EC535 Introduction to Embedded Systems

I/O-oriented programming in Linux

- I/O service is not always available
  - Bandwidth limitation
    - Data not yet ready when one wants to read
    - Old data not yet transmitted when one wants to write again
  - Asynchronous nature of I/O
    - Data arrival time is unpredictable

Rules about sleeping

- Never sleep while running in atomic context.
  - What can go wrong?
    - Process holds a spinlock, other locks
    - Process disabled interrupts
- When process wakes up, it will not know what happened on the CPU while it was sleeping.
  - Make no assumptions based on earlier CPU states, check again.
- Make sure to verify who is going the wake the process up under what condition before implementing sleep.

Quick Poll

https://shorturl.at/uxSyA

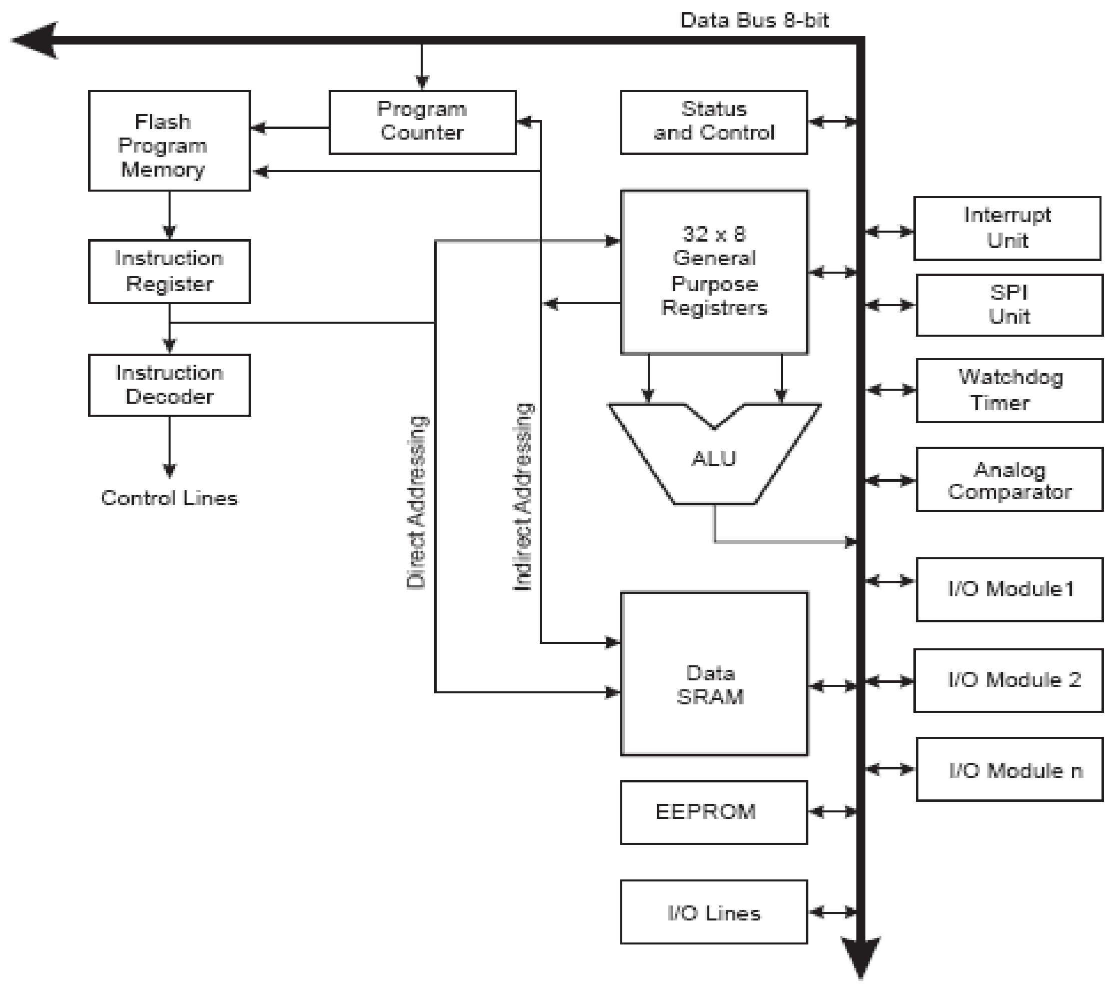
Team: Pairs for Assignments, HW, Labs

Energy Efficiency

- Energy-efficient computing is needed by:
  - portable systems
    - increase battery lifetime
    - optimize thermal design -> form factor
  - non-portable systems
    - minimize cost
    - environmental concerns
- Electronic system design
  - Hardware: processing, storage, communication
  - Software: operating systems, applications
- Electronic system utilization
  - Runtime control and management
  - e.g., Dynamic Power Management (sleep states) & Dynamic Voltage Scaling (lower CPU frequency/voltage)

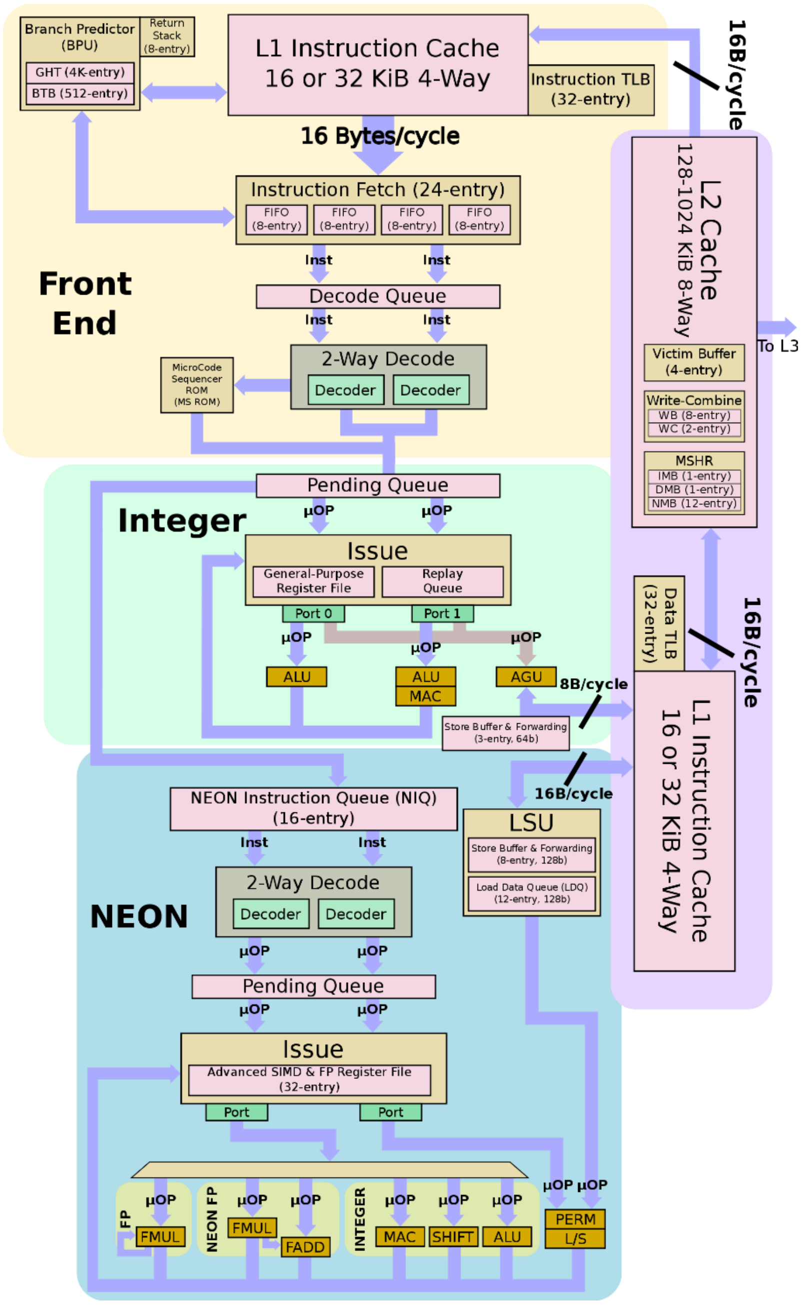
ATmega128L

- 4 or 8 MHz
- 8 bit
- 128KB Flash
- 4KB EEPROM
- 4KB SRAM
- 133 instructions
  - most single cycle
- 32 gen. regs
- 2 cycle multiplier
- 8 channel, 10 bit ADC; 1 kSamps/s max resolutio

