Embedded System 1

November 2018

POLITECNICO DI MILANO





Run-Time Resources and Power Management

William Fornaciari and Davide Zoni

Politecnico di Milano – Dipartimento di Elettronica e Informazione
Milan, Italy

William.fornaciari@polimi.it

Applications & Requirements

- Technology and power trends
- Application scenarios and power/performance trade-off

Basics of Power Management (PM)

- Main components of power consumption
- Methodology vs Management

Architectural Blocks for Power Management

- Hardware mechanisms: Clock Gating, Multi-VDD, Multi-Threshold Logic, Dynamic Voltage and Frequency Scaling (DVFS), Power Gating

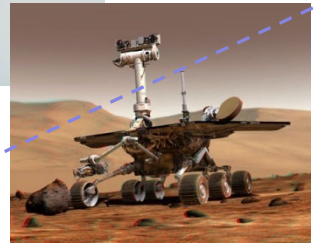
Power Management Techniques

- Software frameworks

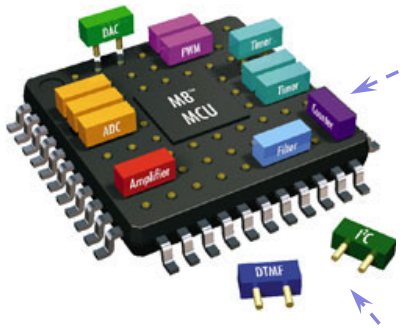
Our Research Directions

LIFE-CRITICAL

REAL-TIME



MOBILE



PROCESSING POWER

Rationale – Power Management Requirements

Needs for even more performance

- Plenty of raw computing power
- Multiple resource types

  (PEs, memories, communication channels)
- Complex resource organization

  (clustered, custom busses)

Applications

- Compete for resources
- QoS specification required
- Different Priority levels

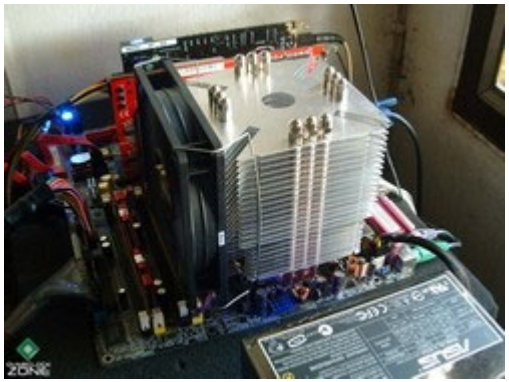
Embedded Systems

- Limited resources
- Optimized power consumption
- Mobile Devices

Rationale – Power Trends

Today some of the most powerful microprocessor chips can dissipate 100-150 Watts, for an average power density of 50-75 Watts per sqare centimeter. Local hot spots on the die can be several times higher than this number.

Low Power Methodology Manual, 2007



For battery-powered , hand-held devices, the numbers are smaller but the problem just as serious. Battery life peaked in 2004. Since then the battery life reduced due as the features have been added faster than power has been reduced

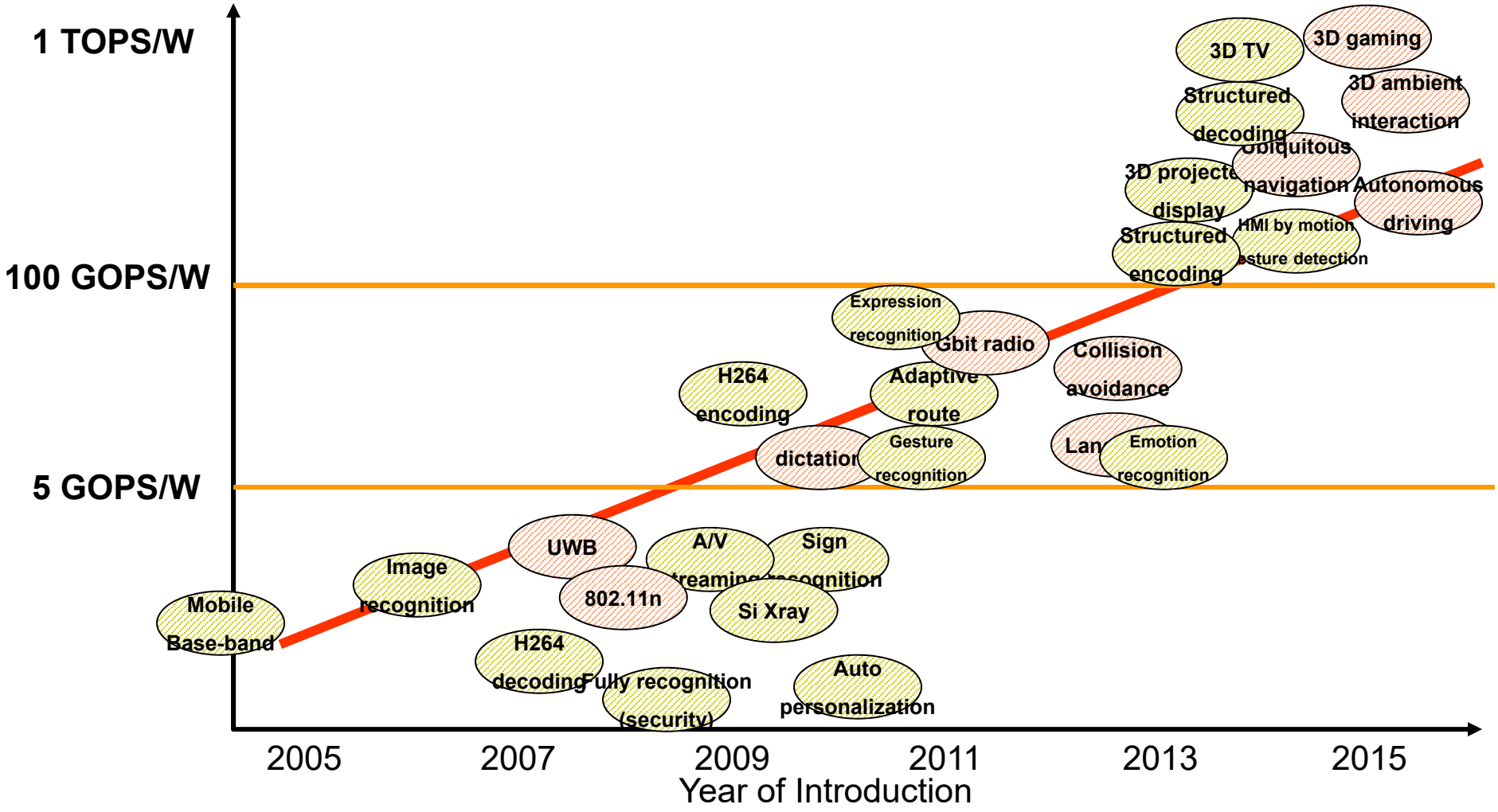
ITRS

The total power consumption of microprocessor chips presents a significant problem for server farms, where infrastructure costs can equal the cost of the computer themselves.

Low Power Methodology Manual, 2007

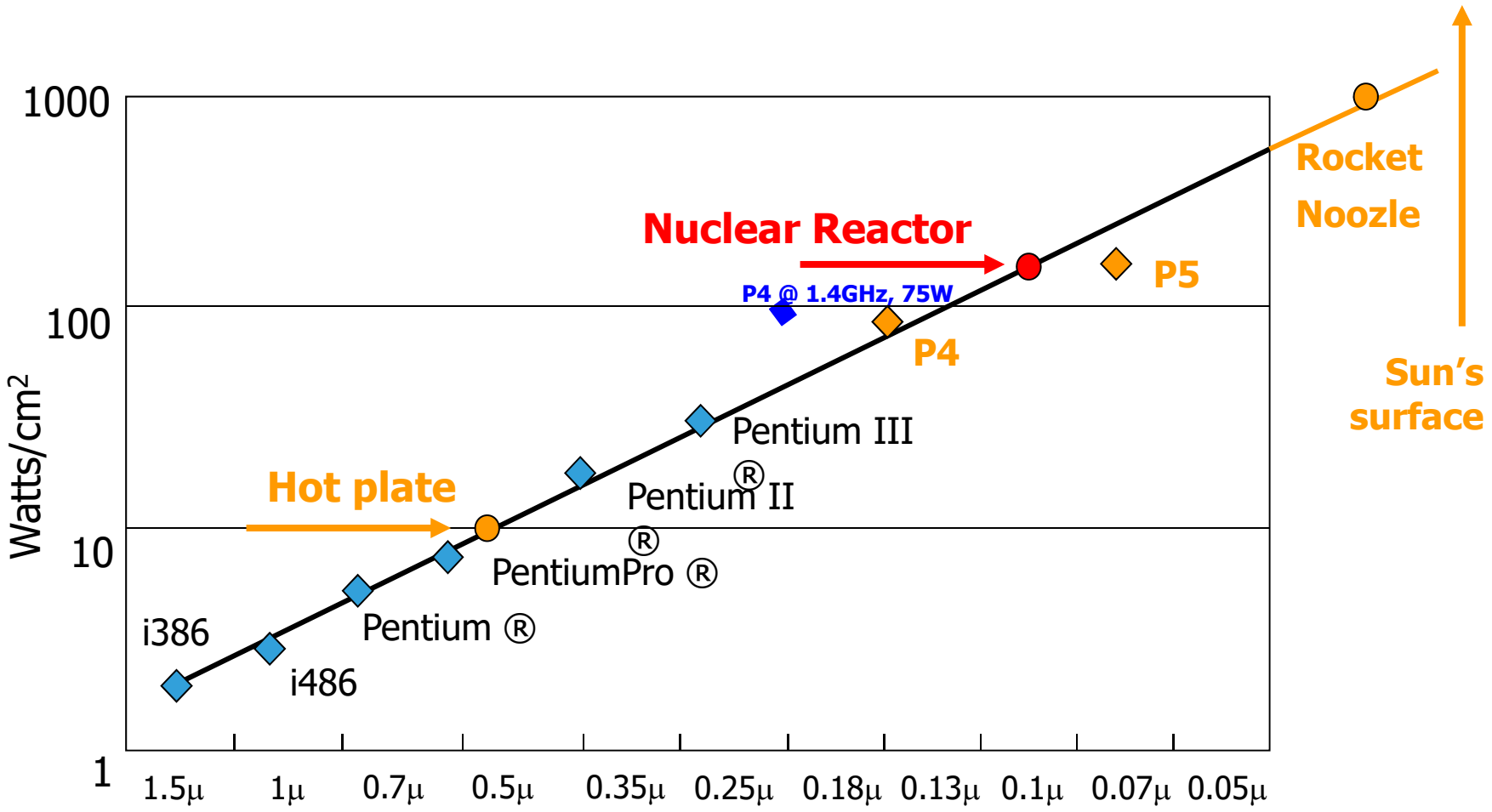
Applications & Requirements

Market application rush

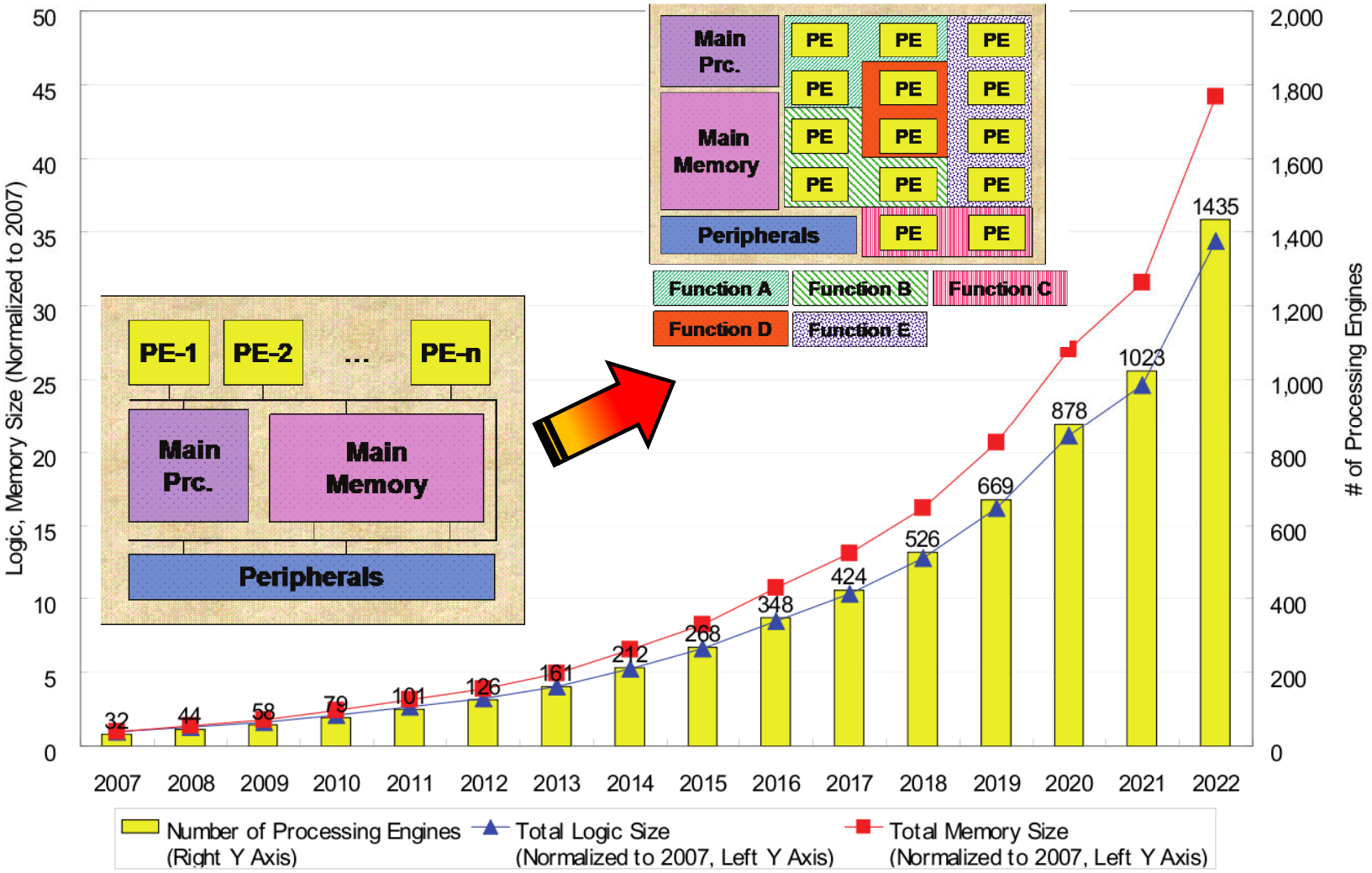


Applications & Requirements

Power density trend in Intel's microprocessors – Single GPP trend is stopped

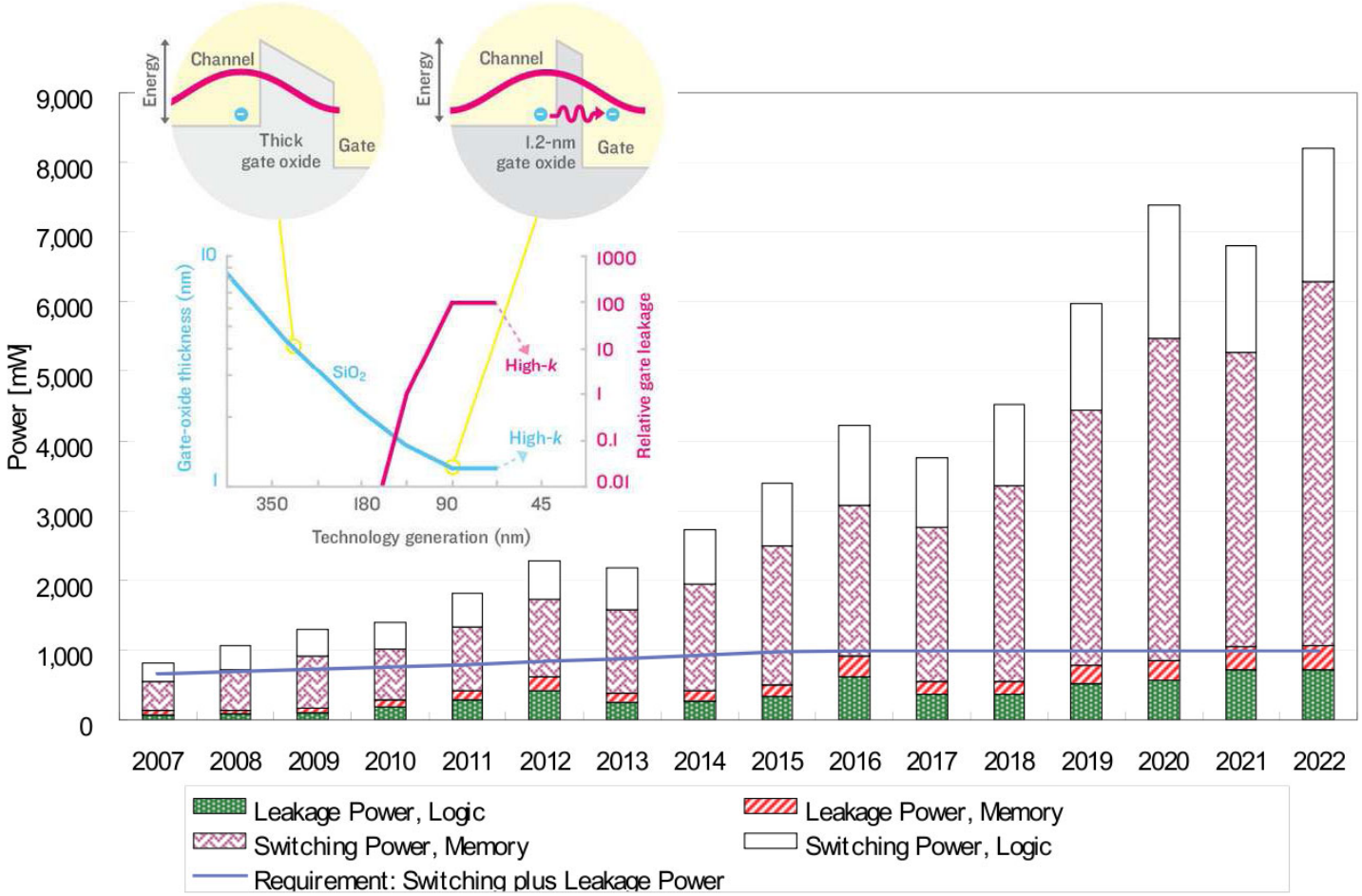


MPSoC architectures evolution



Leakage is becoming the major contribution

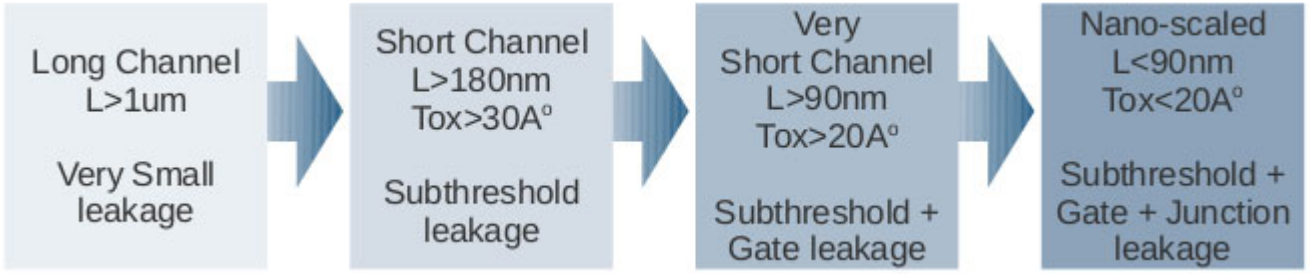
- High-k dielectrics still reduce leakage impacts



