McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures

Sheng Li, Junh Ho Ahn, Richard Strong, Jay B. Brockman, Dean M Tullsen, Norman Jouppi

MICRO 2009

How do u estimate access time ?

How do u estimate power?

How do you estimate area?

CACTI started in 1994 from the following papers.
Only area and time initially. Power added later.

[1] J. M. Mulder, N. T. Quach, and M. J. Flynn, "An area model for on-chip memories and its application," *IEEE Journal of Solid-State Circuits, Vol. 26, No. 2*, pp. 98–106, Feb. 1991.

[2] T. Wada, S. Rajan, and S. A. Przybylski, "An analytical access time model for on-chip cache memories," *IEEE Journal of Solid-State Circuits, Vol. 27, No. 8*, pp. 1147–1156, Aug. 1992.

How do u estimate access time ?

$$T_{access\_dm} = \max(T_{dataside} + T_{outdrive\_data}, T_{tagside\_dm} + T_{outdrive\_valid})$$

$$T_{cache} = T_{access} + T_{wordline\ delay} + 4 * (\text{inverter delay})$$

$$T_{step} = [R_2C_2 + (R_1 + R_2)C_1] \ln\left(\frac{v_{start}}{v_{end}}\right)$$

$$R_p = \frac{-\text{desired rise time}}{C_{eq} * \ln(0.5)}$$

$$\text{delay} = t_f * \sqrt{\left[\log\left(1 - \frac{v_{th1}}{V_{dd}}\right)\right]^2 + \frac{2t_{rise} * b * v_{th1}}{t_f * V_{dd}}} + t_f\left[\log\left(1 - \frac{v_{th1}}{V_{dd}}\right) - \log\left(1 - \frac{v_{th2}}{V_{dd}}\right)\right]$$

What type of models are these?

Are they accurate?

How could you estimate more accurately?

SPICE

[1] J. M. Mulder, N. T. Quach, and M. J. Flynn, "An area model for on-chip memories and its application," *IEEE Journal of Solid-State Circuits, Vol. 26, No. 2*, pp. 98–106, Feb. 1991.

[2] T. Wada, S. Rajan, and S. A. Przybylski, "An analytical access time model for on-chip cache memories," *IEEE Journal of Solid-State Circuits, Vol. 27, No. 8*, pp. 1147–1156, Aug. 1992.

Power Estimation

- Circuit Level Power Estimation using SPICE
- Gate level power estimation
- The first such tool integrated with Synopsis around 1995 (Power Mill)
- Low Power libraries provided along with high speed libraries for use in synthesis
- Synopsis, Cadence, Magma all have built-in power estimation now
- Power management primitives are being incorporated into HDL level design

Motivation

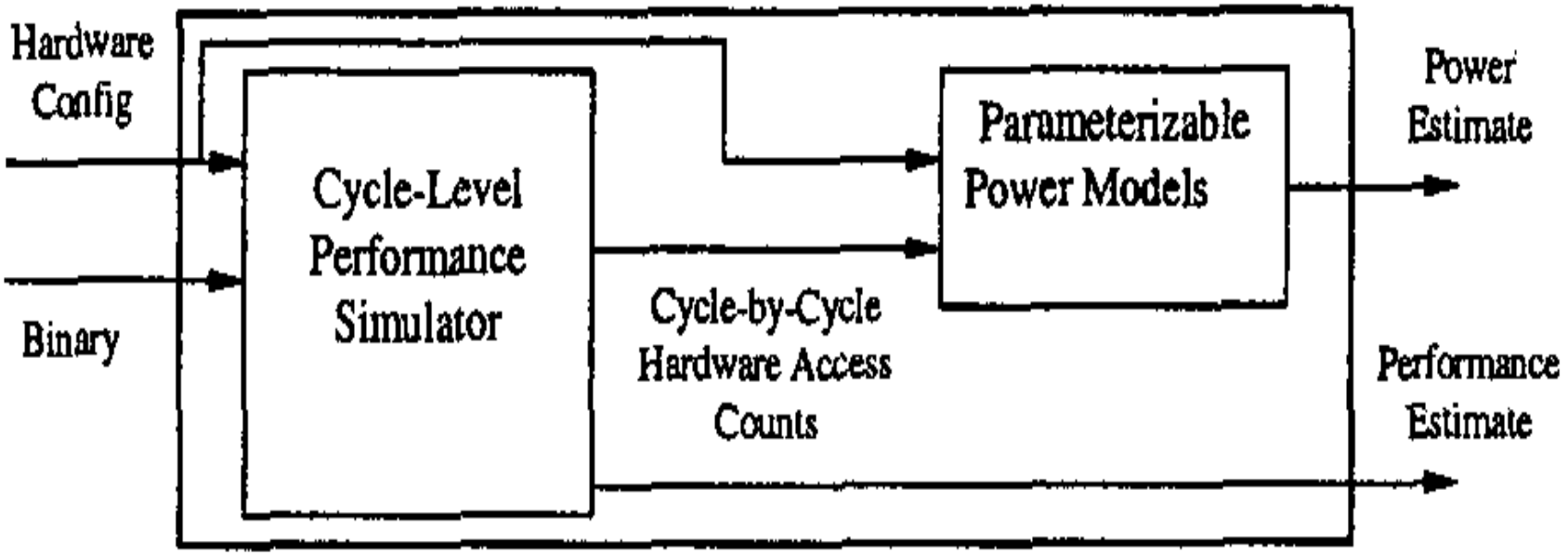
- “Tools both limit and drive research directions”

- New demands
  - Multicore/manycore
  - Evaluate power, timing and area
  - Different source of power dissipation
  - Deep-submicron

Related Work

- Wattch:
  - Enabling a tremendous surge in power-related research
- Orion:
  - Combined Wattch's core power model with a router power model
- CACTI:
  - First tool in rapid power, area, and timing estimation

How did Wattch work?



Wattch methodology

- Array Structures: Data and instruction caches, cache tag arrays, all register files, register alias table, branch predictors, and large portions of the instruction window and load/store queue.
- Fully Associative Content-Addressable Memories: Instruction window/reorder buffer wakeup logic, load/store order checks, and TLBs, for example.
- Combinational Logic and Wires: Functional Units, instruction window selection logic, dependency check logic, and result buses.
- Clocking: Clock buffers, clock wires, and capacitive loads.

Node	Capacitance Equation
Regfile Wordline Capacitance =	$C_{diff}(WordLineDriver)$ + $C_{gate}(CellAccess) * NumBitlines$ + $C_{metal} * WordLineLength$
Regfile Bitline Capacitance =	$C_{diff}(PreCharge)$ + $C_{diff}(CellAccess) * NumWdlines$ + $C_{metal} * BLLength$
CAM Tagline Capacitance =	$C_{gate}(CompareEn) * NumberTags$ + $C_{diff}(CompareDriver)$ + $C_{metal} * TLLength$
CAM Matchline Capacitance =	$2 * C_{diff}(CompareEn) * TagSize$ + $C_{diff}(MatchPreCharge)$ + $C_{diff}(MatchOR)$ + $C_{metal} * MLLength$
ResultBus Capacitance =	$.5 * C_{metal} * NumALU * ALUHeight)$ + $C_{metal} * (RegfileHeight)$

Table 1: Equations for Capacitance of critical nodes.

Hardware Structure	Model Type
Instruction Cache	Cache Array (2x bitlines)
Wakeup Logic	CAM
Issue Selection Logic	Complex combinational
Instruction window	Array/CAM
Branch Predictor	Cache Array (2x bitlines)
Register File	Array (1x bitlines)
Translation Lookaside Buffer	Array/CAM
Load/Store Queue	Array/CAM
Data Cache	Cache Array (2x bitlines)
Integer Functional Units	Complex combinational
FP Functional Units	Complex combinational
Global Clock	Clock

Table 2: Common CPU hardware structures and the model type used by Wattch.

