Recap from last class

- Types of hardware power consumption
  - Startup
  - Static
  - Dynamic
- Hardware support on power management
  - Disabling function units – clock gating
  - Power shutdown
  - Voltage and frequency scaling
- Dynamic power management
  - Power state machine

ECE 1175
Embedded System Design

Power Management - II

Wei Gao

Outline

- Hardware support
- Power management policy
- Power manager
- Holistic approach

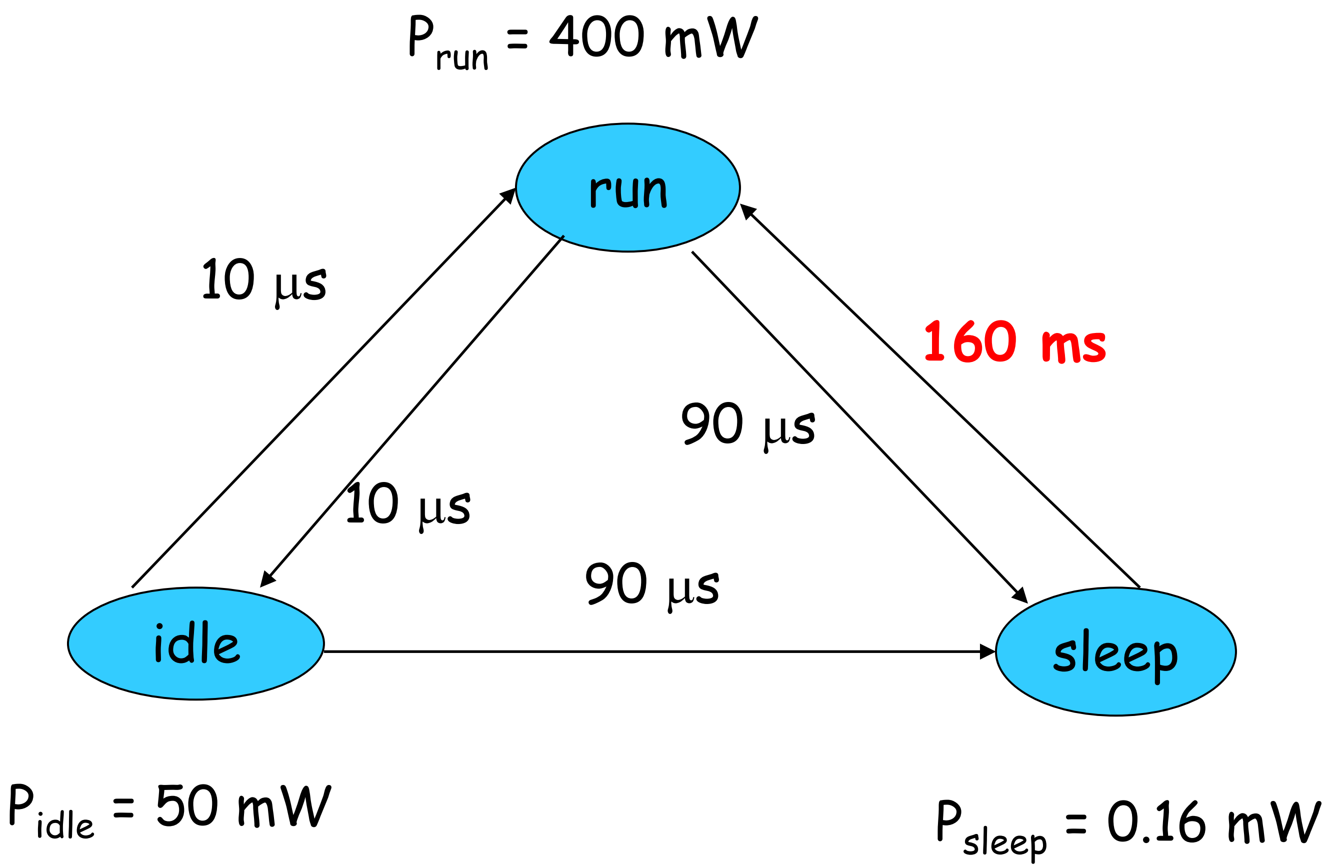
PSM Example: SA-1100

- SA-1100 is a StrongARM processor from Intel
  - Designed to provide sophisticated power management capabilities controlled by the on-chip power manager
- Three power modes:
  - Run: normal operation.
  - Idle: stops CPU clock, with I/O logic still powered.
  - Sleep: shuts off most of chip activity