Rationale – Power Trends 2

These power issues are expected to get worse as we move to the next technology nodes, as predicted by the ITRS

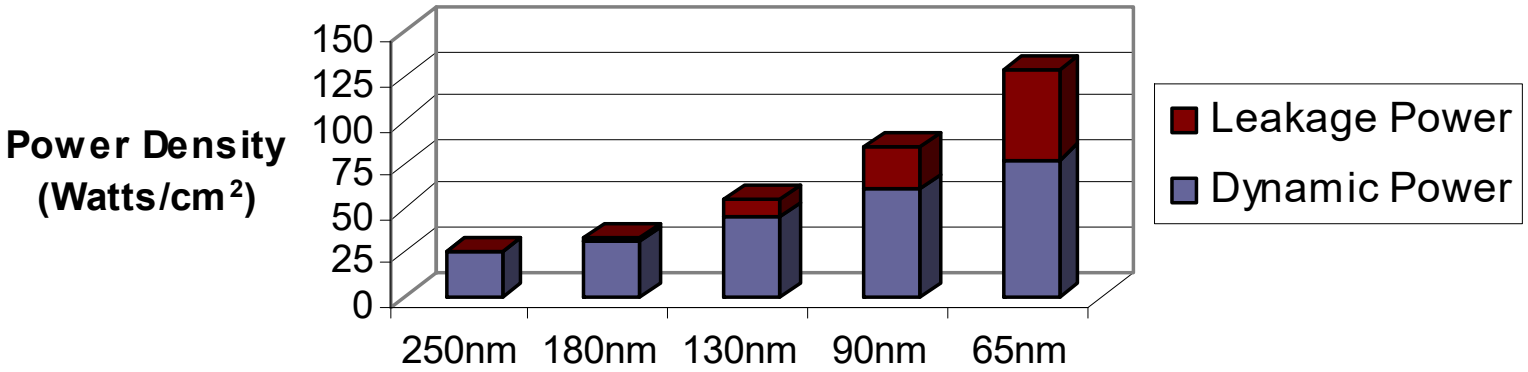
Node	90nm	65nm	45nm
Dynamic Power per cm2	1X	1.4X	2X
Static Power per cm2	1X	2.5X	6.5X
Total Power per cm2	1X	2X	4X



Michael Keating Davidd Flynn et. al.,"**Low Power Methodology Manual"**, ARM&Synopsis, 2007

Leakage power is increasing its contribution

- ASICs



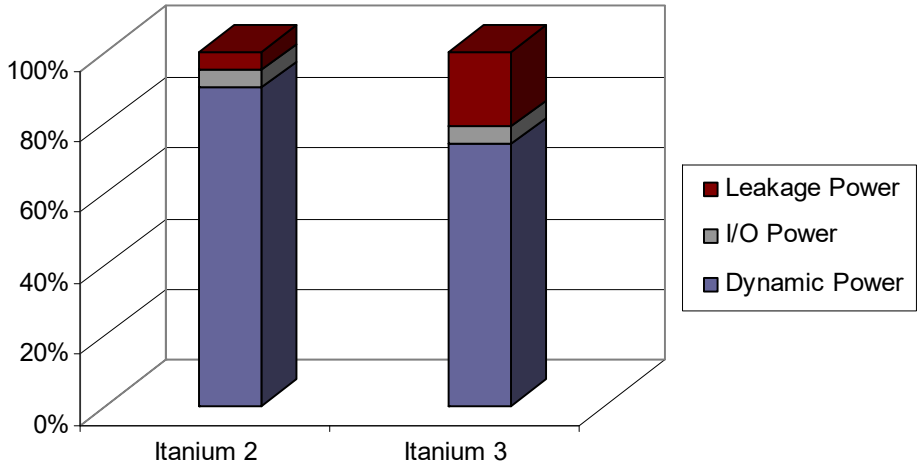
- Microprocessors

Itanium 2:

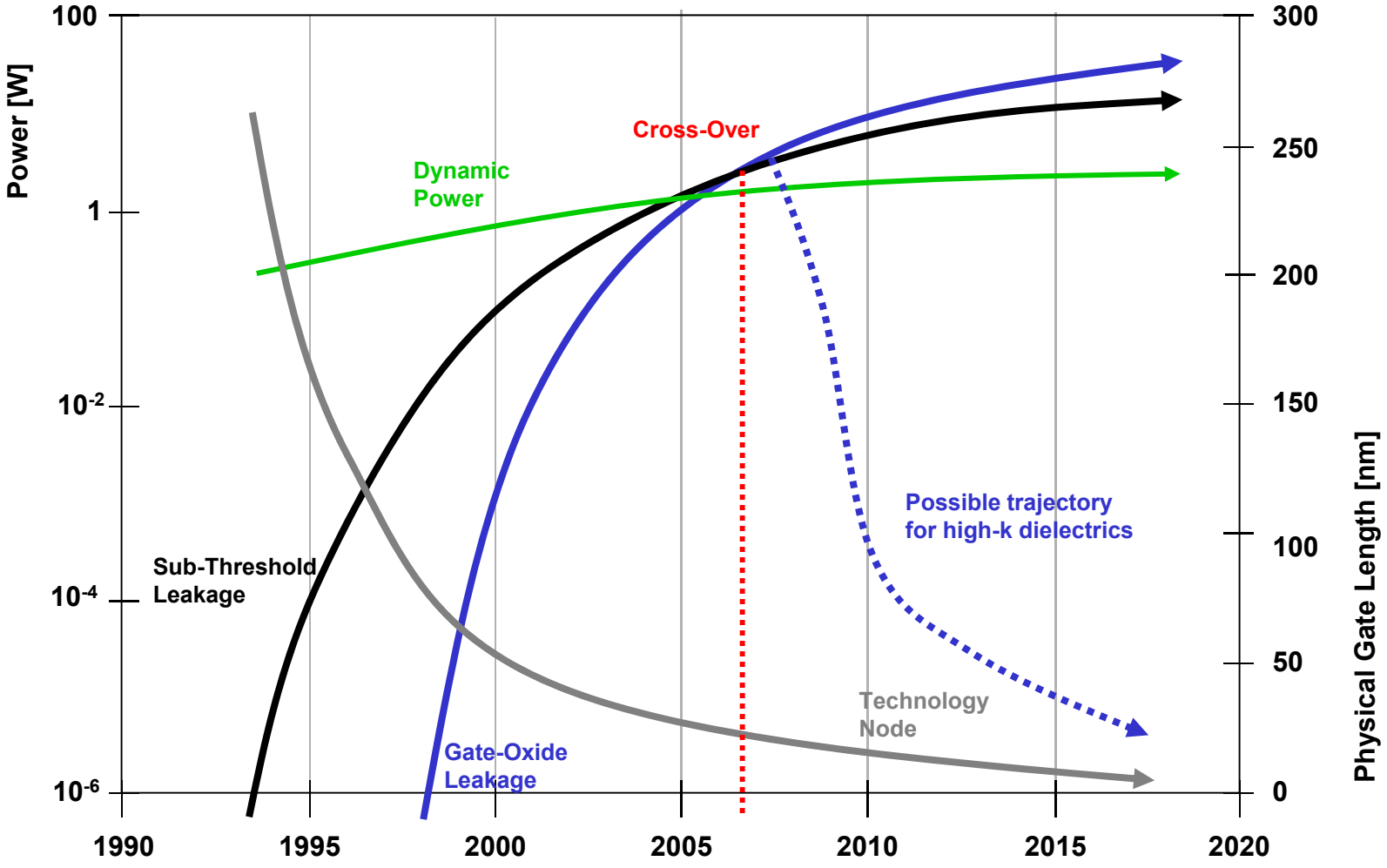
180nm, 1.5V, 1.0GHz, 221MTx (core+cache)

Itanium 3:

130nm, 1.3V, 1.5GHz, 410MTx (core+cache)



Critical regions vs technology scaling



Practical market issue

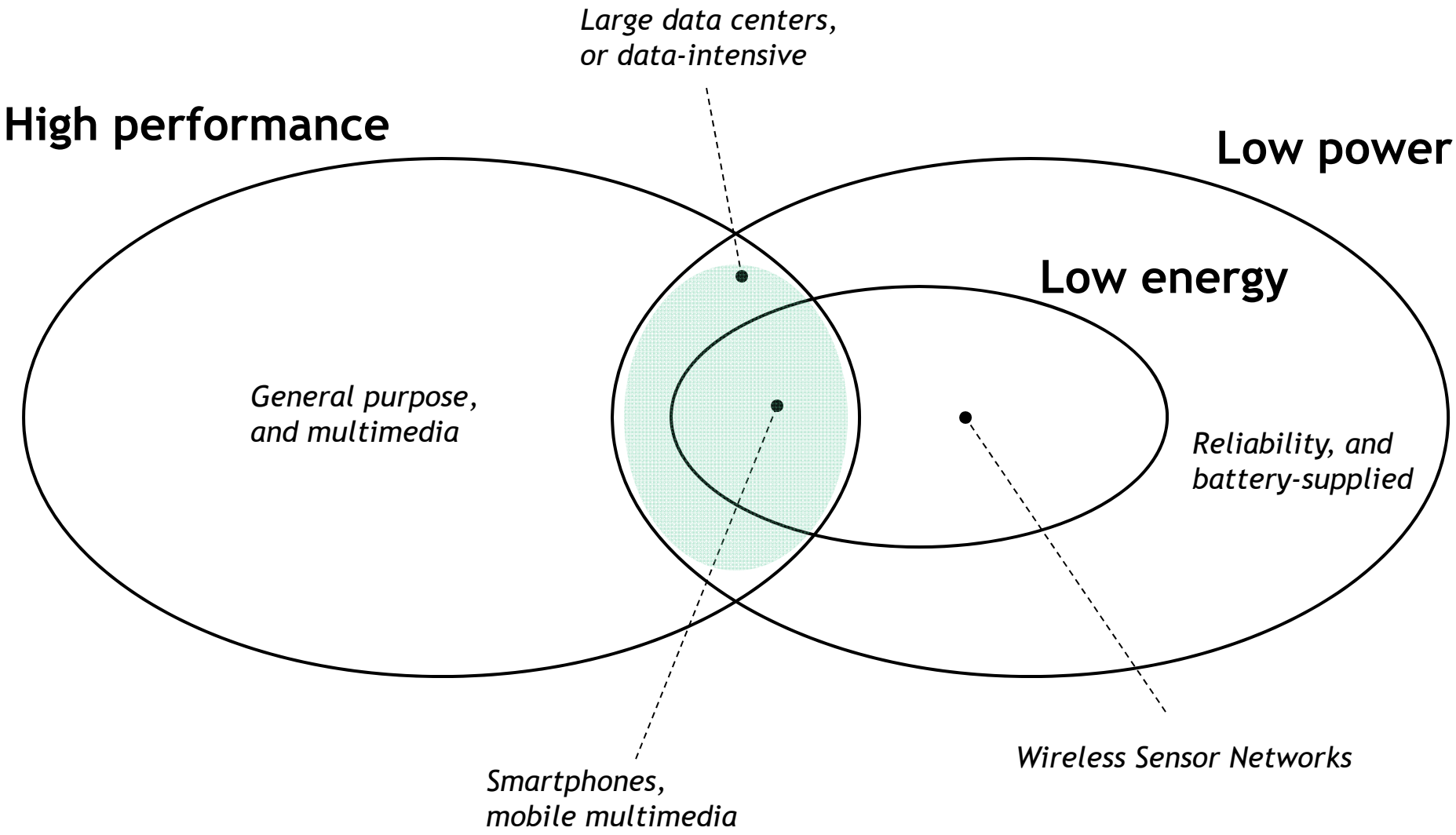
- Increasing market share of mobile, asking for longer cruising life
- Limitations of battery technology

Economic issue

- Reducing packaging costs and achieving energy savings

Technology issue

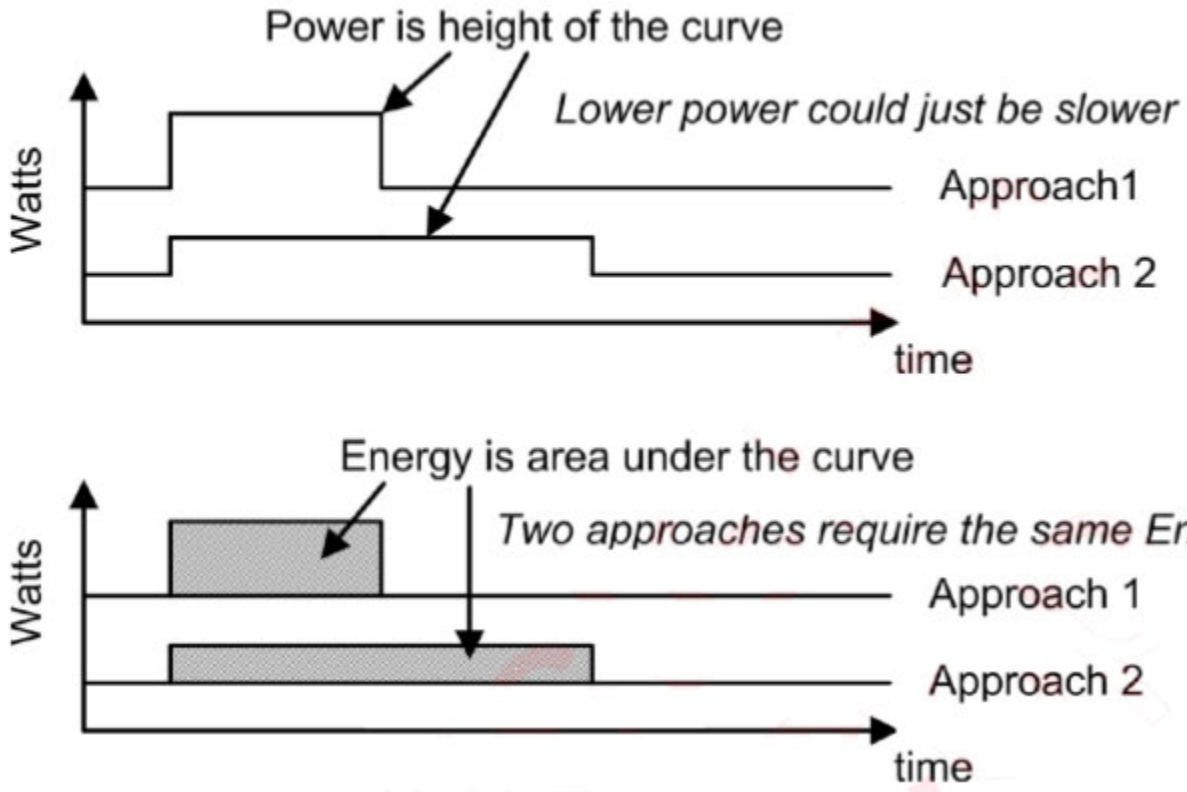
- Enabling the realization of high-density chips (heat poses severe constraints to reliability)


Large data centers,
or data-intensive
High performance
Low power
Low energy
General purpose,
and multimedia
Reliability, and
battery-supplied
Smartphones,
mobile multimedia
Wireless Sensor Networks

Basics: Difference between Power and Energy!

