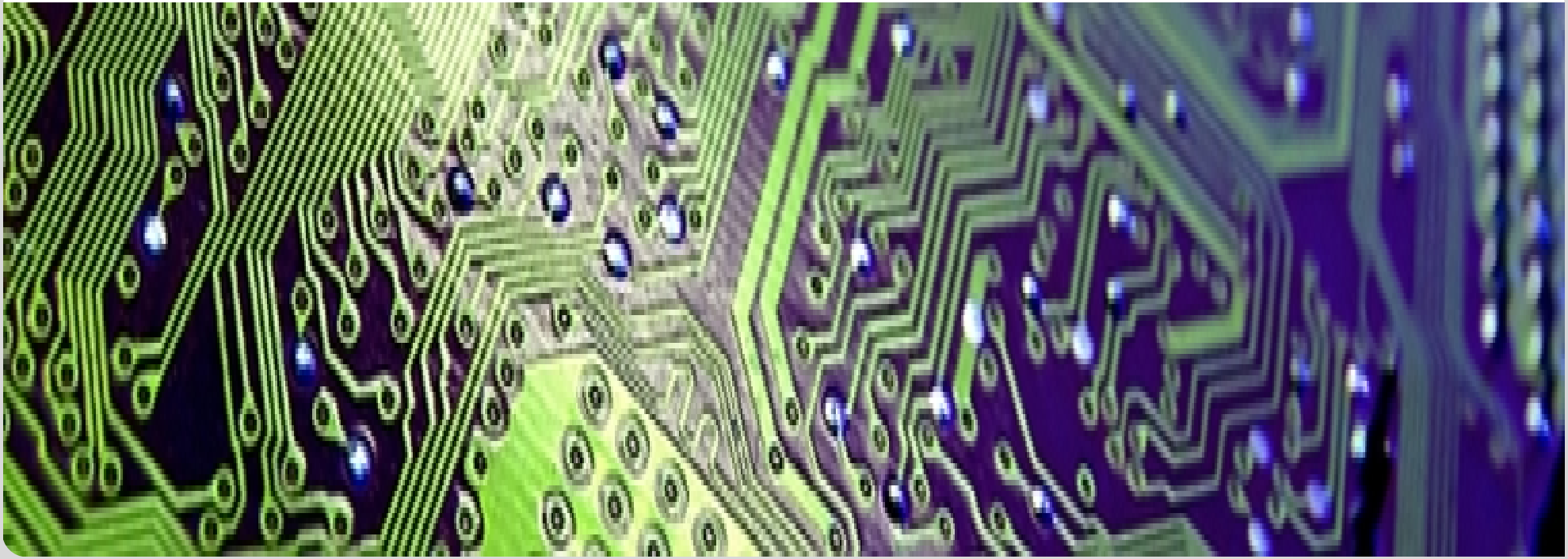
ArISE: Aging-Aware Instruction Set Encoding for Lifetime Improvement

F. Oboril and M. Tahoori | January 21, 2014

INSTITUTE OF COMPUTER ENGINEERING (ITEC) – CHAIR FOR DEPENDABLE NANO COMPUTING (CDNC)



Purpose of this Work

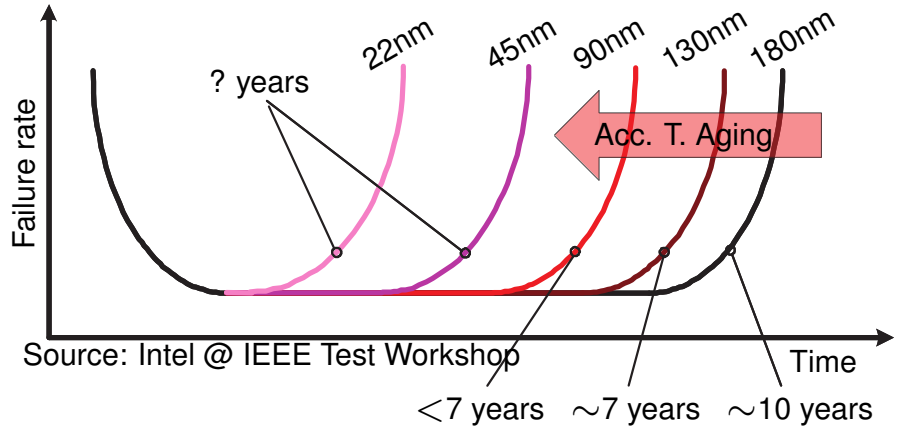
Nanoscale Reliability Challenge

Transistor aging increases device delays during runtime

Purpose of this Work

Nanoscale Reliability Challenge

Transistor aging increases device delays during runtime

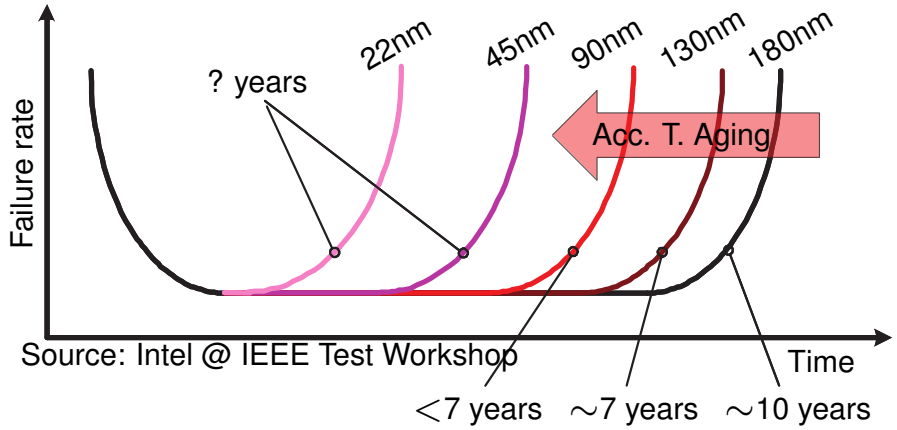


Source: Intel @ IEEE Test Workshop

Purpose of this Work

Nanoscale Reliability Challenge

Transistor aging increases device delays during runtime



Source: Intel @ IEEE Test Workshop

Problem Statement

Instruction opcode significantly impacts lifetime of various pipeline stages

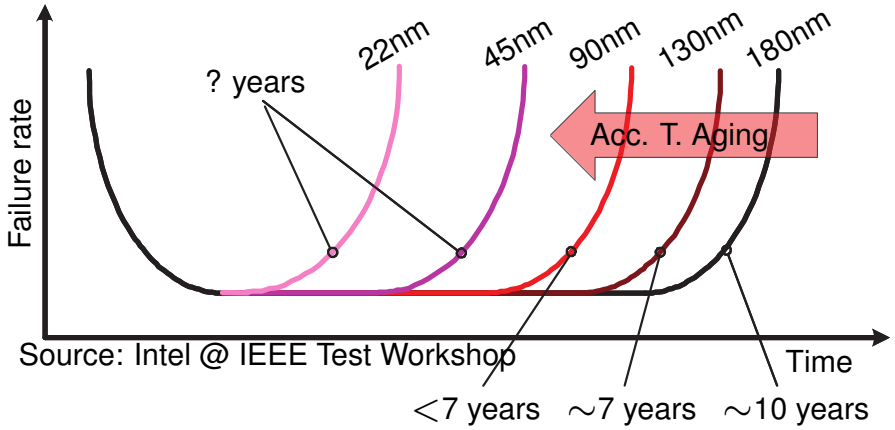
$\Rightarrow$ Optimization potential: Performance, area, power and MTTF[1]

[1]: Mean Time To Failure (MTTF) := Time to first timing violation due to aging

Purpose of this Work

Nanoscale Reliability Challenge

Transistor aging increases device delays during runtime



Source: Intel @ IEEE Test Workshop

Problem Statement

Instruction opcode significantly impacts lifetime of various pipeline stages
$\Rightarrow$ Optimization potential: Performance, area, power and MTTF[1]

Proposed Solution

Aging-Awa**r**e **I**nstruction **S**et **E**ncoding: ArISE

[1]: Mean Time To Failure (MTTF) := Time to first timing violation due to aging

Outline

Outline

Preliminaries: Terminology

- Opcode

$\rightarrow$ Binary representation of an instruction

e.g. ADD $\rightarrow$ 00011101

- Instruction Set Encoding (ISE):

$\rightarrow$ Mapping of instructions to their opcodes

e.g.

$$\begin{array}{lcl} BNE & \rightarrow & 02 \\ LW & \rightarrow & c5 \\ SB & \rightarrow & a3 \\ ADD & \rightarrow & 1d \\ OR & \rightarrow & 2a \\ JMP & \rightarrow & 05 \end{array}$$

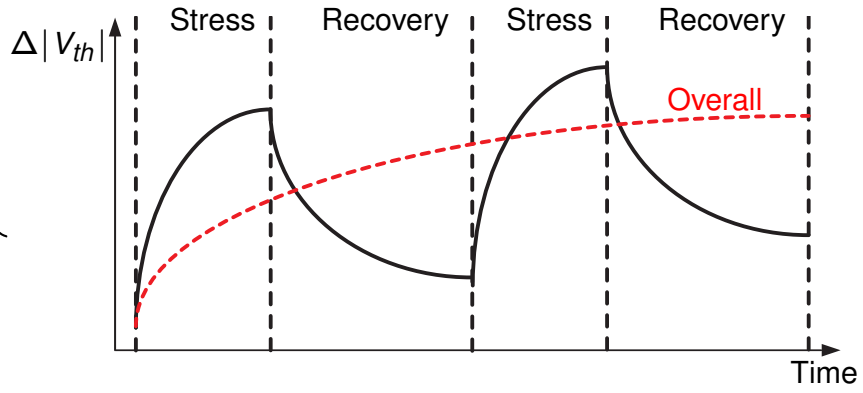
Preliminaries: Transistor Aging

Transistor Aging is caused by:

Preliminaries: Transistor Aging

Transistor Aging is caused by:

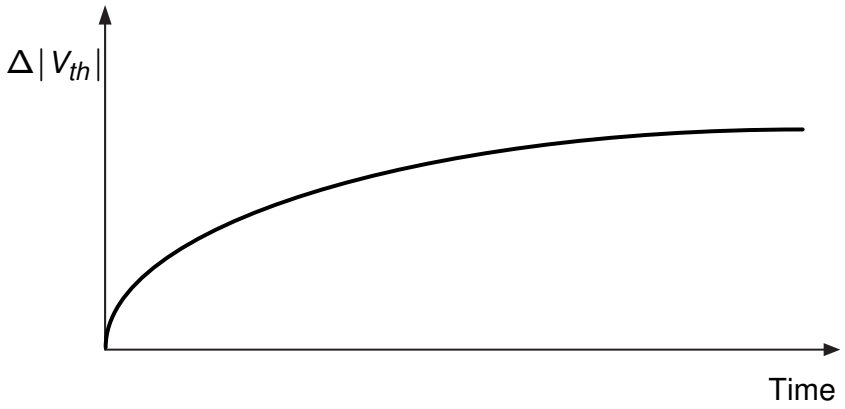
- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$



Preliminaries: Transistor Aging

Transistor Aging is caused by:

- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$
- Hot Carrier Injection (HCI)
  - Affects mainly NMOS: $|V_{th}| \uparrow$



Preliminaries: Transistor Aging

Transistor Aging is caused by:

- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$
- Hot Carrier Injection (HCI)
  - Affects mainly NMOS: $|V_{th}| \uparrow$

$\Rightarrow$ Path delays increase over time $\Rightarrow$ Pipeline stage delays increase

Preliminaries: Transistor Aging

Transistor Aging is caused by:

- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$
- Hot Carrier Injection (HCI)
  - Affects mainly NMOS: $|V_{th}| \uparrow$

$\Rightarrow$ Path delays increase over time $\Rightarrow$ Pipeline stage delays increase

1st order factors on BTI and HCI:

		BTI	HCI
Temperature	(T)	exponential	exponential
Frequency	(f)	-	sublinear
Voltage	(V_{dd})	exponential	exponential
Exec. Time	(t)	sublinear	sublinear
Usage		sublinear	sublinear

Preliminaries: Transistor Aging

Transistor Aging is caused by:

- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$
- Hot Carrier Injection (HCI)
  - Affects mainly NMOS: $|V_{th}| \uparrow$

$\Rightarrow$ Path delays increase over time $\Rightarrow$ Pipeline stage delays increase

1st order factors on BTI and HCI:

Depends on stage inputs ($\rightarrow$ opcode)

		BTI	HCI
Temperature	(T)	exponential	exponential
Frequency	(f)	-	sublinear
Voltage	(V_{dd})	exponential	exponential
Exec. Time	(t)	sublinear	sublinear
Usage		sublinear	sublinear

Preliminaries: Transistor Aging

Transistor Aging is caused by:

- Bias Temperature Instability (BTI)
  - Negative BTI (NBTI): PMOS' $|V_{th}| \uparrow$
  - Positive BTI (PBTI): NMOS' $|V_{th}| \uparrow$
- Hot Carrier Injection (HCI)
  - Affects mainly NMOS: $|V_{th}| \uparrow$

$\Rightarrow$ Path delays increase over time $\Rightarrow$ Pipeline stage delays increase

1st order factors on BTI and HCI:

Depends on stage inputs ($\rightarrow$ opcode)

		BTI	HCI
Temperature	(T)	exponential	exponential
Frequency	(f)	-	sublinear
Voltage	(V_{dd})	exponential	exponential
Exec. Time	(t)	sublinear	sublinear
Usage		sublinear	sublinear

$\Rightarrow$ Delay increase depends on the instruction opcodes

Outline

Motivation

Question: How much influence has the $\underbrace{\text{instruction set encoding (ISE)}}$?

