



NICTA

Unifying DVFS and Offlining in Mobile Multicores

Aaron Carroll
Gernot Heiser

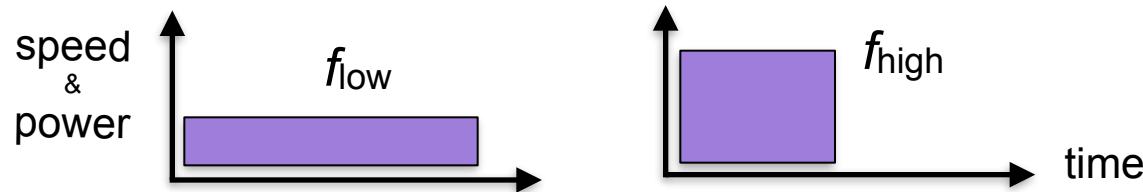


NICTA Funding and Supporting Members and Partners

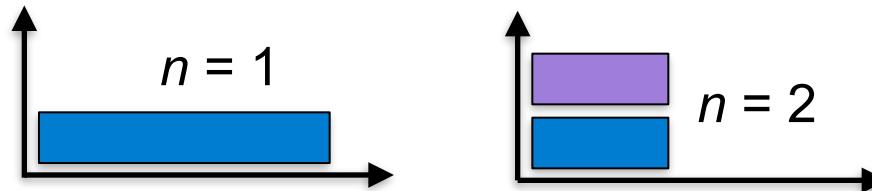


Power control knobs

DVFS:

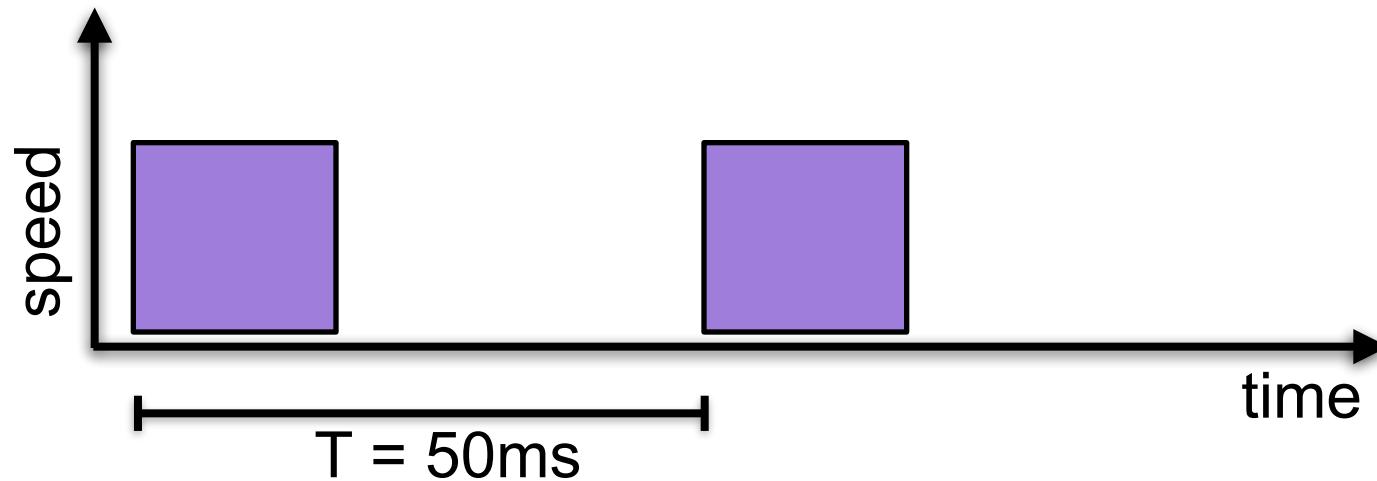


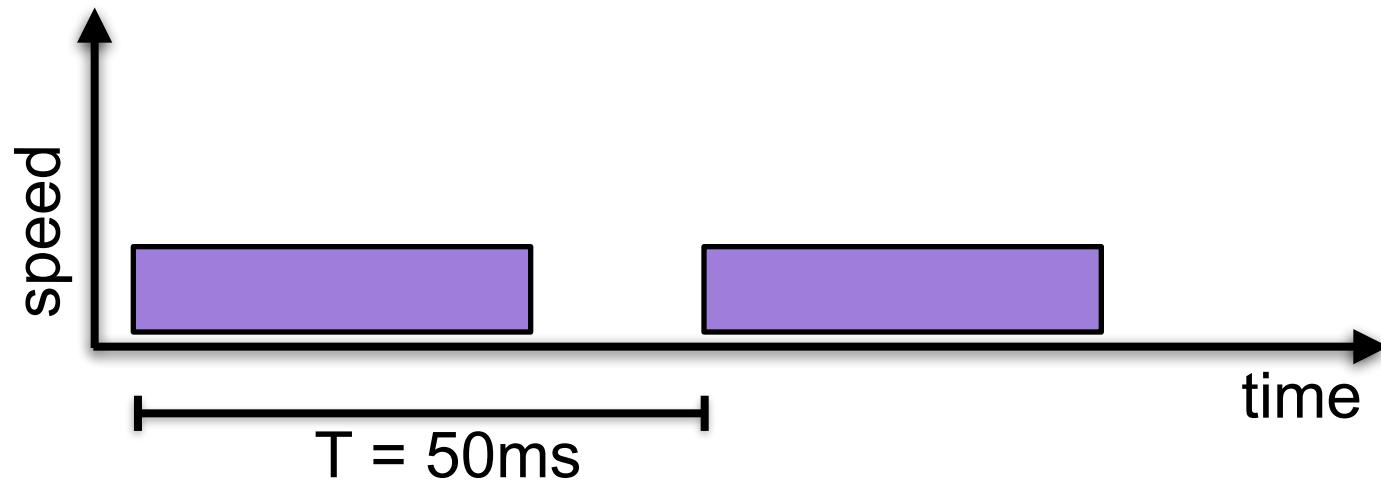
#online cores:

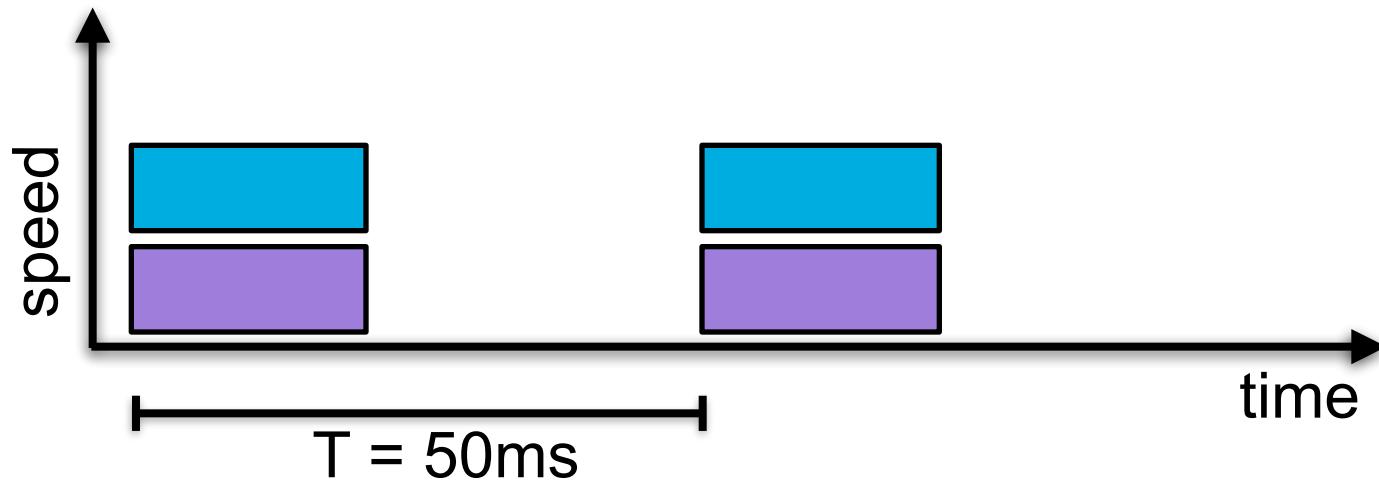


Problem

- trade-off between the two mechanisms
- optimal **operating point**
 - frequency and #online cores
- **idle** energy management
 - under-utilised CPU
 - reduce energy
 - performance unaffected (ideally...)