For battery operated devices, the distinction between power and energy is critical:

- Power is the instantaneous power in the device
- Energy is the integral of the power over time



Rationale

- The driving trade-off in future MPSoCs
- Highly application-dependent

Power optimization goals

- Peak-power

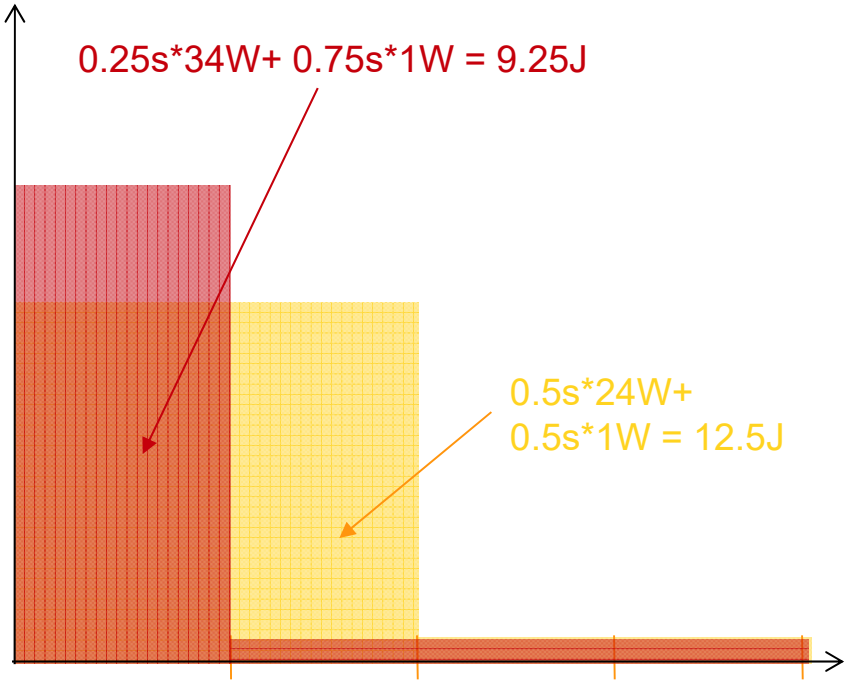
  Minimize for device and system reliability, life-time

- Energy

  Minimize for availability

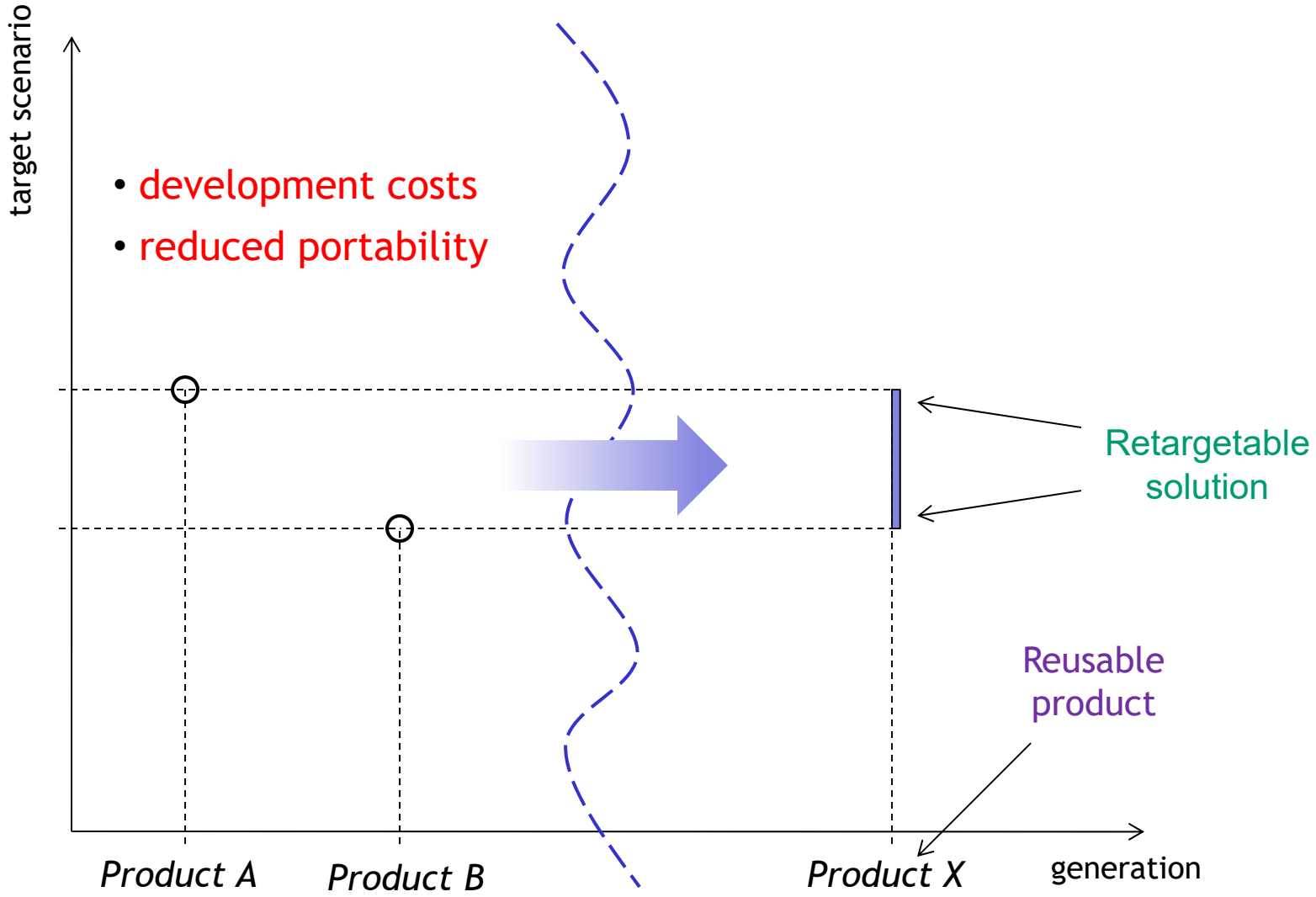
User perceived performance

- Quality-of-Service, e.g. throughput, responsiveness



Applications & Requirements

Retargetable development – ideally orthogonal to power issues



Benefits

- Reduced TTM
- Reduced development costs
- Reusability and flexibility
- Broad range of applications

Enabling aspects:

- **TECHNOLOGY**: integration capabilities of MPSoCs and many-core chips

- **METHODOLOGY**: *platform-based design* vs *ad-hoc design*

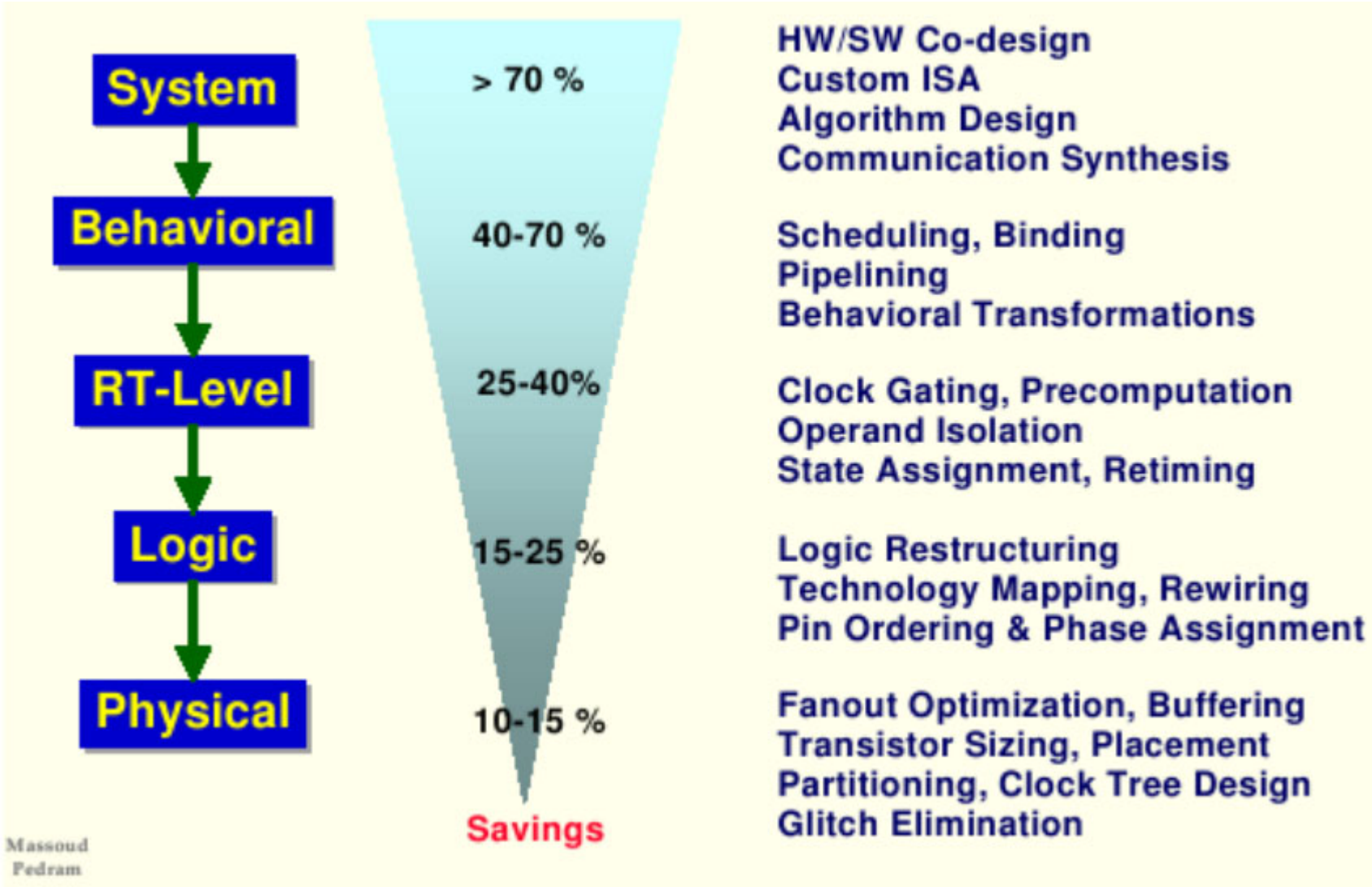
  Optimal solution vs development costs & TTM

Application requirements are very different each other

- How to meet application requirements using “generalized” development?
- How to solve power/performance trade-off?
- How to efficientely tune on-chip resources to applications?

What about current Power Management support?

Power saving opportunities



Power Consumption in CMOS devices

Power = Dynamic Power + Static power

Dynamic Power: the power consumed when the device is active, that is when signals are changing values:

- **Switching power:** the power required to charge and discharge the output capacitance on a gate
- **Internal Power:** short circuit currents that occur when both NMOS and PMOS transistors are on

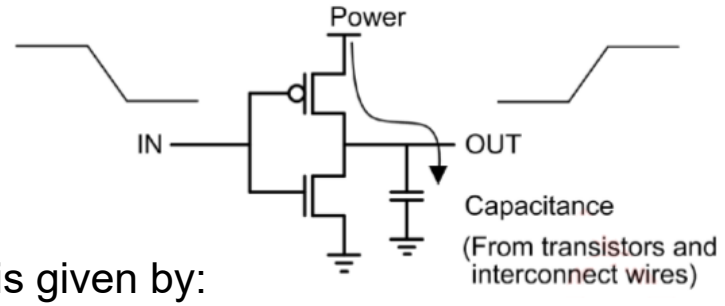
Static Power: the power consumed when the device is not switching values:

- **Sub-threshold Leakage (Isub):** the current which fows from the drain o the source of the transistor operating in the weak inversion region
- **Gate Leakage** (Igate): the current which flows directly from gate through the oxide to the substrate due to gate oxide tunneling and hot carrier injection
- **Other leakage sources: Gate Induced Drain, reverse bias Junction Leakage**

Dynamic Power: Switching Power

Switching power is the primary source of dynamic power consumption.

NOTE: Switching power is data dependent, since it is not a function of transistor size but rather a function of the switching activity and load capacitance



- The energy per transition is given by:

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

- Where C_L is the load capacitance and Vdd is the power supply. We can desribe the dynamic power as:

$$P_{dyn} = Energy/transition \bullet f = C_L \bullet V_{dd}^2 \bullet P_{trans} \bullet f_{clock}$$

- Where f is the frequency of transitions, P_{trans} is the probability of output transition, and fclock is the frequency of the system clock. If we define:

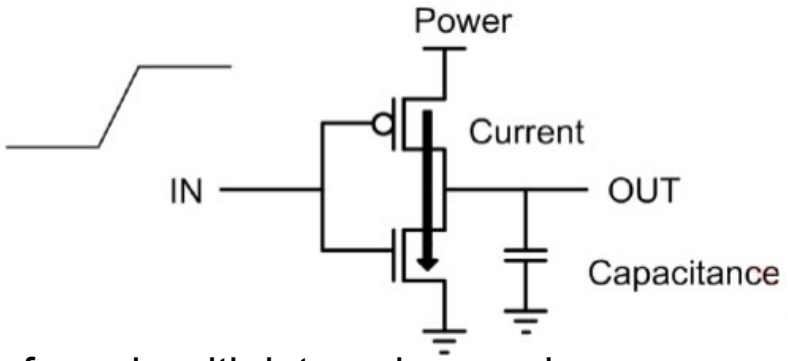
$$C_{eff} = P_{trans} \bullet C_L$$

- Now, we can define dynamic power with the following formulation

$$P_{dyn} = C_{eff} \bullet V_{dd}^2 \bullet f_{clock}$$

Dynamic Power: Internal Power

Internal power accounts for short circuits that happen when both NMOS and PMOS are ON.



- The total dynamic power formula with internal power is:

$$P_{dyn} = \left(C_{eff} \bullet V_{dd}^2 \bullet f_{clock}\right) + \left(t_{sc} \bullet V_{dd} \bullet I_{peak} \bullet f_{clock}\right)$$

Where t_{sc} is the time duration of the short circuit current, and Ipeak is the total internal switching current.

- As long as the ramp time of the input signal is kept short, the overall dynamic power is dominated by the switching power:

$$P_{dyn} = C_{eff} \bullet V_{dd}^2 \bullet f_{clock}$$

Dynamic Power Reduction Techniques

There are a lot of techniques at architectural, logic design and circuit design than can reduce power, while all of them are focused on **voltage, frequency and switching activity** components of the dynamic power equation

- Switching activity:
  - Data dependent
- Frequency techniques (linear dependency of power on frequency)
  - **Clock gating**: driving the frequency to zero drives the power to zero also
- Voltage techniques (quadratic dependency of power on voltage)
  - **Multi-voltage:** assign different voltages to different blocks
  - **Variable supply voltage:** different voltage levels for the same block either dynamically or statically assigned
- Combined techniques, i.e. DVFS to mix advantages of different techniques

Static Power: Sub-threshold Leakage

Sub-threshold leakage occurs when the CMOS gate is not completely turned off

$$I_{SUB} = \mu C_{ox} V_{th}^2 \frac{W}{L} \cdot e^{\frac{V_{GS}-V_T}{nV_{th}}}$$

where **W** and **L** are the dimensions of the transistor, and $\mathbf{V_{th}}$ is the thermal voltage **kT/q** (25.9mV at room temperature). The parameter n is a function of the device fabrication process and ranges from 1.0 to 2.5.

- Sub-threshold leakage current increase exponentially with temperature allowing for great issues in low power system design. Moreover, the temperature increase with the power density, then it is possible to have a positive feedback between temperature and leakage current

- Sub-threshold leakage depends exponentially on the difference between $\mathbf{V_{GS}}$ and $\mathbf{V_T}$, thus **Vdd** scaling must be carefully considered

NOTE: we do not consider gate leakage here, while it can be nearly 1/3 of the sub-threshold leakage power @90nm, roughly equal to sub-threshold leakage @ 65nm. The common technological solution to face gate leakage is the introduction of the high-k dielectric material under 65nm.

Leakage Power Reduction Techniques

- Detailed discussion
  - **Multi-VT**: using high V_T cells wherever performance goals allow and low V_T cells where necessary to meet timing
  - **Power Gating**: shutdown/cut-off power supply to a block of logic when it is not active

- Other approaches
  - **Variable Threshold CMOS (VTCMOS)**: applying a reverse bias voltage to the substrate, it is possible to reduce the value of the term (V_{GS}-V_T), effectively increasing V_T. It adds complexity to the technology library requiring additional power networks to separately control voltage, while the effectiveness of reverse body bias decreases with technology scaling
  - **Long Channel Devices**: using non-minimal length channels will reduce leakage. However performance degrades, since lower dynamic current. Moreover, the greater gate capacitance increases the dynamic power consumption.

The Conflict between Dynamic and Static Power 1

The most effective way to reduce DYNAMIC POWER is to reduce the supply voltage, due to the quadratic relation between Vdd and dynamic power

(Vdd has lowered from 5V to 3.3V to 2.5V to 1.2, while ITRS road map predicted 1.0V @ 2009)

Lower Vdd means lower IDS (on or drive current in MOS), then slower speed, according to the following formula:

$$I_{DS} = \mu C_{ox} \frac{W}{L} \cdot \frac{(V_{GS} - V_T)^2}{2}$$

where \mu is the carrier mobility,C_{OX} is the gate capacitance, V_T is the threshold voltage and V_{GS} is the gate-source voltage.

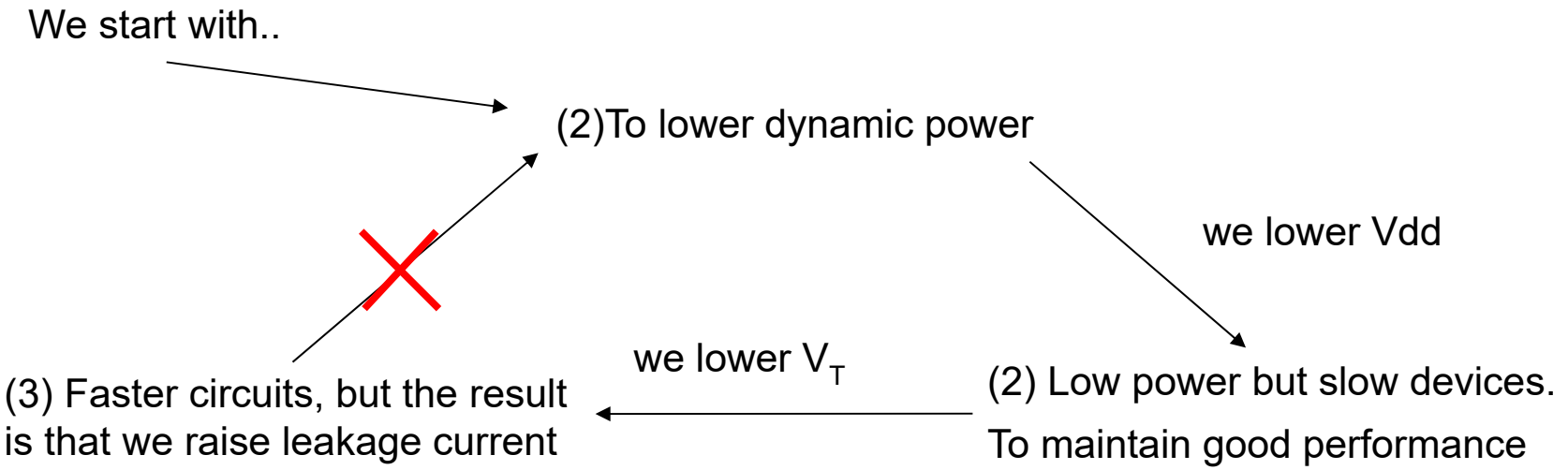
One viable solution to maintain good performance seems to be “setting a lower V_T value”,

... but there is a problem...

