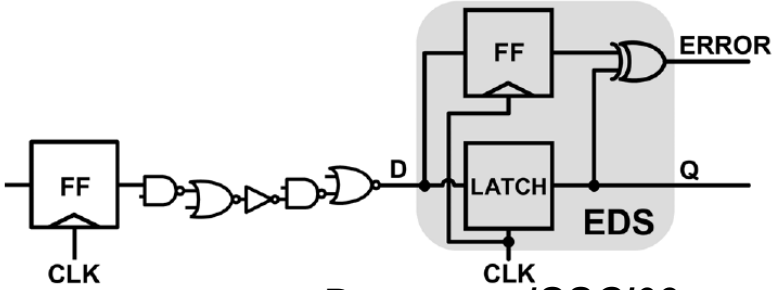
Bowman, *JSSC'09*

Reduced Guardband Causes Timing Error

Reducing guardband



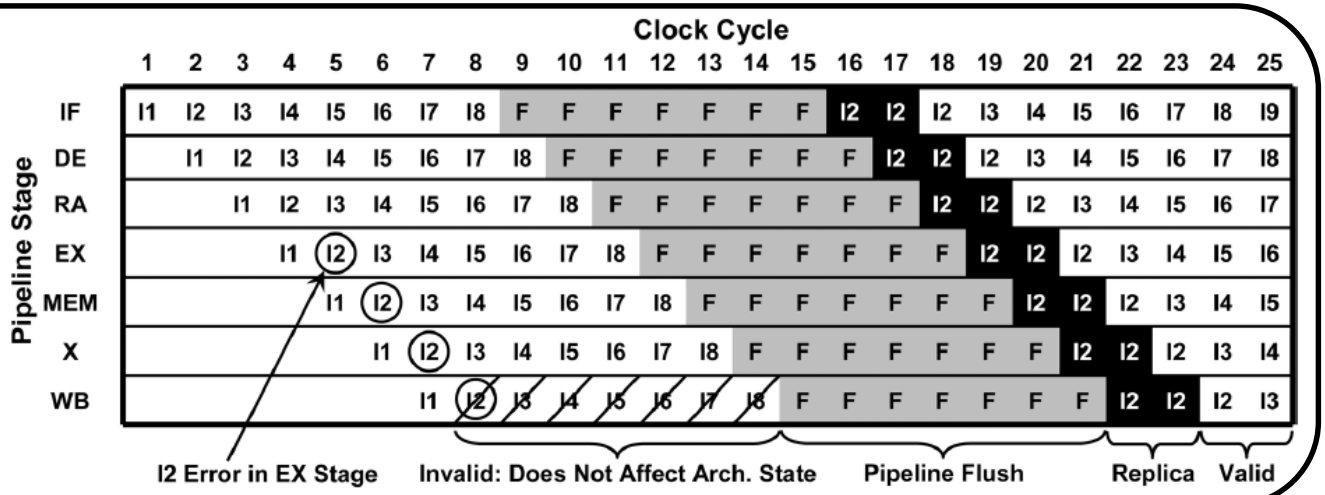
Timing error



Bowman, *JSSC'09*

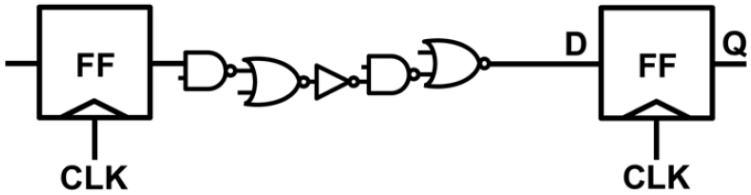
Costly error recovery

3×N recovery cycles!
N = # of stages
Bowman, *JSSC'11*

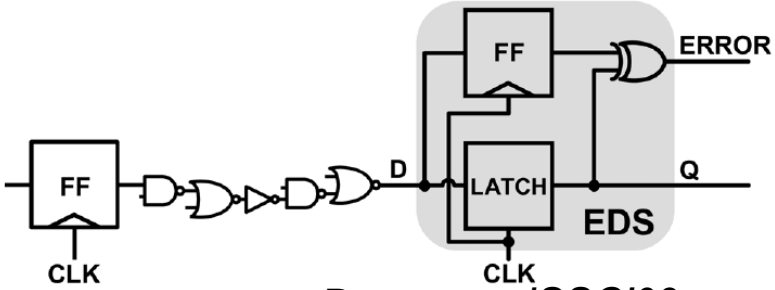


Reduced Guardband Causes Timing Error

Reducing guardband

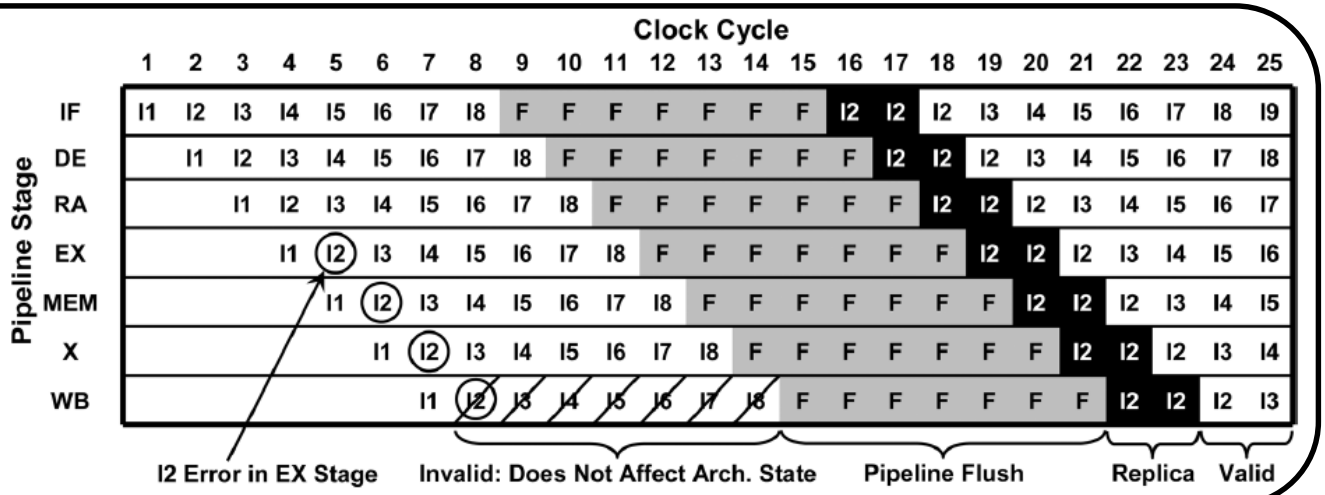


Timing error



Bowman, *JSSC'09*

Costly error recovery

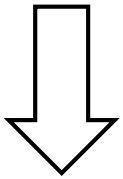


3×N recovery cycles!
N = # of stages
Bowman, *JSSC'11*

Grand challenge: **low-cost** and **scalable** variability-tolerance

Reducing Guardband by

Sensing Circuits and Architectural Variability-Tolerance

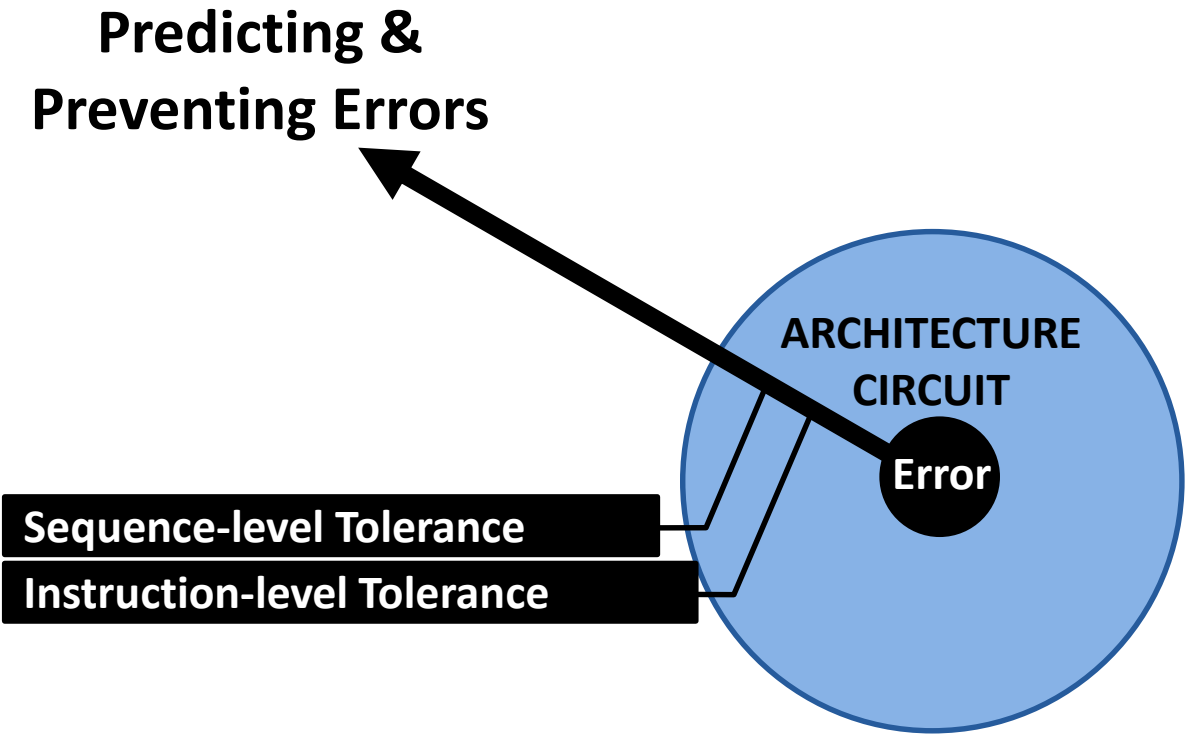


Software Techniques to Enhance Entire of Computing Stack

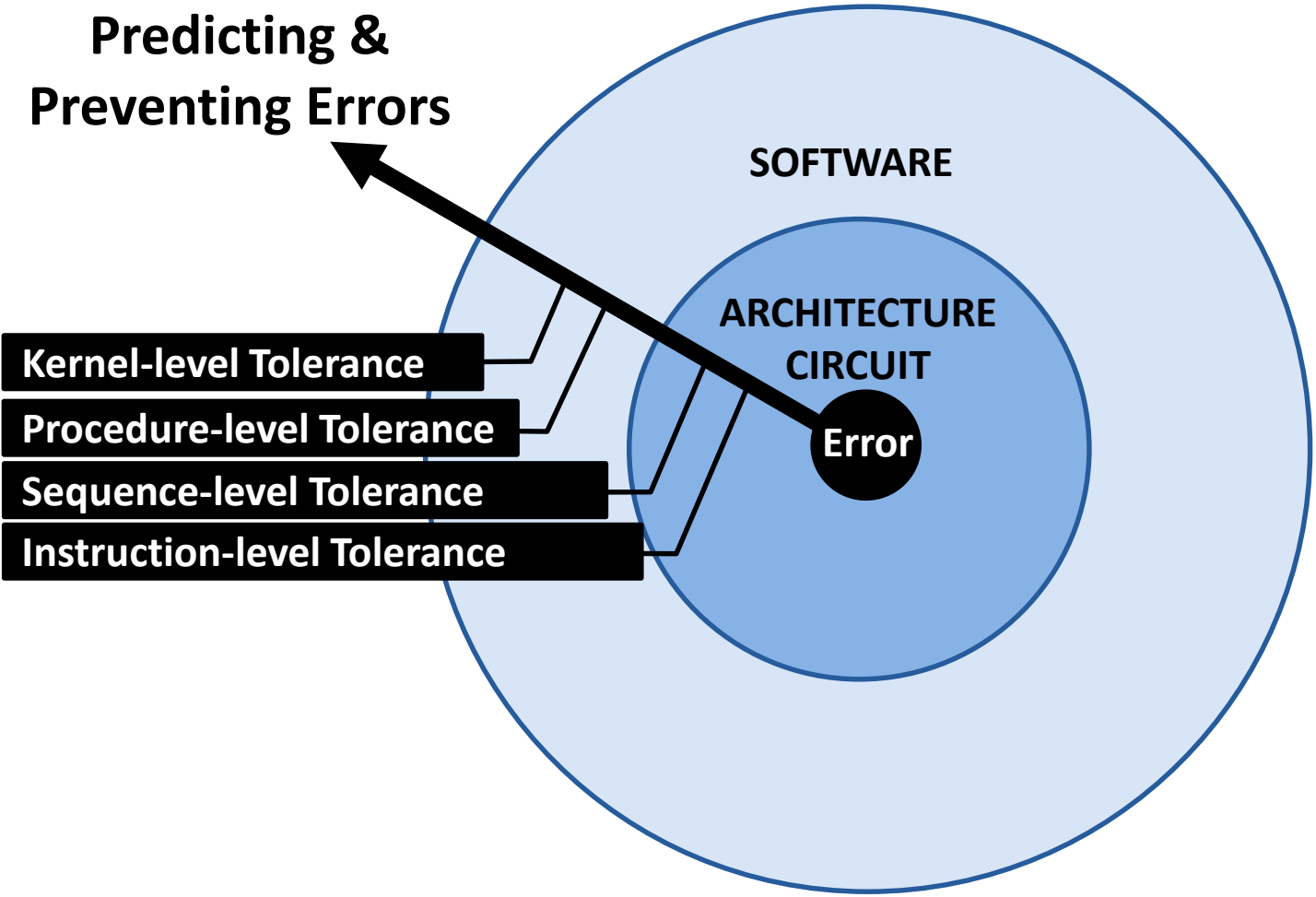
Our Approach to Variability-Tolerance

Error

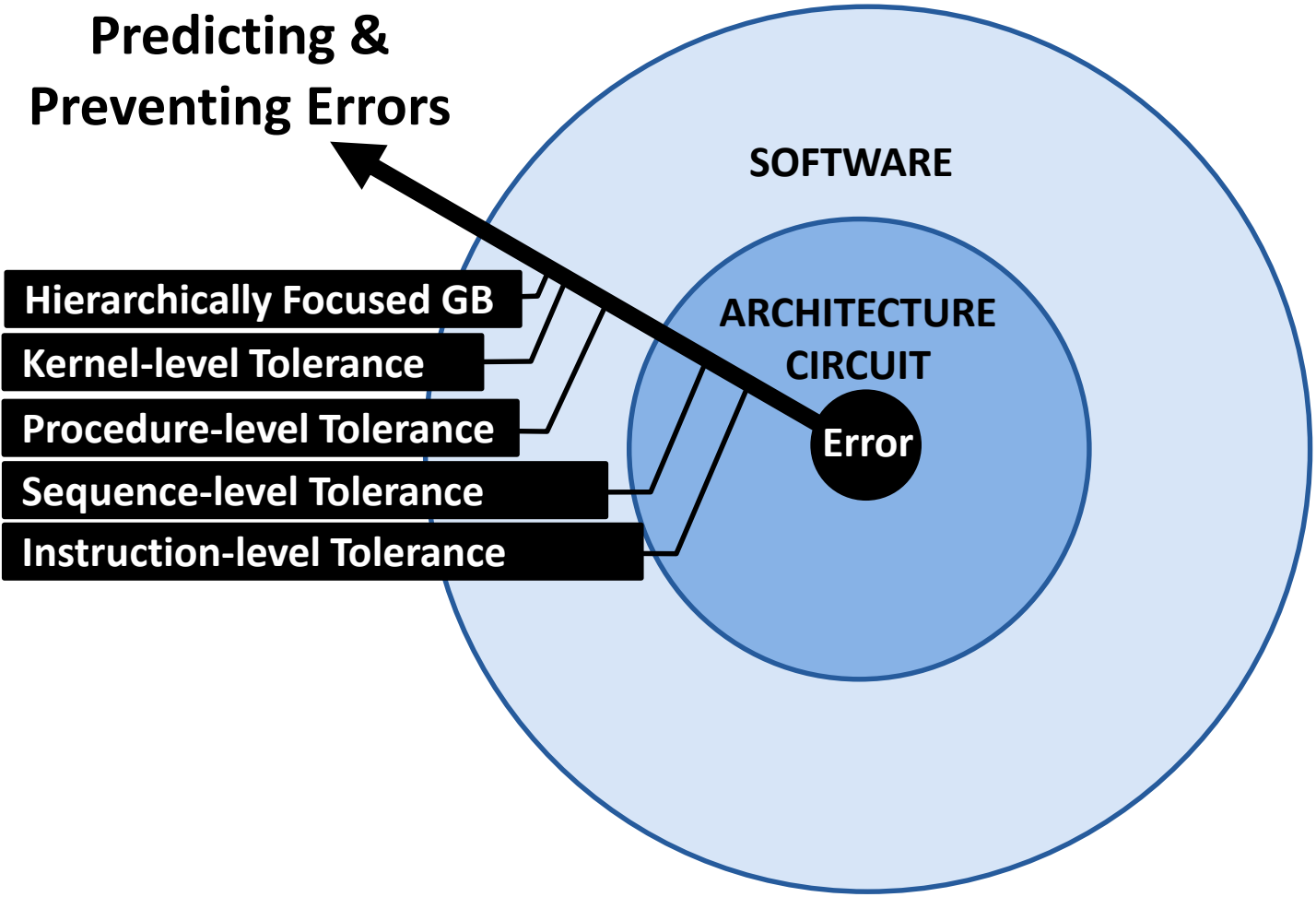
Our Approach to Variability-Tolerance



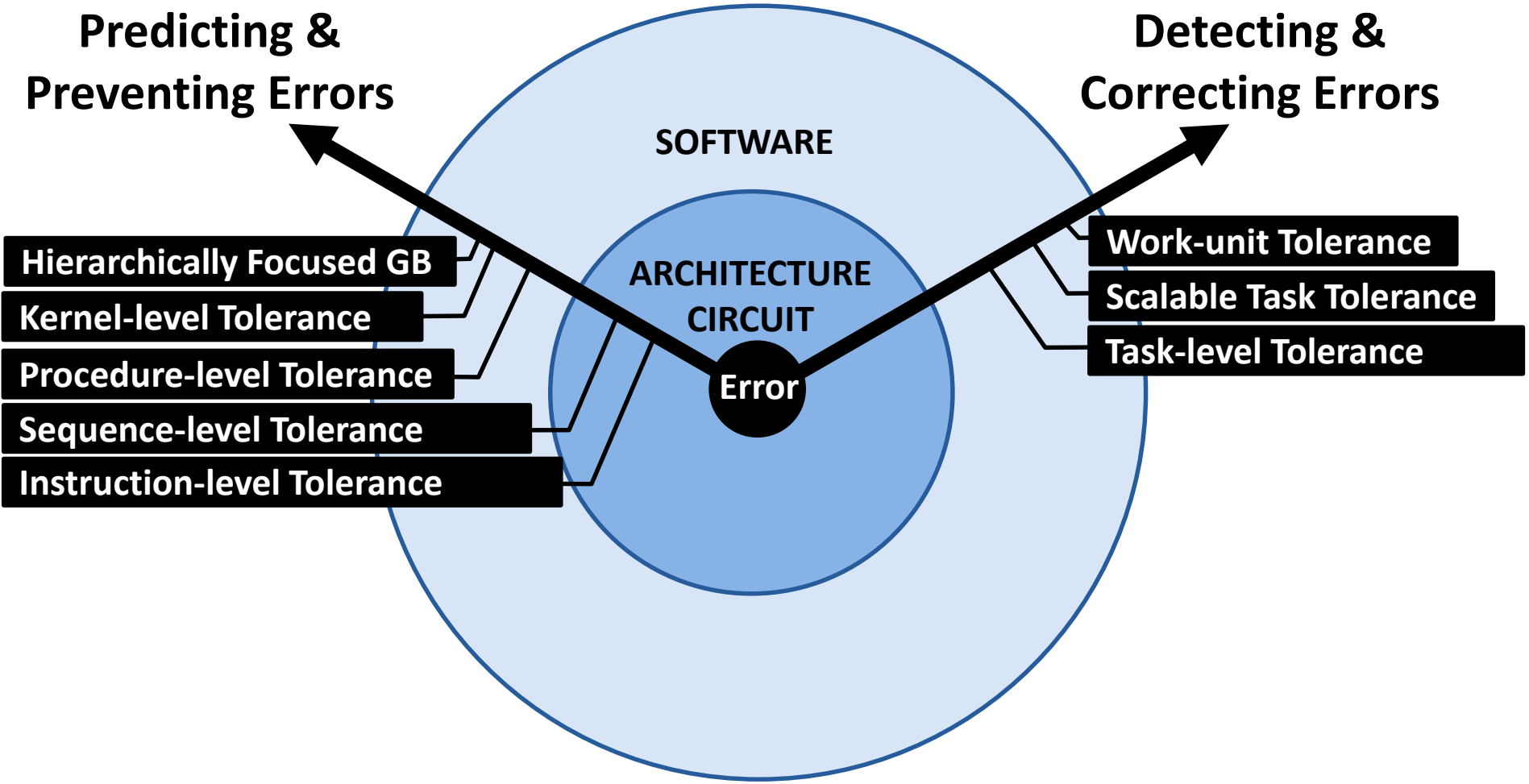
Our Approach to Variability-Tolerance



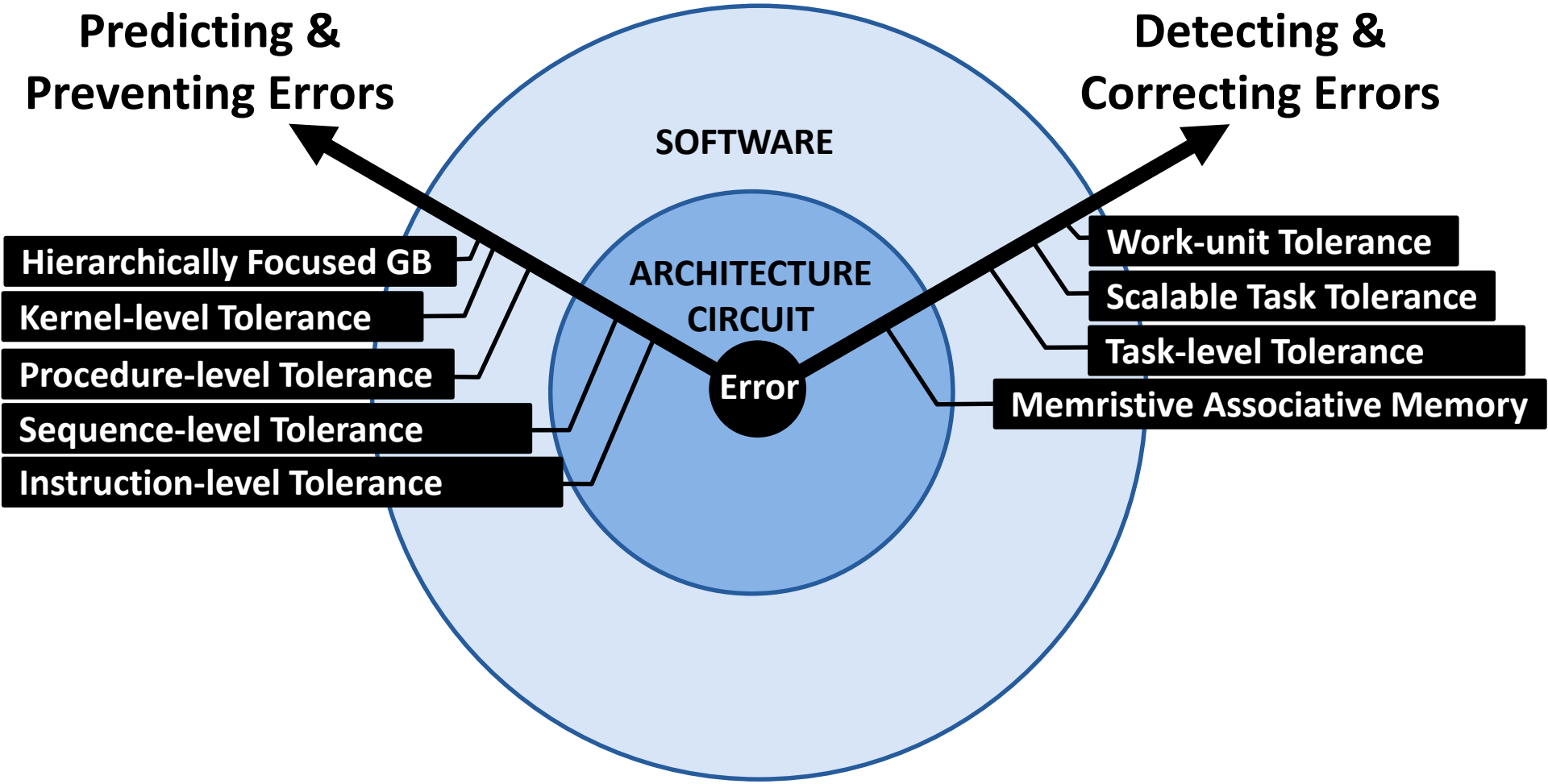
Our Approach to Variability-Tolerance



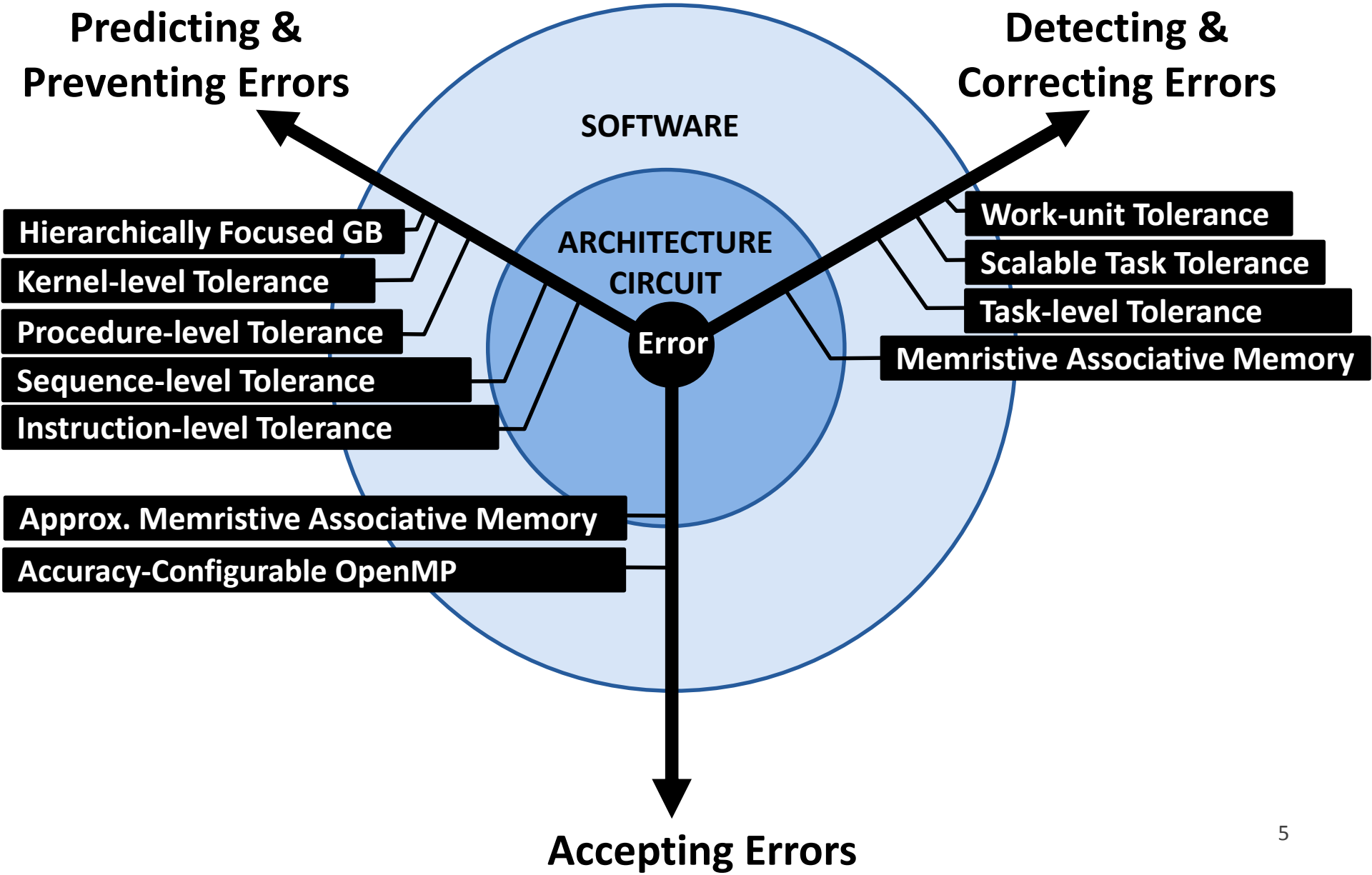
Our Approach to Variability-Tolerance



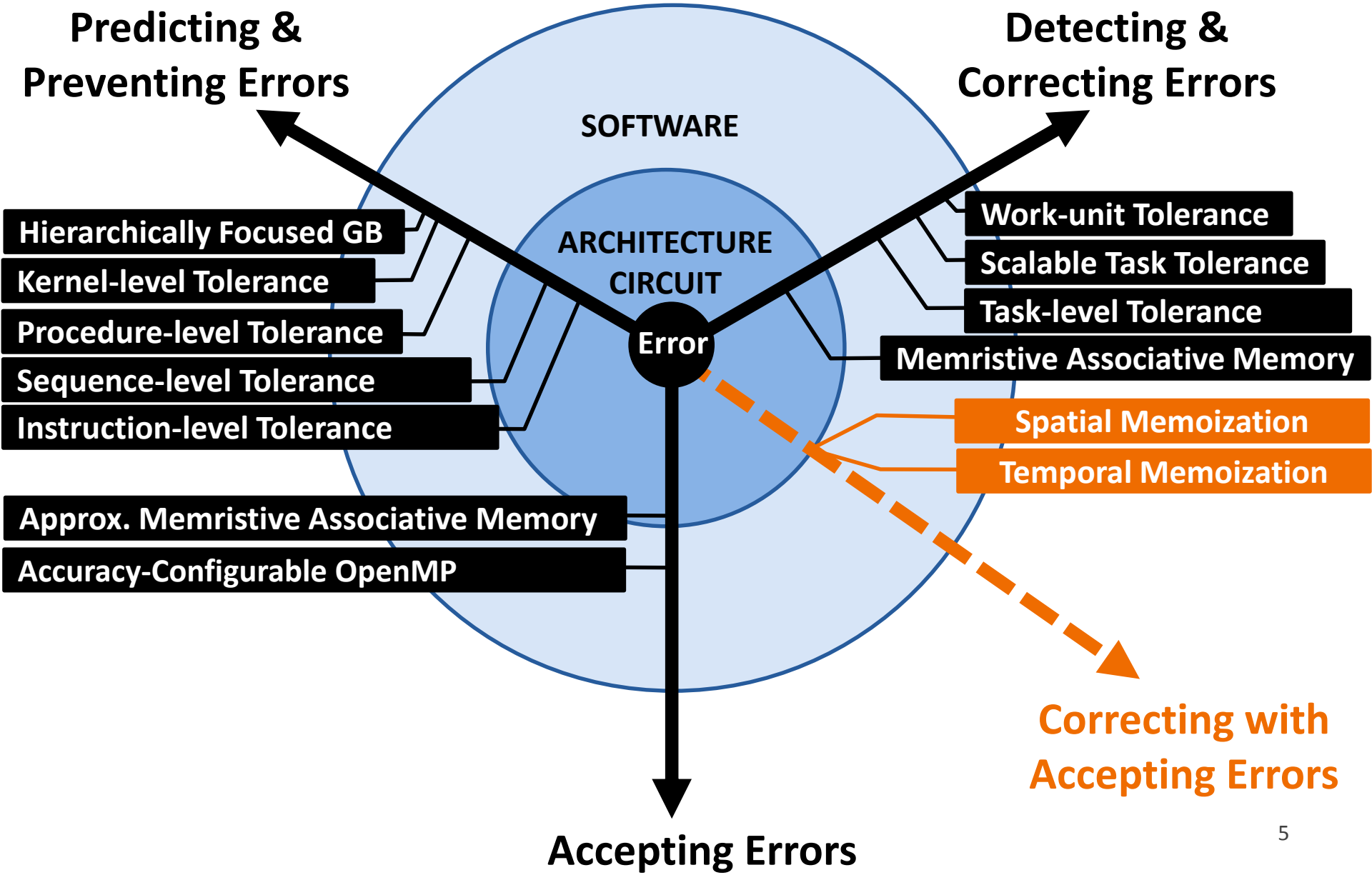
Our Approach to Variability-Tolerance



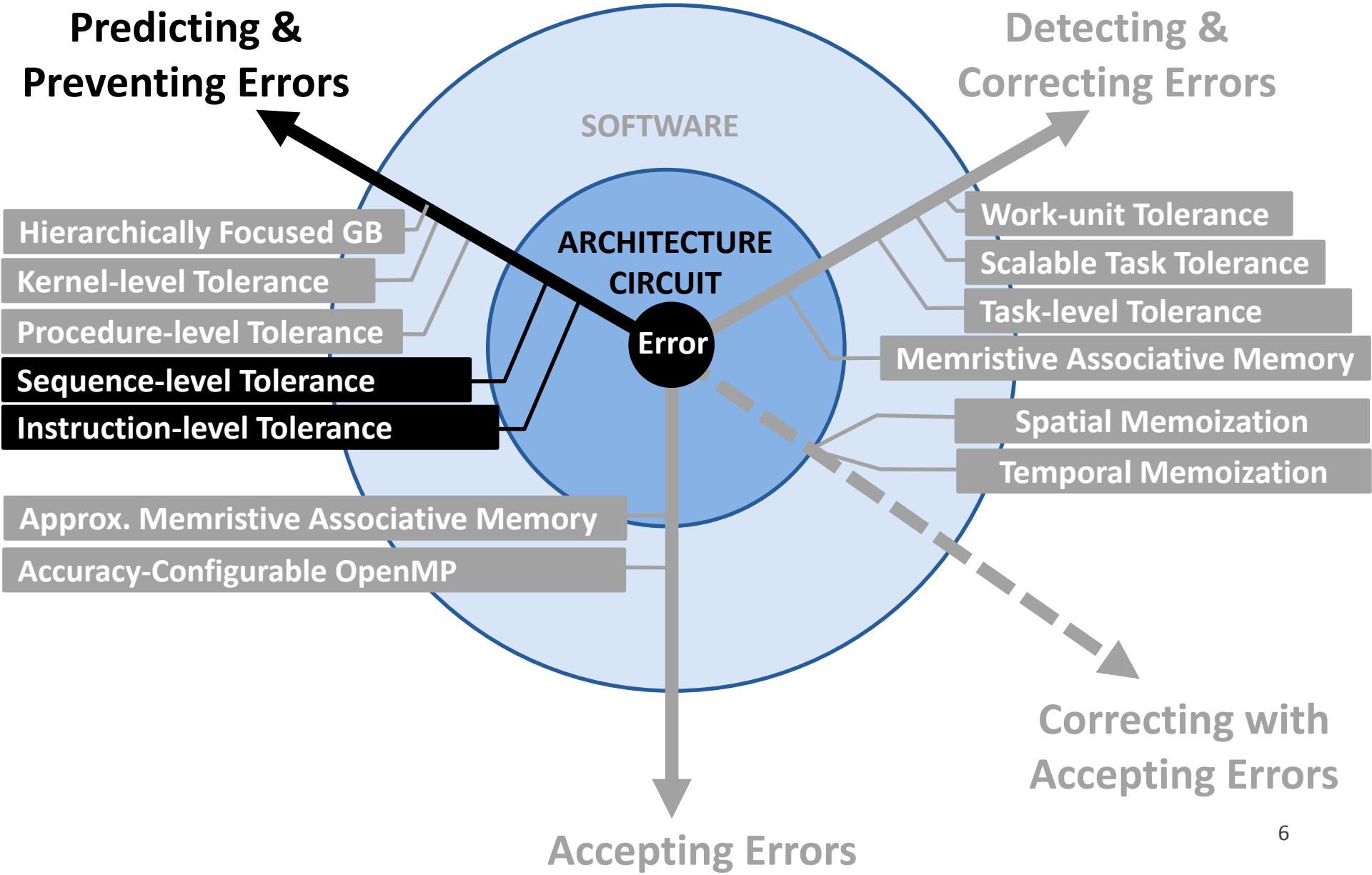
Our Approach to Variability-Tolerance



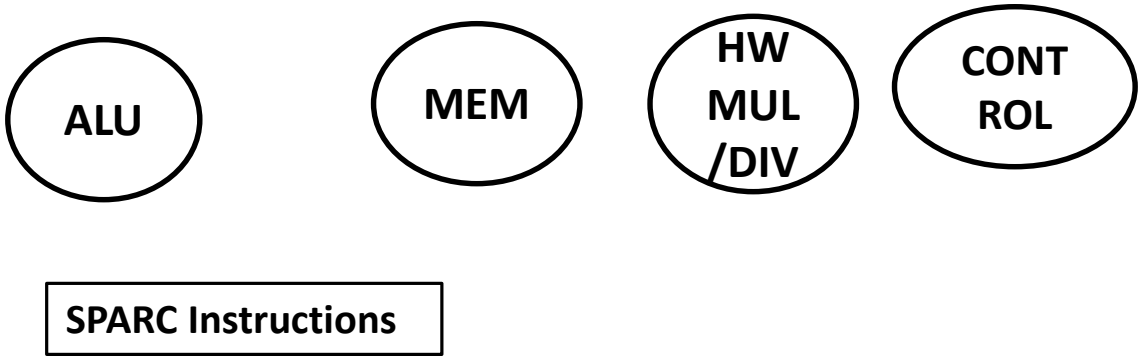
Our Approach to Variability-Tolerance



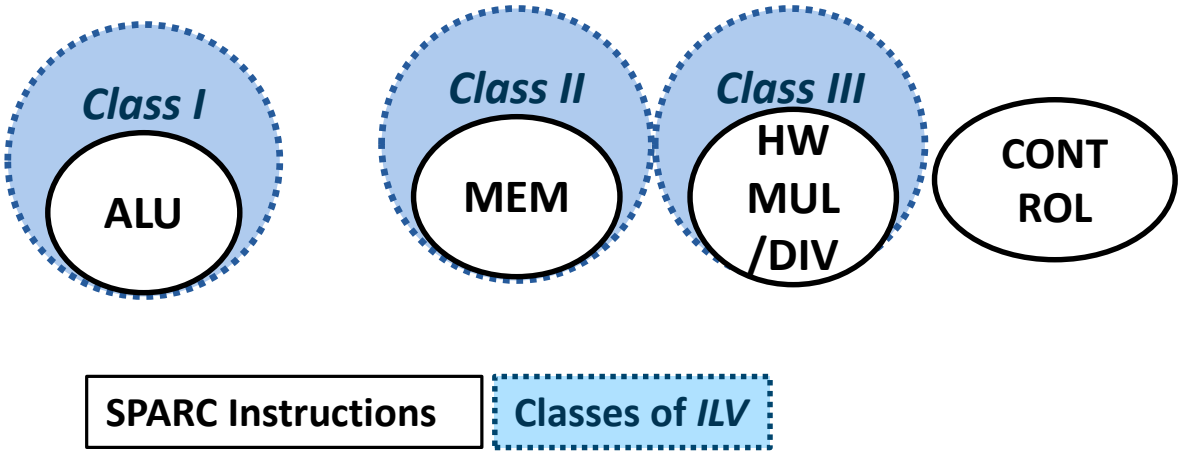
Our Approach to Variability-Tolerance



Instruction/Sequence-Level Classifications

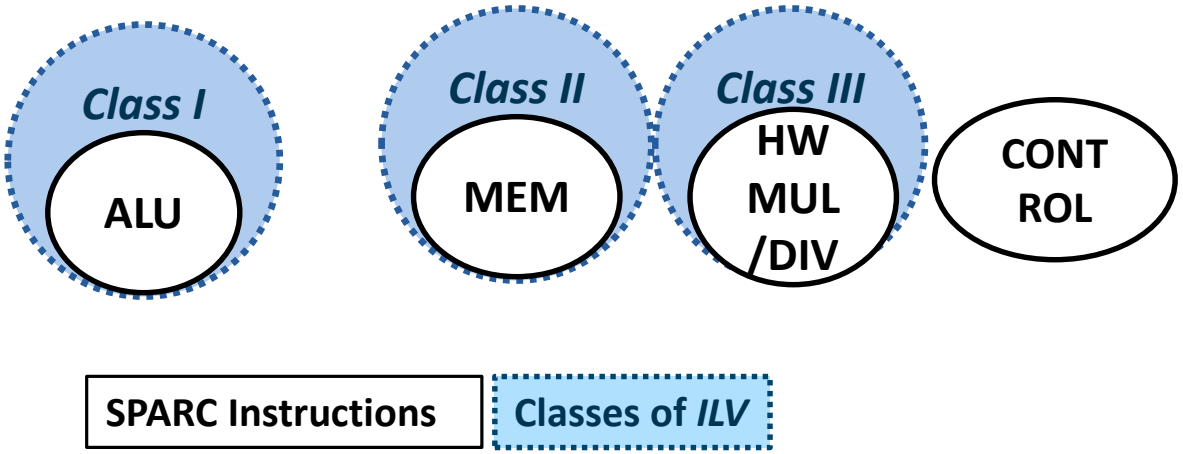


Instruction/Sequence-Level Classifications



ILV(*Class III*) ≥ ILV(*Class II*) ≥ ILV(*Class I*)