What's wrong with previous work?

- Wattch
  - No timing and area models
  - Only models dynamic power consumption
  - Use simple linear scaling model
  - RUU model from Simplescalar
- Orion2
  - No short circuit power or timing
  - "Incomplete"
- CACTI
  - ???

Contributions

- First integrated power, area, and timing modeling framework
- Model all three types of power dissipation
- Complete, integrated solution for multithreaded and multicore processor power
- Deep-submicron tech. that no longer linear

Overview

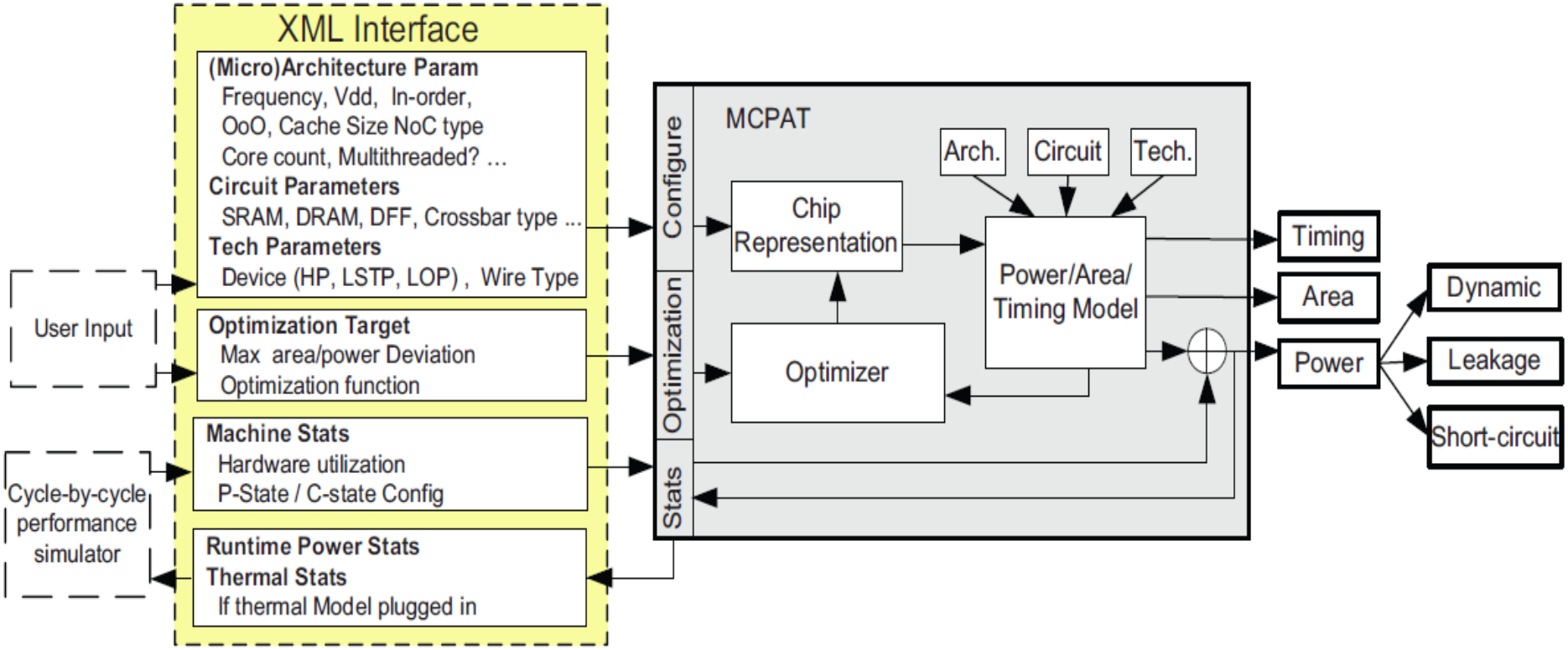


Figure 1: Block diagram of the McPAT framework.

Integrated Approach

Power	**Dynamic: Similar to Wattch** **Short Circuit: IEEE TCAD'00** **Leakage: MASTAR & Intel**
Timing	CACTI with extension
Area	CACTI Empirical modeling for complex logic

Hierarchical Approach

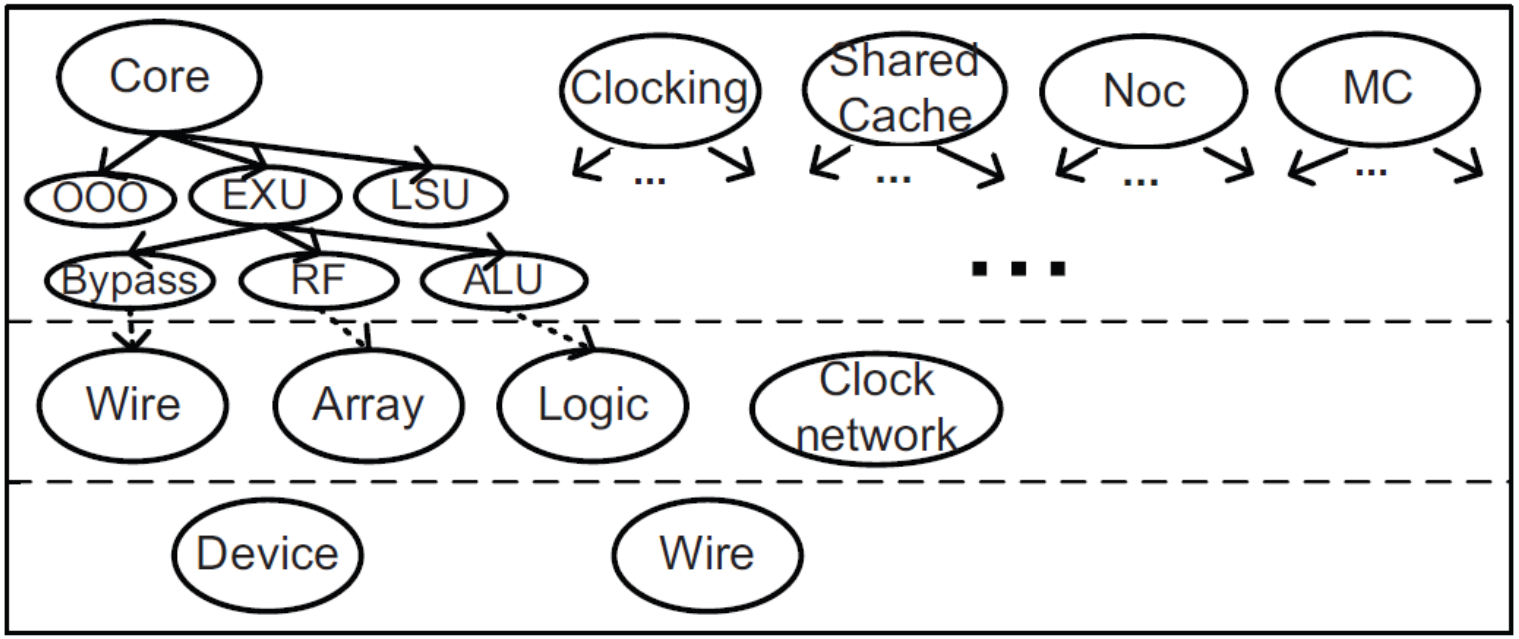


Figure 2: Modeling methodology of McPAT.