The Conflict between Dynamic and Static Power 2

Lowering VT results in an exponential increase in the sub-threshold leakage current, thus leakage power, according to the following formulation:

Thus there is a conflict. To lower dynamic power we lower Vdd; to maintain good performance we lower VT; but the result is that we raise leakage current. While this process was reasonable until 90nm technology node, we are now experiencing the point where static power and dynamic one are comparable.



NOTE: Vdd reduction is no more a practical solution under 32nm

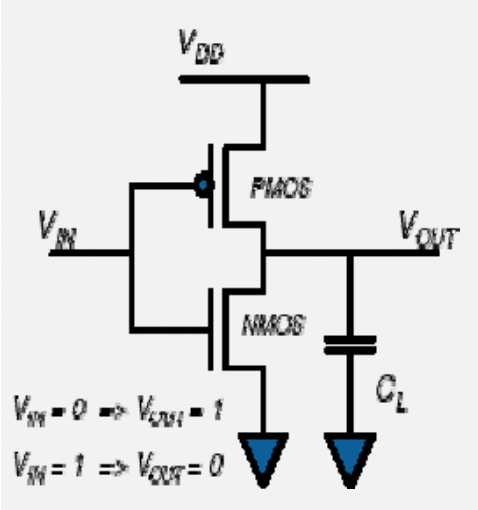
Basics of Power Management- SUMMARY

Power consumption components

$$P = 0.5 V_{DD}^2 f_{clock} C_L E_{sw} + t_{sc} V_{DD} I_{peak} f_{0\rightarrow 1} + V_{DD} I_l$$

Dynamic power

- E_{sw} is the switching activity
- Data-dependent



Short-circuit power (Internal, dynamic)

Leakage power (static)

- 250nm technology scenario: marginal w.r.t. switching power
- Beyond 90nm: 35-50% of (logic) power budget

Increasing device density

- Smaller capacitance per gate (C_L)
- … but more gates per chip

➔Increasing switched capacitance (C_{eff})

Increased dynamic power

Higher clock frequency

Lower supply voltage

- Lower switching power and speed
- … but lower threshold voltage

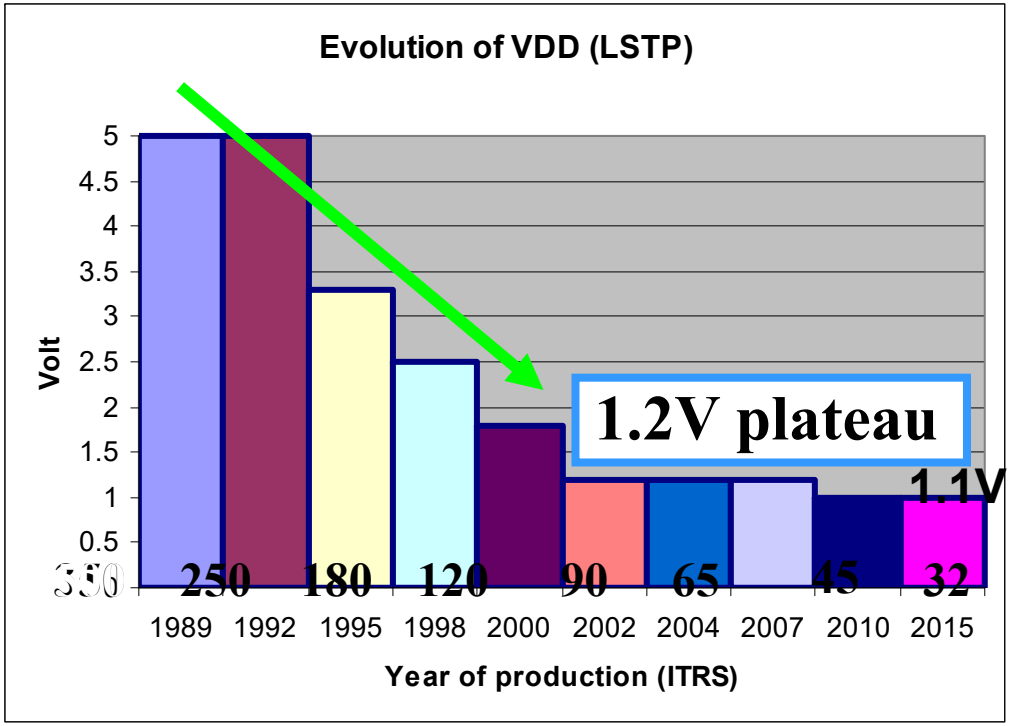
Increased leakage power

Higher on-die temperature

VDD is no more scaling down

Regular decrease
5V to 1.2V (0.7x per node)

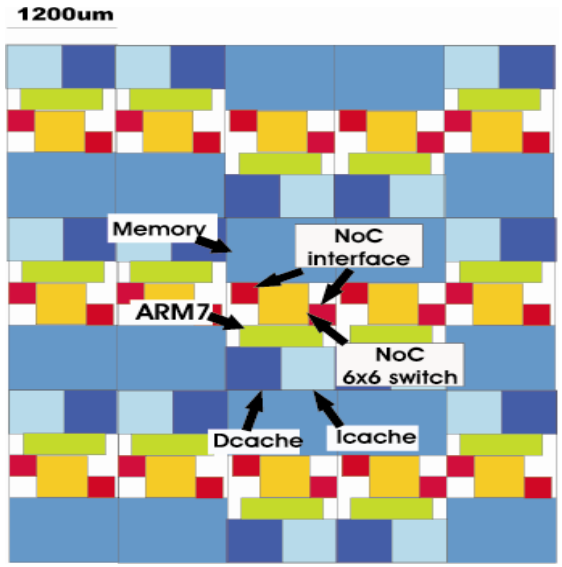
5V plateau



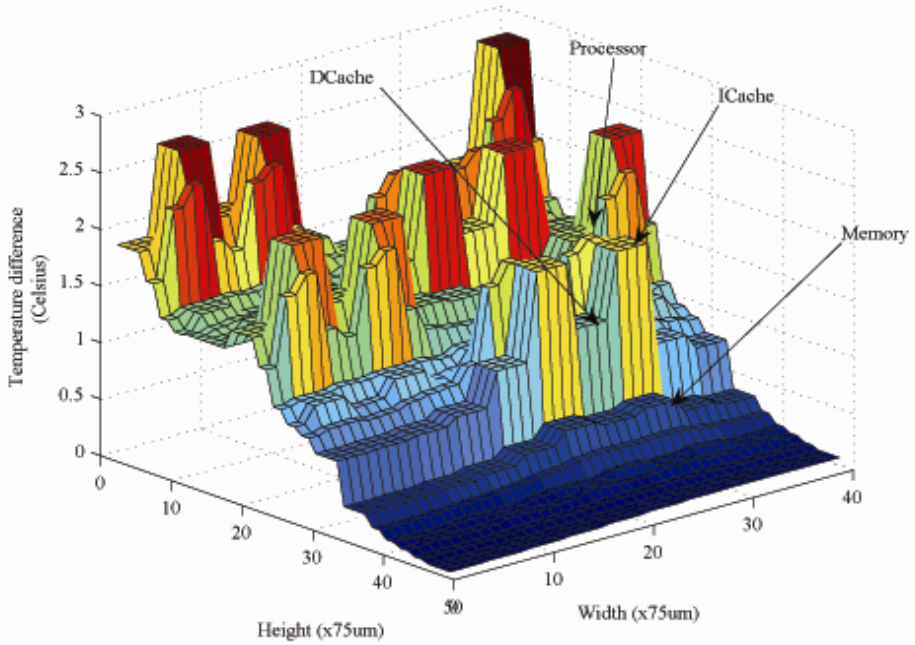
1V plateau?

Basics of Power Management

Hot spots and Thermal problems



Chip floorplan



Steady state temperature

Some hot spots in steady state:

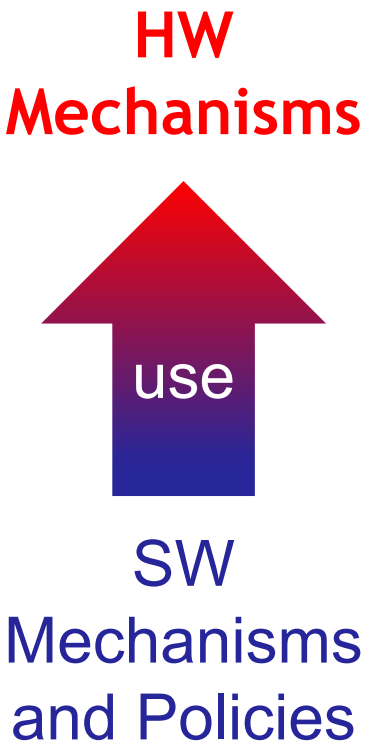
- Silicon is a good thermal conductor (only 4x worse than Cu) and temperature gradients are likely to occur on large dies
- Lower power density than on a high performance CPU (lower frequency and less complex HW)

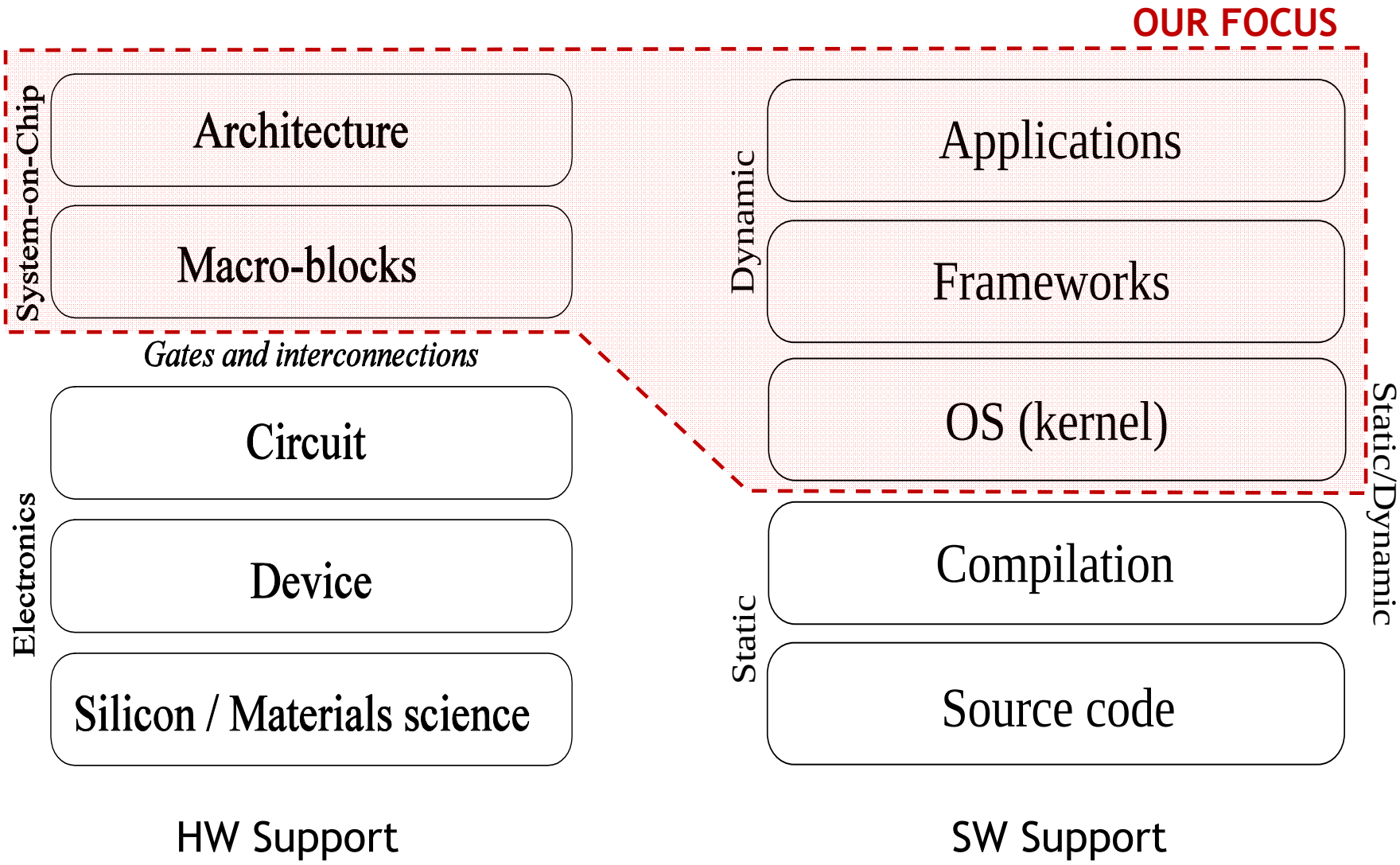
Methodologies

- The *static* view of the problem
- Define building blocks for Power Management
  e.g., Silicon technology, design guidelines…
- Should properly support run-time management

Management

- The *dynamic* view of the problem
- Exploit building blocks to support system-wide optimization
  Monitoring current application requirements
  Accounting for resources availability
- Should be flexible and adaptable




OUR FOCUS
System-on-Chip
Architecture
Macro-blocks
Gates and interconnections
Electronics
Circuit
Device
Silicon / Materials science
HW Support
Dynamic
Applications
Frameworks
OS (kernel)
Static
Compilation
Source code
Static/Dynamic
SW Support

Architectural Power Reduction Approaches

$$P = 0.5 V_{DD}^2 f_{clock} C_L E_{sw} + t_{sc} V_{DD} I_{peak} f_{0 \rightarrow 1} + V_{DD} I_l$$

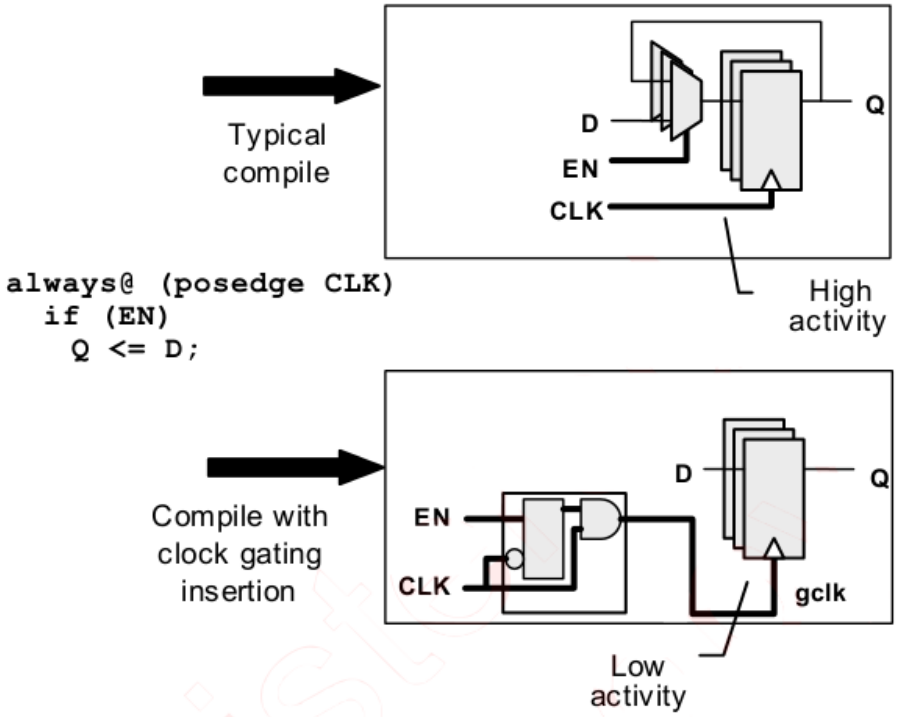
Architectural Power Reduction Approaches

- Switching power
  - Reduce supply voltage
  - Reduce clock frequency
  - Reduce wasteful switching (operate on data)
- Short-circuit Power
  - Reduce V_{DD}
- Leakage Power
  - Reduce V_{DD}

Clock Gating

Up to 50% of the dynamic power is spent for clock distribution. Turn the clock off, when the logic block is idle, can reduce significantly dynamic power.

- **Automatic stuff**: modern synthesis tools support automatic clock gating identifying the logic block where the clock gating can be applied without changing the logical function.
  - Most libraries include specific clock gating cells that are recognized the synthesis tool
  - Clock gating explicitly coding is too error prone, it may produce glitches, functional errors ...