SA-1100 SLEEP

- RUN → SLEEP
  - (30 μs) Flush to memorize CPU states (registers)
  - (30 μs) Reset processor state and wakeup event
  - (30 μs) Shut down clock
- SLEEP → RUN
  - (**10 ms**) Ramp up power supply
  - (**150 ms**) Stabilize clock
  - (negligible) CPU boot
- Overhead of sleep to run is much larger

SA-1100 Power State Machine



Baseline: Greedy Policy

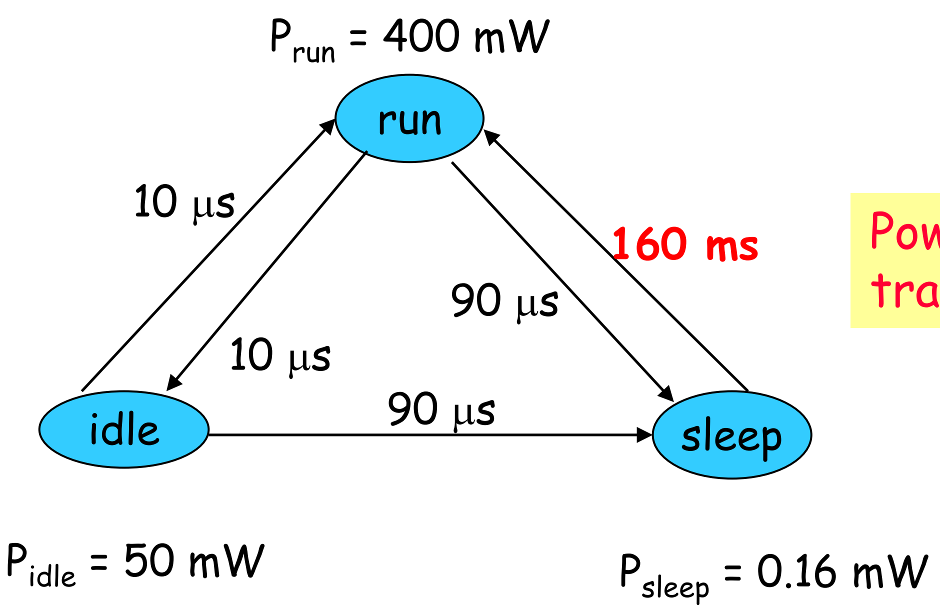
- Immediately goes to sleep when system becomes idle
- Works when transition time is negligible
  - Ex. between IDLE and RUN in SA-1100
- Doesn’t work when transition time is long!
  - Ex. between SLEEP and RUN/IDLE in SA-1100
  - Need better solutions!

Break-Even Time T_{BE}

- Definition
  - Minimum idle time required to compensate the cost of entering an inactive state
    - Cost: transition time and extra power
- Enter an inactive state is beneficial only if idle time is longer than the break-even time
  - Break-even time can be viewed as transition overhead
  - Ex. going to Hawaii is worthwhile only if vacation is long enough
- Assumptions
  - Cannot delay workload to extend idle time
  - An ideal power manager (PM) knows how long the idle time is going to be

