Computer Architecture *and* Organization

Lecture Series for VIT, Chennai

Jaiber John, Intel

2022 - 2023

Module 7
Energy Efficient Architectures

Challenges and Designs

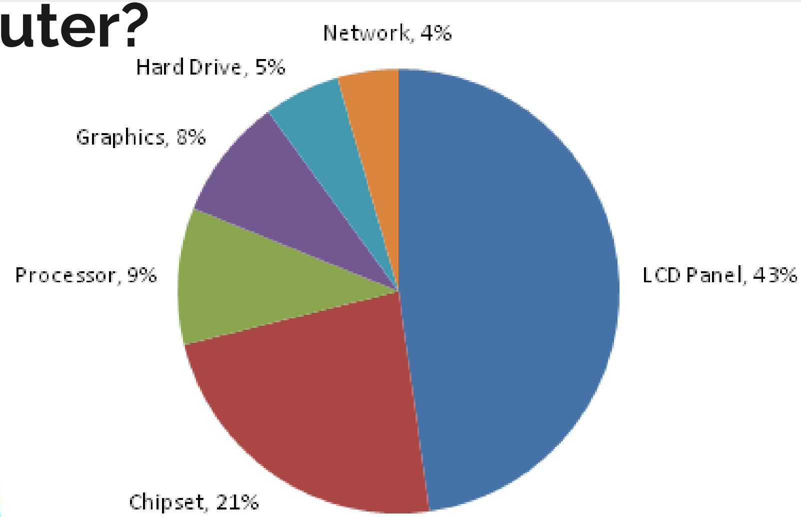
Energy Efficient Architectures

- Computer Power trends
- CMOS Power dissipation
  - Static and Dynamic power
- Power management techniques
  - Clock gating
  - Power gating
  - DVFS
  - OS Scheduling
- ACPI
  - Active and Idle states
  - Sleep states

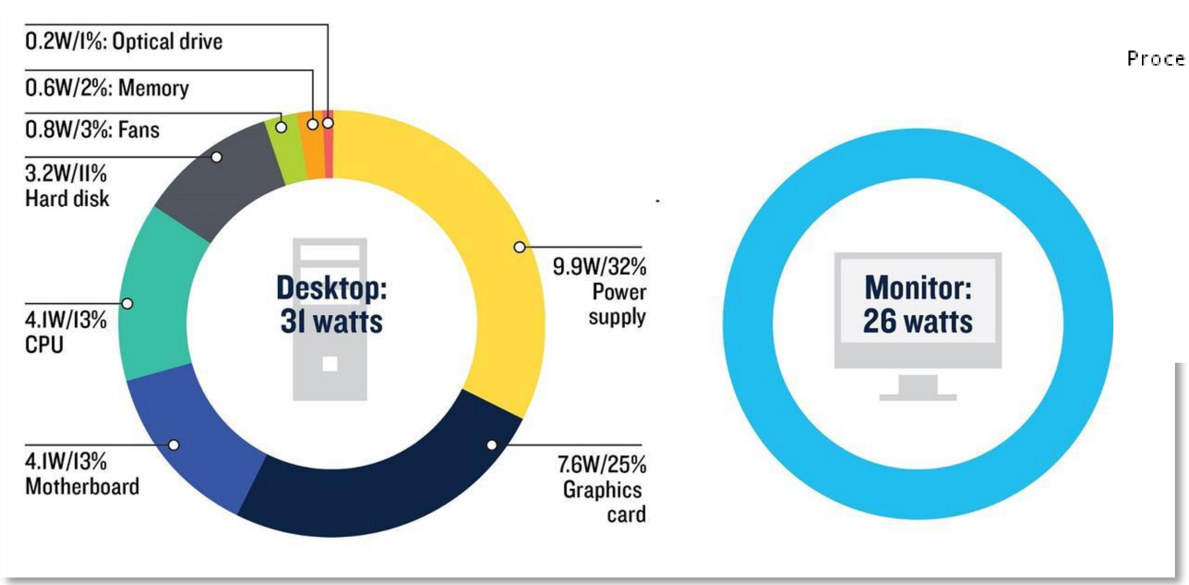
- CPU Frequency Scaling
- Low Power designs
  - Processor architectures vs. Power
  - RISC vs. CISC

- Low Power Strategies
  - SoC/Integration
  - Special purpose processors
  - big.LITTLE
  - Efficient HW design
  - Custom Instructions
  - Energy aware scheduling

What consumes power in Computer?



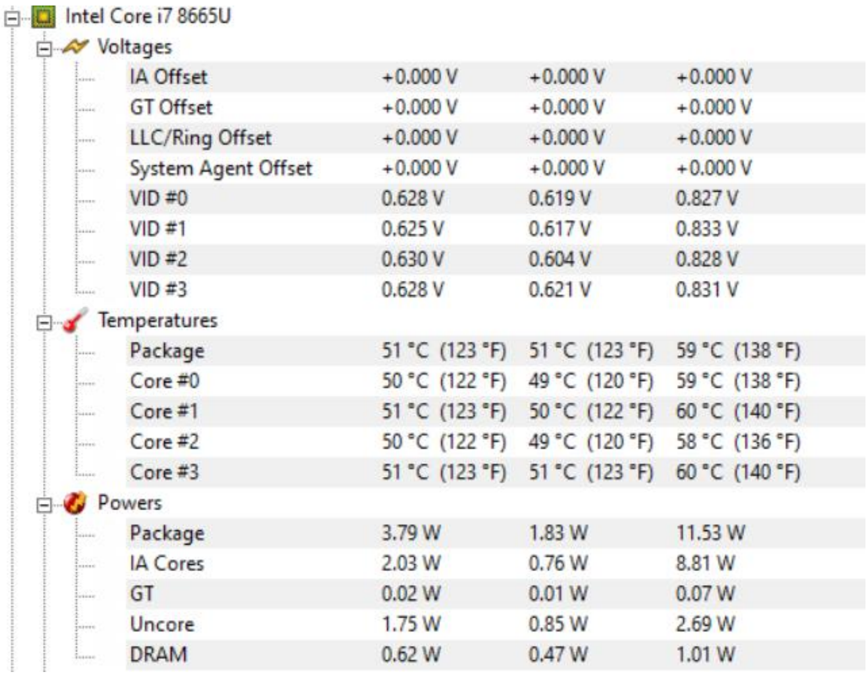
Power draw in a sample **Laptop**



Power draw in a sample **desktop** and monitor in idle mode

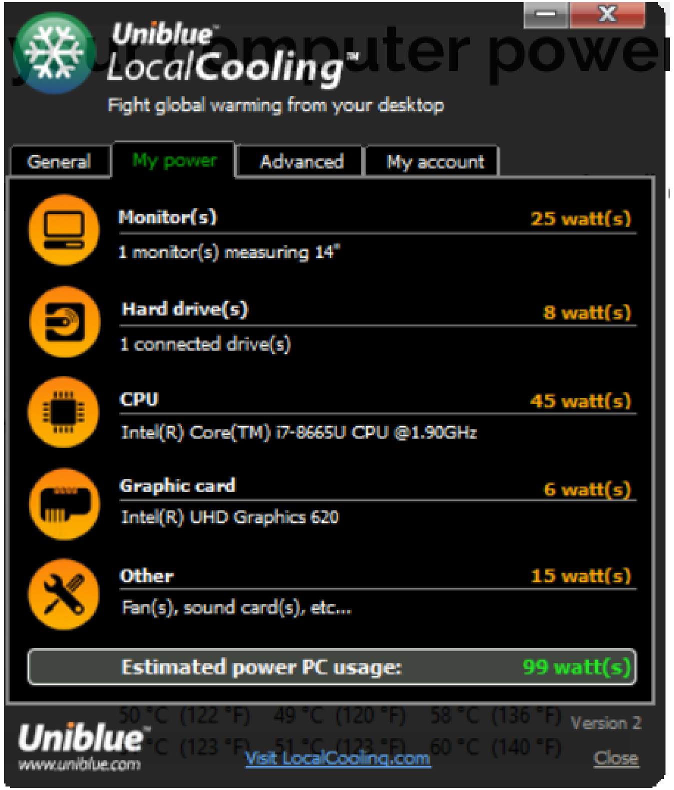
Check your computer power usage

- Use Apps like “LocalCooling”, “HWMonitor” on Windows



Intel Core i7 8665U			
Voltages			
IA Offset	+0.000 V	+0.000 V	+0.000 V
GT Offset	+0.000 V	+0.000 V	+0.000 V
LLC/Ring Offset	+0.000 V	+0.000 V	+0.000 V
System Agent Offset	+0.000 V	+0.000 V	+0.000 V
VID #0	0.628 V	0.619 V	0.827 V
VID #1	0.625 V	0.617 V	0.833 V
VID #2	0.630 V	0.604 V	0.828 V
VID #3	0.628 V	0.621 V	0.831 V
Temperatures			
Package	51 °C (123 °F)	51 °C (123 °F)	59 °C (138 °F)
Core #0	50 °C (122 °F)	49 °C (120 °F)	59 °C (138 °F)
Core #1	51 °C (123 °F)	50 °C (122 °F)	60 °C (140 °F)
Core #2	50 °C (122 °F)	49 °C (120 °F)	58 °C (136 °F)
Core #3	51 °C (123 °F)	51 °C (123 °F)	60 °C (140 °F)
Powers			
Package	3.79 W	1.83 W	11.53 W
IA Cores	2.03 W	0.76 W	8.81 W
GT	0.02 W	0.01 W	0.07 W
Uncore	1.75 W	0.85 W	2.69 W
DRAM	0.62 W	0.47 W	1.01 W

HWMonitor App



Monitor(s) 25 watt(s)
1 monitor(s) measuring 14"

Hard drive(s) 8 watt(s)
1 connected drive(s)

CPU 45 watt(s)
Intel(R) Core(TM) i7-8665U CPU @1.90GHz

Graphic card 6 watt(s)
Intel(R) UHD Graphics 620

Other 15 watt(s)
Fan(s), sound card(s), etc...