Hardware blocks fed with the same gated clock supporting clock gating

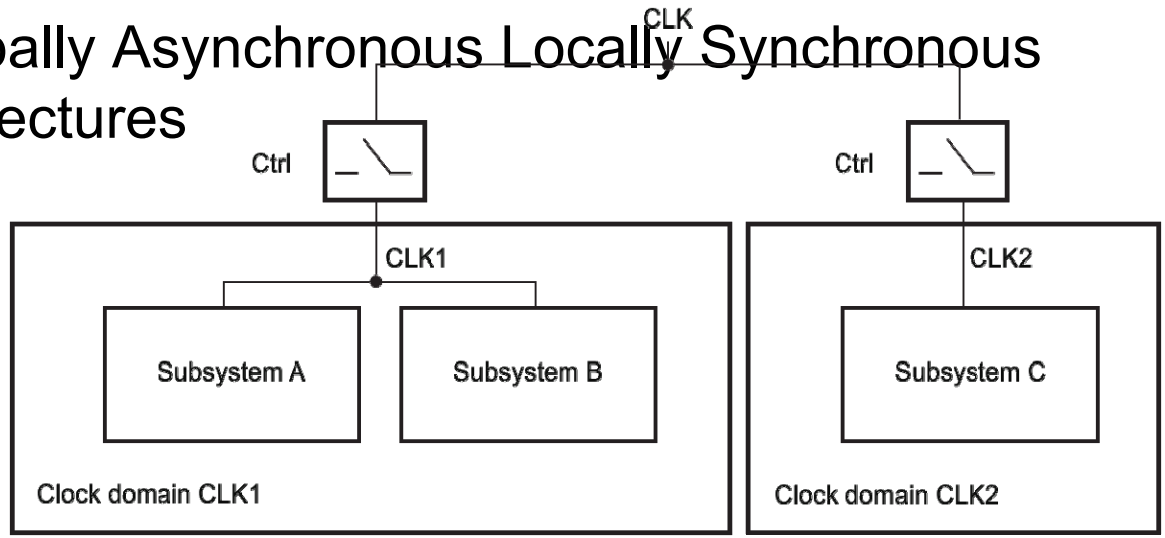
Support clock gating to control dynamic power

- Cut a clock to a group of inactive blocks
- Lower active-power consumption

  Two possible states: active or inactive

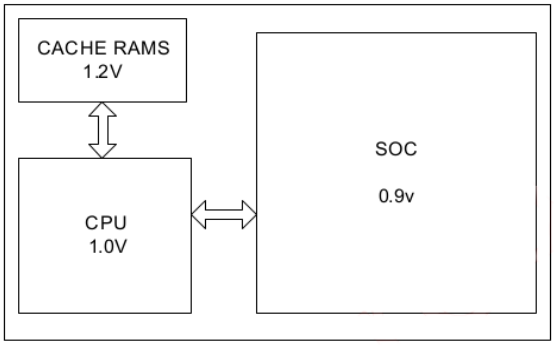
Each domain can have its own frequency

- Enabling Globally Asynchronous Locally Synchronous (GALS) architectures

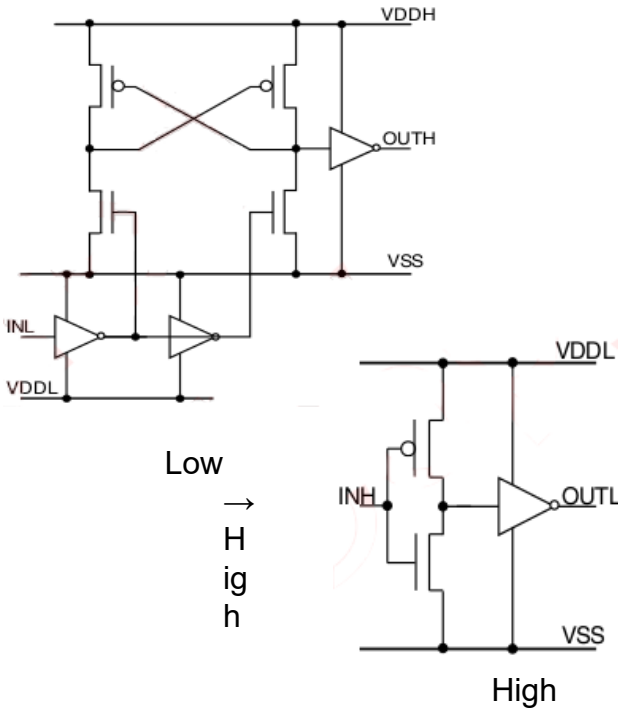


Multi Vdd

Since dynamic power is proportional to Vdd^2, lowering Vdd on selected blocks helps reduce power significantly.



- **Apply different voltage to different logic blocks according to performance requirement**
  - Account for lower performance
  - Additional complexity to the design
  - More complex power grid
- **What about design issues?**
  - Unidirectional level shifter design
    - **HighV → LowV** are easier to design as 2 not in series, a buffers
    - **LowV → HighV** are much complicated to design
  - Level shifter placement
    - **HighV → LowV** are placed in the lower domain, destination
    - **LowV → HighV** placed in the higher domain with a VddL line from the lower domain
  - Timing analysis done in parallel with Muti-VDD generation



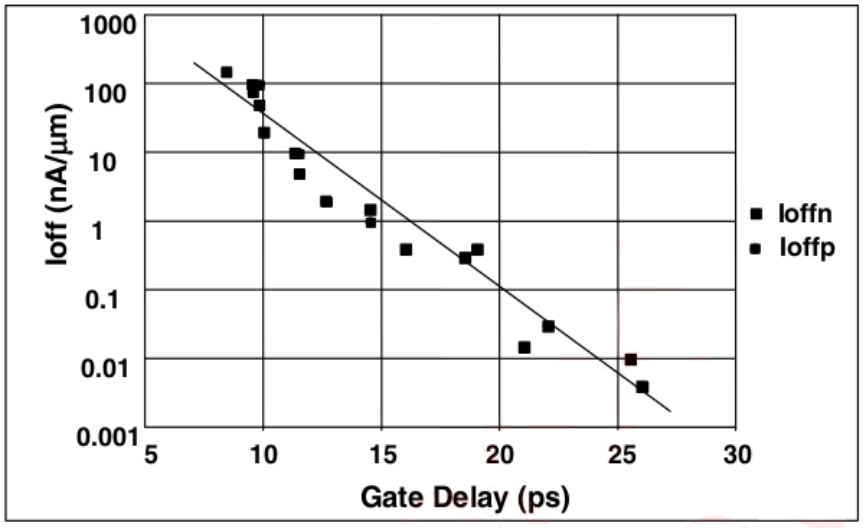
Multi-Threshold Logic

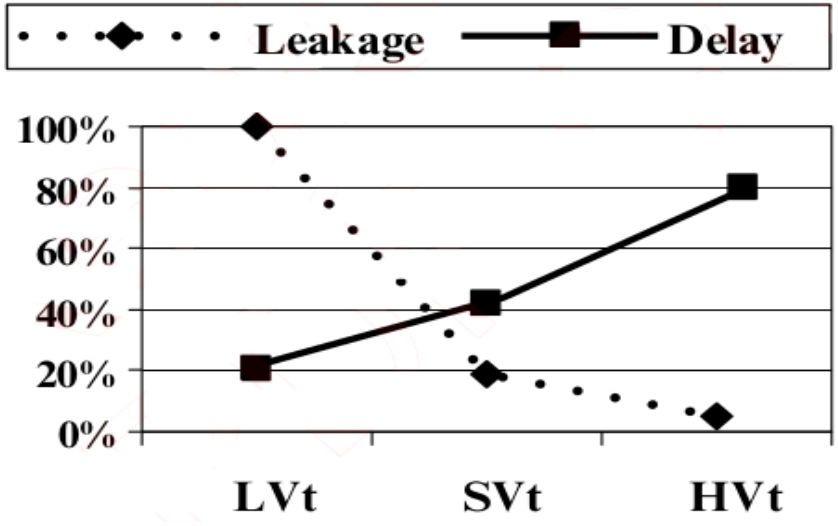
As geometries have shrunk to 90nm and below, using libraries with multiple VT has become a common way of reducing leakage current

- **Many libraries today offer two or three versions of their cells:**
  - Low VT, Standard VT, High VT

  The synthesis tools can take advantage of these libraries to optimize timing and power simultaneously

- **Dual VT flows are quite common**. The goal of this approach is to minimize the total number of fast transistor by deploying them only when required to meet timing.



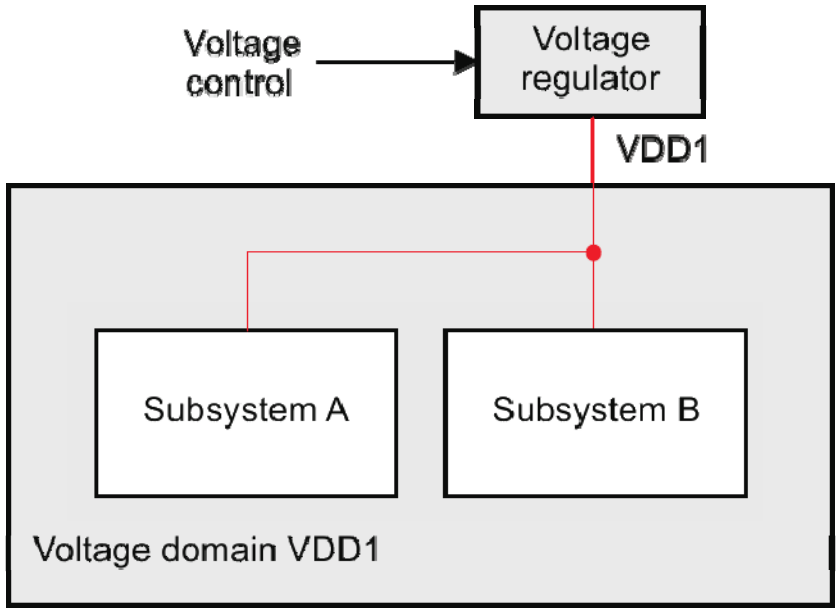


Summary on standard low power techniques

Tech-nique	Power Benefit	Timing Penalty	Area Penalty	Impact: Architec-ture	Impact: Design	Impact: Verifica-tion	Impact: Place & Route
Multi Vt	Medium	Little	Little	Low	Low	None	Low
Clock Gating	Medium	Little	Little	Low	Low	None	Low
Multi Voltage	Large	Some	Little	High	Medium	Low	Medium

Groups of HW blocks supplied by same voltage regulator

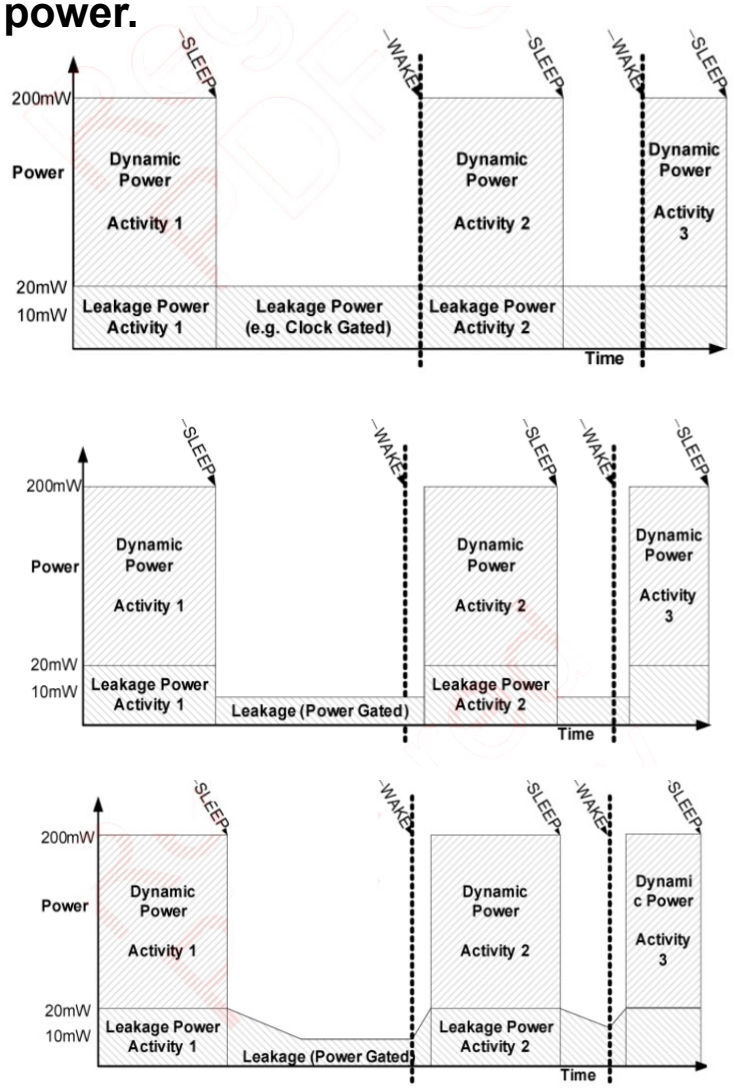
- Power consumption can be controlled by regulating voltages independentely
- Assign different operating V to different HW blocks
  - Voltage scaling of device subsections
- Voltage scaling allows to reduce power consumption



Power Gating

Allows to reduce both leakage and dynamic power at the same time, while it is first designed to face leakage power.

- **More invasive than clock gating:**
  - Affects inter-block communication
  - Adds significant time delay
  - Require retention capabilities to save the state
- **Activated both at software and hardware level**
  - Block shutting down can be scheduled by control software as part of device driver
  - Initiated via hardware by timers or power management controllers
- **Any event, there is an architectural trade-off:**
  - Amount of possible leakage power saving
  - Entry and exit penalties
  - Entry and exit dissipated energy



Power Gating impact on different sub-systems

- **Cached CPU subsystem**
  - Power gating the entire CPU provides very good leakage power reduction
  - Wake-up time response to an interrupt has significant system level design implications
  - Cache content are lost every time CPU is powered down, then I will spend power to restore cache blocks
  - The net energy savings depend on the sleep/wake activity profile

- **Peripheral subsystem**
  - The device driver requires explicitly load/restore key state
  - A better approach for peripheral is the internal state retention, even partial

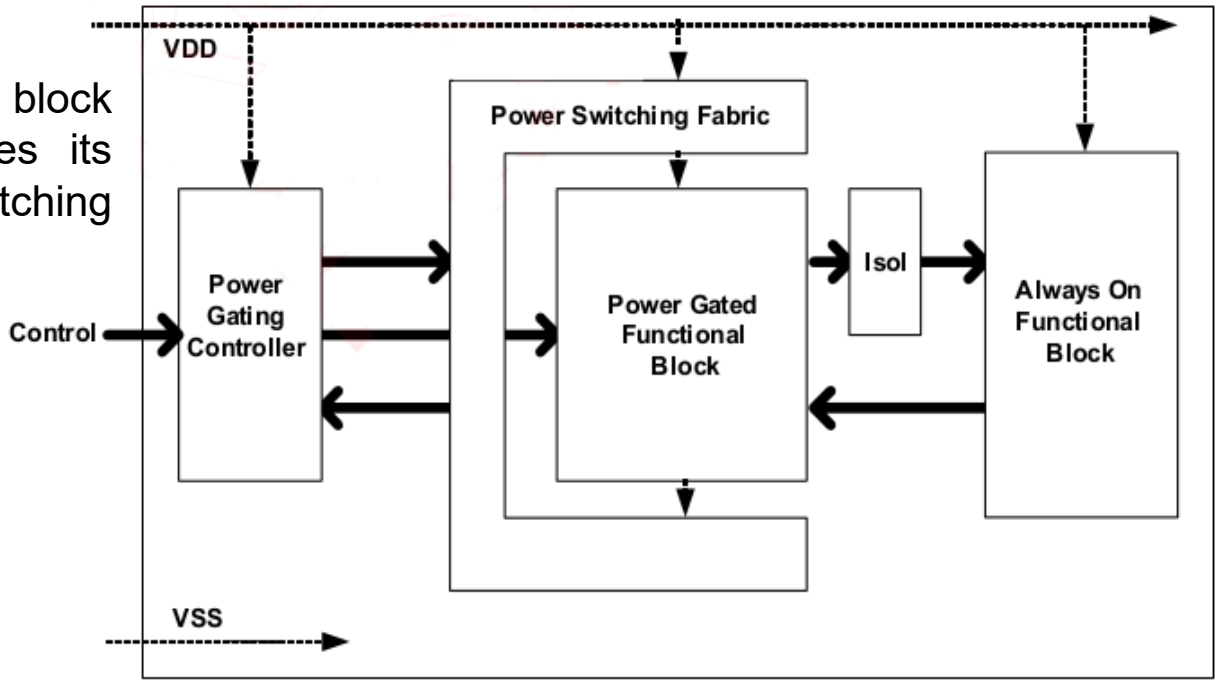
- **Multi-core CPU**
  - Power gating individual CPUs provides very good leakage power reduction
  - Power gated CPUs have completed the task, thus local cache state is useless
  - More flexible and allows for dynamic power saving algorithms

Power Gating Basics

- Selectively powering down certain blocks in the chip while keeping other blocks powered on

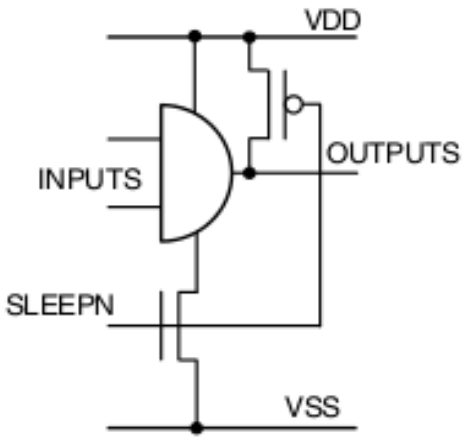
- Unlike an always powered on block the power gated block receives its power through a power-switching network

- We will briefly detail:
  - Power switching fabric
  - Isolation cells
  - Power gating controller
  - State retention
  - Area and timing impact