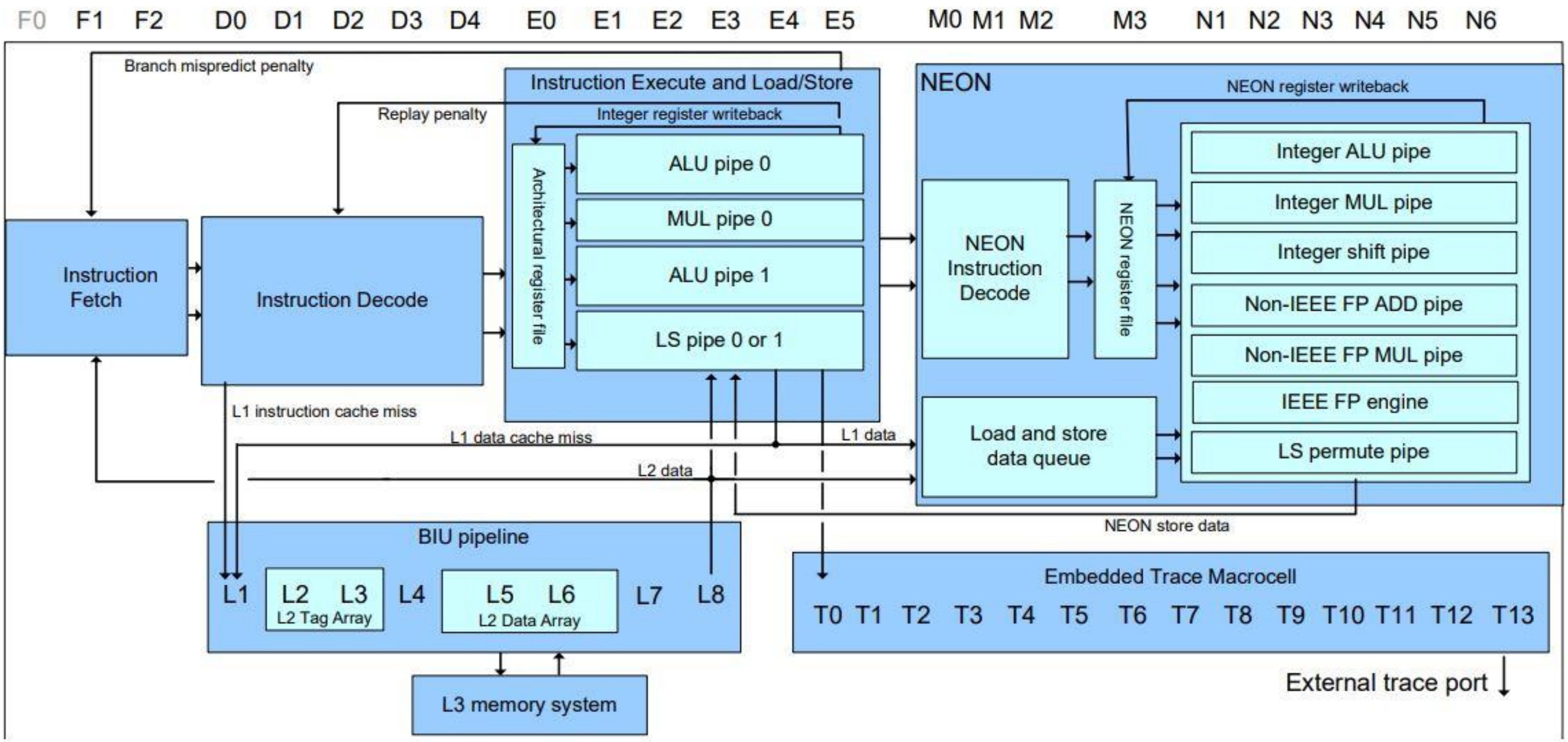


ARM Cortex-A8 Processor

- Pipeline: 13 stages
- 32-bit, ARMv7
- 16/32 KiB L1 I/D-Cache
- Support for various peripherals
- CPU freq: 600MHz-1.1GHz
- NEON SIMD instruction set extension
- Superscalar dual issue
- SoCs with Cortex-A8: Apple A4, Samsung Exynos 3110, TI OMAP3, many others



Cortex-A8 Pipeline



Comparing Power & Performance

- Cortex-A8
  - 0.43mW/MHz (e.g., <300mW for 600MHz)

- ATmega
  - Active: 4 MHz => 17 mW
  - Idle: 4 MHz => 12mW
  - Sleep: 45 uW

Portable Computers

- Lots of energy consumed in: display, hard disks, WLAN

Hard Disk

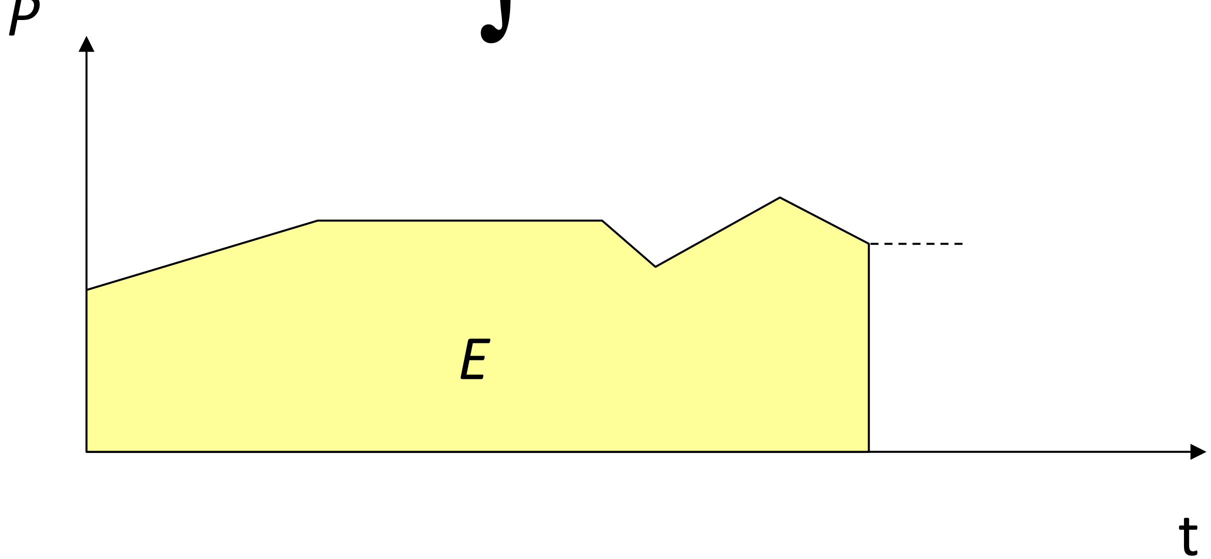
- Active power: 0.95 -2.5 W
- Idle power: 0.95 W
- Sleep: 0.13 W
- Sleep time: 0.67 s
- Wake-up time: 1.6 s

WLAN

- Transmission: 1.65 W
- Receiving: 1.4 W
- Doze: 0.045 W
- Down time: 62ms
- Wake-up time: 34 ms

Power (P) and Energy Relationship

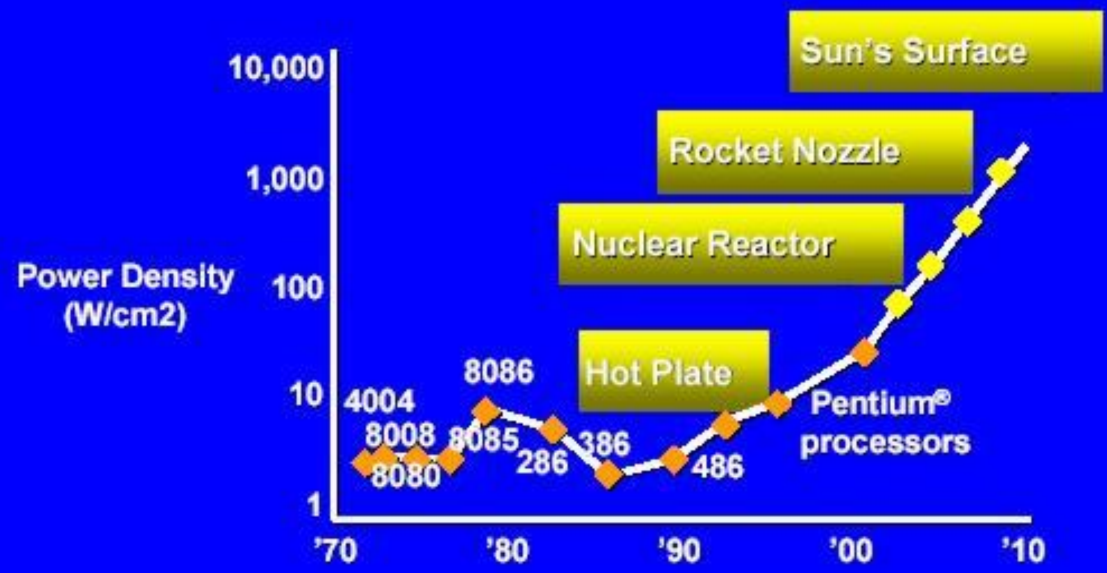
$$E = \int P\,dt$$



Low Power vs. Low Energy

- Minimizing the **power consumption** is important for
  - the design of the power supply
  - the design of voltage regulators
  - short term cooling
- Minimizing the **energy consumption** is important due to
  - restricted availability of energy (mobile systems)
    - limited battery capacities (only slowly improving)
    - high costs of energy
  - cooling
    - high costs
    - limited space
  - long lifetimes, low temperatures

