九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Variation Aware Compilation for Improving Energy-Efficiency of Nanometer Processor Caches

Goudarzi, Maziar
System LSI Research Center, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

Yasuura, Hiroto
System LSI Research Center, Kyushu University

https://hdl.handle.net/2324/9149

出版情報：SLRC プレゼンテーション, 2006-10-27. 九州大学システムLSI研究センター
バージョン：
権利関係：

Variation Aware Compilation for Improving Energy-Efficiency of Nanometer Processor Caches

Maziar Goudarzi, Tohru Ishihara, Hiroto Yasuura

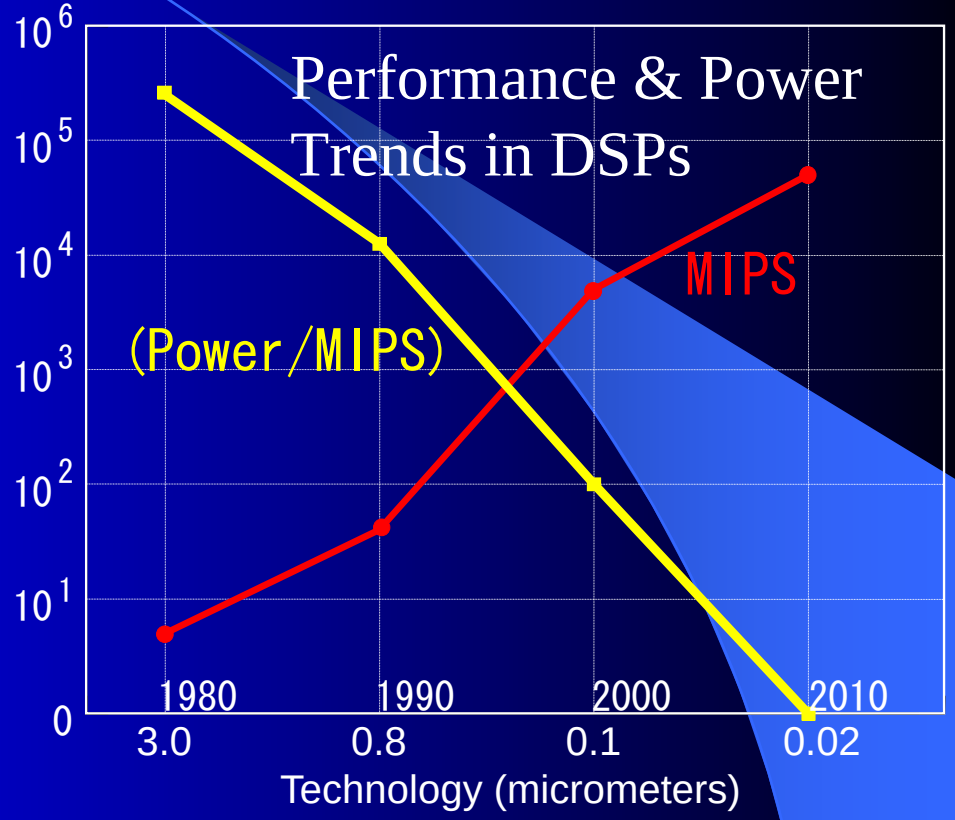
System LSI Research Center
Kyushu University

Outline

- Background
  - Process variation in nanometer caches
    - Delay variation
    - Leakage variation
- Our previous and current work on SRAM
  - Mitigating effects of ultra-slow SRAM cells
  - Suppressing leakage of ultra-leaky ones

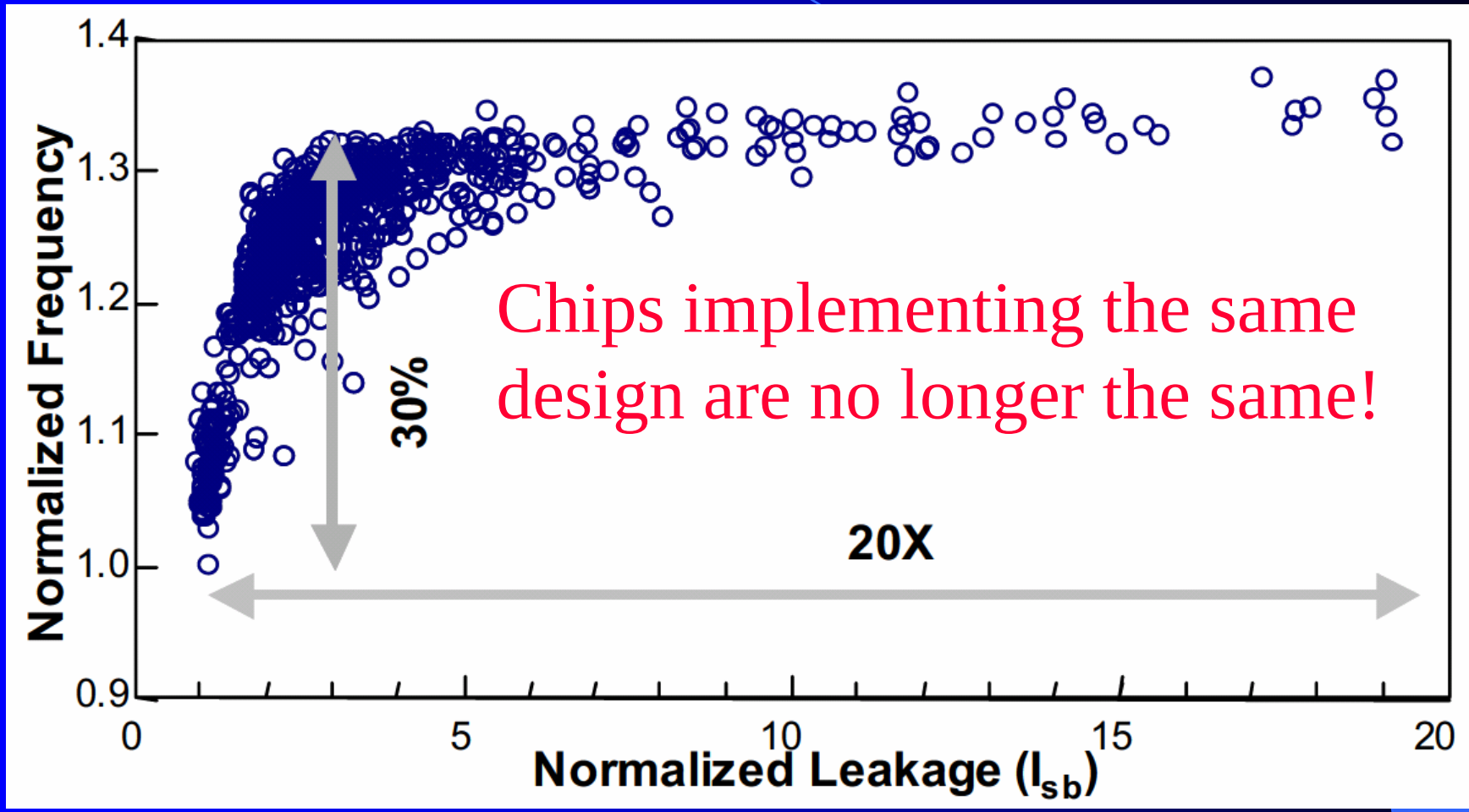
Trends of DSPs

- An improvement of power efficiency highly depends on the process technology
- This trend continues more than 10 years



Source: G. Frantz, “Digital Signal Processor Trends”, MICRO, vol.20, no.6, December 2000