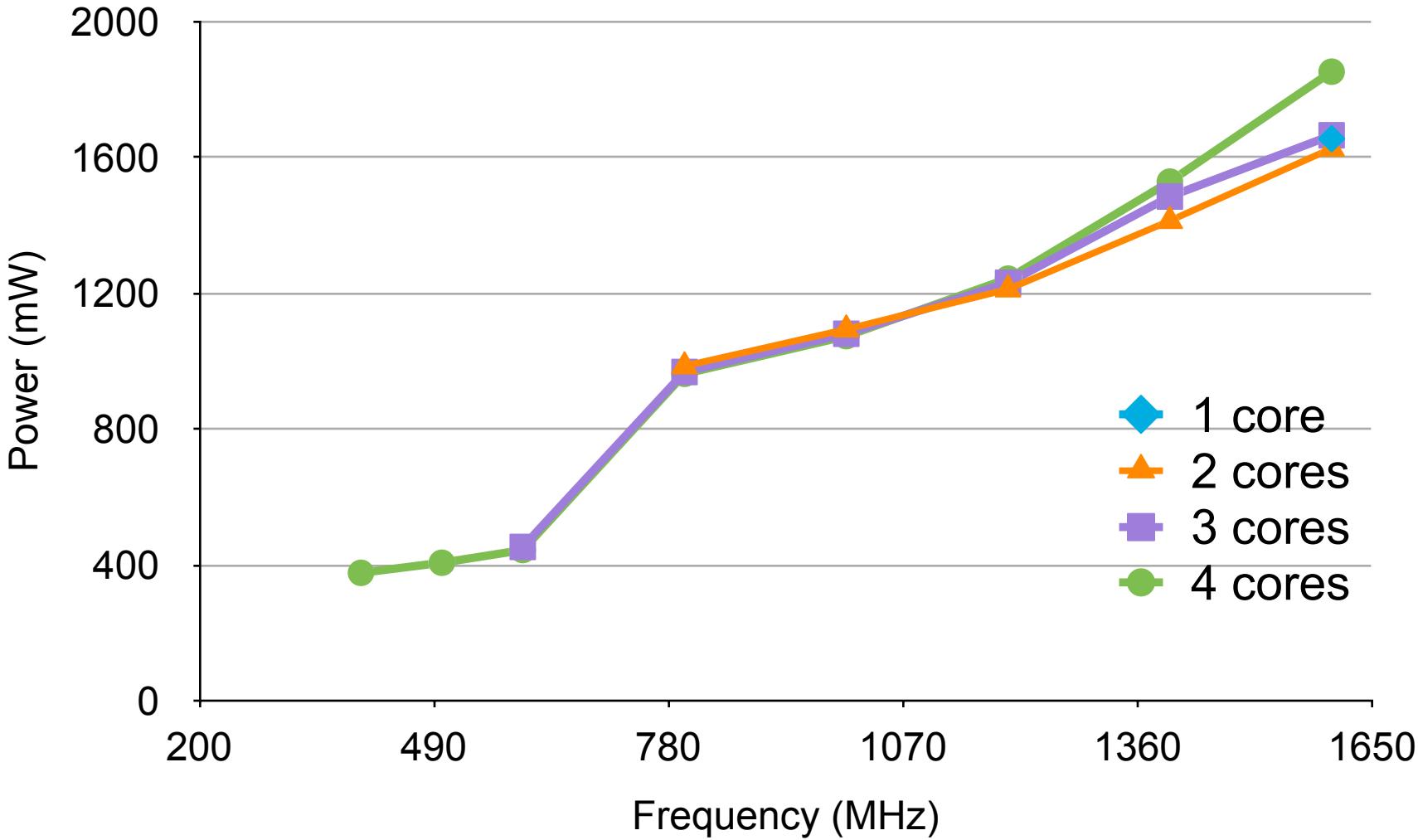


Devices



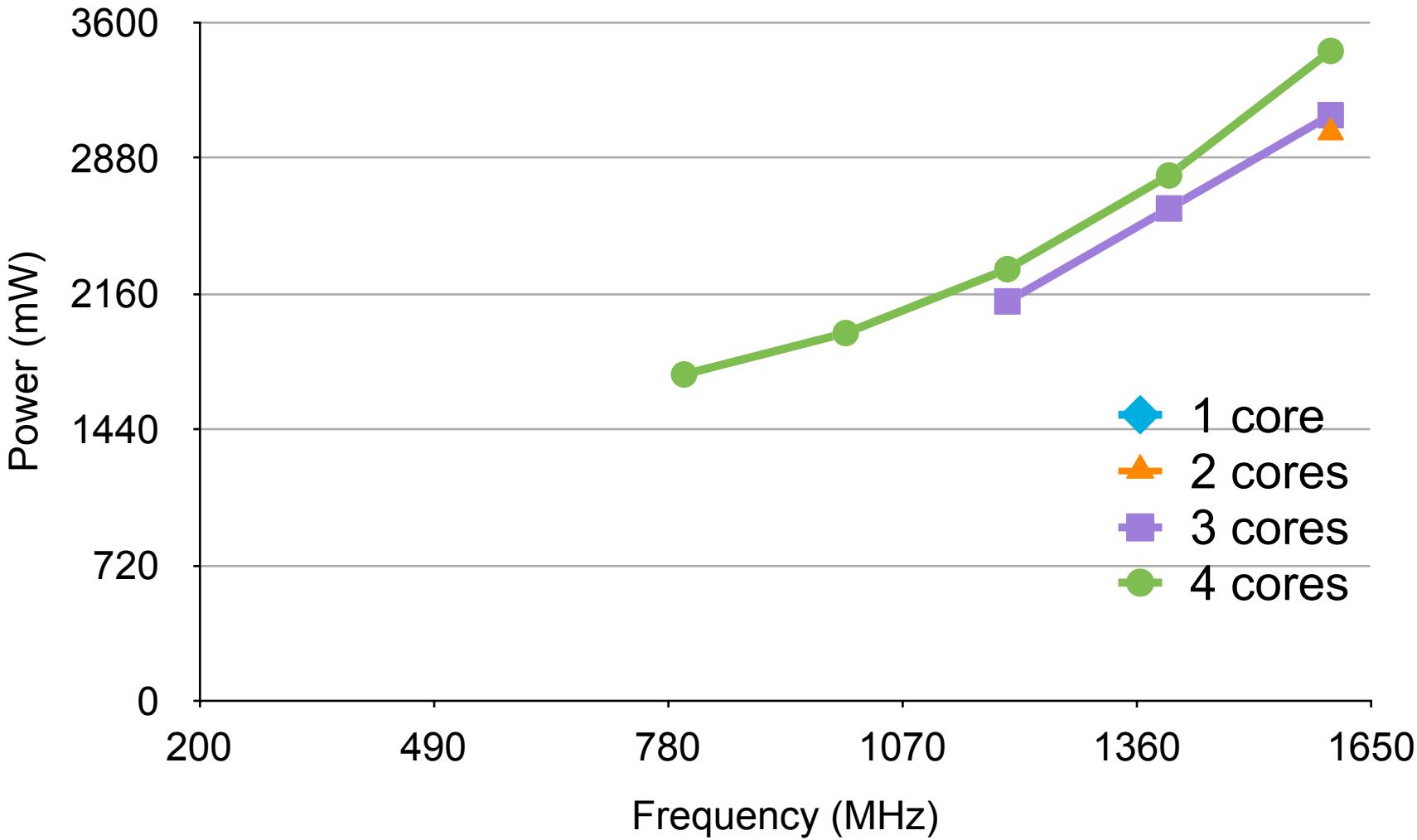


Exynos: loadcpu 25%



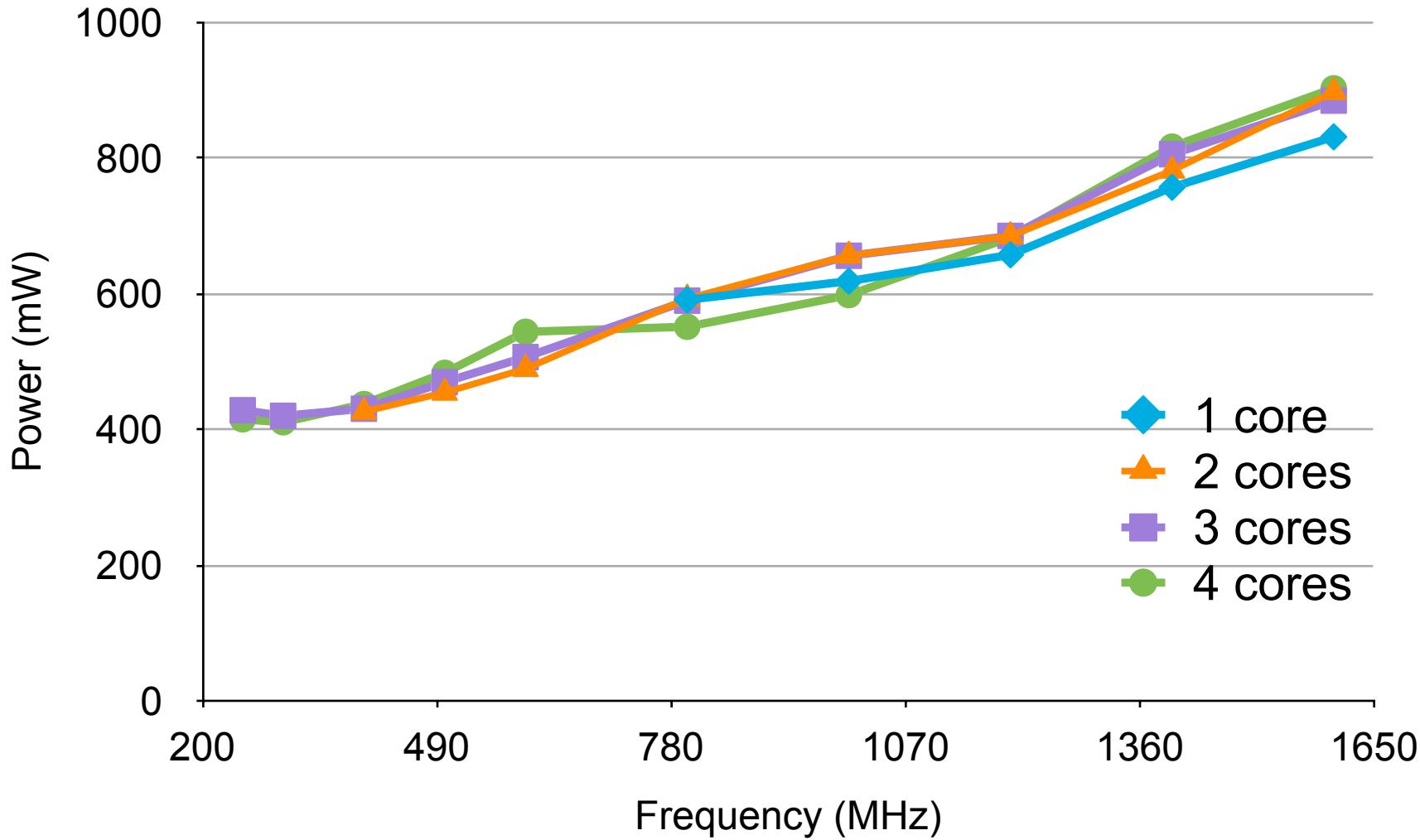