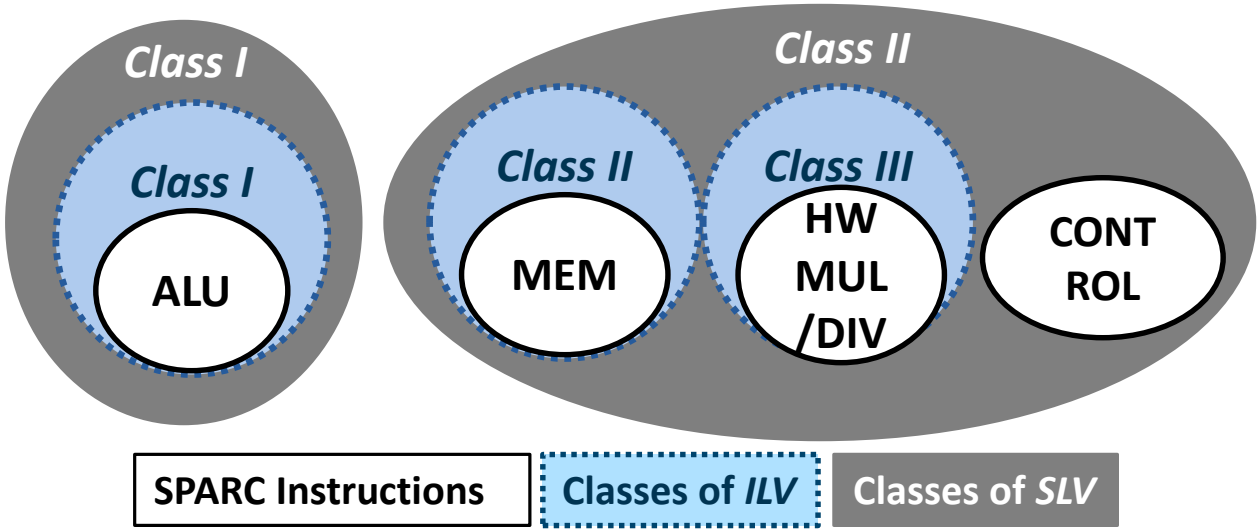
Instruction/Sequence-Level Classifications



(0.81V, 125°C)		
Clk (ns)	*Class II*	*Class I*
1.26	1	1
1.27	1	0.69
1.28	1	*0*
1.29	1	*0*
1.30	1	*0*
1.31	1	*0*
1.32	1	*0*
1.33	0.81	*0*
1.34	0.81	*0*
1.35	*0*	*0*

ILV(*Class III*) ≥ ILV(*Class II*) ≥ ILV(*Class I*)

Instruction/Sequence-Level Classifications

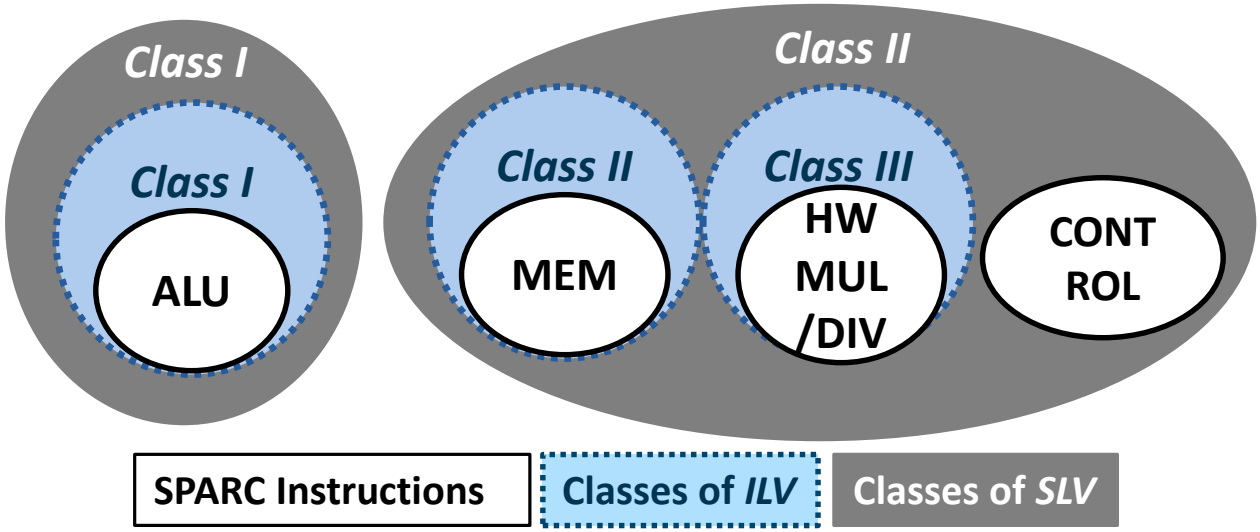


(0.81V, 125°C)		
Clk (ns)	*Class II*	*Class I*
1.26	1	1
1.27	1	0.69
1.28	1	*0*
1.29	1	*0*
1.30	1	*0*
1.31	1	*0*
1.32	1	*0*
1.33	0.81	*0*
1.34	0.81	*0*
1.35	*0*	*0*

ILV(*Class III*) ≥ ILV(*Class II*) ≥ ILV(*Class I*)

SLV (*Class II*) ≥ SLV (*Class I*)

Instruction/Sequence-Level Classifications

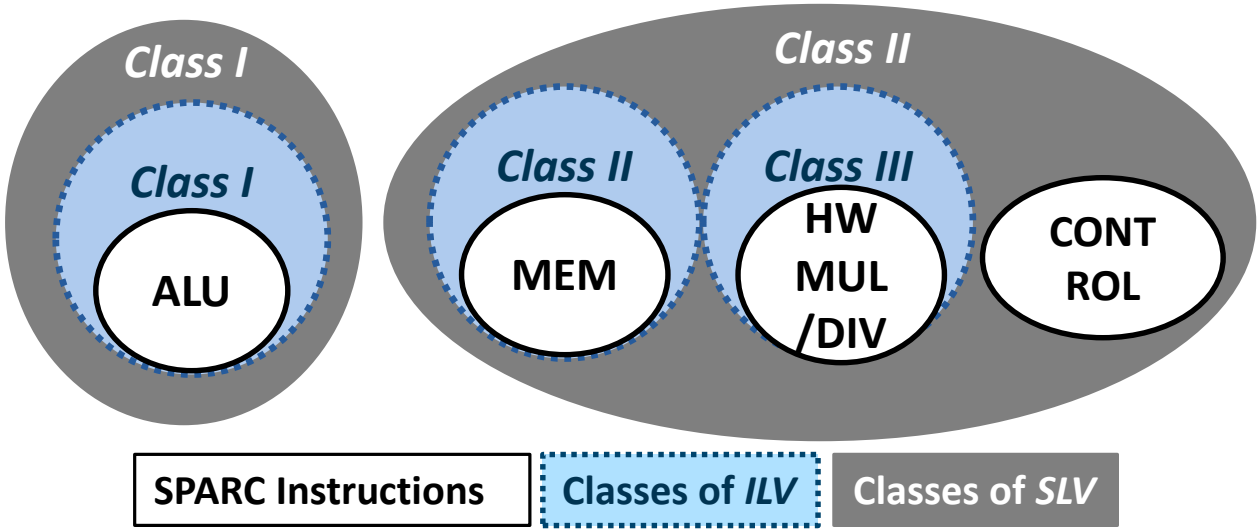


(0.81V, 125°C)			(0.72V, 125°C)		
Clk (ns)	*Class II*	*Class I*	Clk (ns)	*Class II*	*Class I*
1.26	1	1	1.78	1	1
1.27	1	0.69	1.79	1	0.58
1.28	1	*0*	1.80	1	*0*
1.29	1	*0*	1.81	1	*0*
1.30	1	*0*	1.82	1	*0*
1.31	1	*0*	1.83	1	*0*
1.32	1	*0*	1.84	0.81	*0*
1.33	0.81	*0*	1.85	0.13	*0*
1.34	0.81	*0*	1.86	*0*	*0*
1.35	*0*	*0*	1.87	*0*	*0*

ILV(*Class III*) ≥ ILV(*Class II*) ≥ ILV(*Class I*)

SLV (*Class II*) ≥ SLV (*Class I*)

Instruction/Sequence-Level Classifications



(0.81V, 125°C)			(0.72V, 125°C)		
Clk (ns)	*Class II*	*Class I*	Clk (ns)	*Class II*	*Class I*
1.26	1	1	1.78	1	1
1.27	1	0.69	1.79	1	0.58
1.28	1	*0*	1.80	1	*0*
1.29	1	*0*	1.81	1	*0*
1.30	1	*0*	1.82	1	*0*
1.31	1	*0*	1.83	1	*0*
1.32	1	*0*	1.84	0.81	*0*
1.33	0.81	*0*	1.85	0.13	*0*
1.34	0.81	*0*	1.86	*0*	*0*
1.35	*0*	*0*	1.87	*0*	*0*

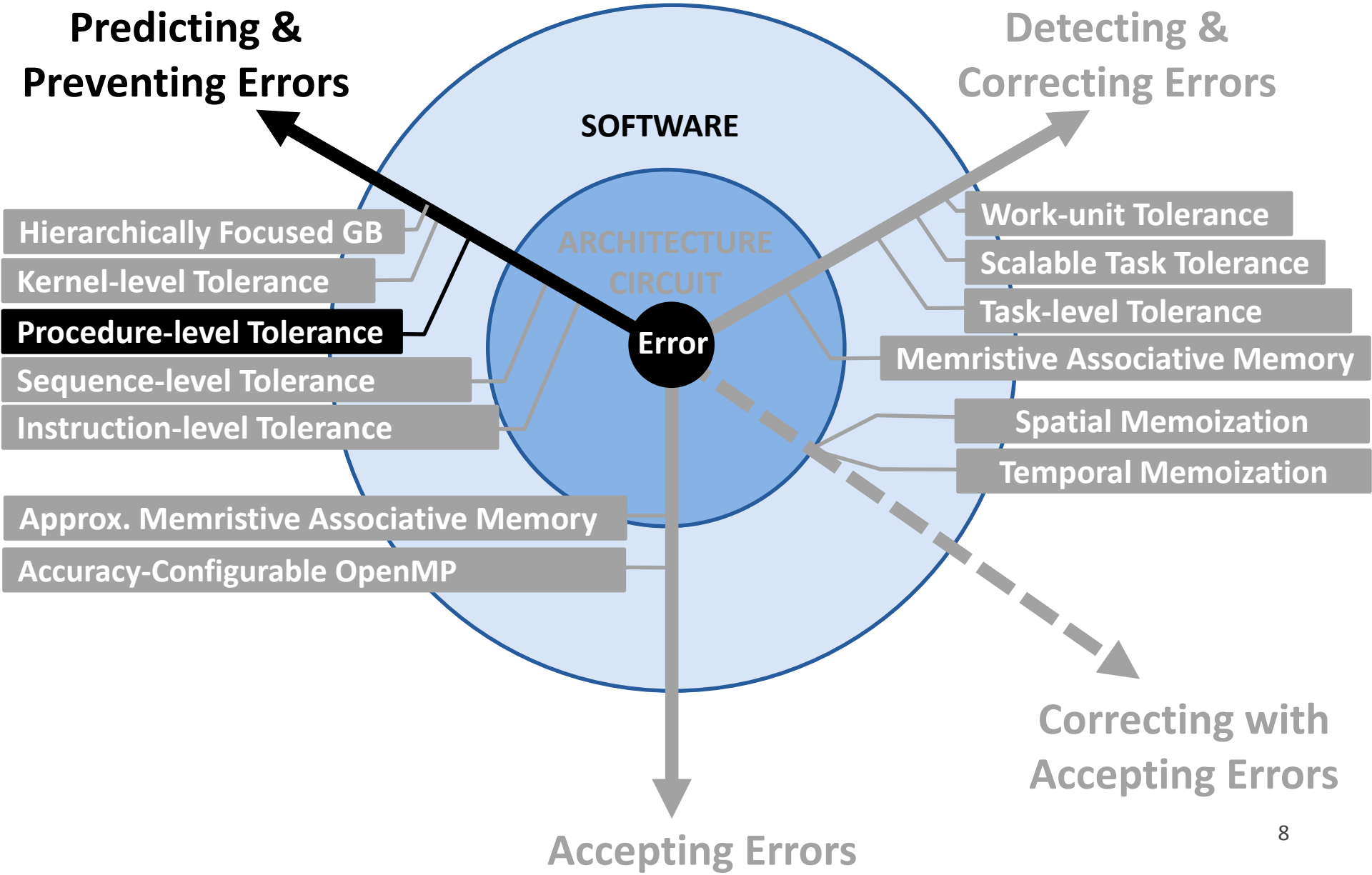
ILV(*Class III*) ≥ ILV(*Class II*) ≥ ILV(*Class I*)

SLV (*Class II*) ≥ SLV (*Class I*)

- Enables compiler optimization, e.g., loop unrolling
  - ↑ Ratio of Class I to Class II → 5% ↑ throughput

Lessen learned: **Single-core** optimization is not sufficient!

Our Approach to Variability-Tolerance



1-Core to Multi-Core: Procedure Hopping

V_{DD} = 0.81V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

☹ **Three cores cannot meet the target frequency of 830MHz.**

1-Core to Multi-Core: Procedure Hopping

V_{DD} = 0.81V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

☹ **Three cores cannot meet the target frequency of 830MHz.**

V_{DD} = 0.99V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

😐 **All cores of the same cluster meet the target frequency of 830MHz.**

1-Core to Multi-Core: Procedure Hopping

V_{DD} = 0.81V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

☹ **Three cores cannot meet the target frequency of 830MHz.**

V_{DD} = 0.99V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

😐 **All cores of the same cluster meet the target frequency of 830MHz.**

VA-V_{DD}-Hopping=(0.81V,0.99V)

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

😐 **VA-V_{DD}-hopping can accordingly tune the cores' voltage based on their delay.**

1-Core to Multi-Core: Procedure Hopping

V_{DD} = 0.81V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

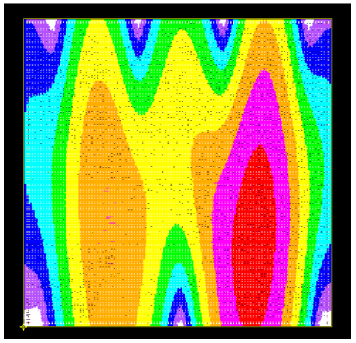
V_{DD} = 0.99V

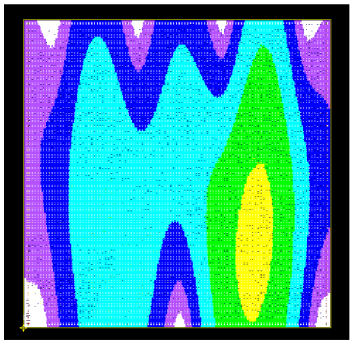
862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

VA-V_{DD}-Hopping=(0.81V,0.99V)

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

FIR IR-drop in various cores





(Vol., Temp.)	(0.99V, 125°C)	(0.81V, 125°C)
Power density	0.66 μW/μm²	0.18 μW/μm²
Max IR-drop	44 mV ✗	< 28 mV ✓

1-Core to Multi-Core: Procedure Hopping

V_{DD} = 0.81V

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

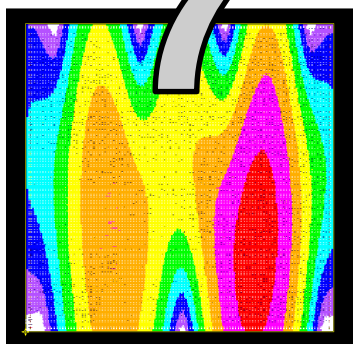
V_{DD} = 0.99V

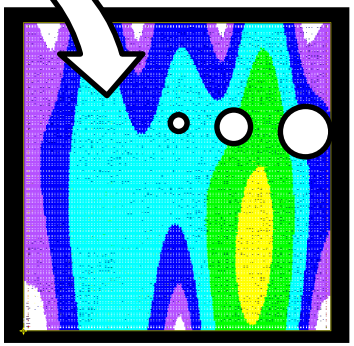
862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

VA-V_{DD}-Hopping=(0.81V,0.99V)

862 MHz	909 MHz	870 MHz	847 MHz
826 MHz	855 MHz	877 MHz	893 MHz
820 MHz	**826 MHz**	909 MHz	847 MHz
901 MHz	917 MHz	847 MHz	901 MHz

FIR IR-drop in various cores



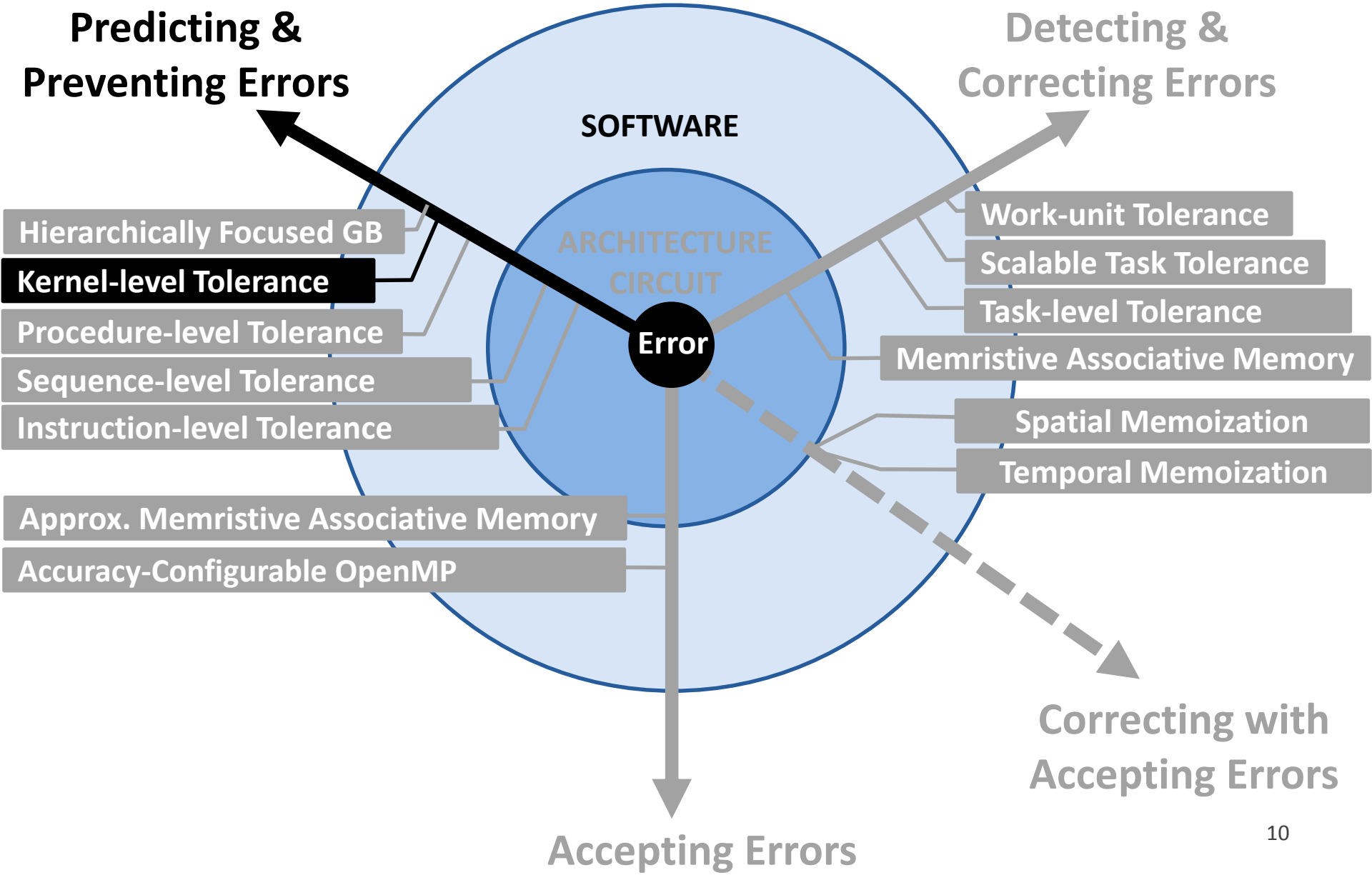


Procedure hopping ☺

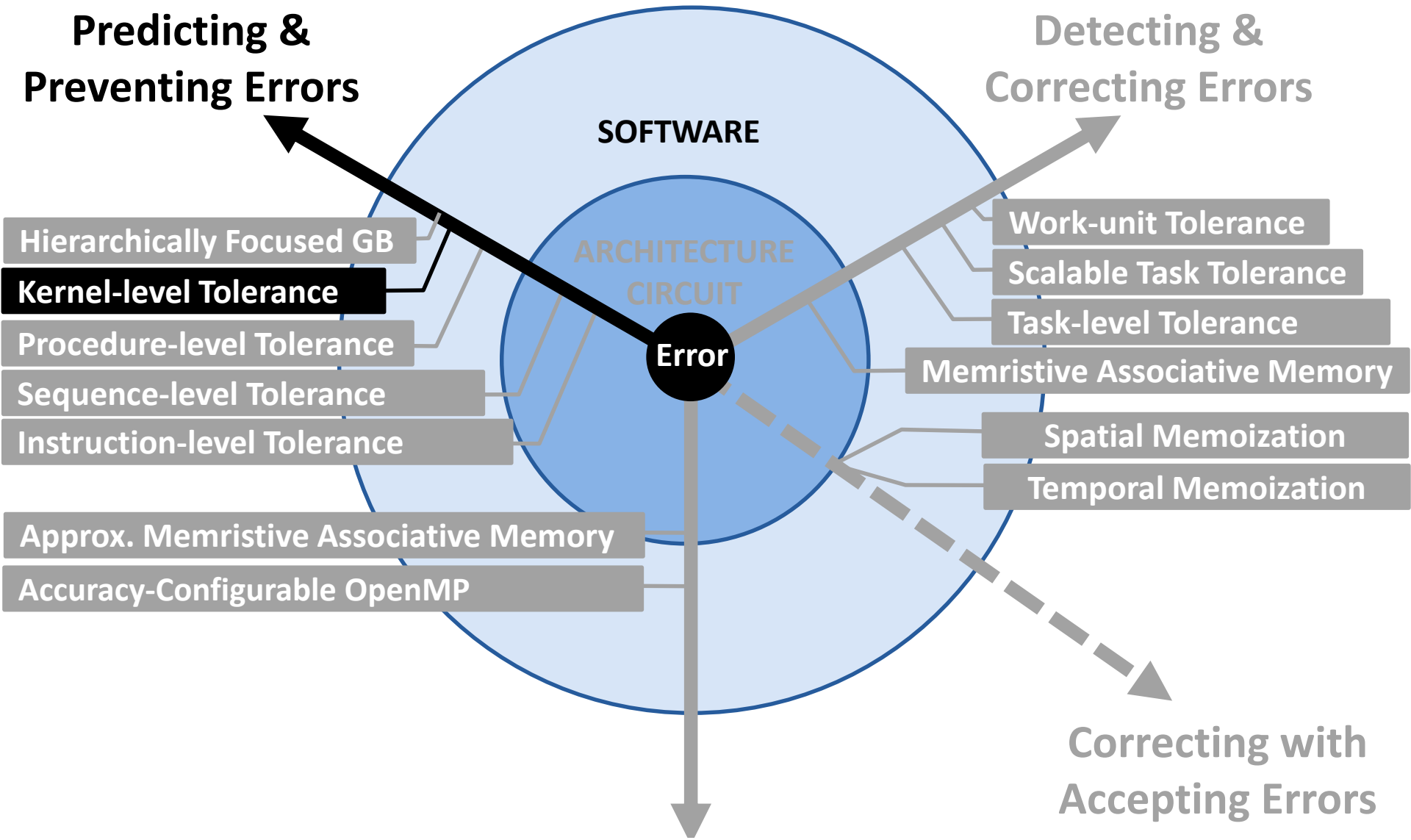
(Vol., Temp.)	(0.99V, 125°C)	(0.81V, 125°C)
Power density	0.66 μW/μm²	0.18 μW/μm²
Max IR-drop	44 mV ✗	< 28 mV ✓

Programming **model** and **RT** awareness → OpenMP!

Our Approach to Variability-Tolerance

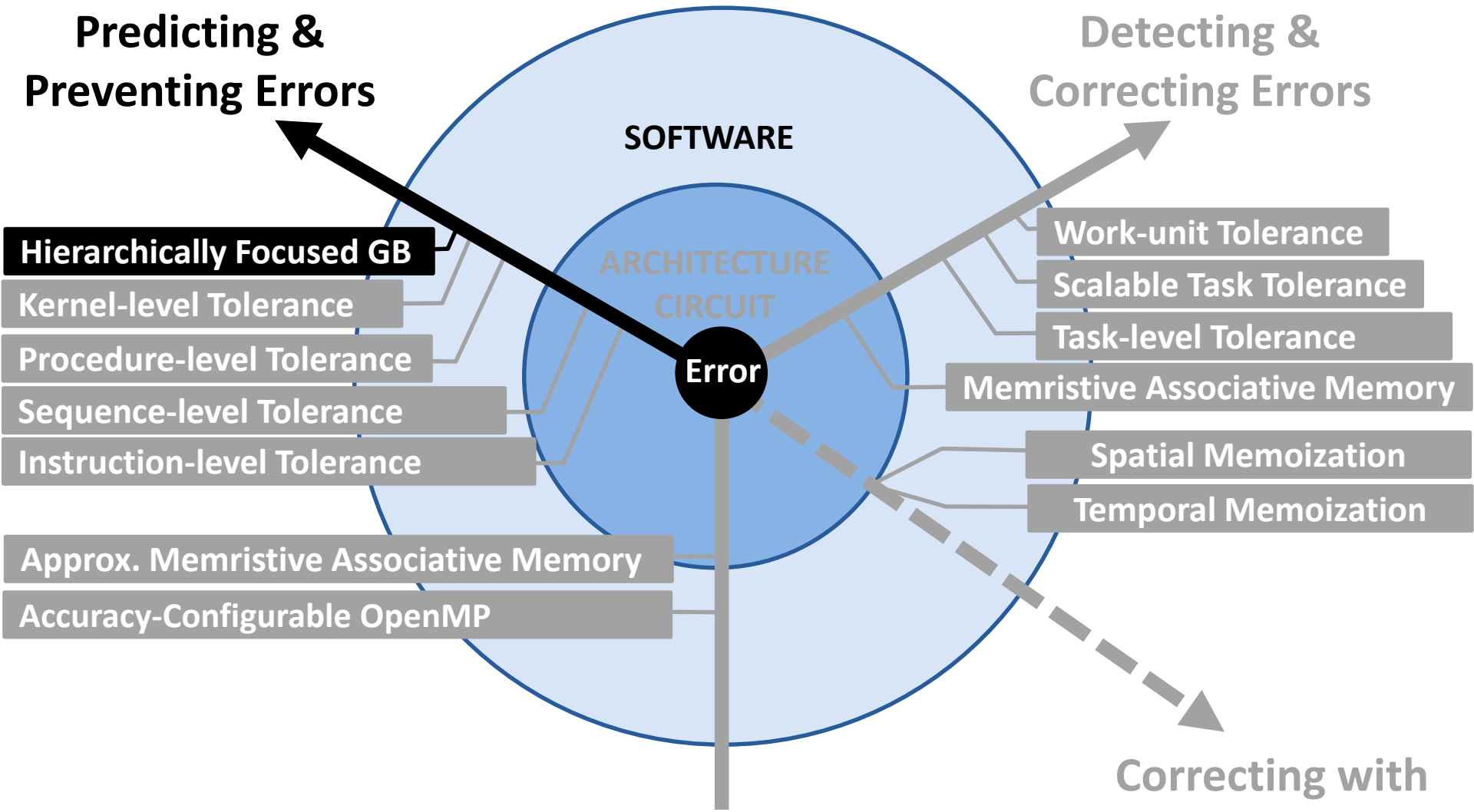


Our Approach to Variability-Tolerance



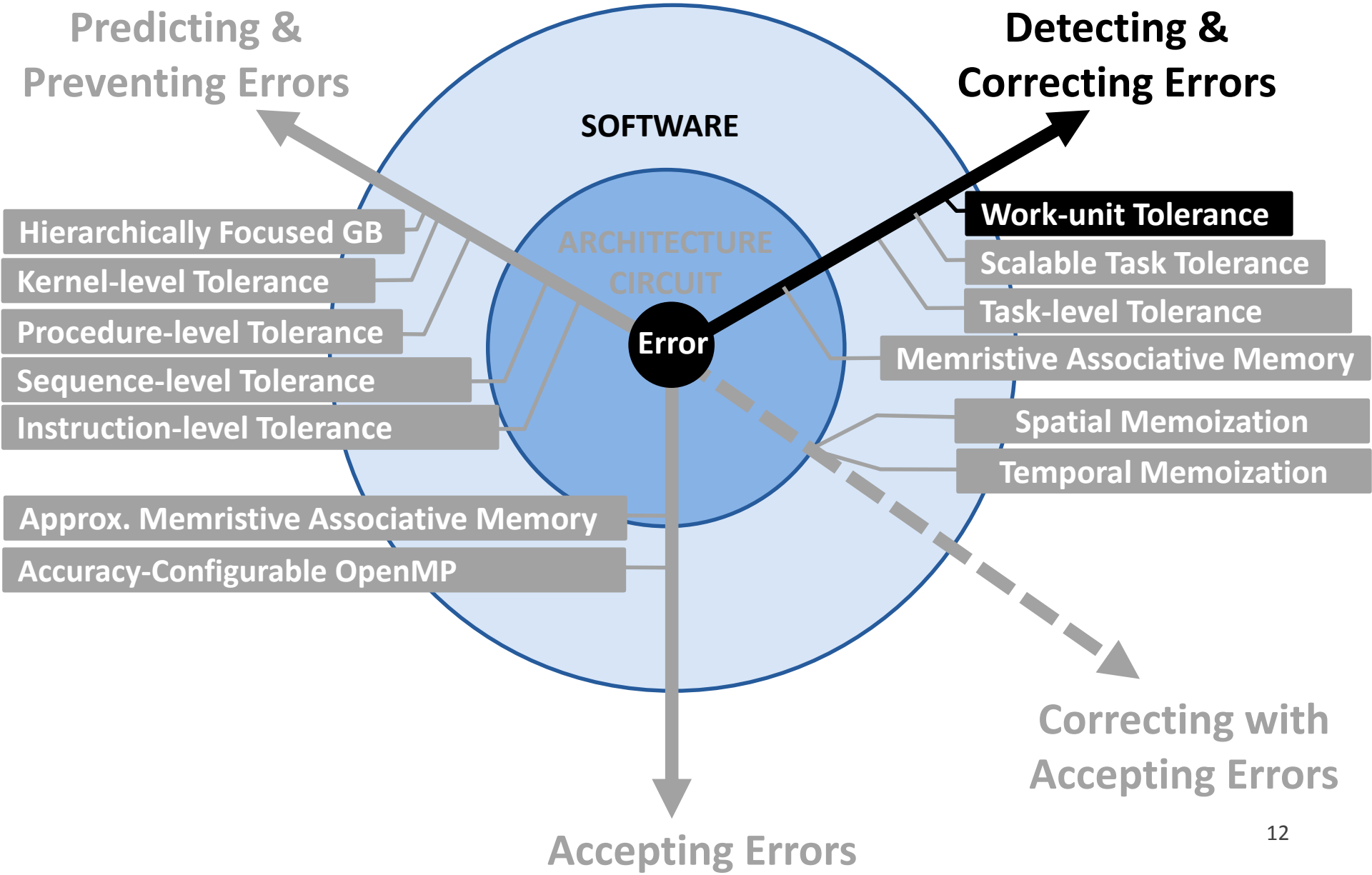
Execution of ***healthy*** kernels reduces aging-induced ΔV_{th} up to 49%.

Our Approach to Variability-Tolerance



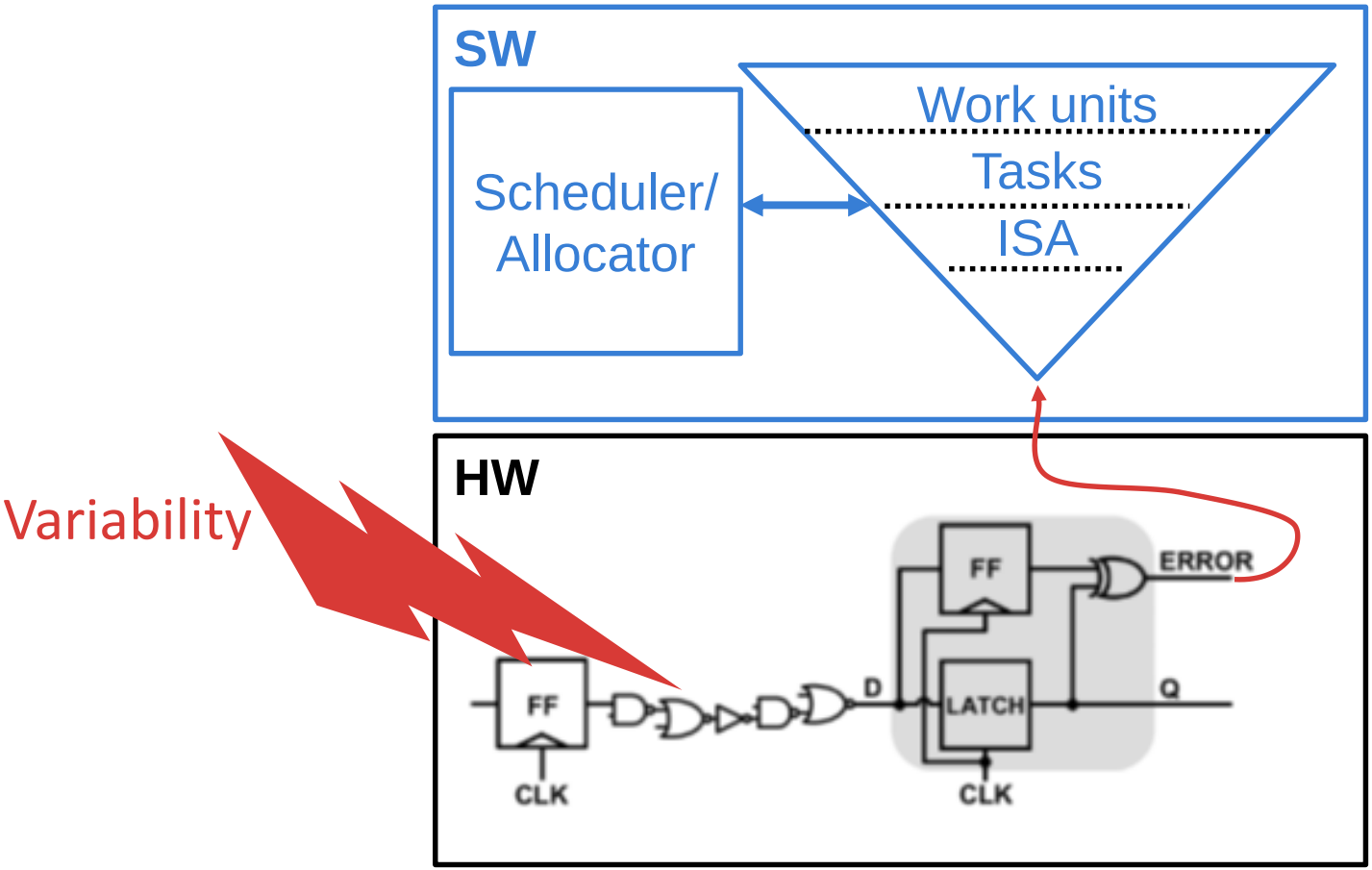
Using supervised learning models to derive guardband reduction (1.8×–2.1×) in view of PVTA monitors.