Process Variation



S. Borkar, Parameter variations and impact on circuits and microarchitecture, DAC 2003.

Intra-Die Variations

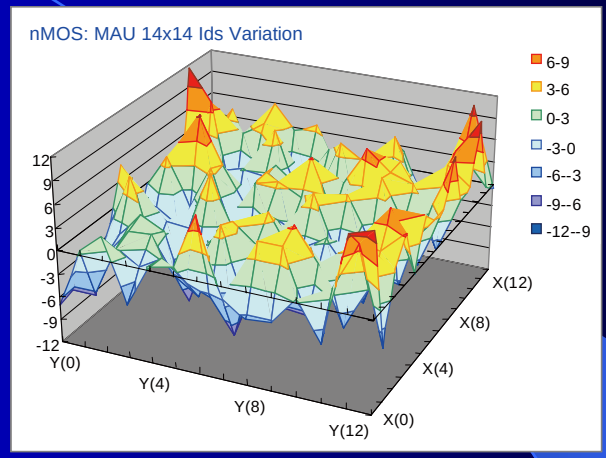
Large Intra-Die Variation

Current 3-sigma = 13%
Vth 3-sigma = 67mV

Variation is huge in small transistors

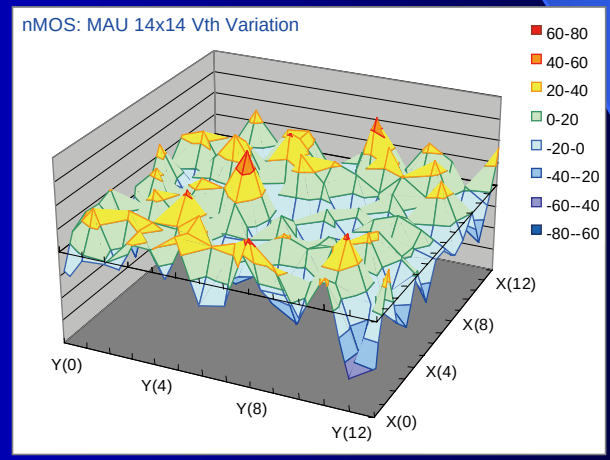
$$\sigma_{Vth} = \frac{q}{C_{ox}} \sqrt{\frac{N_a \cdot W_{dm}}{3 \cdot L \cdot W}}$$

L, W: Effective channel length and width
q: electron charge
C_{ox}: oxide capacitance
N_a: substrate doping concentration
W_{dm}: maximum depletion width



L = 0.1um
W= 0.4um

Av. = 203.7uA
Sigma = 4.4%
min. = -11.4%
max. = 11.4%



L = 0.1um
W= 0.4um

Av. = 308.3uA
Sigma = 22.1mV
min. = -66.6mV
max. = 57.0mV

Eijiro Toyoda, "DFM: Device & Circuit Design Challenges", Int'l Forum on Semiconductor Technology, 2004

Process Variation at 90nm

$$I_{Subthreshold} \propto \frac{W \cdot V_T^2}{T_{ox} \cdot L} \cdot \exp\left(\frac{-V_{th}}{\alpha \cdot V_T}\right)$$

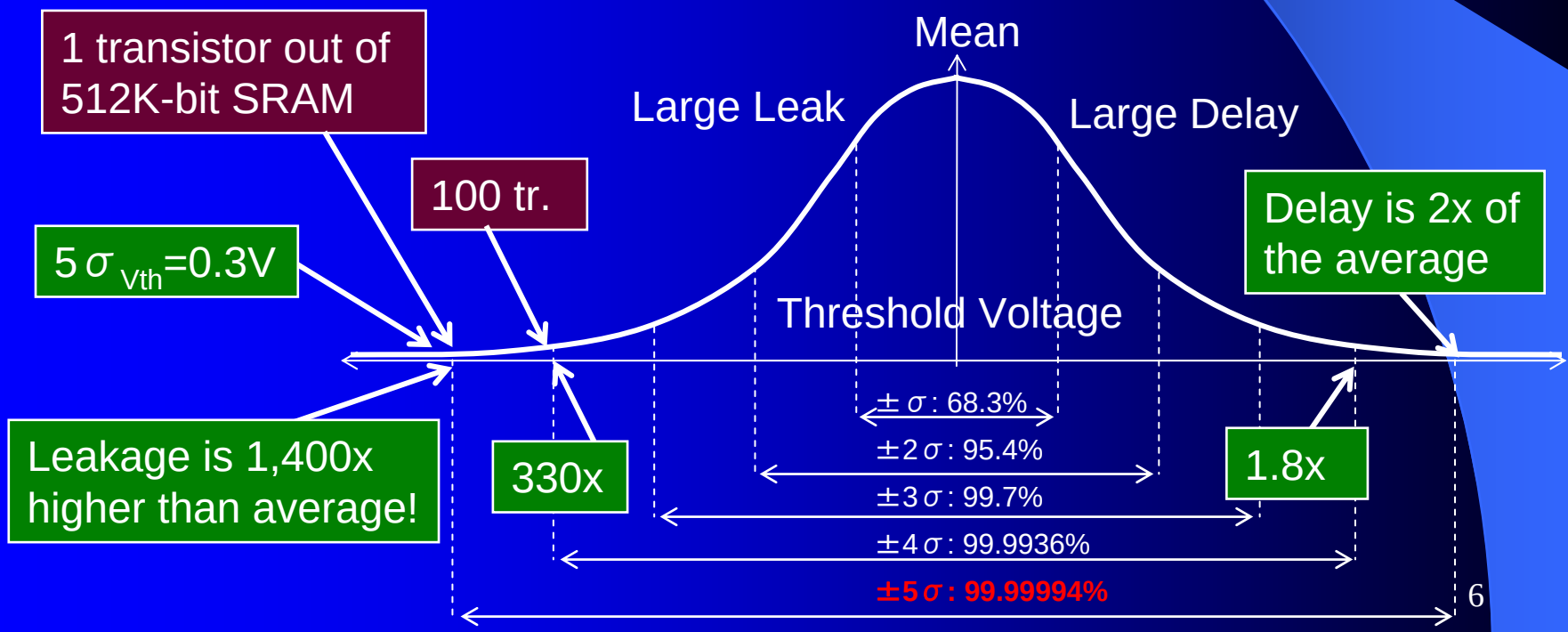
V_T: Thermal voltage (25mV@room temperature)

α : Sub-threshold factor (1.40~1.65)

T_{ox}: Oxide thickness

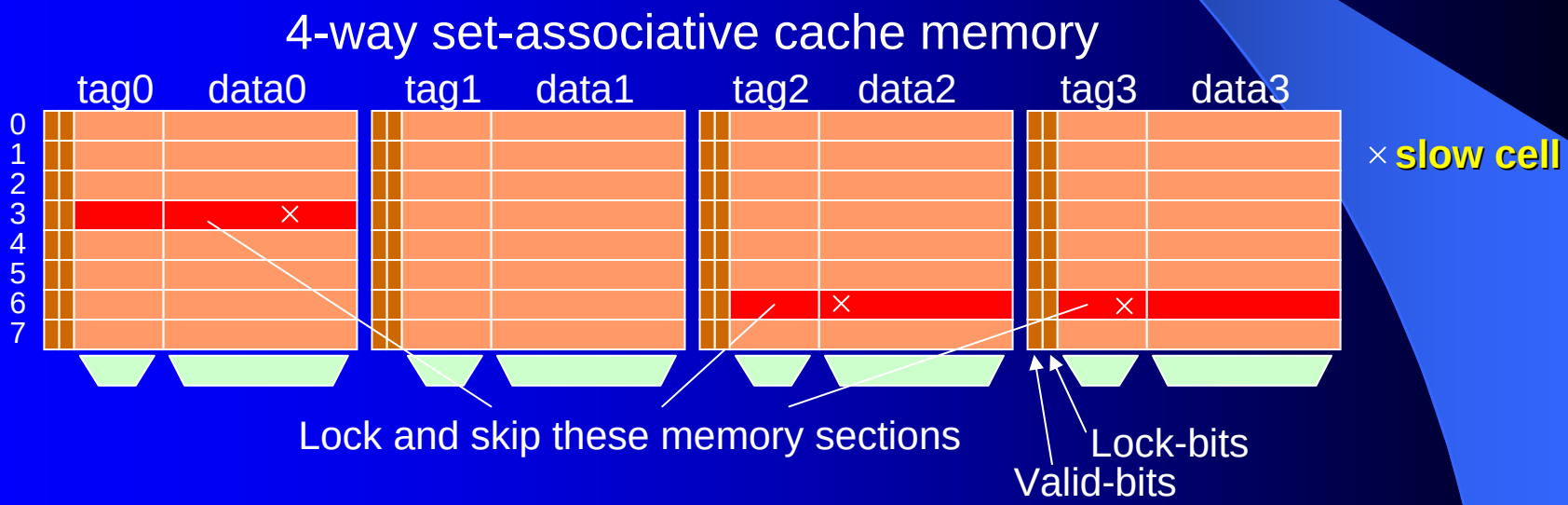
Year	min. L [nm]	$^1V_{TH}$ [V]	$^2V_{TH}$ [V]
2004	37 (90)	0.32	0.12
2005	32 (80)	0.33	0.09
2006	28 (70)	0.34	0.06