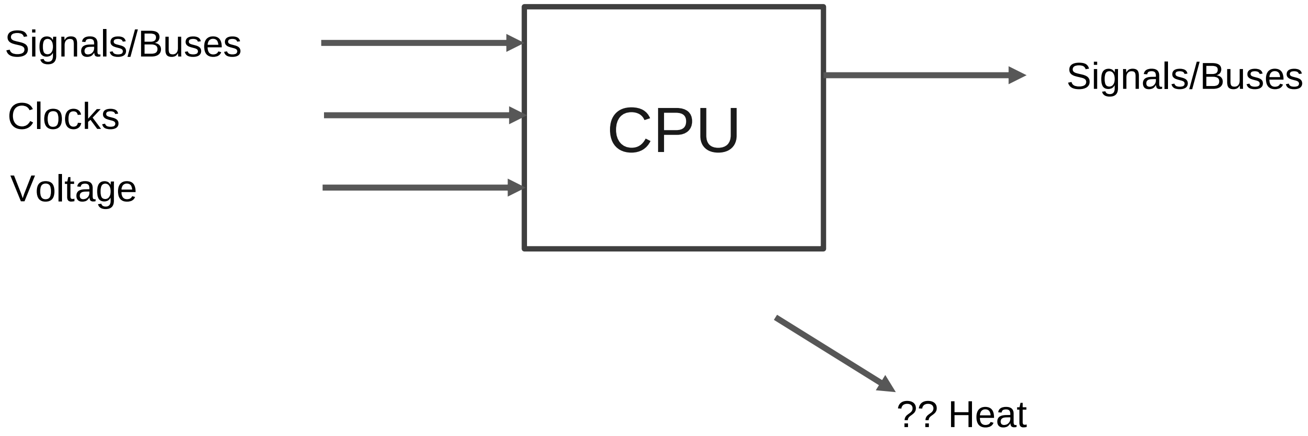
Estimated power PC usage: 99 watt(s)

LocalCooling App

CPU Power model

- Power vs. Energy
  - Power – Energy converted or transferred per unit of time

- What happens to the energy consumed by CPU?



Power Density Trends..

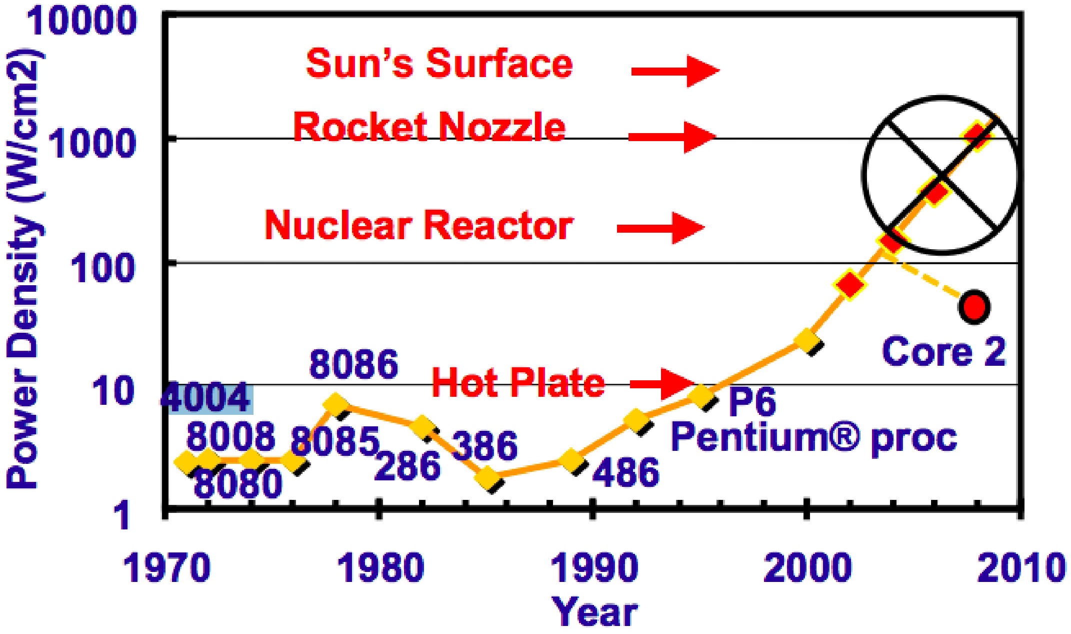
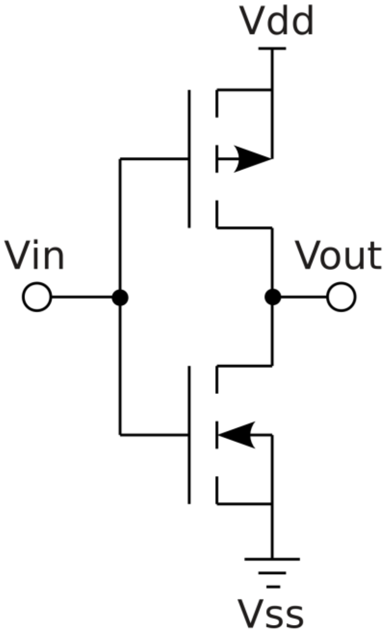
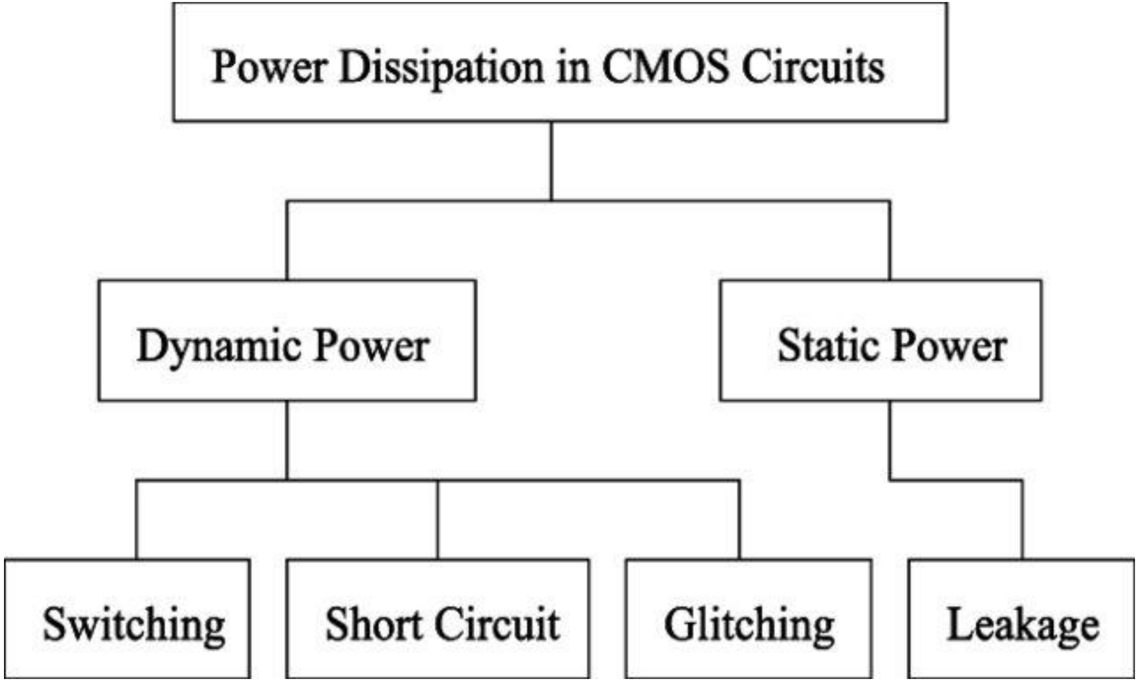


Figure 1.1 Today, typical CMOS microprocessors operate at around the power density of a nuclear reactor, *e.g.* 65 W and 130 W for Intel® CoreTM2 Duo E6320 (die size of 143 mm^2) and Intel® CoreTM i7-990X (die size of 239 mm^2), respectively. Courtesy of Shekhar Y. Borkar, Intel Corp. [1].