Our Approach to Variability-Tolerance

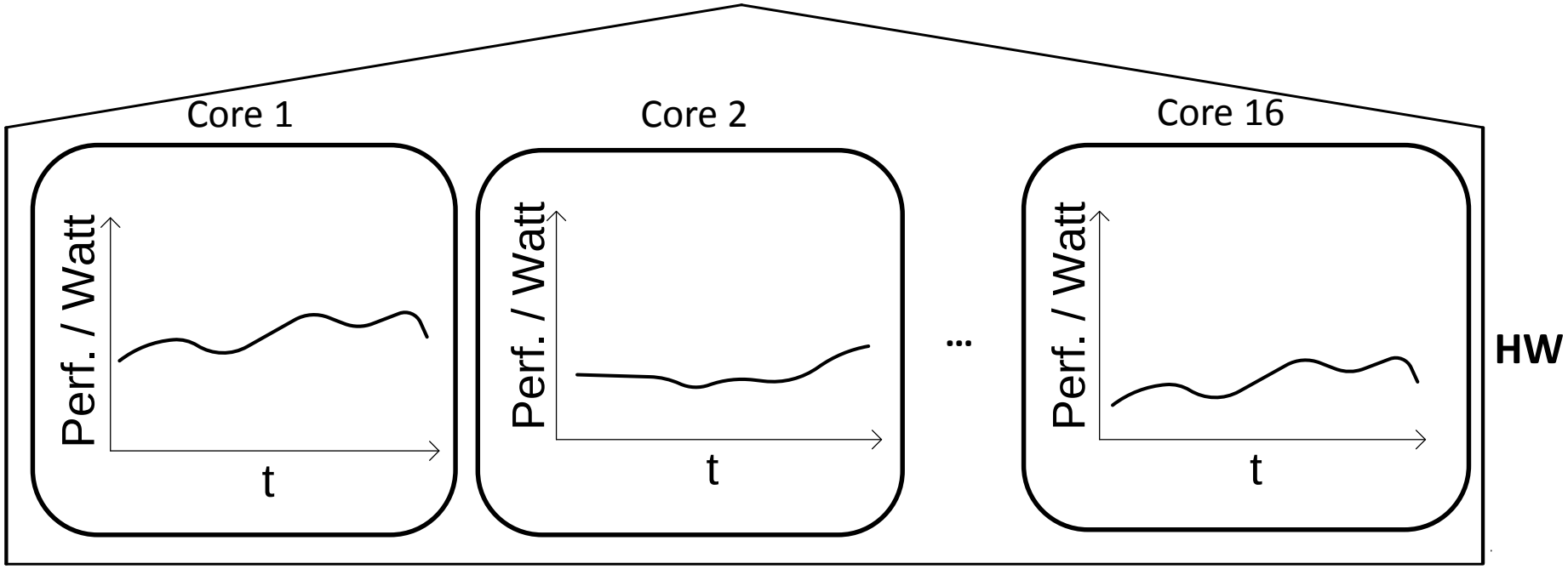


Work-Unit Tolerance

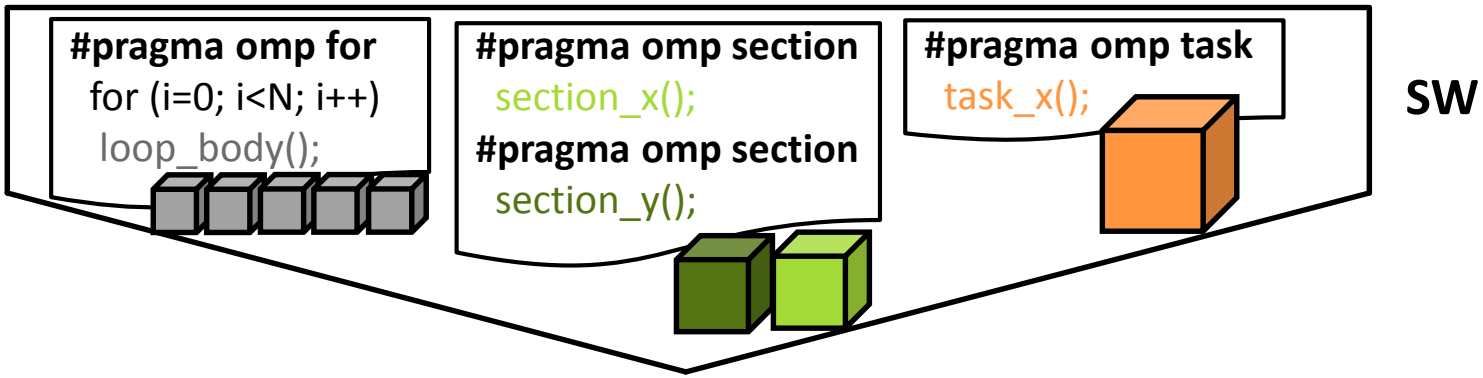
Timing Error Abstraction

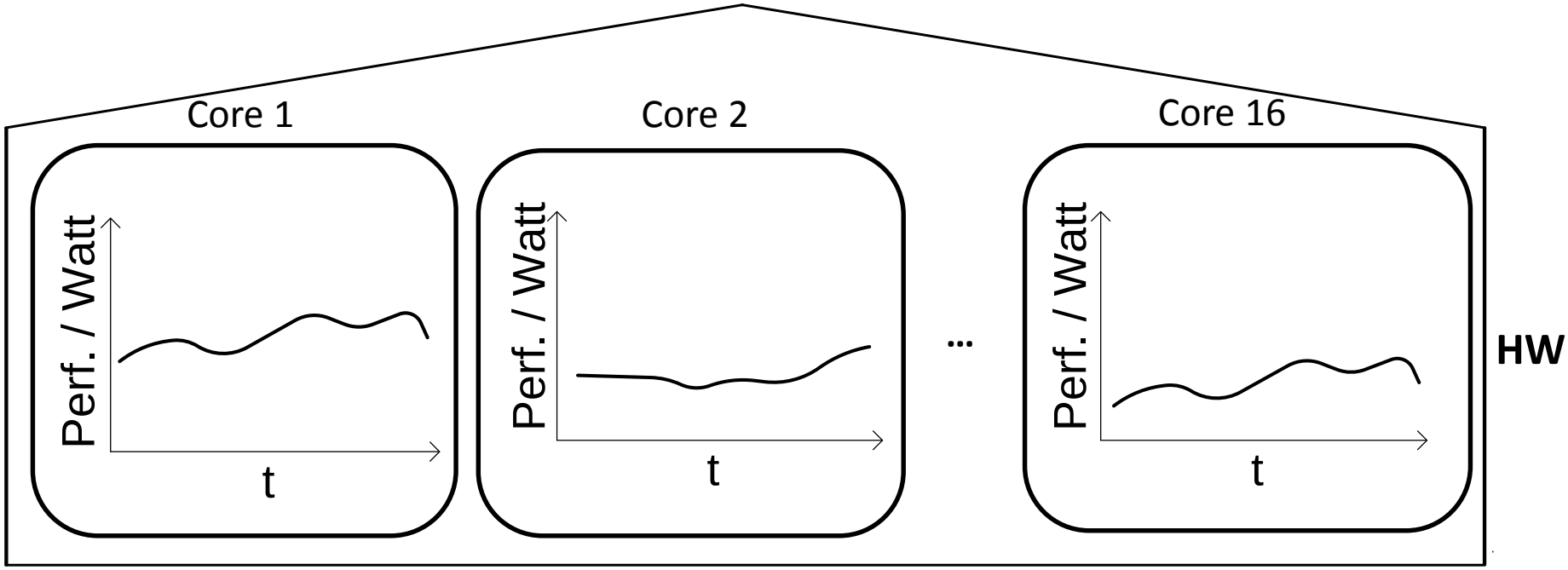


Exposing Errors to OpenMP

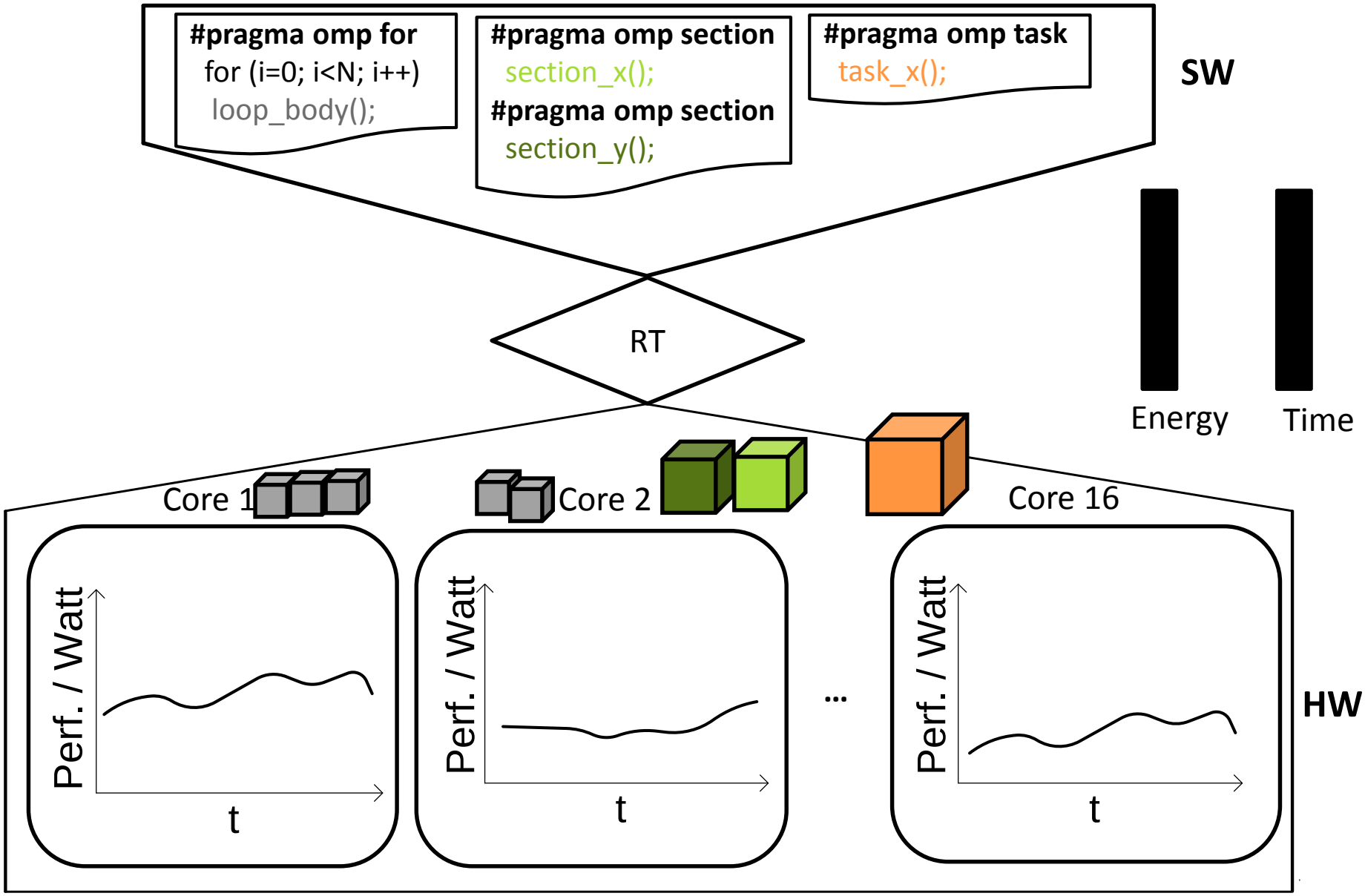


Exposing Errors to OpenMP

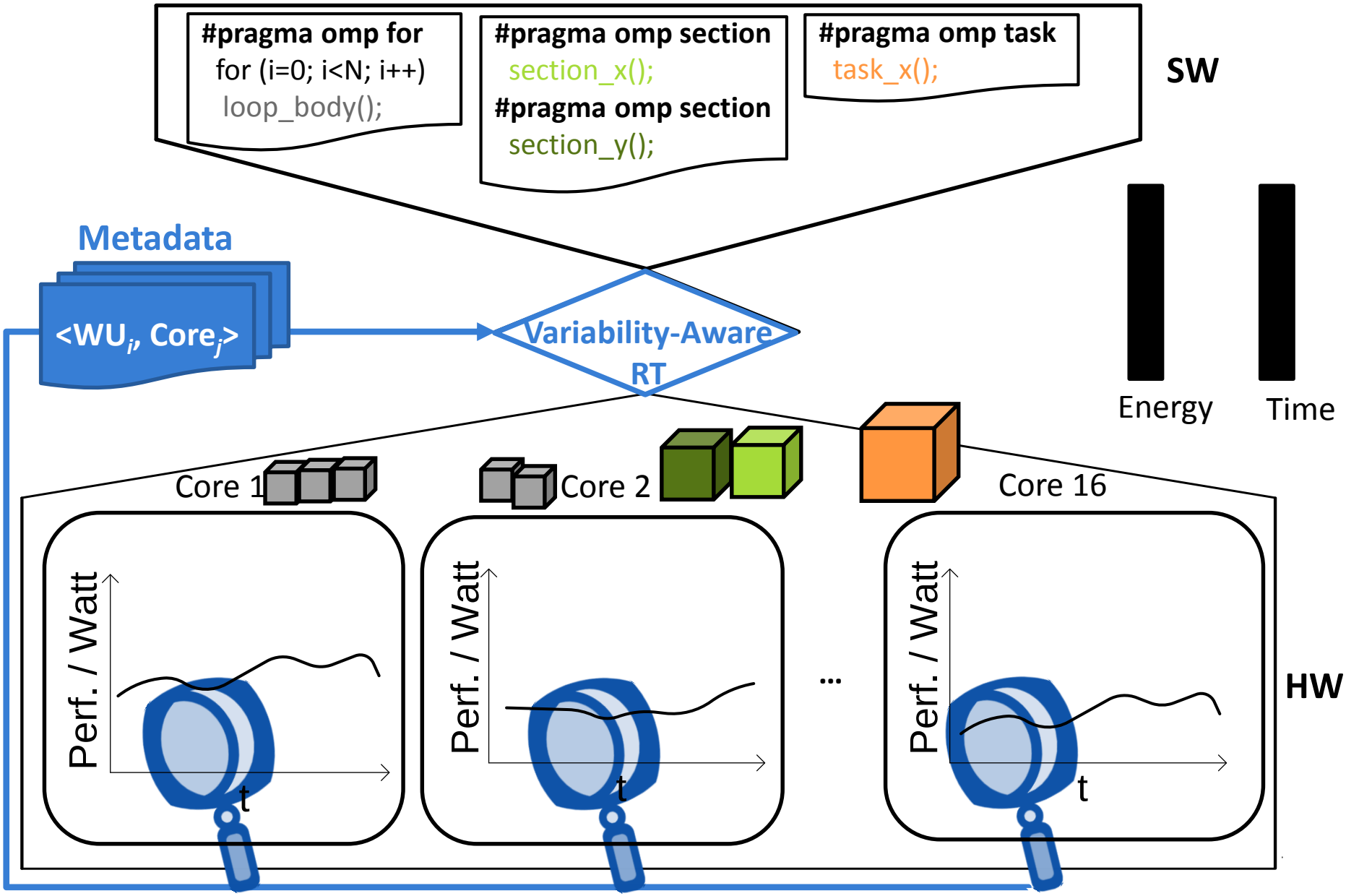




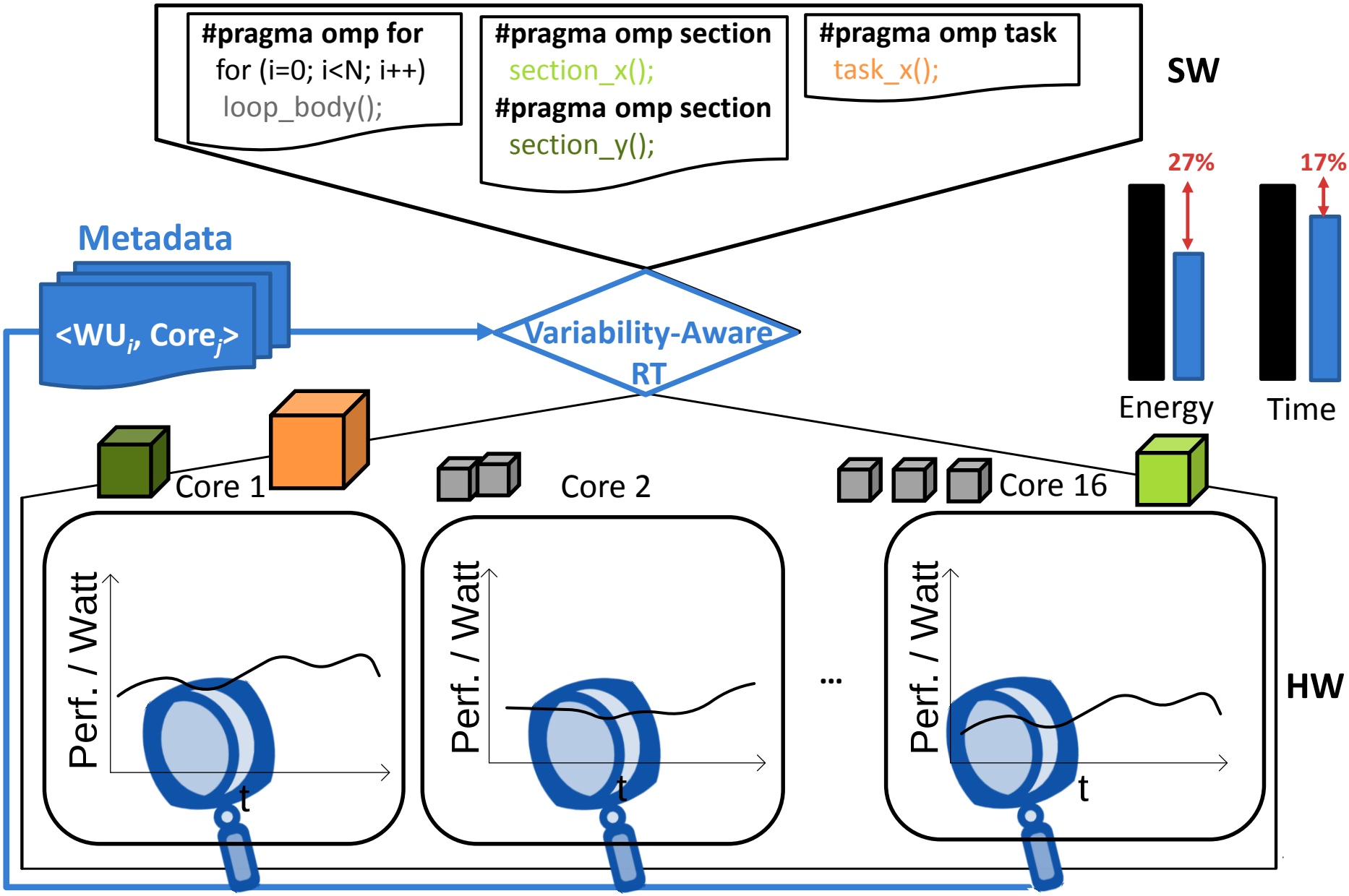
Exposing Errors to OpenMP



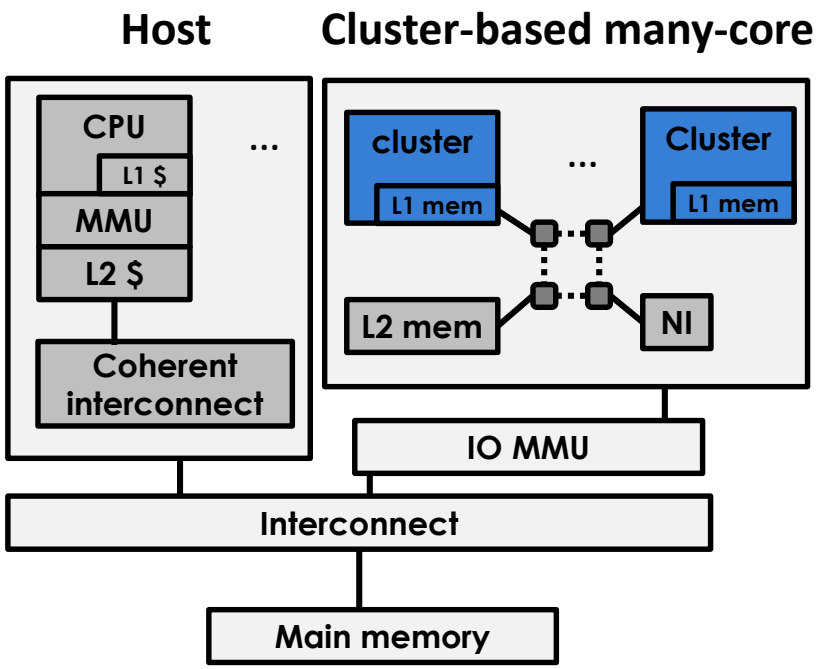
Exposing Errors to OpenMP



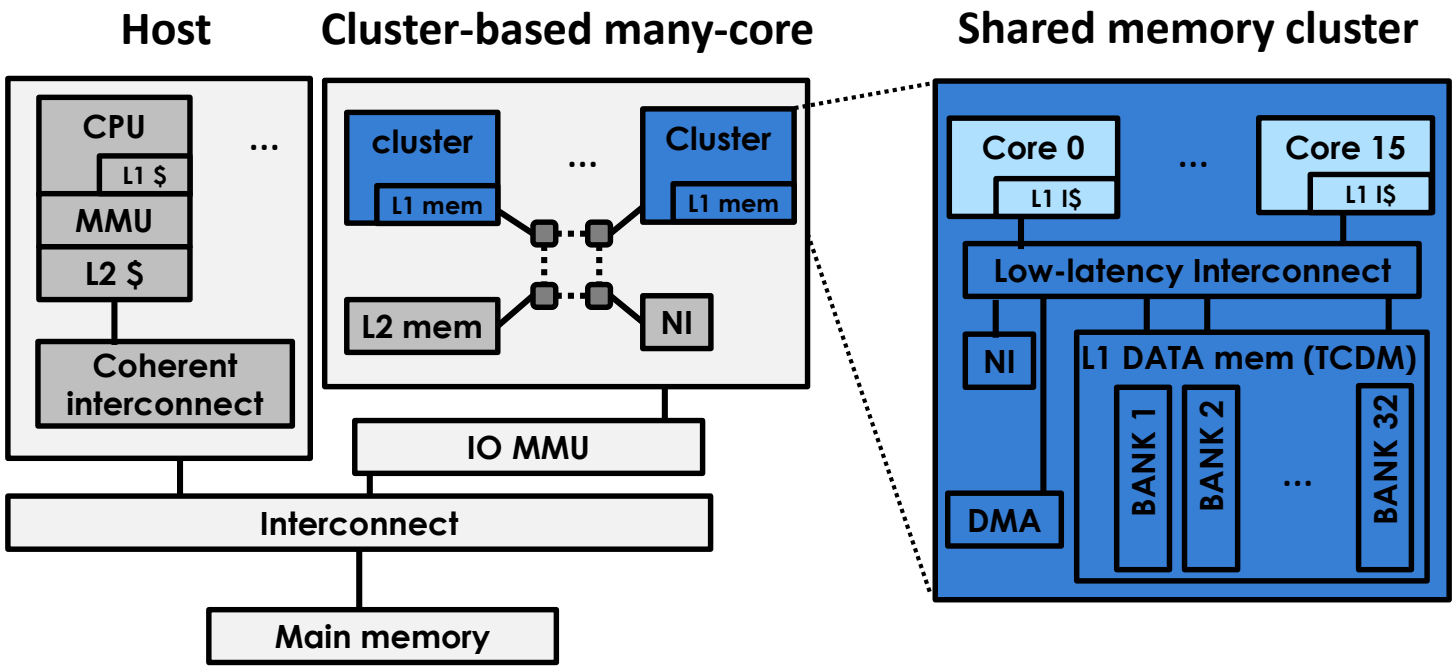
Exposing Errors to OpenMP



Cluster-Based Many-Core Architecture



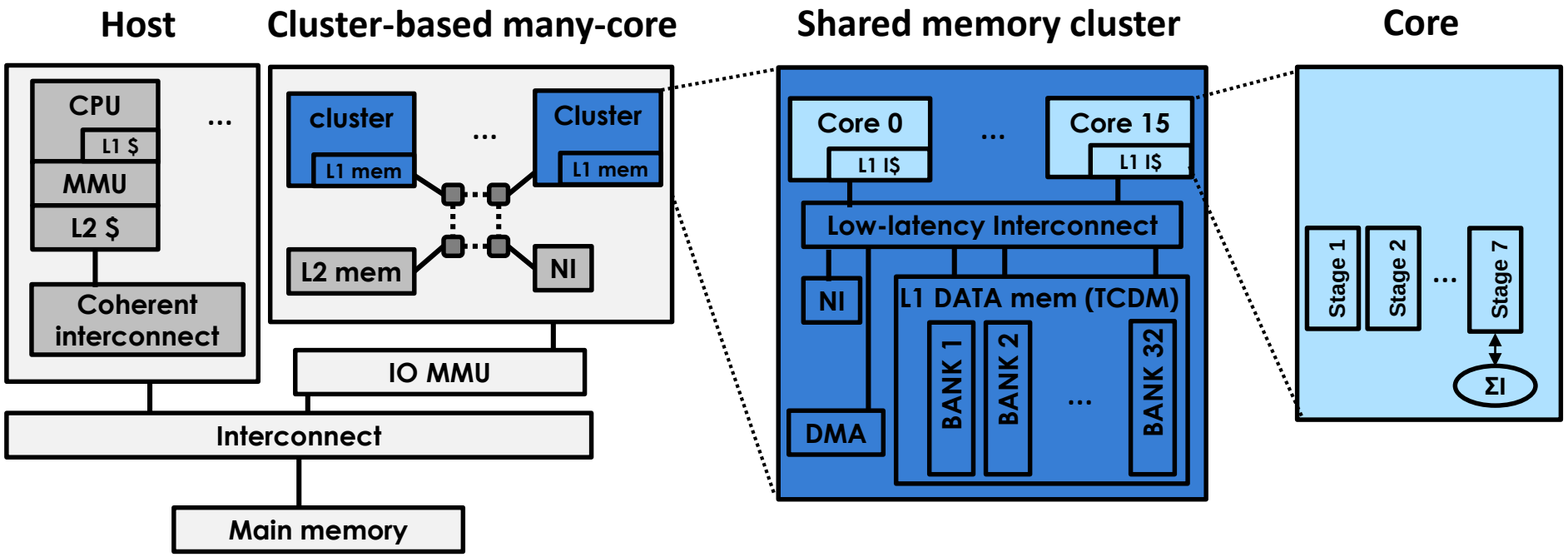
Cluster-Based Many-Core Architecture



Shared memory cluster:

- **16 32-bit in-order RISC processors**
- **Shared L1 tightly-coupled data memory (TCDM)**

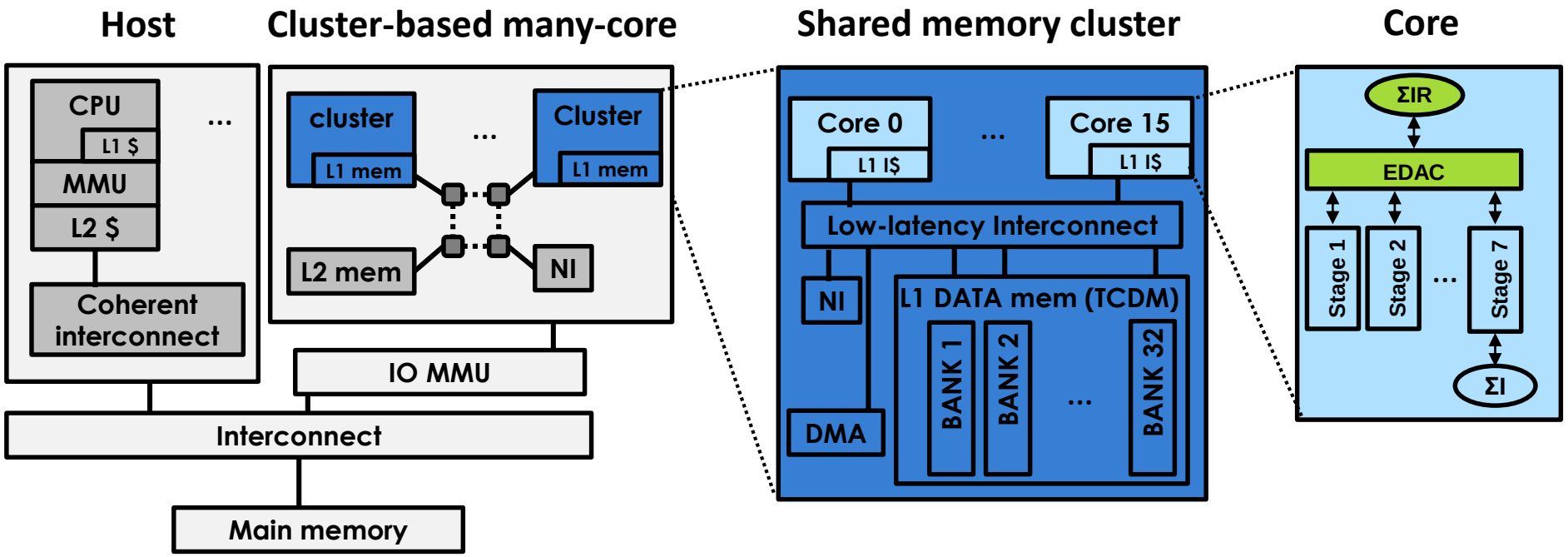
Cluster-Based Many-Core Architecture



Shared memory cluster:

- **16 32-bit in-order RISC processors**
- **Shared L1 tightly-coupled data memory (TCDM)**

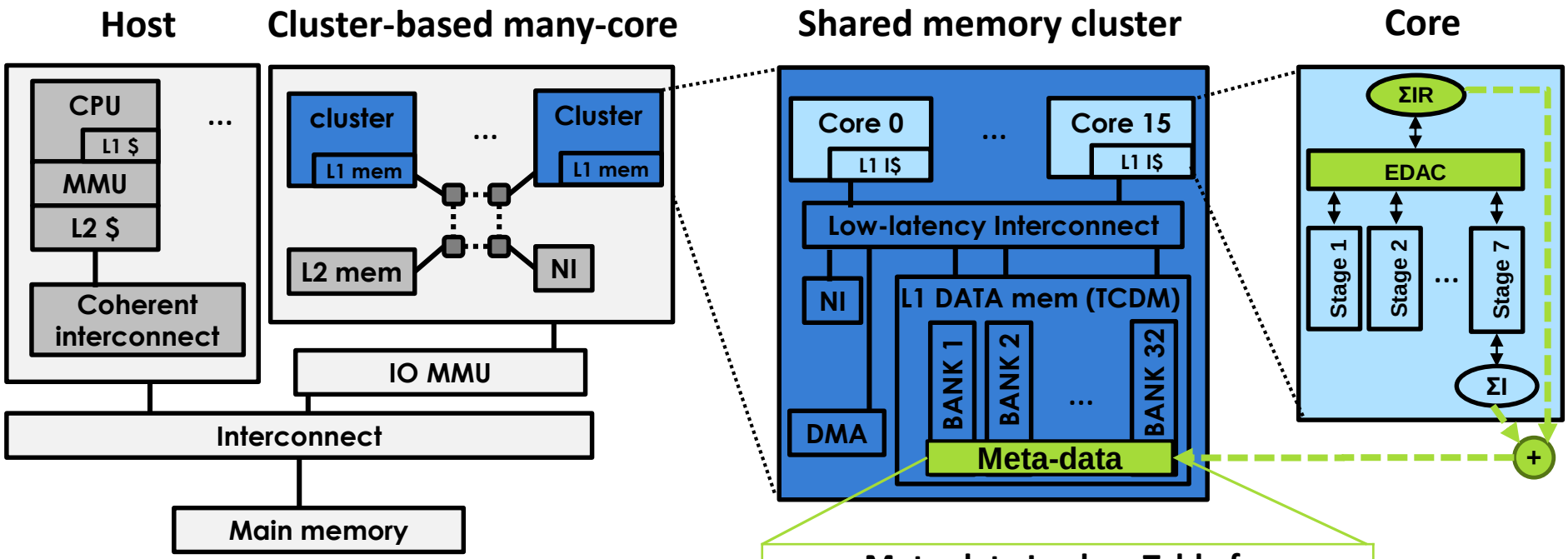
Cluster-Based Many-Core Architecture



Shared memory cluster:

- **16 32-bit in-order RISC processors**
- **Shared L1 tightly-coupled data memory (TCDM)**
- **Each core uses EDAC with multiple-issue instruction replay**

Cluster-Based Many-Core Architecture

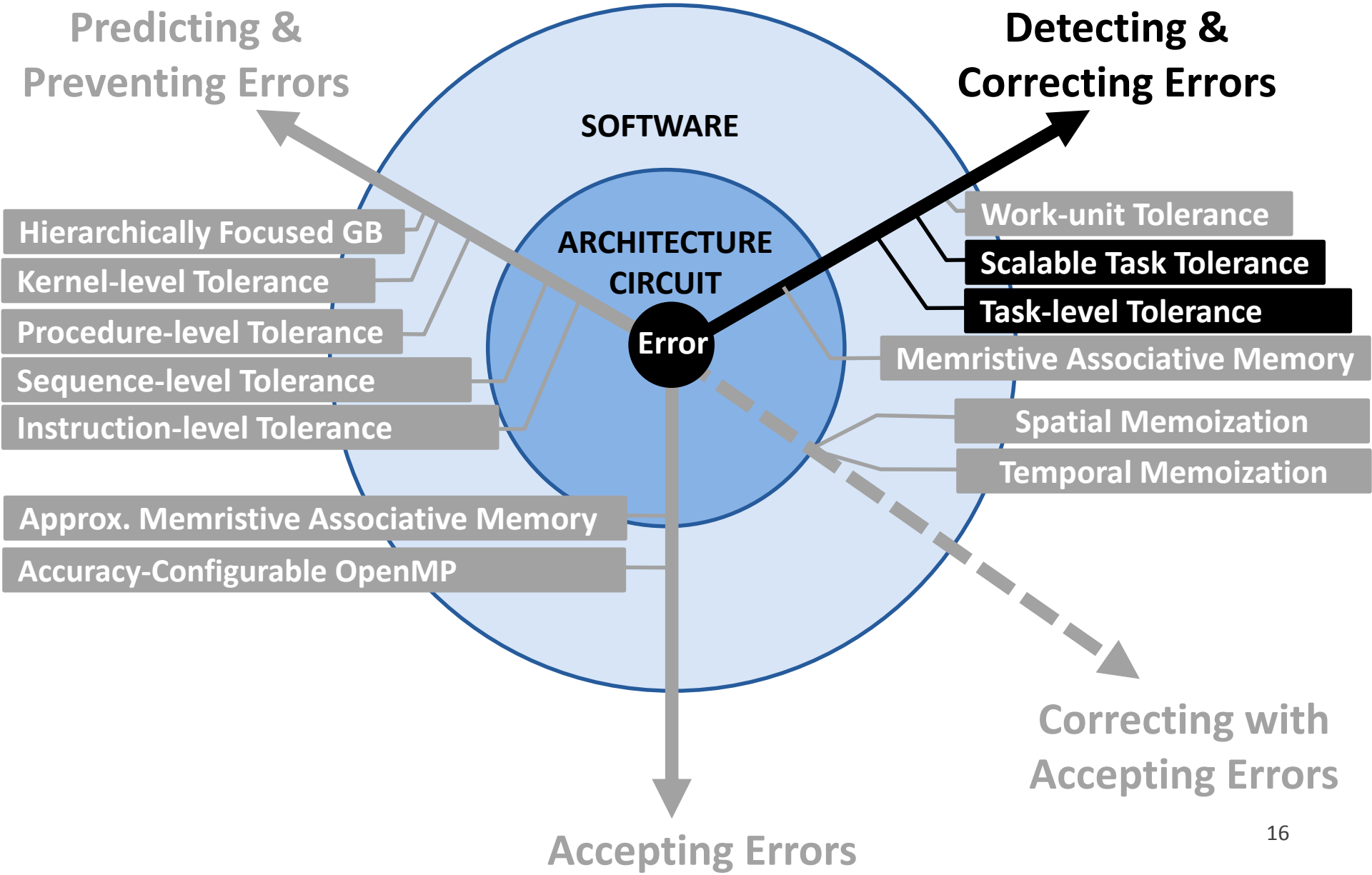


Meta-data Lookup Table for Application$_X$		
Core 1	*Core 2*	...
Work-unit *type1*	Work-unit *type1*	
Work-unit *type2*	Work-unit *type2*	
Work-unit *type3*	Work-unit *type3*	
Work-unit *type4*	Work-unit *type4*	

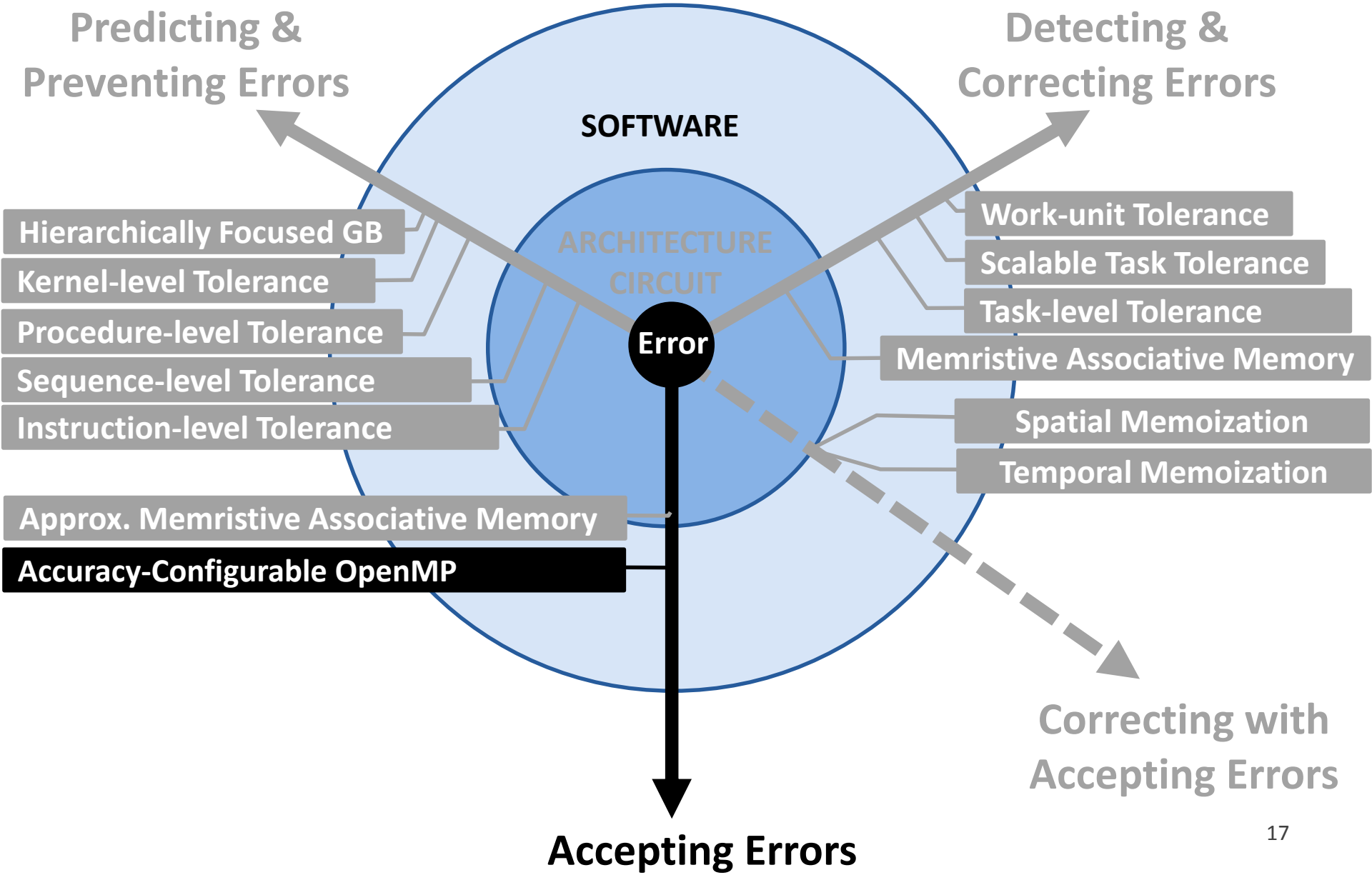
Shared memory cluster:

- **16 32-bit in-order RISC processors**
- **Shared L1 tightly-coupled data memory (TCDM)**
- **Each core uses EDAC with multiple-issue instruction replay**

Our Approach to Variability-Tolerance



Our Approach to Variability-Tolerance



Accepting Errors (Approximate Computing)