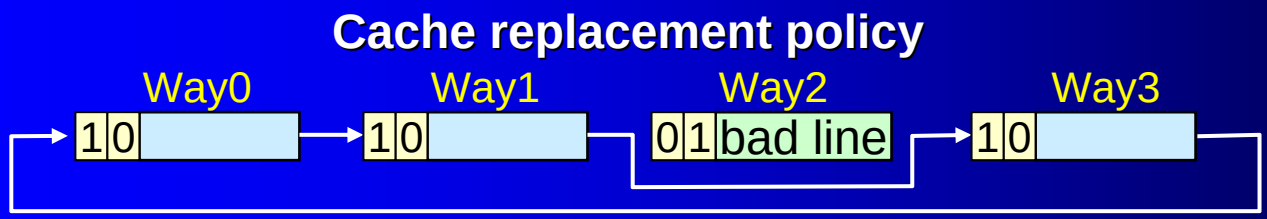
1: Low Operating Power Process 2: MPU process



Skipping Slow Cache-Lines

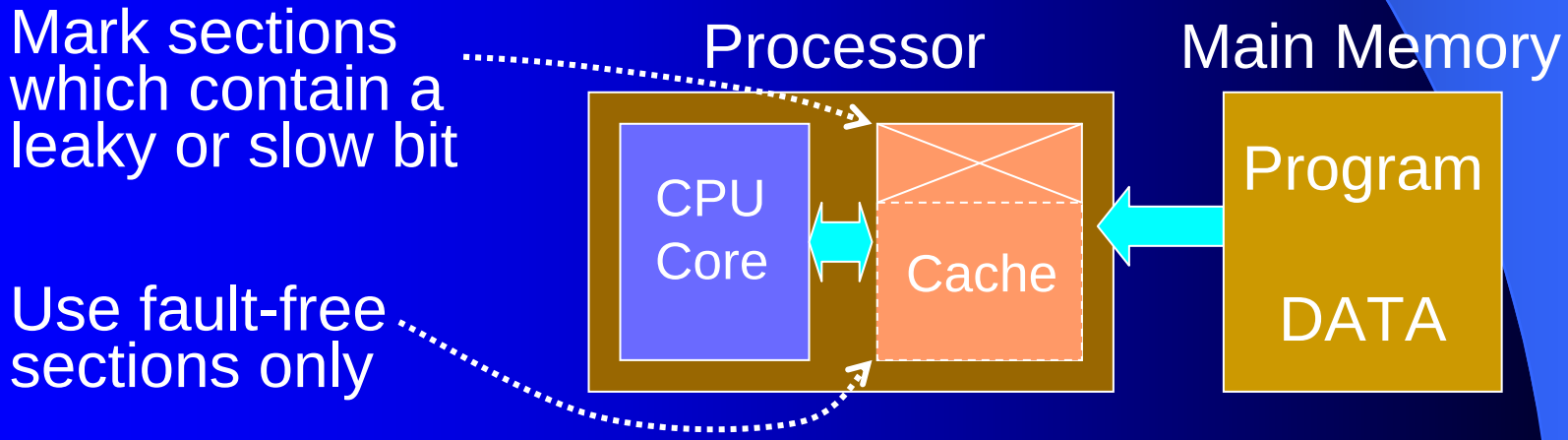
- Use an unused combination of existing flag bits to indicate a slow SRAM cell in a specific cache-line.
- Invalidate and skip a cache-line if it is marked.





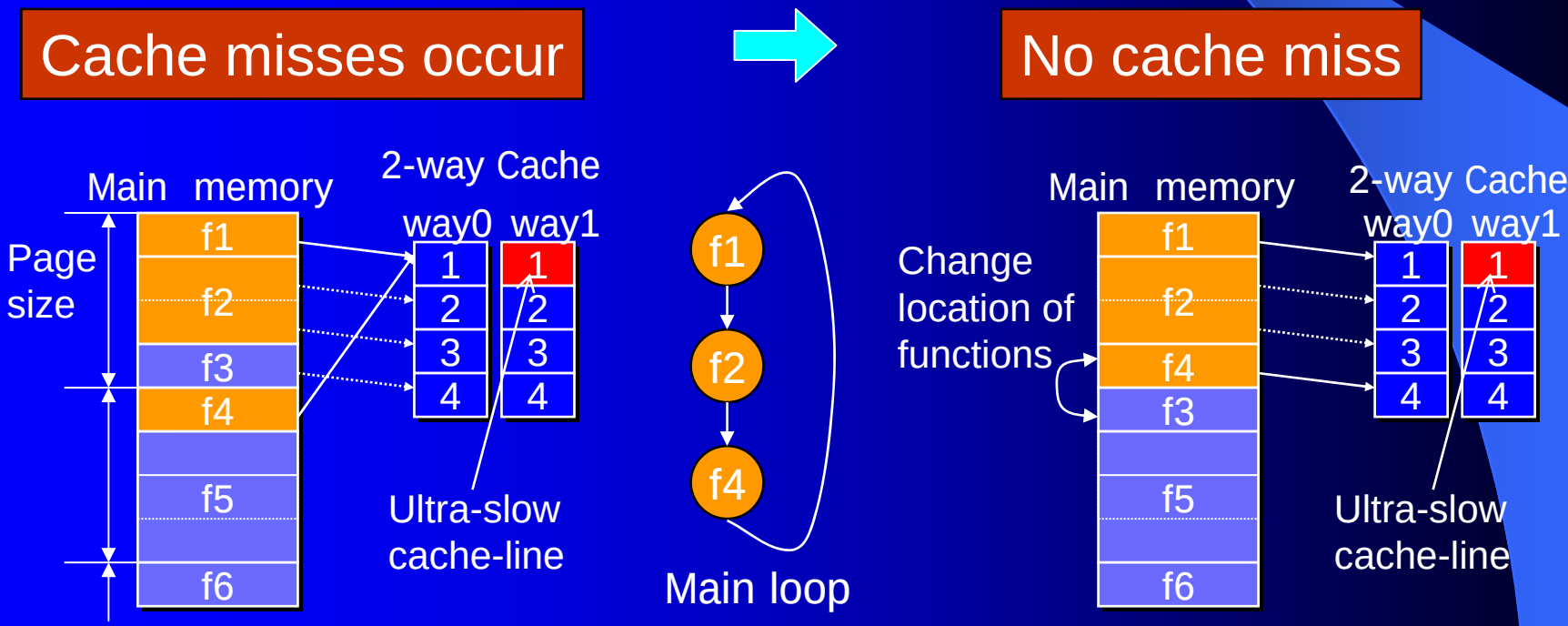
Cache Miss Reduction

- Using a smaller cache memory does not affect the correct operation of a processor.
- The problem is an increase of a cache miss rate due to a reduced cache size.