CMOS Power dissipation





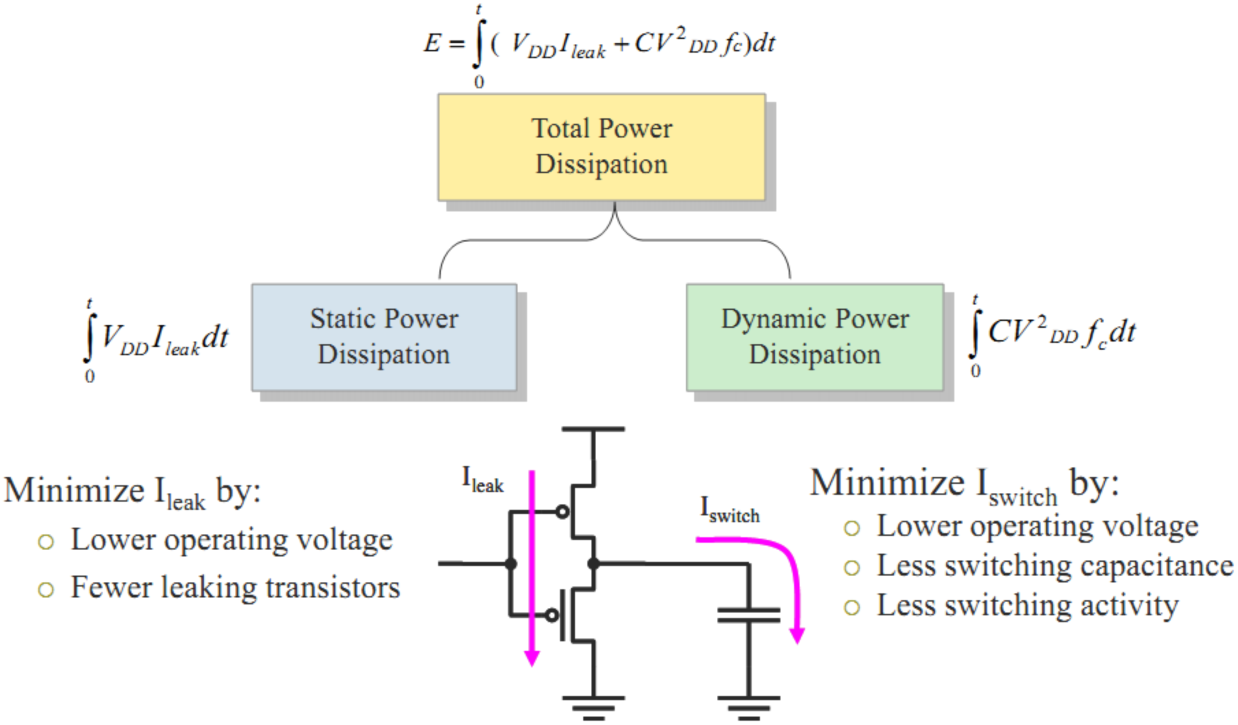
CMOS Power Dissipation

CMOS – n-type + p-type MOSFET

- Very high leakage at < 22nm
- So, 5 – 22 nm -> FinFET (non-planar)
- < 5 nm -> RibbonFET

CMOS Power Dissipation

- **Static** power dissipation
  - Mostly due to Leakage
  - Occurs when all inputs are held at some valid logic level and the circuit is not in charging states
- **Dynamic** Power Dissipation
  - Due to the current that flows only when the transistors of the devices are switching from one logic state to another.
  - Depends on frequency and the type of internal node



$$E = \int_0^t (V_{DD} I_{leak} + CV^2{}_{DD} fc)dt$$

Total Power Dissipation

Static Power Dissipation $\int_0^t V_{DD} I_{leak} dt$

Dynamic Power Dissipation $\int_0^t CV^2{}_{DD} f_c dt$

Minimize I_{leak} by:

- Lower operating voltage
- Fewer leaking transistors

Minimize I_{switch} by:

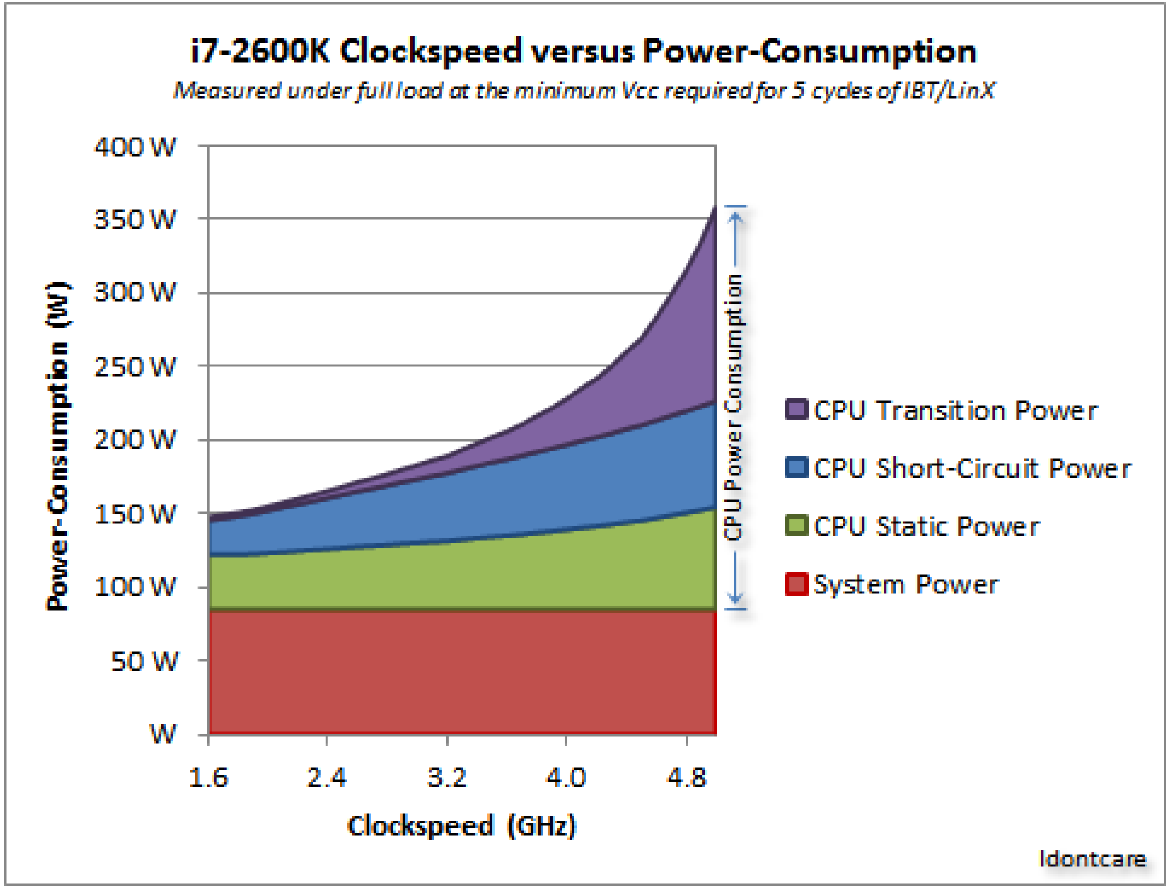
- Lower operating voltage
- Less switching capacitance
- Less switching activity

CMOS Power Dissipation

Static and Dynamic Power vs Frequency

- Frequency impacts dynamic power

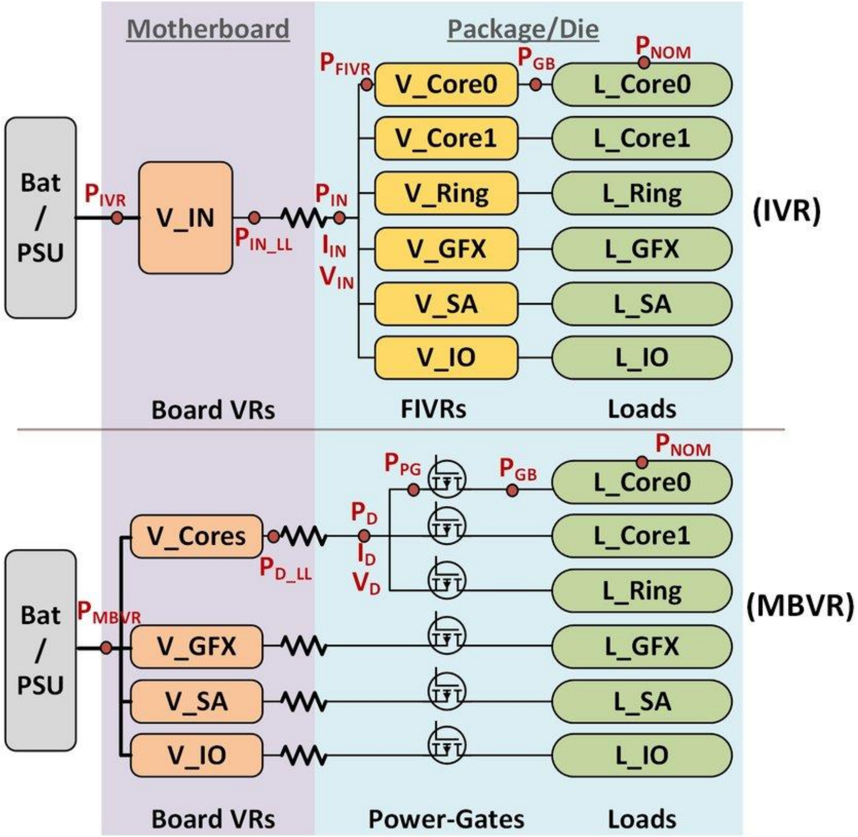
- Static power does not vary as much with frequency



Processor Power Delivery

- The power delivery network (PDN) supplies reliable power to all transistors

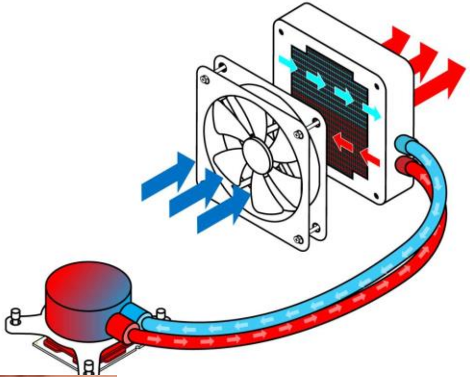
- Challenge:
  - Deliver with excellent efficiency
  - Swiftly respond to changes in power draw.