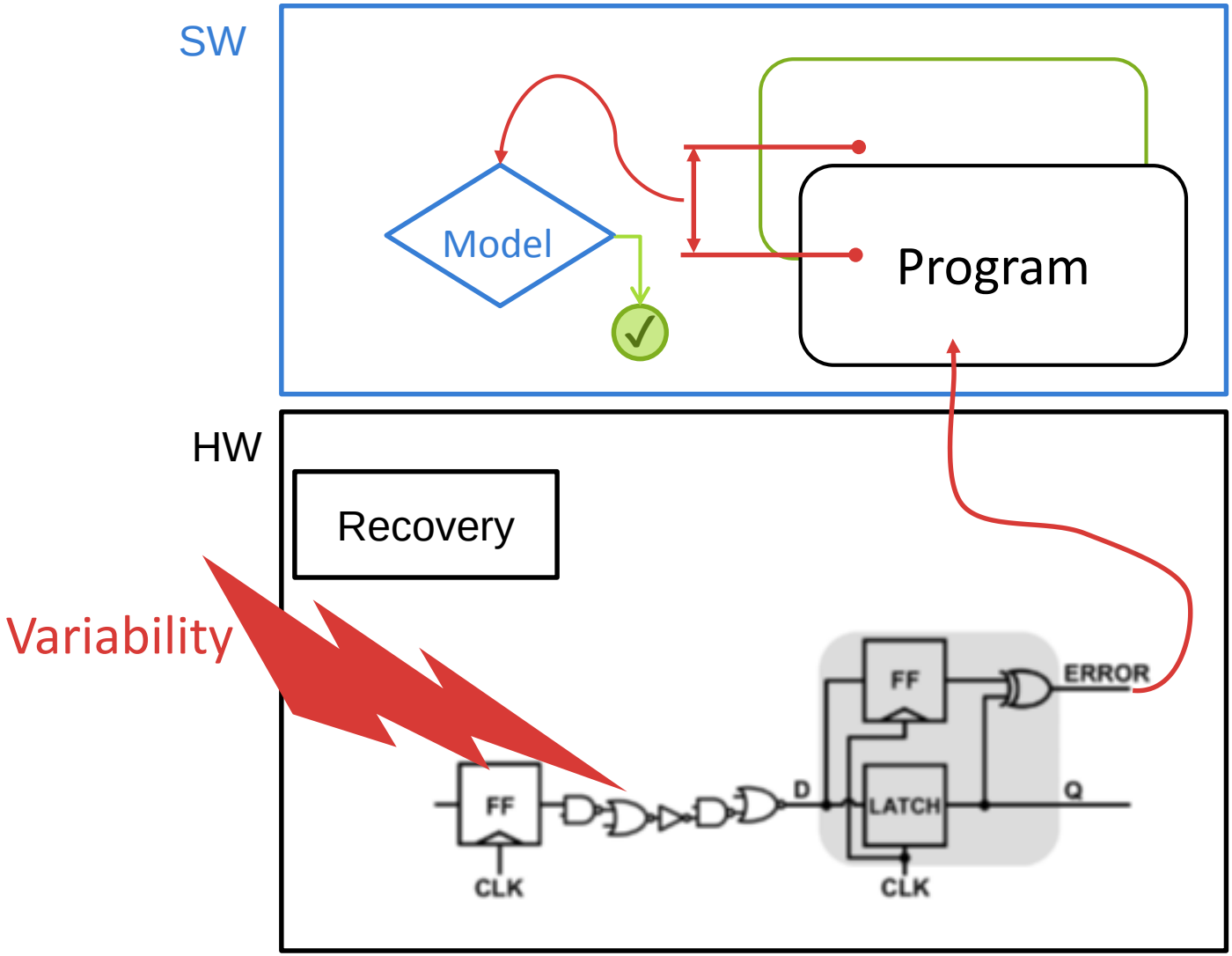
1. What kind of programs?
   - Multimedia, web search, cyber-physical, data analytics, machine learning, robotics, probabilistic
2. How to isolate exact and approximate segments?
   - Annotation and directives
3. How to run each segment?
   - Accuracy-reconfigurable units

Accepting Errors (Approximate Computing)

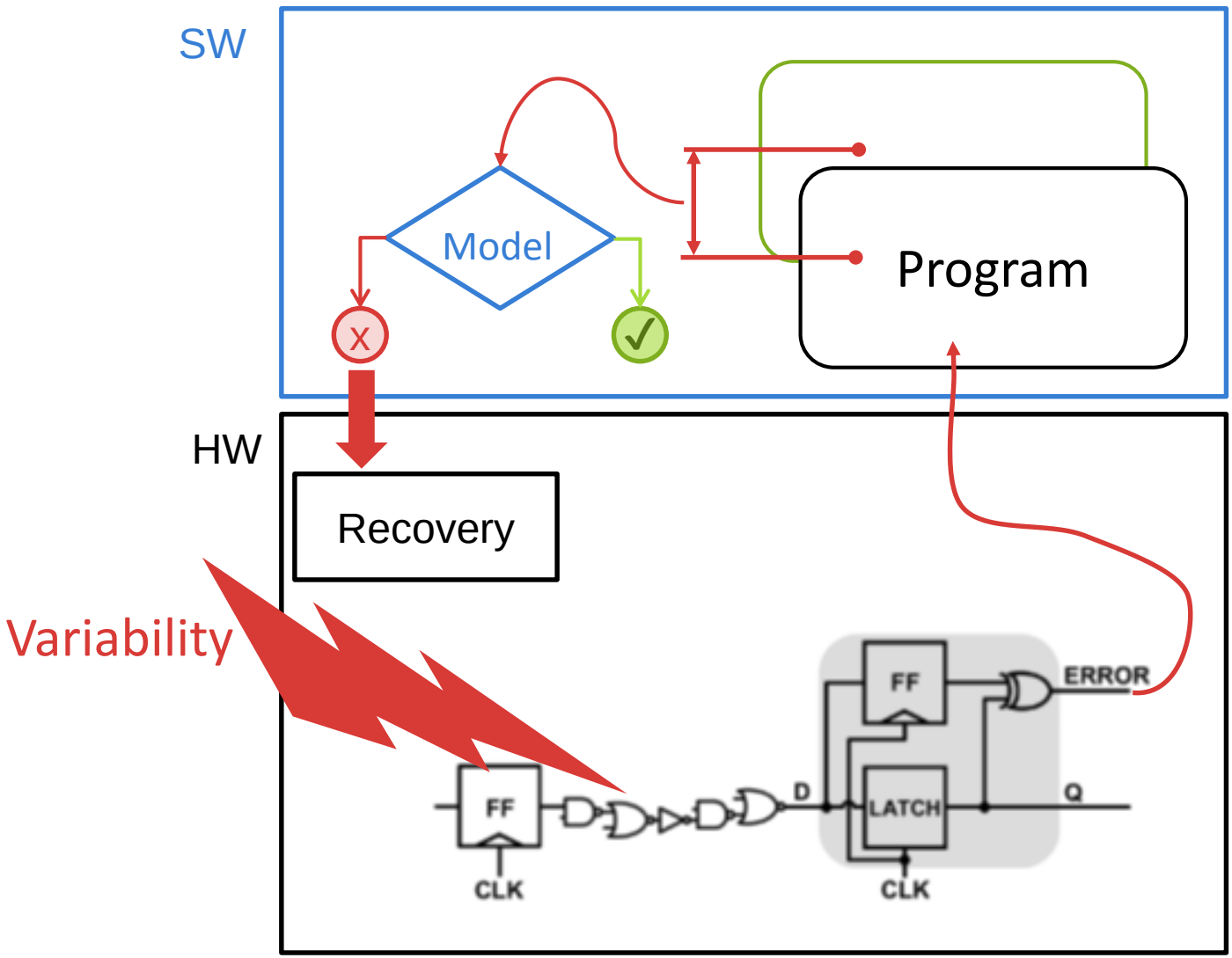
1. What kind of programs?
   - Multimedia, web search, cyber-physical, data analytics, machine learning, robotics, probabilistic
2. How to isolate exact and approximate segments?
   - Annotation and directives
3. How to run each segment?
   - Accuracy-reconfigurable units

Orchestrating Compiler, Runtime, Architecture, and Circuit

Accepting Errors in OpenMP Program



Accepting Errors in OpenMP Program



Accuracy-Configurable OpenMP

```
#pragma omp parallel {

 ....
 #pragma omp for
   for (i=-K; i<=K; ++i)
    for (j=-K; j<=K; ++j) {
     float data = in[i+n][j+m];
     float coef = coeffs[n+K][m+K];


      float result = data * coef;
      sum += result;

    }
   out[n][m] = sum/scale;

 ....
 }
```

Accuracy-Configurable OpenMP

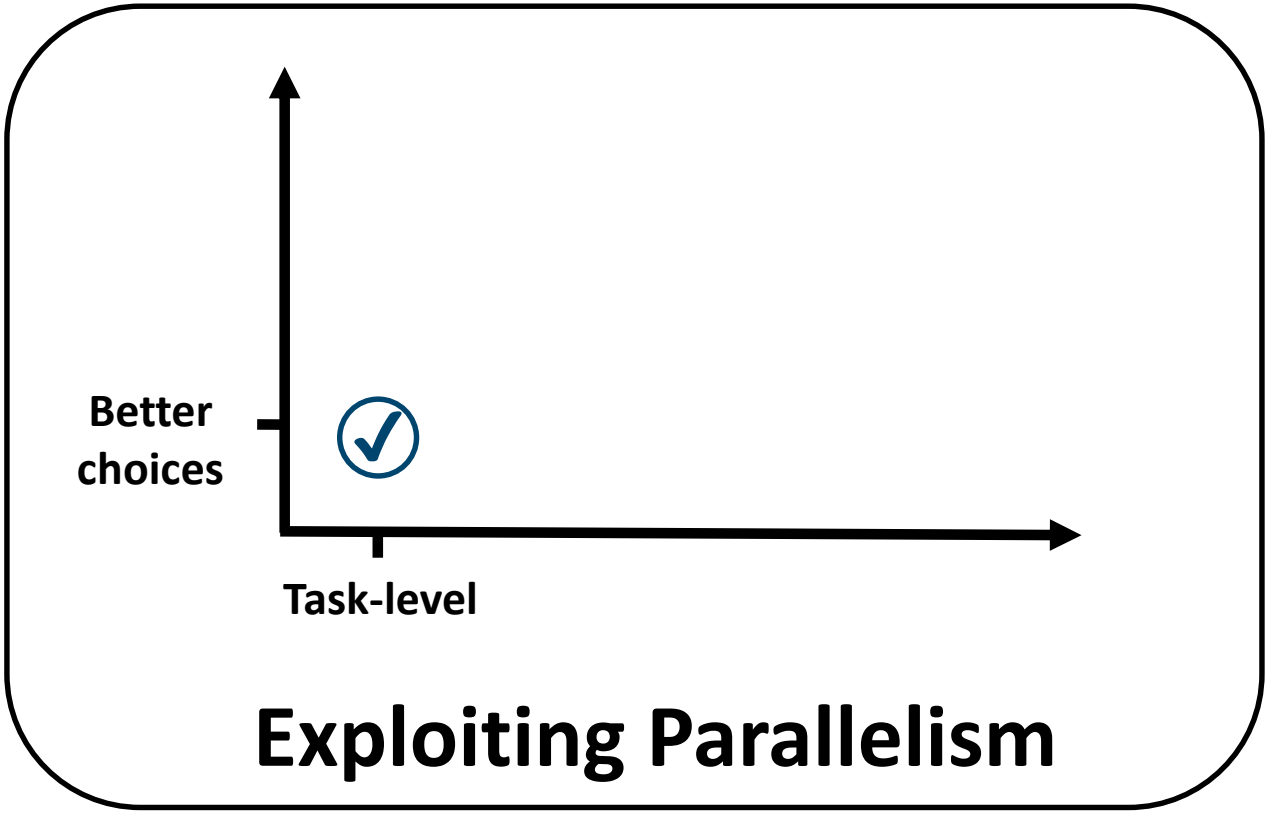
```
#pragma omp parallel {
 #pragma omp accurate {
 ....
 #pragma omp for
   for (i=-K; i<=K; ++i)
    for (j=-K; j<=K; ++j) {
     float data = in[i+n][j+m];
     float coef = coeffs[n+K][m+K];


      float result = data * coef;
      sum += result;

    }
   out[n][m] = sum/scale;
  }
 ....
 }
```

Accuracy-Configurable OpenMP

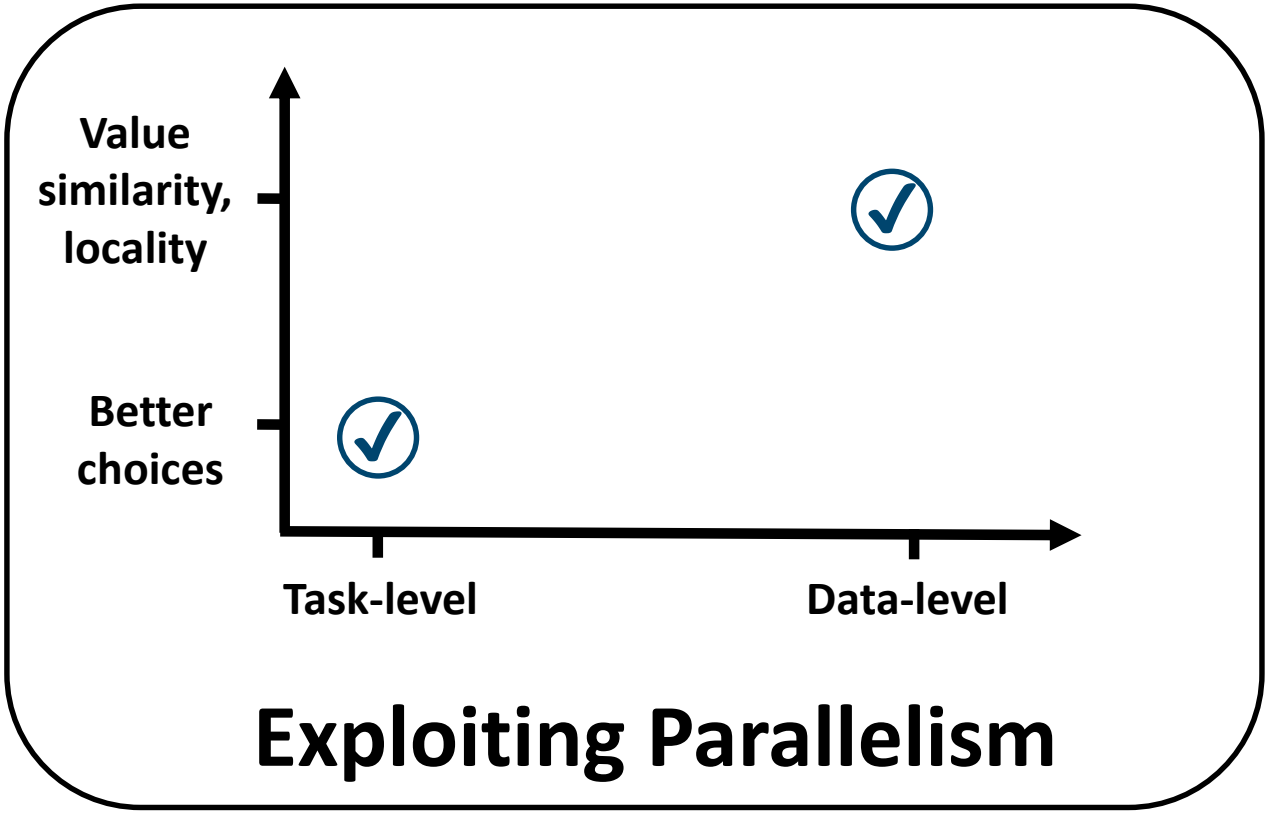
```
#pragma omp parallel {
 #pragma omp accurate {
 ....
 #pragma omp for
   for (i=-K; i<=K; ++i)
    for (j=-K; j<=K; ++j) {
     float data = in[i+n][j+m];
     float coef = coeffs[n+K][m+K];
     #pragma omp approximate error_significance (12)
     {
      float result = data * coef;
      sum += result;
     }
    }
   out[n][m] = sum/scale;
 }
```

36% energy saving compared to unannotated code

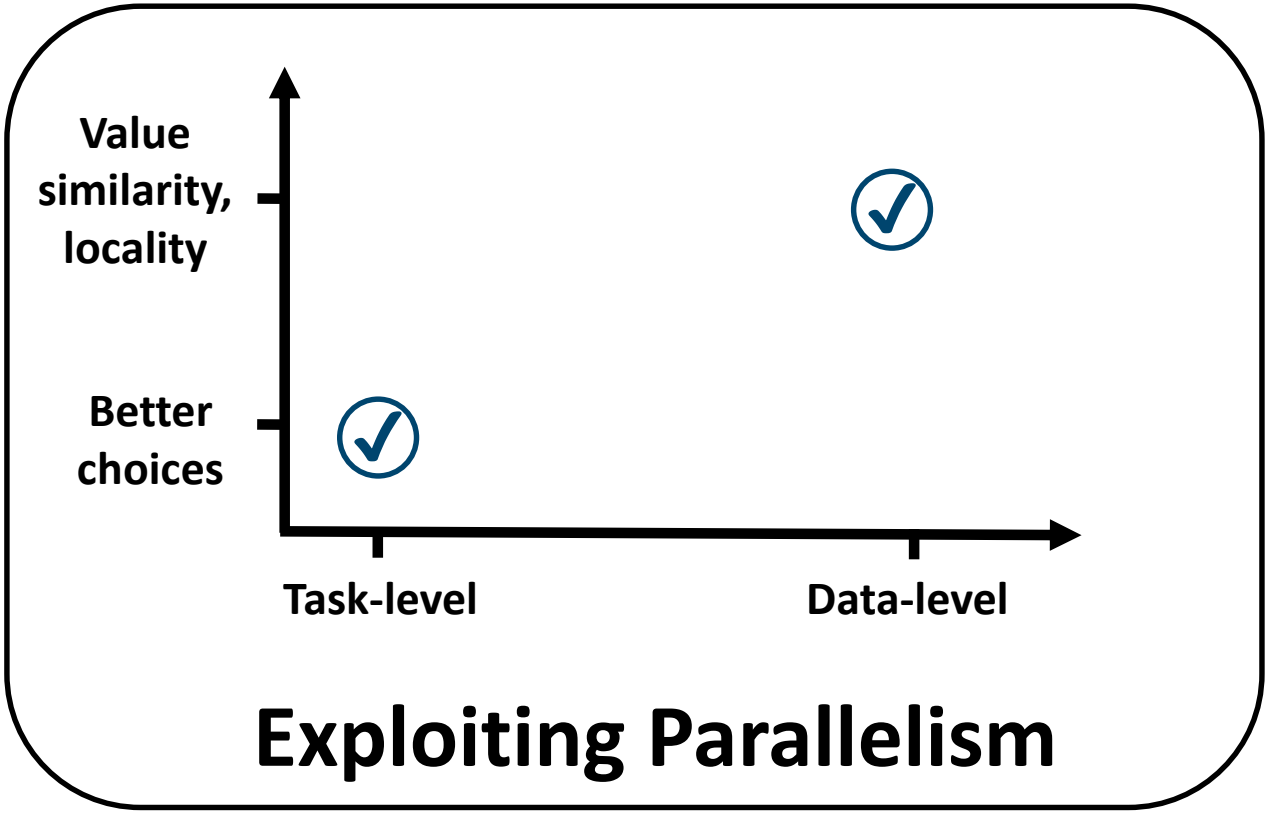
Variability-Tolerance



Variability-Tolerance

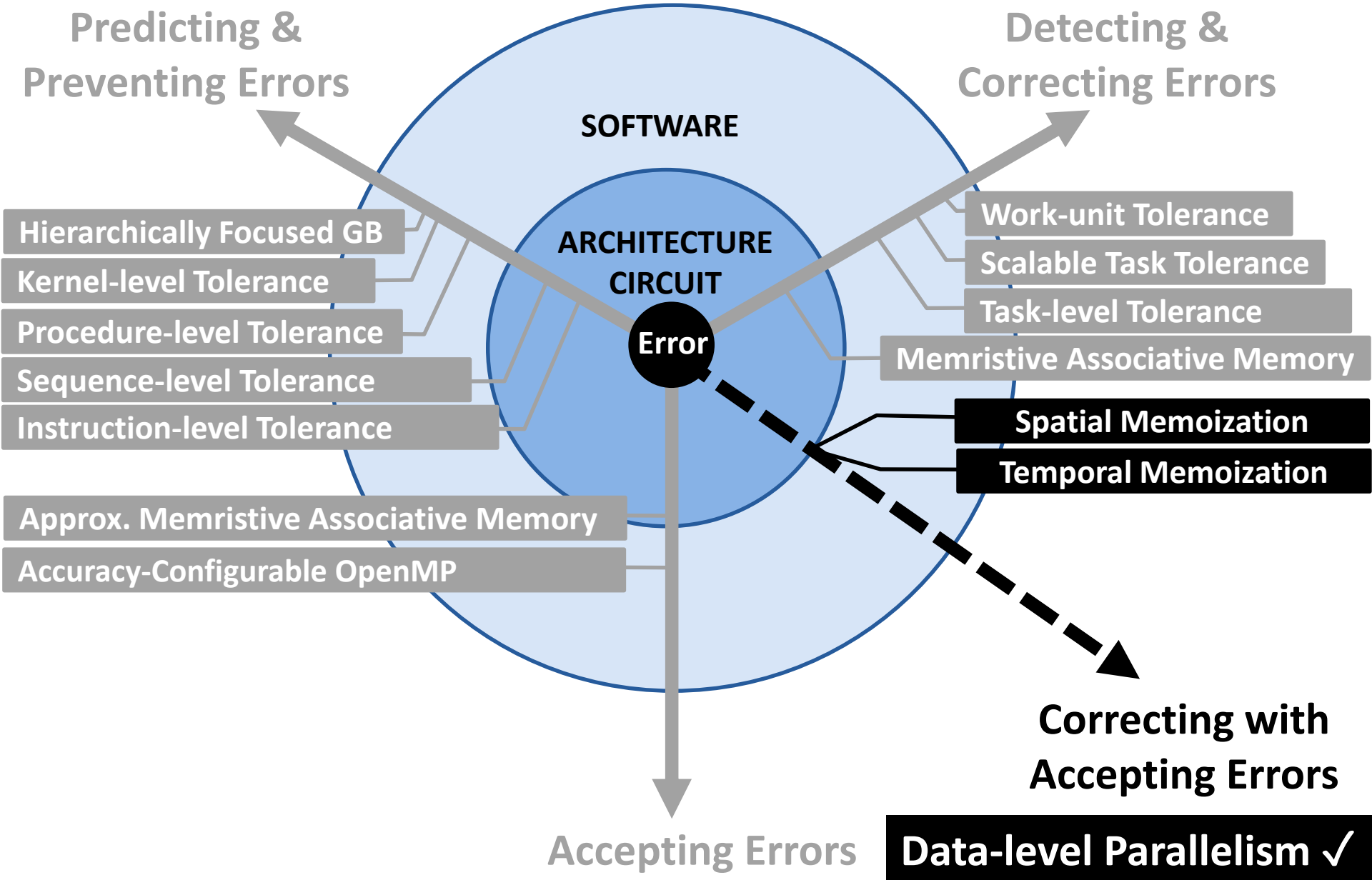


Variability-Tolerance



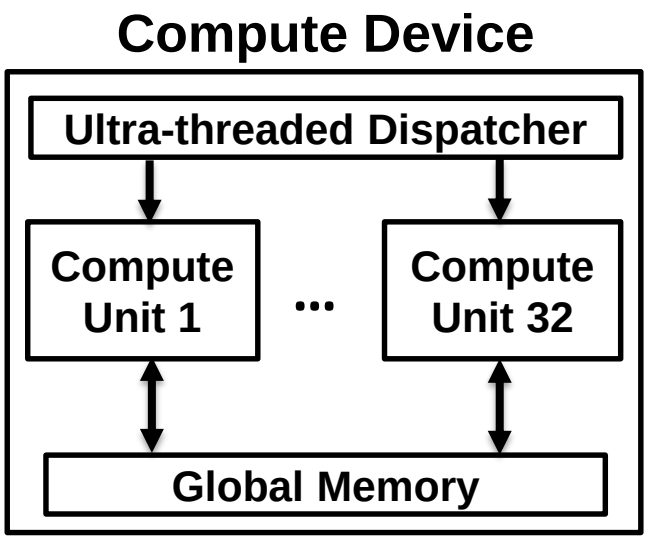
Enabling Approximate Computing

Our Approach to Variability-Tolerance



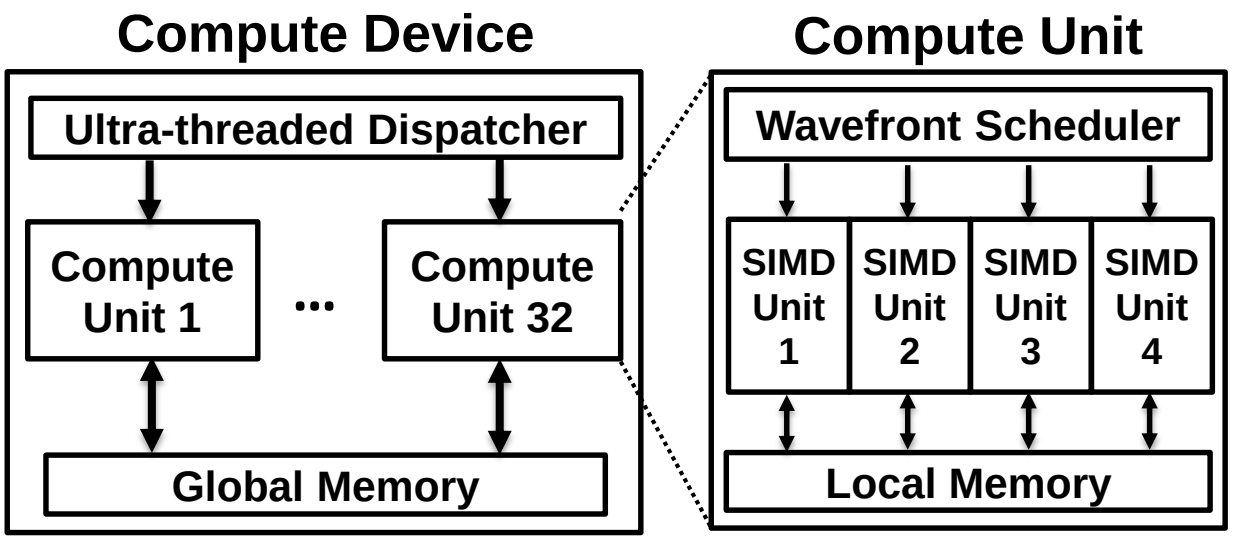
AMD GPU Architecture

- Radeon HD 7970 device from Southern Islands family
  - 32 compute units



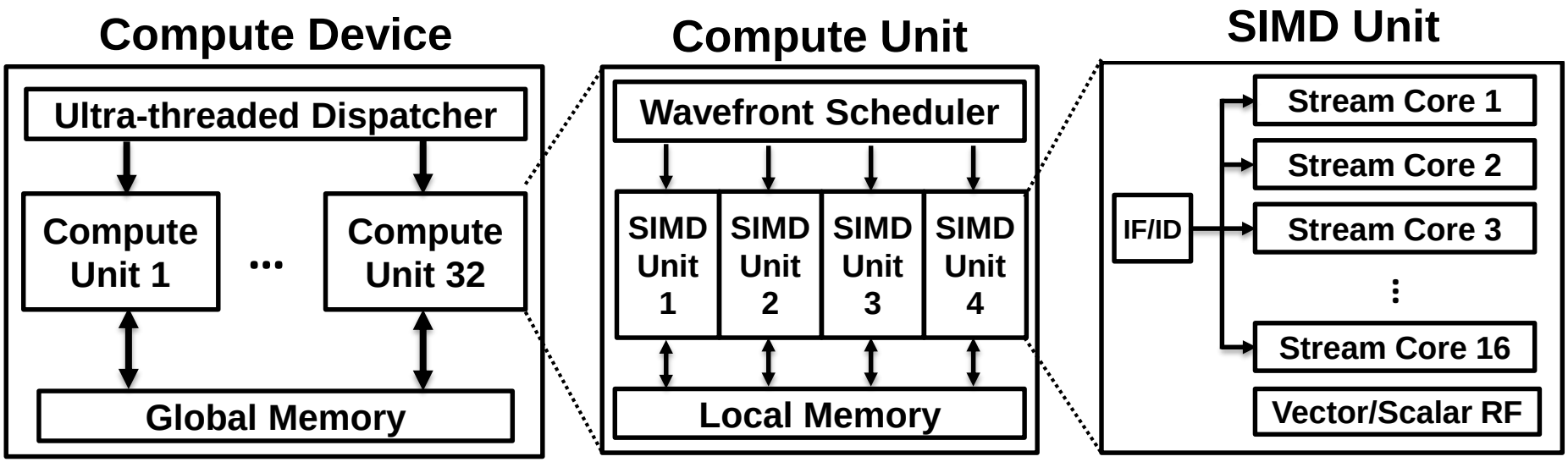
AMD GPU Architecture

- Radeon HD 7970 device from Southern Islands family
  - 32 compute units
    - 4 SIMD units



AMD GPU Architecture

- Radeon HD 7970 device from Southern Islands family
  - 32 compute units
    - 4 SIMD units
      - 16 stream cores (parallel lanes)
- 2048 stream cores per device



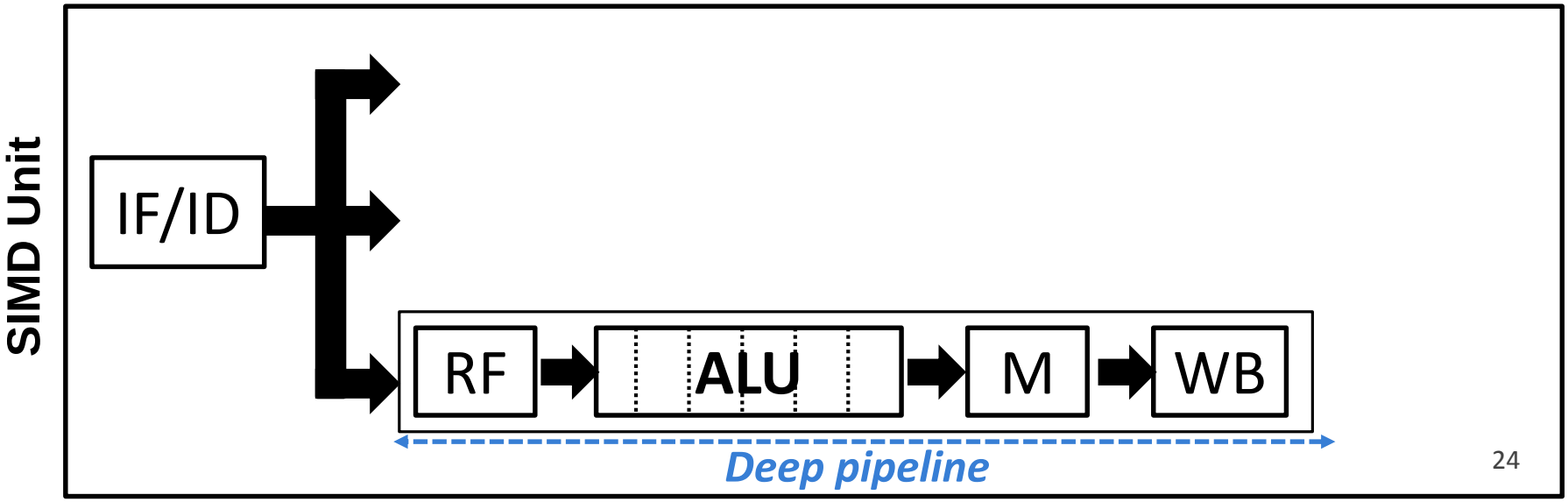
More Expensive Error Correction in SIMD!

- Homogenous workload:
  - No more choices!
  - We do need full utilization (parallelism)
- Cost of recovery is exacerbated in SIMD pipelined:

More Expensive Error Correction in SIMD!

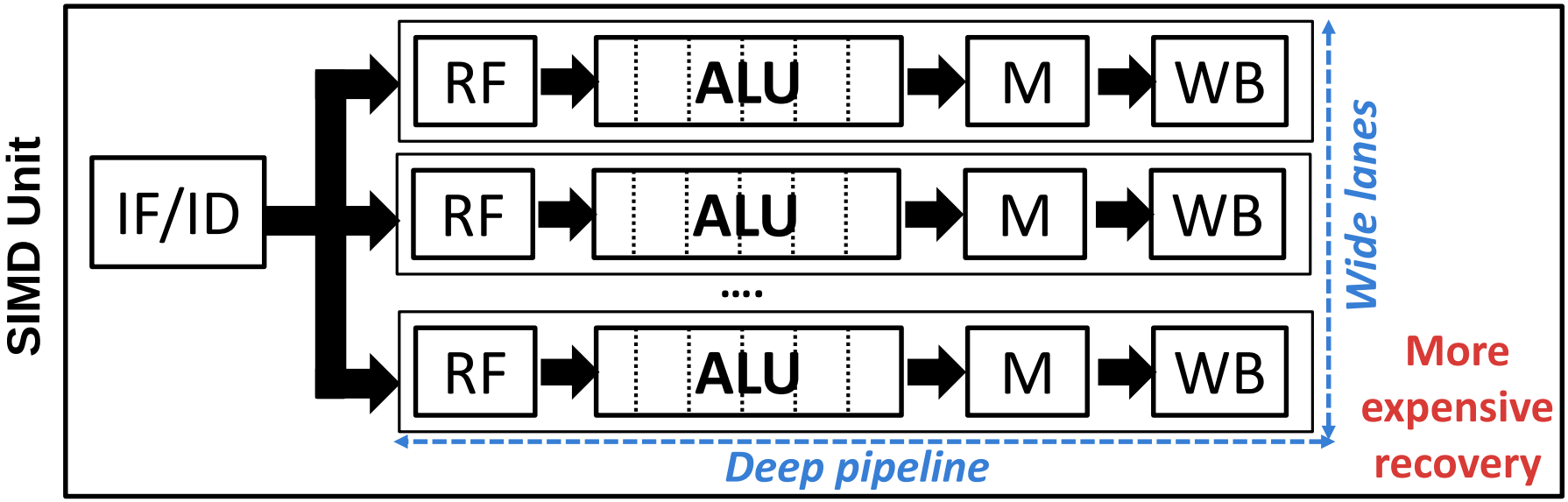
- Homogenous workload:
  - No more choices!
  - We do need full utilization (parallelism)
- Cost of recovery is exacerbated in SIMD pipelined:

  I. Horizontally: higher pipeline latency → higher cost of recovery through flushing and replaying

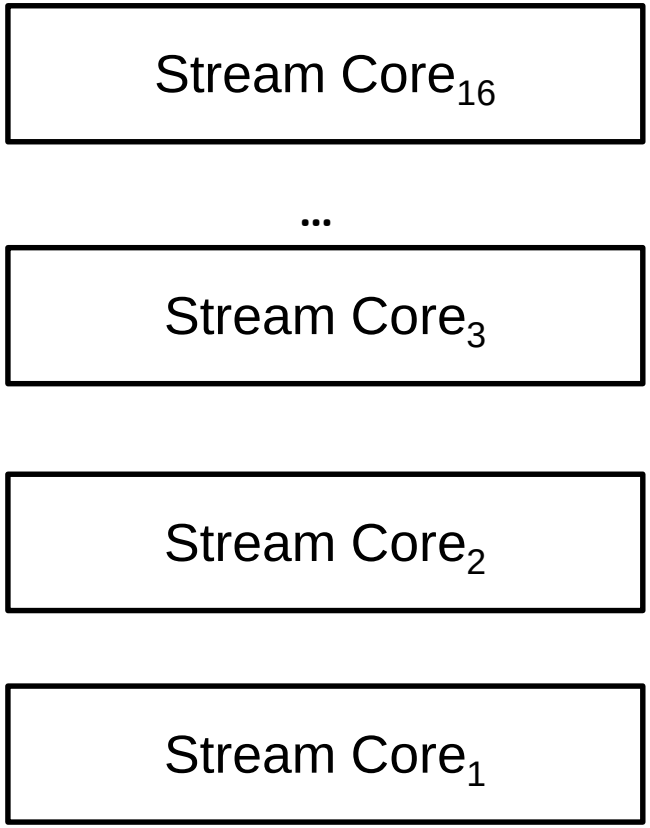


More Expensive Error Correction in SIMD!