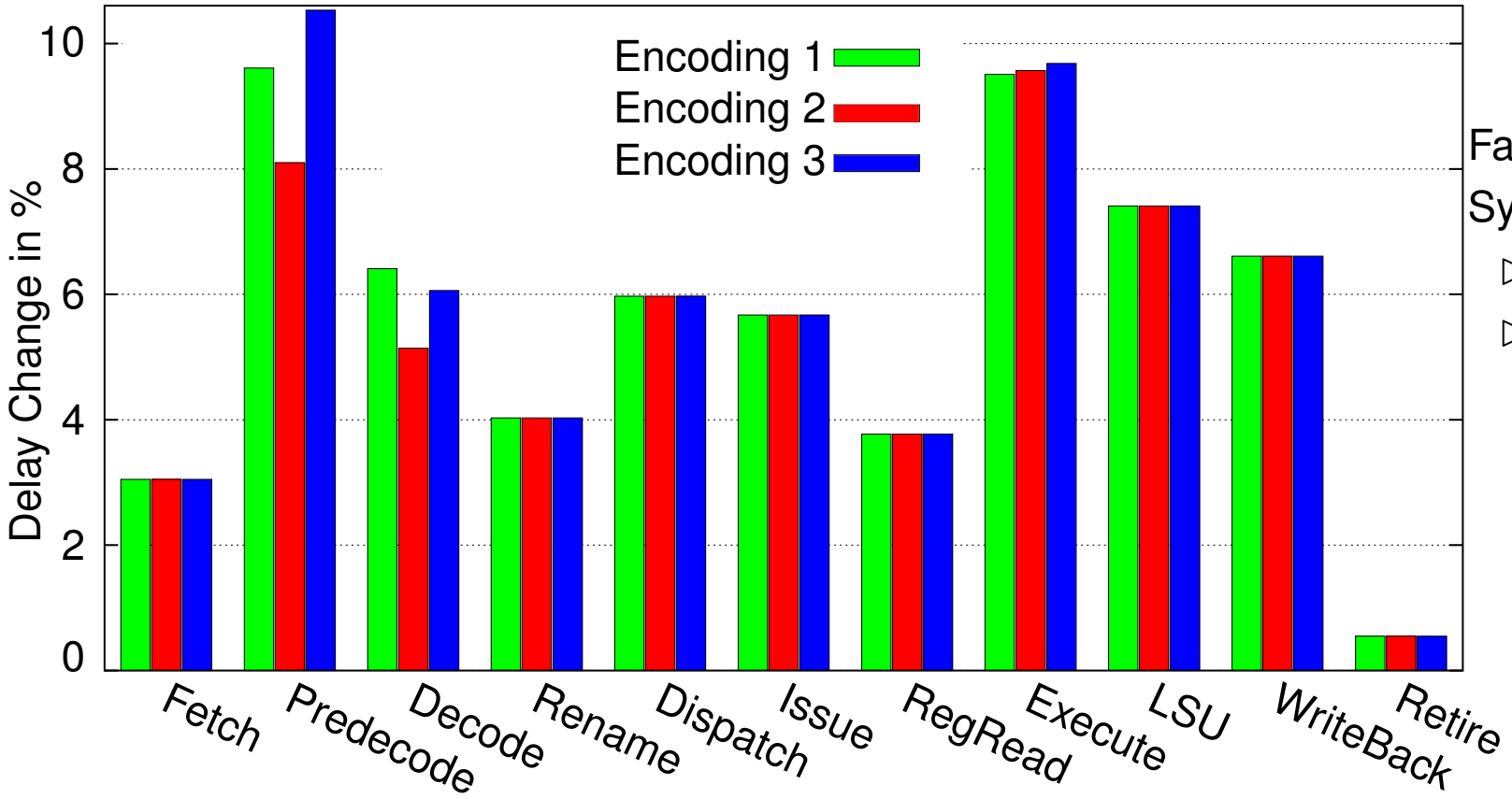
mapping of instructions to their opcodes

$\downarrow$

affects circuit design and inputs

Motivation

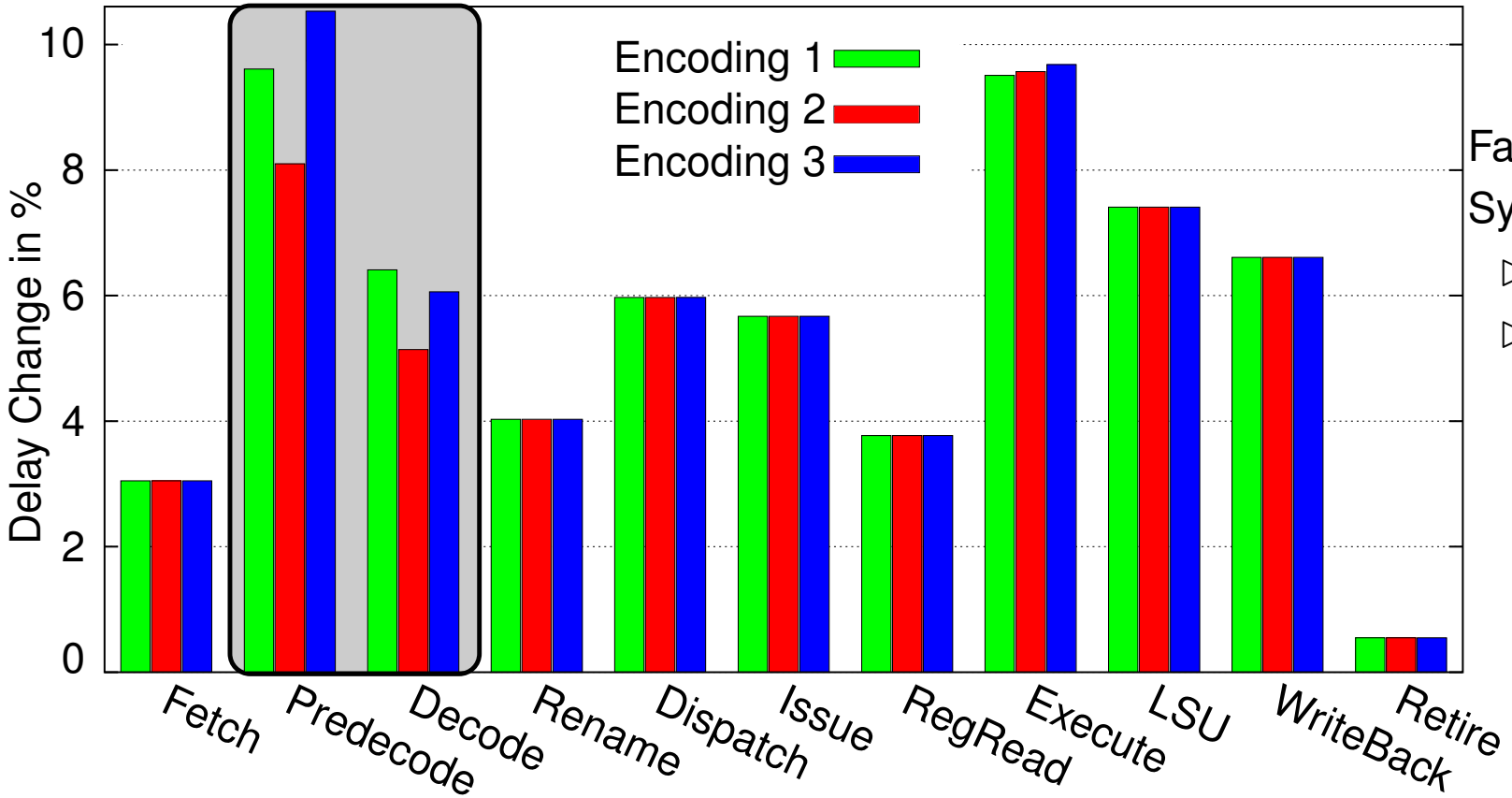
Question: How much influence has the instruction set encoding (ISE) ?



FabScalar
Synthesized with:
- ▷ Synopsys DC
- ▷ TSMC 65nm

Motivation

Question: How much influence has the instruction set encoding (ISE) ?



FabScalar
Synthesized with:
- ▷ Synopsys DC
- ▷ TSMC 65nm

Observations:

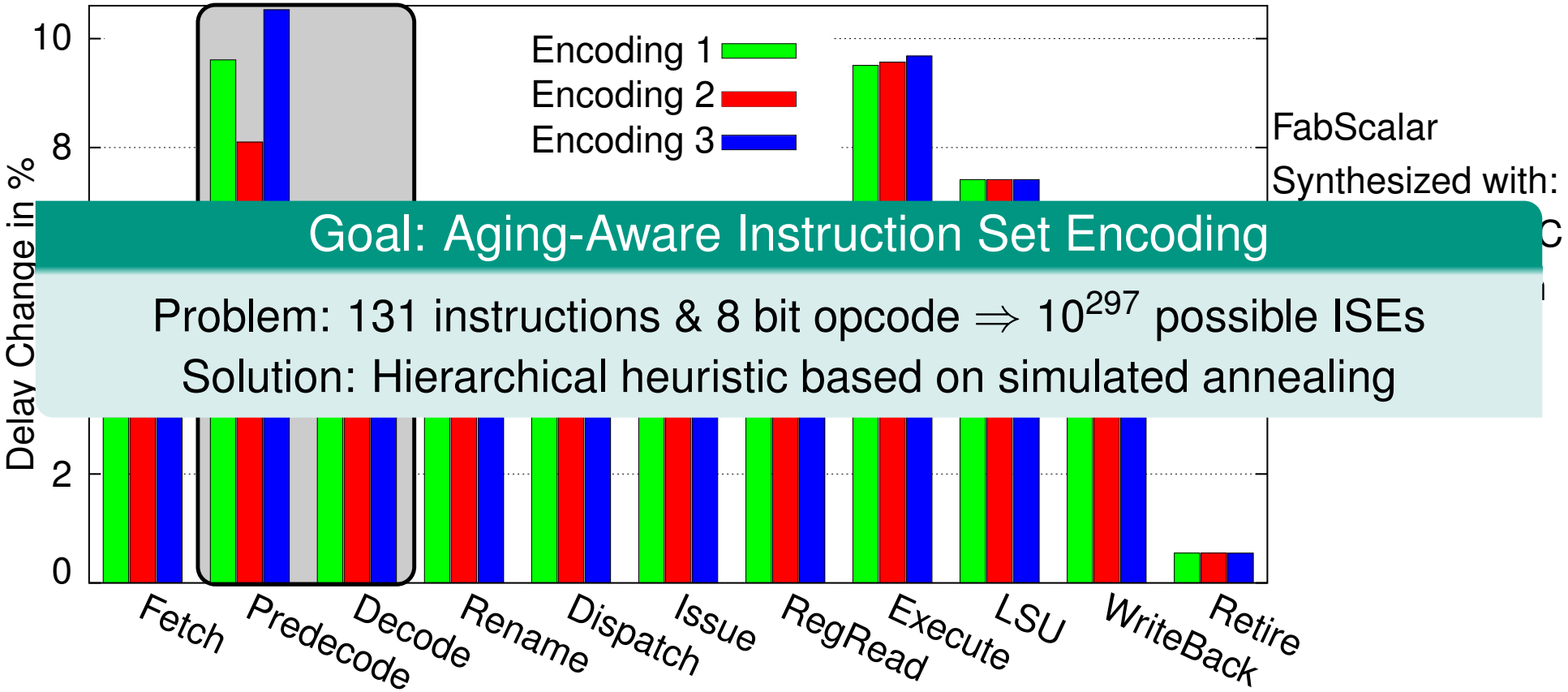
1. ISE has significant impact on decoding stages

→ Difference of 2 % or more than 2x in MTTF

2. Predecode can limit the entire microprocessor lifetime

Motivation

Question: How much influence has the instruction set encoding (ISE) ?



Goal: Aging-Aware Instruction Set Encoding

Problem: 131 instructions & 8 bit opcode $\Rightarrow$ 10^{297} possible ISEs

Solution: Hierarchical heuristic based on simulated annealing

Observations:

1. ISE has significant impact on decoding stages

$\rightarrow$ Difference of 2 % or more than 2x in MTTF

2. Predecode can limit the entire microprocessor lifetime

Outline

Related Work

- Aging mitigation techniques at various design levels
  - $\rightarrow$ Orthogonal to this work
  - At (micro)-architecture-level focus is mostly on execution stage
    - For pipeline frontend [DeBole2009] proposed periodical opcode inversion

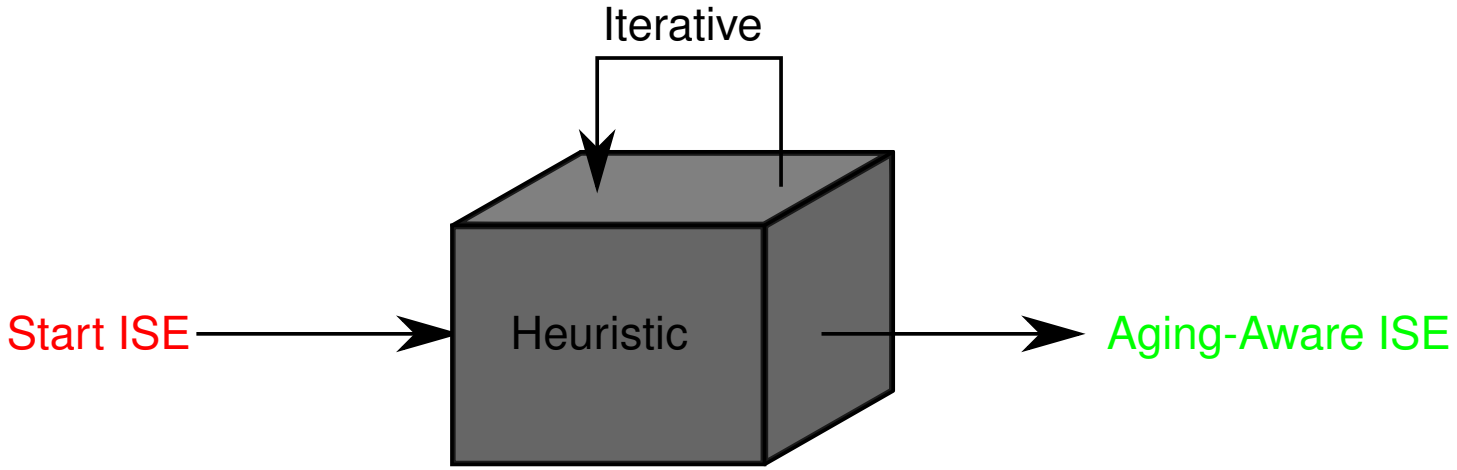
- Instruction Set Encoding
  - $\Rightarrow$ Well-known for energy reduction (e.g. reduce switching activity)
  - Most works focus on memories, e.g. instruction buffers