Our Approach

- Modify the order of functions in the address space such that ultra-slow cache-lines are not accessed frequently.



Process Variation at 90nm

$$I_{Subthreshold} \propto \frac{W \cdot V_T^2}{T_{ox} \cdot L} \cdot \exp\left(\frac{-V_{th}}{\alpha \cdot V_T}\right)$$

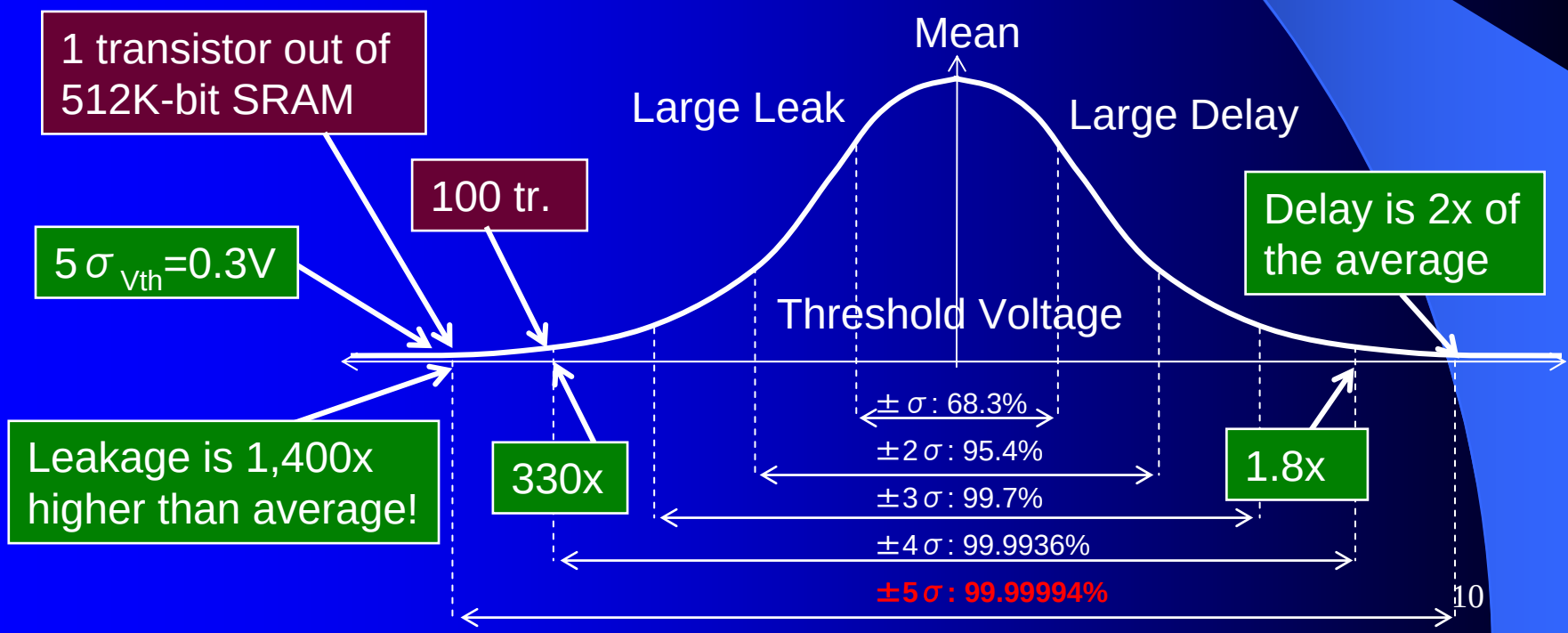
V_T: Thermal voltage (25mV@room temperature)

α : Sub-threshold factor (1.40~1.65)

T_{ox}: Oxide thickness

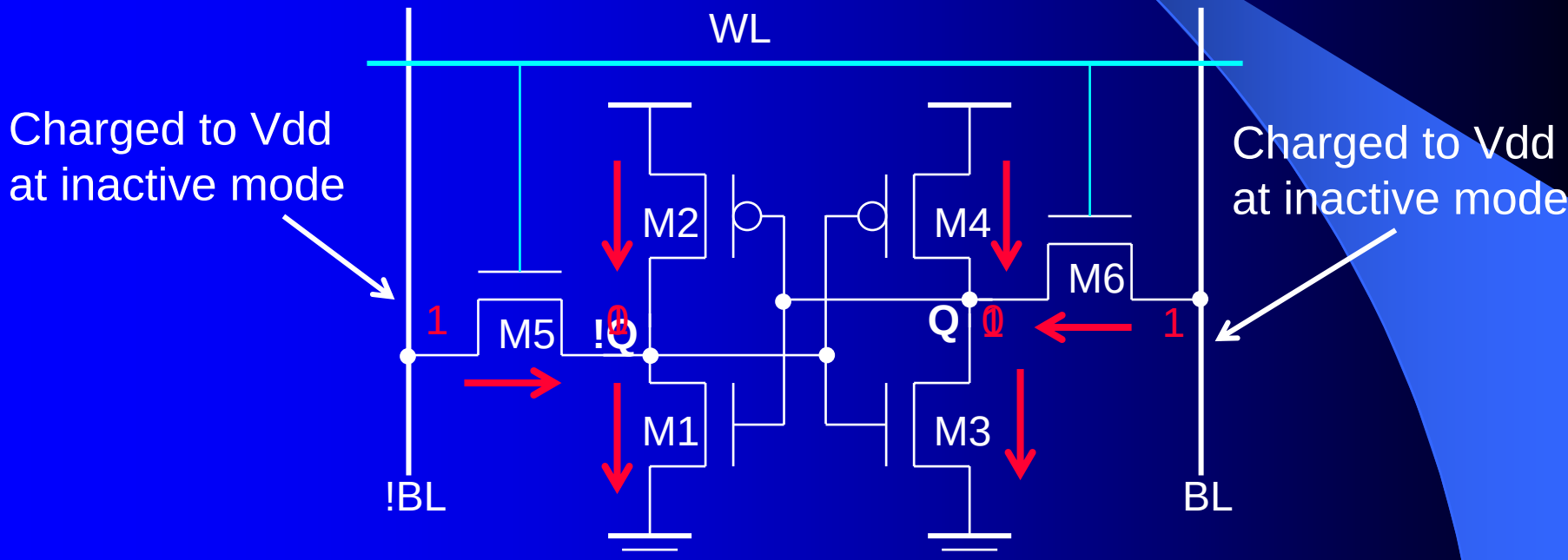
Year	min. L [nm]	$^1V_{TH}$ [V]	$^2V_{TH}$ [V]
2004	37 (90)	0.32	0.12
2005	32 (80)	0.33	0.09
2006	28 (70)	0.34	0.06