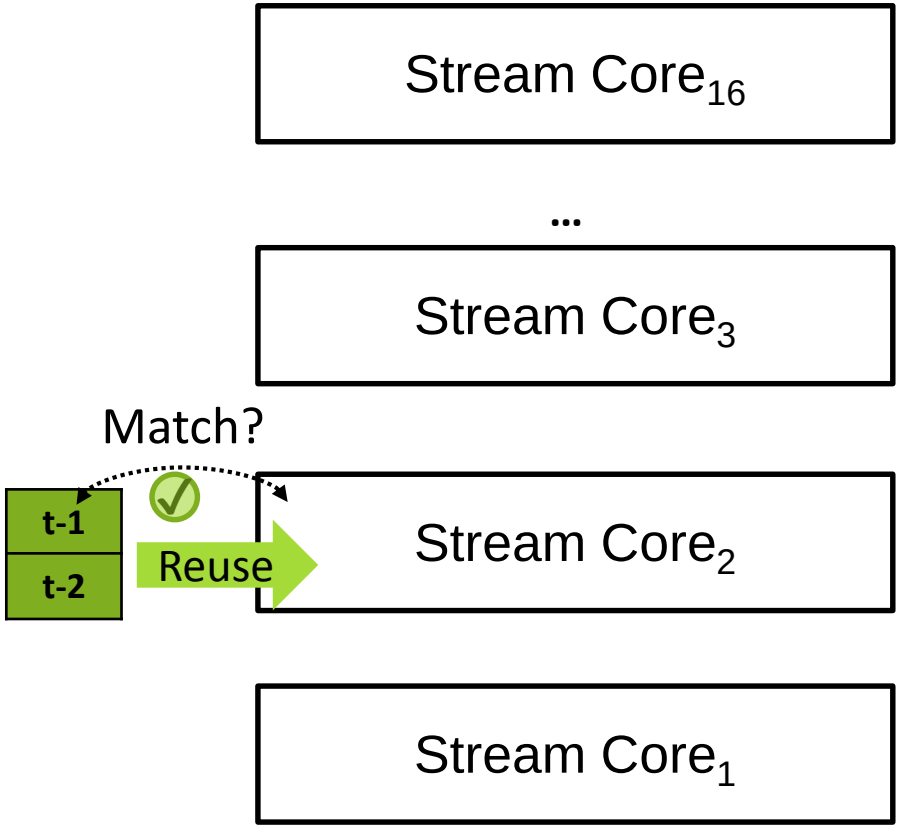
- Homogenous workload:
  - No more choices!
  - We do need full utilization (parallelism)
- Cost of recovery is exacerbated in SIMD pipelined:
  I. Horizontally: higher pipeline latency → higher cost of recovery through flushing and replaying
  II. Vertically: any error within any of the lanes → global stall → recovery of the entire SIMD



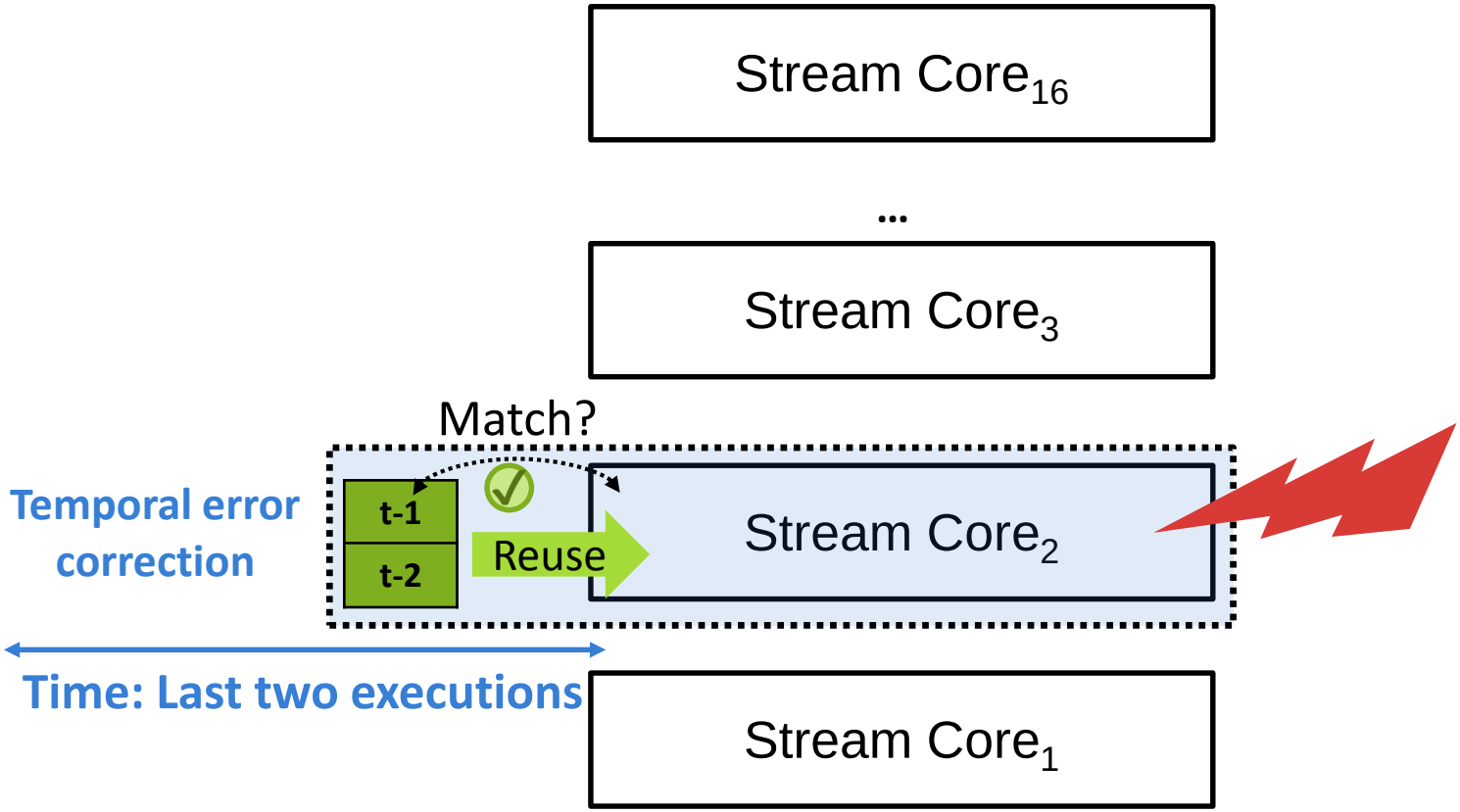
Memoization: Temporal and/or Spatial



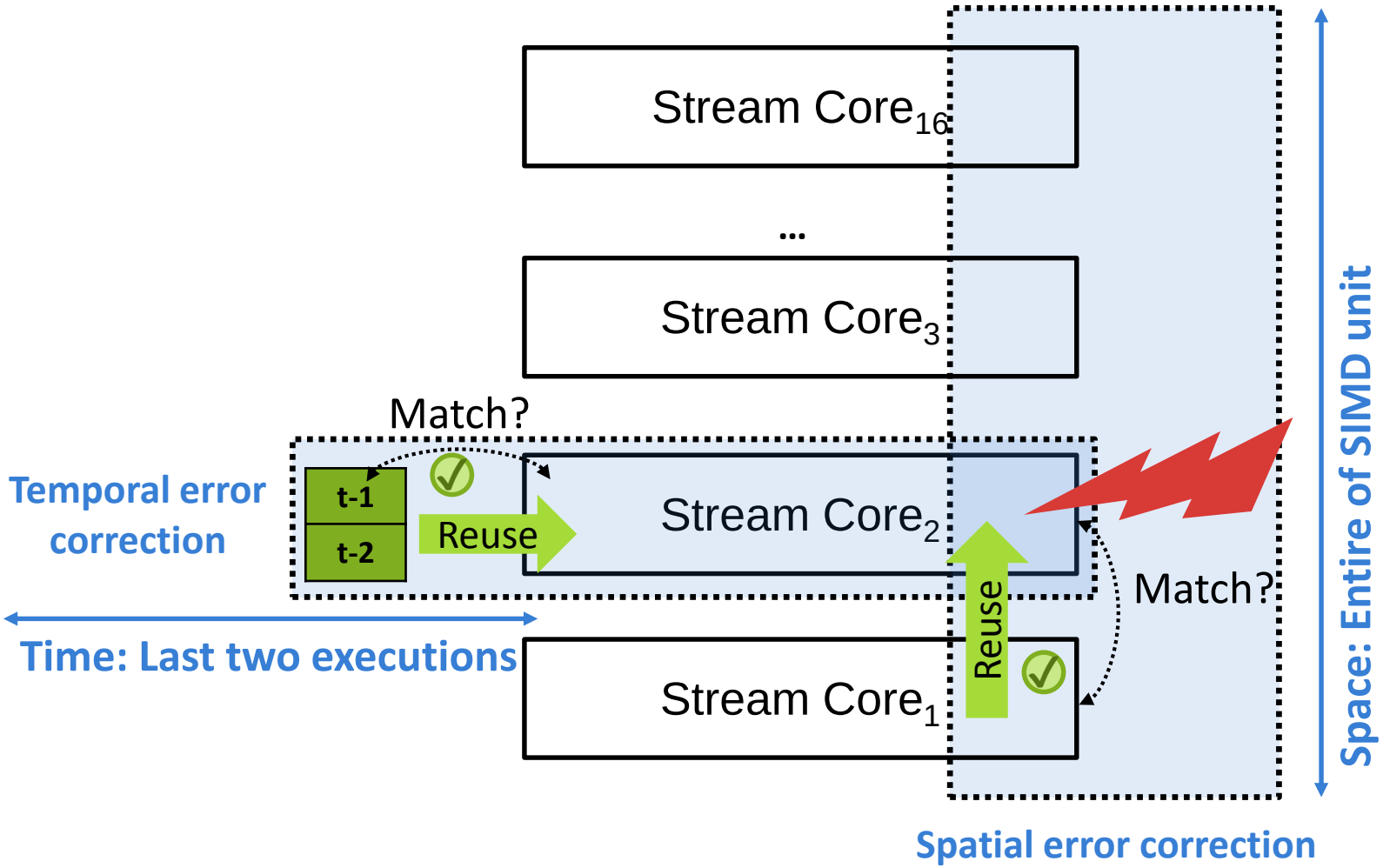
Memoization: Temporal and/or Spatial



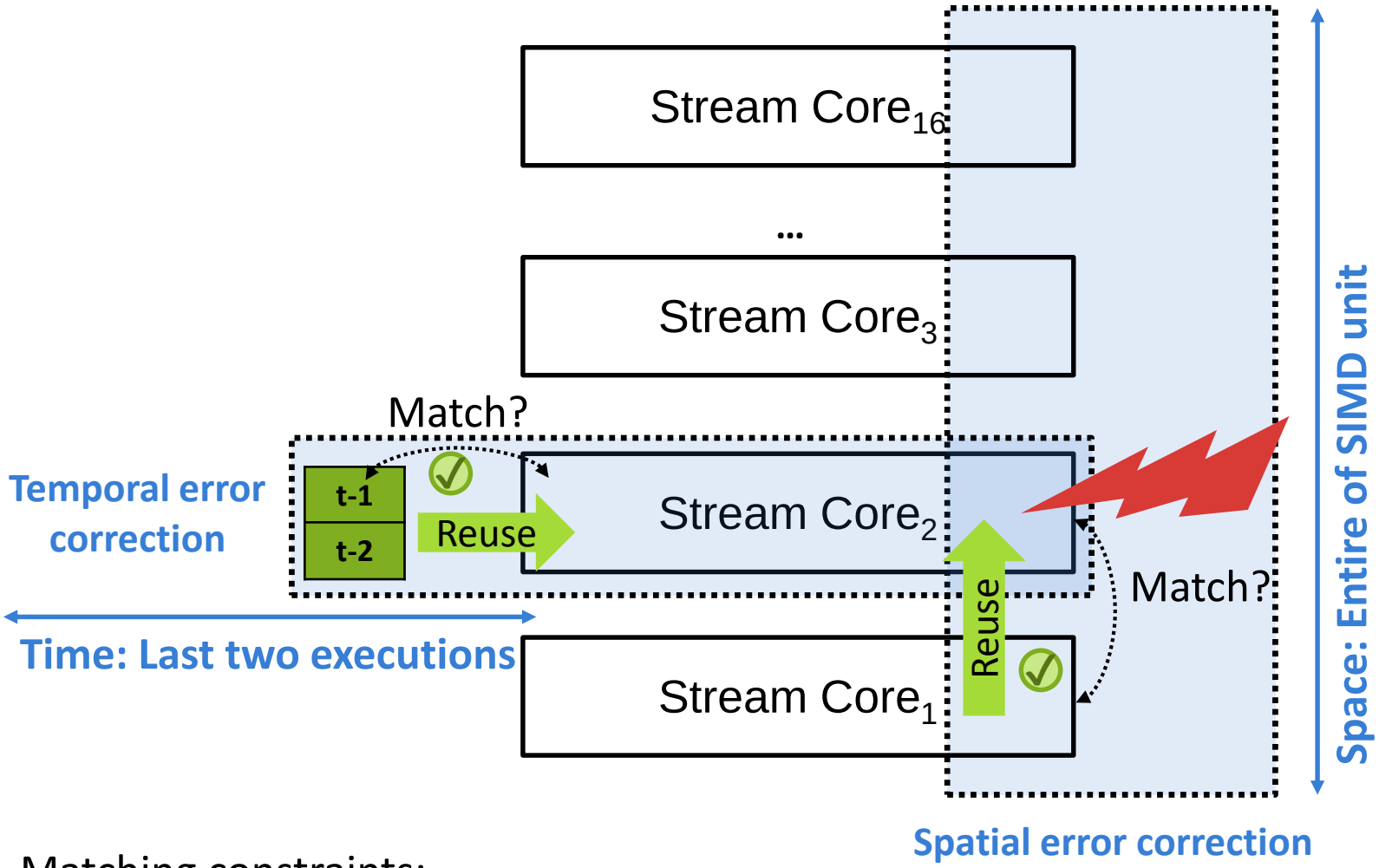
Memoization: Temporal and/or Spatial



Memoization: Temporal and/or Spatial



Memoization: Temporal and/or Spatial



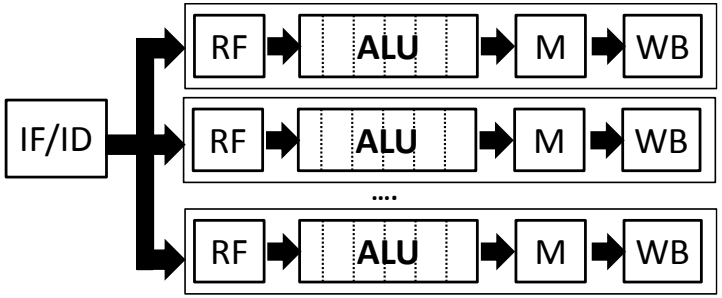
Matching constraints:

I. Exact matching: **full** bit-by-bit matching

II. Approximate matching: **masking** LSB (12-bit) in mantissa

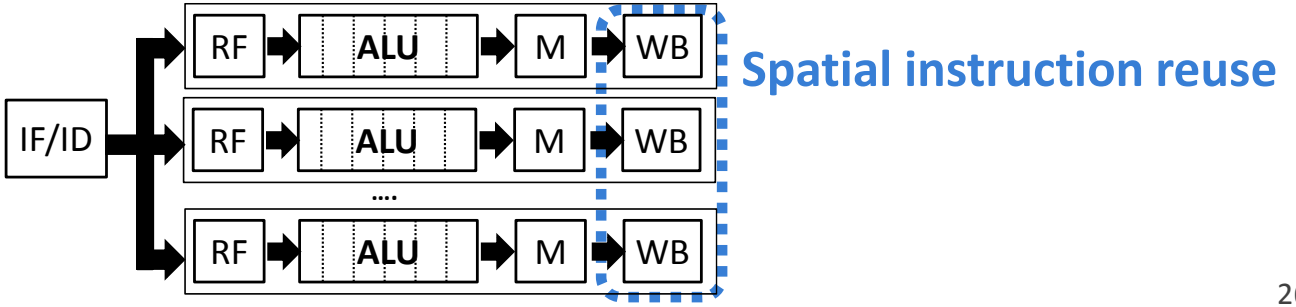
Reducing Cost of Error Correction by Memoization

- Leveraging inherent value locality and similarity



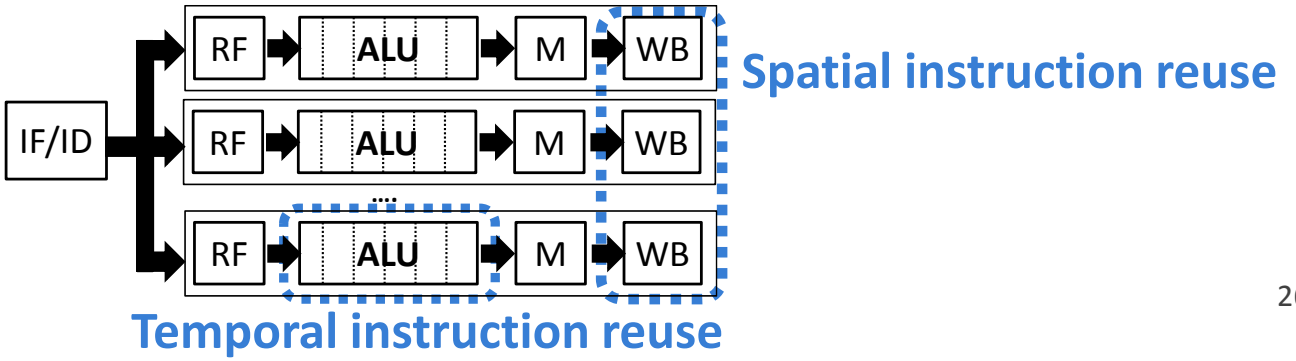
Reducing Cost of Error Correction by Memoization

- Leveraging inherent value locality and similarity
  - Whether an instruction can be reused spatially across various parallel lanes?



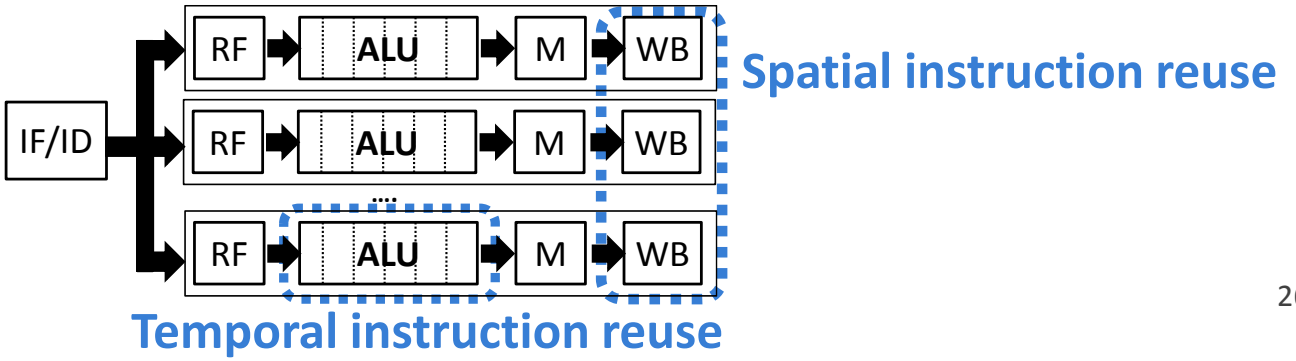
Reducing Cost of Error Correction by Memoization

- Leveraging inherent value locality and similarity
  - Whether an instruction can be reused spatially across various parallel lanes?
  - Whether an instruction can be reused temporally for a lane itself?



Reducing Cost of Error Correction by Memoization

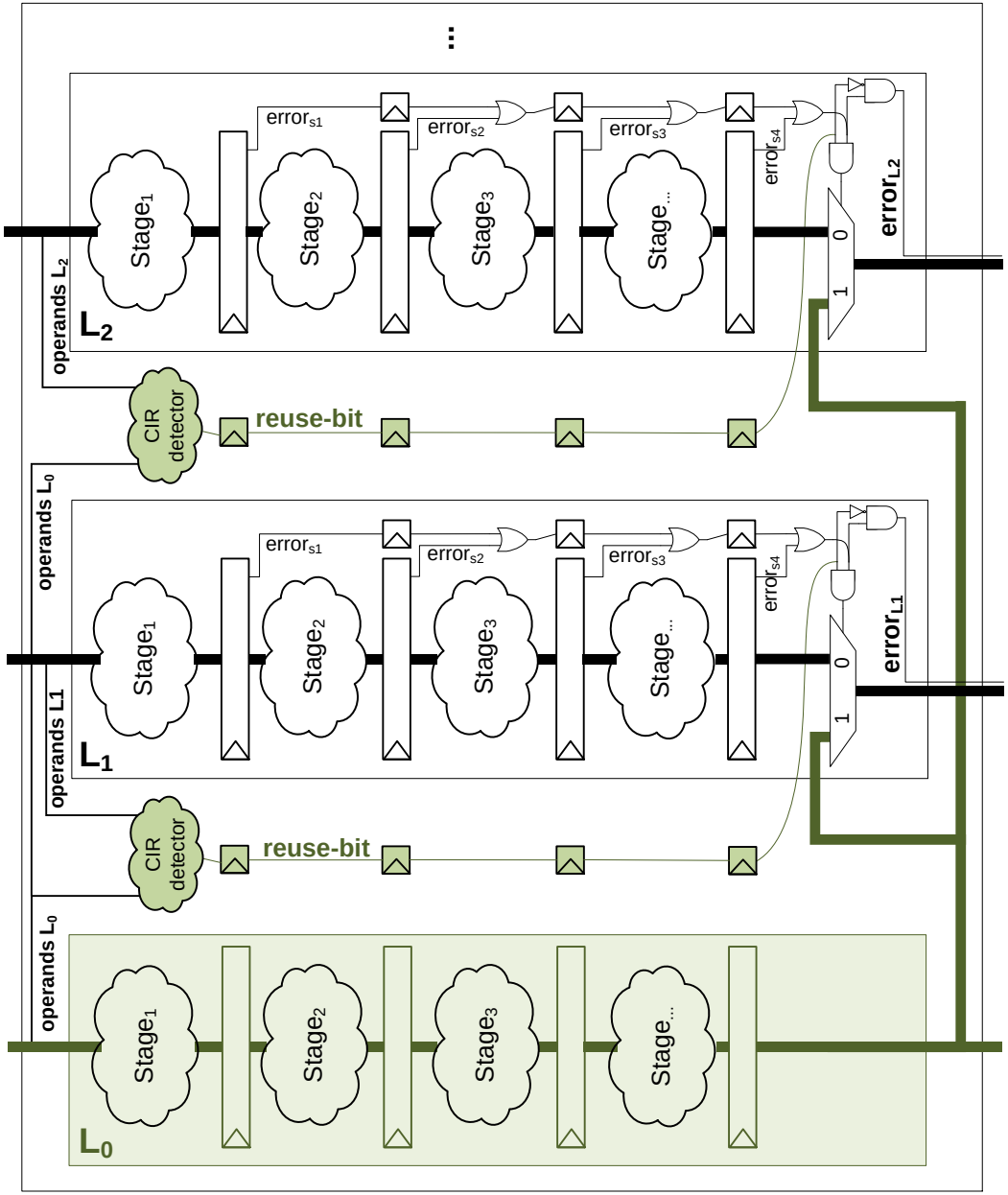
- Leveraging inherent value locality and similarity
  - Whether an instruction can be reused spatially across various parallel lanes?
  - Whether an instruction can be reused temporally for a lane itself?
- **Memoization for exact/approximate error correction:**
  1) **Recalls** the result of an error-free execution on an instance of data.
  2) **Reuses** this memoized context when the matching constraint is met.



Architectural Support for Memoization

Single Strong lane

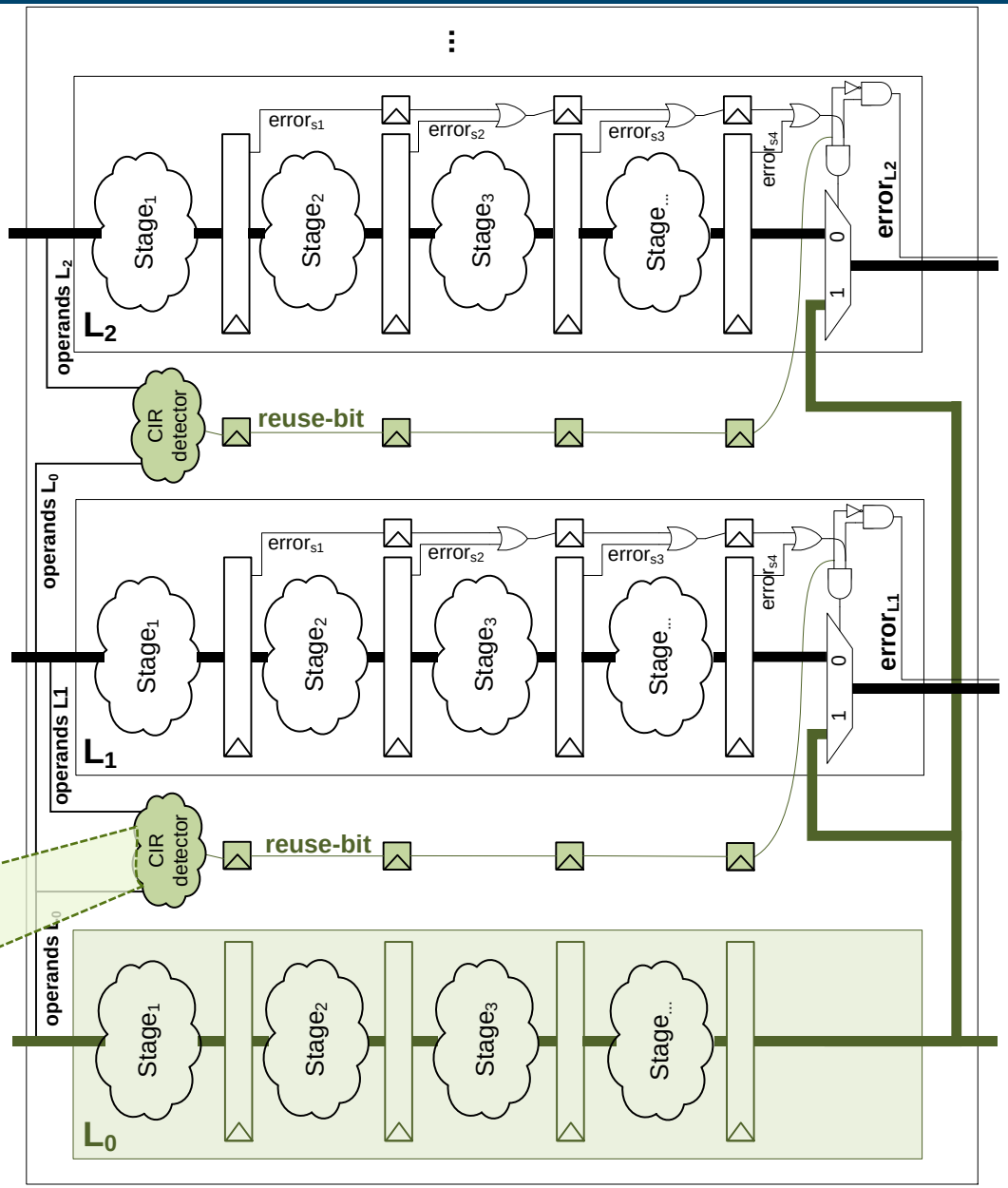
Multiple Weak lanes (SSMW)

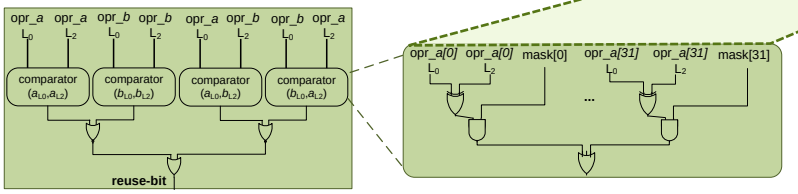


Architectural Support for Memoization

Single Strong lane

Multiple Weak lanes (SSMW)



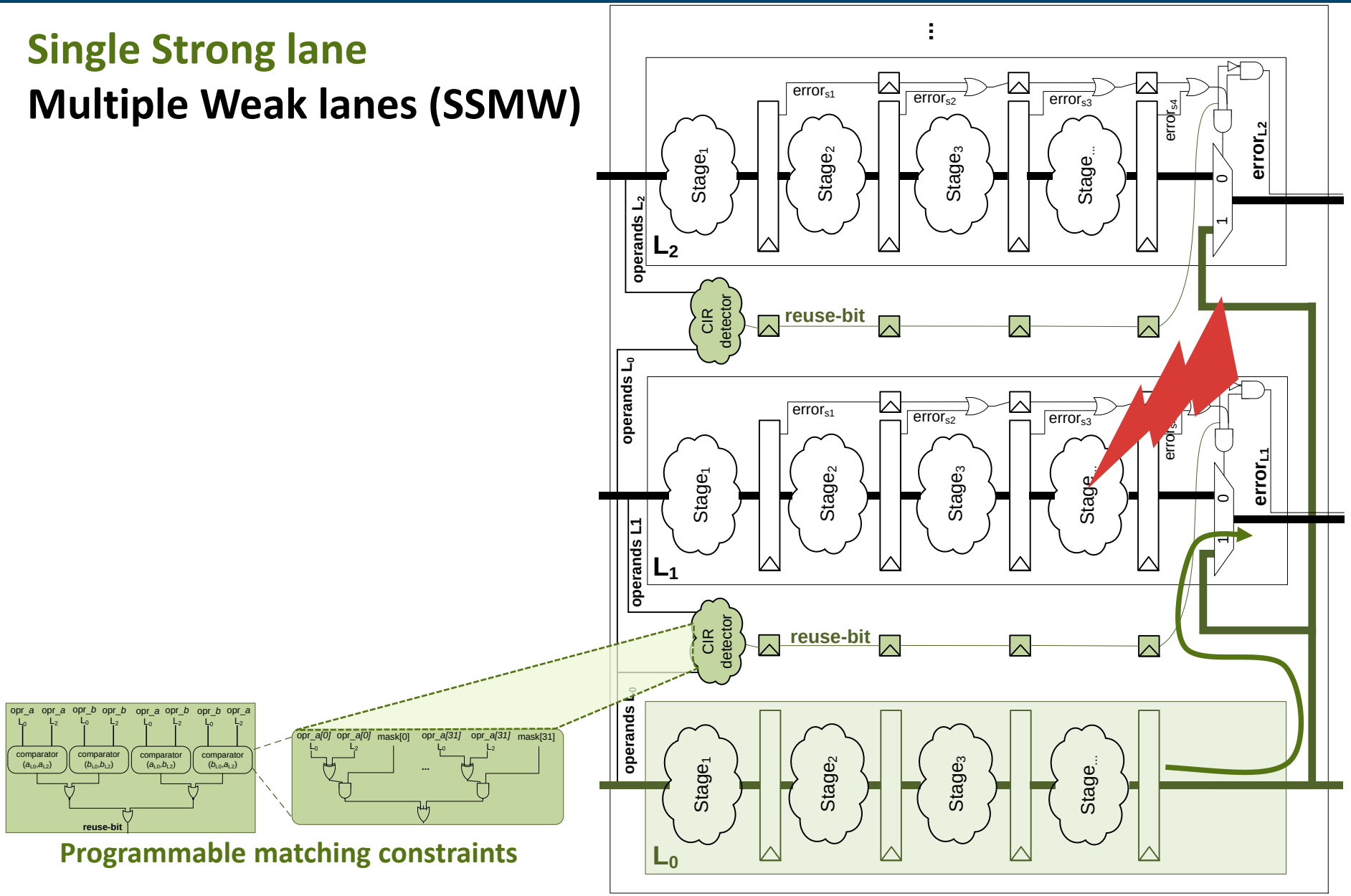


Programmable matching constraints

Architectural Support for Memoization

Single Strong lane

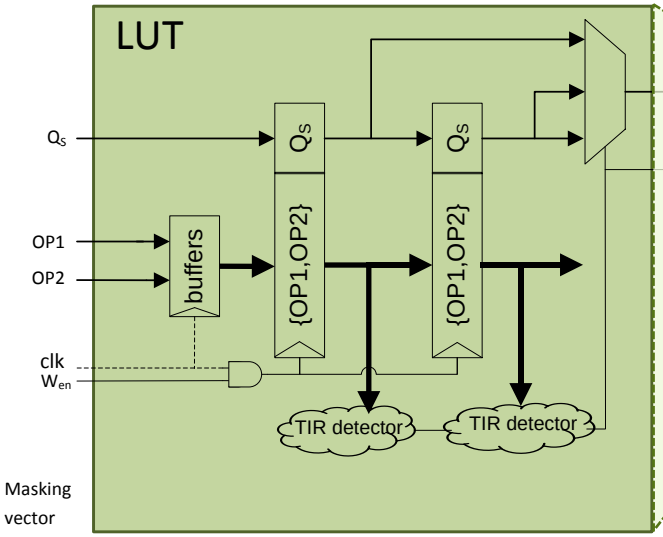
Multiple Weak lanes (SSMW)

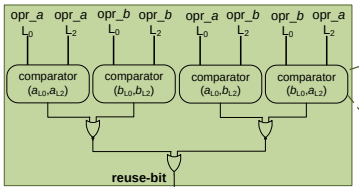


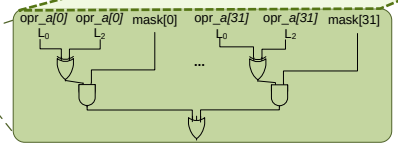
Architectural Support for Memoization

Single Strong lane

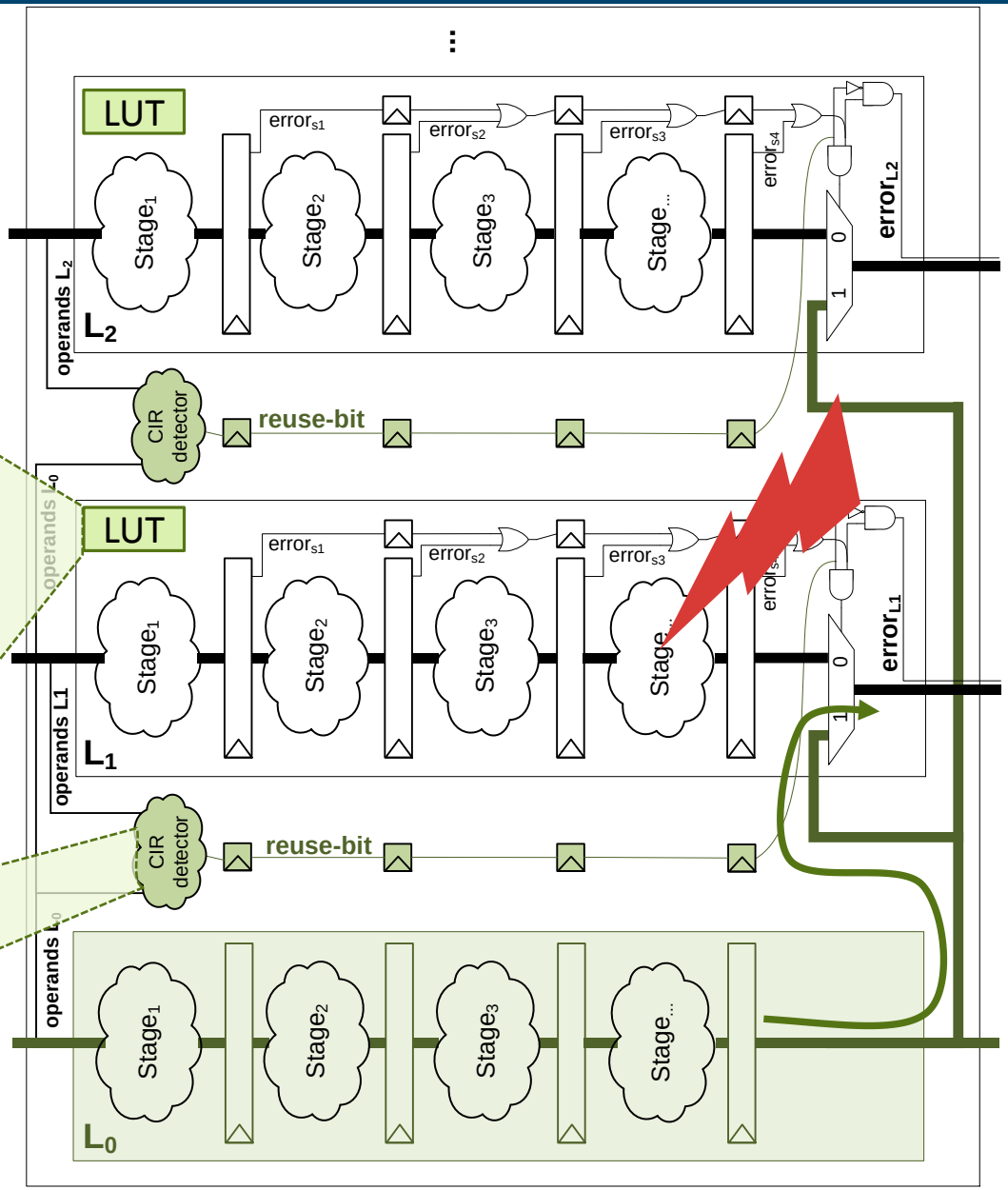
Multiple Weak lanes (SSMW)







Programmable matching constraints



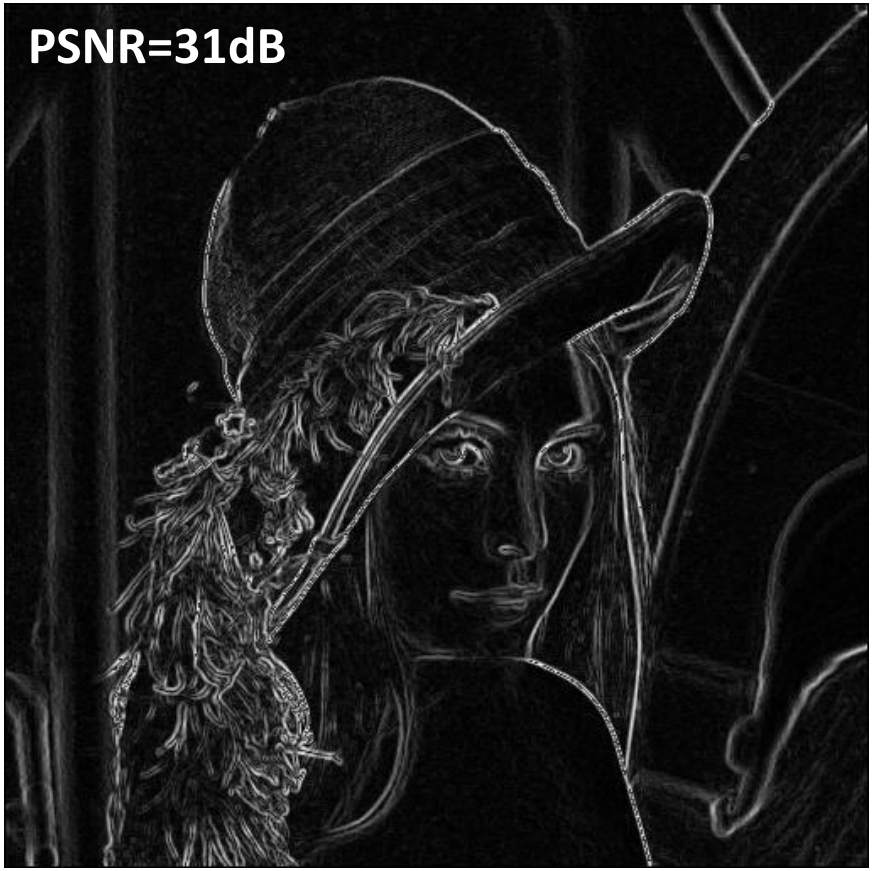
Sobel Filter: Exact vs. Approximate

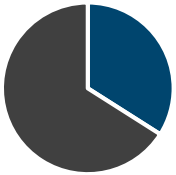
Sobel Filter: Exact vs. Approximate

Exact

100% Exact instruction

Approximate



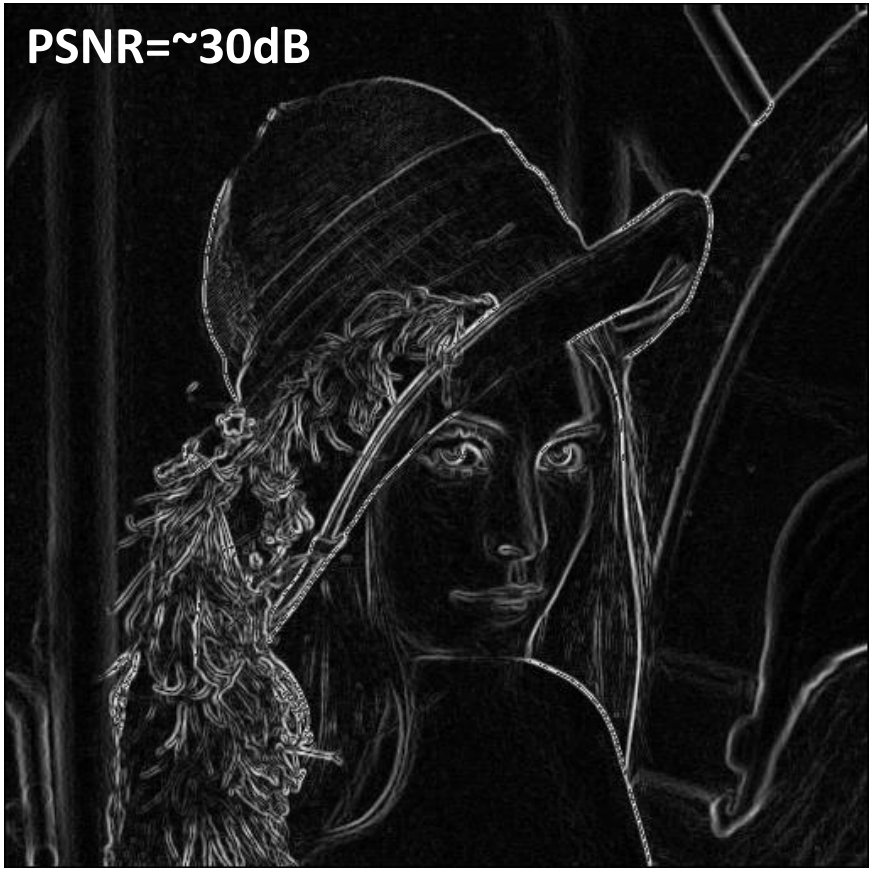


34% Spatial reuse

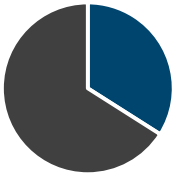
Sobel Filter: Exact vs. Approximate

Exact

Approximate



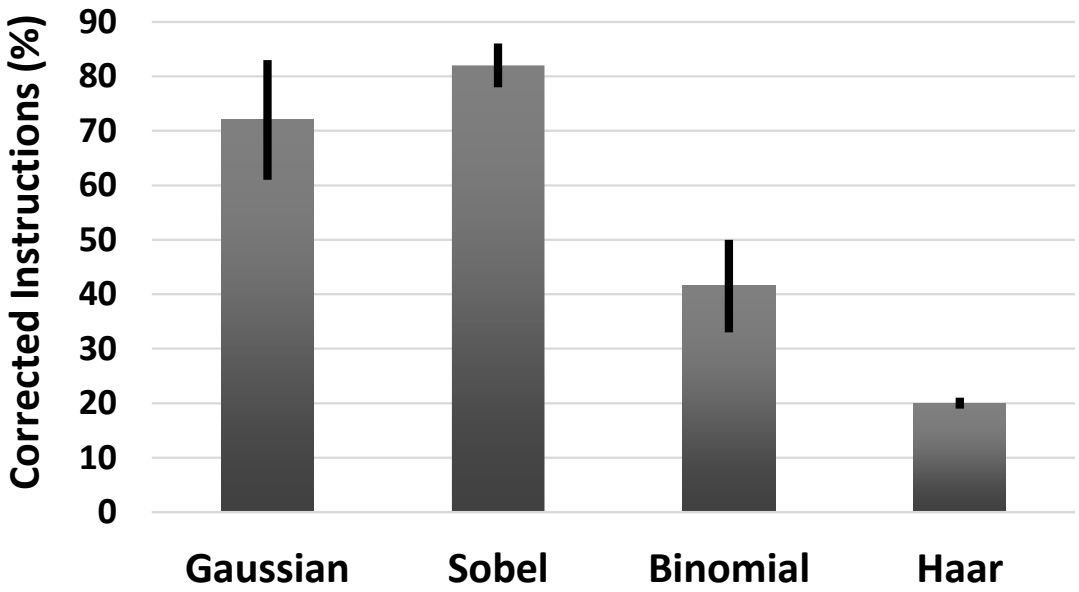
100% Exact instruction



34% Spatial reuse

42% Spatial + Temporal reuse

Corrected Errors by Memoization



- Injecting 3%–6% voltage droops during kernel execution
- TSMC 45nm with 10% guardband for the strong lane
- AMD GPU Radeon HD 5870
- Kernels from AMD Accelerated Parallel Processing SDK 2.5
- Spatial memoization avoids the recovery for **53%** of the errant instructions.