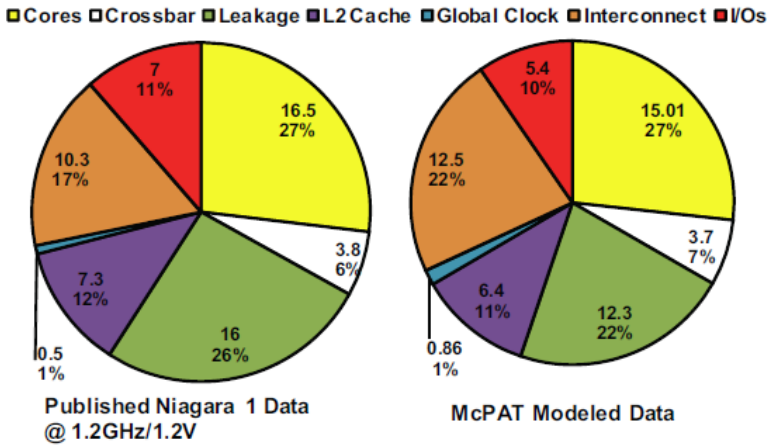
Architecture Level

Core	**Divided into several main units:** **e.g. IFU, EXU, LSU, OOO issue/dispatch unit**
NoC	Signal links and routers
On-Chip Caches	Coherent cache
Memory Controller	3 main hardware structure Empirical model for physical interface (PHY)
Clocking	PLL and clock distribution network Empirical model for PLL power

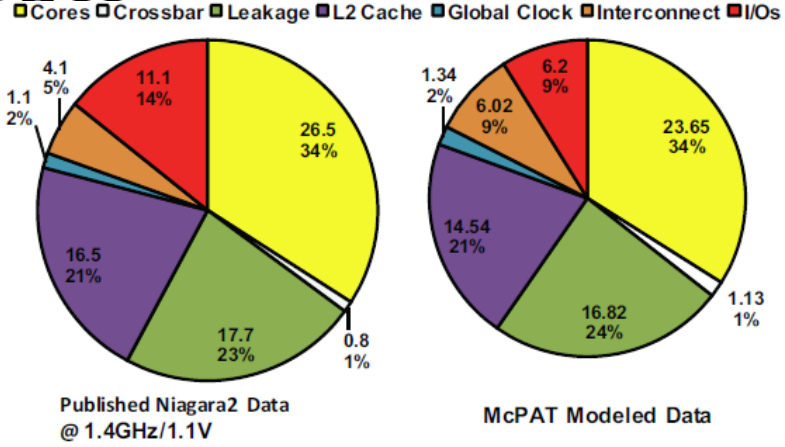
Circuit Level

Wires	**Short wires as one-section Pi-RC model** **Long wires as a buffered wire model**
Arrays	Based on CACTI with extensions
Logic	Highly regular: CACTI Less regular: Model from Intel, AMD and Sun Highly customized: Empirical, from Intel and Sun
Clock Distribution Network	Separate circuit model Global, Domain, and Local

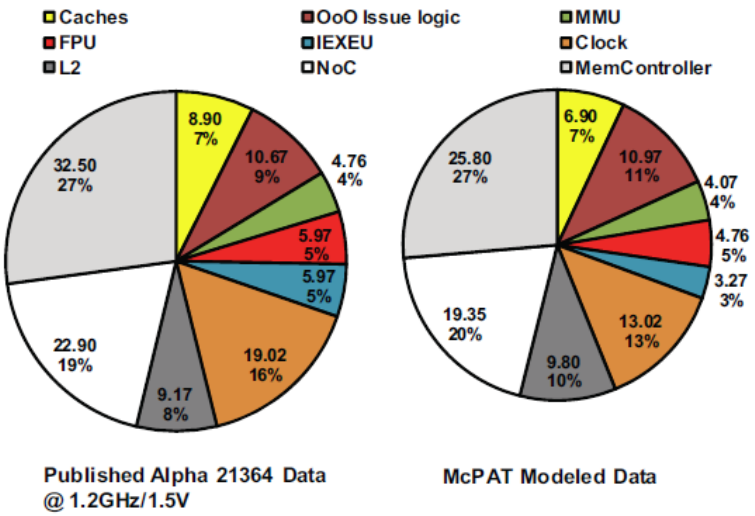
Validation



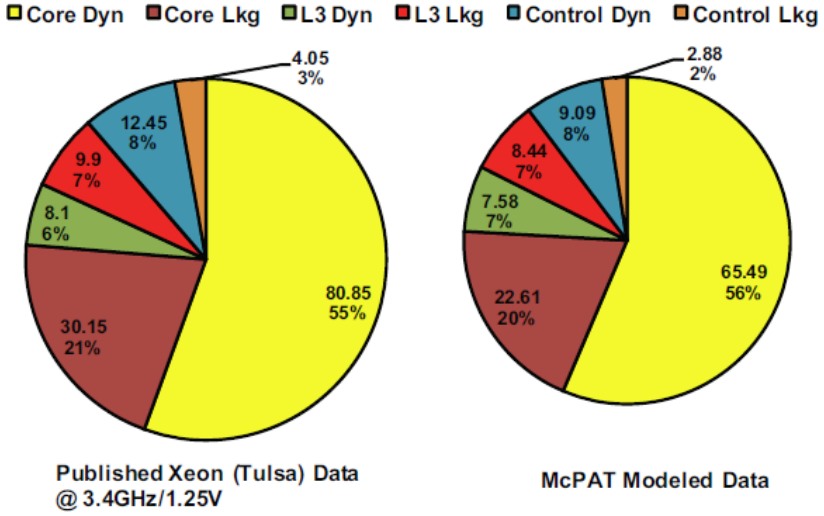
(a) Validation against Niagara processor



(b) Validation against Niagara2 processor



(c) Validation against Alpha 21364 processor



(d) Validation against Xeon Tulsa processor

Validation

Processor	Published total Power and Area	McPAT Results	% McPAT error
Niagara	63 W / 378 mm^2	56.17 W / 295 mm^2	-10.84 / -21.8
Niagara2	84 W / 342 mm^2	69.70 W / 248 mm^2	-17.02 / -27.3
Alpha 21364	125 W / 396 mm^2	97.9 W / 324 mm^2	-21.68 / -18.2
Xeon Tulsa	150 W / 435 mm^2	116.08 W / 362 mm^2	-22.61 / -16.7

Table 1: Validation results of McPAT with regard to total power and area of target processors.

	Average error	1st Contributor Error / % in total pwr
Niagara	1.47W / 23%	74% / 1%
Niagara2	1.87W / 26%	47% / 5%
Alpha 21364	3.4W / 26%	45% / 3%
Xeon Tulsa	4.2W / 17%	29% / 3%

Scaling & Clustering

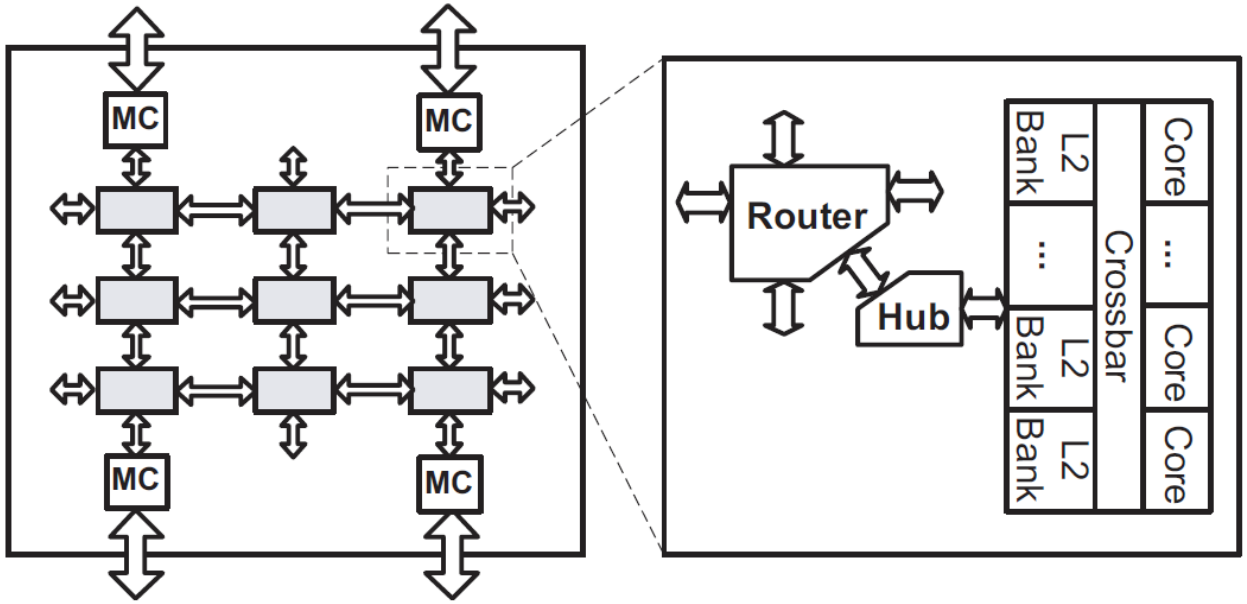


Figure 4: Manycore system architecture. MCs refer to memory controllers.