Break-Even Time T_{BE}

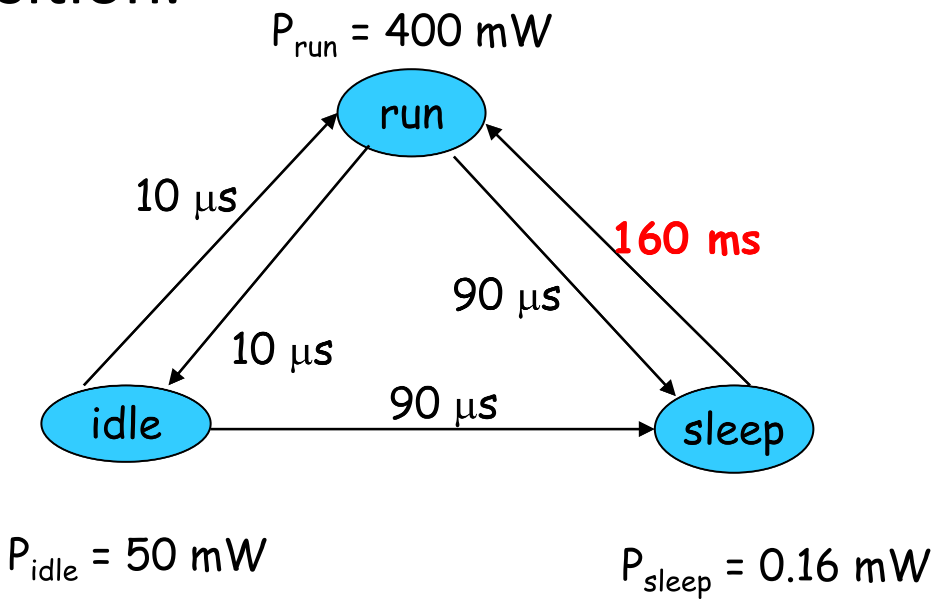
- T_{BE} of an inactive state is the total time for entering and leaving the state
  - Assumption: transition doesn't cause extra power consumption
- $T_{BE} = T_{TR} = T_{On,Off} + T_{Off,On}$
  - Ex. T_{BE} = 160 ms + 90 μs for SLEEP in SA-1100



Power consumption during transition ≈ P_{run}

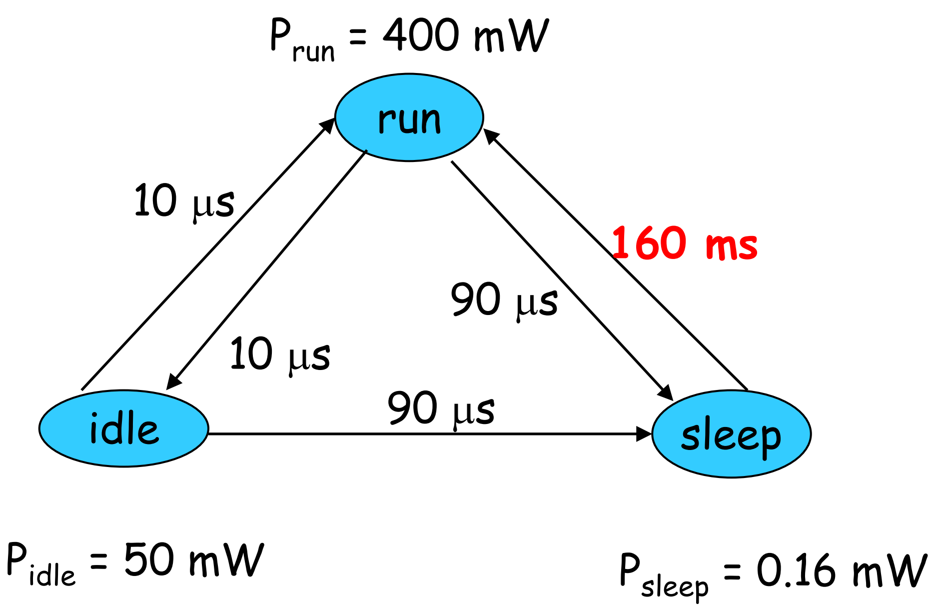
Break-Even Time When $P_{TR} > P_{On}$

- P_{TR}: Power consumption during transition
- P_{On}: Power consumption when active
- T_{BE} must include additional inactive time to compensate extra power consumption during transition.