Power Gating: Coarse VS Fine Grain

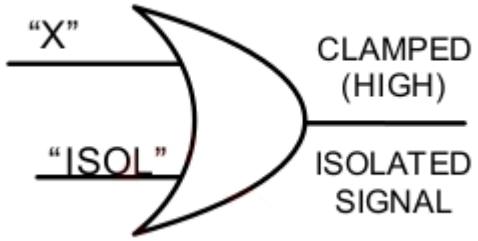
- **Fine grain**
  - The switch is placed locally inside each standard cell in the library.
  - The switch is quite large since it has to provide the maximum current possible required by the cell
  - 2x-4x area overhead
  - The behavior of the entire cell can be easy characterize, allowing to use traditional synthesis flows
- **Coarse grain**
  - A block of gates has its power switched by a collection of switch cells.
  - More difficult to characterize
  - Less area penalty
  - Easier to introduce outside an already implemented cell

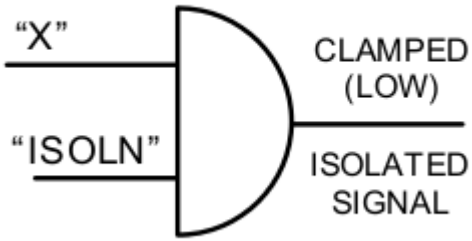


NOTE: strong convergence to coarse grain in last years

Power Gating: Isolation cells

- **Outputs:** We want to be sure than none of the floating outputs of the power down block will result in spurious behavior in the power-up blocks:
  - Logic isolation cells:
    - Clamp output value to '0' or '1'
    - High area penalty (area issues)
    - Full port delay (timing issues)
    - Drive the “ISOLx” line
    - **Recommended**
  - Transistor based isolation:
    - Pull-up or pull-down transistors
    - Low timing and are penalties
    - Electro migration issues
    - Accurate timing to drive the output when block is completely off





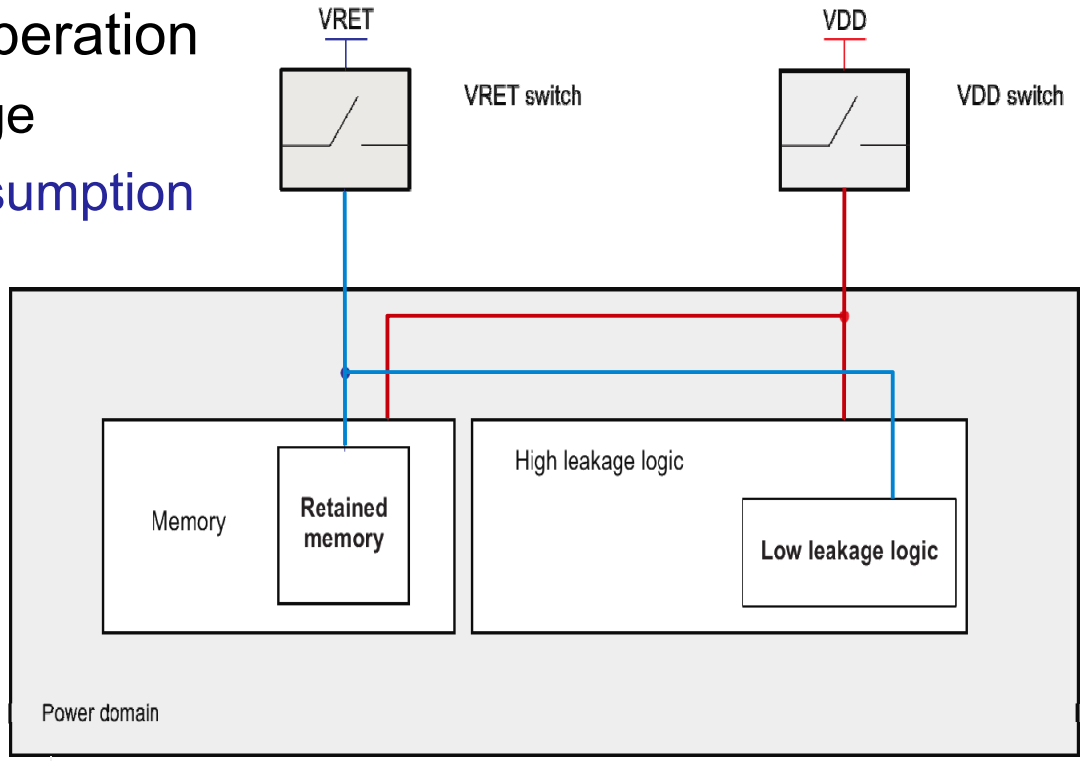
- **Inputs:** inputs of a powered down block are usually not an issue, just drive them to valid logic values by powered up blocks

Power Gating: Isolation Recommendations

- **Isolate the output of power gated blocks:**
  - Otherwise multiple isolation cells are required if signal is split
- **Use isolation cells instead of pull-up pull-down transistors**
  - Otherwise timing complication
  - More complex power gating protocol
- **Ensure stuck-at-0 and stuck-at-1 faults can be detected during test on the isolation logic**
- **Make sure the isolation cells really are always powered on**
  - They must stay outside the clock gated area
- **Avoid clock generated in a power gated block**
  - And used externally to the block
  - It considerably complicates the clock tree synthesis

HW blocks supplied by dedicated power rails

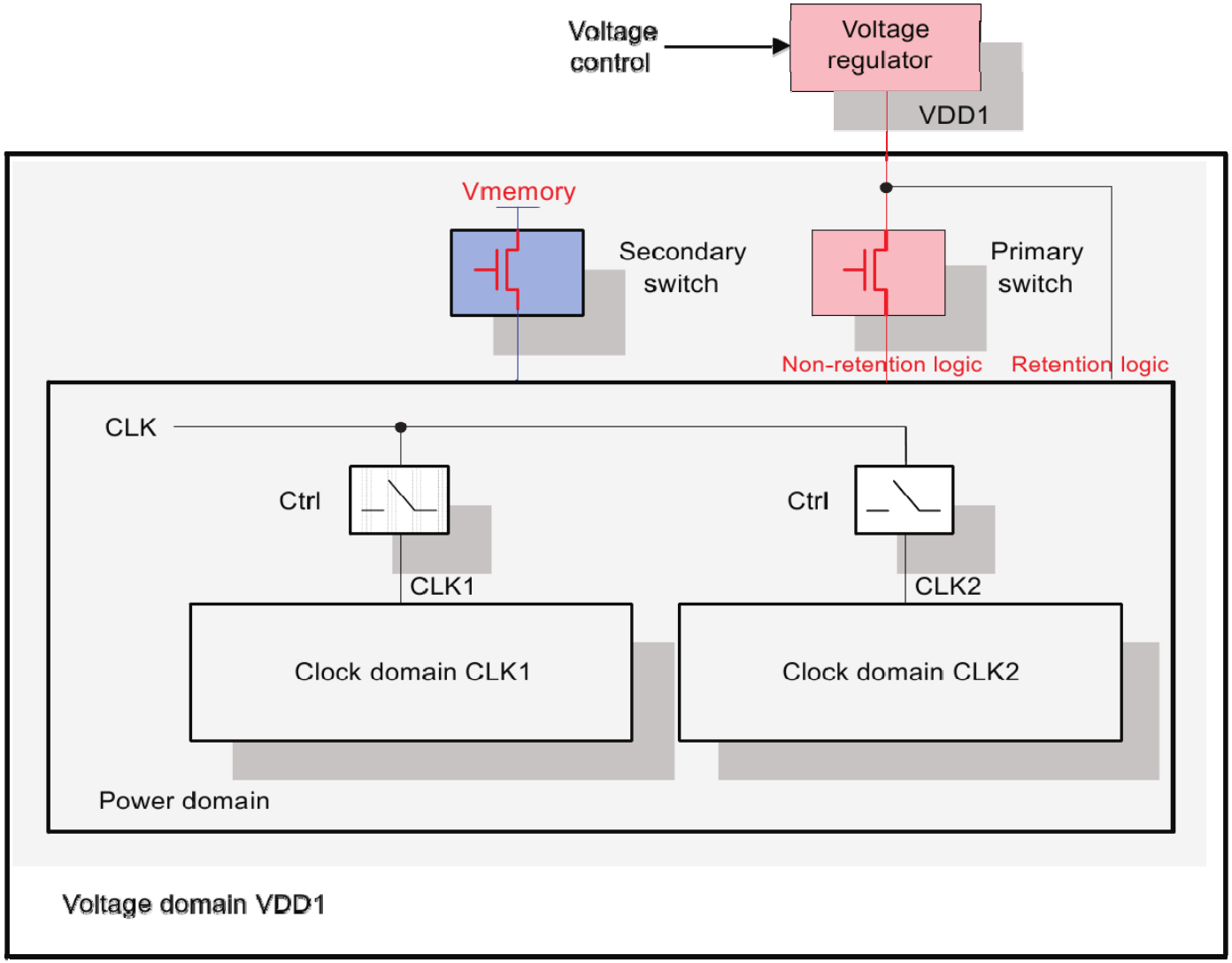
- VDD for normal operation
- VRET for low-power operation
  lower than active voltage
  => reduced power consumption
  logic and memory are
  not operational,
  but in a *retention state*



- Retention state
  In addition to ON/OFF
  Useful to switch to low-power idle mode without losing the context, and quickly switching back to active state when necessary

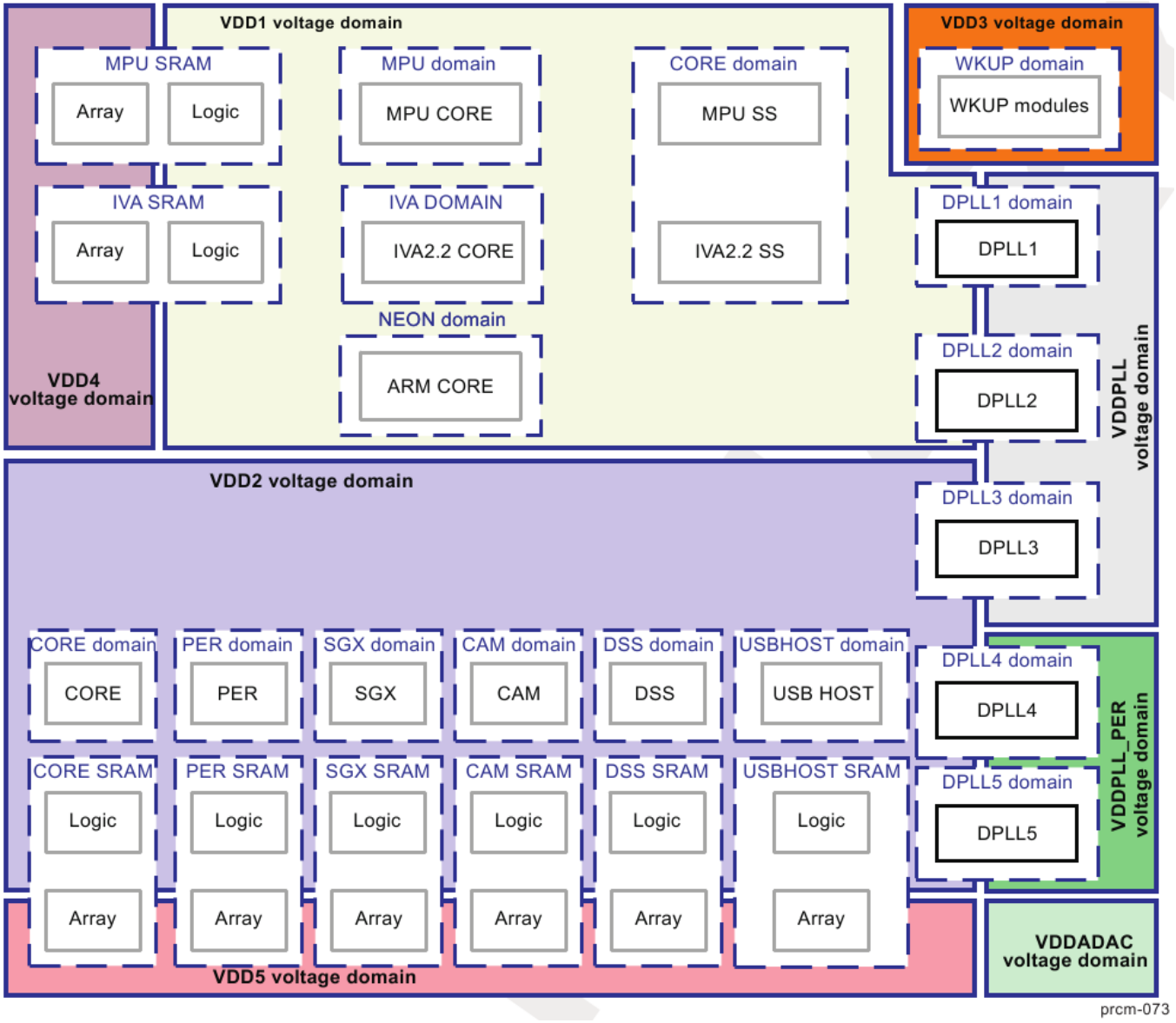
Architectural Blocks for PM

Clock, power and voltage domains hierarchical architecture



Architectural Blocks for PM

OMAP35xx voltage domains



DVFS - Dynamic Voltage and Frequency Scaling

Allocate a variable amount of energy to perform a task power consumption of a digital CMOS circuits

$$P = \alpha \cdot C_{eff} \cdot V^2 \cdot f$$

α *switching factor*
C_{eff} *effectivecapacitance*
V *operating voltage*
f *operating frequency*

$$E = P \cdot T \propto V^2 \qquad (assuming\ f \propto V, T \propto f^{-1})$$

ISSUES

- **V contributes quadratically to P**
- **Frequency only reduction means linear power reduction**

- **There is a linear region between frequency and voltage**
- **Current and future sub-micron technologies cannot exploit V reduction too much**

Yasuko Eckert et.al. ,**"Something Old and Something New: P-states can Borrow Microarchitecture Techniques Too"**, ISLPED 2012

DVFS cond't

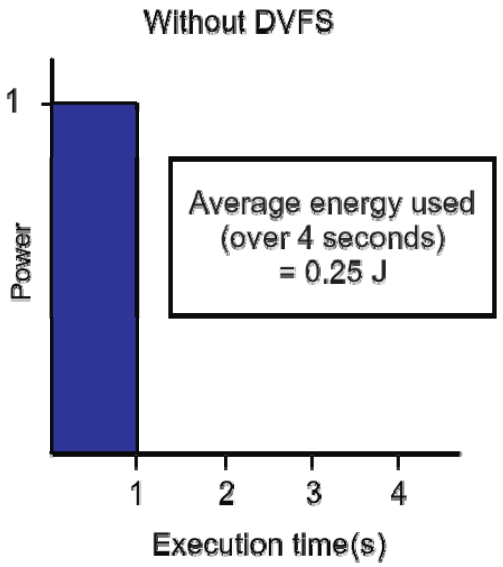
Different approaches with respect to your goals:

- **Maximize system idle time**
  - runs tasks at the highest OPP (operating performance point), complete the task quickly
  - automatic switch to a low-power mode when possible
- **Minimize system idle time**
  - dynamic selection of optimal frequency and voltage
  - allow a task to be performed in the required amount of time
  - Operating Performance Points (OPP) voltage (V) and frequency (F) pair
  - The system always runs at the lowest OPP
- **AVC - Adaptive Voltage Control**
  - Automatic control of the operating voltage
  - Silicon performances/power trade-off (deps on power consumed technology process, operating temperature variations)
  - Power-supply voltage is adapted to silicon performance based on:
    - Performance points (statically)
    - Temp, real-time device performance (dynamically)

Basic principle

- Allocate a variable amount of energy resource to perform a task
- Energy required to run a task

$$E \propto P \cdot T \propto V^2$$



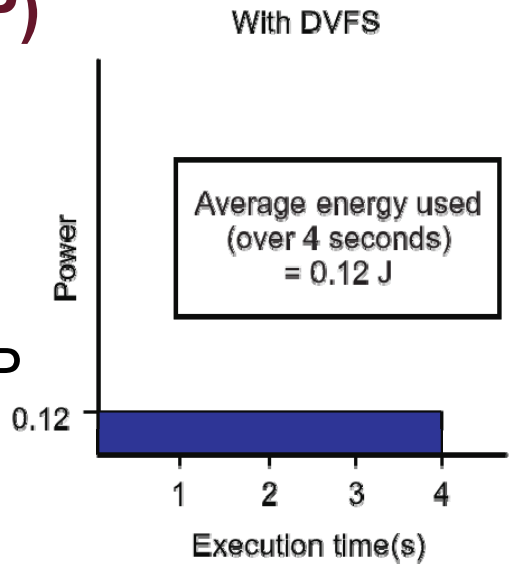
Select Operating Performance Point (OPP)

- A voltage (V) and frequency (f) pair

Minimize system idle time

- The system always runs at the lowest OPP that meets performance requirements

  Reduce dynamic *and* static power

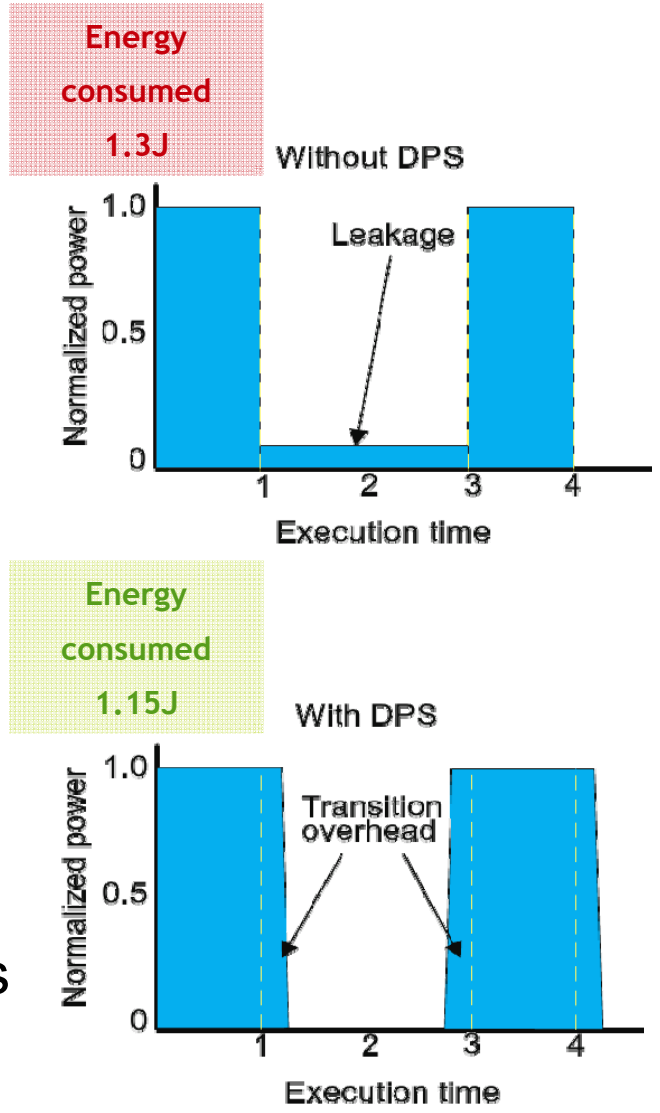


Maximize system idle time

- Automatic switch to low-power mode
- Run tasks at high OPPs to complete tasks quickly
- Reduces leakage power