Exynos: loadcpu 50%





Exynos: loadcache 10%

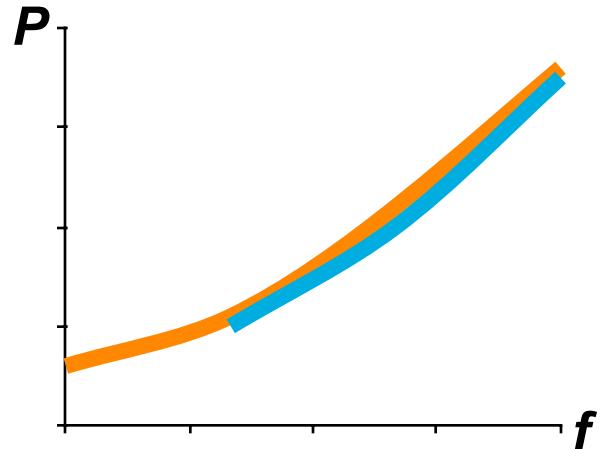




Exynos: observations

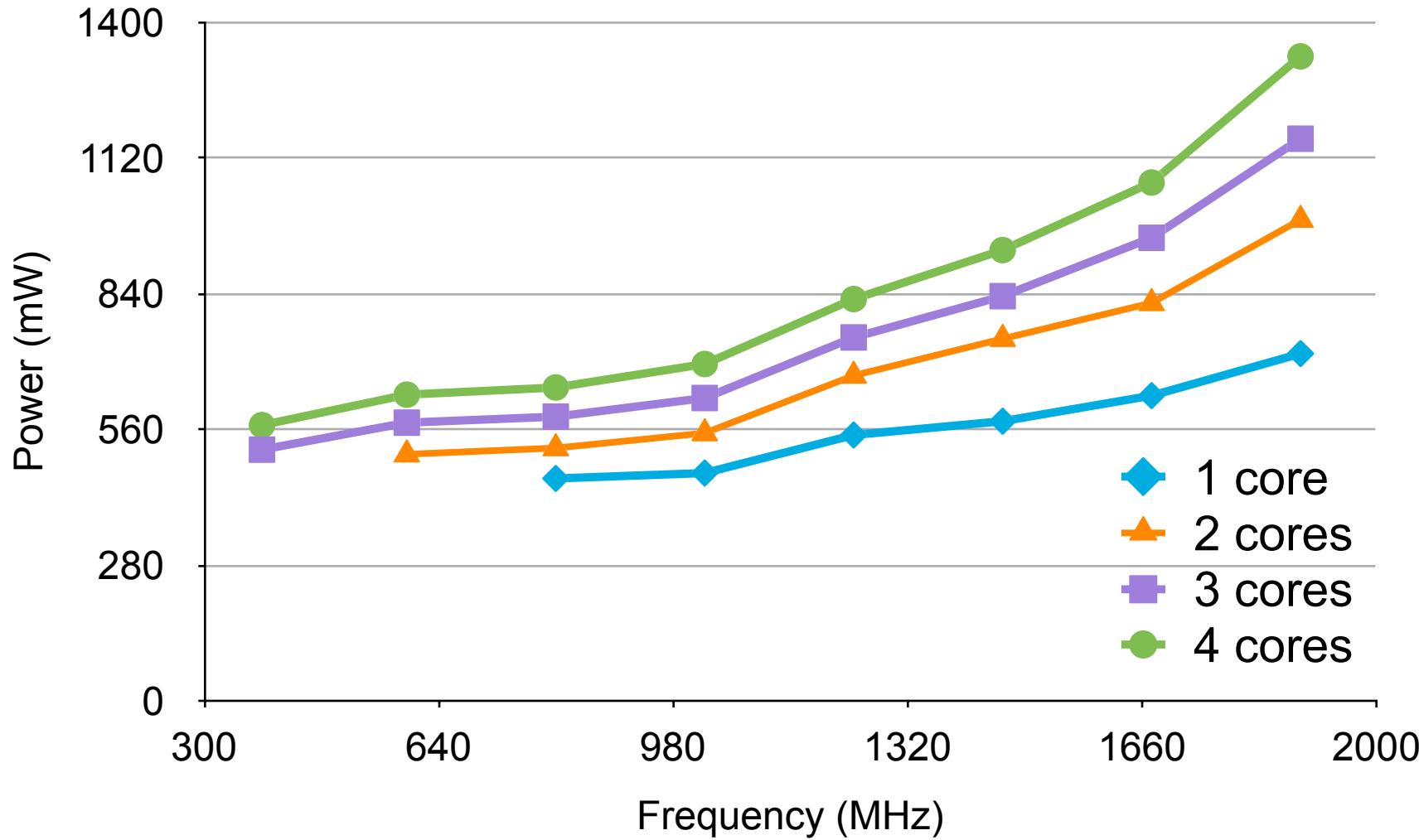


- Fix n : lower $f \Rightarrow$ lower P
- Fix f : P is independent of n
- Run at $\max \{ n \}$, $\min \{ f \}$