New generation:

Double # of cores

Keep the same micro-architecture

Parameters and Benchmarks

Parameters	90nm	65nm	45nm	32nm	22nm
Clock rate (GHz)	2.0	2.3	2.7	3.0	3.5
The number of cores	4	8	16	32	64
The number of memory controllers	2	3	4	6	8
Memory capacity per channel (GB)	2	4	4	8	8
Main memory type	DDR2-667	DDR3-800	DDR3-1066	DDR3-1333	DDR3-1600

Table 2: Parameters of the manycore architecture across technology generations. Each memory controller has one memory channel.

SPLASH-2

Application	Dataset	Application	Dataset
Barnes	16K particles	Cholesky	tk17.O
FFT	1024K points	Radiosity	room
FMM	16K particles	Raytrace	car
LU	512×512 matrix	Volrend	head
Ocean	258×258 grids		
Radix	8M integers		
Water-Sp	4K molecules		

SPEC CPU2006

Set	Applications
CINT	
high	429.mcf, 462.libquantum, 471.omnetpp, 473.astar
med	403.gcc, 445.gobmk, 464.h264ref, 483.xalancbmk
low	400.perlbench, 401.bzip2, 456.hmmer, 458.sjeng
CFP	
high	433.milc, 450.soplex, 459.GemsFDTD, 470.lbm
med	410.bwaves, 434.zeusmp, 437.leslie3d, 481.wrf
low	436.cactusADM, 447.dealII, 454.calculix, 482.sphinx3

Table 3: SPLASH-2 datasets and SPEC 2006 application mixes for high, med, and low memory bandwidth.

Area and Power

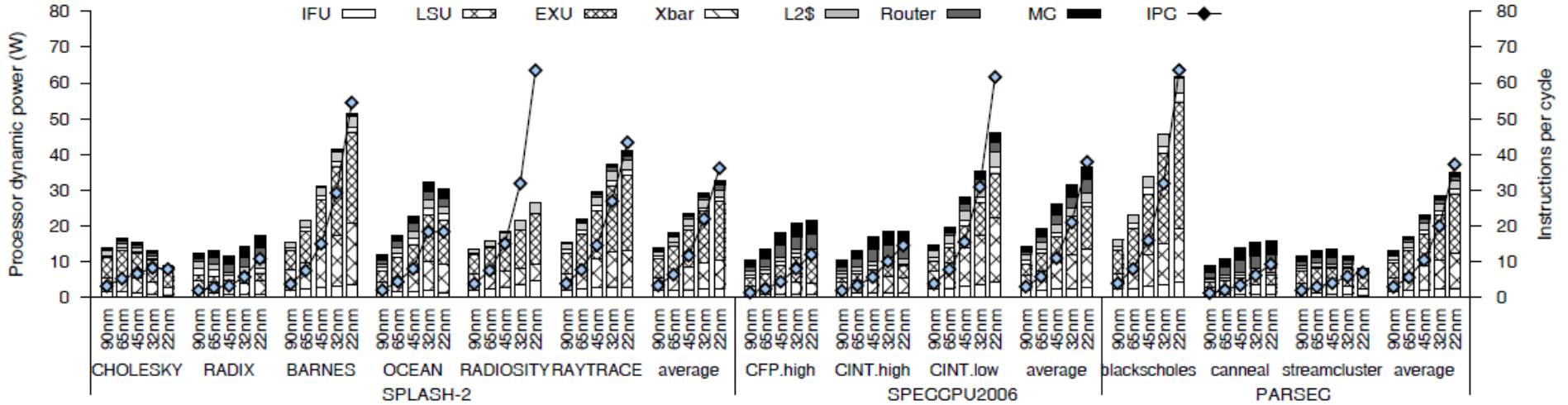
	NoC size	Die size (mm^2)
1 core per cluster	8×8	239.1
2 cores per cluster	4×8	246.3
4 cores per cluster	4×4	250.6
8 cores per cluster	2×4	278.6

Table 5: NoC sizes and die areas of the manycore architecture at 22nm. The NoC bisection bandwidth is kept constant across configurations.

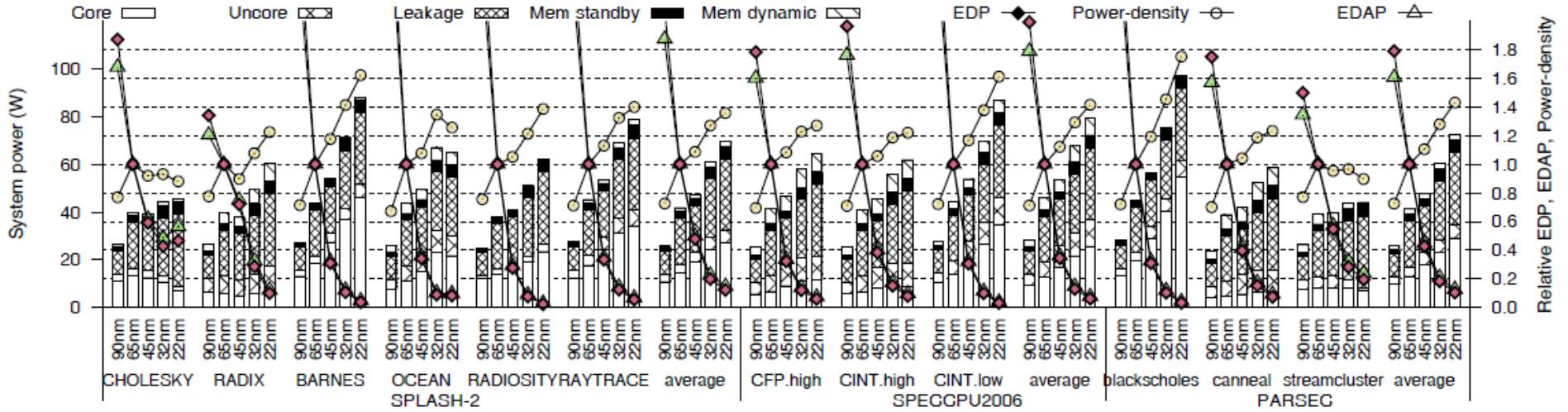
	90nm	65nm	45nm	32nm	22nm
Core area (mm^2)	81.9	96.4	113.4	133.5	157.1
Uncore area (mm^2)	104.3	111.3	102.7	101.6	93.5
Die area (mm^2)	186.3	207.7	216.2	235.1	250.6
Max core dynamic power (W)	24.1	30.7	41.7	48.3	56.4
Max uncore dynamic power (W)	20.6	36.1	45.9	54.5	61.8
Total subthreshold leakage (W)	6.5	11.2	17.6	21.5	25.8
Total gate leakage (W)	2.6	6.7	0.7	1.6	2.5
Chip max power (W)	53.8	84.8	106.0	125.9	146.7

Table 4: Area and maximum power of configurations with 4 cores per cluster across technology generations.