- Our work
  - Find best ISE in terms of lifetime
  - $\rightarrow$ Power or energy can be affected
  - More efficient than [DeBole2009] as “our” ISE is aging-aware

Outline

ArISE – Overview

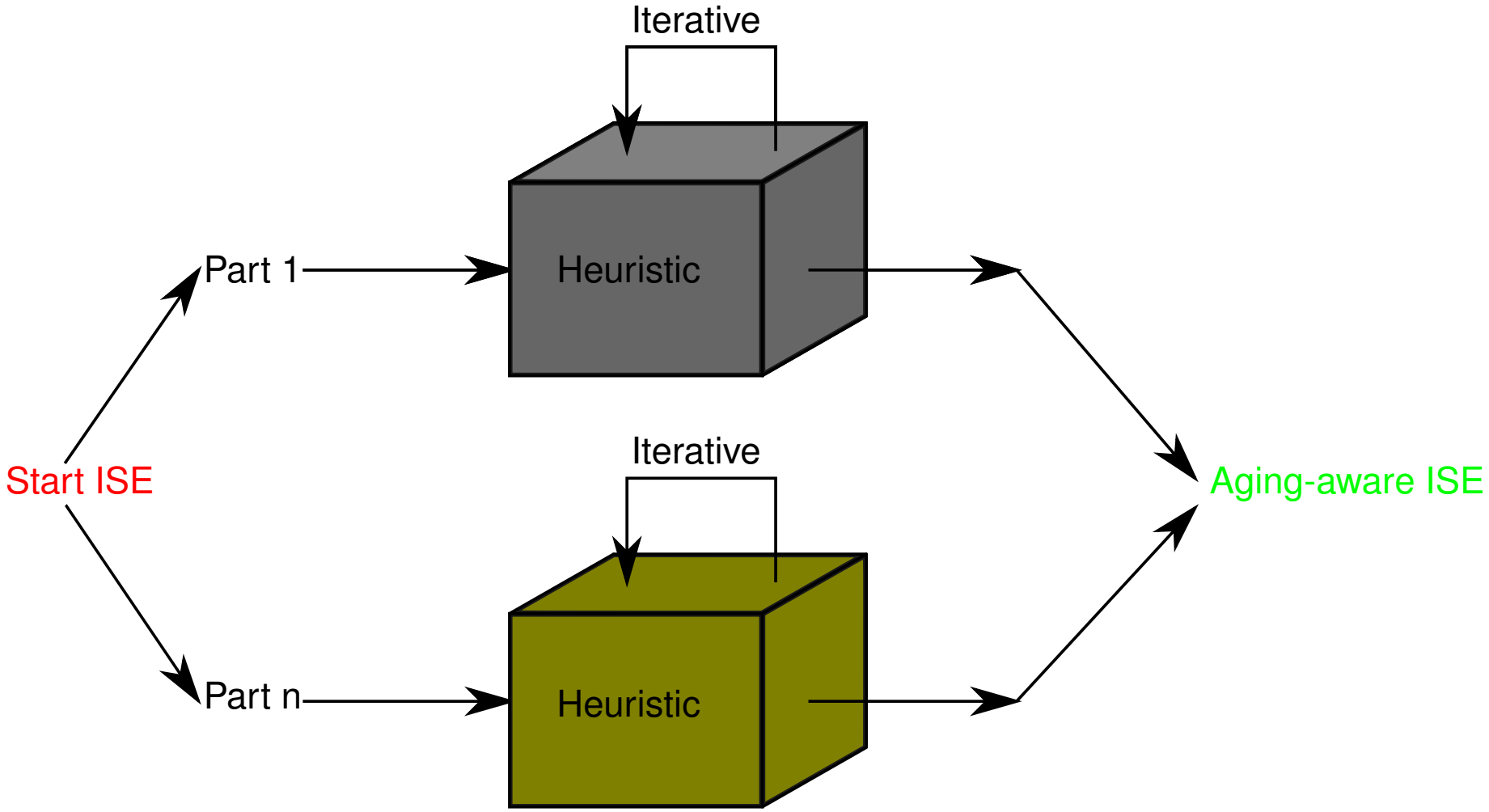
Naïve approach: All opcode bits are optimized in parallel



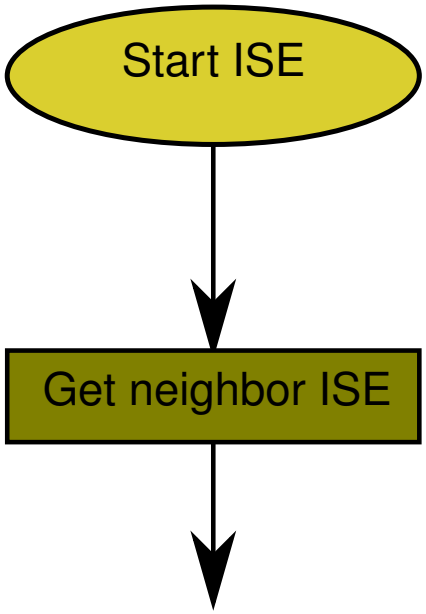
ArISE – Overview

Naïve approach: All opcode bits are optimized in parallel

Enhanced approach: Partition opcode bits and optimize the partitions

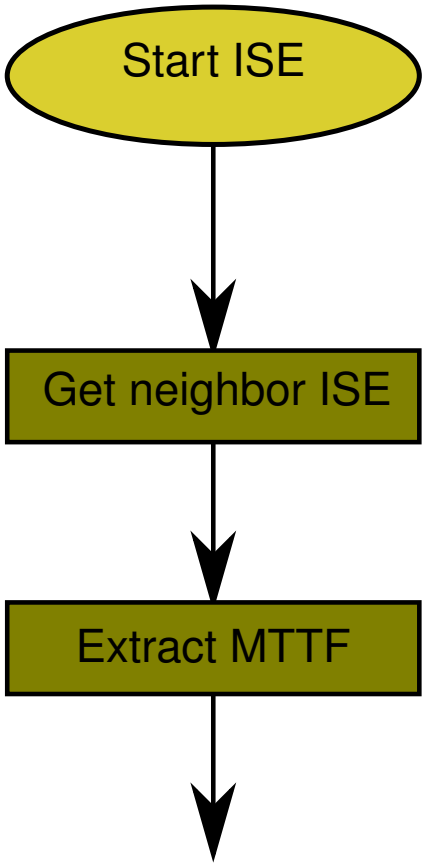


Arise – Heuristic: Simulated Annealing



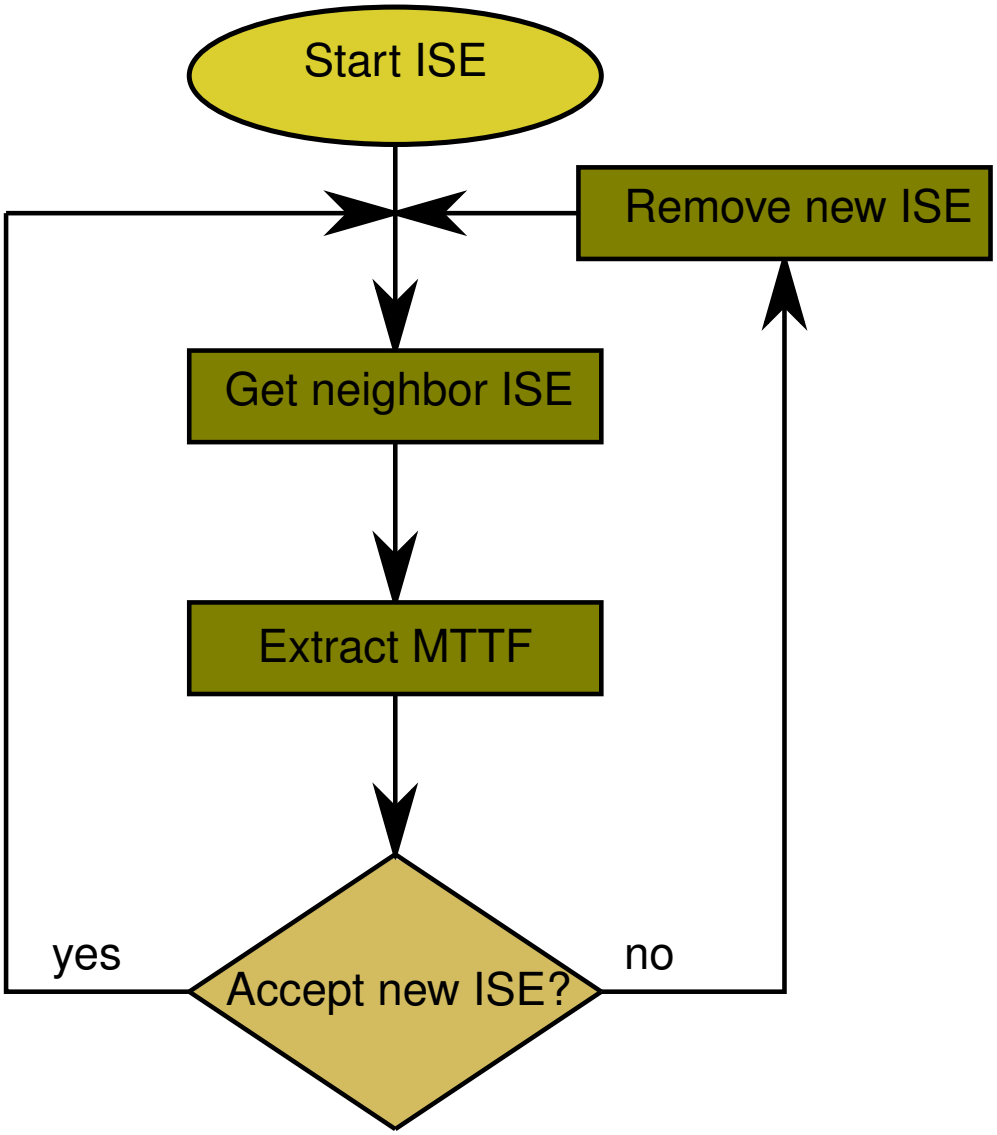
1. 2 ISEs are neighbors $\Leftrightarrow$
   a) ISEs differ in only 1 opcode
      or
   b) 2nd ISE is derived from 1st by exchanging 2 opcodes

Arise – Heuristic: Simulated Annealing



1. 2 ISEs are neighbors $\Leftrightarrow$
   a) ISEs differ in only 1 opcode
      or
   b) 2nd ISE is derived from 1st by exchanging 2 opcodes
2. Extract MTTF & aged delays for each pipeline stage

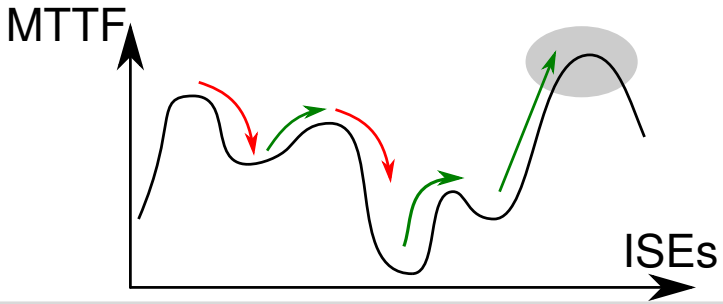
Arise – Heuristic: Simulated Annealing



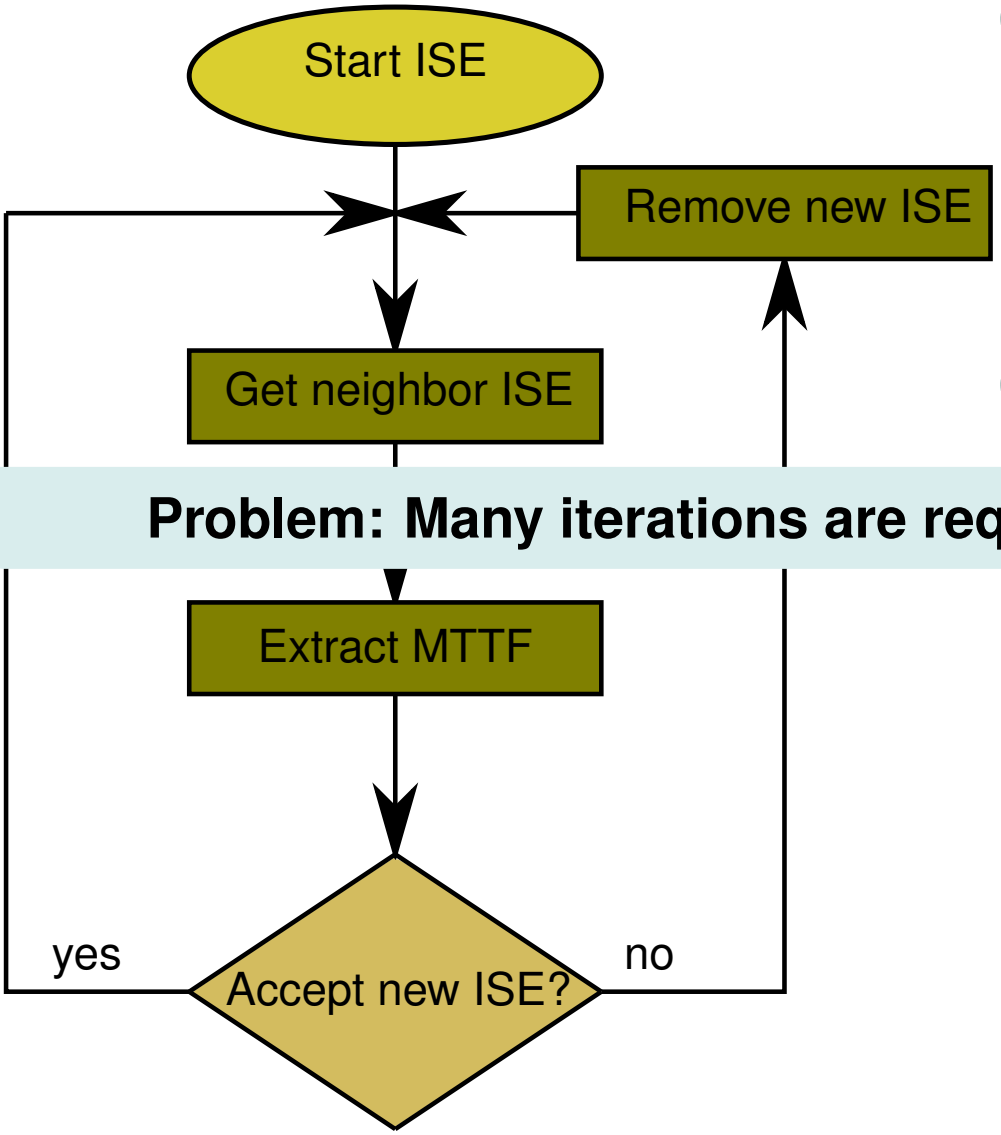
1. 2 ISEs are neighbors $\Leftrightarrow$
   a) ISEs differ in only 1 opcode
   or
   b) 2nd ISE is derived from 1st by exchanging 2 opcodes
2. Extract MTTF & aged delays for each pipeline stage
3. Is the new ISE acceptable?