Increasing Effectiveness of Memoization

Variability

↓

Reducing Guardband

↓

Timing Errors

↓

Correcting Errors by Memoization (Architectural)

Increasing Effectiveness of Memoization

Variability

↓

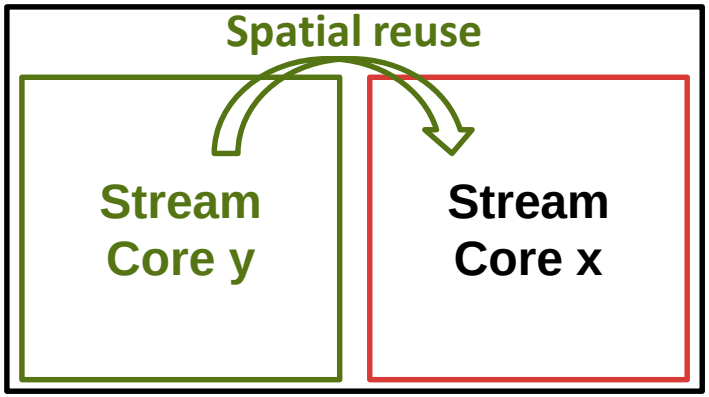
Reducing Guardband

↓

Timing Errors

↓

Correcting Errors by Memoization (Architectural)



× **Scalability**

Increasing Effectiveness of Memoization

Variability

↓

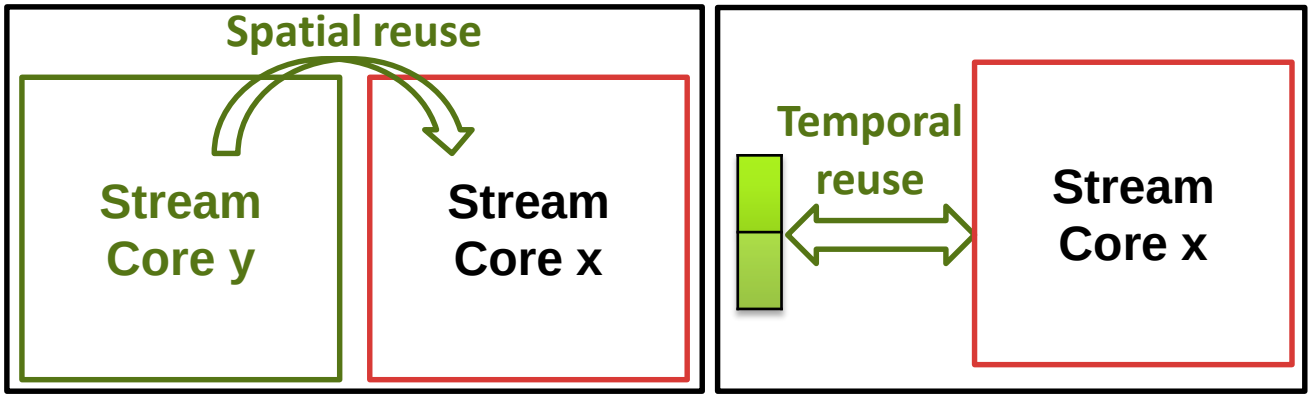
Reducing Guardband

↓

Timing Errors

↓

Correcting Errors by Memoization (Architectural)



× **Scalability** × **Density**

Increasing Effectiveness of Memoization

Variability

↓

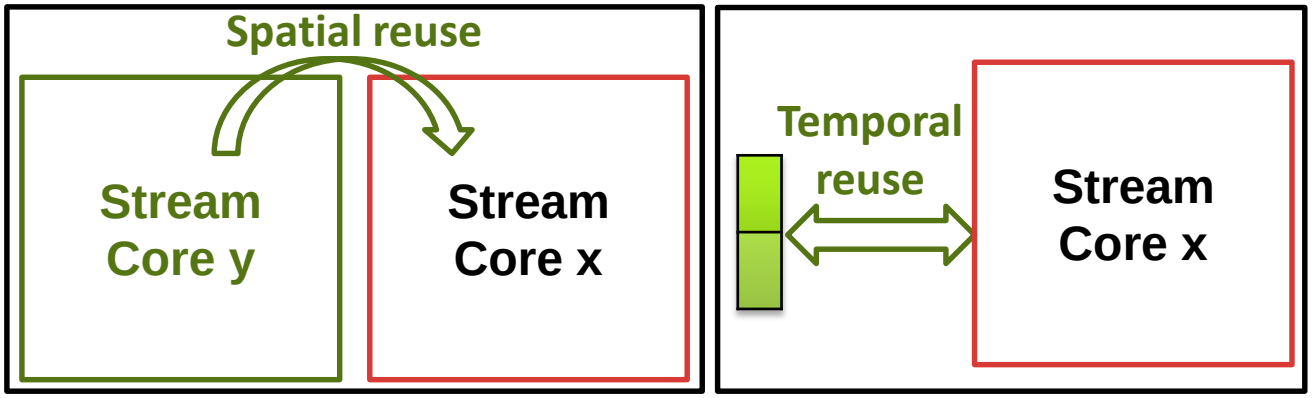
Reducing Guardband

↓

Timing Errors

↓

Correcting Errors by Memoization (Architectural)

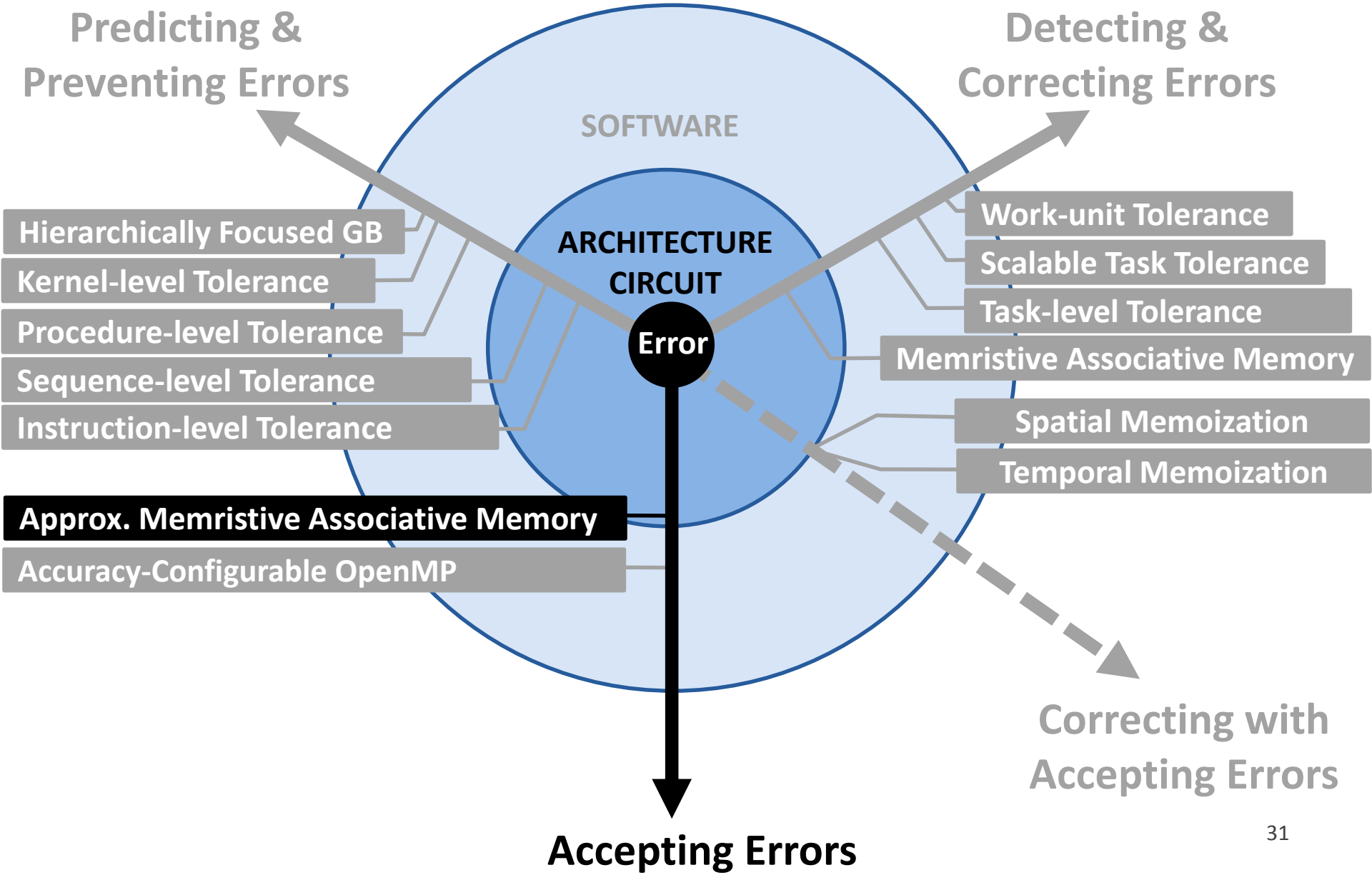


× **Scalability** × **Density**

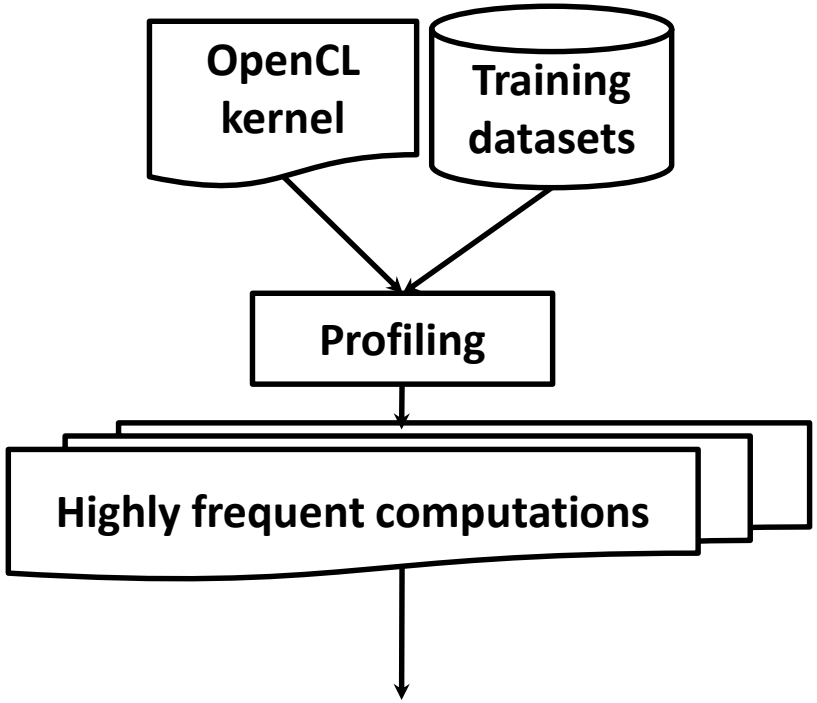
↓

Efficient Memoization (Using Memristor Technology)

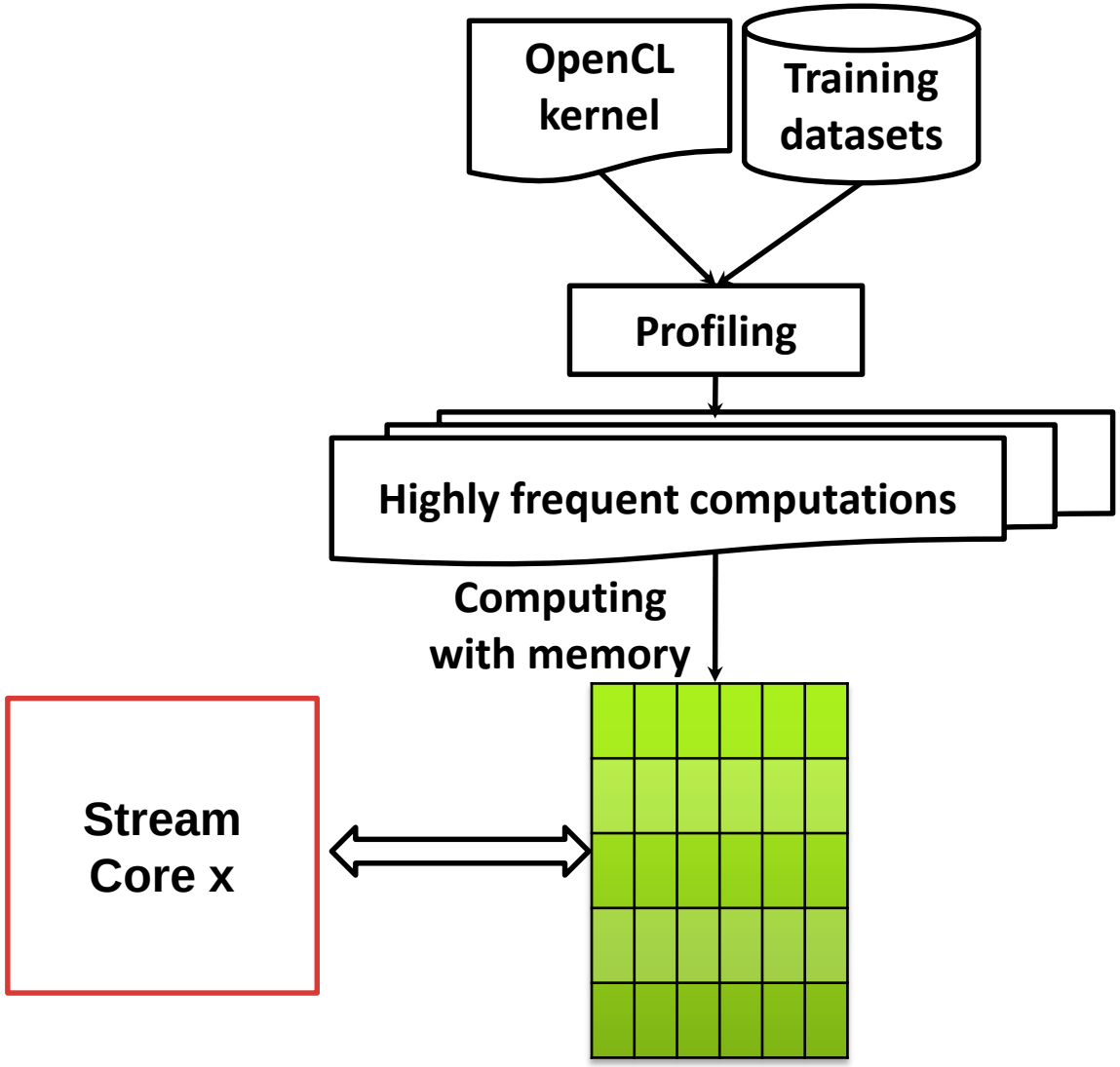
Our Approach to Variability-Tolerance



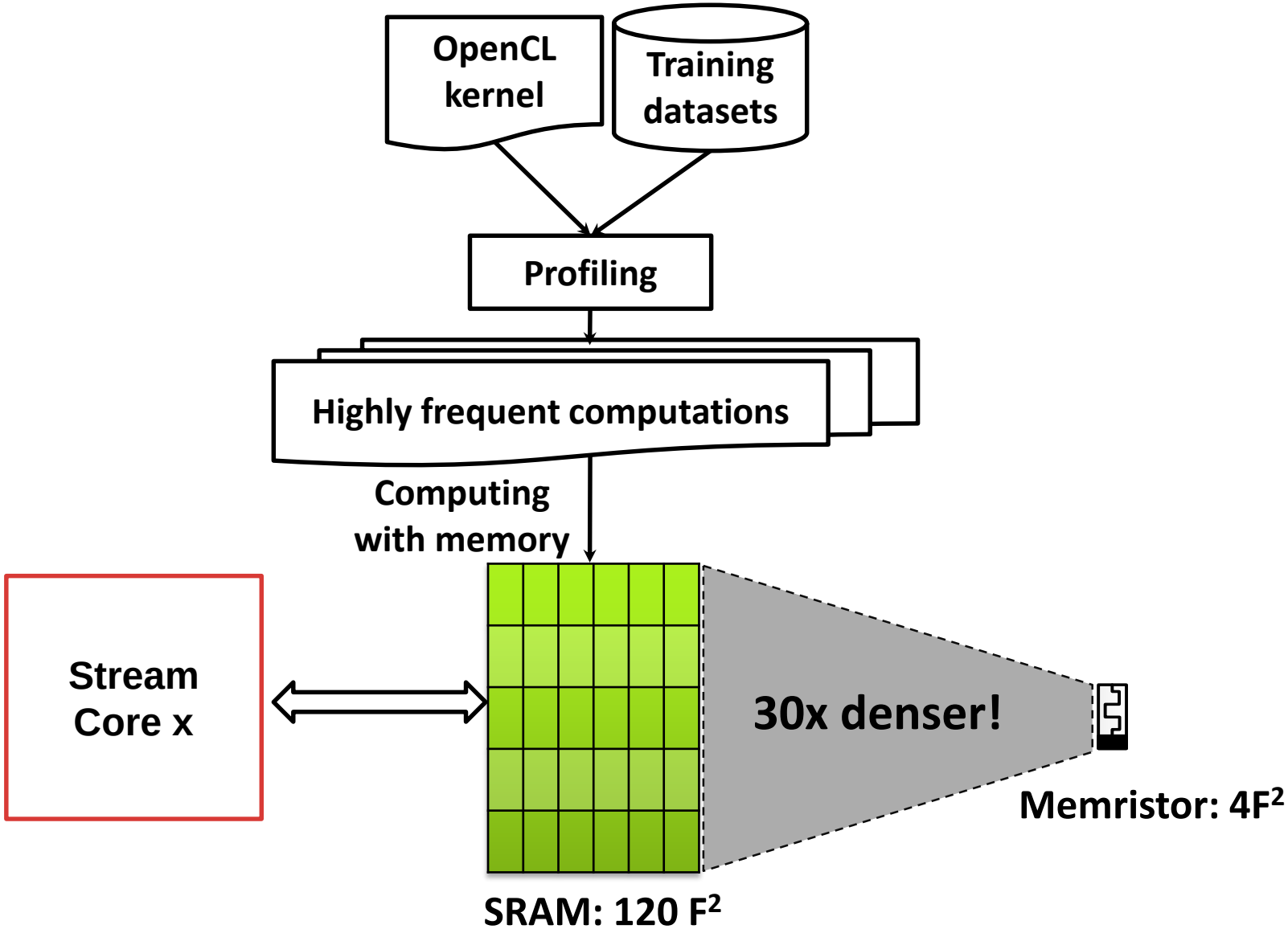
Memristive Associative Memory



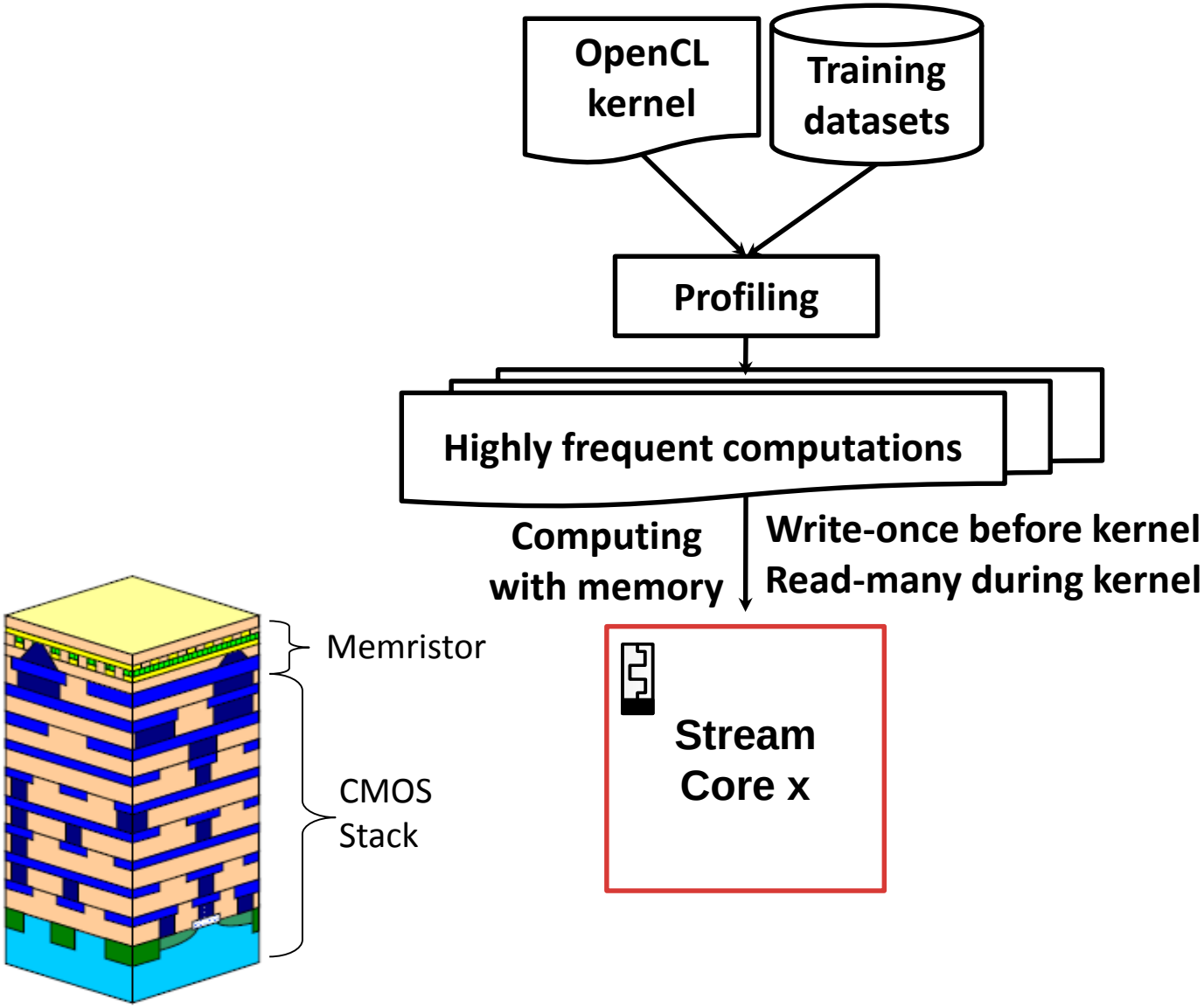
Memristive Associative Memory



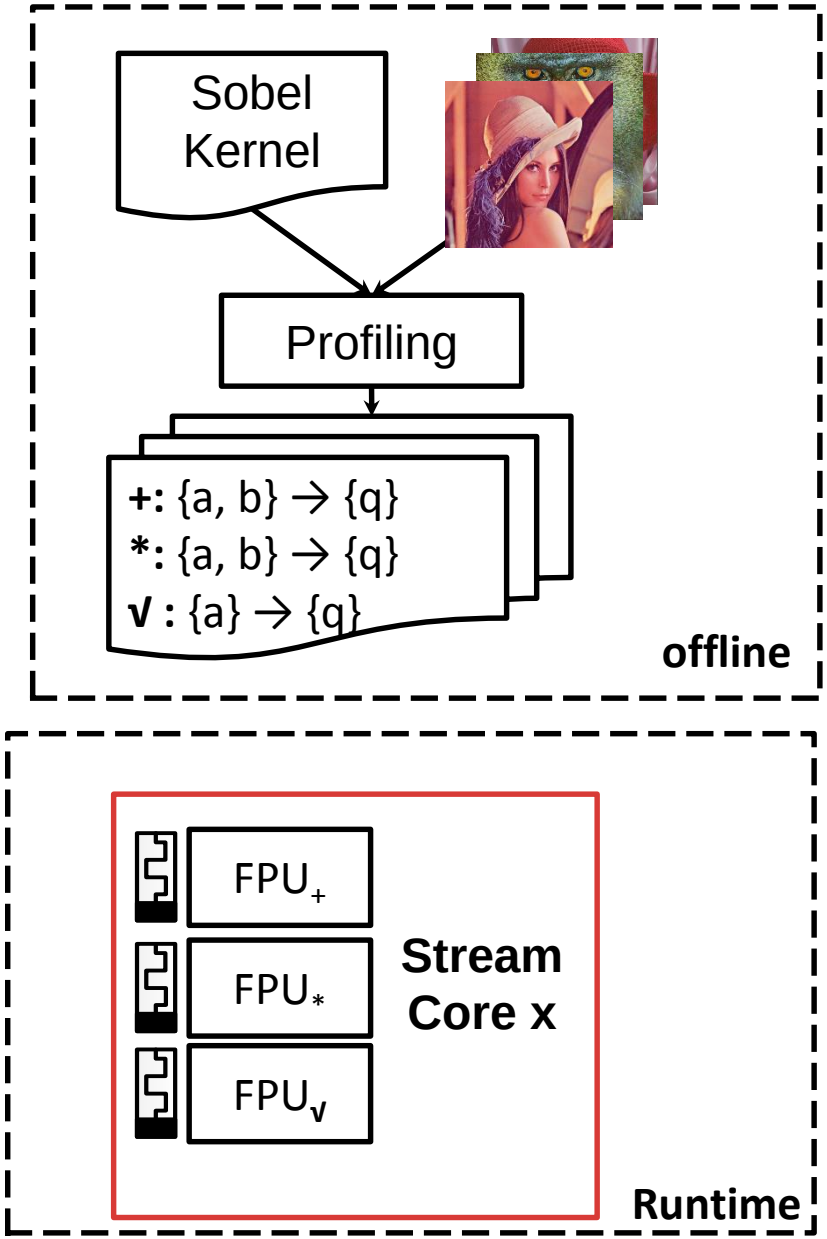
Memristive Associative Memory



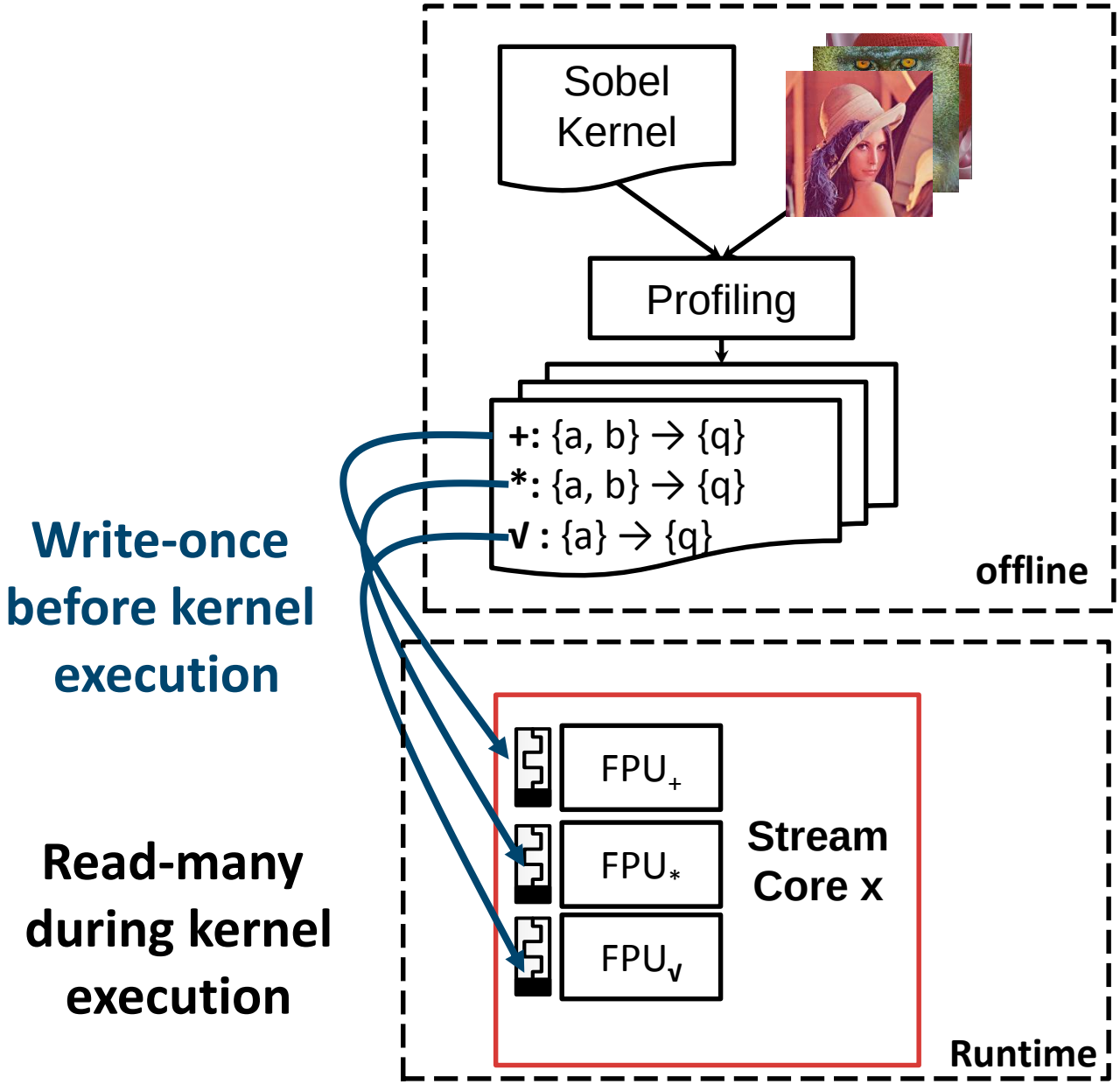
Memristive Associative Memory



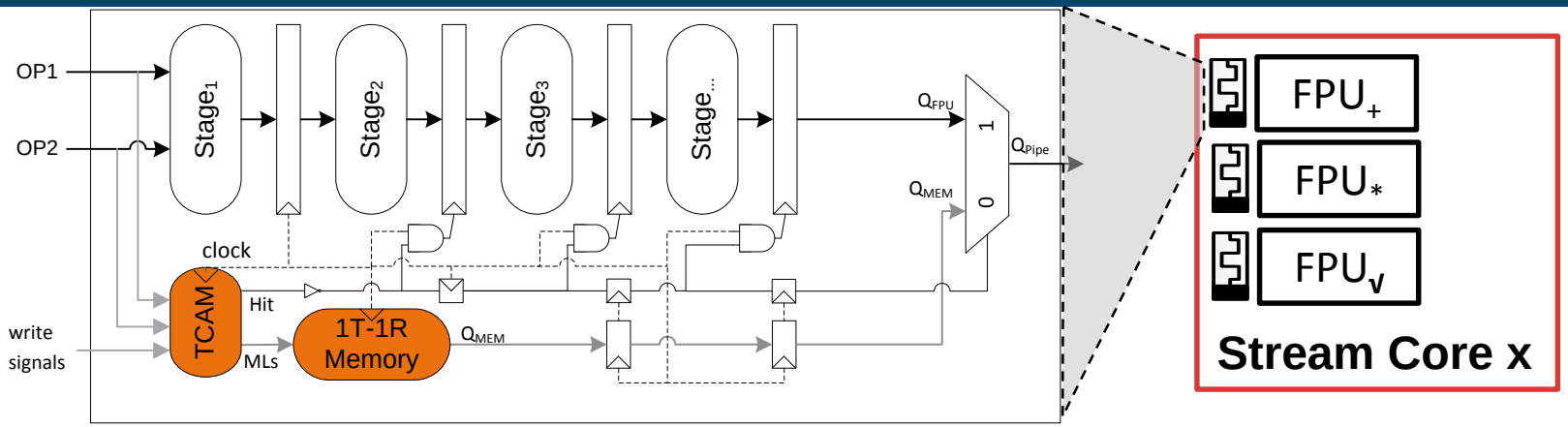
Associative Memory Programming Flow



Associative Memory Programming Flow

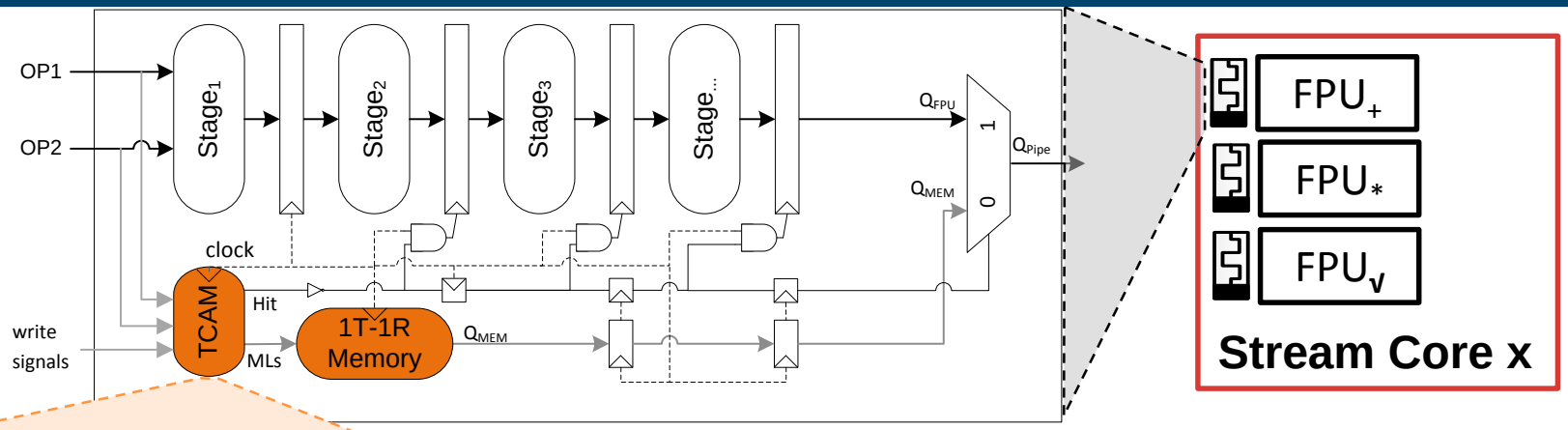


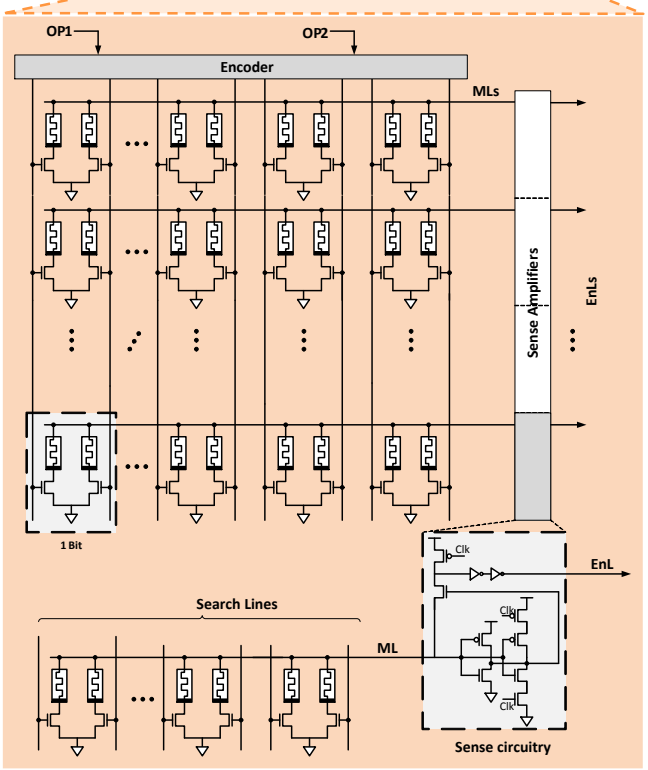
Associative Memory: TCAM + Memory



Execution stage of FPU

Associative Memory: TCAM + Memory

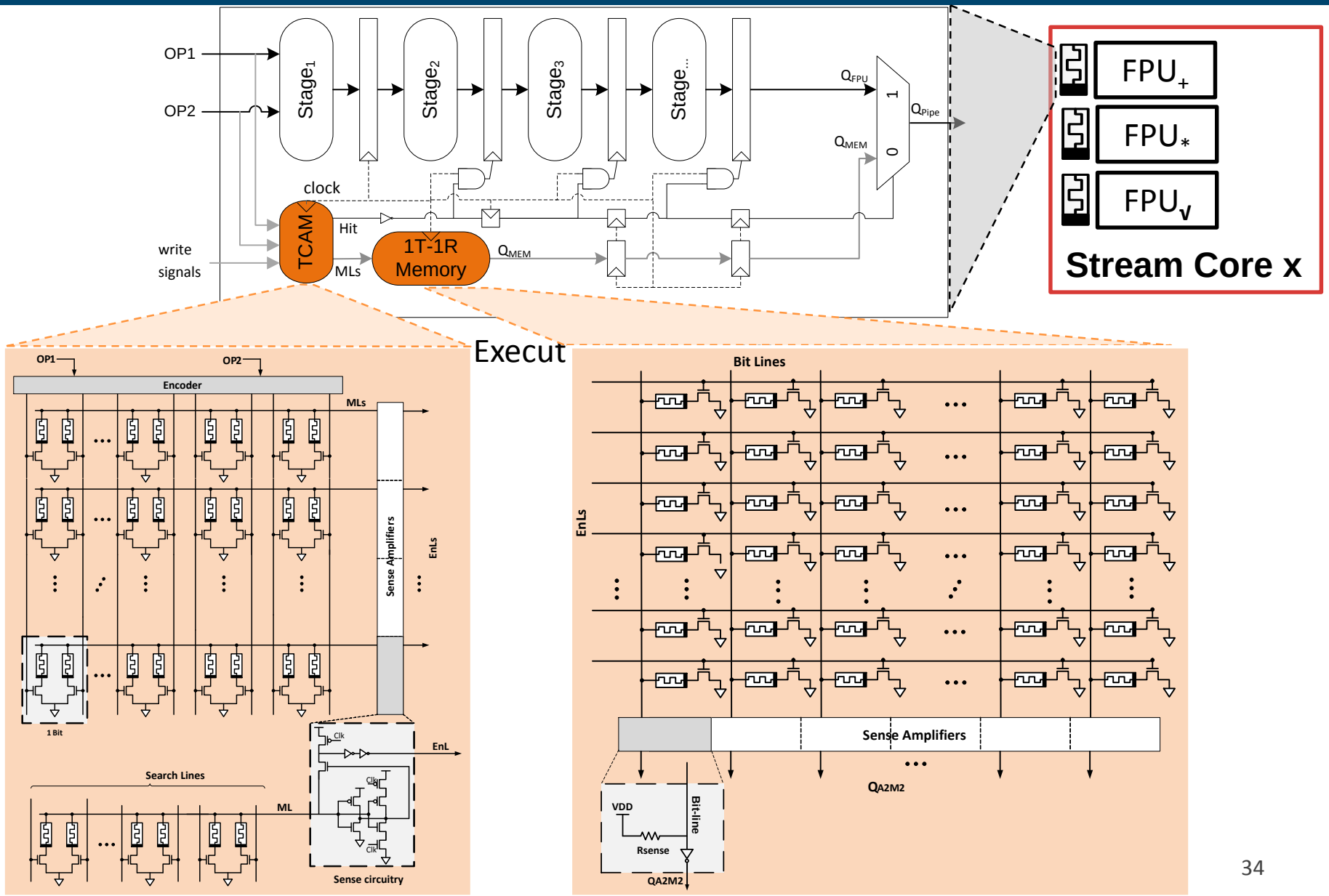




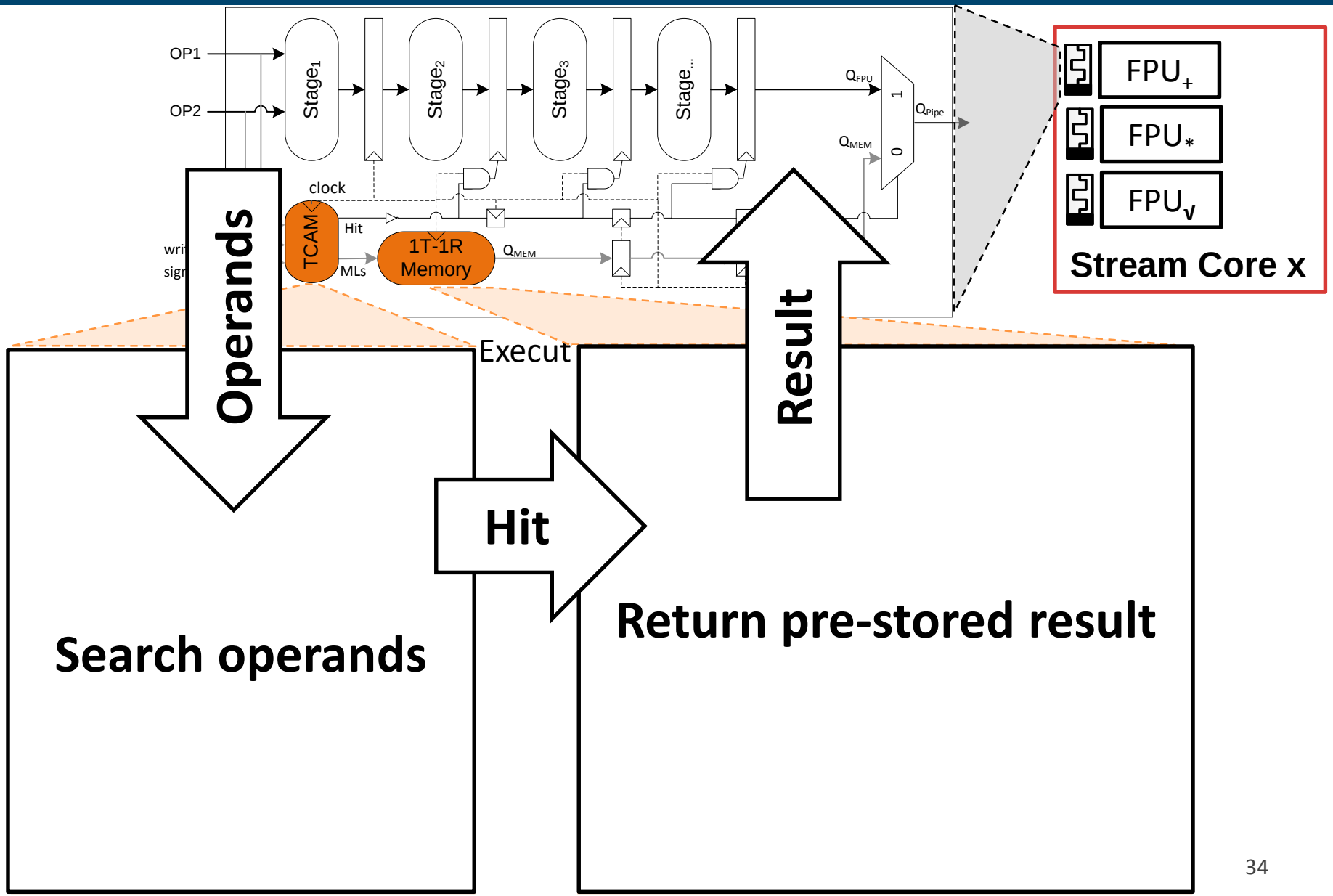
Execution stage of FPU

- Ternary content addressable memory (TCAM)
- 2T-2R (2-transistor/2-resistive)
- 2-bit encoding
- Stores highly frequent input patterns
- 15% positive slack at 45nm