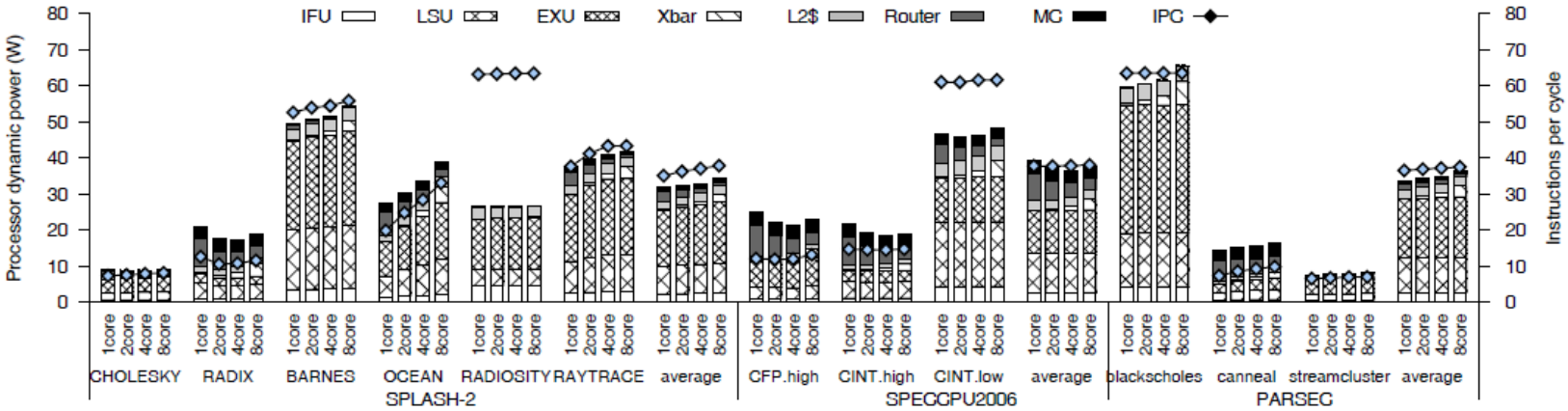
Performance and Efficiency



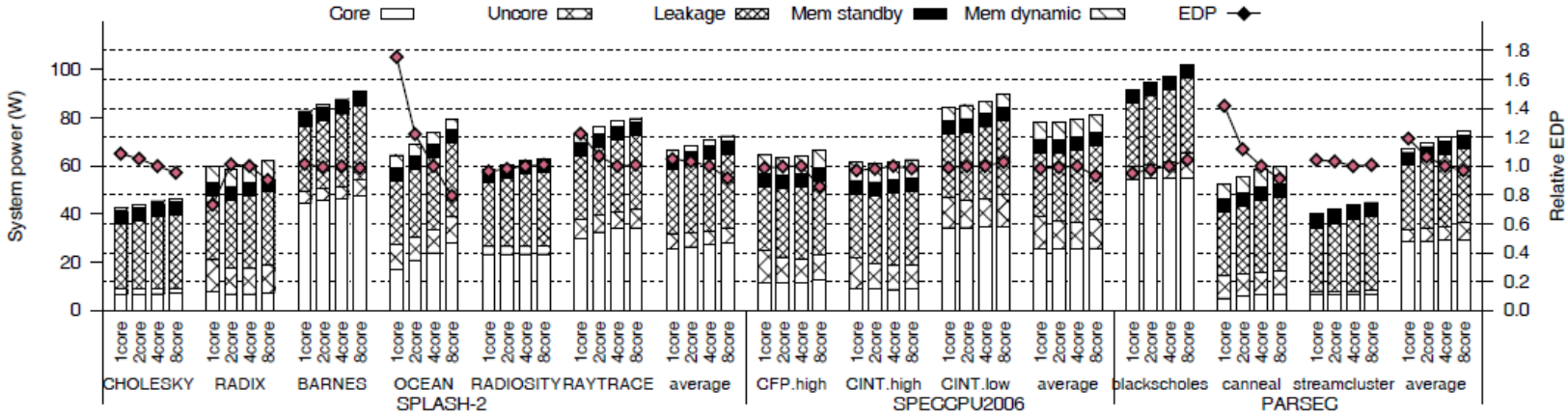
(a) Processor dynamic power and IPC



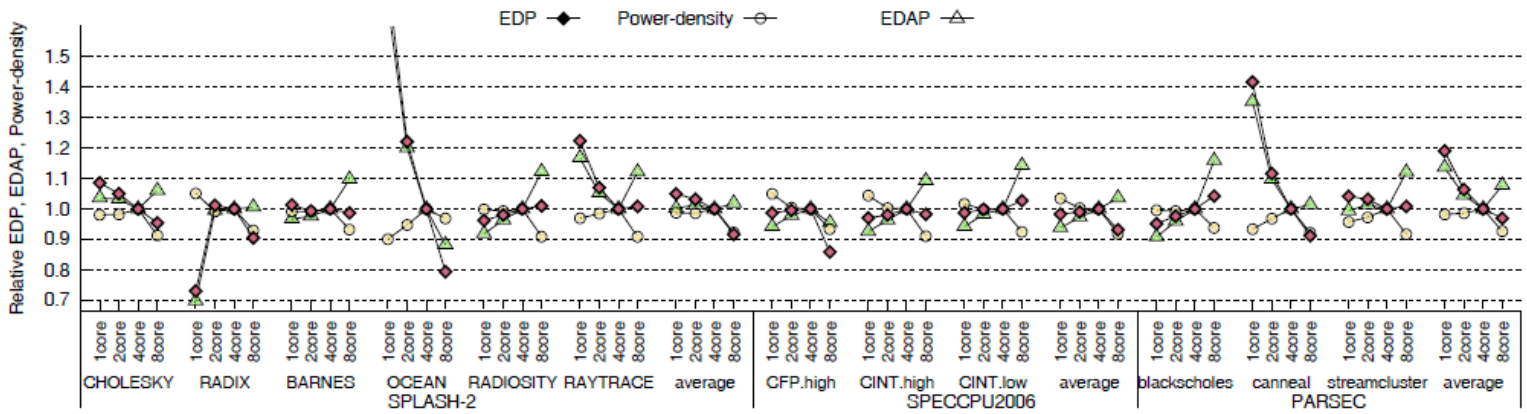
(b) System power, EDP, and EDAP



(a) Processor dynamic power and IPC



(b) System power and EDP



(c) System EDAP, EDP, and power-density

In-order vs. OOO with M5

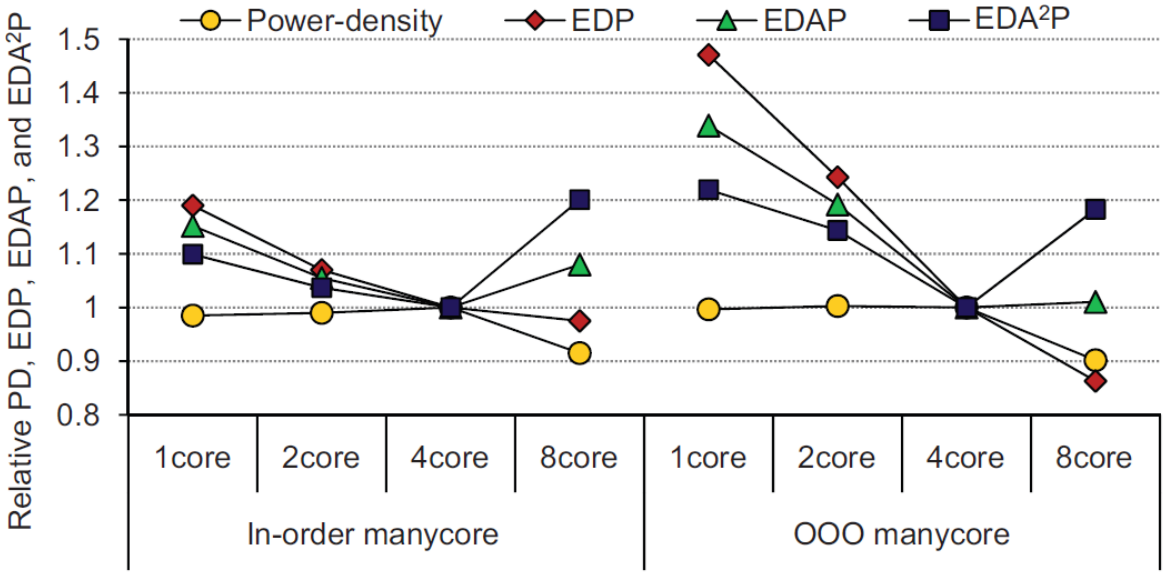


Figure 7: Averaged power-density, EDP, EDAP, and EDA^2P of both in-order and OOO manycore architectures at the 22nm technology node running PARSEC benchmarks. Total numbers of cores/threads are 64/256 and 16/16 for in-order and OOO processors, respectively. The number of cores per cluster is changed from 1 to 8 for both in-order and OOO processors.

Figure of Merit Metrics

1. EDA^2
2. EDA
3. ED
4. ED^2
5. AT
6. AT^2

Power and Energy

- **Power is drawn from a voltage source attached to the V_{DD} pin(s) of a chip.**
- **Instantaneous Power:**

$$P(t) = i_{DD}(t)V_{DD}$$

- **Energy:**

$$E = \int_0^T P(t)dt = \int_0^T i_{DD}(t)V_{DD}dt$$

- **Average Power:**

$$P_{\text{avg}} = \frac{E}{T} = \frac{1}{T}\int_0^T i_{DD}(t)V_{DD}dt$$

Power Dissipation in CMOS ICs

- There are 4 sources of power dissipation in digital ICs

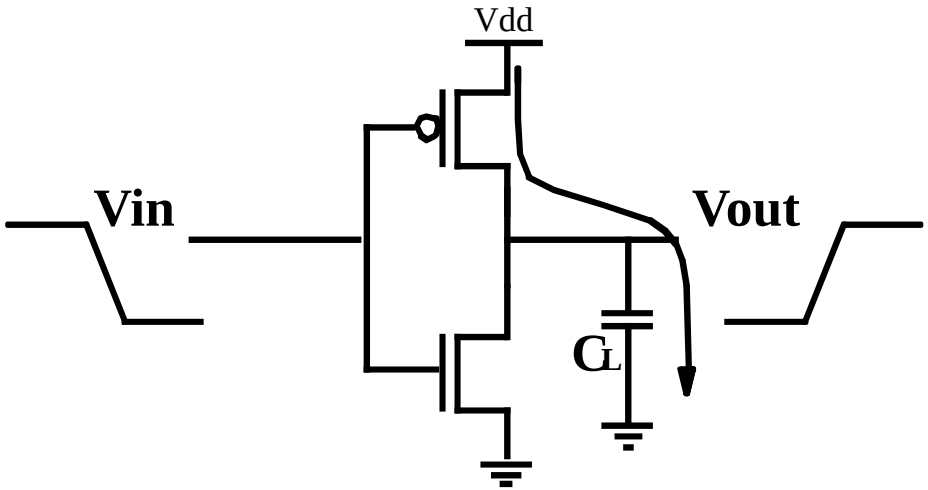
- $P_{avg} = P_{switching} + P_{shortcircuit} + P_{leakage} + P_{static}$