1: Low Operating Power Process 2: MPU process



Masking Leaky Transistors

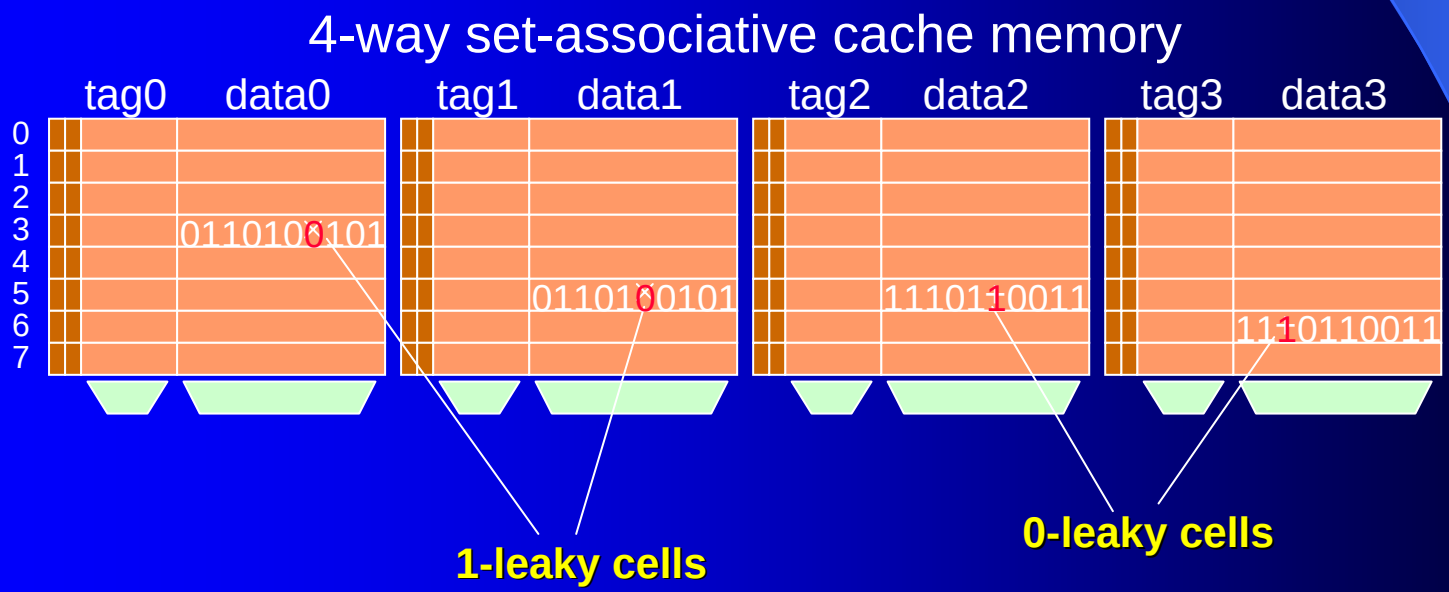
Leakage current of a SRAM cell depends on the logic value stored



If M2, M3, or M5 is ultra-leaky, the SRAM cell is 1-leaky
If M1, M4, or M6 is ultra-leaky, the SRAM cell is 0-leaky

Masking Leaky Cache-Lines

1. Skip using leaky cache lines by marking the lines and store “leakage-safe” values to the leaky cache lines
2. Modify the order of instruction codes considering binary expressions of the codes and locations of 0/1-leaky bits in a cache so that the total leakage current is minimized



Leakage-Aware Scheduling

Instruction memory

	br …
1000	1111 0000
1001	0000 1111
1002	0001 1000
	jc …
	before rescheduling

Instruction cache
(Direct-map 512-set, 1-byte line size)

488	01** ***1
489	**1* 00**
490	0*0* 01*1

Instruction-to-Cell matching table

	488	489	490
1000	1	3	1
1001	2	0	4
1002	1	1	2

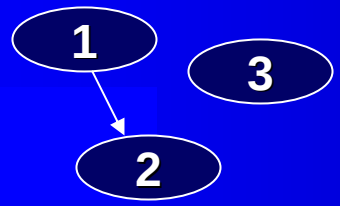
Instruction memory

	br …
1000	0001 1000
1001	1111 0000
1002	0000 1111
	jc …
	after rescheduling

Leakage-preference of cache cells:

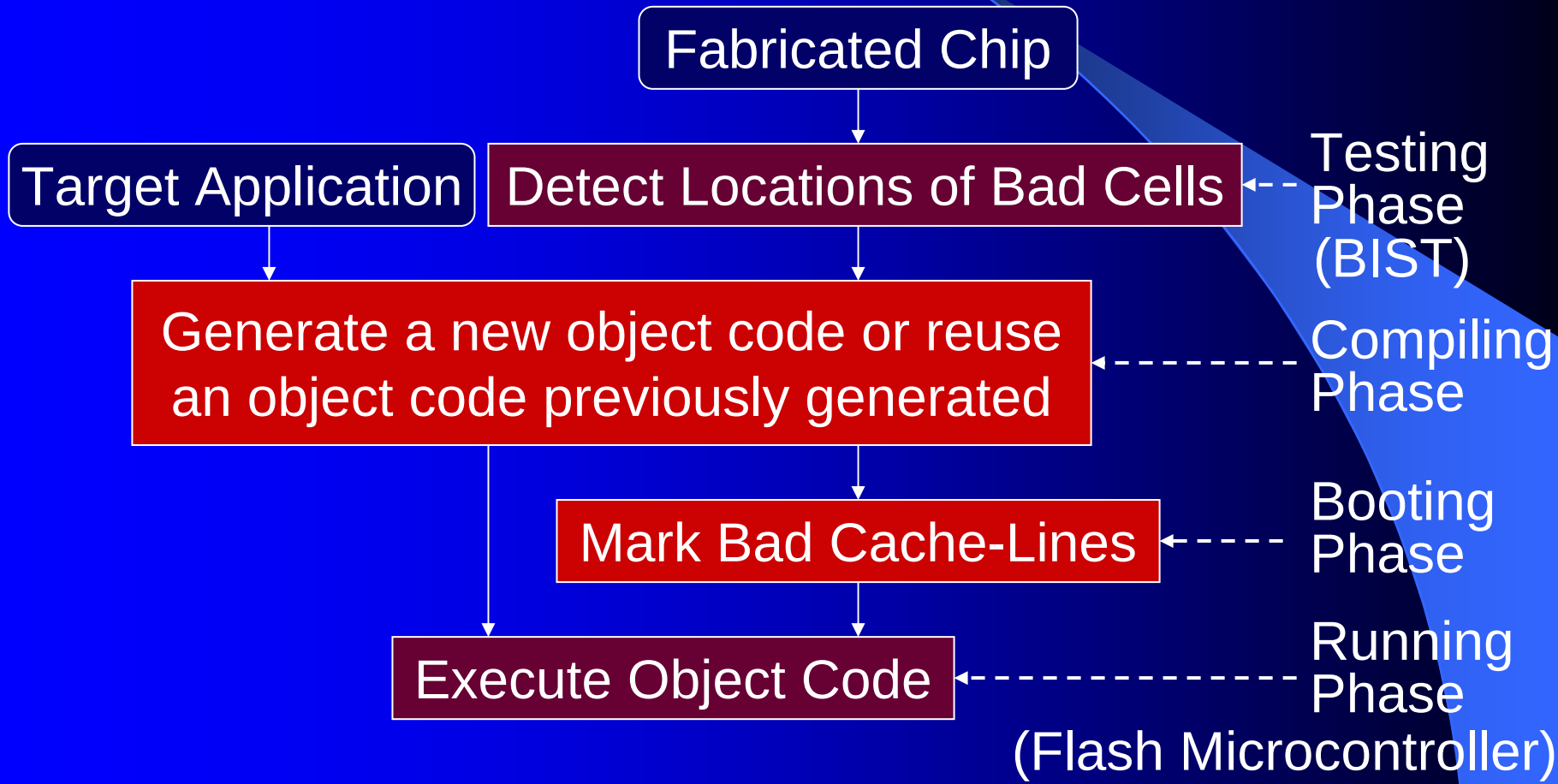
1: the cell prefers 1 (i.e., leaks less if storing 1)
0: the cell prefers 0 (i.e., leaks less if storing 0)
*: the cell not having preference