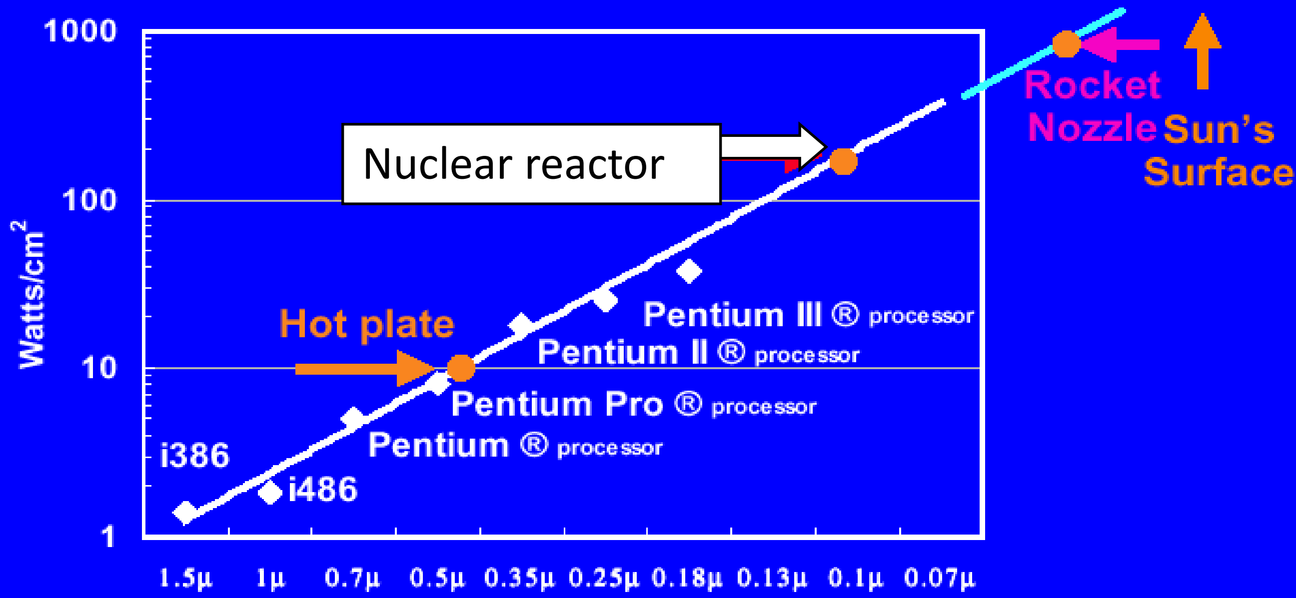
Power Density Extrapolation



Gelsinger's Slide from ISSCC 2001

intel.

Power density continues to get worse

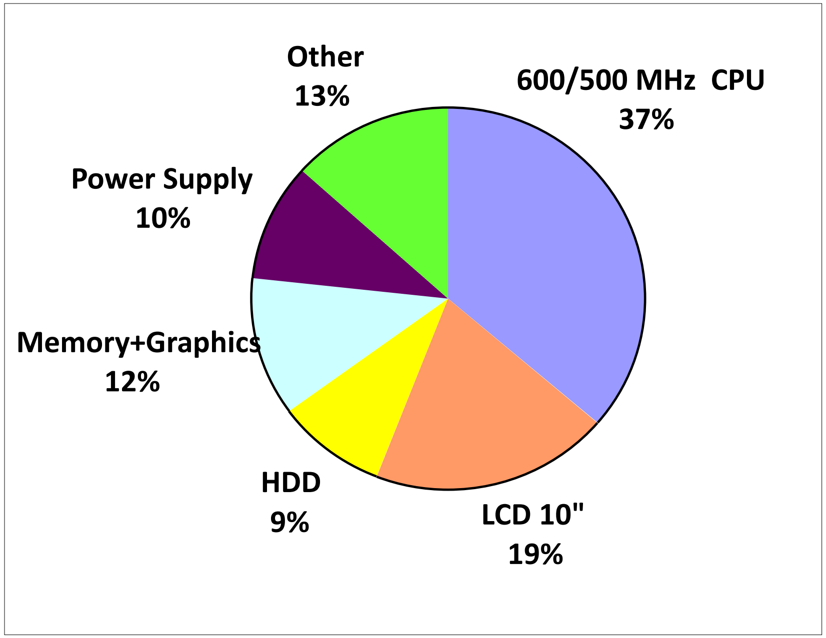


Surpassed hot-plate power density in 0.5µ

Not too long to reach nuclear reactor

Consider CPU & System Power

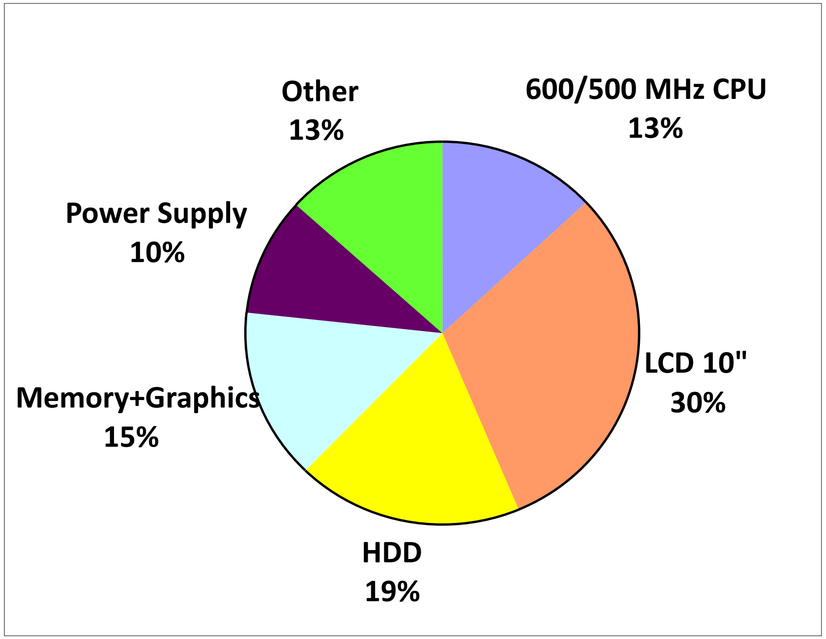
Mobile PC
Thermal Design (TDP) System Power



Note: Based on Actual Measurements

CPU Dominates Thermal Design Power

Mobile PC
Average System Power



Multiple Platform Components Comprise Average Power

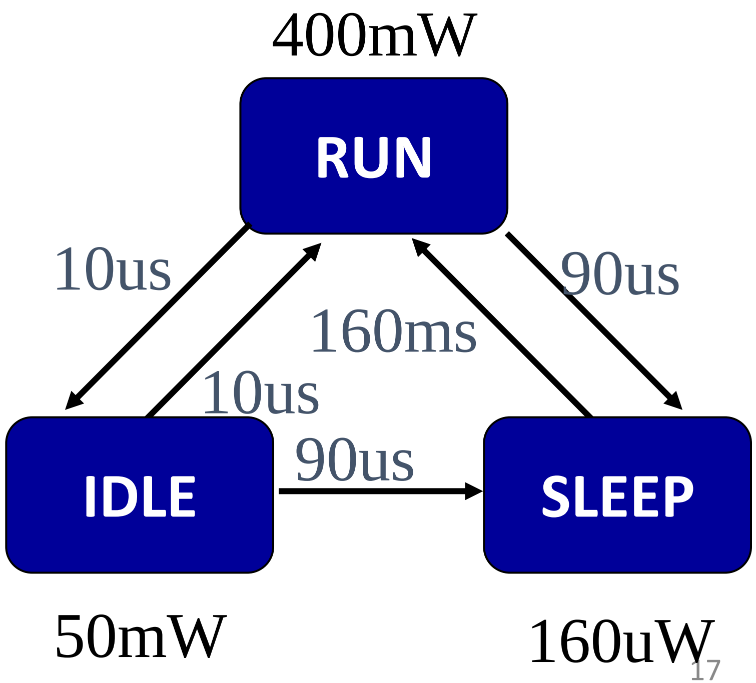
[Courtesy: N. Dutt; Source: V. Tiwari]