$$P(acc) = e^{(-(D_{new} - D_{old})/T)} \overset{?}{<} P_{reject}$$

Yes: Continue with new ISE
No: Remove new ISE & continue with old



Arise – Heuristic: Simulated Annealing



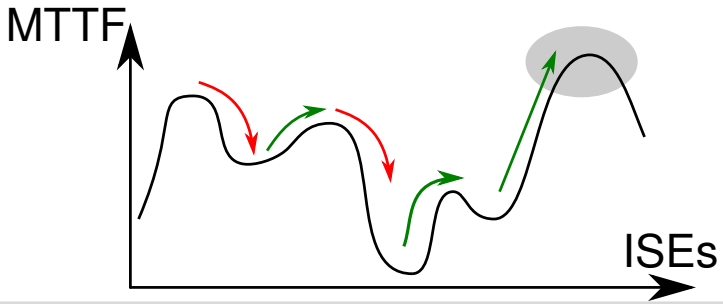
1. 2 ISEs are neighbors ⇔
   a) ISEs differ in only 1 opcode
      or
   b) 2nd ISE is derived from 1st by exchanging 2 opcodes
2. Extract MTTF & aged delays for each pipeline stage

Problem: Many iterations are required ⇒ Time consuming

$$P(acc) = e^{(-(D_{new} - D_{old})/T)} \overset{?}{<} P_{reject}$$

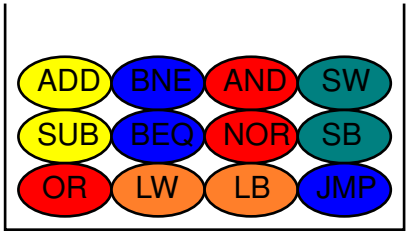
Yes: Continue with new ISE

No: Remove new ISE & continue with old



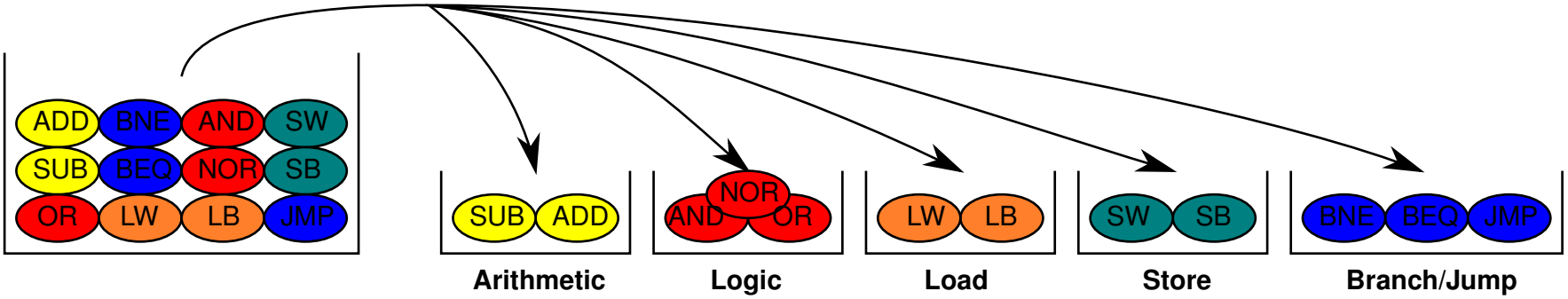
Arise – Hierarchical Approach

Idea: Create instruction groups that are optimized independently



Arise – Hierarchical Approach

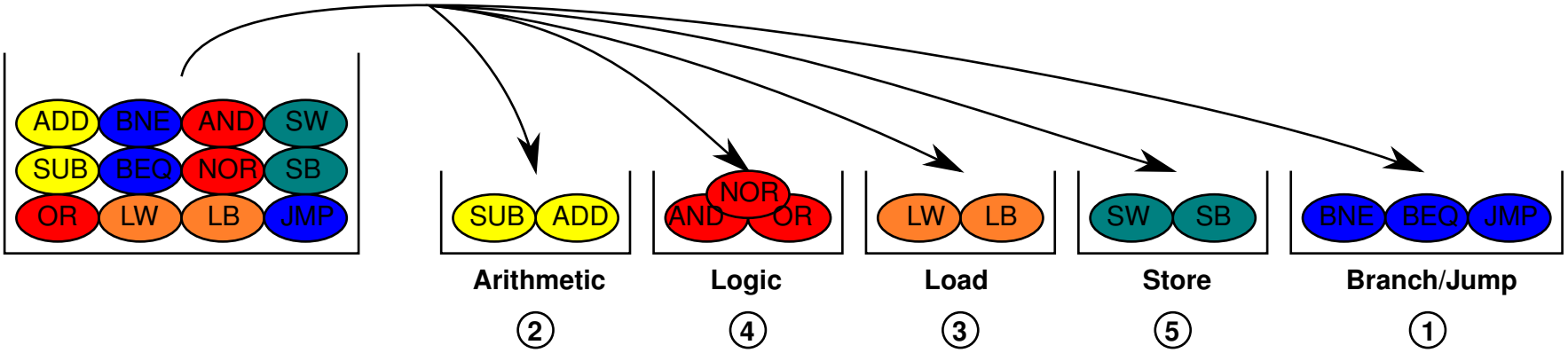
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions

Arise – Hierarchical Approach

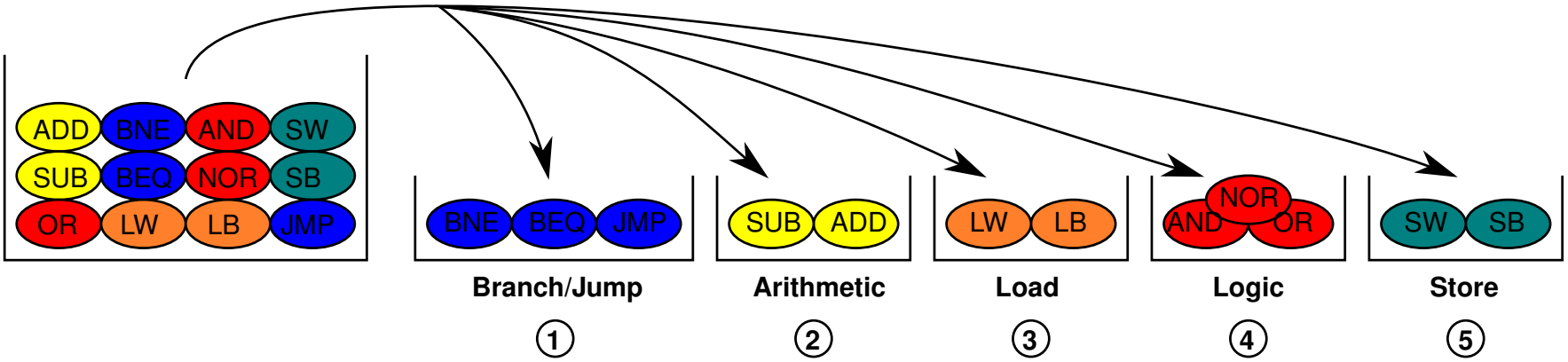
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used

Arise – Hierarchical Approach

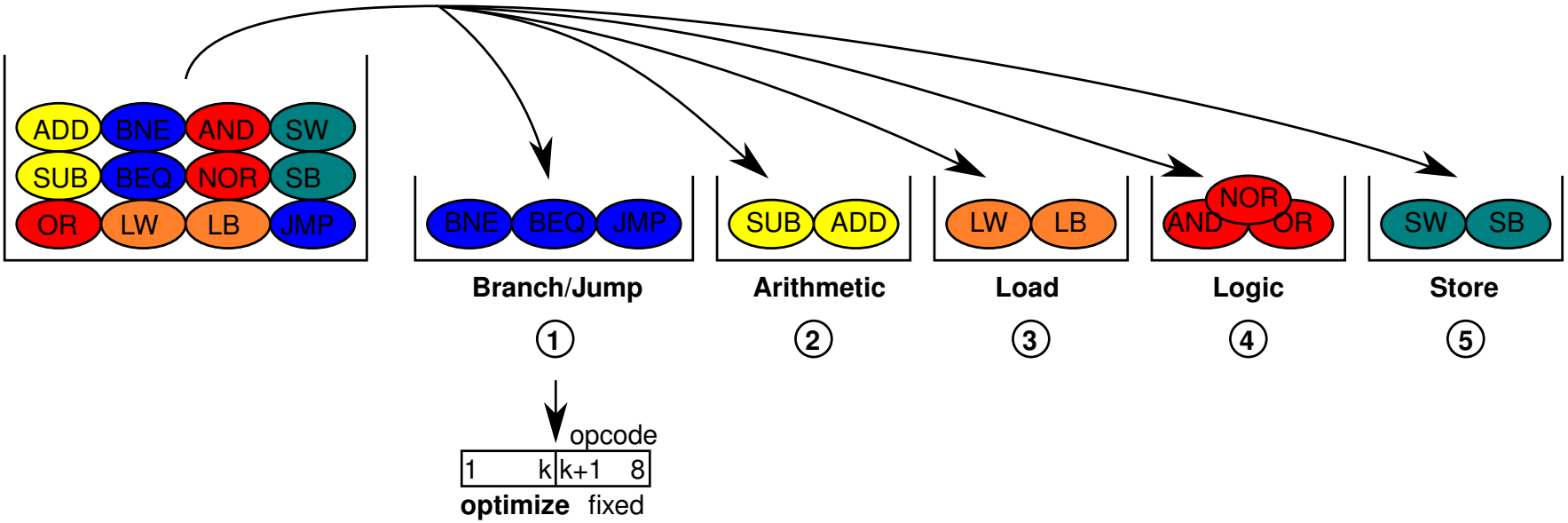
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used

Arise – Hierarchical Approach

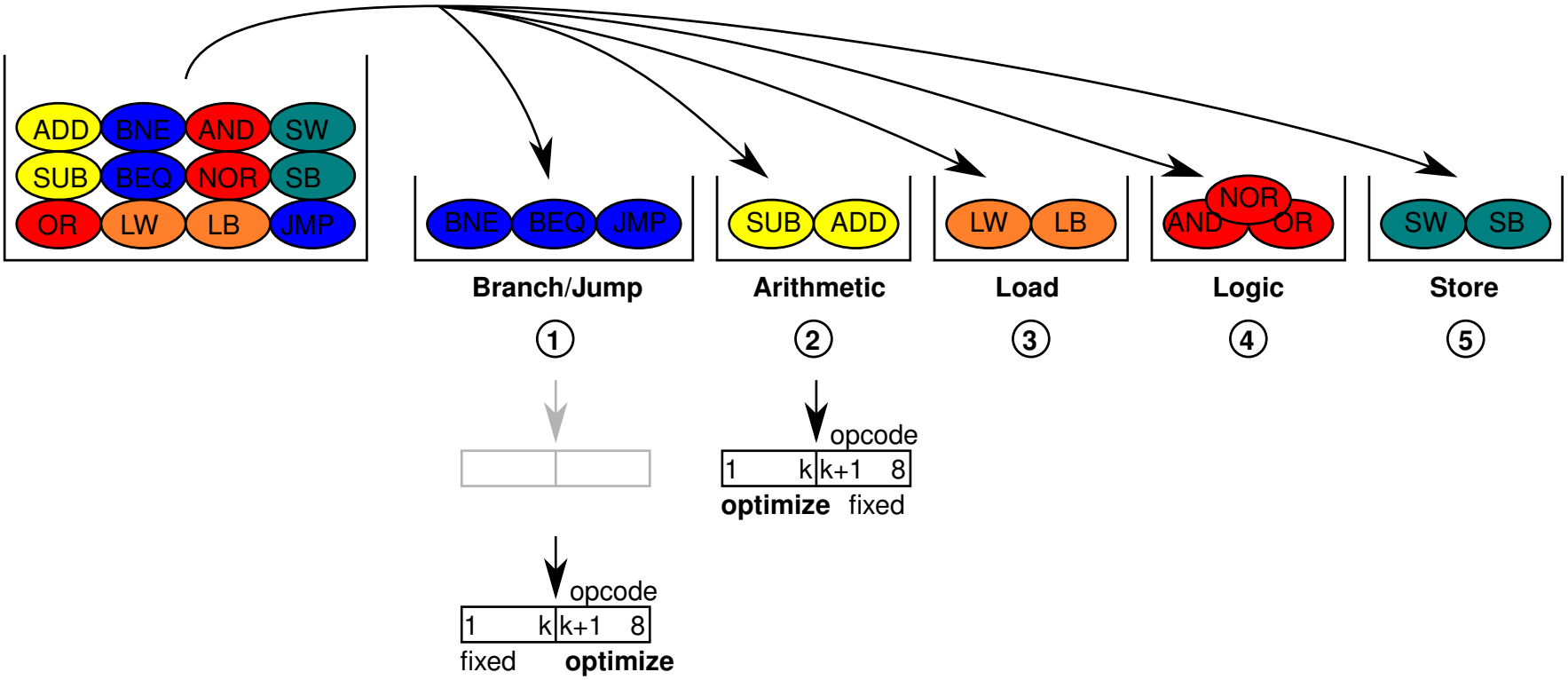
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