Instructions dependencies graph

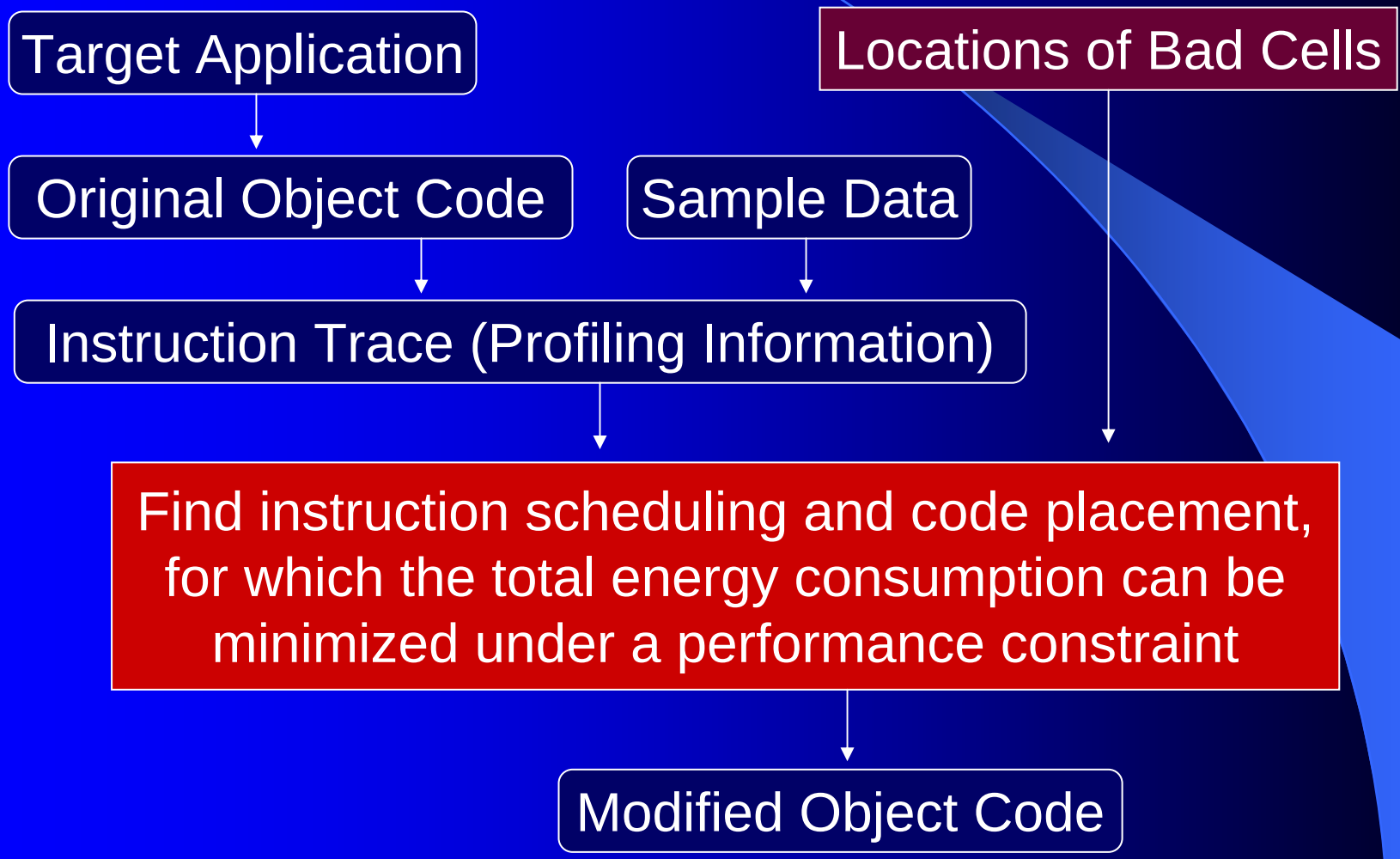


Possible schedules	# matches	Leakage improvement	
		# mismatches	%
1-2-3 (original)	1+0+2=3	2+3+3=8	0
1-3-2	1+1+4=6	2+2+1=5	38%
3-1-2	1+3+4=8	2+0+1=3	63%

The Flow

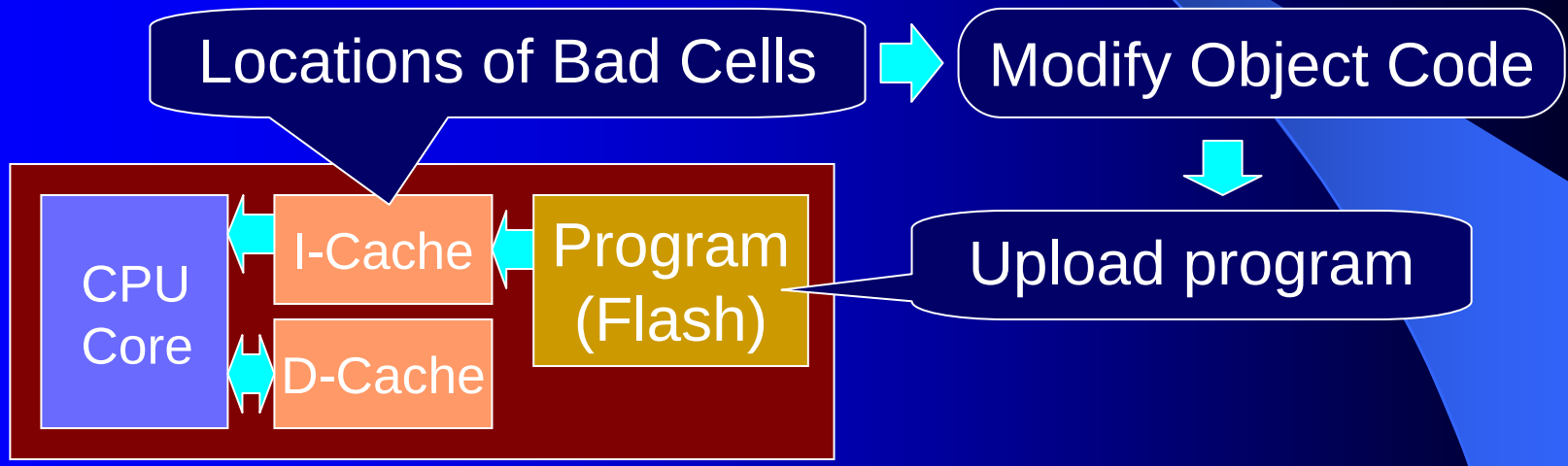

Fabricated Chip
Target Application
Detect Locations of Bad Cells
Testing Phase (BIST)
Generate a new object code or reuse an object code previously generated
Compiling Phase
Mark Bad Cache-Lines
Booting Phase
Execute Object Code
Running Phase
(Flash Microcontroller)