Power Delivery System

Processor Cooling

- Cooling is very important for CPU
- Types
    - Active (Fan)
    - Fanless cooling
    - Liquid Cooling



Liquid cooling

Fan + Heatsink

Fanless cooling

TDP, Energy, Power, Turbo

- TDP (Thermal Design Power)
  - The maximum amount of heat generated by a computer chip that the cooling system in a computer is designed to dissipate under any workload

- Max or Peak Power
  - Maximum power that can be consumed by CPU, at highest frequency of operation

- Turbo Mode
  - Boost frequency – overclock for short period of time to handle bust of workload activity

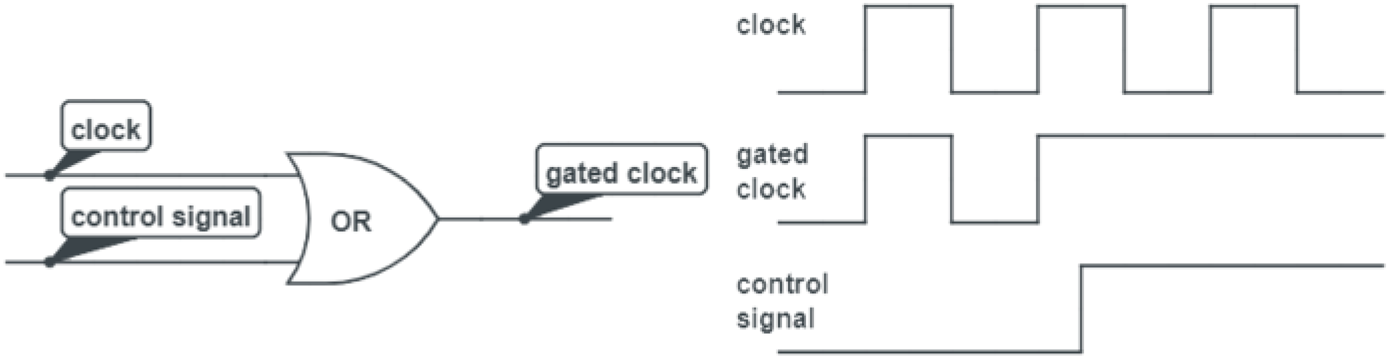
Power Management

Contents

- Hardware Power management techniques
  - Clock, Power gating
  - DVFS
  - ACPI
- Software
  - Idle vs active state management
  - Frequency governors (Linux)

Hardware Power management techniques

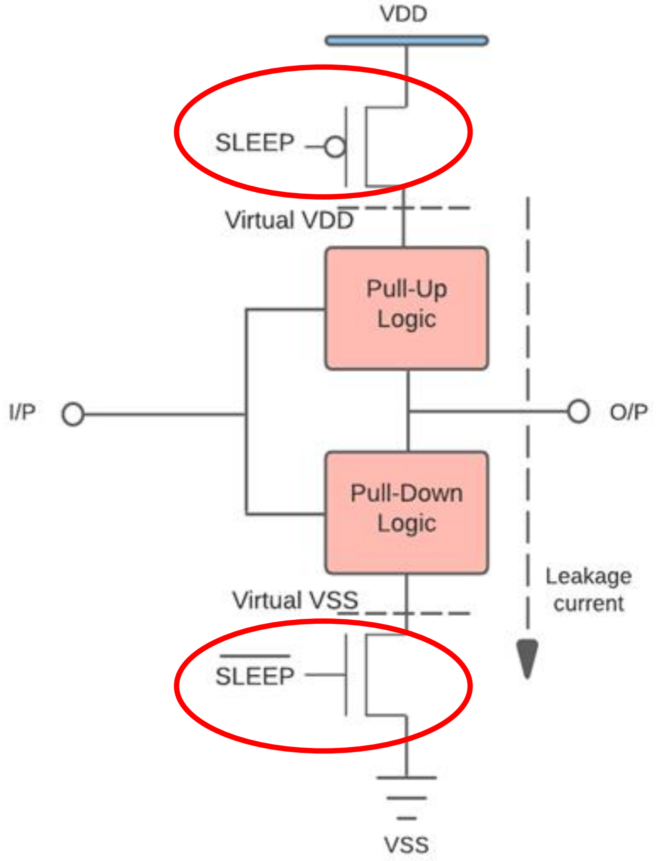
- **Clock gating**
    - Clock gating is a popular power management technique used in many synchronous circuits for reducing dynamic power dissipation, by removing the clock signal when the circuit is not in use or ignores clock signal



Simple Clock Gating

Power Gating

- Reduce power consumption, by shutting off the current to blocks of the circuit that are not in use
- Gate controlled by:
  - **software** – using driver software.
  - **hardware** – using hardware timers.
  - By using a dedicated power gating **controller** in the design.



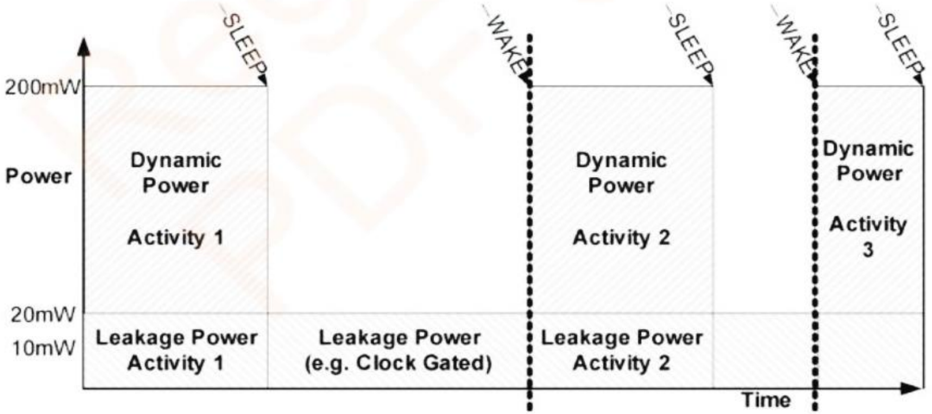
Power Gating in CMOS

Clock gating vs Power gating

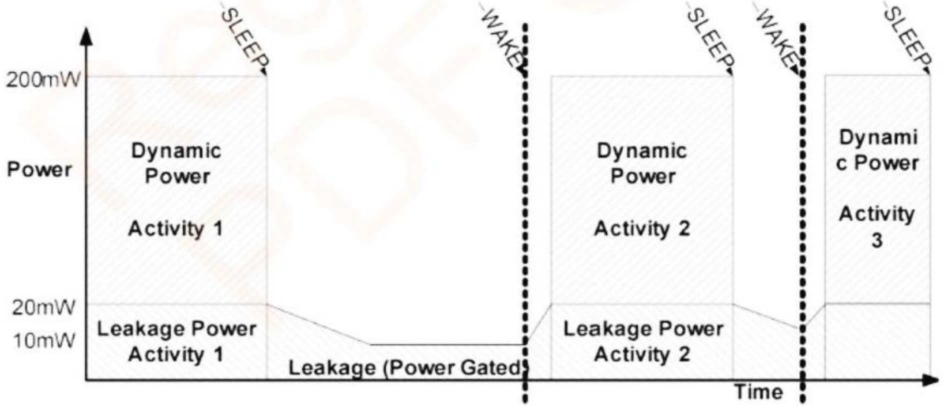
<table>
<tr><th>Clock gating</th><th>Power gating</th></tr>
<tr><td>Clock gating reduces dynamic power by reducing switching frequency.</td><td>Power gating reduces dynamic and static (leakage) power.</td></tr>
<tr><td>Clock gating is simpler to implement.</td><td>Power gating requires additional circuits to maintain states.</td></tr>
<tr><td colspan="2">Power gating is more efficient at saving power.</td></tr>
<tr><td colspan="2">Both require extensive verification to cover corner cases.</td></tr>
</table>

Clock gating vs Power gating

Clock gating
still gives leakage power consumption



Power gating
reduces leakage power consumption with power on/off transition overhead