- P_{avg} = Time Averaged Power
- $P_{switching}$ = Switching Component of Power
- $P_{shortcircuit}$ = Short Circuit Component of Power
- $P_{leakage}$ = Leakage Component of Power
- P_{static} = Static Component of Power

Dynamic Power Dissipation

- Caused by the switching of the circuits ($P_{switching} + P_{shortcircuit}$)
- Higher operating frequency -> more frequent switching activities and results in increased power dissipation

Switching Power Dissipation



$$\text{Energy/transition} = C_L * V_{dd}^2$$

$$\text{Power} = \text{Energy/transition} * f = C_L * V_{dd}^2 * f * \alpha$$

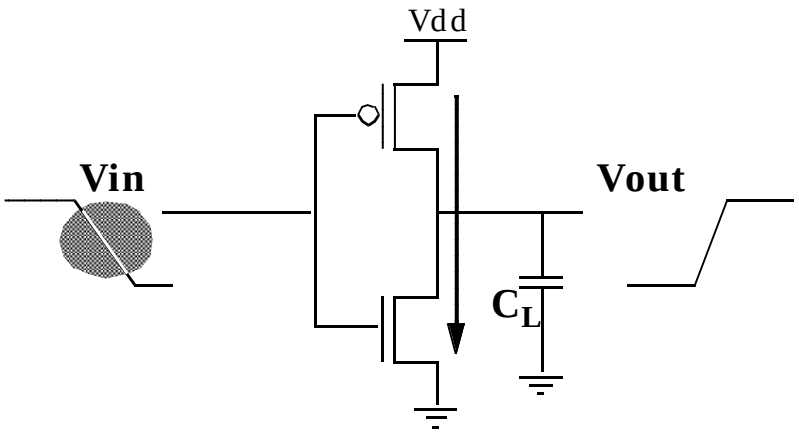
Switching power is required to charge and discharge load capacitances when transistors switch.

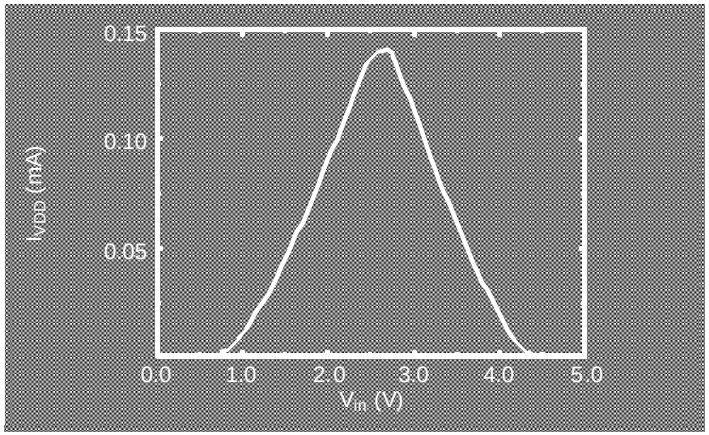
CL = Load Capacitance
f = Clock Frequency
α = Activity Factor

Short Circuit Current

- When transistors switch, both nMOS and pMOS networks may be momentarily ON at once
- Leads to a blip of "short circuit" current.
- < 10% of dynamic power if rise/fall times are comparable for input and output

Short Circuit Current





Power Dissipation in CMOS ICs

Pavg = Pswitching + Pshortcircuit + Pleakage + Pstatic

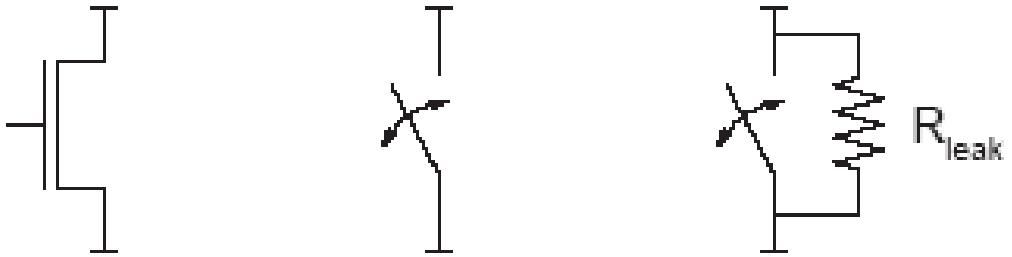
= α CL V VDD f + ISC VDD + Ileakage VDD + Istatic VDD