Compiler Optimization Flow



New Paradigm

- Use different object codes for different chips

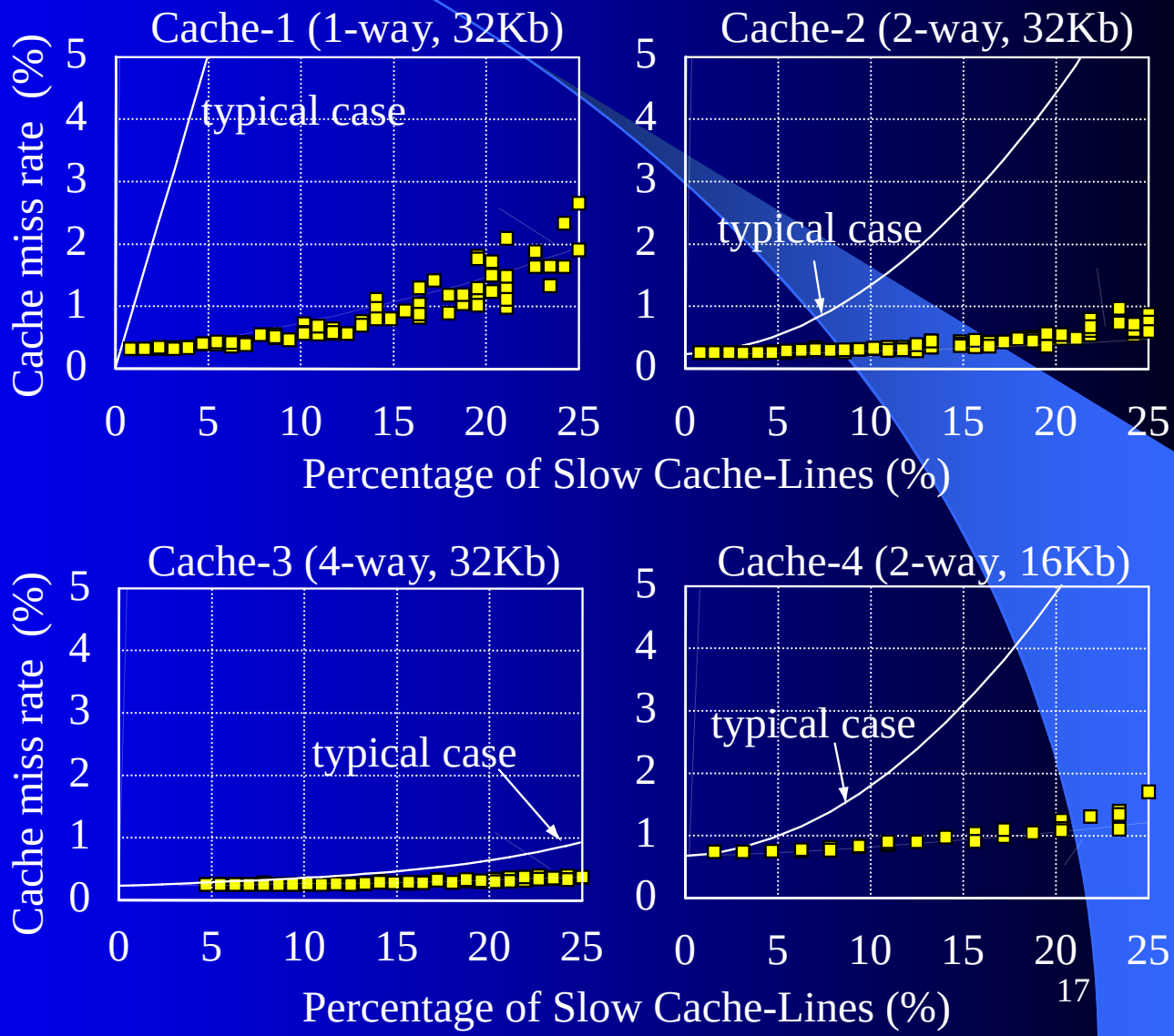


- Future work: Reducing the cost involved

Code Placement Results

Target CPU:
ARMv4T architecture

Benchmark:
MPEG2 encoder



Inst. Scheduling Results

Algorithm	Benchmark	Cache configurations (sets x ways x line size)			
		512x1x4	256x2x4	128x4x4	64x8x4
Exhaustive Search	MPEG2	10.85	7.68	5.48	3.92
	FFT	10.74	7.39	5.44	4.26
	JPEG	8.59	5.69	2.81	1.12
	Compress	10.96	6.84	6.01	5.27
	FIR	12.51	12.98	12.57	12.55
	Average	**10.73**	**8.12**	**6.46**	**5.42**
List Scheduling	MPEG2	9.05	6.35	4.61	3.45
	FFT	8.80	6.01	4.56	3.61
	JPEG	6.34	4.20	2.10	0.78
	Compress	10.21	5.94	3.05	2.65
	FIR	11,90	12.30	11.86	11.75
	Average	**9.26**	**6.96**	**5.24**	**4.45**

Leakage reduction in a cache memory (%)

Inst. Scheduling Results

Algorithm	Benchmark	Cache configurations (sets x ways x line size)			
		512x1x4	256x2x4	128x4x4	64x8x4
Exhaustive Search	MPEG2	220.10	366.43	672.09	1086.61
	FFT	107.46	129.75	249.39	290.74
	JPEG	150.09	169.89	133.43	132.46
	Compress	91.98	94.84	93.51	187.86
	FIR	9.00	17.66	33.88	67.80
	Average	**115.73**	**155.71**	**272.20**	**394.40**
List Scheduling	MPEG2	0.06	0.06	0.07	0.09
	FFT	0.03	0.03	0.04	0.05
	JPEG	0.06	0.05	0.04	0.04
	Compress	0.03	0.03	0.02	0.02
	FIR	0.00	0.00	0.00	0.00
	Average	**0.03**	**0.03**	**0.04**	**0.04**

* Computational time on 3.8GHz Xeon processor with 3.5GB memory (sec.)

Summary

- Cancel the degradation of cache hit-rate even in presence of 25% slow cache-lines.
- Worst case delay can be reduced by 15%.
- Higher V_{th} (lower leakage) can be used without any performance degradation
- Leakage power can be reduced by 10%.
- No major HW modification is required.

Future work

- Reduction of test and recompilation costs

Conclusion

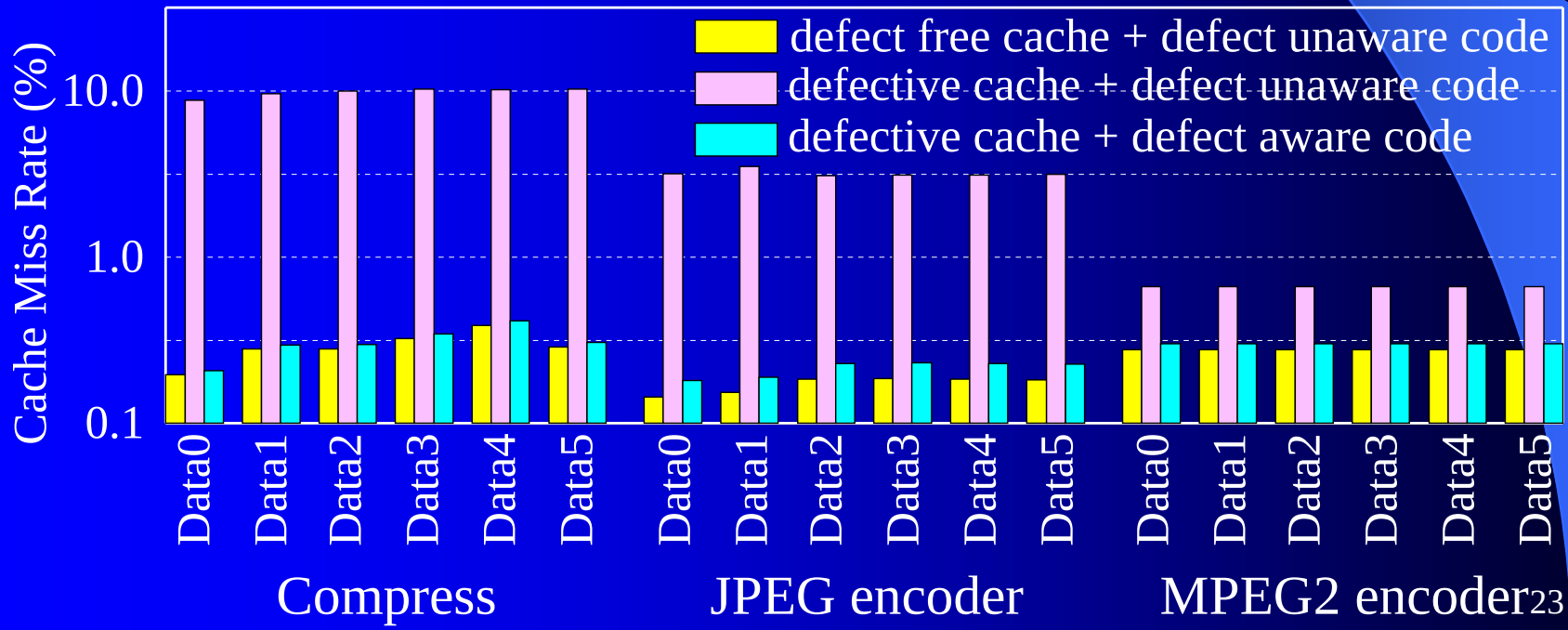
- Chips implementing the same design are no longer the same!
  - Traditional example: Frequency binning
  - Intensifies with each new technology node
- Per-chip customization now reasonable
  - Expensive as a manufacturing post-processing step
- Solution:
  - Let the RTOS customize the application to the chip
  - Variation-Aware Self-Calibrating RTOS