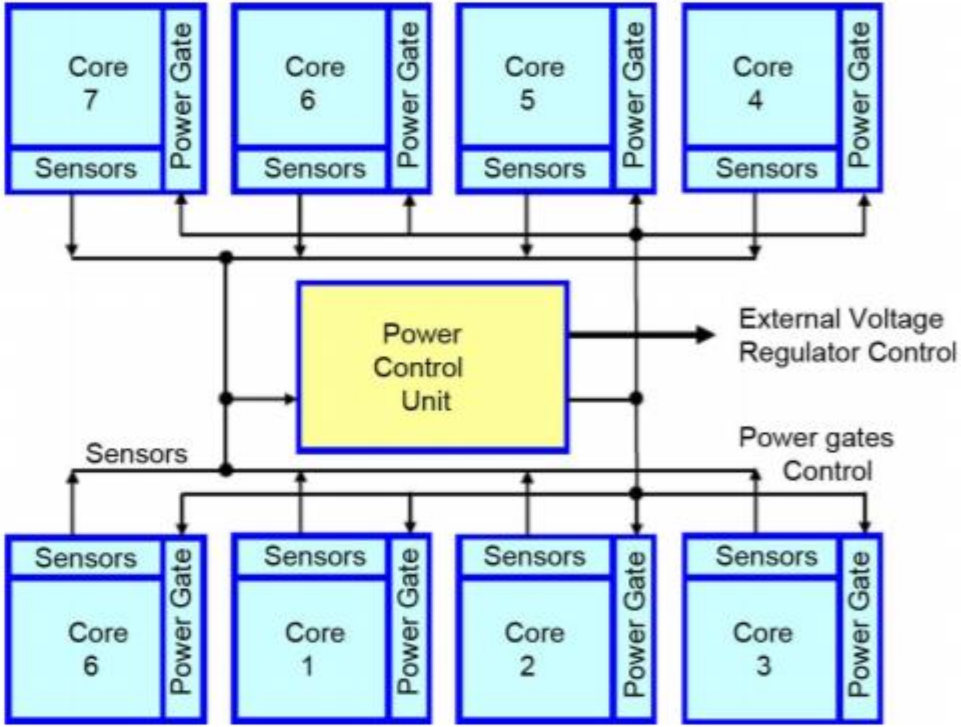


DVFS – Dynamic Voltage and Frequency Scaling

- Dynamic power (switching power) = **$C * V^2 * A * f$**
  - where C is the capacitance being switched per clock cycle
  - V is voltage
  - A is the Activity Factor
- Voltage has higher impact (since V^2), controlled by VRs
- Frequency – controlled by the PLL (Phase locked loop)
- Disadvantages:
  - Adds complexity to design and verification
  - Introduces performance fluctuations
  - If not done correctly, can end up using more energy

Power Control Unit

- **PCU** - Tiny microcontrollers within the processor or SoC
- Monitors sensors, and controls power gating and voltage regulation
- Why? Since power control done directly by OS is complicated

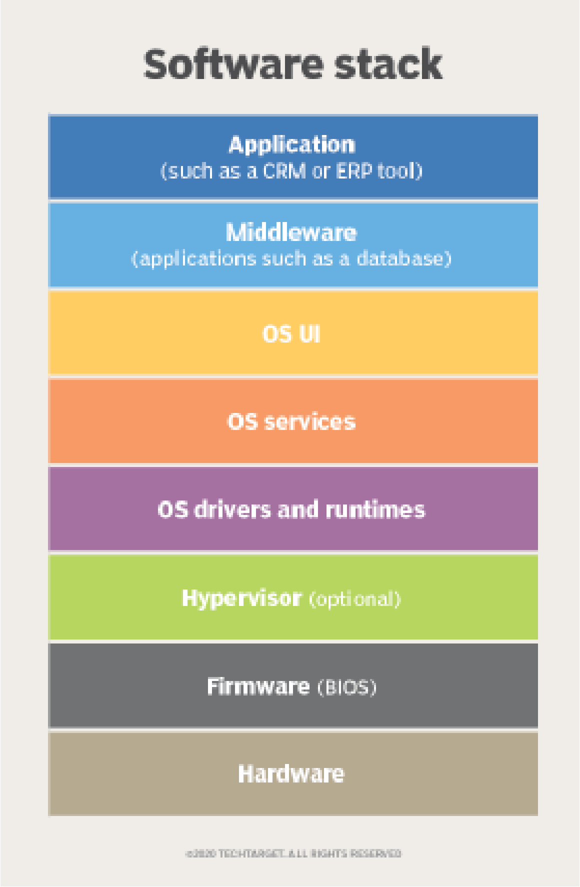


Sample Power Control Unit

ACPI – Advanced Configuration and Power Interface

- **ACPI** - Open standard, implemented in firmware (BIOS) of most x86 systems
- Abstracts hardware configurations and presents to OS

- **OSPM** – OS-directed Power Manager controls power states
- OSPM calls ACPI to manage power states

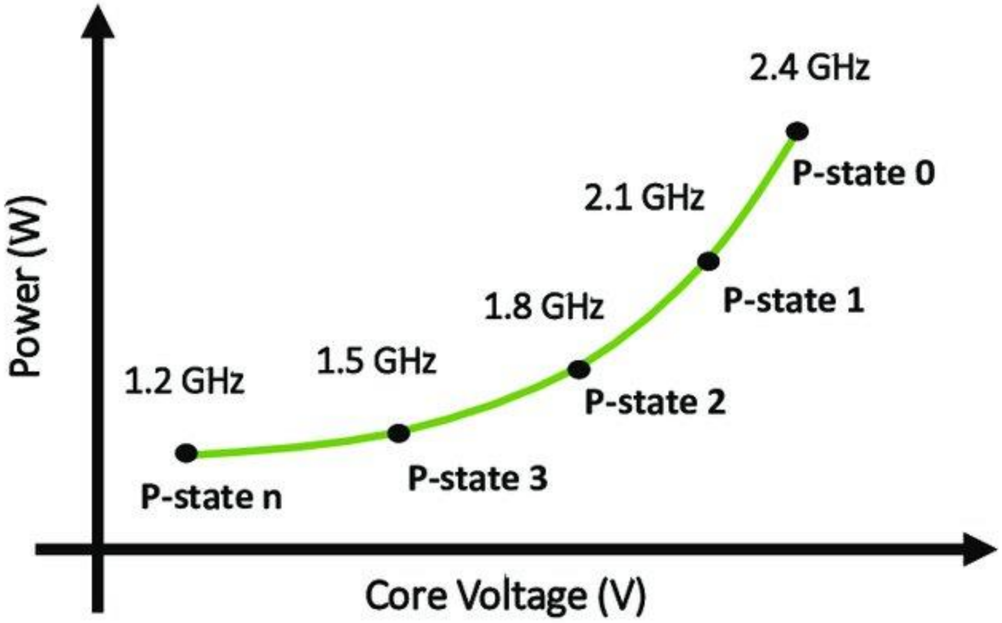


OSPM

ACPI

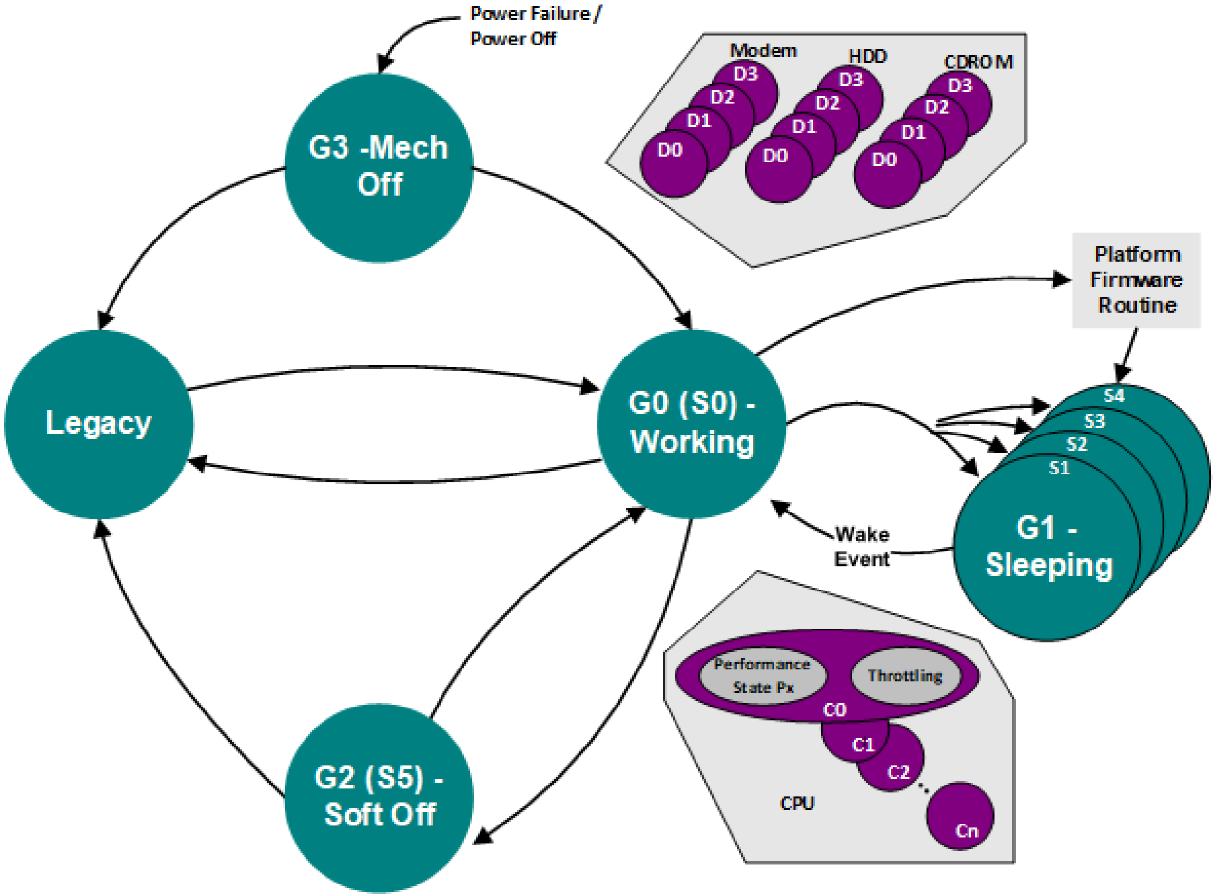
ACPI Power States

- Instead of directly controlling Voltage or frequency, ACPI provides ‘states’
- **Performance** states
  - P0, P1, P2, P3..
- **Power** states
  - C0, C1, C2, C3..
- **Device** states
  - D0, D1, D2, D3..