Break-Even Time When $P_{TR} > P_{On}$

- $T_{BE} = T_{TR} + T_{TR}(P_{TR} - P_{On})/(P_{On} - P_{Off})$
  - $T_{TR}(P_{TR} - P_{On})$: extra energy consumed for transition
  - $/(P_{on}-P_{off})$: the idle time needed to compensate this energy consumption back
- It is easier to save power with a shorter T_{BE}
  - Shorter T_{TR}
    - IDLE in SA-1100
  - Higher difference between $P_{On} - P_{Off}$
    - SLEEP in SA-1100
  - Lower P_{TR}



Energy Saving Calculation

- Given an idle period $T_{idle} > T_{BE}$
  - $E_S(T_{idle}) = (T_{idle} - T_{TR})(P_{On} - P_{OFF}) + T_{TR}(P_{On} – P_{TR})$
    - $P_{On} > P_{TR}$: total = idle saving + transition saving
    - $P_{On} < P_{TR}$: total = idle saving - transition cost
- Achievable power saving depends on workload!
  - Distribution of idle periods

Predictive Techniques

- Don't know how long the idle time will be?
- Predict whether idle time $T_{idle} > T_{BE}$ based on history
  - Ex: someone takes break once an hour?
  - Predicted event: $p = \{T_{idle} > T_{BE}\}$
  - Observed event: o
    - Triggers state transition: RUN -> IDLE/SLEEP

Metrics of Prediction Quality

- **Safety**: conditional probability Prob(p|o)
  - If an observed event happens, what's the probability of $T_{idle} > T_{BE}$?
  - Ideally, safety = 1.
- **Efficiency**: Prob(o|p)
  - If $T_{idle} > T_{BE}$, what's the probability of successfully predicting it in advance (e.g., o happens)?
- Overprediction
  - State transition too much
  - High performance penalty → poor safety
- Underprediction
  - State transition not enough
  - Wastes energy → poor efficiency

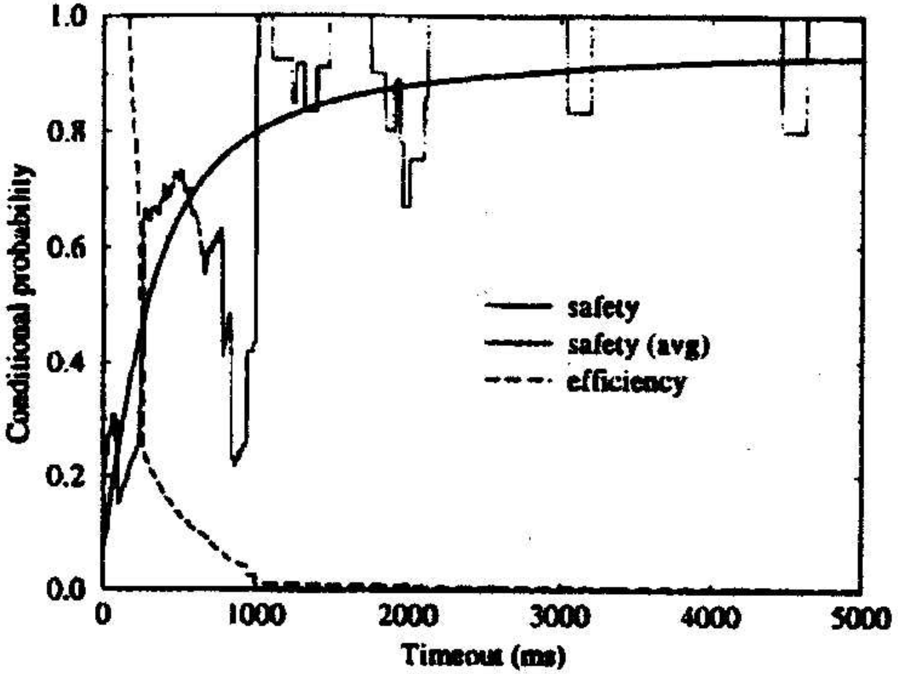
Fixed Timeout Policy

- Enter inactive state when the system has been idle for T_{TO} sec.
  - o: $T_{idle} > T_{TO}$
- Wake up in response to activity
- Assumption: If a system has been idle for T_{TO} sec, it is likely that it will continue to be idle for $T_{idle}-T_{TO} > T_{BE}$.