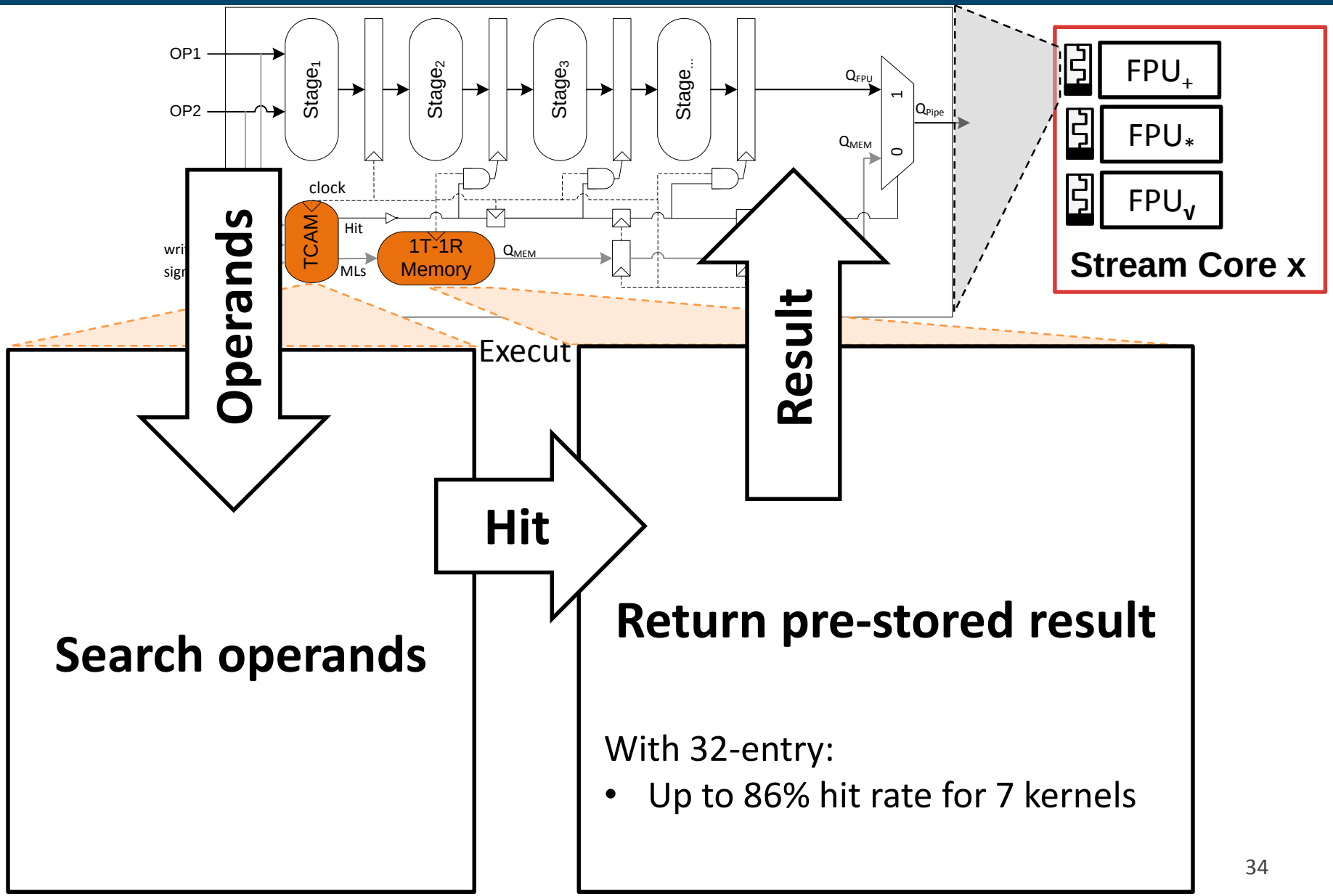
Associative Memory: TCAM + Memory



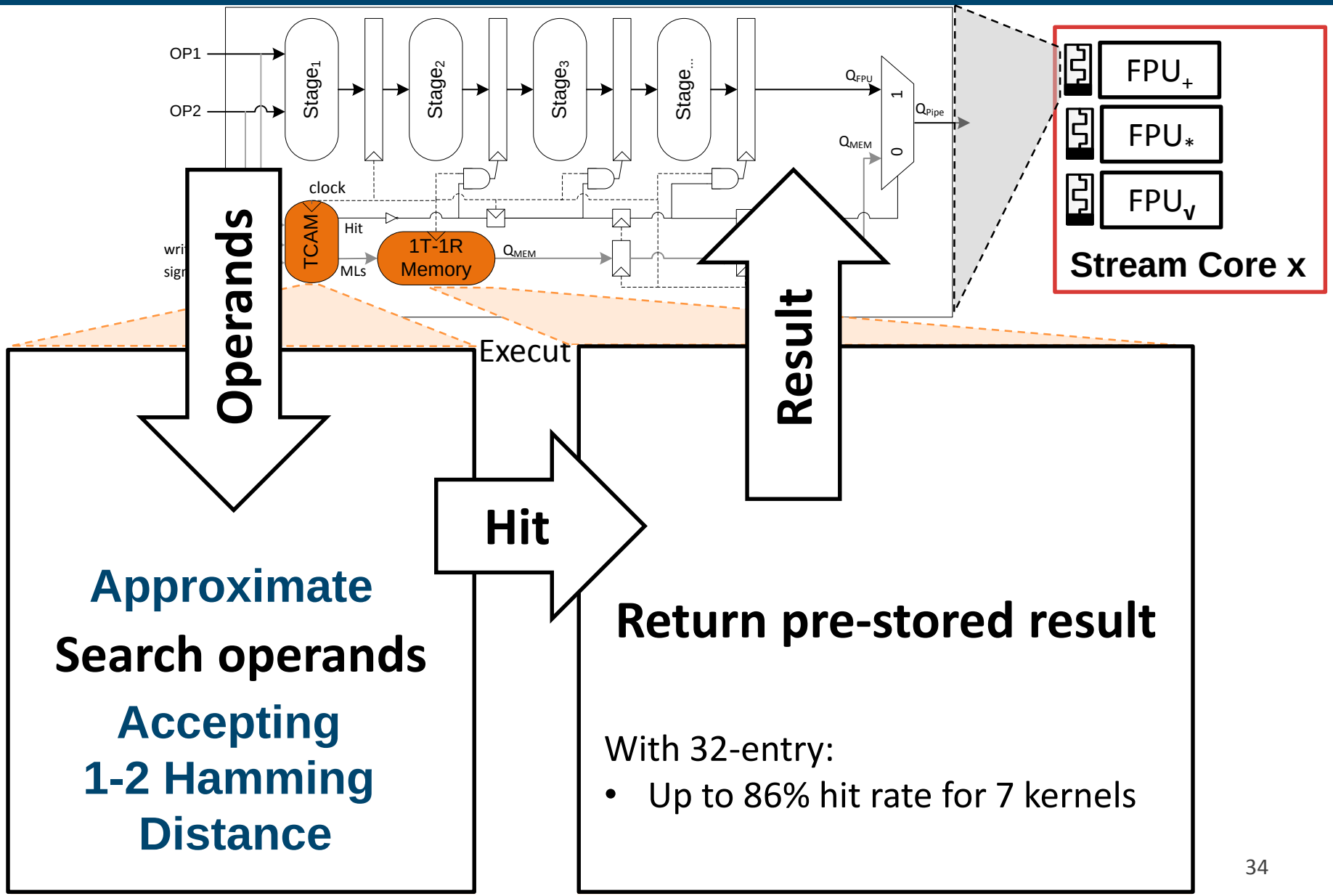
Associative Memory: TCAM + Memory



Associative Memory: TCAM + Memory

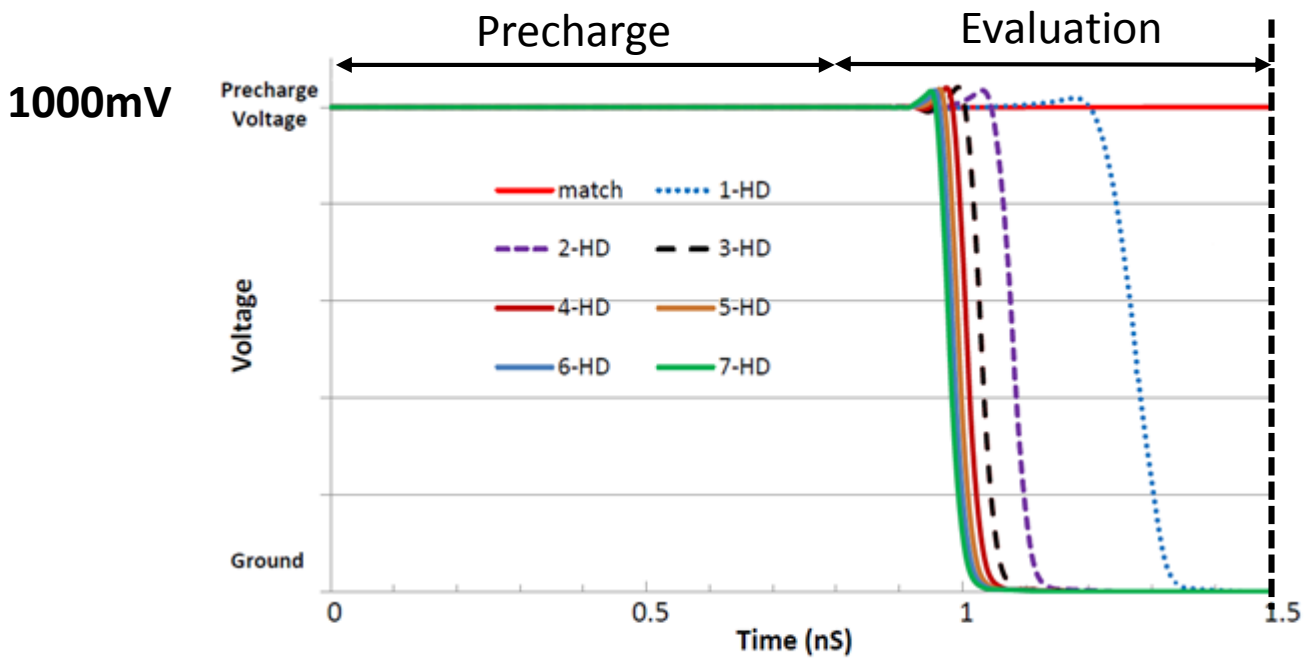


Associative Memory: TCAM + Memory



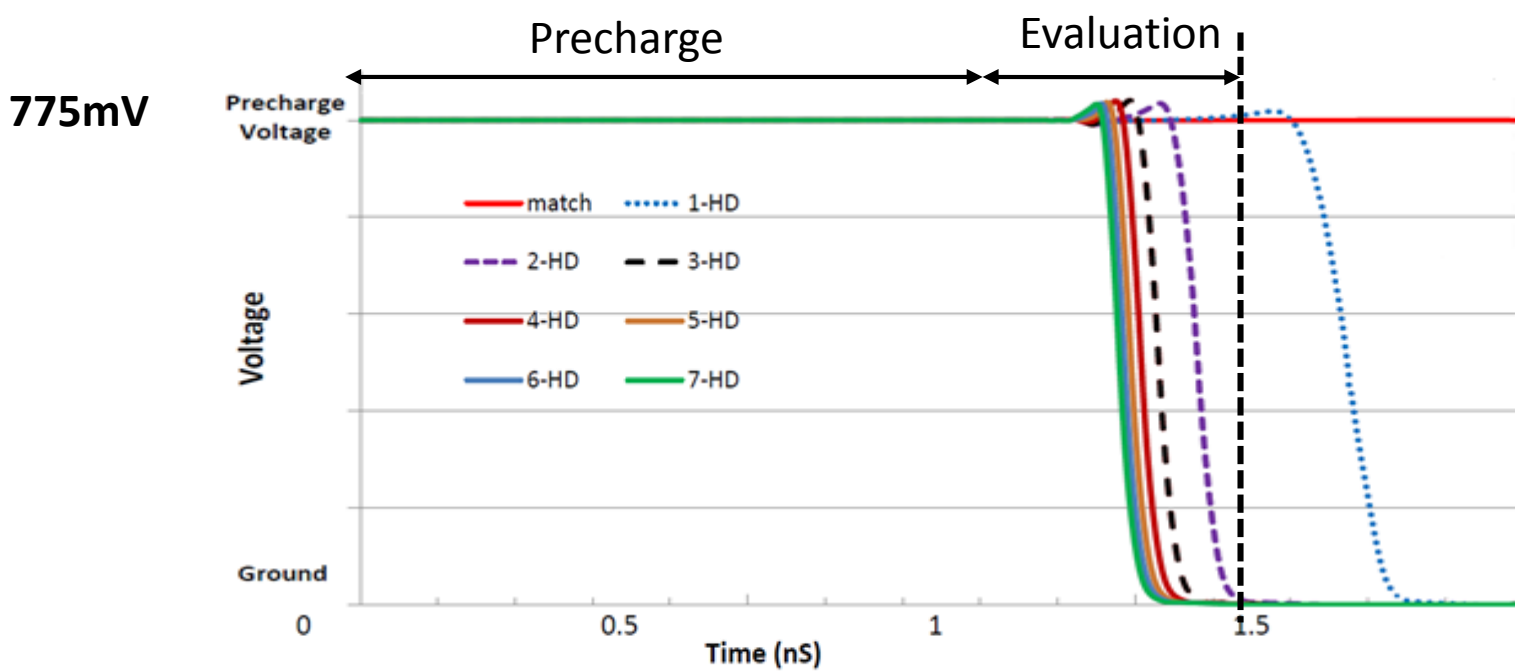
Approximate Search Operation in TCAM

- Lower voltages subject to *controllable* approximate matching:
  - 1000mV: Exact Matching
  - 775mV: 1-HD Matching
  - 720mV: 2-HD Matching



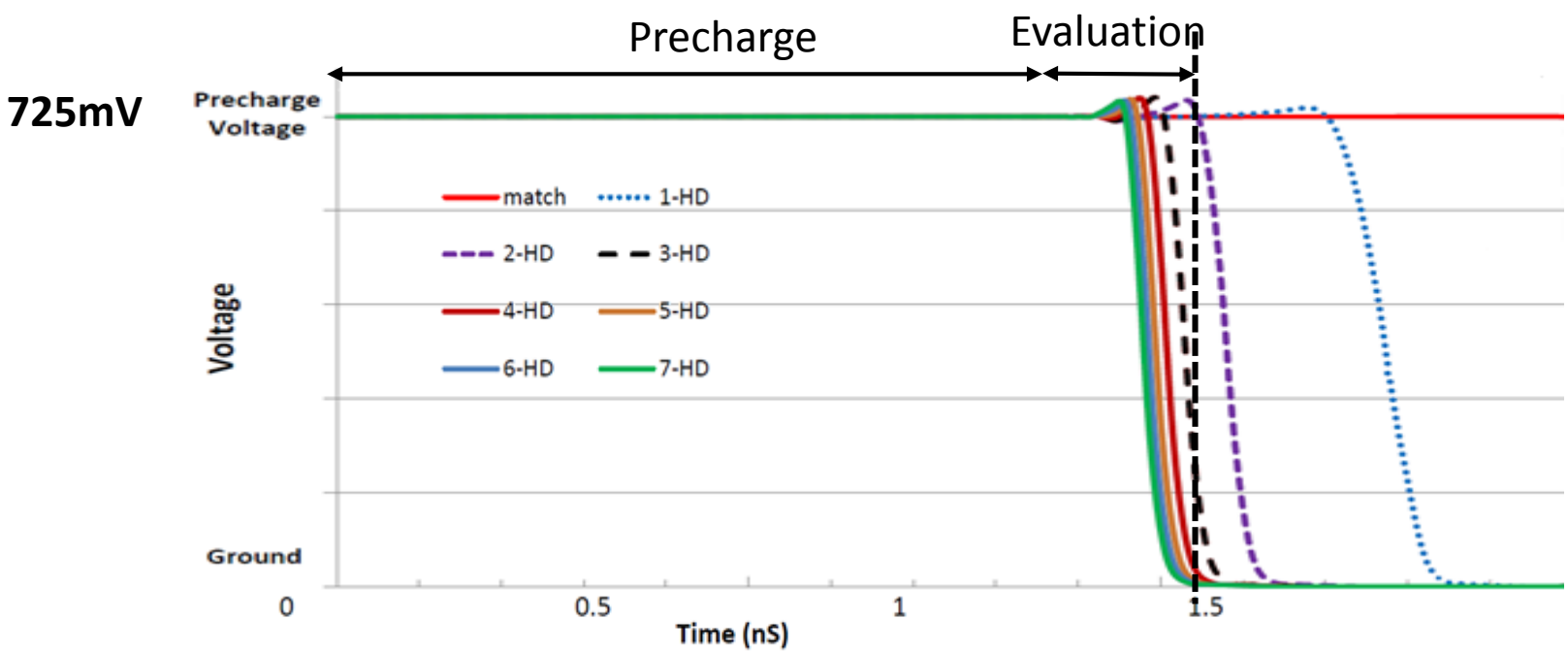
Approximate Search Operation in TCAM

- Lower voltages subject to *controllable* approximate matching:
  - 1000mV: Exact Matching
  - 775mV: 1-HD Matching
  - 720mV: 2-HD Matching



Approximate Search Operation in TCAM

- Lower voltages subject to *controllable* approximate matching:
  - 1000mV: Exact Matching
  - 775mV: 1-HD Matching
  - 720mV: 2-HD Matching



Visual Depiction of Output Quality

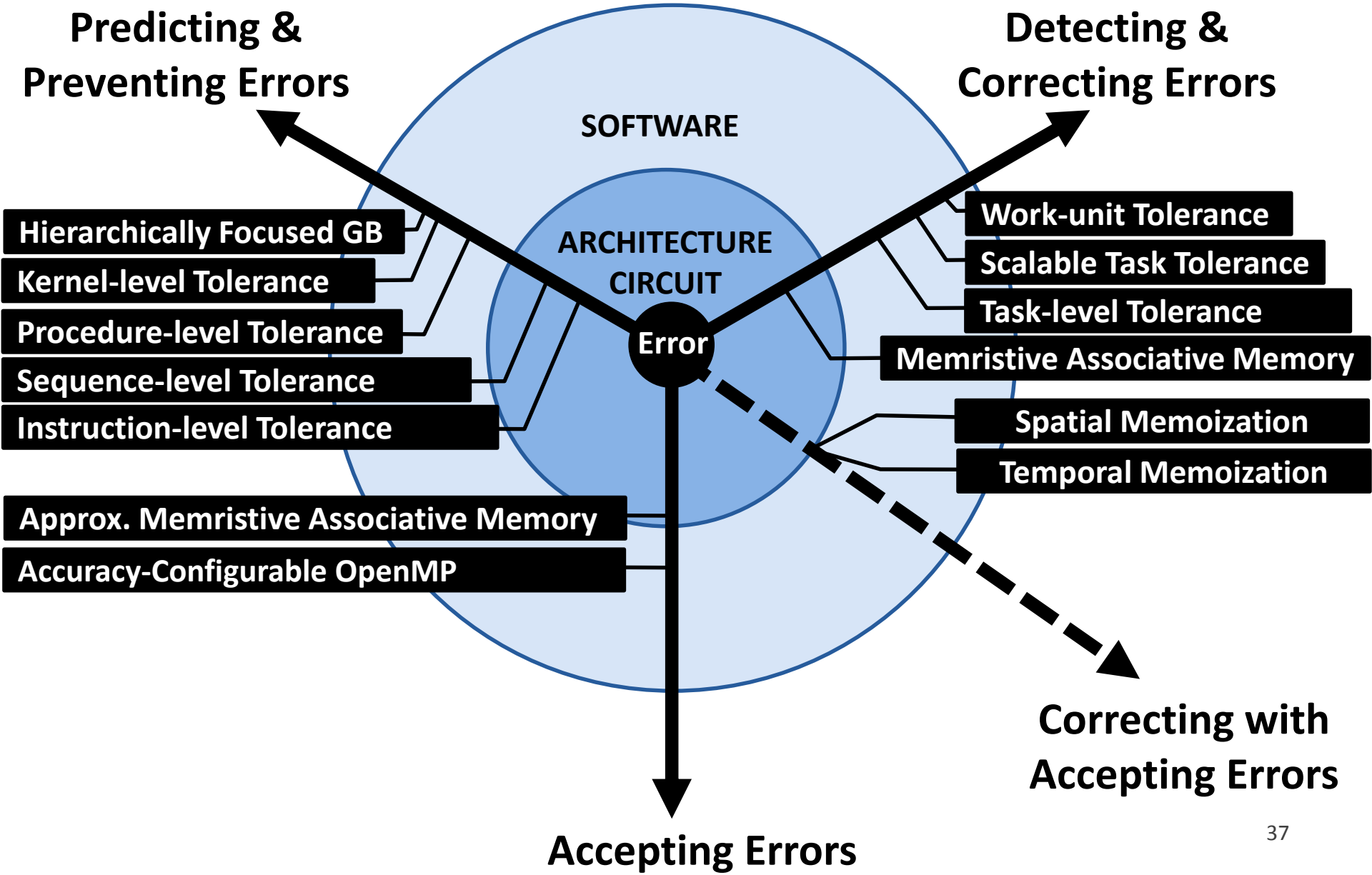
Exact matching
No noise

1-HD approximate matching
PSNR=61dB

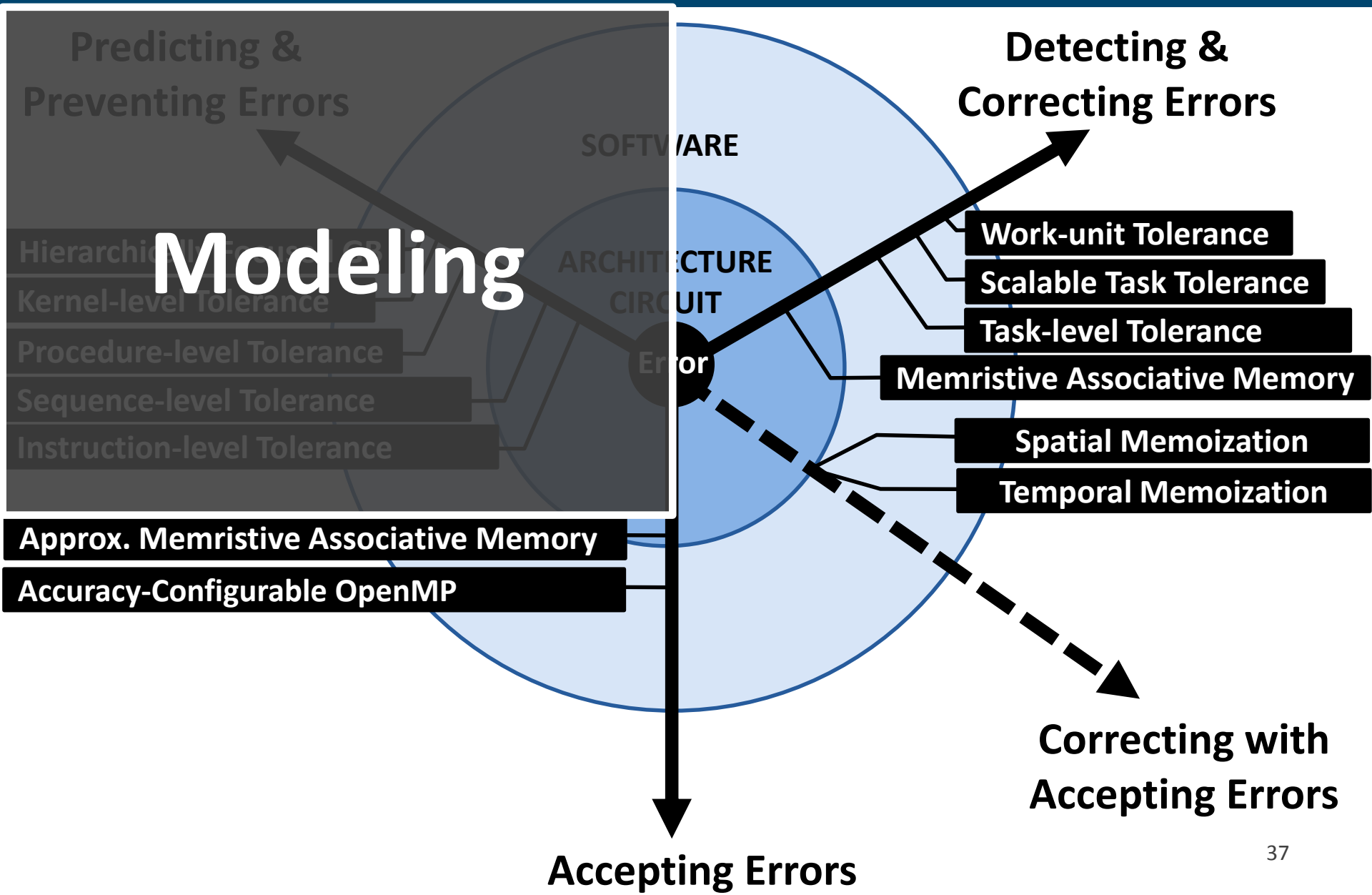
2-HD approximate matching
PSNR=42dB

- **AMD GPUs**
  - **Image processing applications from AMD APP SDK v2.5**
  - **Multi2Sim for AMD Southern Islands GPU, Radeon HD 7970 device**
- **FPU ASIC flow**
  - **6-stage balanced FPUs in 45-nm TSMC**
- **Quality loss is nearly indiscernible to the eye (PSNR>30dB)**
- **On average 32% energy saving for 4 kernels**

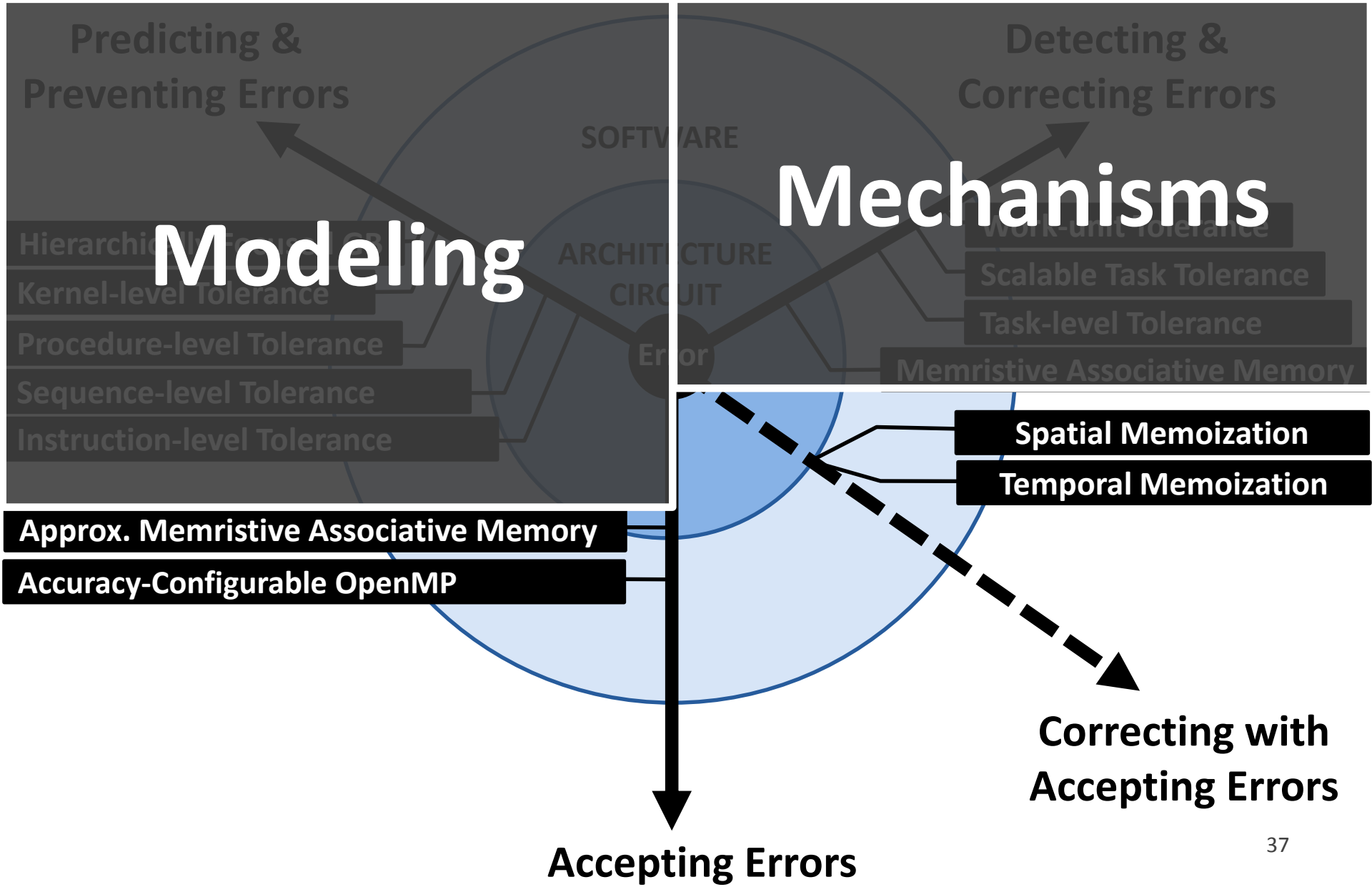
Recap


Predicting & Preventing Errors
Detecting & Correcting Errors
SOFTWARE
ARCHITECTURE
CIRCUIT
Error
Hierarchically Focused GB
Kernel-level Tolerance
Procedure-level Tolerance
Sequence-level Tolerance
Instruction-level Tolerance
Approx. Memristive Associative Memory
Accuracy-Configurable OpenMP
Work-unit Tolerance
Scalable Task Tolerance
Task-level Tolerance
Memristive Associative Memory
Spatial Memoization
Temporal Memoization
Correcting with Accepting Errors
Accepting Errors

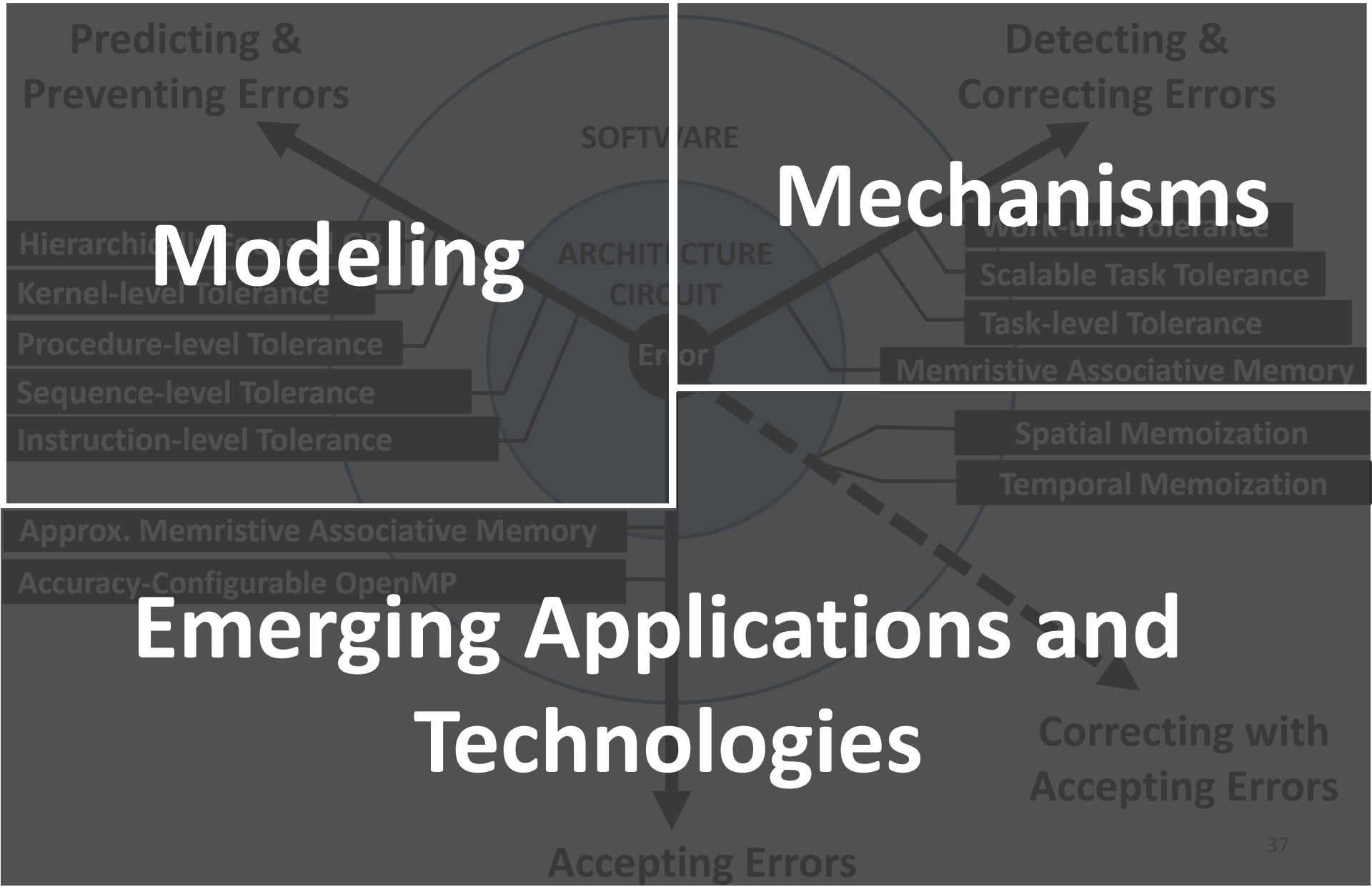
Recap


Predicting &
Preventing Errors
Modeling
Hierarchical Workload Characterization
Kernel-level Tolerance
Procedure-level Tolerance
Sequence-level Tolerance
Instruction-level Tolerance
SOFTWARE
ARCHITECTURE
CIRCUIT
Error
Approx. Memristive Associative Memory
Accuracy-Configurable OpenMP
Detecting &
Correcting Errors
Work-unit Tolerance
Scalable Task Tolerance
Task-level Tolerance
Memristive Associative Memory
Spatial Memoization
Temporal Memoization
Correcting with
Accepting Errors
Accepting Errors

Recap


Predicting &
Preventing Errors
Modeling
Kernel-level Tolerance
Procedure-level Tolerance
Sequence-level Tolerance
Instruction-level Tolerance
Detecting &
Correcting Errors
Mechanisms
Scalable Task Tolerance
Task-level Tolerance
Memristive Associative Memory
Spatial Memoization
Temporal Memoization
Approx. Memristive Associative Memory
Accuracy-Configurable OpenMP
Correcting with
Accepting Errors
Accepting Errors

Recap


Predicting &
Preventing Errors
Detecting &
Correcting Errors
SOFTWARE
ARCHITECTURE
CIRCUIT
Error
Modeling
Mechanisms
Kernel-level Tolerance
Procedure-level Tolerance
Sequence-level Tolerance
Instruction-level Tolerance
Scalable Task Tolerance
Task-level Tolerance
Memristive Associative Memory
Spatial Memoization
Temporal Memoization
Approx. Memristive Associative Memory
Accuracy-Configurable OpenMP
Emerging Applications and
Technologies
Correcting with
Accepting Errors
Accepting Errors

Putting it Together

Sequence-level Tolerance [*TC'13*]

Instruction-level Tolerance [*DATE'12*]