Example: P-states (V vs. f) of a processor

ACPI State Machine

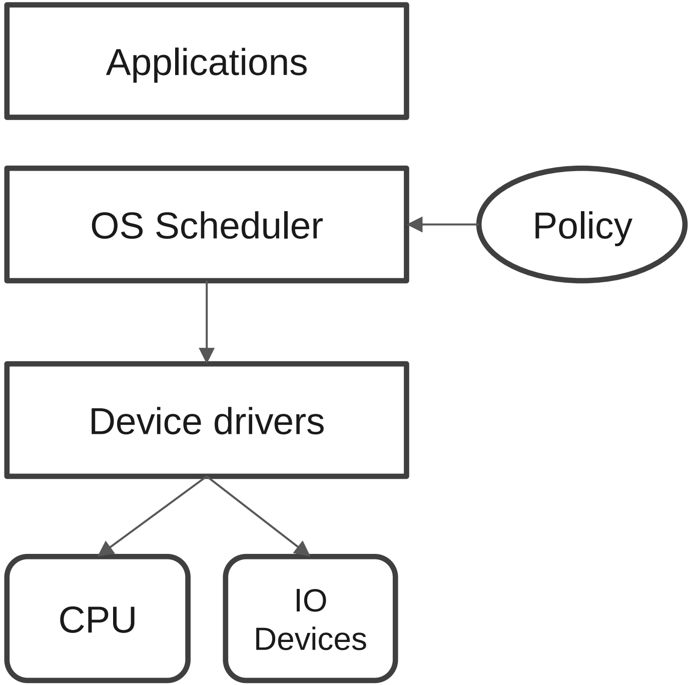


- Actual power management is quite complicated
- Involves Firmware, Operating System and device drivers

Power Management – Software level

- OS plays a major role in power management
- It is usually the ‘brain’ behind power management control

- OS has knowledge of task activity, priority, etc
- OS uses system provided knobs (e.g., ACPI) to set power states



Active and Idle states

- Generally, system state can be:
    - **Active** – when tasks are running. Performance state can be adjusted.
    - **Idle/Sleep** – no tasks available to run. Power states can be adjusted. Can easily wake up to active state.
    - **Suspend** – Suspend to RAM or Suspend to disk (hibernation). System state is maintained. Very low power required. Takes some time to wake up.
    - **Off** – System powered off, no states maintained. Need to start from Power-on.

CPU Frequency Scaling Algorithms (Linux)

Algorithms that control the frequency scaling are called ‘governors’, they implement the PM policy

Performance	Always chooses the highest performance state. Highest performance but low power savings
Powersave	Always chooses the lowest performance state. Low performance but offers high power savings
Schedutil	Chooses performance states to improve task scheduling efficiency
Ondemand	Chooses performance states based on CPU load. Higher load -> boost performance and vice versa
Conservative	Chooses performance based on CPU load like Ondemand, but does not change processor states so aggressively but in small steps

Power saving design approaches

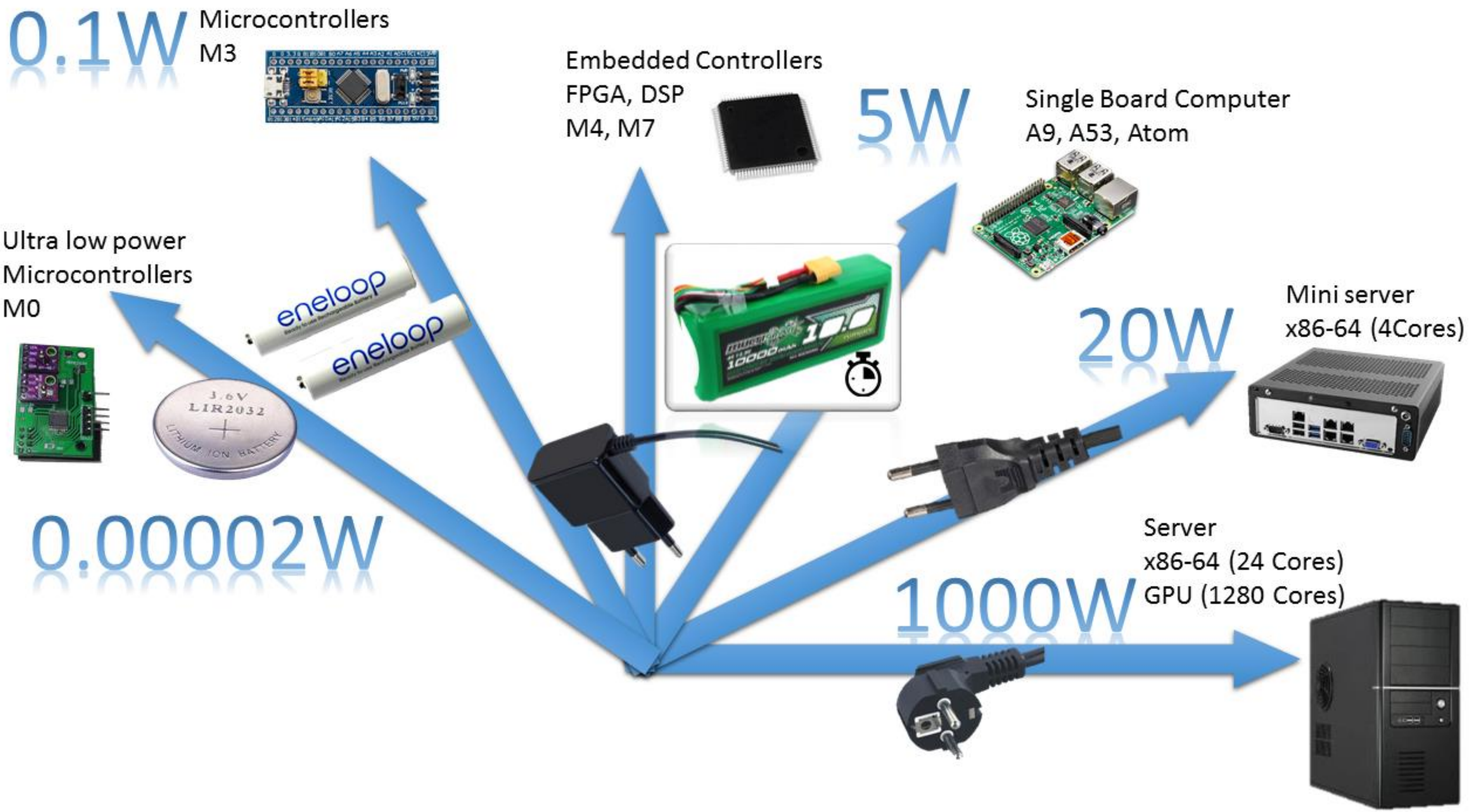
- What policy or algorithm would you devise to:
  - Optimize for thermal efficiency (control temperature) in handheld device?
  - Optimize for power efficiency in real time applications?
  - Optimize for maximum performance in a supercomputer?
  - Can predictive (ML techniques) be used to optimize power and performance?
  - Can application specific scheduling be helpful?
    - Ex: Database server vs Network server vs DL inference server
  - Design optimal scheduling for smartphones – ex. Android?
    - Only one foreground app runs, other apps frozen in background

Low Power Designs

Contents

- What are Low Power systems?
- RISC vs. CISC microarchitecture
- Strategies for low power/energy designs
    - SoC and Integration
    - Special purpose processors
    - Big.LITTLE architecture
    - Efficient Hardware designs
    - Energy aware scheduling

Power consumption of various devices



0.1W Microcontrollers
M3

Embedded Controllers
FPGA, DSP
M4, M7

5W

Single Board Computer
A9, A53, Atom

Ultra low power
Microcontrollers
M0

20W

Mini server
x86-64 (4Cores)

0.00002W

Server
x86-64 (24 Cores)
GPU (1280 Cores)

1000W

Low Power Designs

- Roughly consume less than 5-10W of power
- Finds usage in IoT, embedded, mobile devices, smartwatches,..
- Primarily dominated by microcontrollers, ARM/RISC-V designs
- SoC designs contribute to power efficiency
- Low power designs tend to be custom designs

Various factors from hardware (circuit/gate) design to software scheduling play a key role