Main isses

- Transitions overhead
  - Recovery time and power
- Need to dynamically predict applications performance requirements



Trades static power consumption for wake-up latency

- Switching the system between high- and low-power modes

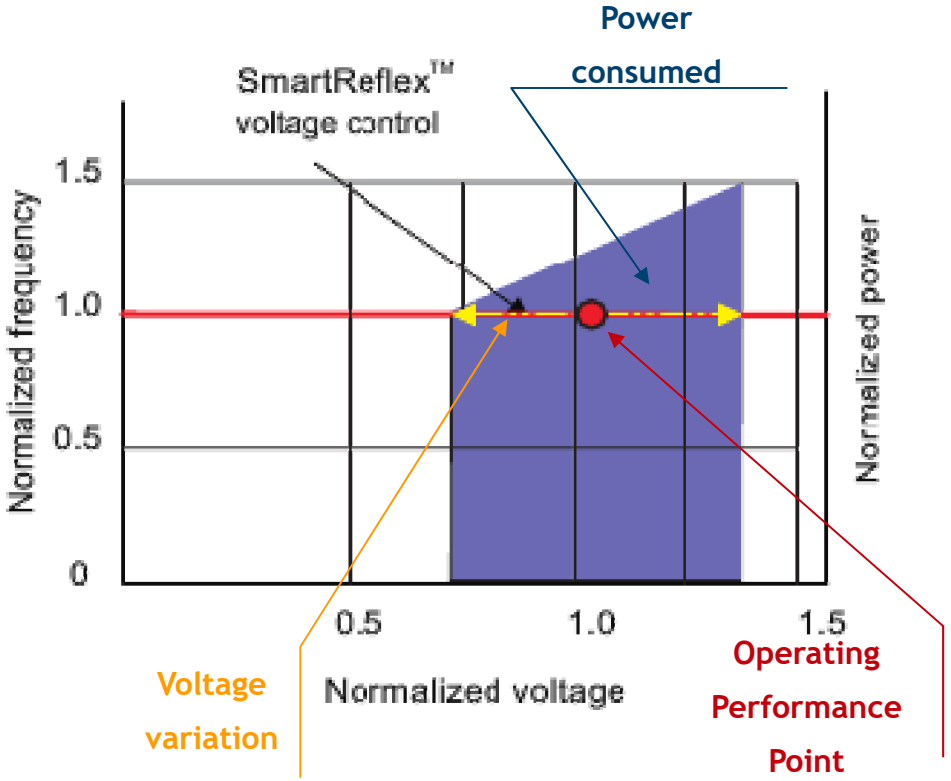
Similar to DPS

- Different operating timescale
  - *Latency allowed for mode transitions*
    - *DPS: compared to time constraints or deadlines of the application*
    - *SLM: compared to user sensitivity so that they do not degrade user experience*
- Different context
  - *Who define the transition constraints*
    - *DPS: tasks are running and we must grant application performances*
    - *SLM: applications not running and must grant system responsiveness*
- Different wake-up events
  - *Events used to exit the low-power mode*
    - *DPS: application-related, e.g. timer, DMA request, peripheral interrupt, ...*
    - *SLM: user-related, e.g. touch screen, key pressed, peripheral connections, ...*

Silicon performance/power trade-off

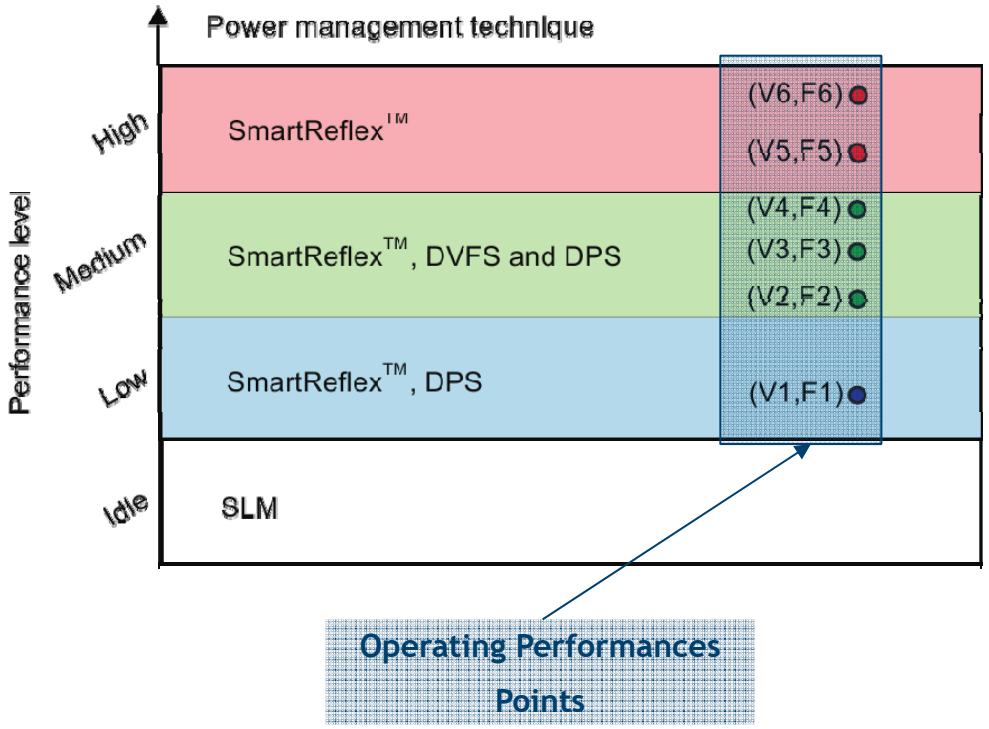
- Power supply adapted to Silicon performance
  - **Statically**, based on performance points
  - **Dynamically**, based on temperature-induced real-time performance

- Optimal power/performance trade-off for all devices, across the technology process spectrum and temperature variations



Best active power savings through combination

- AVS
  - at boot-time to adapt to device process profiles
  - always to compensate temperature variations
- DVFS
  - without DPS to scale f and keep V constant
- DPS
  - with DVFS to set f to max for given V
- SLM
  - no applications running



	Power component optimization		Exploited tuning parameter		Required support	
	Static	Dynamic	V	f	HW	SW
Clock Gating		X		X	X	~
Voltage Scaling	X	X	X		X	X
Power Gating	X		X		X	X

Key points for effective power management

- Exploit partial activity
  - Disable parts of the system when not needed
- SW does part of the work, HW dependencies do the rest
- Exploit existing system framework
  - Track dependencies
  - Track usage
  - System constraints assertion
  - Chains of notification

Driver support required

- Support efficientely OFF modes
- Use constraints to require operational restrictions

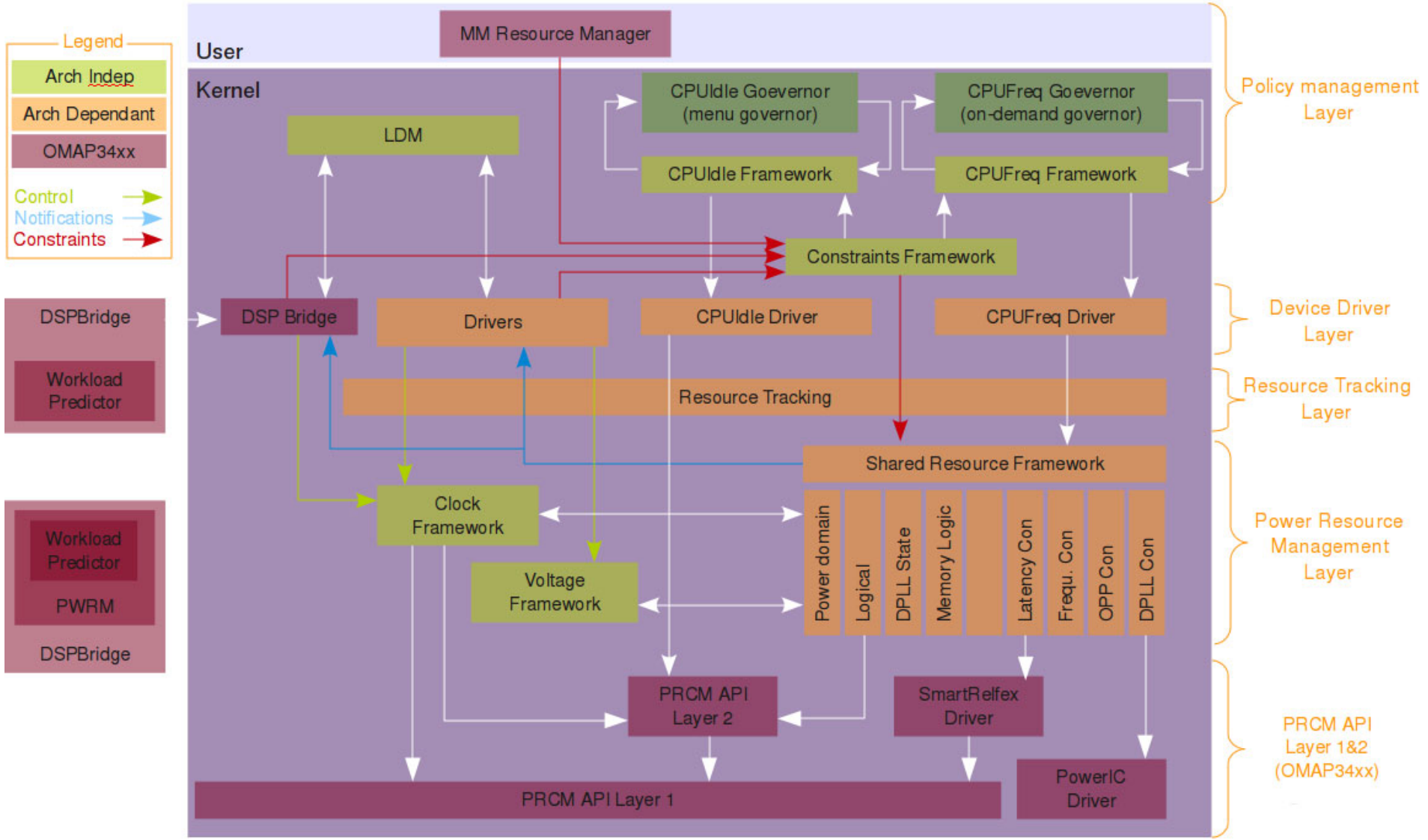
Power Management Techniques

Modern approaches to Power Management (cont'd)

- Control power sources
  - Clock and power domains
  - Voltage domains
- Inactive state
  - Power saving in OS idle
  - Automatic choice among C-states (*idle states*)
  - System-wide sleep states
- Active state
  - Dynamic power management
  - Automatic choice among P-states (*performance states*)

Power Management Techniques

What we already have (OMAP35xx Example)



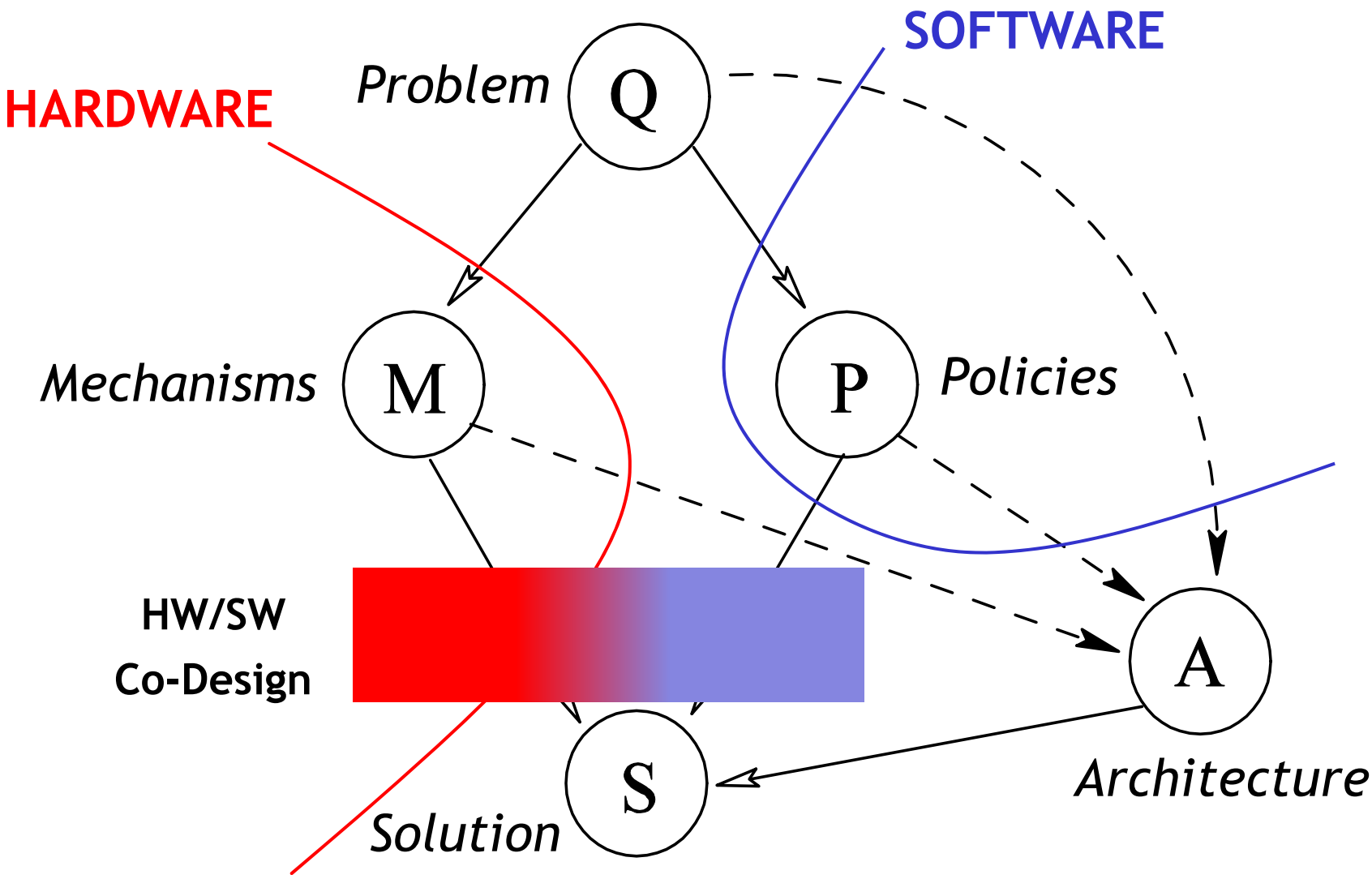
Hardware support

- Observation points
- Control points
- Power management *mechanisms*

Software support

- HW logical view
- Control software
- Control *policies*

HW/SW co-design for true holistic power management

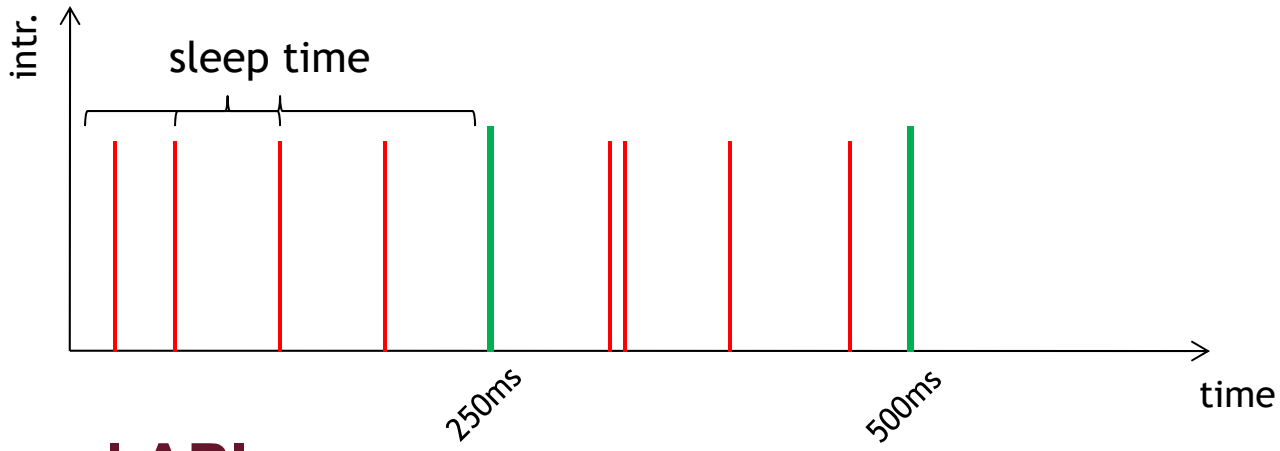

SOFTWARE
HARDWARE
Problem
Q
Mechanisms
M
P
Policies
HW/SW
Co-Design
A
S
Solution
Architecture