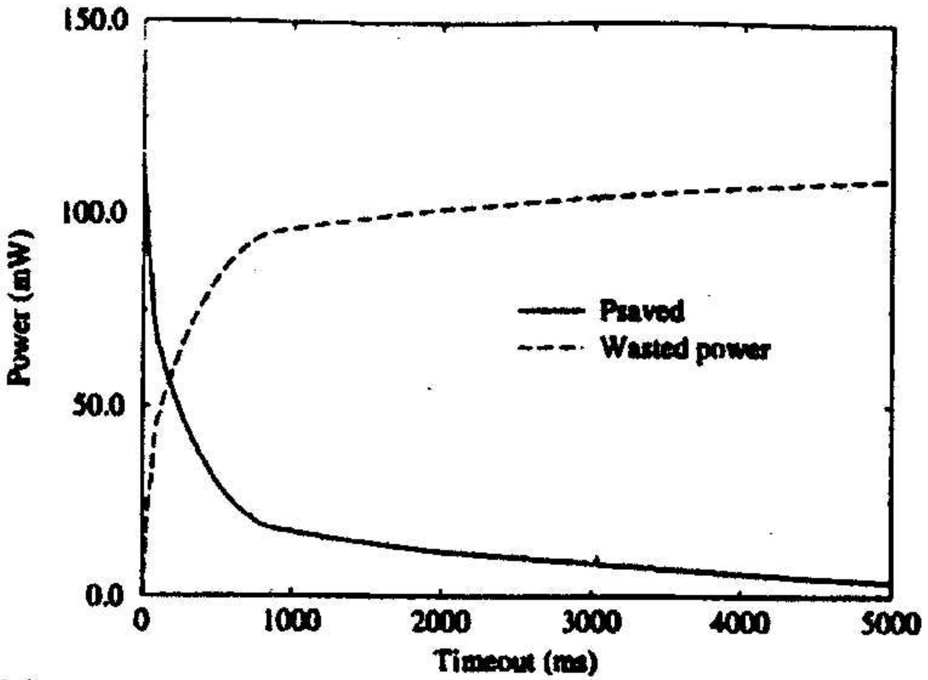
Selecting Best Timeout T_{TO}

- Increasing T_{TO} improves safety, but reduces efficiency at the same time
  - Safety: $Pr(T_{idle}>T_{TO}+T_{BE} \mid T_{idle} > T_{TO})$
- Highly workload dependent
- Karlin's result: $T_{TO} = T_{BE}$
  - → Energy consumption is at worst twice the energy consumed by an ideal policy

Impact of Timeout Threshold



a) Safety and efficiency as functions of timeout for the *games* workload (break-even time is 160ms)



b) Power saving as a function of timeout

Critiques on Fixed Timeout

- ✓ Simple!
- How to set timeout threshold?
  - Tradeoff between safety and efficiency
  - Works best when workload trace is available
- ✗ Always waste energy before reaching the timeout threshold
  - Solution: Predictive shutdown
- ✗ Always incur performance penalty for wake up
  - Solution: Predictive wakeup

Possible Improvement

- Predictive shutdown: shutdown *immediately* when an idle period starts
  - Avoid wasting energy before reaching the timeout threshold
  - More efficient, less safe
- Predictive wakeup: wake up before the *predicted* idle time expires, even if no new activity has occurred.
  - Avoid performance penalty for wake up
  - Less efficient, safer

More

- Stochastic control: based on statistical models (e.g., Markov chain)
- Read this paper
  - L. Benini, A. Bogliolo and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," IEEE Transactions on VLSI, pp. 299-316, June 2000.