Processors – Features and Power

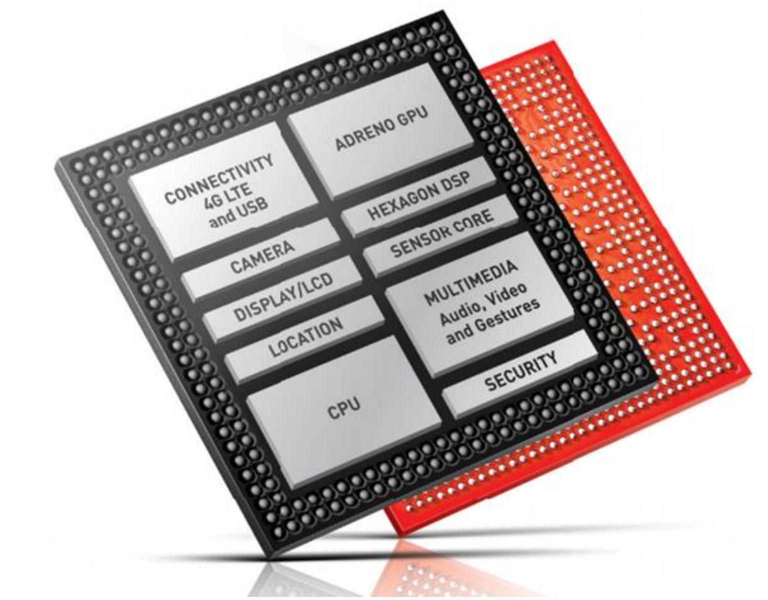
Name	Architecture	Features	Power
Arduino Uno (microcontroller)	RISC 8-bit	20MHz, USART, SPI, I2C	0.29 W
Raspberry Pi (education, robotics)	ARM 64-bit SoC	4-core, 1.5GHz, iGPU, WiFI, BT, HDMI	6 W
Shakti (embedded)	RISC-V 64-bit	2-16 cores, 1.2GHz	5-10 W *
Qualcomm Snapdragon 8 G2 (for smartphones)	ARM 64-bit, big.LITTLE, SoC	4-core, 3.2GHz, iGPU, WiFi, modem, BT, ISP, Video, Audio	7.5 W
Apple M2 (for MacBook Air)	ARM 64-bit, big.LITTLE, SoC	8-core, 3.48GHz, iGPU, neural engine, ISP	20 W
Intel AlderLake (Laptop, desktop)	x64, Hybrid	16-core, 5.5GHz, iGPU, PCIe, WiFI, AVX,	125 W
Intel SaphireRapids (Server)	X64	60-cores, several accelerators	350 W

RISC vs. CISC – power perspective

RISC (ARM, RISC-V, ..)	CISC (x86)
Designed for lower power	Designed for higher performance
Fixed width instructions, easier to decode,	Variable length complex instructions, need dedicated circuits to decode
Lower clock speed, Lesser transistors	Higher clock speed More transistors, more legacy features
Easier to pipeline OOO, speculative execution optional	Out of order, speculative execution, branch prediction, pipelining complicated
Can be “fanless” – less power for cooling	Requires fans, additional power needed for cooling

Strategy #1 for Low Power: SoC (Integration of IP)

- SoC – **System on Chip**
- Integrating multiple components –> processor, memory, controllers into a single Chip
- Big reduction in interconnect (bus) power
- Power or Clock gate entire IP blocks
- On chip memory reduces memory accesses and lowers power



Sample SoC

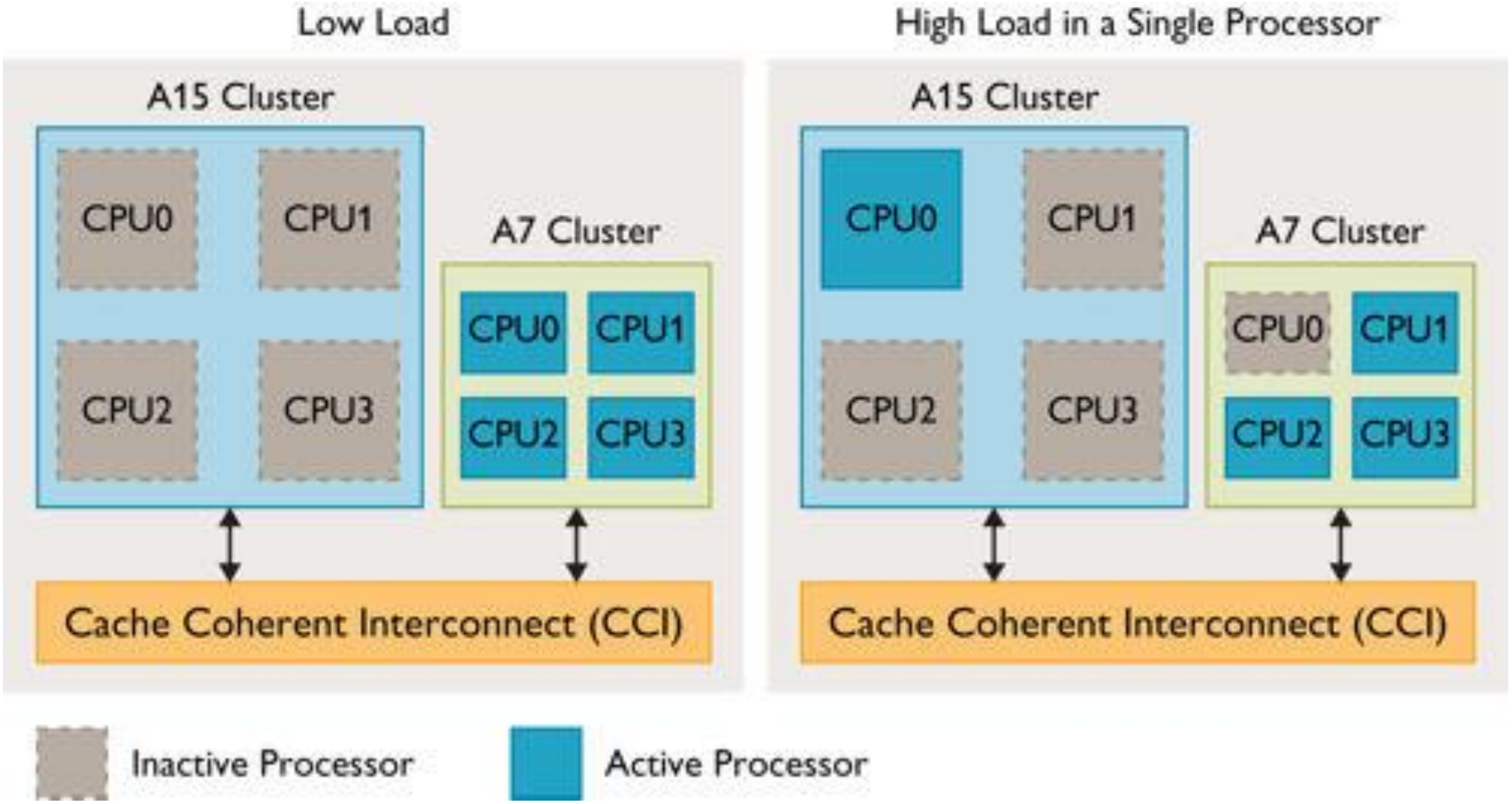
Strategy #2 – Special purpose processors

- General idea is to offload processing from CPU to special purpose processors to optimize power

Examples of special purpose processors:

- DSP – Digital Signal Processors
- ASIC – Application Specific Integrated Circuits
- FPGA – Field Programmable Gate Arrays
- TPU – Tensor Processing Unit
- IPU – Image Processing Unit

Strategy #3 – big.LITTLE architecture



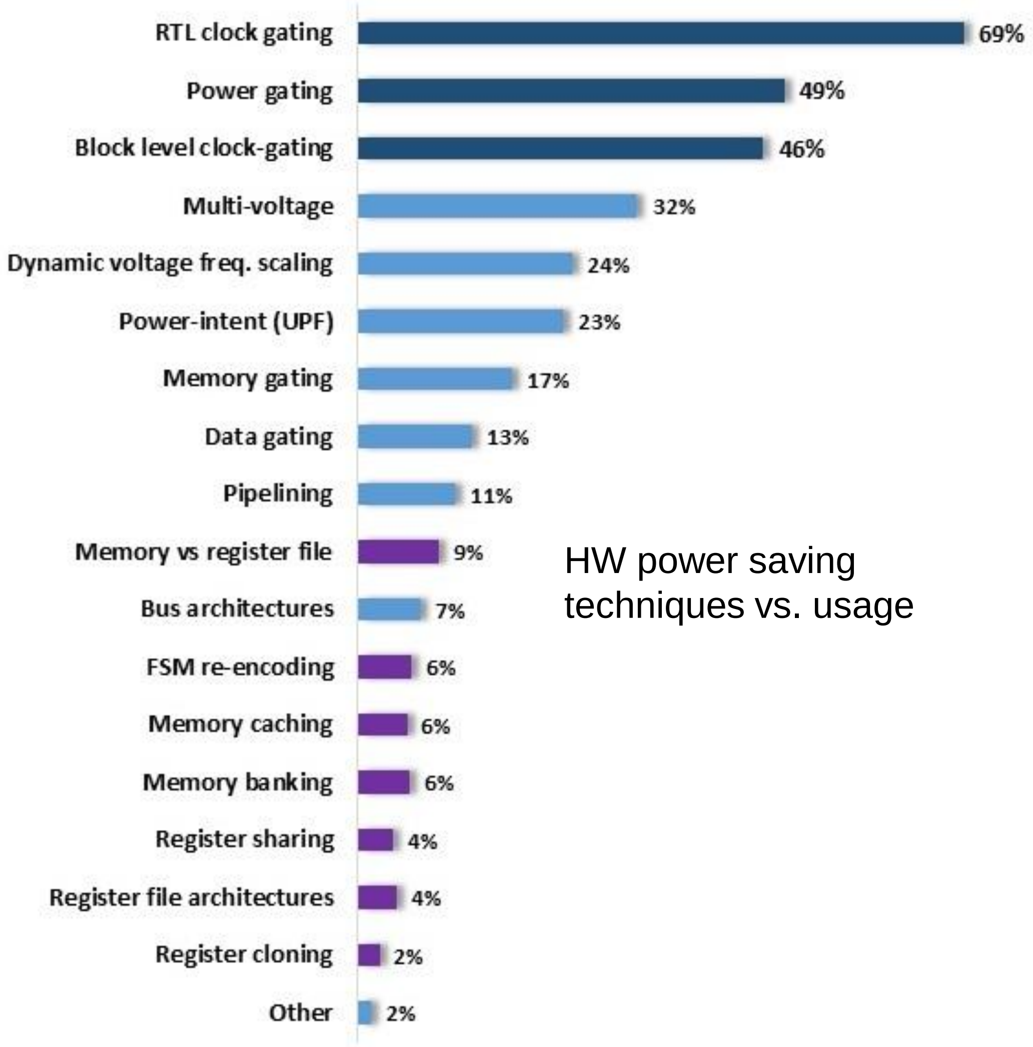
Strategy #3: contd..