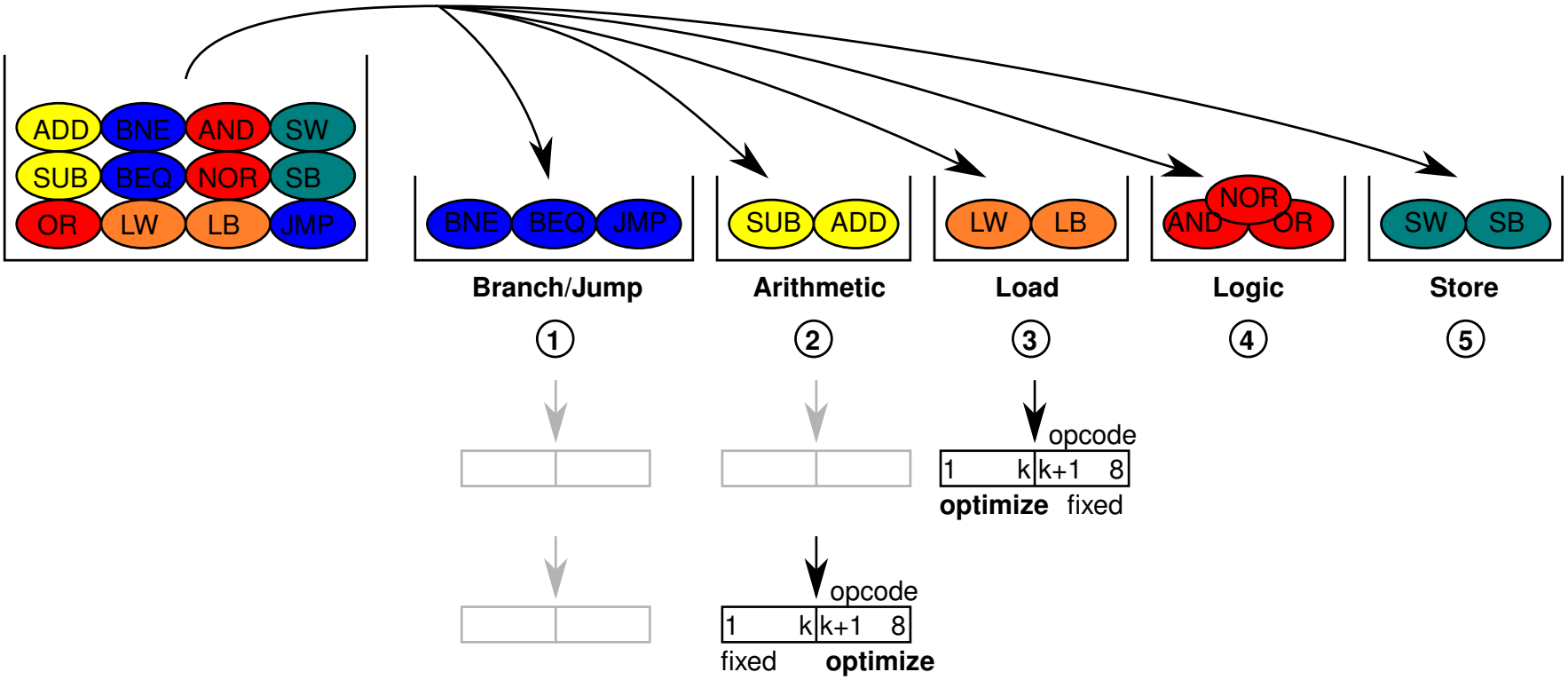
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

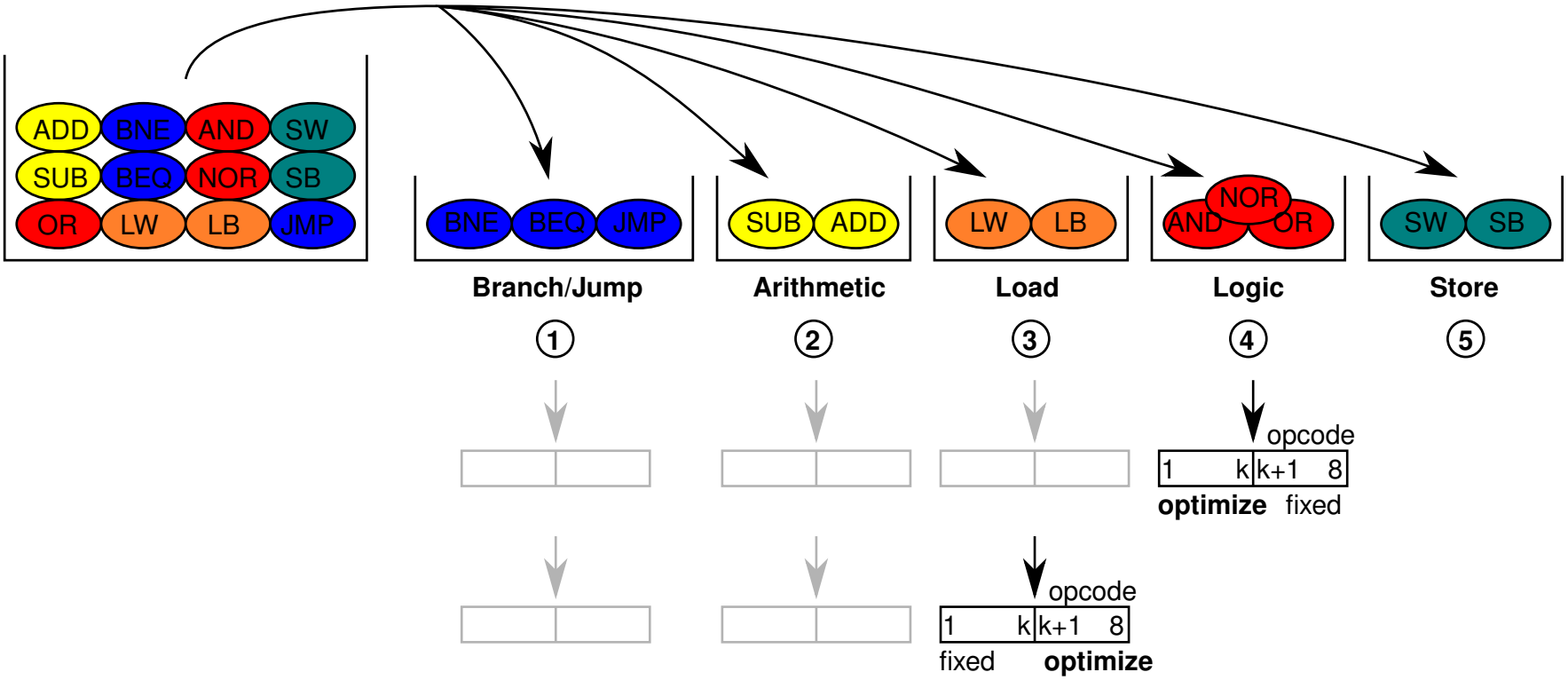
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown → Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

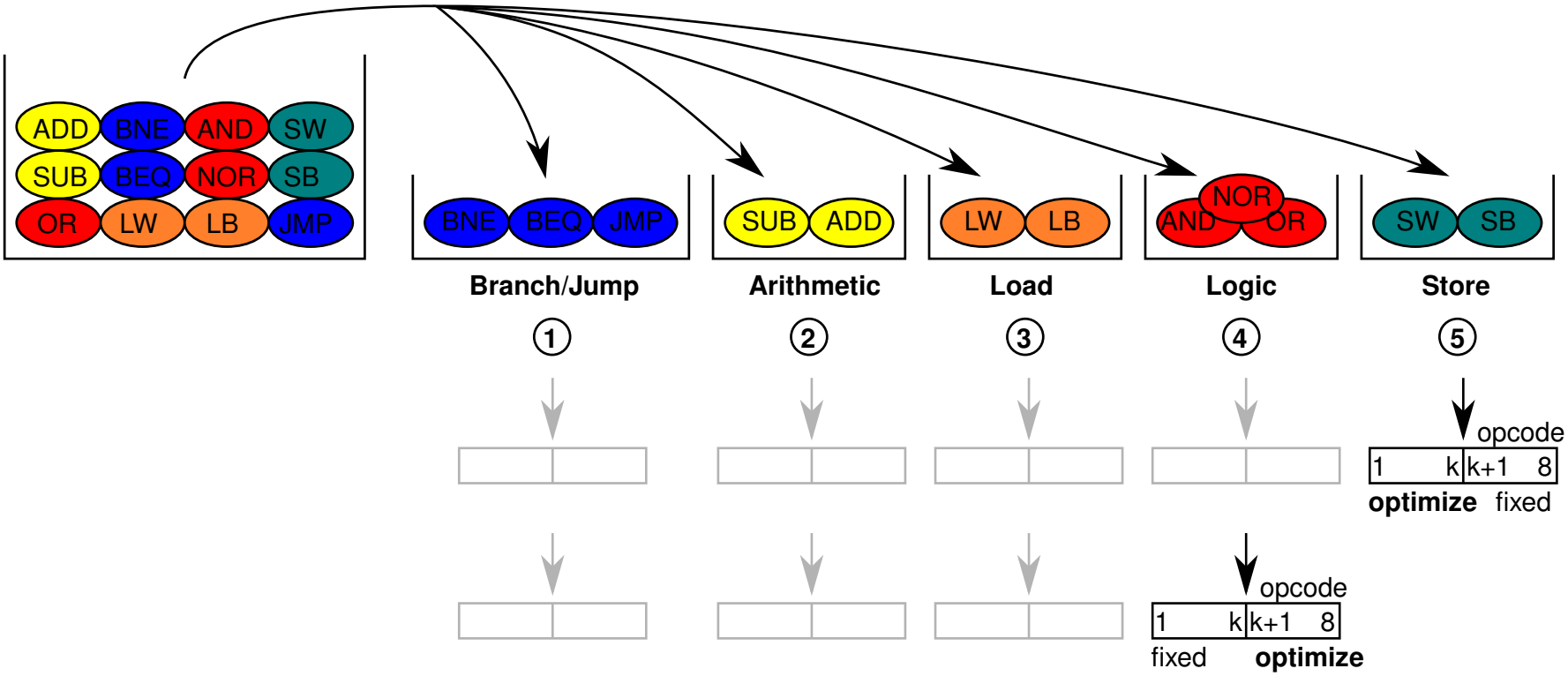
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

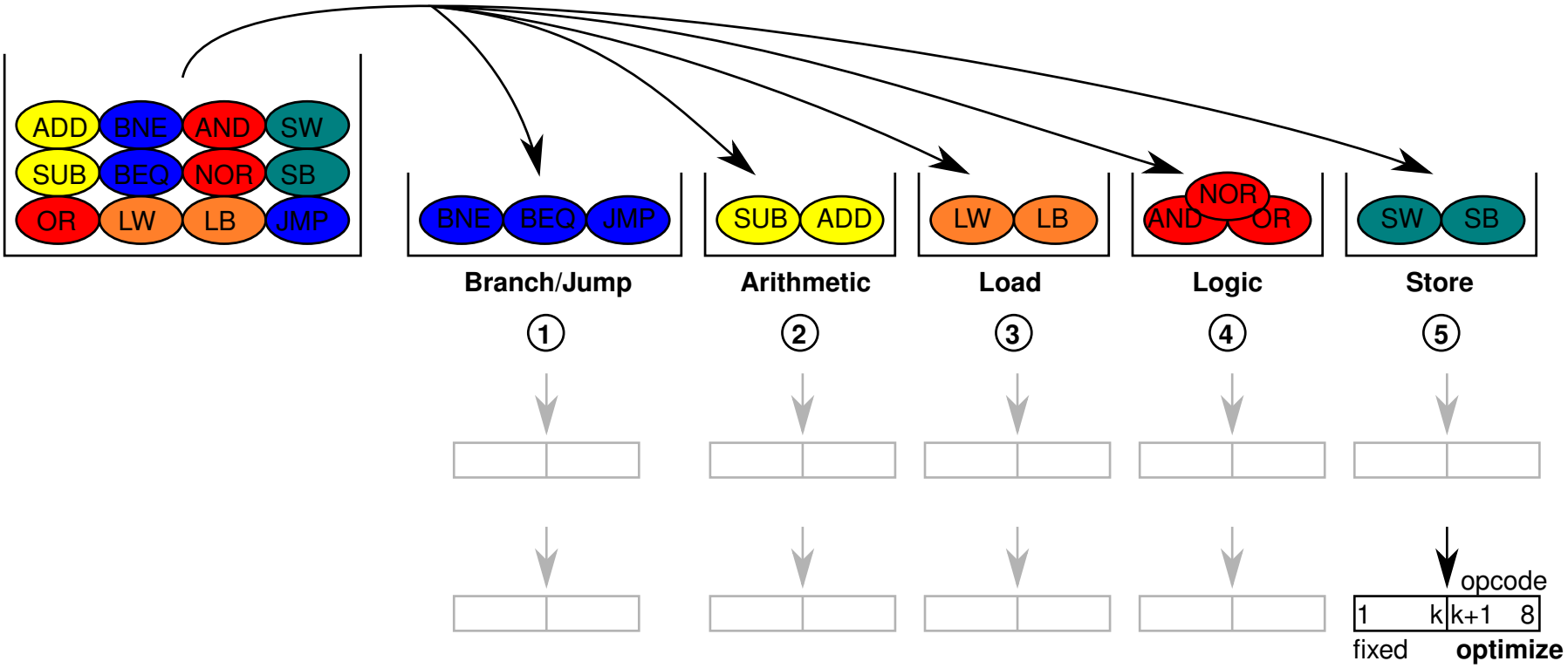
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown → Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