Power Manageable Components

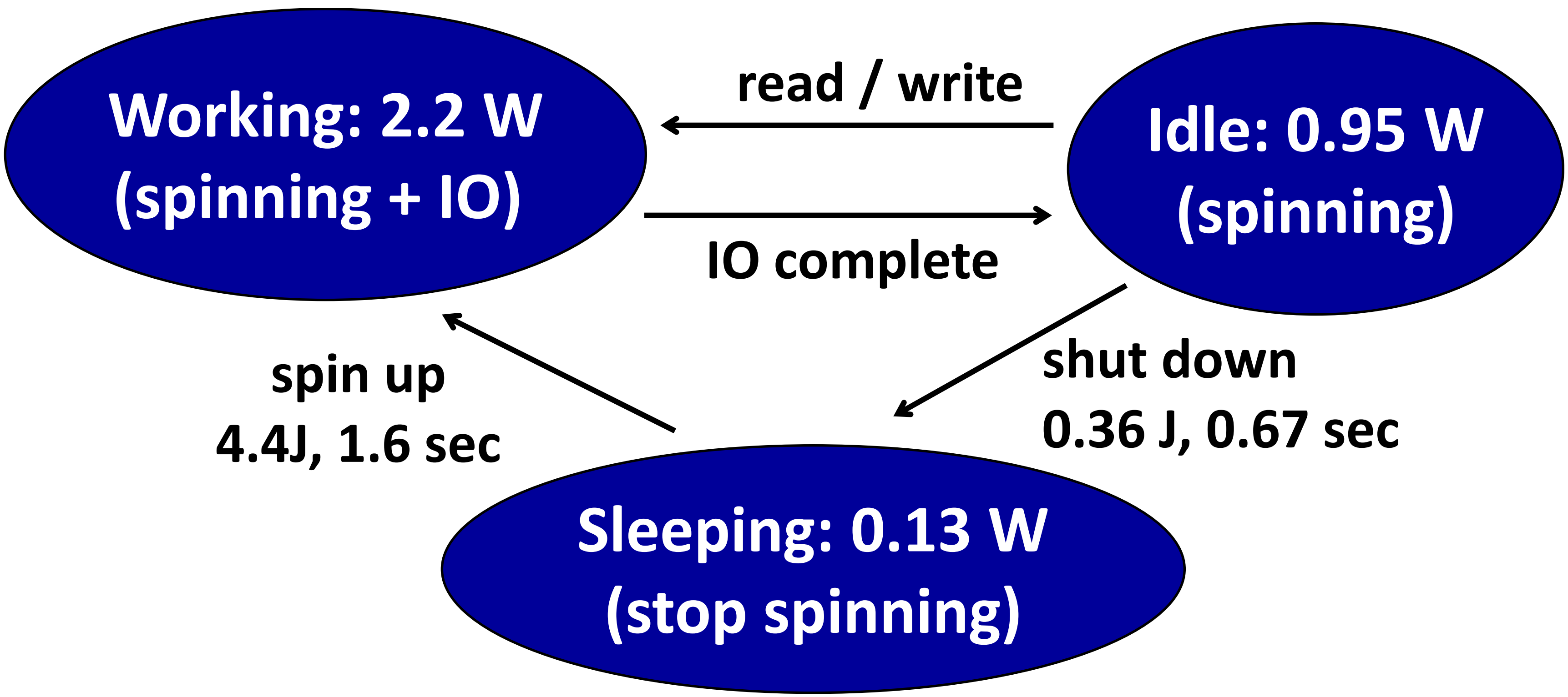
- Components with several internal states
  - Corresponding to power and service levels
- Abstracted as **power state machines**
  - State diagram with:
    - Power and service annotation on states
    - Power and delay annotation on edges

Example: SA-1100

- **RUN:** Operational
- **IDLE**: A SW routine may stop the CPU when not in use, while monitoring interrupts
- **SLEEP:** Shutdown of on-chip activity