Single-Core

Putting it Together

Accuracy-Configurable OpenMP [*CODES'13*]

Work-unit Tolerance [*JETCAS'14*]

Accelerated Task Mapping [*ISLPED'14*]

Scalable Task-level Tolerance [*DAC'15*]

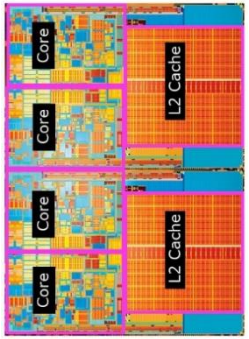
Sequence-level Tolerance [*TC'13*]

Task-level Tolerance [*DATE'13*]

Instruction-level Tolerance [*DATE'12*]

Procedure-level Tolerance [*ISLPED'12*]

Single-Core

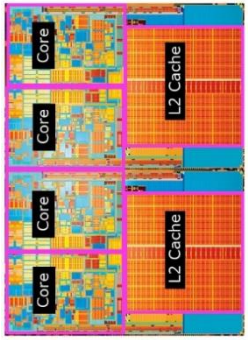


Multi-Core

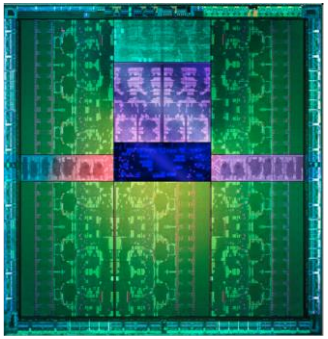
Putting it Together

	Accuracy-Configurable OpenMP [*CODES'13*]	Approximate Memristor Asso. Memory [*DATE'15*]
	Work-unit Tolerance [*JETCAS'14*]	Memristor Associative Memory [*DAC'14*]
	Accelerated Task Mapping [*ISLPED'14*]	Temporal Memoization [*DATE'14*]
	Scalable Task-level Tolerance [*DAC'15*]	Spatial Memoization [*TCAS-II'13*]
Sequence-level Tolerance [*TC'13*]	Task-level Tolerance [*DATE'13*]	Kernel-level Tolerance [*DAC'13, CODES'13*]
Instruction-level Tolerance [*DATE'12*]	Procedure-level Tolerance [*ISLPED'12*]	Hierarchically Focused GB [*DATE'13*]

Single-Core



Multi-Core



GPUs

Putting it Together

Accuracy-Configurable OpenMP [*CODES'13*]

Work-unit Tolerance [*JETCAS'14*]

Accelerated Task Mapping [*ISLPED'14*]

Scalable Task-level Tolerance [*DAC'15*]

Sequence-level Tolerance [*TC'13*]

Task-level Tolerance [*DATE'13*]

Instruction-level Tolerance [*DATE'12*]

Procedure-level Tolerance [*ISLPED'12*]

Approximate Memristor Asso. Memory [*DATE'15*]

Memristor Associative Memory [*DAC'14*]

Temporal Memoization [*DATE'14*]

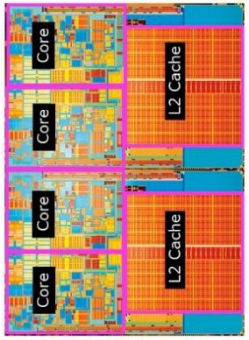
Spatial Memoization [*TCAS-II'13*]

Kernel-level Tolerance [*DAC'13, CODES'13*]

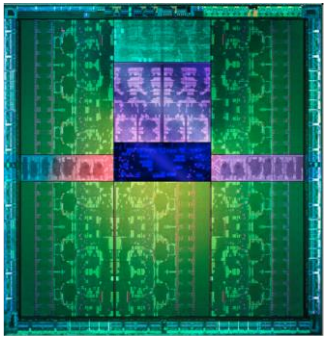
Hierarchically Focused GB [*DATE'13*]

Approximate kernels on contemporary FPGAs

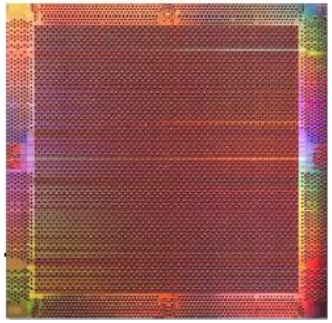
Single-Core



Multi-Core



GPUs



FPGAs

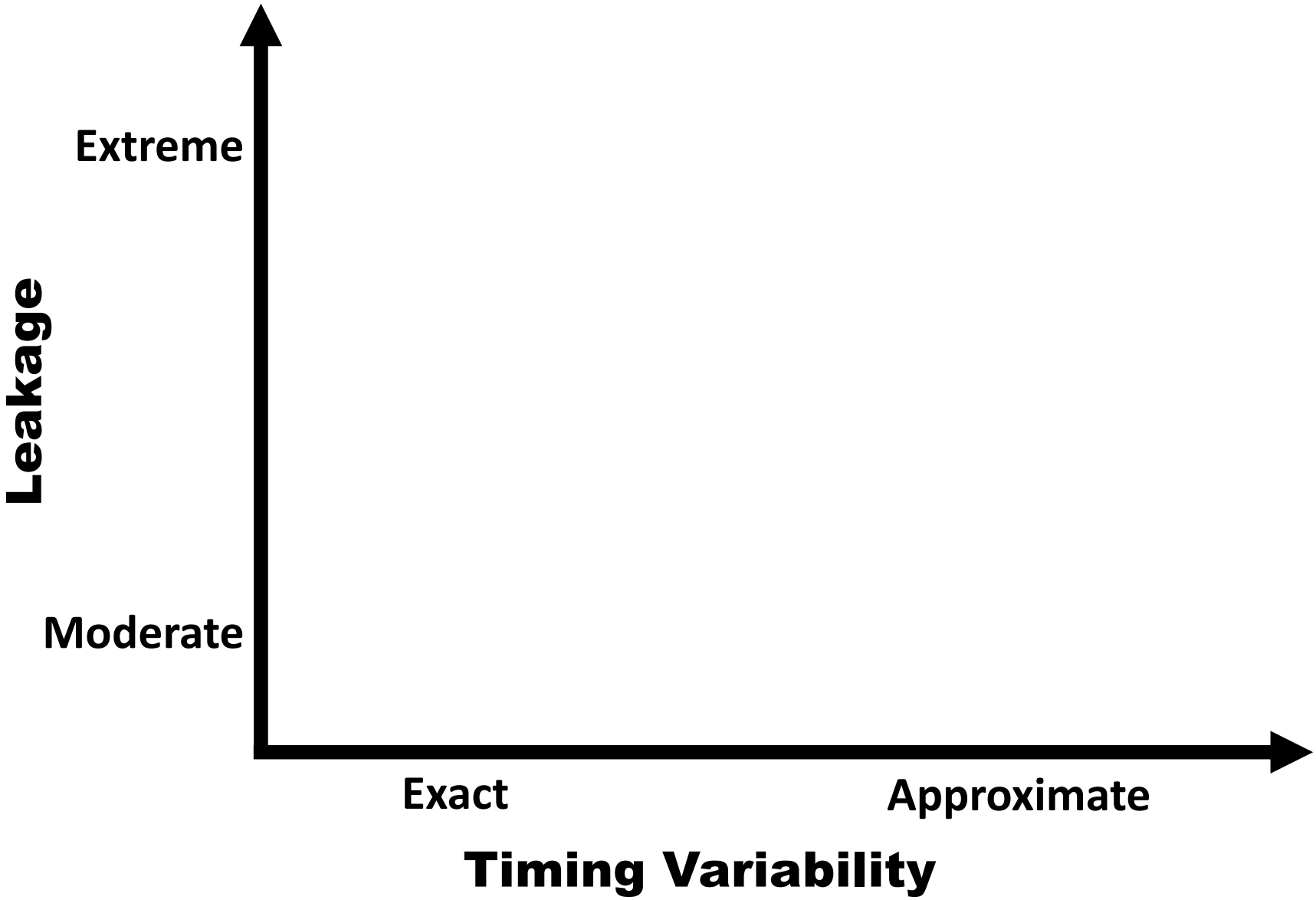
Summary

- Semiconductor field is changing:
  - Variations causes timing errors and increased cost

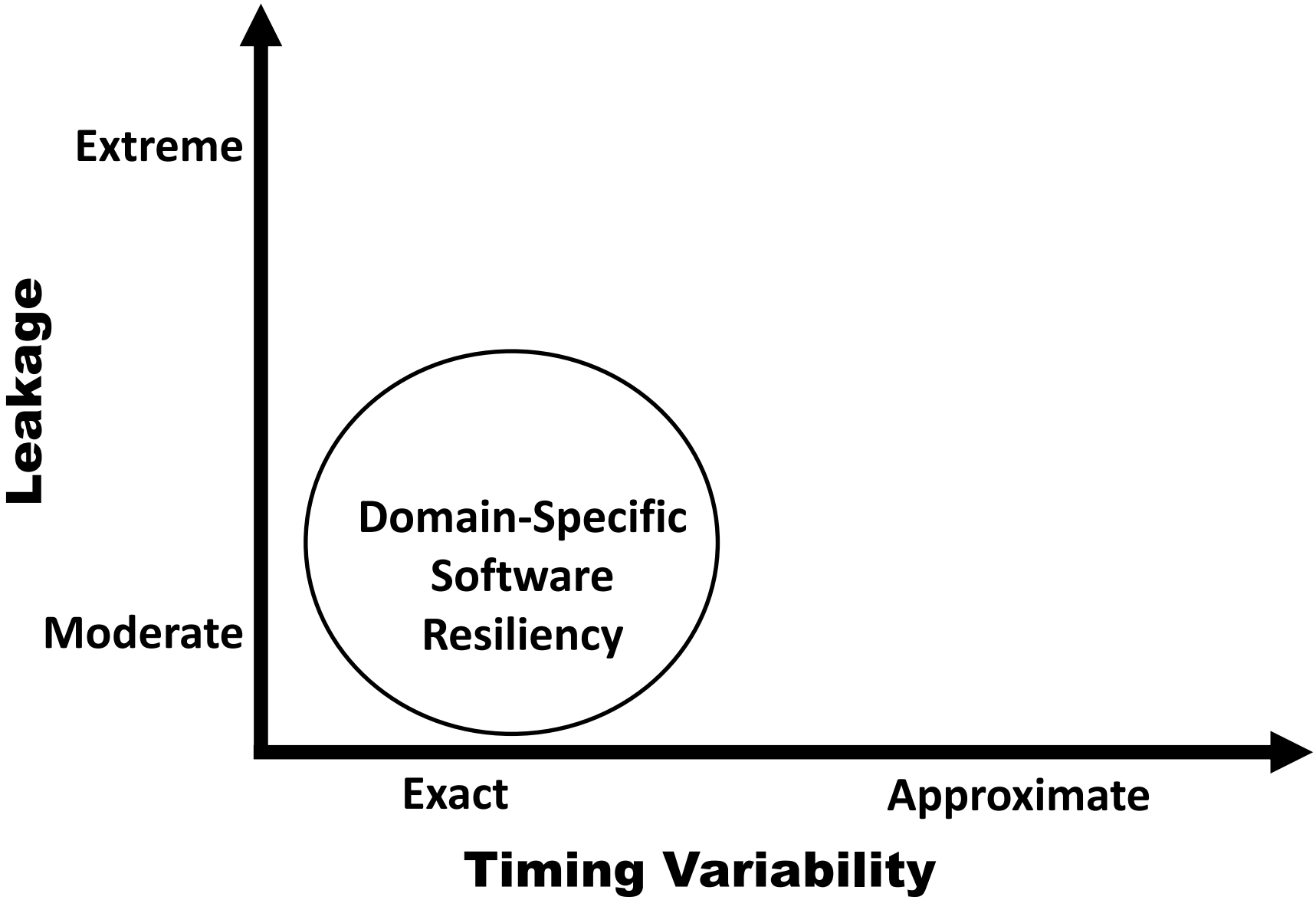
- We reduce the guardband and mitigate timing errors by
  - ➢ exposing hardware variations to the software
  - ➢ exploiting parallel processing
- Naturally enables approximate computing!

- Spatiotemporal locality/similarity, computing with memristor memories, resilient software techniques and transformations are our friends!

Vision

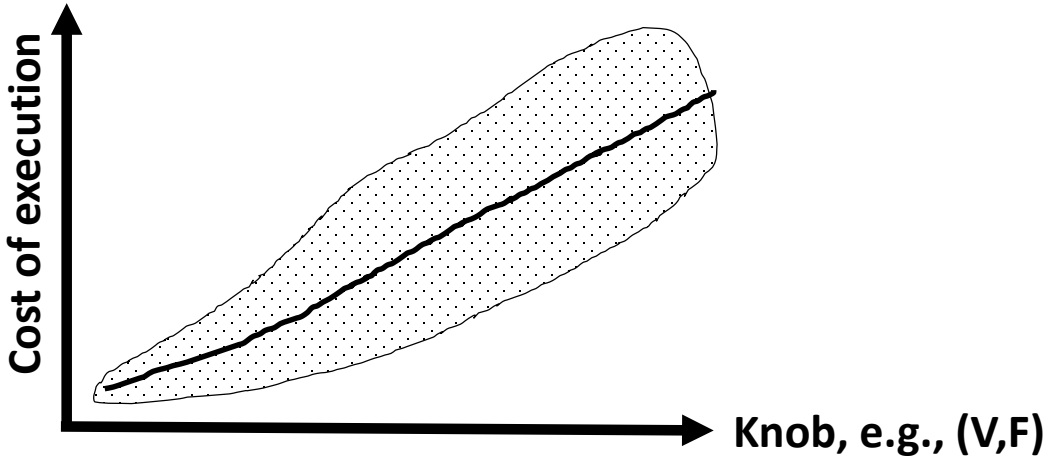

Extreme
Leakage
Moderate
Exact
Approximate
Timing Variability

Vision


Extreme
Leakage
Moderate
Domain-Specific
Software
Resiliency
Exact
Approximate
Timing Variability

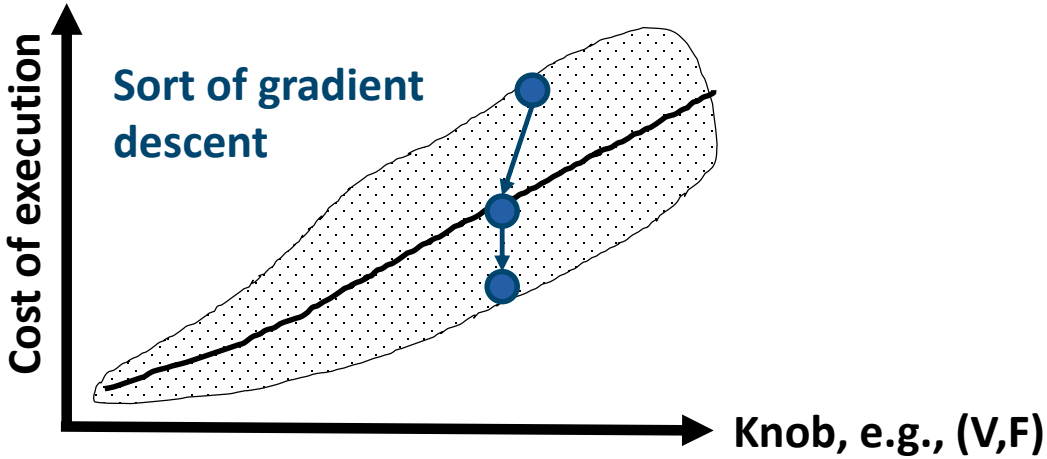
Vision: Domain-Specific Software Resiliency

- Software needs ***runtime monitoring + recalibration***
- Approach to the edge-of-failure but not on the other side of border→ track the "moving" critical point
- Introspective software: sibling clairvoyant, sprinting, etc.



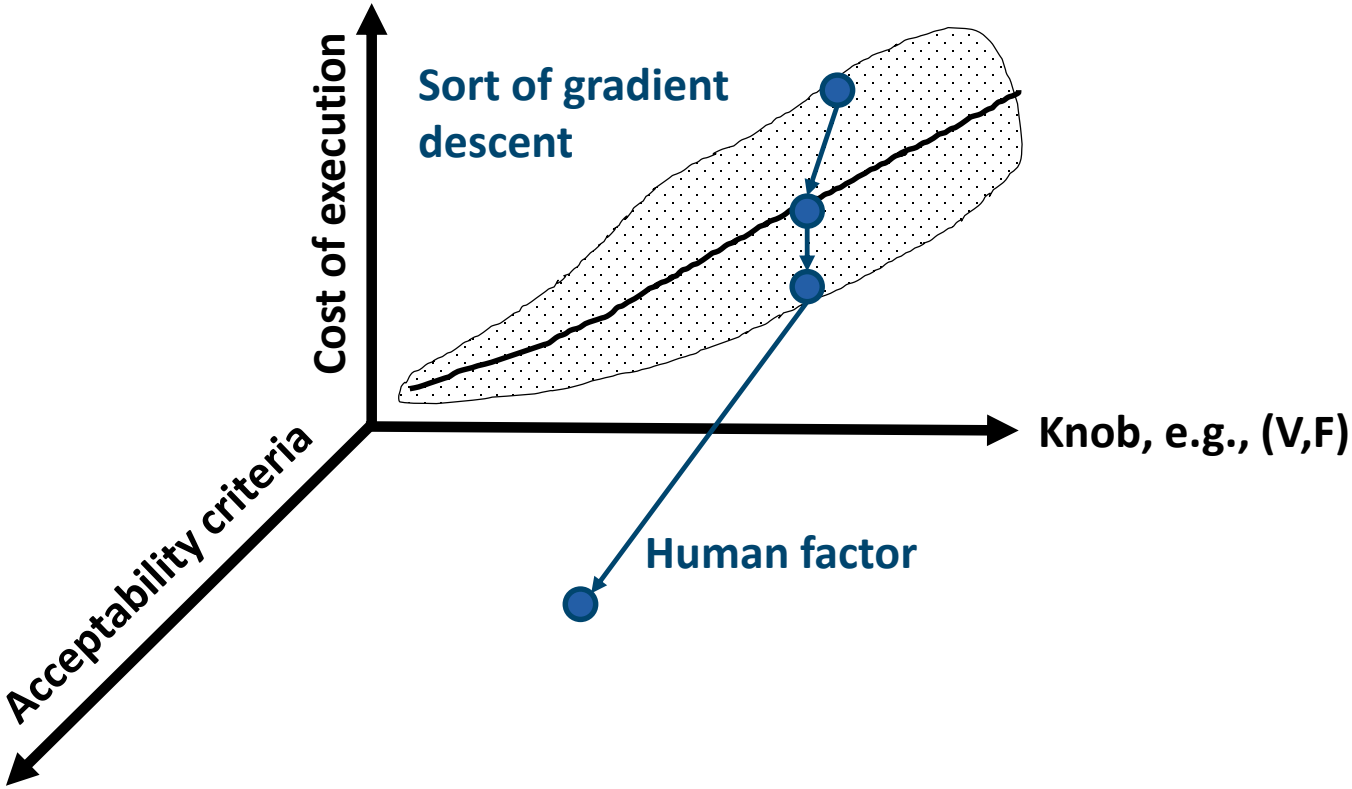
Vision: Domain-Specific Software Resiliency

- Software needs ***runtime monitoring + recalibration***
- Approach to the edge-of-failure but not on the other side of border→ track the "moving" critical point
- Introspective software: sibling clairvoyant, sprinting, etc.

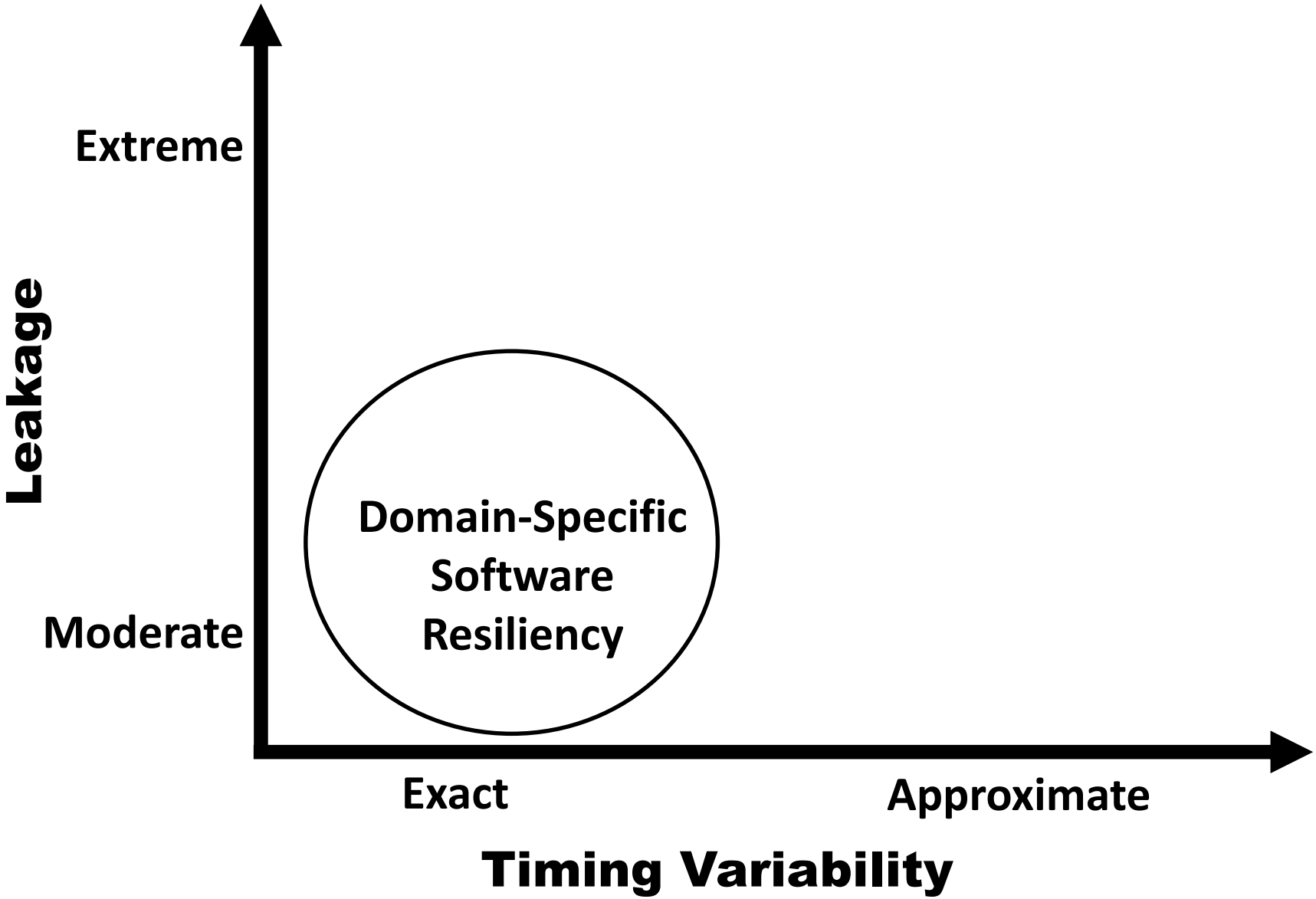


Vision: Domain-Specific Software Resiliency

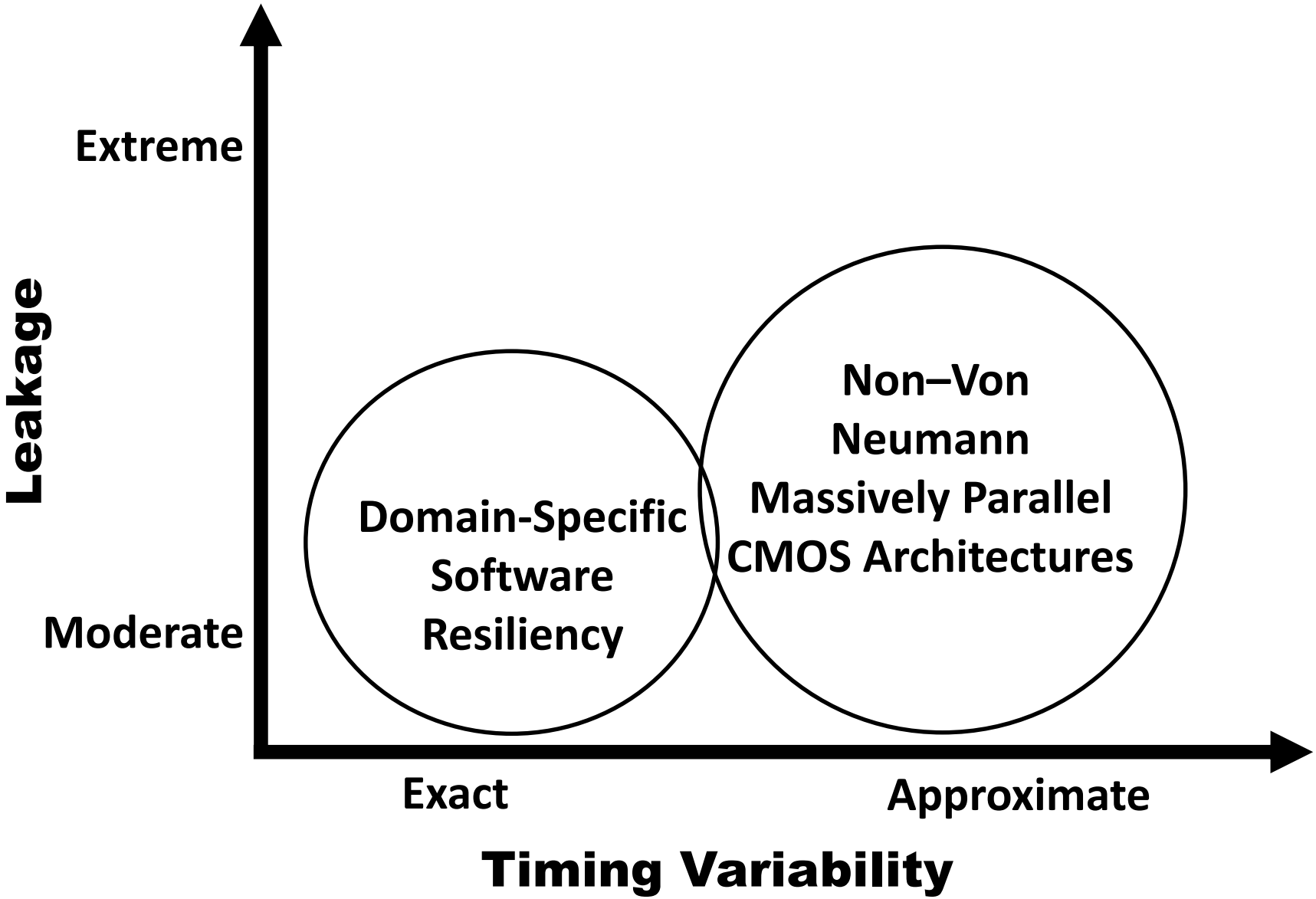
- Software needs ***runtime monitoring + recalibration***
- Approach to the edge-of-failure but not on the other side of border→ track the "moving" critical point
- Introspective software: sibling clairvoyant, sprinting, etc.



Vision


Extreme
Leakage
Moderate
Domain-Specific
Software
Resiliency
Exact
Approximate
Timing Variability

Vision


Extreme
Leakage
Moderate
Domain-Specific
Software
Resiliency
Non–Von
Neumann
Massively Parallel
CMOS Architectures
Exact
Approximate
Timing Variability

Vision: Non–Von Neumann Massively Parallel CMOS Arch.

Lessen learned:

Why Non-Von Neumann?

☹ Such a bad sequence of: Fetch→ Decode→ *EXE* → WB

☹ We can only relax *EXE*

☹ Fine-grained is expensive!

☹ Increased complexity

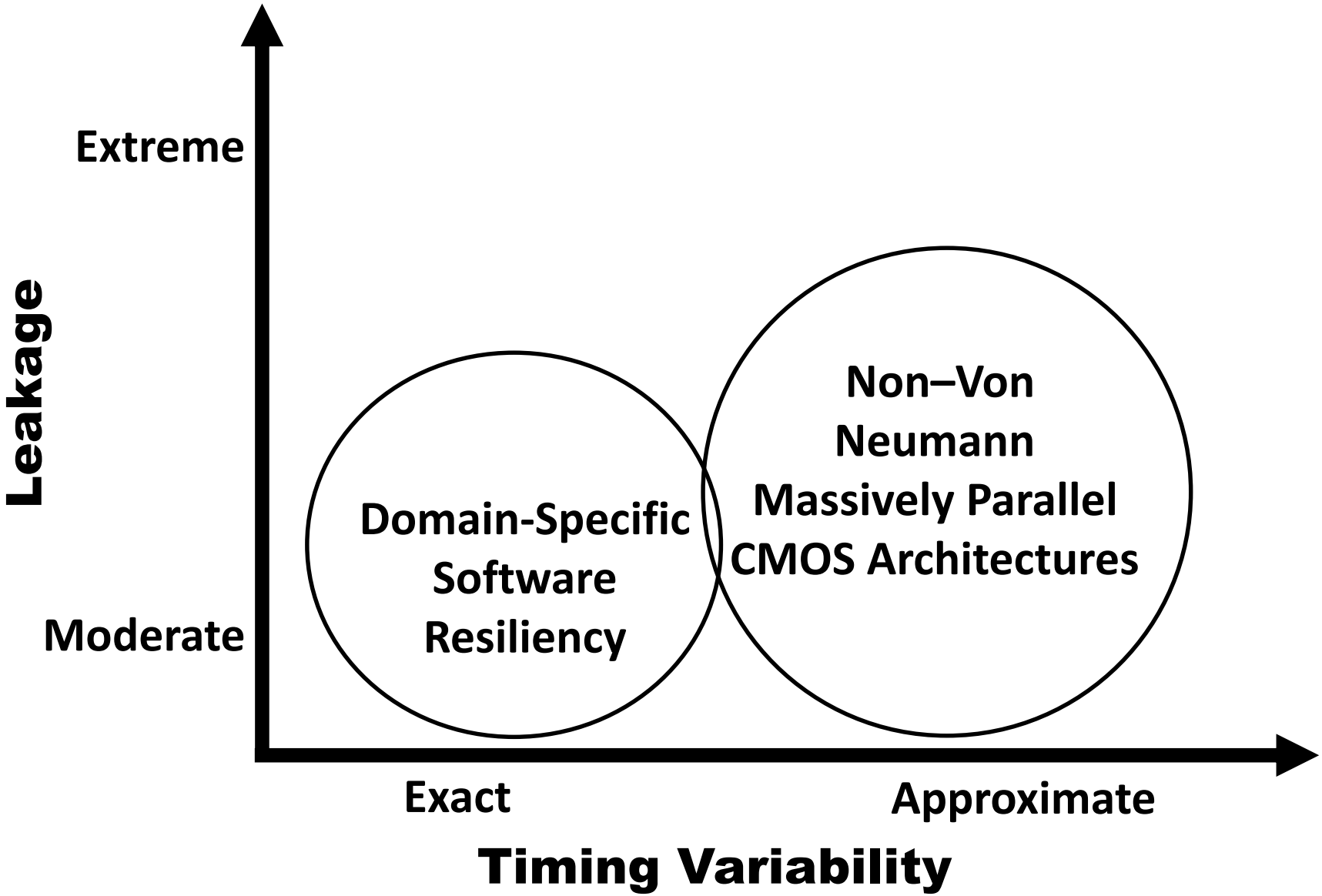
😐 VOS: the pain is more than the gain → QDI circuits?

Why Parallel?

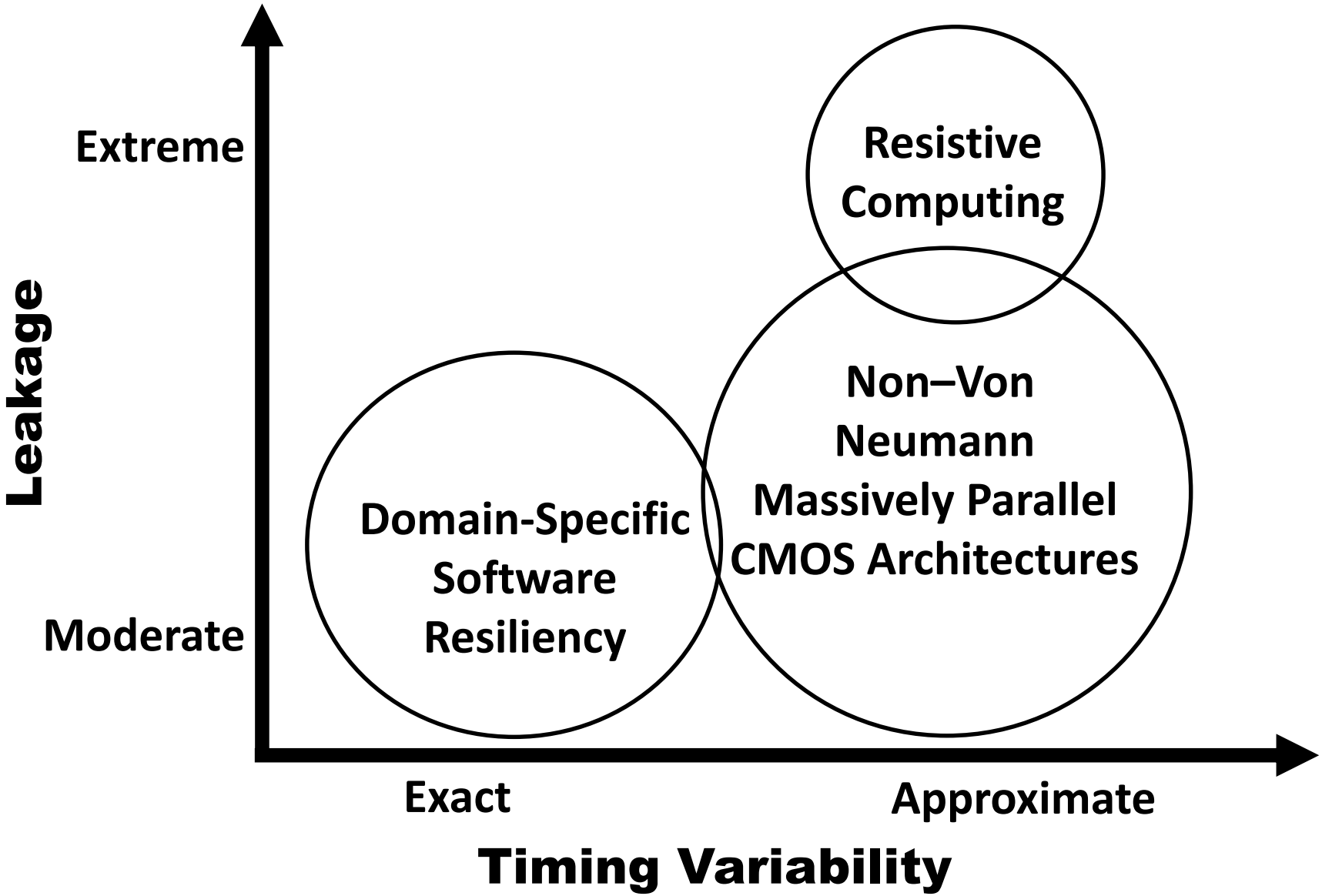
☺ Parallelism provides the best means to exploit variability!

☺ Fast, and highly scalable methods → new robust data representation (sparse and distributed)

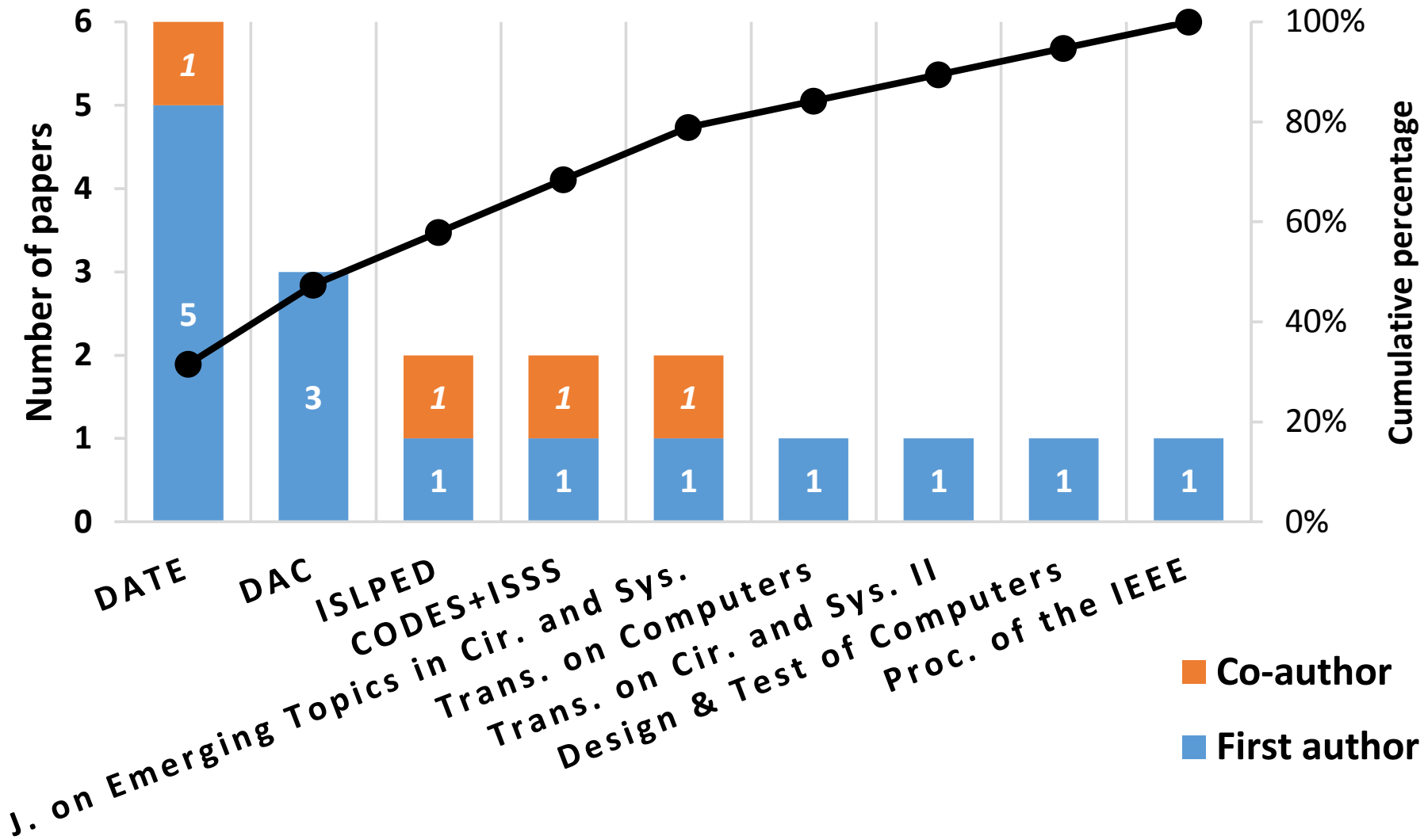
Vision


Extreme
Moderate
Leakage
Domain-Specific
Software
Resiliency
Non–Von
Neumann
Massively Parallel
CMOS Architectures
Exact
Approximate
Timing Variability

Vision


Extreme
Moderate
Leakage
Resistive Computing
Non–Von Neumann Massively Parallel CMOS Architectures
Domain-Specific Software Resiliency
Exact
Approximate
Timing Variability

My Favorite Pareto Chart: Publications



A Wonderful Quote

"Infinite patience produces immediate results."

A Course in Miracles
By Helen Schucman, 1976
Foundation for Inner Peace

A Wonderful Quote

Enough acceptable

"~~Infinite~~ patience produces ~~immediate~~ results."

A Course in Miracles

By Helen Schucman, 1976

Foundation for Inner Peace

Thank you so much!

The fun is just beginning :-)