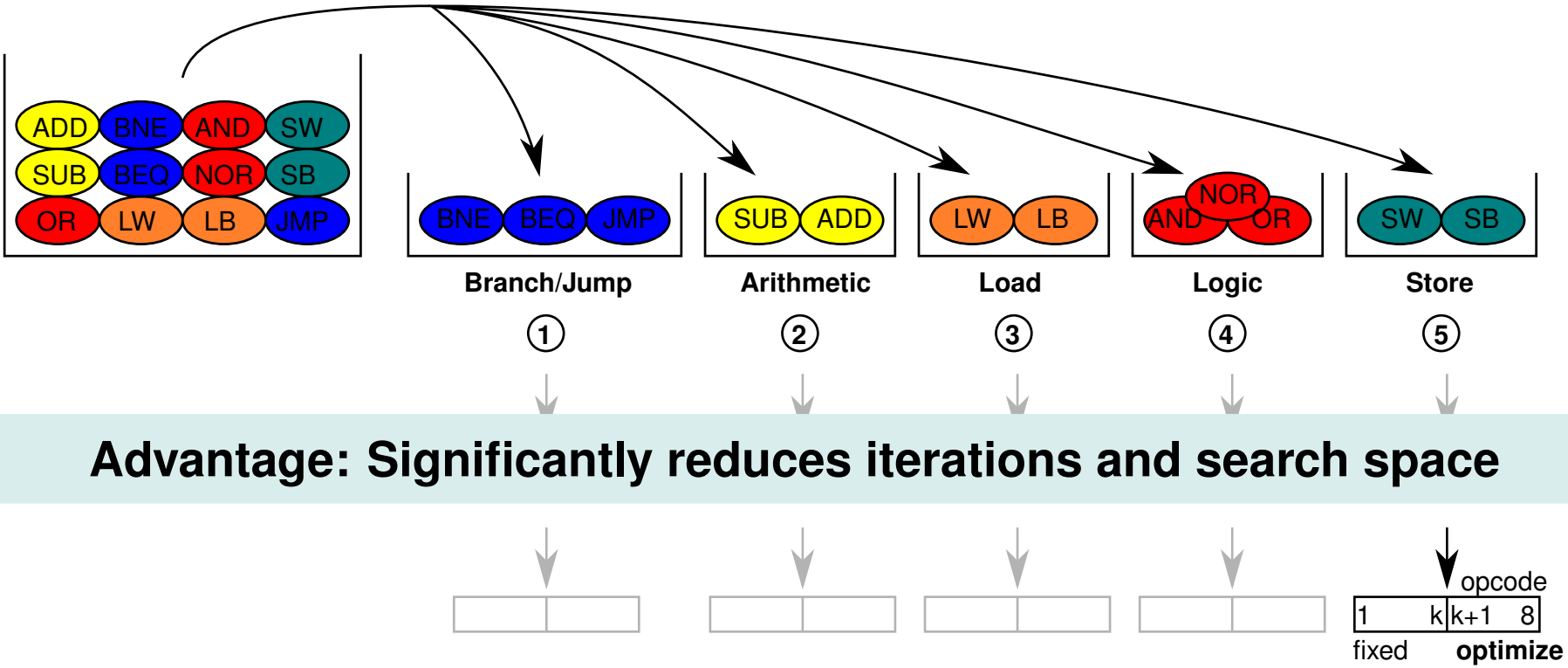
Idea: Create instruction groups that are optimized independently



1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown $\rightarrow$ Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Hierarchical Approach

Idea: Create instruction groups that are optimized independently



Advantage: Significantly reduces iterations and search space

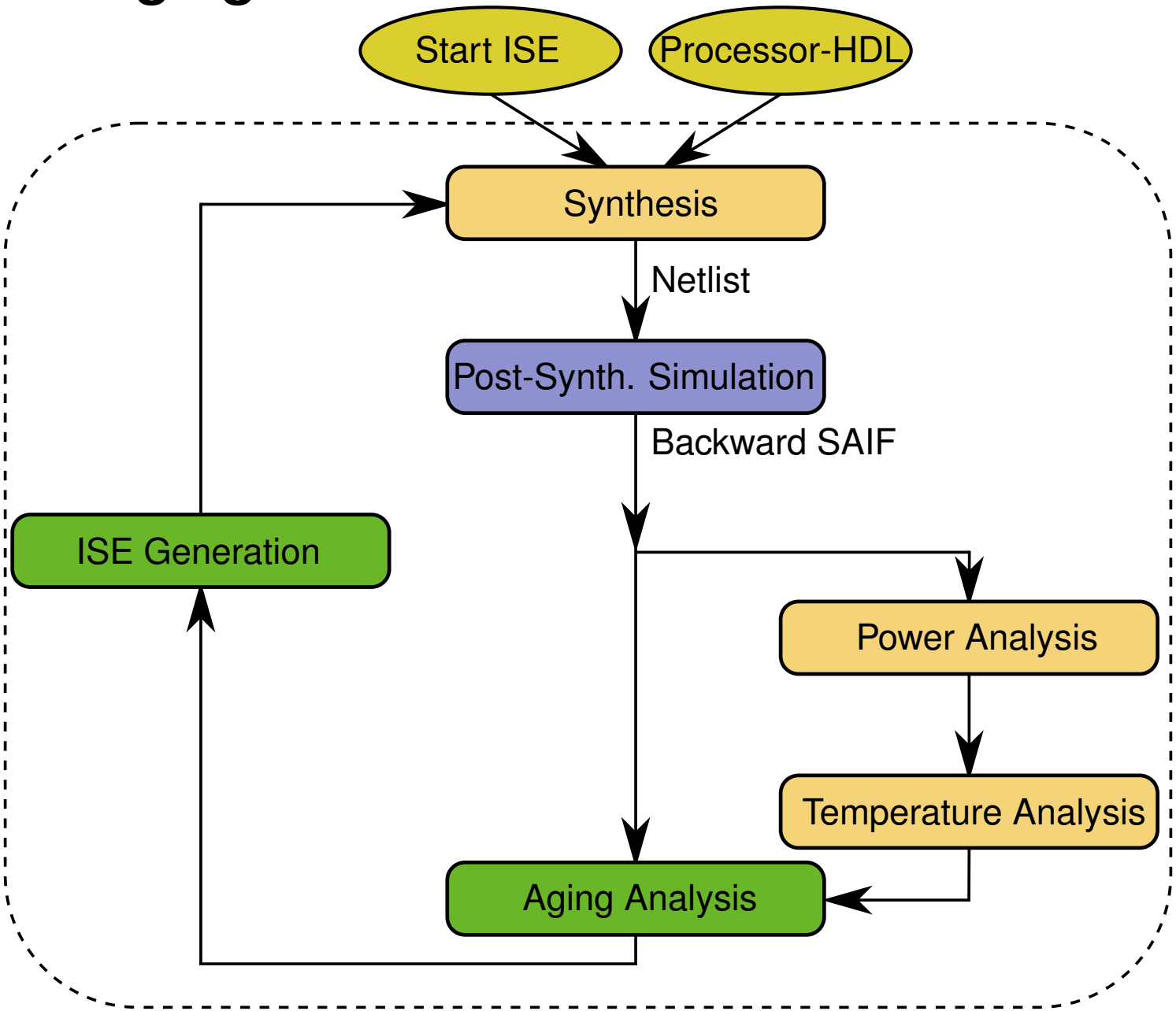
1. Create groups/subgroups for all instructions
2. Rank groups/subgroups according to their aging impact
   - Aging is unknown → Impact on hardware-implementation can be used
3. Optimize each group/subgroup independently

Arise – Further Optimizations

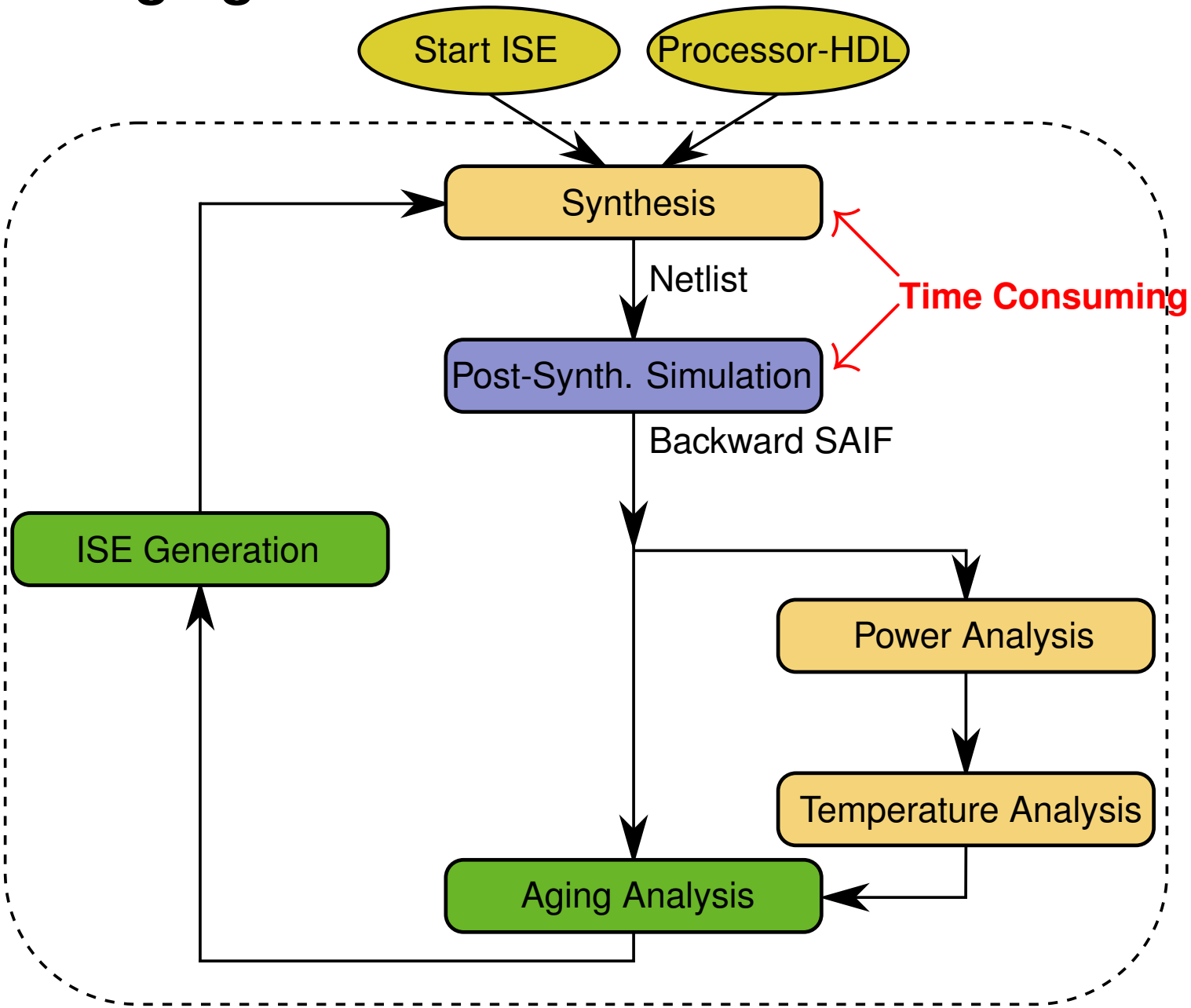
- Store all evaluated ISEs
  - $\rightarrow$ Avoid re-evaluation of ISE
  - $\rightarrow$ Best ISE can be picked, even if it is not the last one

- Aging estimation is time consuming
  - $\rightarrow$ With enhanced aging estimation
    - **1 simulated annealing loop needs 4 min**
    - Limited by time for re-synthesis
  - $\Rightarrow$ 100 steps in less than 6 hours