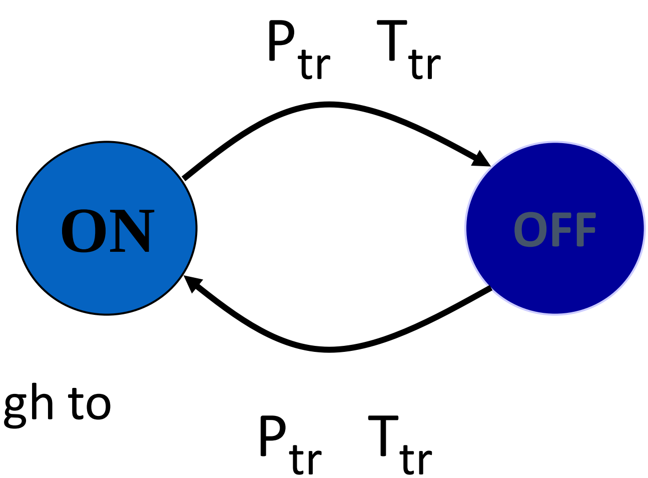


Example: Hard Disk Drive

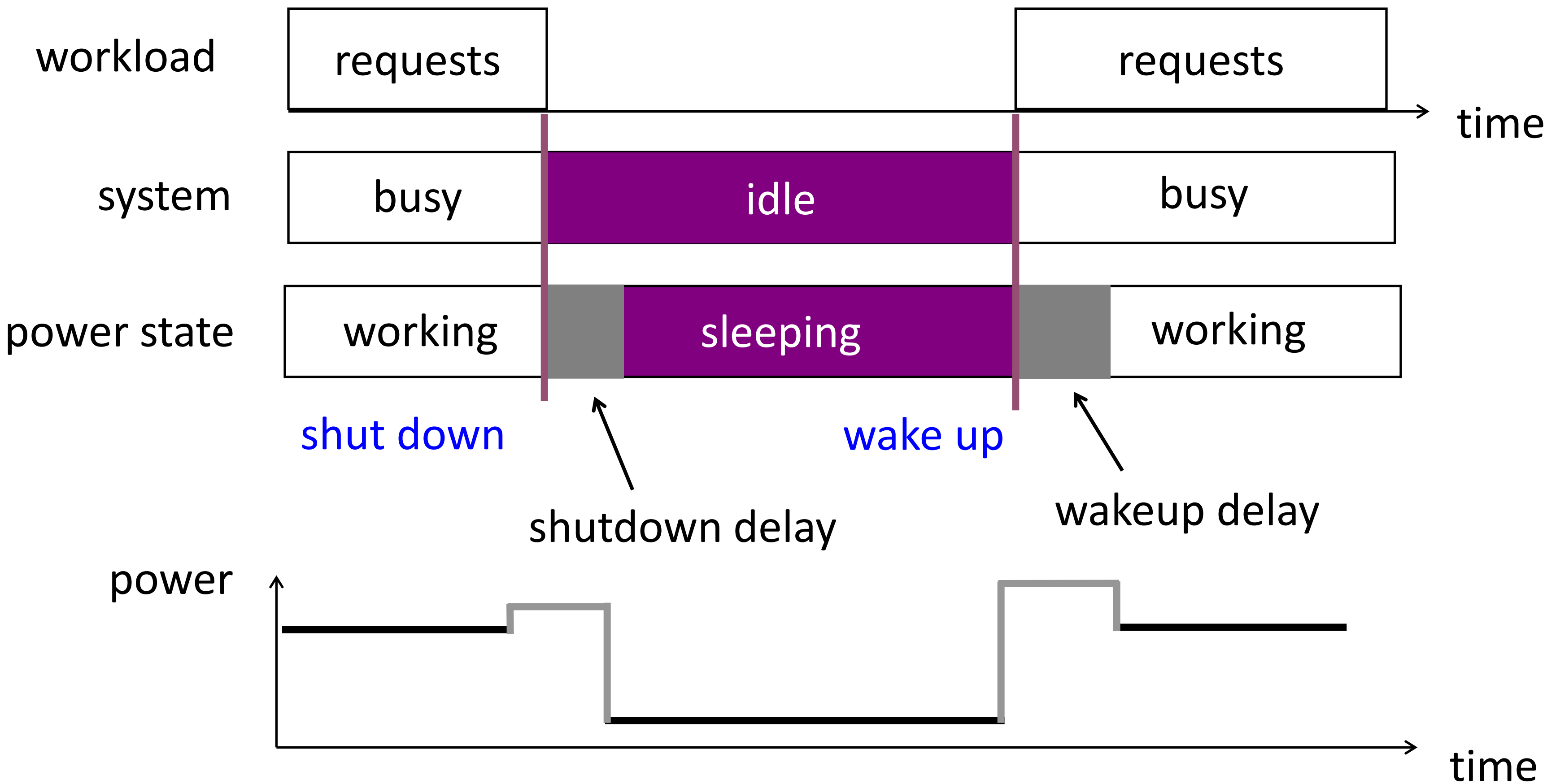


Dynamic Power Management

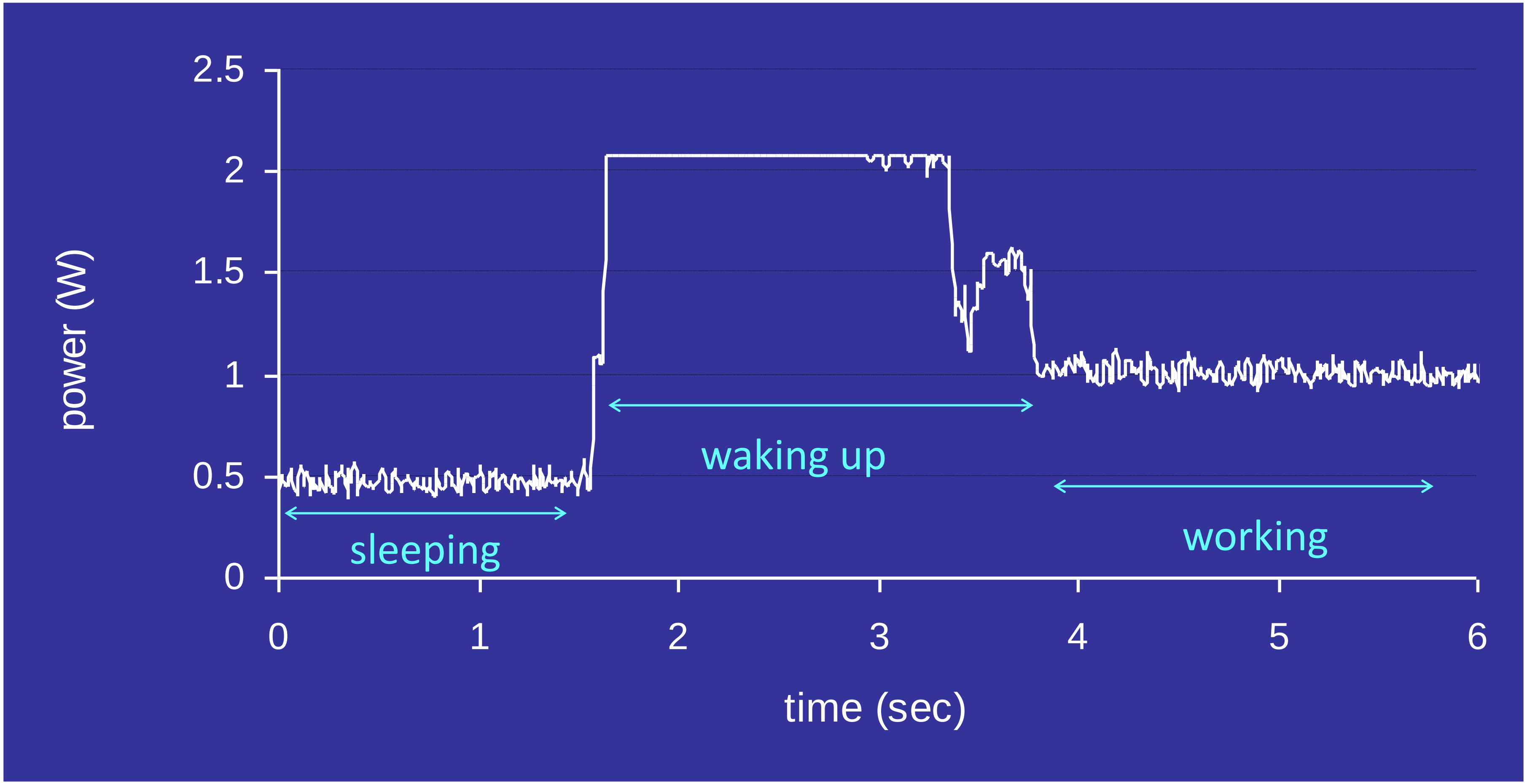
- State transition power (P_{tr}) and delay (T_{tr})
- If $T_{tr} = 0$, $P_{tr} = 0$ the policy is trivial
  - Stop a component when it is not needed
- If $T_{tr} \mathrel{!}= 0$ or $P_{tr} \mathrel{!}= 0$ (always…)
  - E.g., XScale 27x
    - 0.5 ms to sleep state
  - Shutdown only when idleness is long enough to neglect the cost
  - What if T and P fluctuate?



Workload and System Representation



Waking Up Hard Disk



Measurements done on a Fujitsu hard disk

Car Example

Imagine a car engine:
•If you turn it **off** for **5 seconds** but it takes **5 seconds' worth of fuel to restart**, you've **wasted** energy instead of saving any.

•If you leave it **off** for longer than 5 seconds, **then you start seeing energy savings**.

The system needs to be OFF long enough to **recover the transition energy loss**.

Car Example

Imagine a car engine:
•If you turn it **off** for **5 seconds** but it takes **5 seconds' worth of fuel to restart**, you've **wasted** energy instead of saving any.

•If you leave it **off** for longer than 5 seconds, **then you start seeing energy savings**.

The system needs to be OFF long enough to **recover the transition energy loss**.

-> System Break-Even Time

System Break-Even Time: T_{BE}

Minimum idle time for amortizing the cost of component shutdown

$$T_{BE} = T_{tr} + T_{tr}\,\frac{P_{tr} - P_{on}}{P_{on} - P_{off}}$$

Transition delay (T_{tr})

Transition power (P_{tr})

Sleep power (P_{off})

Decision-Making

If $T_{idle} < T_{BE}$

Staying On is better

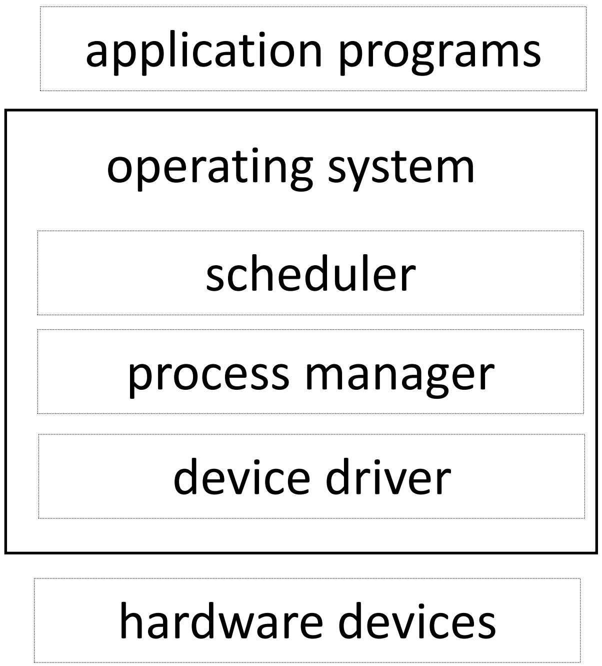
- frequent switching is inefficient

If $T_{idle} > T_{BE}$

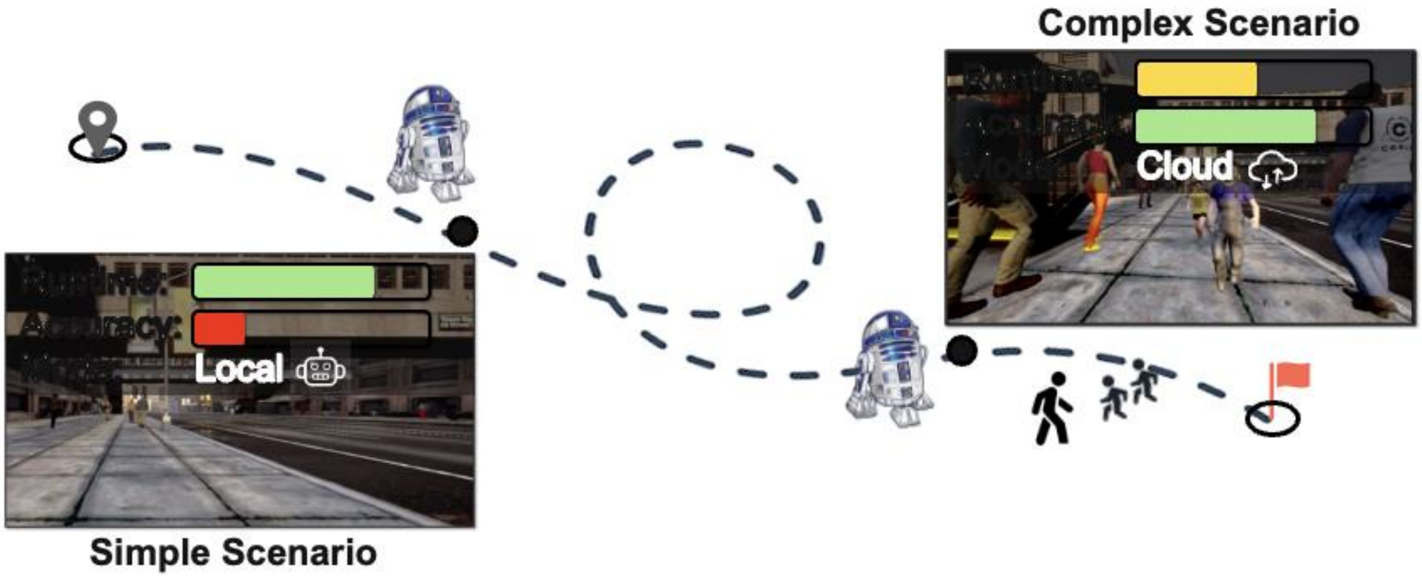
- system remains idle long enough to make up for the transition energy loss