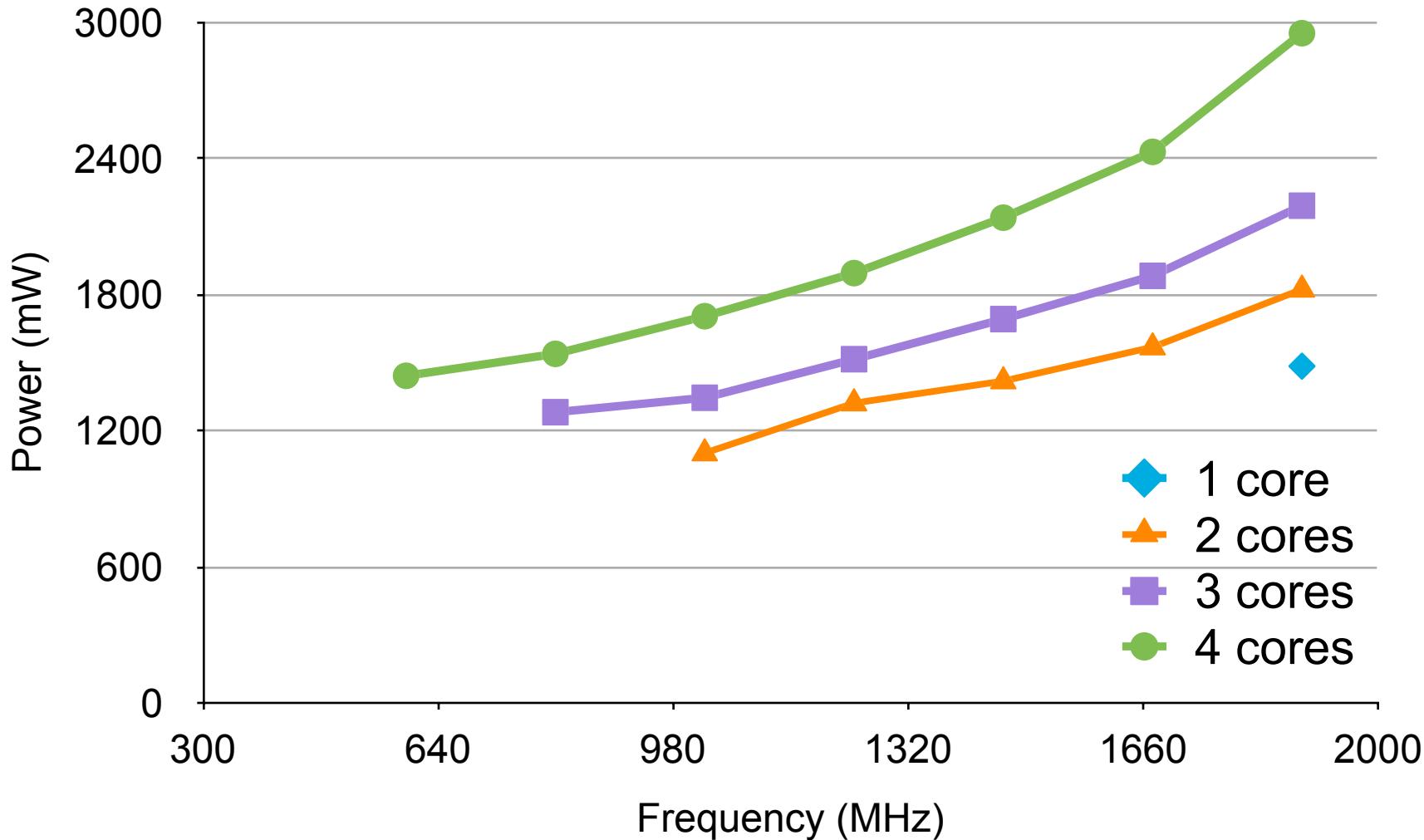


Snapdragon: loadcpu 10%





Snapdragon: loadmem 25%

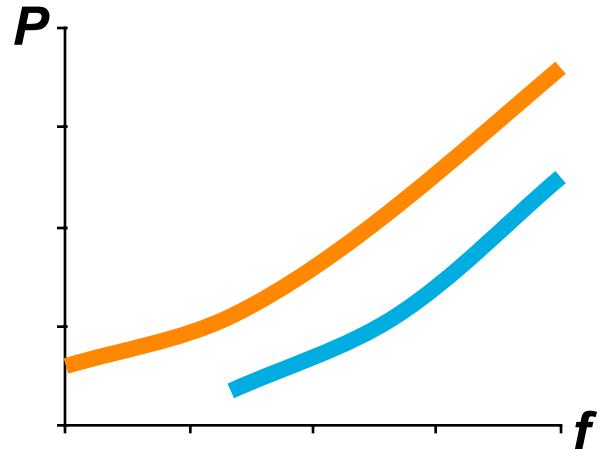




Observations



- Fix n : lower $f \Rightarrow$ lower P
 - Fix f : lower $n \Rightarrow$ lower P
 - Run at ???
- ~~Conflict~~