- Combines high performance + low power cores into single chip
- High-performance cores used for power-intensive tasks, like gaming
- Lower-performance cores are used for less power-intensive tasks, such as web browsing and social media.
- By using the appropriate core for a specific task, the processor can reduce power consumption and extend battery life.

- Operating system dynamically switches between the high-performance and low-performance cores based on the workload.
- Commonly used in smartphones and tablets, embedded systems and IoT devices where power efficiency is a key concern.

Strategy #4: Efficient HW design

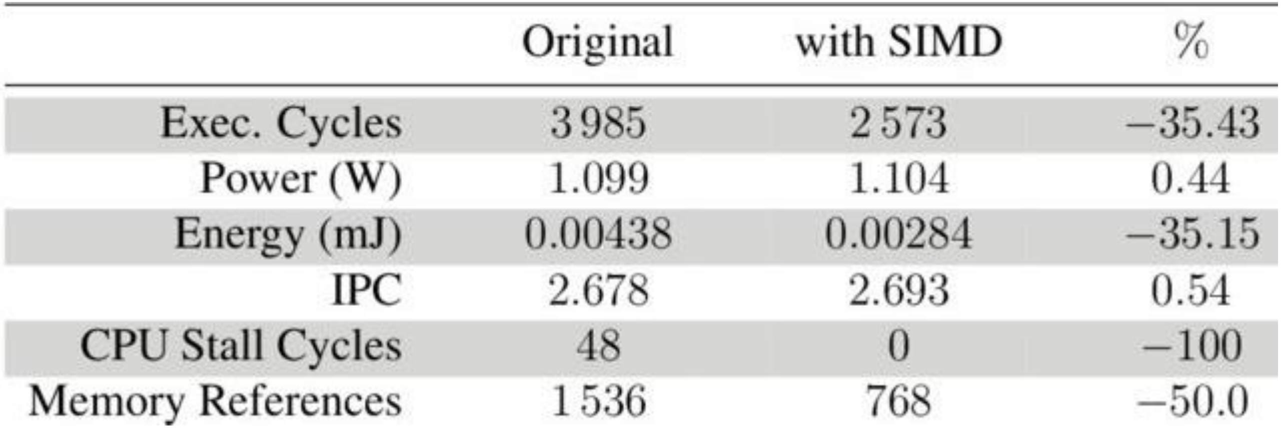
- Various circuit level techniques
- Logic Gate design
  - MOSFT vs FinFET
- Fabrication technology
  - 22nm vs 14nm
- Power aware optimization
  - Netlist simulation and synthesis
- Power gating, Clock gating, DVFS, etc..



HW power saving techniques vs. usage

Strategy #5: Custom Instructions

- Design custom instructions that can perform a set of operations efficiently than general purpose CPU instructions
- Example: Vector instructions, SIMD, cryptographic functions
- Energy can be saved, though power readings could be higher



	Original	with SIMD	%
Exec. Cycles	3 985	2 573	−35.43
Power (W)	1.099	1.104	0.44
Energy (mJ)	0.00438	0.00284	−35.15
IPC	2.678	2.693	0.54
CPU Stall Cycles	48	0	−100
Memory References	1 536	768	−50.0

Example: Energy savings (~35%) with SIMD

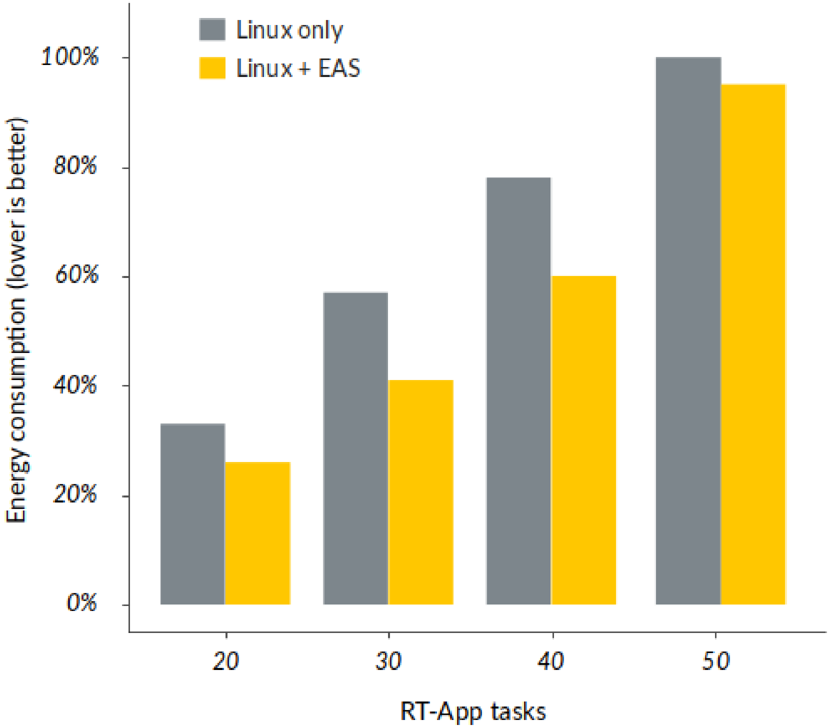
Strategy #6: Energy aware scheduling

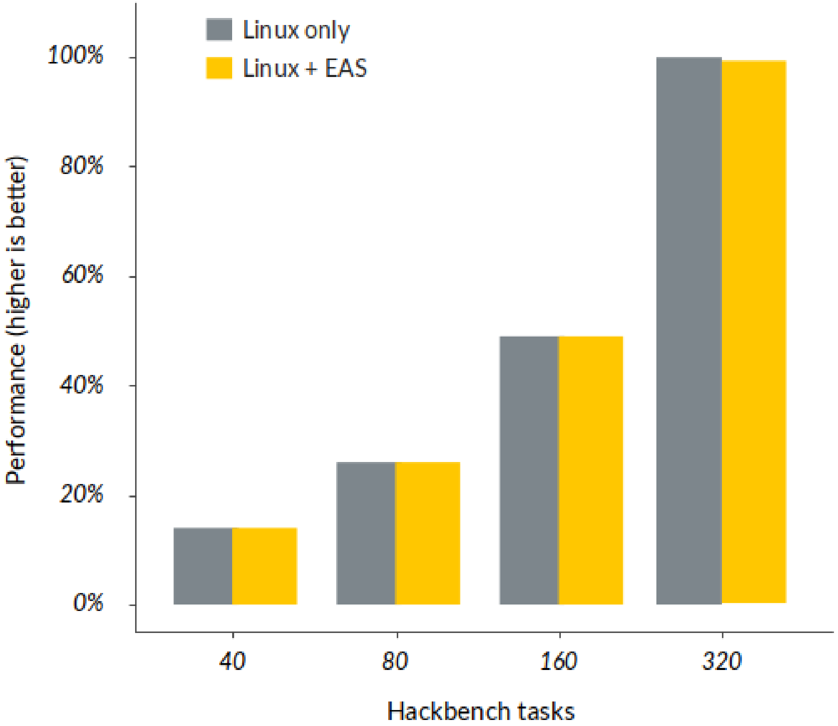
- Performed by the OS Scheduler
- Scheduler aware of system topology
- Primary methods:
  - Task placement
  - Task migration
- Used in embedded real time systems

- **Example** 1: Power aware routing in reconfigurable architectures
- **Example** 2: Linux Energy Aware Scheduler (EAS)

Strategy #6: contd..

- Energy Aware Scheduler (EAS) in Linux


Linux only
Linux + EAS
Energy consumption (lower is better)
RT-App tasks


Linux only
Linux + EAS
Performance (higher is better)
Hackbench tasks