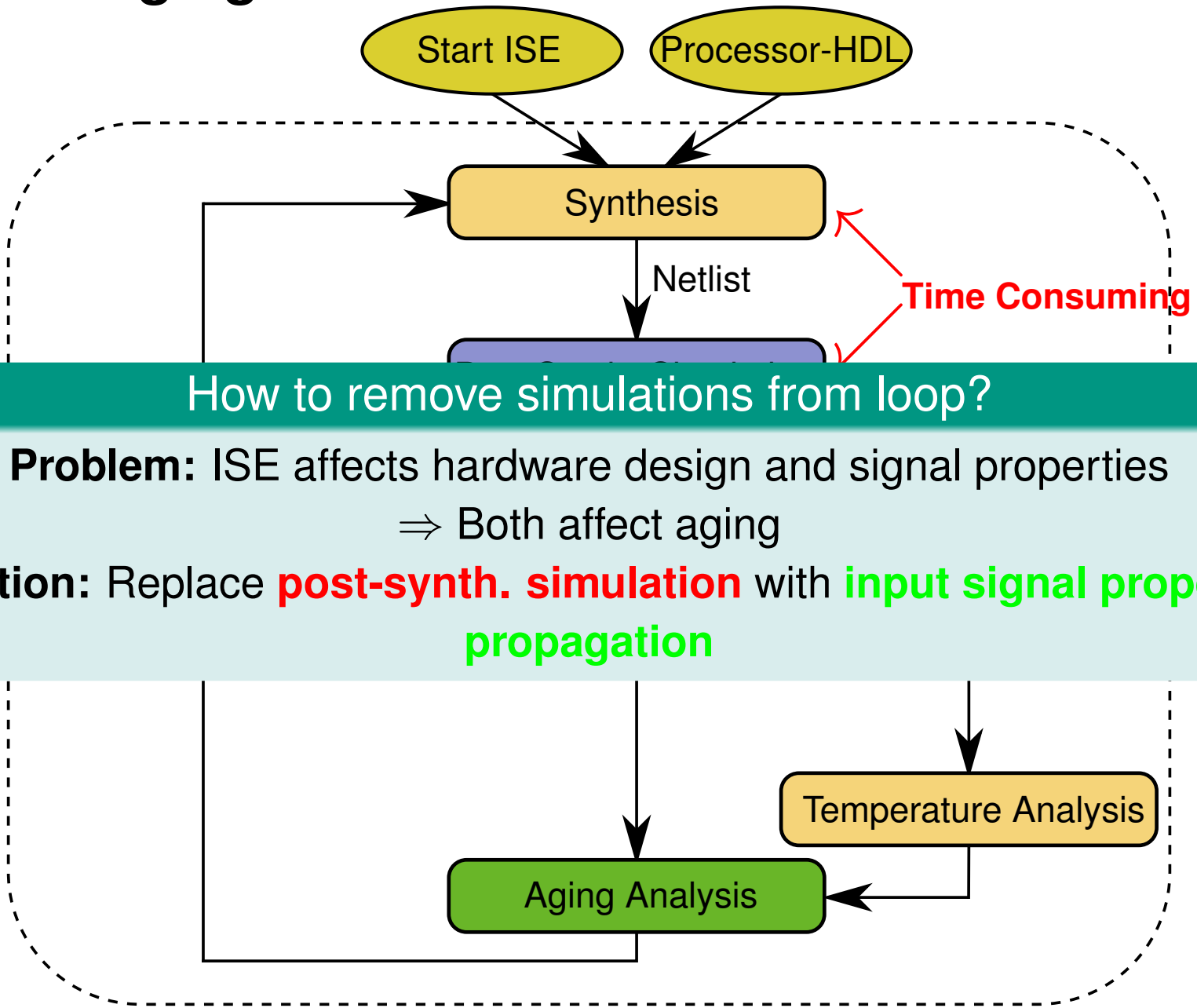
ArISE – Aging Estimation Flow



ArISE – Aging Estimation Flow



ArISE – Aging Estimation Flow



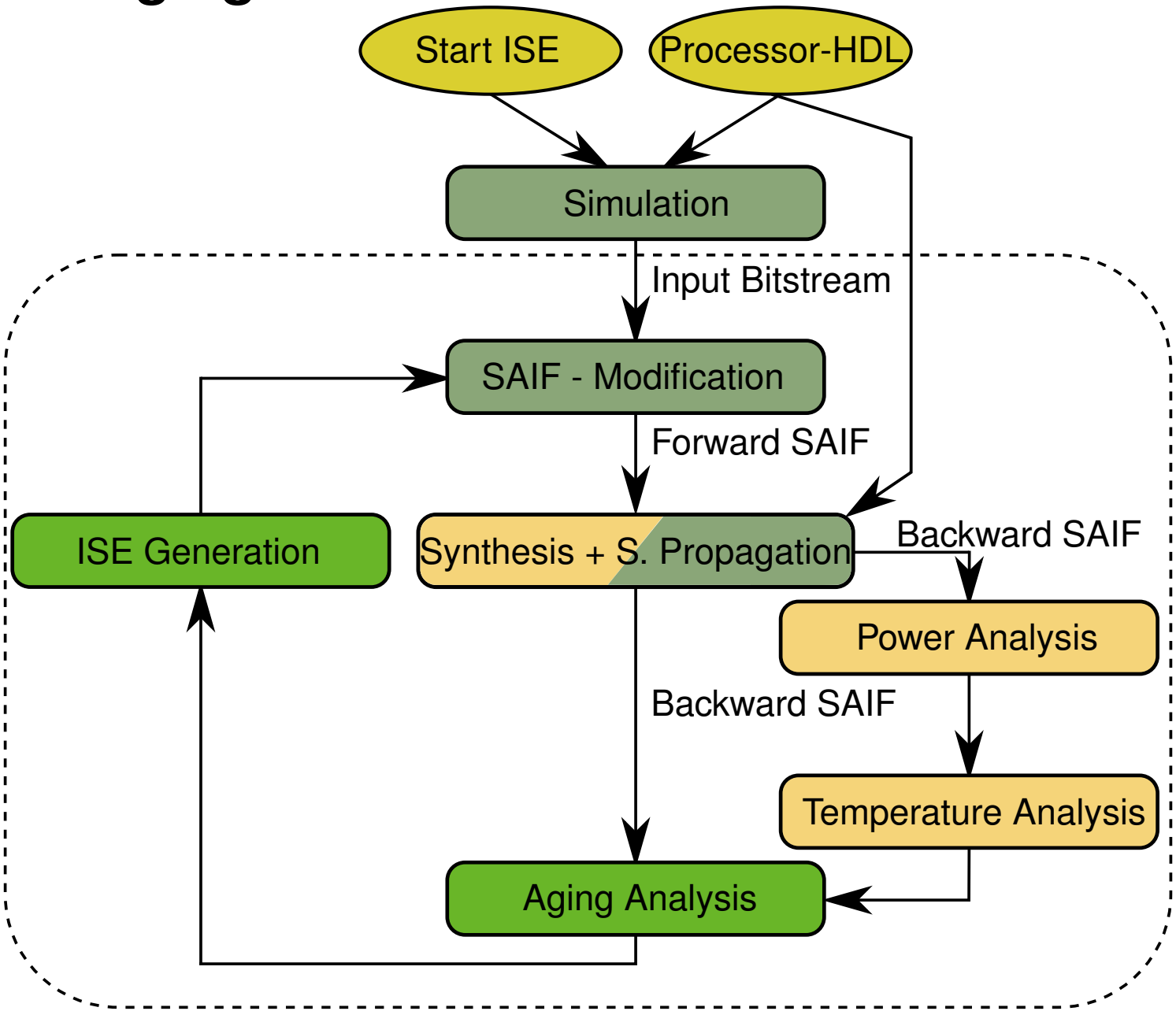
How to remove simulations from loop?

Problem: ISE affects hardware design and signal properties

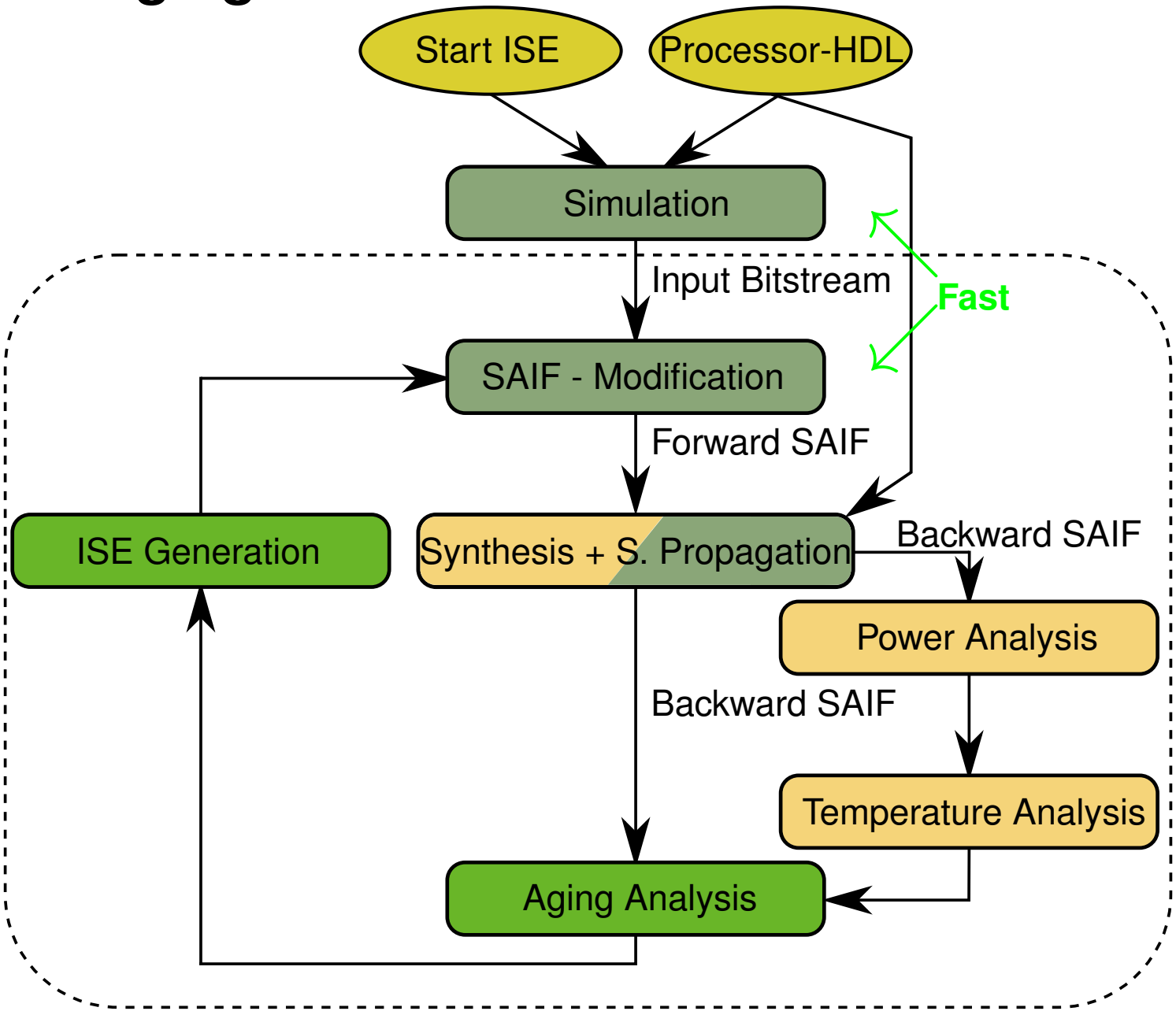
$\Rightarrow$ Both affect aging

Solution: Replace **post-synth. simulation** with **input signal property propagation**

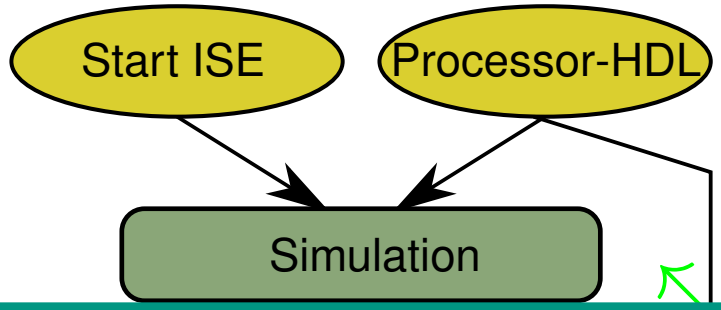
ArISE – Aging Estimation Flow



ArISE – Aging Estimation Flow



ArISE – Aging Estimation Flow



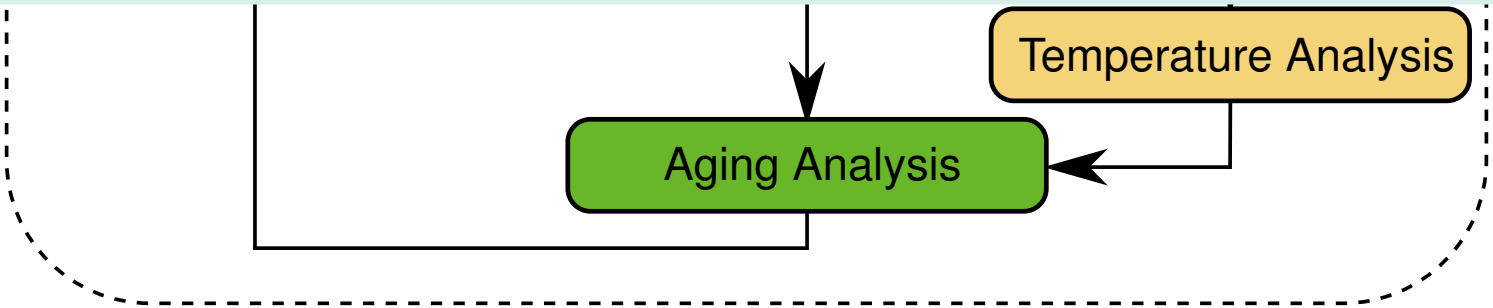
New flow much faster

- Only one simulation step
- Time for signal property propagation is negligible
- Time for SAIF modification is negligible

$\Rightarrow$ Runtime: Few seconds vs. $\approx$ 30 min. for 10^6 clock cycles

- Disadvantage: Reduced accuracy due to signal property propagation
  - But: Inaccuracy less than 0.5 % in delta delay

  $\Rightarrow$ Good enough for optimization



ArISE – Application of Modified ISE

Modified ISE $\Rightarrow$ **Old software binaries are incompatible!**

ArISE – Application of Modified ISE

Modified ISE $\Rightarrow$ **Old software binaries are incompatible!**

- Solution 1: **Software-Based Approach**
  - Build new compiler based on modified ISE
  - $\Rightarrow$ Re-compile applications, either once or always on-the-fly
  - Costly for backward-compatible processors

ArISE – Application of Modified ISE

Modified ISE $\Rightarrow$ **Old software binaries are incompatible!**

- Solution 1: **Software-Based Approach**
  - Build new compiler based on modified ISE
  - $\Rightarrow$ Re-compile applications, either once or always on-the-fly
  - Costly for backward-compatible processors
- Solution 2: **Hardware-Based Approach**
  - Implement a mapper from old to modified ISE
    - e.g., Look-up-Table, logic-statements (if-else)
  - $\Rightarrow$ Low overhead: $< 2\,\%$ area
  - Attention: Critical path needs to be avoided!