- **CL = Load Capacitance**
- **V = Voltage Swing (most cases V = VDD)**
- **f = Clock Frequency**
- **ISC = Short circuit current**
- **Ileakage = Leakage Current**
- **Istatic = Static Current**
- **α = Activity Factor**

Leakage Power

- **The power that is dissipated from the devices, when they are in idle state.**
- **Adds to average power, not peak power**
  - **More expensive than dynamic power**
- **Independent of transistor utilization**

CMOS Leakage Current

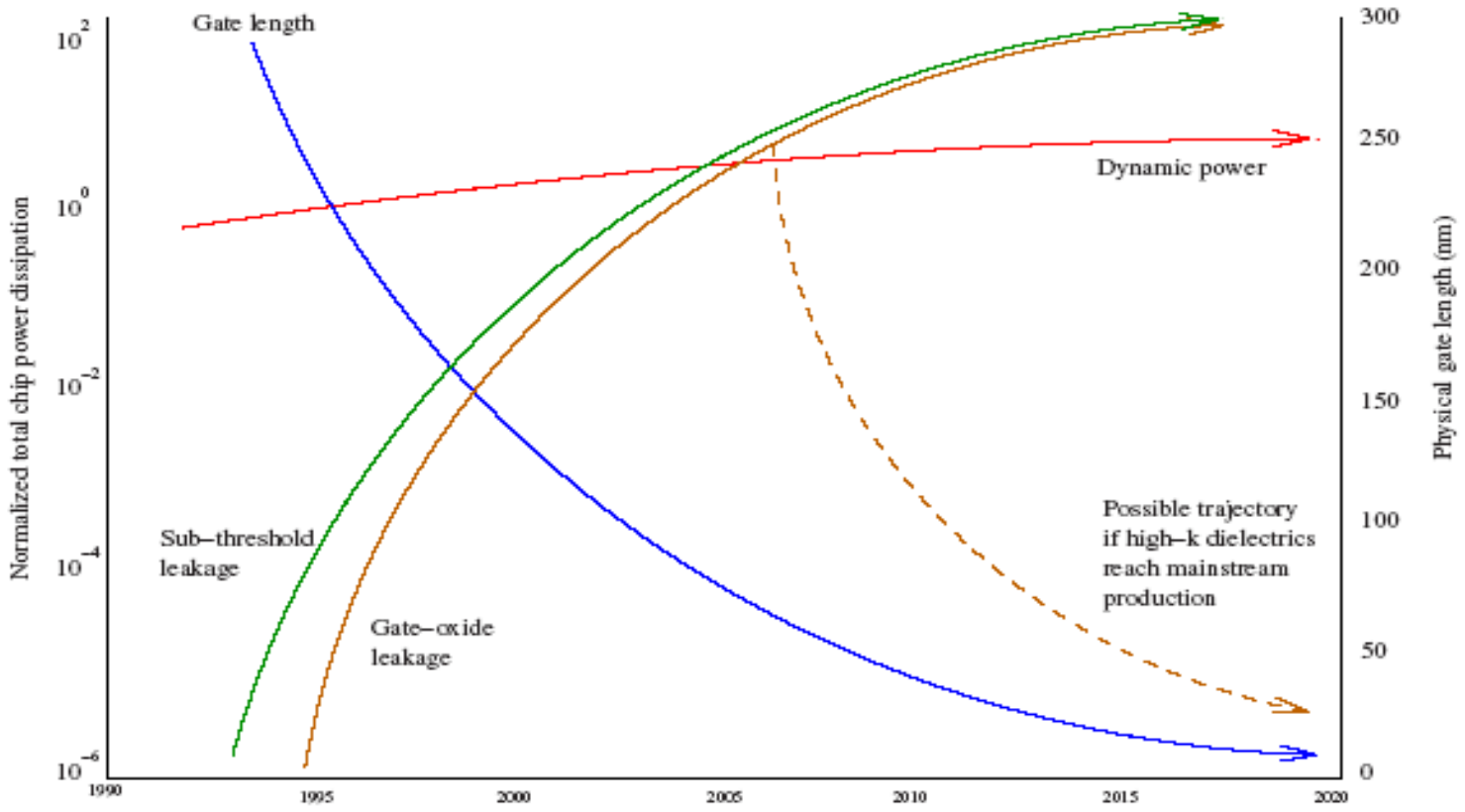


(a) (b) (c)

(a) NMOS-transistor (b) Ideal switch (c) leaking device

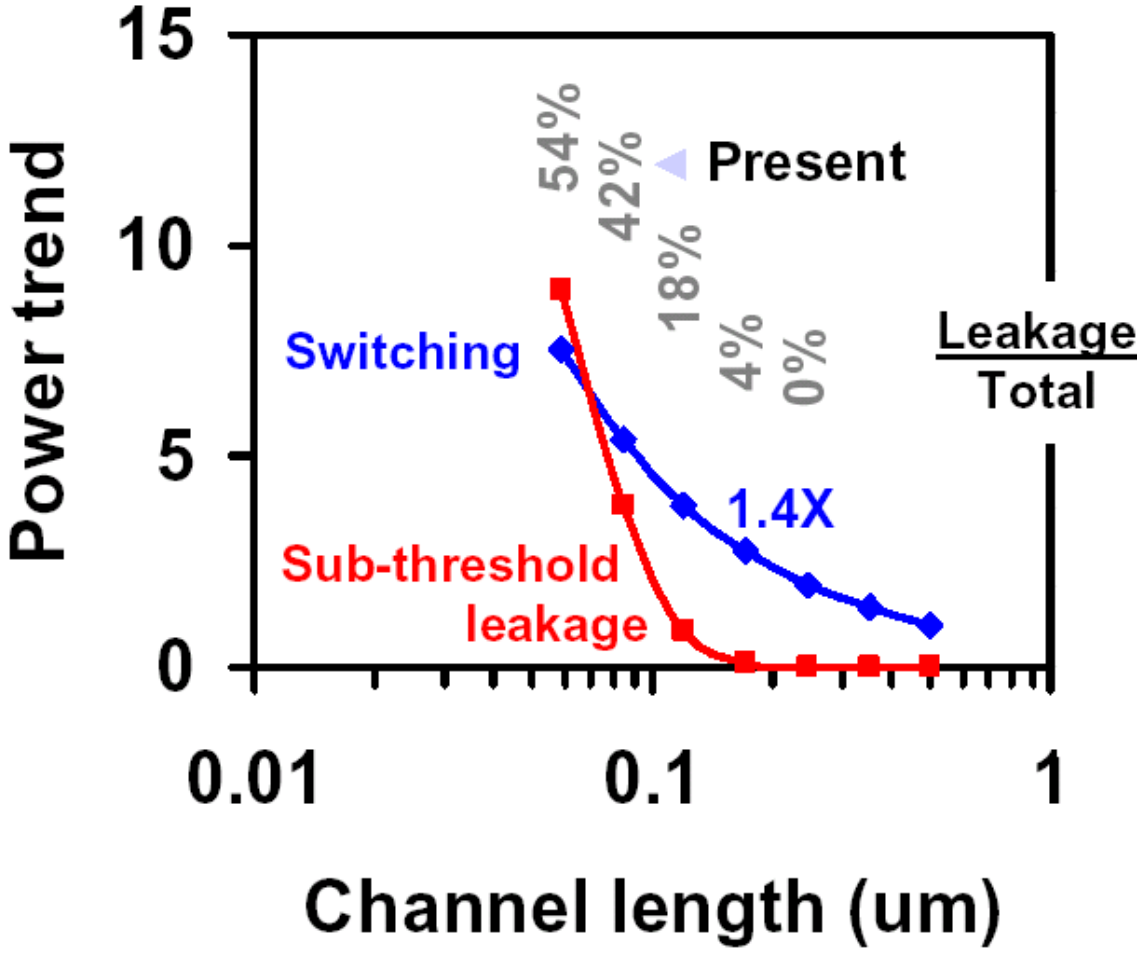
Any guesses on what percent of chip power is leakage power nowadays?