DPM and Operating Systems

- Application
  - should not directly control hardware power
  - no power management in legacy programs
- Scheduler
  - selects processes and affects idle periods
- Process manager
  - knows multiple requesters
  - can estimate idle periods more accurately
- Driver
  - detects busy and idle periods
- Device
  - consumes power
  - should provide mechanism, not policy

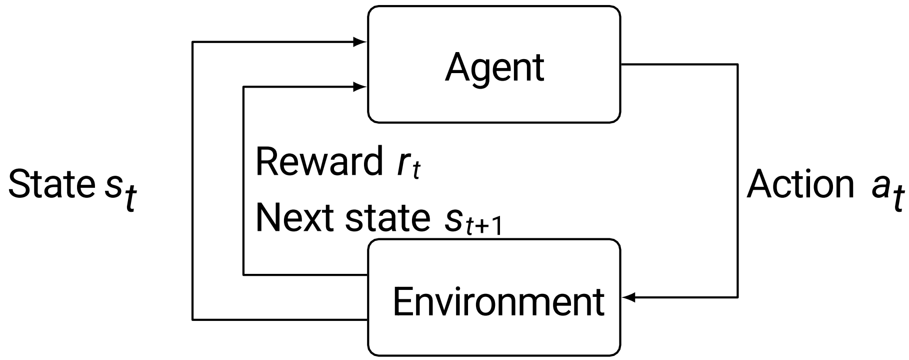


Schedule Should Depend on the Situation!



We present **UniLCD**, a novel local-cloud hybrid framework that dynamically routes computation between low-power local models and powerful cloud resources via reinforcement learning based on scenario complexity.

Sequential Decision Process



Agent oberserves environment state s_t at time t

Agent sends action a_t at time t to the environment

Environment returns the reward r_t and its new state s_{t+1} to the agent

Decision Process

Components:

- State: $s \in \mathcal{S}$ may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ provided by expert demonstrator
- State dynamics: $P(s_{i+1}|s_i, a_i)$ simulator, typically not known to policy
  Often deterministic: $s_{i+1} = T(s_i, a_i)$ deterministic mapping
- Rollout: Given s_0, sequentially execute $a_i = \pi_\theta(s_i)$ & sample $s_{i+1} \sim P(s_{i+1}|s_i, a_i)$ yields trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$
- Loss function: $\mathcal{L}(a^*, a)$ loss of action a given optimal action a^*

Decision Process

Decision Process (MDP) defined by tuple:

$$(S, A, R, P, \gamma)$$

- ► S : set of possible states
- ► A: set of possible actions
- ► R: distribution of reward given (state,action) pair
- ► P :distribution over next state given (state,action) pair
- ► γ: discount factor

Many reinforcement learning problems can be formalized as MDPs

Markov Decision Process

Markov property: Current state completely characterizes state of the world

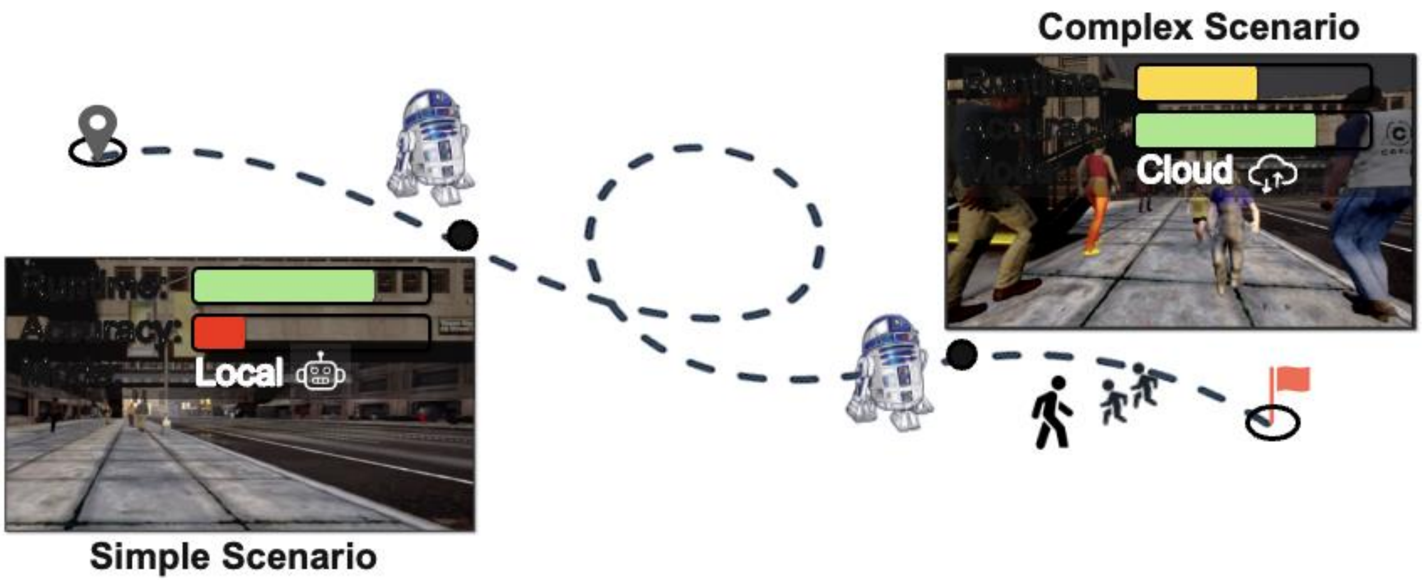
- ► A state s_t is *Markov* if and only if

$$P(s_{t+1} | s_t) = P(s_{t+1} | s_1, \ldots, s_t)$$

- ► "The future is independent of the past given the present"
- ► The state captures all relevant information from the history
- ► Once the state is known, the history may be thrown away
- ► i.e. the state is a sufficient statistic of the future

The picture can't be displayed.

Schedule Should Depend on the Situation!



We present **UniLCD**, a novel local-cloud hybrid framework that dynamically routes computation between low-power local models and powerful cloud resources via reinforcement learning based on scenario complexity.