CPU model

$$P = P_{\text{uncore}} + n(P_{\text{dynamic}} + P_{\text{static}})$$

The diagram illustrates the components of a CPU model. The total power P is the sum of the power of the uncore (P_{uncore}) and the power of n online cores. The power of each online core is the sum of its active compute cost ($P_{\text{dynamic}} + P_{\text{static}}$). Handwritten annotations provide additional context:

- Total CPU power** points to the term P on the left.
- #online cores** points to the term n in the equation.
- Active compute cost** points to the term $P_{\text{dynamic}} + P_{\text{static}}$ on the right.
- Support circuitry** is listed as:
 - #cores independent
 - L2 cache
 - BusesA handwritten arrow points from this list to the P_{uncore} term.
- Cost to online core** is listed as:
 - Cost to support circuitry
 - Cost to memoryA handwritten arrow points from this list to the term $P_{\text{dynamic}} + P_{\text{static}}$.

Analysis

$$P = P_{\text{uncore}} + n(P_{\text{dynamic}} + P_{\text{static}})$$



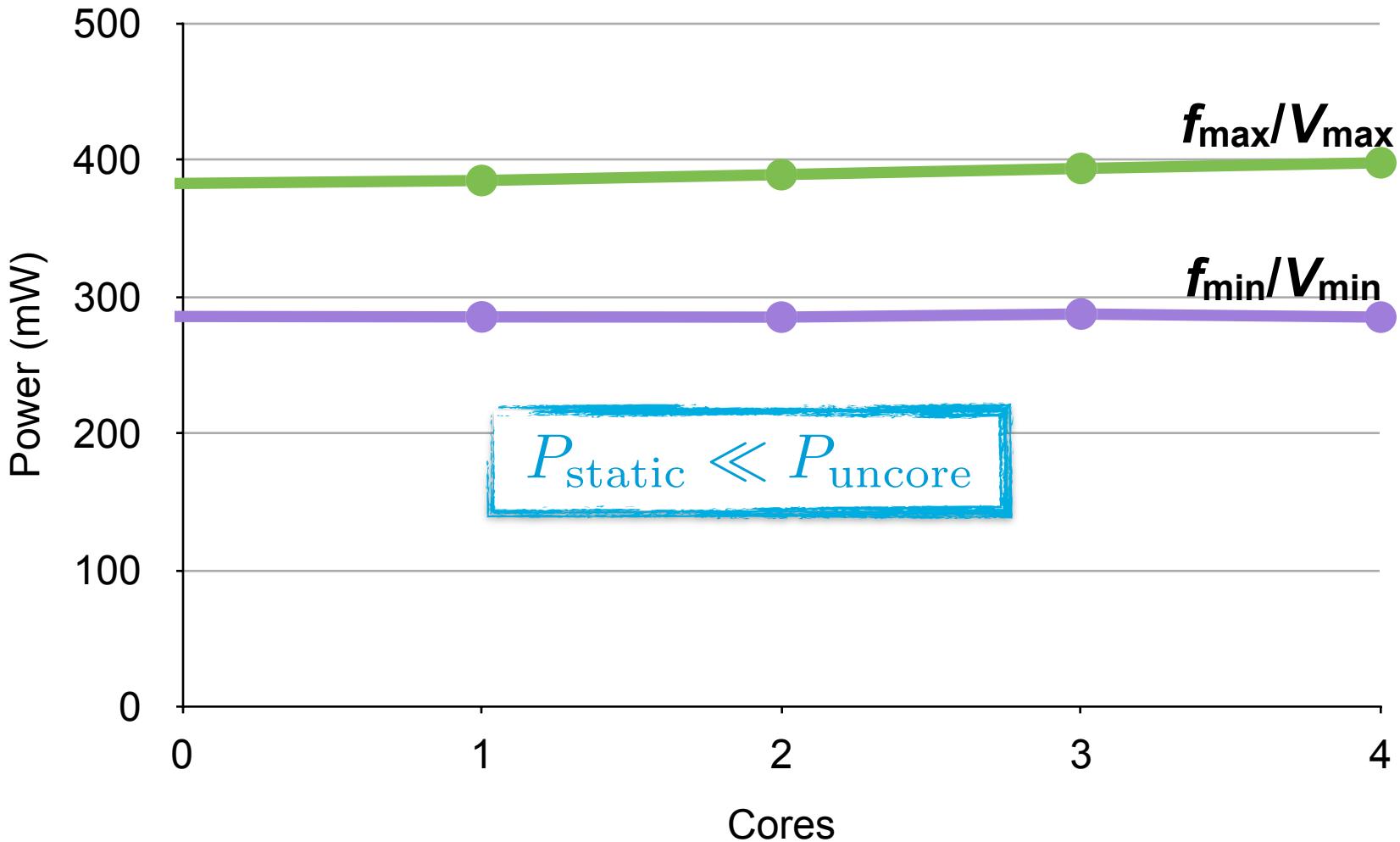
$$E = (P_{\text{uncore}} + nP_{\text{static}})T + k \quad (\textcolor{green}{T} = \text{period})$$