Main Linux PM features

- Platform-specific code
  - Idle-loop, timekeeping (dynamic tick) & clock tree, latency
- Resource Hibernation Frameworks
  - Switch off unused subsystems
  - Low-power states, C-states, suspension (to RAM) and hibernation (to disk)
- Resource Tuning Frameworks
  - Adapt performances to needs
  - P-states and OPPs
- Main focus on x86 architecture
  - Custom and different PM development for SoC-based embedded systems
  - Increasing emphasys on embedded systems

Deferrable timers

- Better exploitation of dynamic-tick, by sleeping longer
- Non-critical timeouts can run later, when the processor wakes up for other reasons

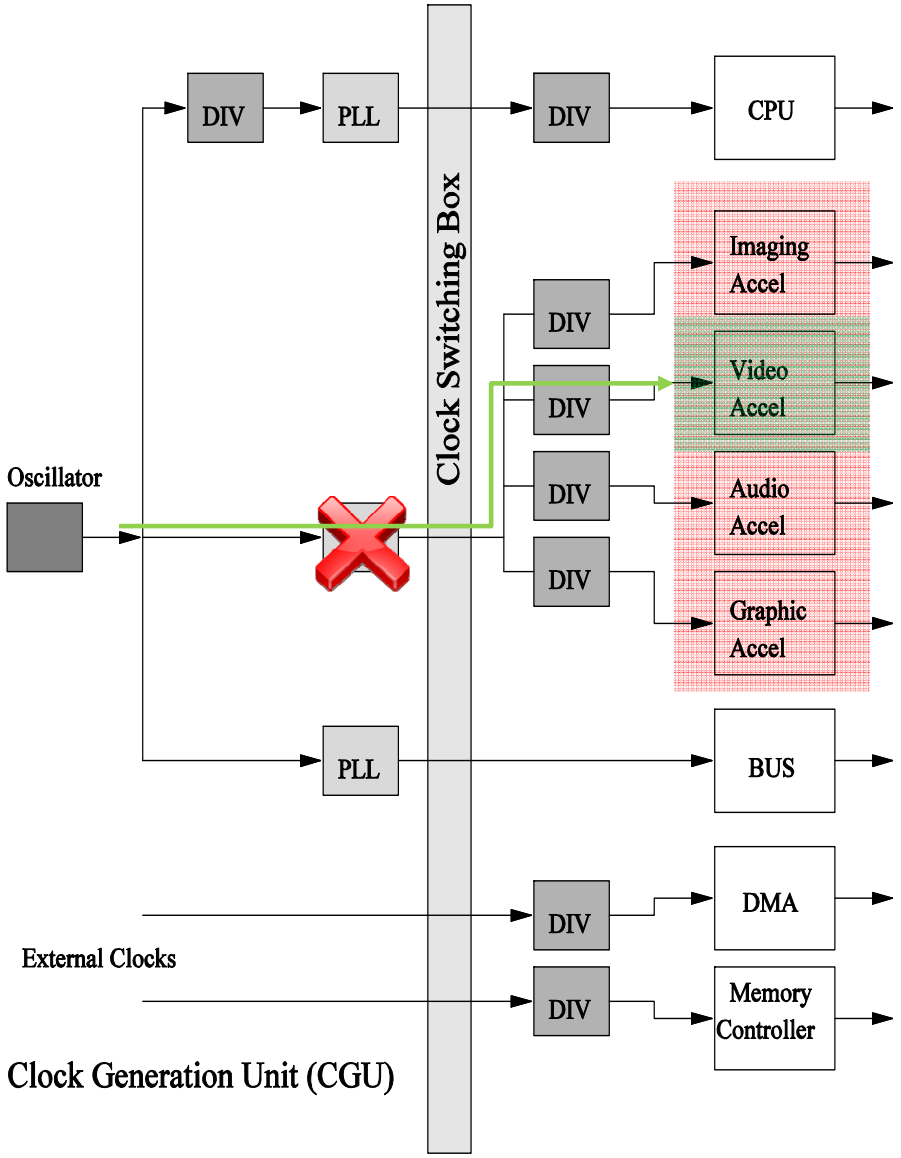


New kernel APIs

- Define timers to be deferrable through appropriate API calls
- Available to user-space application, too

Centralized control for clock distribution

- Track clocks dependencies
- Abstract API specification
  - clock rate set/get
- Required device-driver cooperation
  - aggressive get/put clocks



Keep track of devices power dependencies

- Optimize regulators usage and efficiency
- Current sinks dependency tracking
- Dynamically control regulator modes

Optimize regulator efficiency

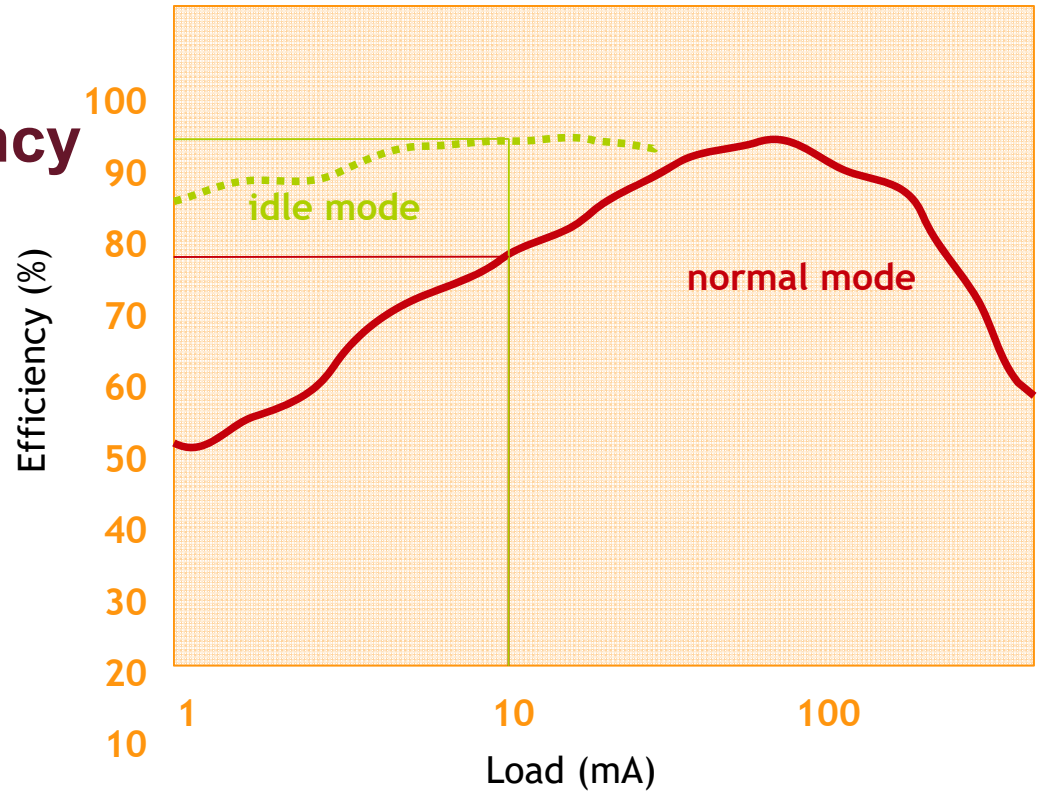
- Depend on current load

e.g. with 10mA load

70% @normal ~ 13mA

90% @idle ~ 11mA

Saving ~ 2mA



Trace admissible latency values

- Power saving actions could inflence system latency

Provide a platform-independent interface

- Set of suitable operations
  - Set/update maximum latency for a driver
  - Remove latency constraints
  - Query system-wide constraints
  - Notify changes

Framework for CPU idle states optimization

- Static power consumption optimization
- Support for multiple idle states, with different power/performance profiles
  - Power consumption, state preservation and constraints, latency

Find appropriate trade-off between

- Application expectations (latency)
- Power saving in idle states

Framework for CPU active states optimization

- Dynamic power cosumption optimization
- Support for DVFS mechanism

Different optimization policies

- *On-demand governor*

  Scaling is based upon current load

- *Conservative goveror*

  Battery-fair policy

Centralized optimization management

- Single optimization policy
- Require system-wide modeling

Configuration states

- Set of power/performance pairs
- These states are called **Operating Points** (OP)

  Defined statically

Optimization strategy

- OPs switch is performed dynamically
- Considering application requirements
- According to system-wide optimization policy

Distributed coordination management

- Devices have specific (local) PM capabilites
- Applications have specific performance requirements

Provide support for a suitable communication strategy

- Among drivers, applications and power manager subsystem
- To better exploit user expectations while meeting power budget

Requirements aggregation

- Provide system-wide requirement perspective
- Support the selection of appropriate (local) configurations

  Meeting *all* requirements

		Power component optimization		Exploited HW mechanism		
		Static	Dynamic	Clock Gating	Voltage Scaling	Power Gating
Pure OS	CPUFreq		X		X	
	CPUIdle	X		X	X	X
	Clock Fw		X	X		
	V/I Fw		X		X	X
	S/R Fw	X				X
Cross Layer	DPM (centralized)	X	X	X	X	X
	QoSPM (distributed)	X	X	~	~	~

Systems are becoming more and more complex

- Multiple applications and usage scenarios
  - Run-time changing requirements
- Multiple devices, subsystems
  - Asymmetric (DSPs) and symmetric (manycores)
- Shared resources among different devices/applications
  - Computation, memory, energy, bandwidth…

Requirements

- Time adaptability
- Retargetability
- System-wide resources management

ACPI Specifies hw-sw interface

- Hw: standardized control mechanisms; Sw (OS): PM policies

Main component: ACPI System Description Tables

- Provided by firmware, describe interface with Hw
- Portable code (AML) interpreted by the OS

Global system state	Software runs	Latency	Power consumpt.	OS restart required	Safe to disassemble	Exit state electronically
G0 Working	Yes	0	Large	No	No	Yes
G1 Sleeping	No	>0, varies with sleep state	Smaller	No	No	Yes
G2/S5 Soft Off	No	Long	Very near 0	Yes	No	Yes
G3 Mechanical Off	No	Long	RTC battery	Yes	Yes	No

Power states for single devices

- D0 (Fully on) to D3 (Off)

Performance states (P0-Pn)

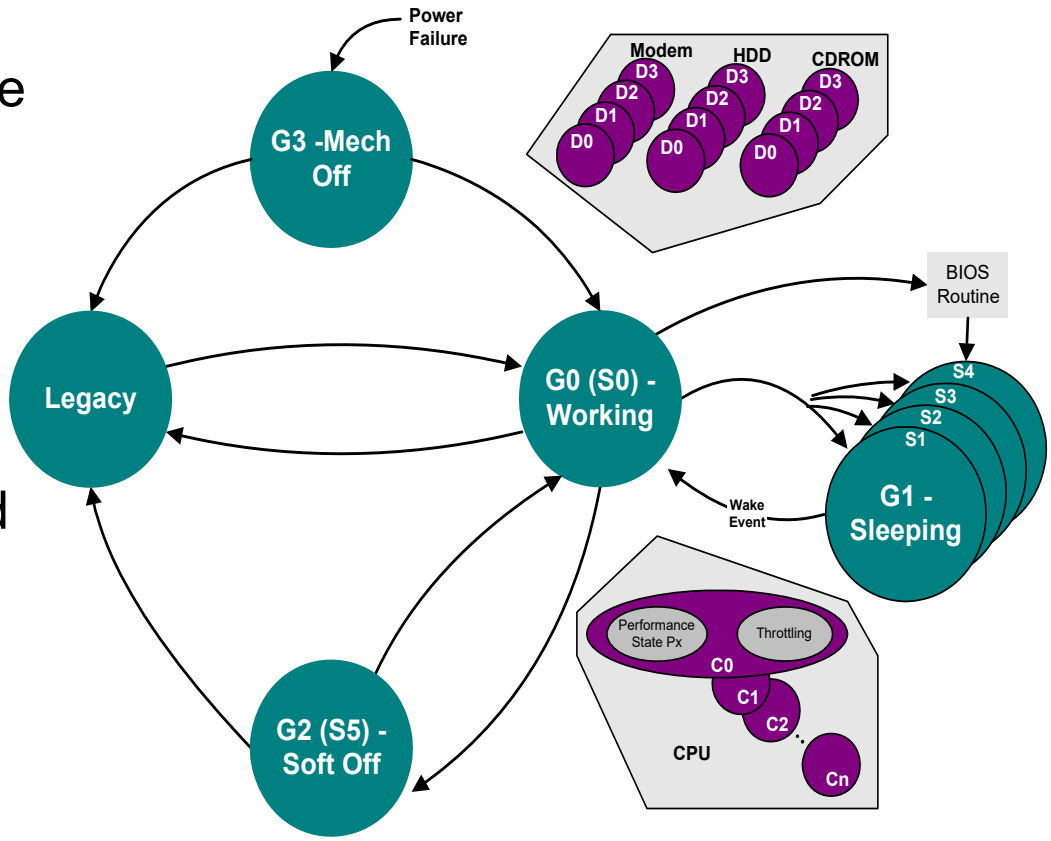
- Defined below C0 for the processor and of D0 for the other devices
- Goal is trading-off performance/power

G1: sleep states (S1-S5)

- differing for consumption, wakeup latency and saved context

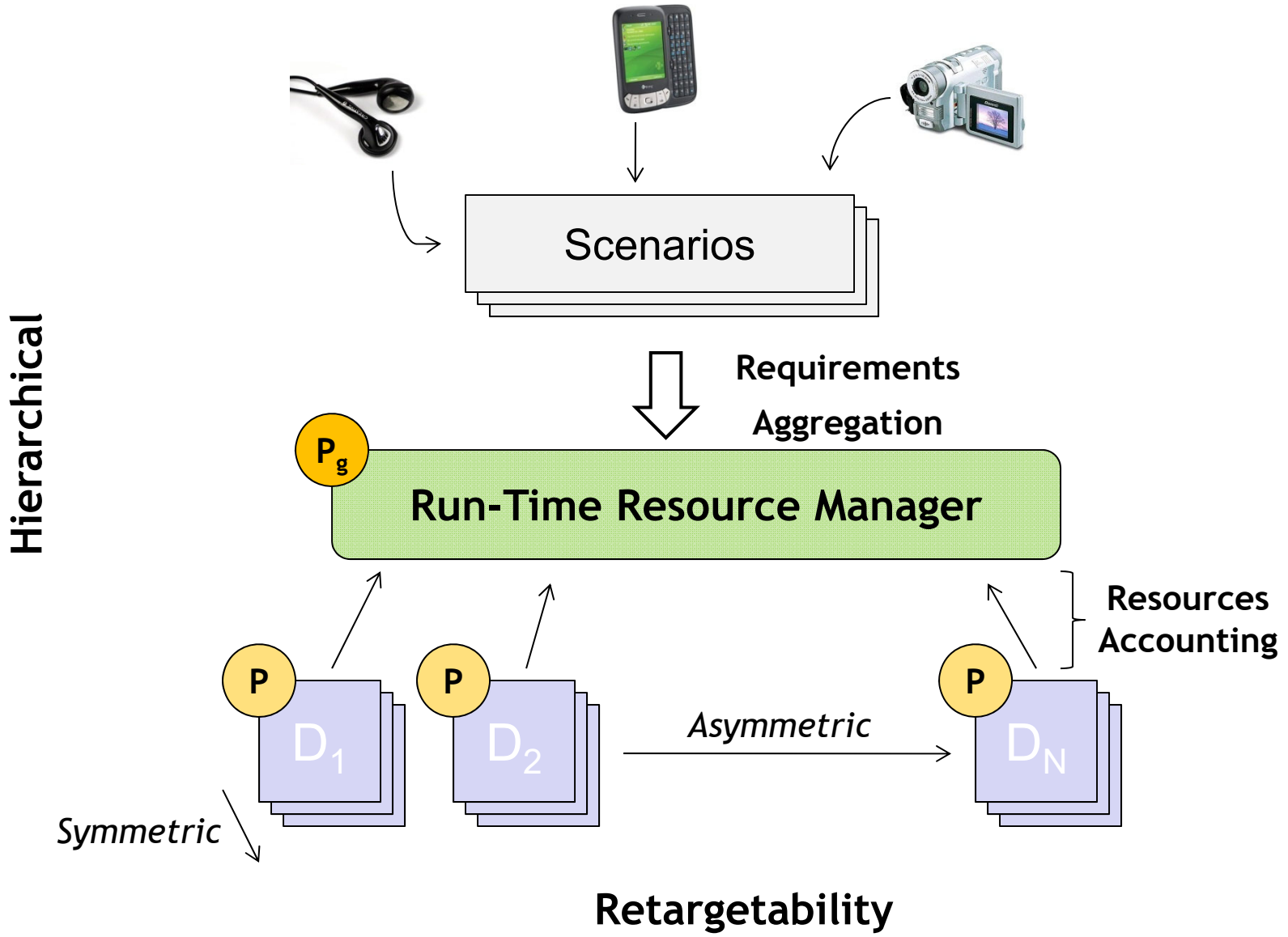
G0: sub-states (C0-C3)

- differing for only the power of processor



Our Research Directions

Run-time resource management



Acknowledgements

People involved, related projects and industrial contributors

Contributors to this presentation and to the ongoing research activity on run-time management for energy efficiency

- **Patrick Bellasi**: In 2014 joined ARM, UK
- **Simone Corbetta**: In 2013 Joined IMEC, Micron
- **D. Zoni, F.Terraneo, G.Massari**: Post-doc at POLIMI
- **Simone Libutti**: In 2018 Joined IBT Solutions

Related EU projects

- **MULTICUBE** - STREP Project (24 Months end 2011)
- **2PARMA, COMPLEX**– STREP, IP projects (3 y, end 2013)
- **HARPA, CONTREX** – STREP, IP (3 y, end fall 2016)
- **MANGO** – IP (3 years, end fall 2019)
- **RECIPE** – IP (3 years, end fall 2021)