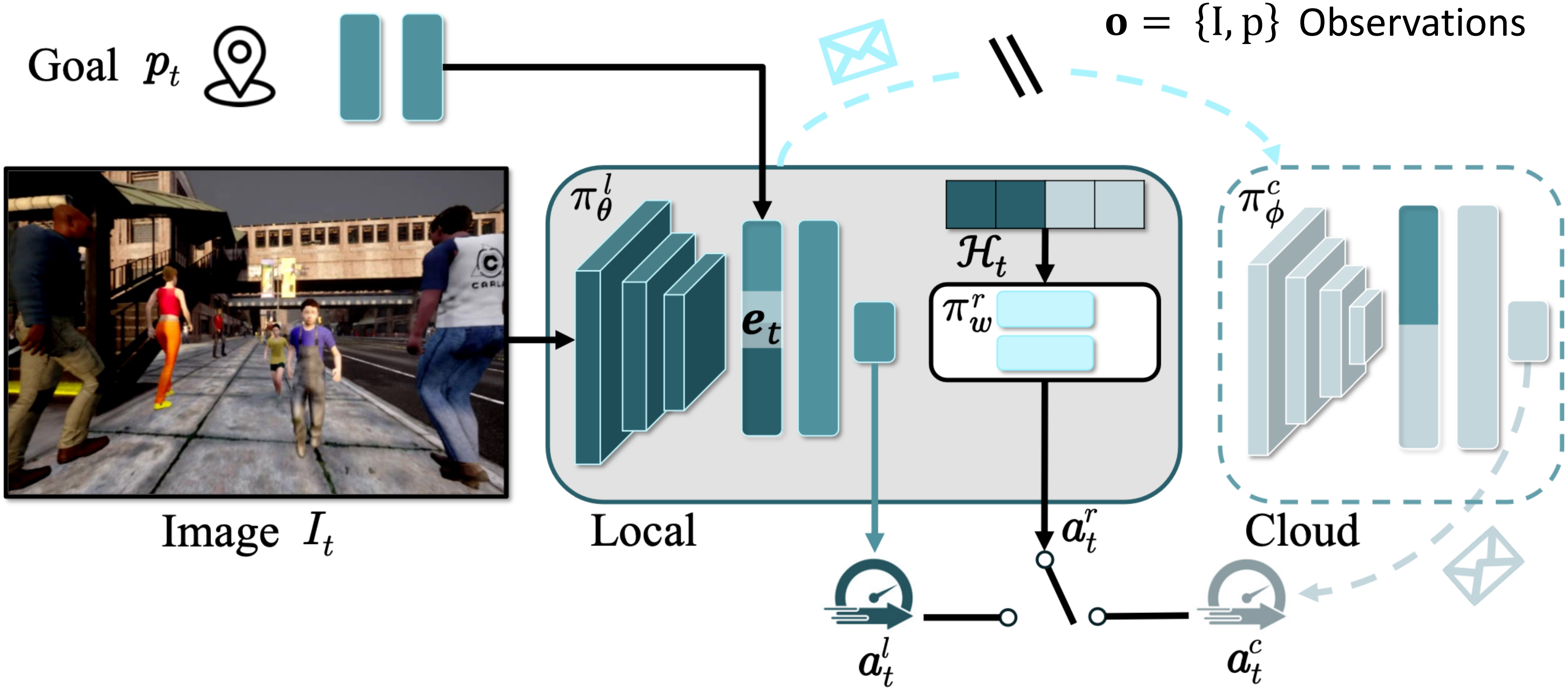
UniLCD framework

π_{φ}^{c} Cloud Policy

π_{θ}^{l} Local Policy

$\pi_{w}^{r}(\mathbf{o}|\,\pi_{\varphi}^{c}, \pi_{\theta}^{l})$ Routing Policy

$\mathbf{o} = \{\mathrm{I}, \mathrm{p}\}$ Observations



Multi-objective Reward Function

Task Reward: $r = (r_{geo}.r_{speed}.r_{energy}.r_{action})^{\alpha} - r_{collision}$

Geodesic Reward: $r_{geo} = (1 - \tanh(d_{geo}))$

Speed Reward: $r_{speed} = \frac{v}{m_v}$

Energy Disadvantage: $r_{energy} = 1 - \frac{e}{m_e}$

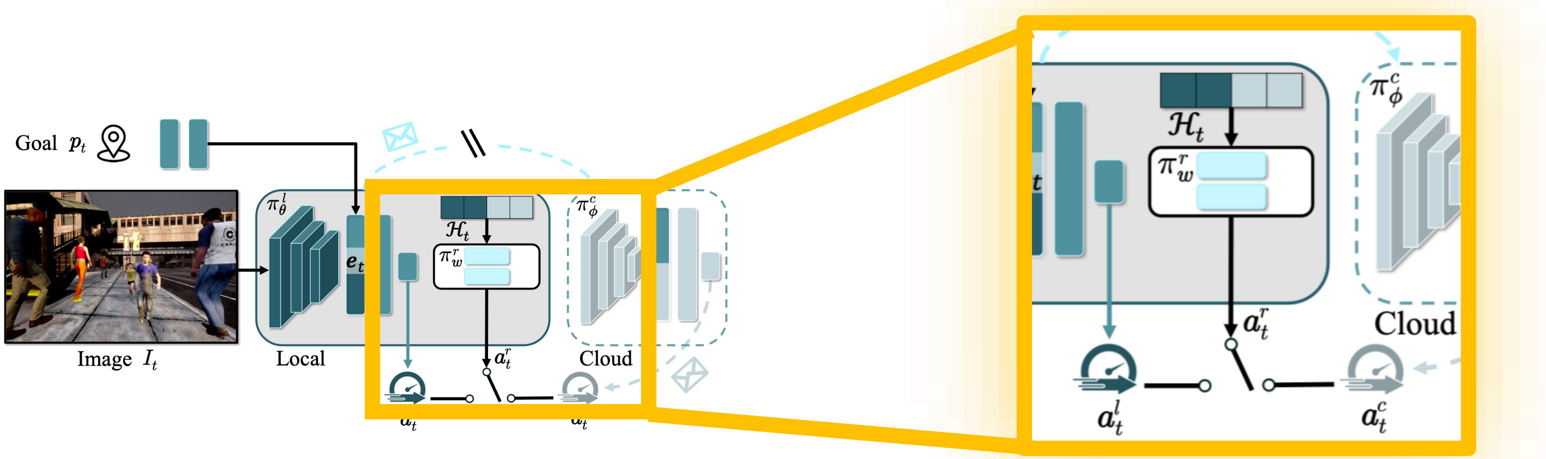
Extreme Action Clip: $r_{action} = \mathbb{I}\left(\left|r_{speed}\right| < \varepsilon\right).\mathbb{I}(\left|\frac{d}{d_m}\right| < \varepsilon)$

d_{geo} : Geodesic distance
m_v: Maximum speed
m_e: Maximum energy
d_m : Maximum possible rotation
ε : 0.97 (Threshold for clipping action)

$r_{collision}$ is a substantially high negative penalty to our robot every time it collides with a pedestrian.

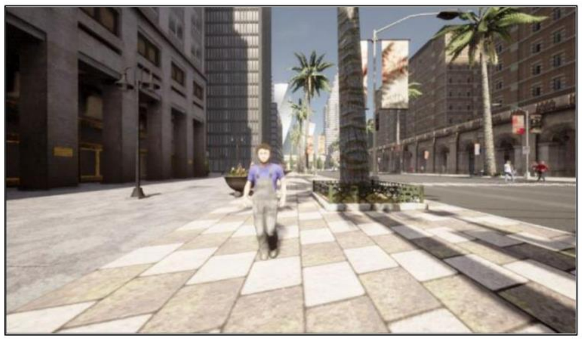
Routing Policy via Reinforcement Learning

Input: Image $\mathbf{I}$, next waypoint $\mathbf{p}$, local policy $\pi^l_{\boldsymbol{\theta}}$, cloud policy $\pi^c_{\boldsymbol{\phi}}$
Initialize: Number of iterations T, history $\mathcal{H}$, routing policy $\pi^r_{\boldsymbol{\omega}}$, reply buffer $\mathcal{S}$
Collect on policy samples:
for t $= 1$ **to** T **do**
    Obtain local action $\mathbf{a}^l_t$ and embeddings $\mathbf{e}_t$ using local policy $\pi^l_{\boldsymbol{\theta}}(\mathbf{I}_t, \mathbf{p}_t)$
    Append $(\mathbf{a}^l_t, 0)$ to history $\mathcal{H}_t$
    if $\pi^r_{\boldsymbol{\omega}_t}(\mathcal{H}_t, \mathbf{e}_t) = 0$ **then** $\mathbf{a}_t = \mathbf{a}^l_t$
    else
        Send $\mathbf{e}_t$ to cloud, $\mathbf{a}_t = \pi^c_{\boldsymbol{\phi}}(\mathbf{I}_t, \mathbf{p}_t)$
        Update last value of $\mathcal{H}_t$ to $(\mathbf{a}_t, 1)$
    end if
    Compute instant reward using Eq. (2)
    if Arrived destination **then** break
    end if
    Update replay buffer $\mathcal{S} = \mathcal{S} \cup \{\mathbf{I}_t, \mathbf{p}_t, \mathcal{H}_t, r_t\}$
    Update routing policy parameters with PPO
end for

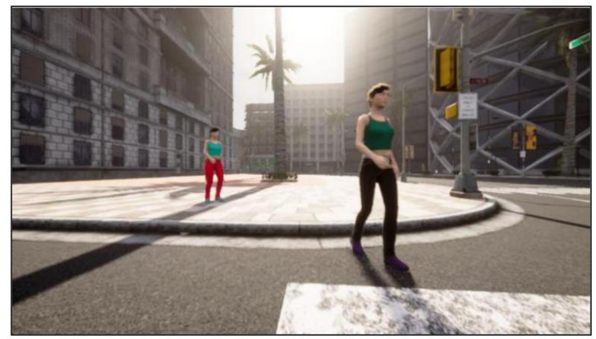


Simulation Environment

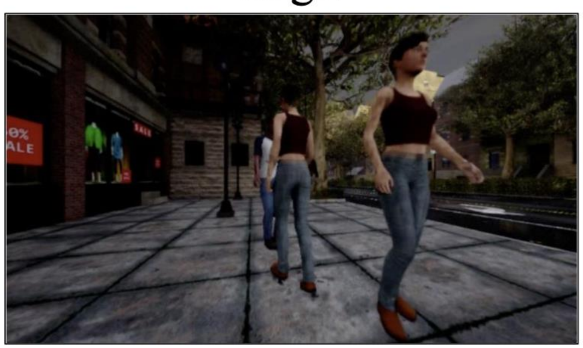
Low



Medium



High



Crowd

How to Evaluate Performance? Navigation Score

Navigation Score$(\mathrm{NS}) = RC.\, P_I^{IC}.\, P_{RD}$

Infraction Penalty(P_I) = 0.5

Route Deviation Penalty $(P_{RD}) = \begin{cases} 0.8, if\ RD > \varepsilon_{RD} \\ 1.0, otherwise \end{cases}$