1. M. Goudarzi, T. Ishihara, H. Yasuura, “A Software Technique to Improve Yield of Processor Chips in Presence of Ultra-Leaky SRAM Cells Caused by Process Variation ”, (to appear in) Proc. of ASP-DAC 2007, Jan. 2007
2. T. Ishihara and F. Fallah, “A Cache-Defect-Aware Code Placement Algorithm for Improving the Performance of Processors”, ICCAD 2005, November 2005
3. T. Ishihara and F. Fallah, “A Non-Uniform Cache Architecture for Low Power System Design”, ISLPED 2005, August 2005
4. T. Ishihara and F. Fallah, “A Code Placement Technique for Improving the Performance of Processors with Defective Caches”, IWLS 2005, June 2005

Thank you!!

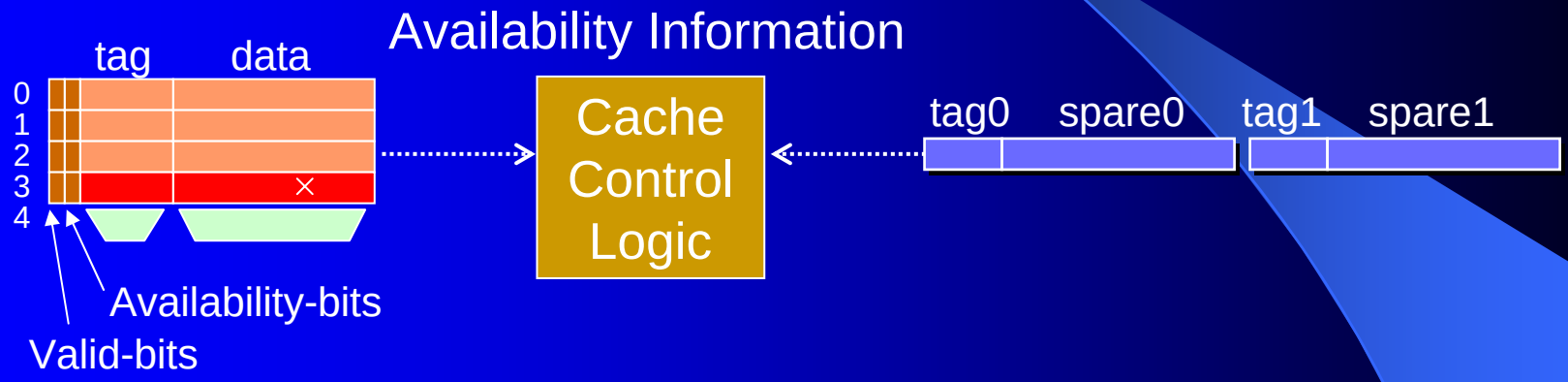
Input Data Dependency

- Compared cache miss rates for 6 different input values.
- The optimized code for Data0 achieves very good results for other input values too.

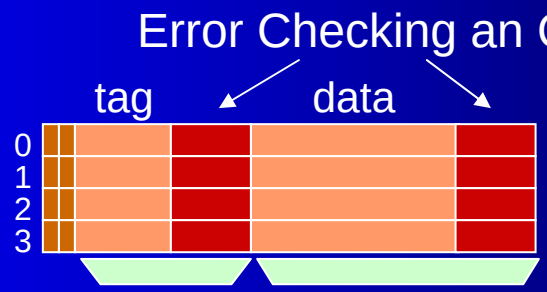


Previous Work (1/2)

- Vergos et al. proposed a technique using spare cache

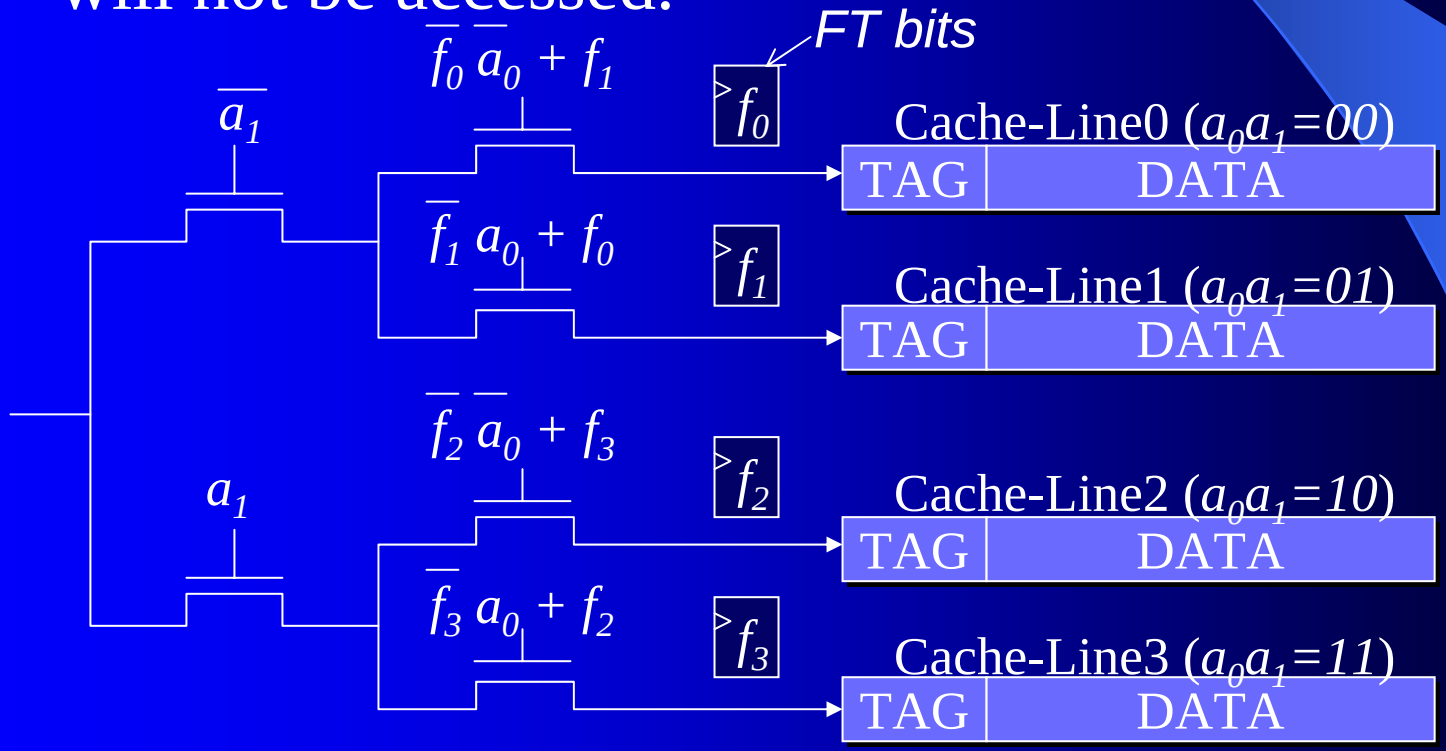


- Sohi proposed a technique using error correcting code



Previous Work (2/2)

- Shiravani et al. proposed *PADded* cache
  - Customize an address decoder so that faulty blocks will not be accessed.



Effect of Technology Scaling

- Savings improve in future technologies
  - Results for Exhaustive-Search (max. BB-len=10) applied to MPEG2 on 512x1x4 2KB cache