Leakage Power Trends



Source: ITRS

Leakage Power Trends in Nanoscale



Source: Intel

Transistor Leakage Current Mechanisms

- There are 6 different leakage current mechanisms in CMOS devices:
  - Subthreshold leakage current (I_{sub})
    - Weak inversion conduction current
  - Gate Oxide Tunneling Current (I_{gate})
    - Tunneling of electrons from substrate to gate
  - Reverse Biased PN Junction Leakage Current (I_{rev})
    - Current flows into the well from the drain and source
  - Gate Induced Drain Leakage Current (I_{gidl})
    - Due to high field effect in the drain junction
  - Hot Carrier Gate Leakage Current (I_{hot})
    - Due to high electric field near the Si-SiO_2
  - Punch Through Leakage Current (I_{PT})
    - Proximity of the drain and source

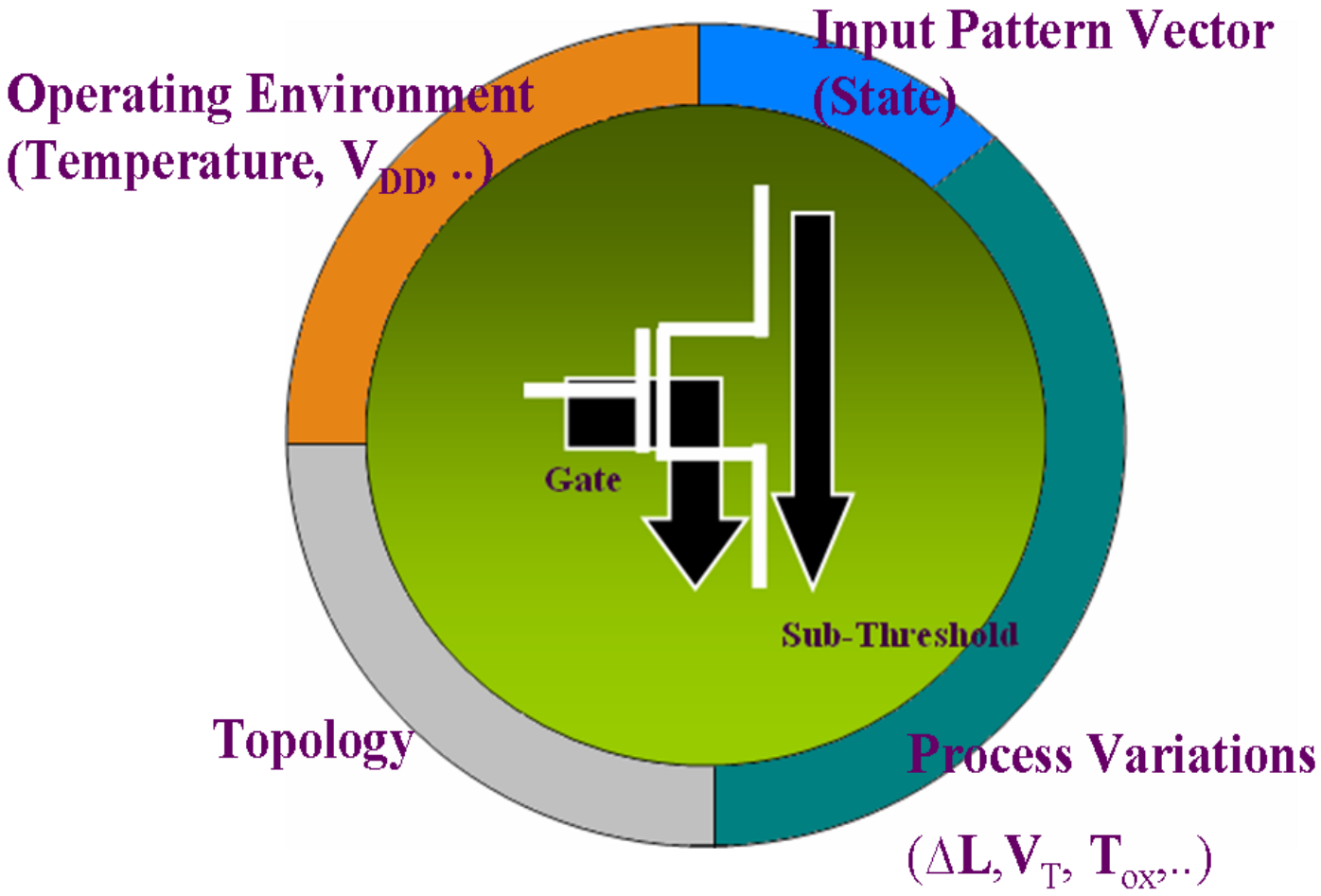
- Subthreshold leakage current and Gate Oxide Tunneling Current are the most significant ones

Subthreshold Leakage Current

- Even though the transistor is logically turned OFF, there is a non-zero leakage current through the channel.

- This current is known as the sub-threshold leakage current because it occurs when the gate voltage is below its threshold voltage

What Affects Leakage Power?



Static Current

- Static CMOS circuits are not supposed to consume constant static power from constant static current flow
- Sometimes designers deviate from CMOS design style → Pseudo NMOS circuits
- This is an example where power is traded for area efficiency. The pseudo NMOS circuit does not require a P-transistor network and saves half the transistors required for logic computation, as compared to CMOS logic
- If an output signal is known to have a very high probability of logic '1' (say 0.99), it may make sense to implement the computation in pseudo NMOS logic

Static Current (contd..)

- The circuit has a special property: The static current flows only when the output is at logic '0'
- Make use of this property in low power design
- In general, the pseudo NMOS circuit is not used on random logic
- For special circuits such as PLAs or register files it is useful due to its efficient area usage
- In such as circuit there is a constant current flow from Vdd to Vss

Power Dissipation in CMOS ICs

Pavg = Pswitching + Pshortcircuit + Pleakage + Pstatic

= α CL V VDD f + ISC VDD + Ileakage VDD + Istatic VDD

- **CL = Load Capacitance**
- **V = Voltage Swing (most cases V = VDD)**
- **f = Clock Frequency**
- **ISC = Short circuit current**
- **Ileakage = Leakage Current**
- **Istatic = Static Current**
- **α = Activity Factor**

Basic Principles of Low Power Design

- **Power reduction can be achieved at most design abstraction levels:**
  - **Material**
  - **Process technology**
  - **Physical Design (Floor Plan, placement & routing)**
  - **Circuit design techniques**
  - **Transistor sizing**
  - **Logic restructuring**
  - **Architecture transformation**
  - **Alternative computation algorithm**

Basic Principles of Low Power Design

- Reduce Voltage
- Reduce Frequency
- Reduce Capacitance
- Reduce activity factor
  - Use of different number representation, counting sequence etc.
  - Alternate logic implementation
- Reduce Static components of power
  - Transistor Sizing
  - Layout Techniques
  - Careful circuit design

Popular Techniques in Power Aware Design

- DVFS (Dynamic Voltage and Frequency Scaling)
- Low Power Modes
- Power Gating
- Clock Gating
- Logic Restructuring
- Compiler Control of Power
- Operating System Management of Power

Power Estimation

- Circuit Level Power Estimation using SPICE
- Gate level power estimation
- The first such tool integrated with Synopsis around 1995 (Power Mill)
- Low Power libraries provided along with high speed libraries for use in synthesis
- Synopsis, Cadence, Magma all have built-in power estimation now
- Power management primitives are being incorporated into HDL level design