Outline

Experimental Setup

- FabScalar microprocessor
  - 11 Stage pipeline
  - Out-of-order, 4-issue
  - 170k Gates (w/o memory)
  - max clock with our setup: 740 MHz
- TSMC 65nm HP CMOS library
- Synthesis: Synopsys Design Compiler
- Simulation: Cadence NC Verilog + SPEC2000
- Power Analysis: Synopsys PrimeTime
- Temperature Analysis: HotSpot

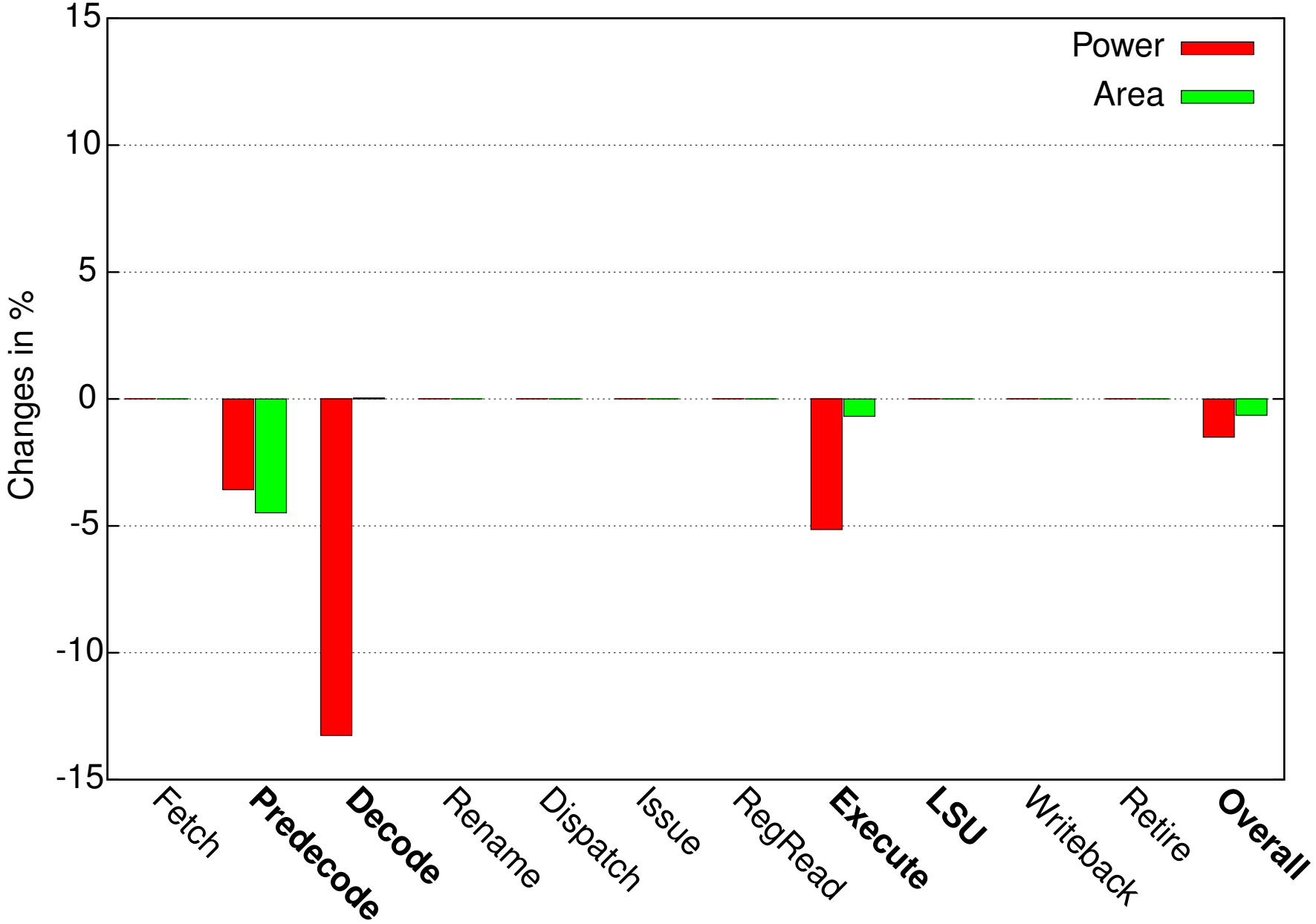
Results – Lifetime

- Recall: Predecode and Decode stage are sensitive to ISE
- Lifetime-Results:

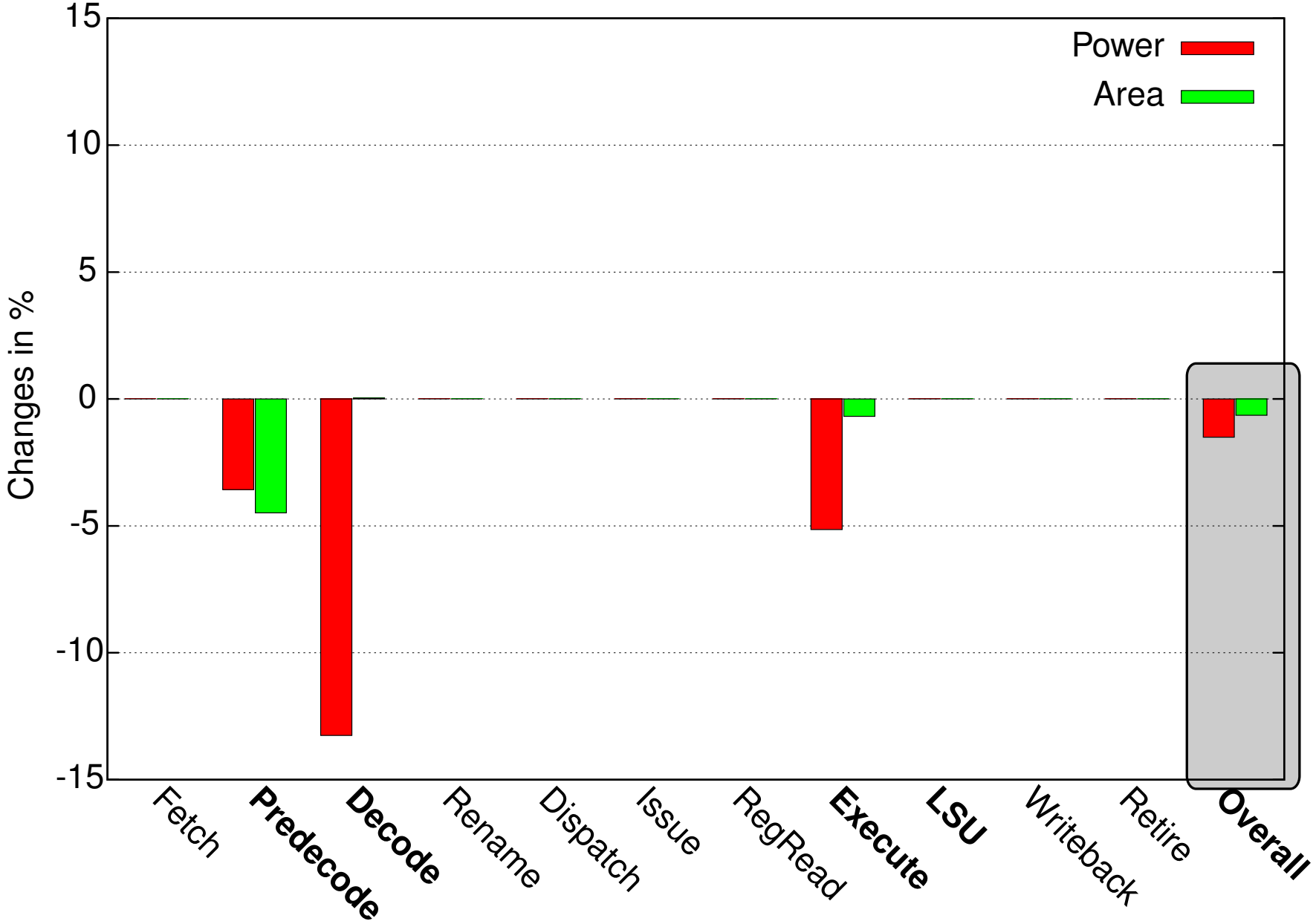
Stage	Standard ISE			Best ISE		
	Delay [ns] (0y)	Delay [ns] (3y)	MTTF [years]	Delay [ns] (0y)	Delay [ns] (3y)	MTTF [years]
Predecode	1.35	1.48	**3.0**	1.35	1.46	**5.8**
Decode	1.34	1.43	15.9	1.34	1.42	19.1
Overall	1.35	1.48	3.0	1.35	1.46	5.8 **+1.93x**

- Best ISE was obtained with hierarchical optimization flow
  - Branch instructions are critical $\Rightarrow$ Most important instruction group
  - 16 iterations to find best encoding for branch group (exhaustive!)
  - 25 iterations for instructions inside this group ($<$ 100 min.)

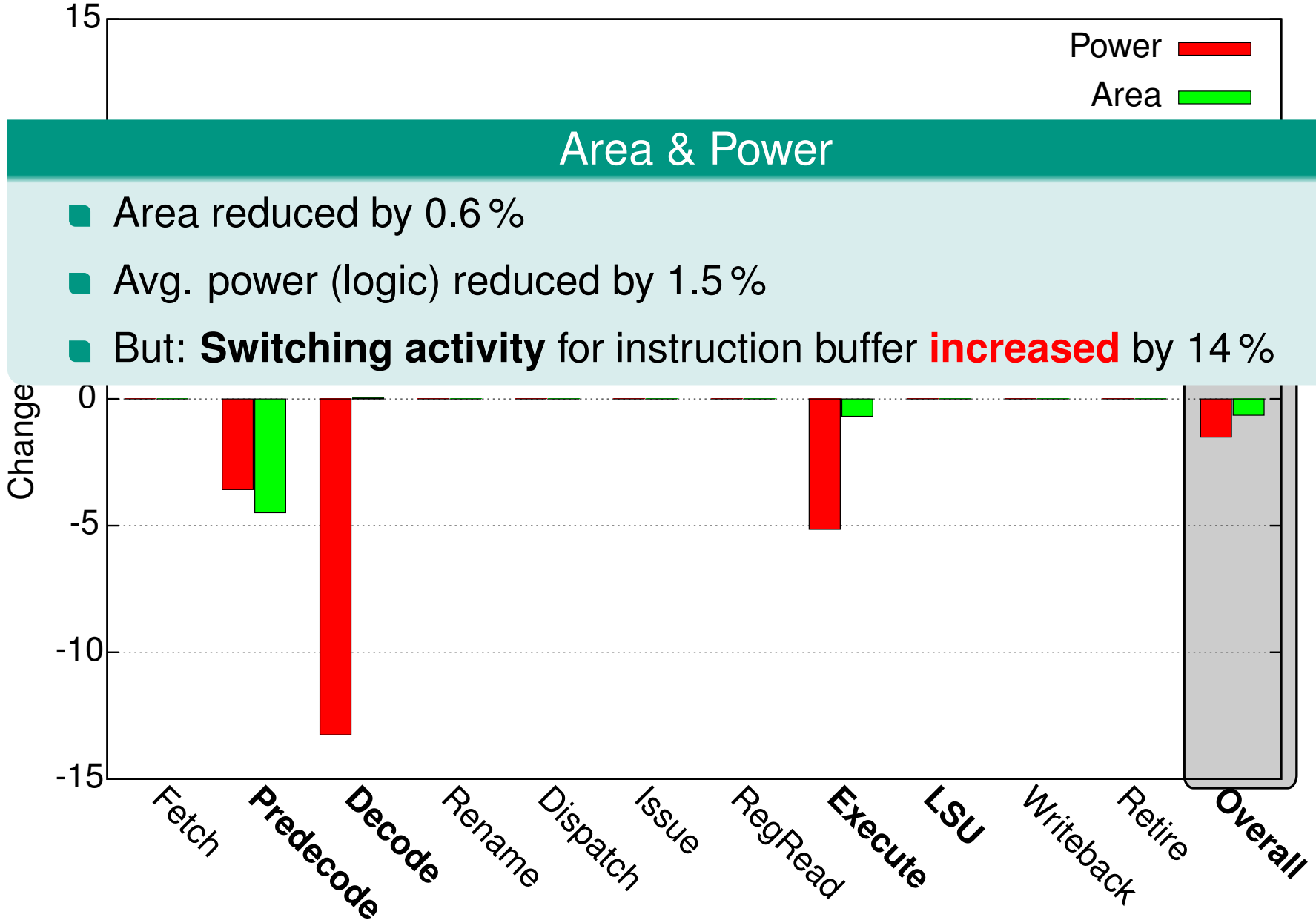
Results – Area & Power



Results – Area & Power



Results – Area & Power



Area & Power

- Area reduced by 0.6 %
- Avg. power (logic) reduced by 1.5 %
- But: **Switching activity** for instruction buffer **increased** by 14 %

Results – Comparison with Related Work

- Recall: [DeBole2009] proposed to periodically invert opcodes
- Evaluation for Predecode stage

	Our Technique	Periodical Inversion [DeBole2009]		
		never	always	every 10^3 cyc
Δ-Delay @ 3y	8.1 %	9.1 %	9.0 %	9.1 %
MTTF	5.8 years	4.0 years	4.1 years	4 years

- Periodical opcode inversion balances signal probabilities ≈ 0.5
- Instead: Our techniques optimizes signal probabilities for aging-critical gates as much as possible
  - $\rightarrow$ e.g. high signal probability is favorable for NBTI

Outline

Conclusion

- Reliability is a major design constraint at nanoscale
  - $\rightarrow$ Accelerated transistor aging has to be considered through out the entire design phase
- Most (micro)-architectural aging mitigation techniques focus on execution stage
  - $\rightarrow$ But also decoding stages can become critical
- Aging-Aware Instruction Set Encoding increases lifetime of decoding stages with no impact on performance

Thank you!