Collision Count Per Meter IC

How to Evaluate Performance? Navigation Score

Navigation Score$(\mathrm{NS}) = RC.\, P_I^{IC}.\, P_{RD}$

Infraction Penalty(P_I) = 0.5

Route Deviation Penalty $(P_{RD}) = \begin{cases} 0.8, if\ RD > \varepsilon_{RD} \\ 1.0, otherwise \end{cases}$

Collision Count Per Meter IC

Ecological Navigation Score

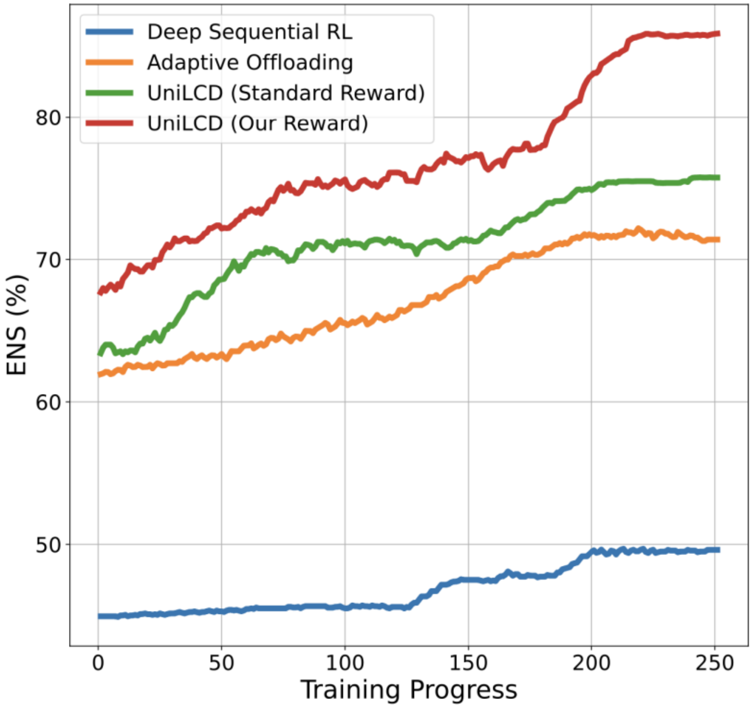
Ecological Navigation Score$(\mathrm{ENS}) = \ P_E \cdot \mathrm{NS}$

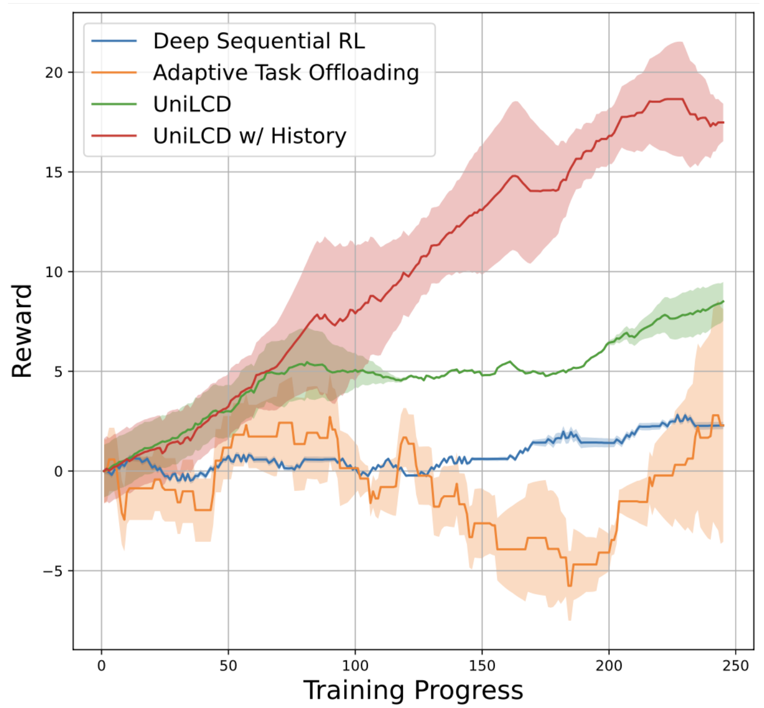
Penalty term (P_E) = 1 - $\frac{\mathrm{Energy}}{N_E}$

Normalization Factor($N_E) = (E_{local} + E_{cloud}).\,(N_{local} + N_{cloud})$

Total energy consumption (Energy) = $E_{local} \cdot N_{local} + \ E_{cloud} \cdot N_{cloud})$

Performance over training progression





Comparing UniLCD with other baselines

Method	ENS↑	NS↑	SR↑	RC↑	Infract.↓	Energy↓	FPS↑
† Cloud-Only [84]	0.00	96.47	93.33	98.50	0.03	36.49	7.11
Local-Only [82]	63.43	67.33	0.00	75.23	0.16	4.33	65.40
Baseline Methods:							
Compressive Offloading [108]	13.98	80.16	0.00	80.16	0.00	90.66	1.82
† Selective Query [39]	24.14	61.28	0.00	82.68	0.11	45.35	18.14
† Adaptive Offloading [95]	37.42	40.37	70.00	94.05	1.22	4.80	30.14
Neurosurgeon [40]	39.85	63.10	0.00	80.54	0.03	28.31	12.53
SPINN [52]	36.31	72.75	60.00	92.73	0.35	18.94	20.37
Deep Sequential RL [97]	58.84	61.83	0.00	79.36	0.36	3.77	77.94
UniLCD Module Ablations:							
† Standard Reward	48.35	54.99	0.00	75.23	0.13	3.57	50.20
† Standard Reward w/ History	50.04	57.21	10.00	77.71	0.12	8.38	49.07
† Our Reward (Eq. (2))	48.30	79.90	56.66	91.15	0.19	21.72	16.05
† Our Reward w/ History	71.70	87.71	83.33	94.66	0.11	7.83	33.98
Our Reward (Eq. (2))	57.20	87.39	60.00	91.10	0.06	6.60	**12.49**
Our Reward w/ History	**85.97**	**94.58**	**93.33**	**95.90**	**0.02**	**2.90**	26.49

Local policy backbone ablations

Local Model Size	Params	ENS↑	NS↑	SR↑	RC↑	Infract.↓	Energy↓	FPS↑
UniLCD:								
† Tiny	1.37	12.80	93.29	90.00	97.93	0.07	34.01	7.35
† Small	2.54	48.30	79.90	56.66	91.15	0.19	21.72	16.05
† Medium	3.50	50.92	87.87	80.00	95.50	0.12	21.19	15.10
UniLCD w/ History:								
† Tiny	1.37	0.00	91.27	**93.33**	**98.50**	0.11	36.52	6.42
† Small	2.54	71.70	87.71	83.33	94.66	0.11	7.83	33.98
† Medium	3.50	73.46	83.86	90.00	96.22	0.15	5.22	11.53
UniLCD:								
Stage 1	0.53	57.20	87.39	60.00	91.10	0.06	6.60	12.49
Stage 2	0.95	74.12	81.54	93.33	91.10	0.16	1.80	**65.40**
UniLCD w/ History:								
Stage 1	0.53	85.97	94.58	**93.33**	95.99	**0.02**	2.90	26.49
Stage 2	0.95	**86.78**	**95.47**	**93.33**	**98.15**	0.04	**1.77**	36.50

Reward component ablations

Reward	ENS↑	NS↑	SR↑	RC↑	Infract.↓	Energy↓	FPS↑
All Terms	**85.97**	**94.58**	**93.33**	**95.90**	**0.02**	**2.90**	**26.49**
w/o r_{geo}	67.04	74.65	0.00	76.22	0.03	7.42	58.82
w/o r_{speed}	66.05	72.66	0.00	77.34	0.09	6.61	65.40
w/o r_{energy}	0.00	93.53	90.00	95.50	0.03	43.75	6.42

Qualitative Results

System Break-Even Time: T_{BE}

Minimum idle time for amortizing the cost of component shutdown

$$T_{BE} = T_{tr} + T_{tr} \frac{P_{tr} - P_{on}}{P_{on} - P_{off}}$$

Transition delay (T_{tr})

Transition power (P_{tr})

Sleep power (P_{off})

Example: Calculate Break-even Time

- Processor consumes:
  - 10mW when idle,
  - 1mW while sleeping,
  - 100mW while transitioning into/out of sleep state.
- Transition time:
  - 100ms into and out of sleep state (total)
  - No transition time into/out of idle state
- What is the “breakeven” time?