So if $P_{\text{static}} \ll P_{\text{uncore}}$:

E is independent of n

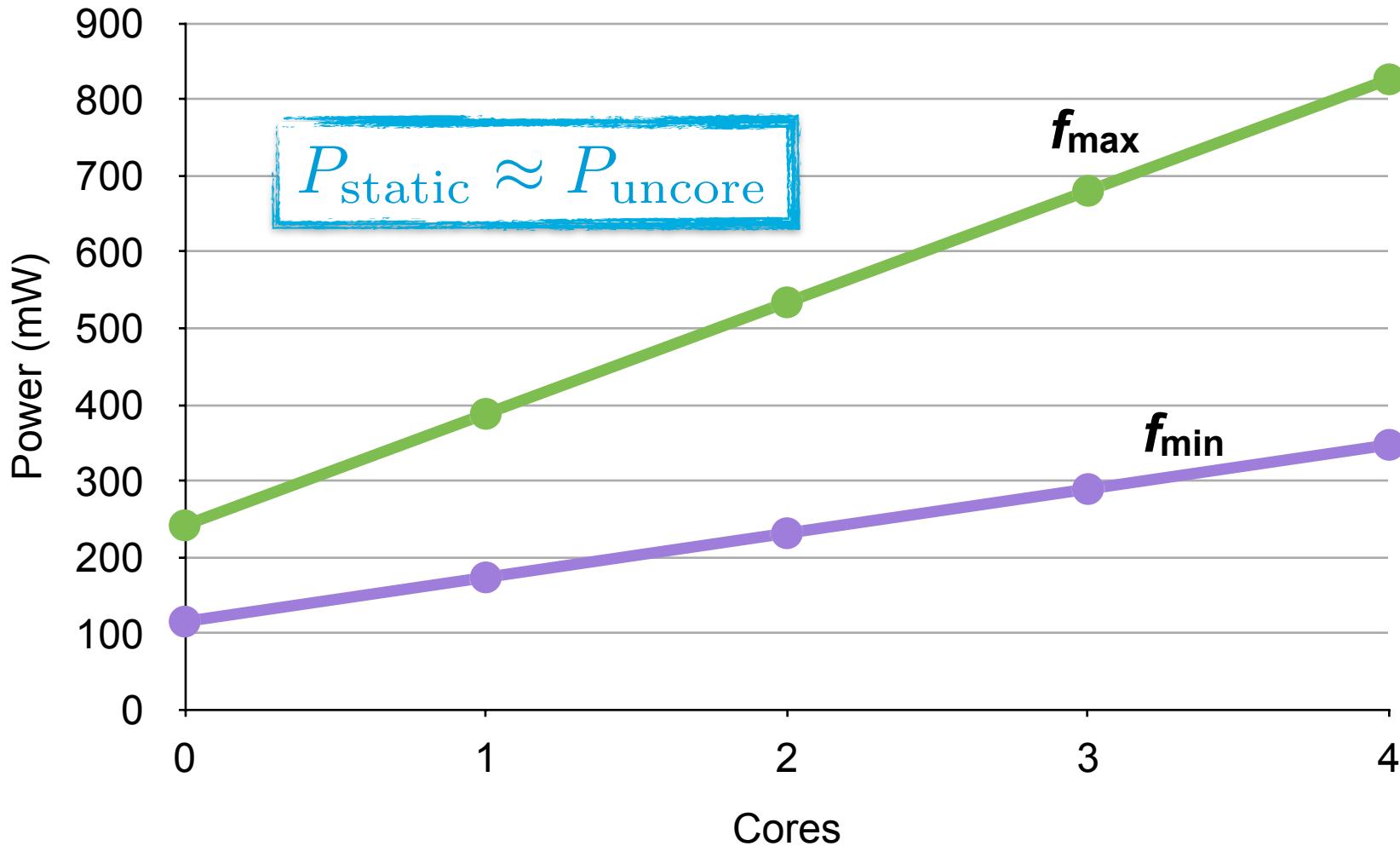


Exynos: idle power





Snapdragon: idle power



Analysis

- What if P_{static} is significant?

$$P = P_{\text{uncore}} + n(P_{\text{dynamic}} + P_{\text{static}})$$

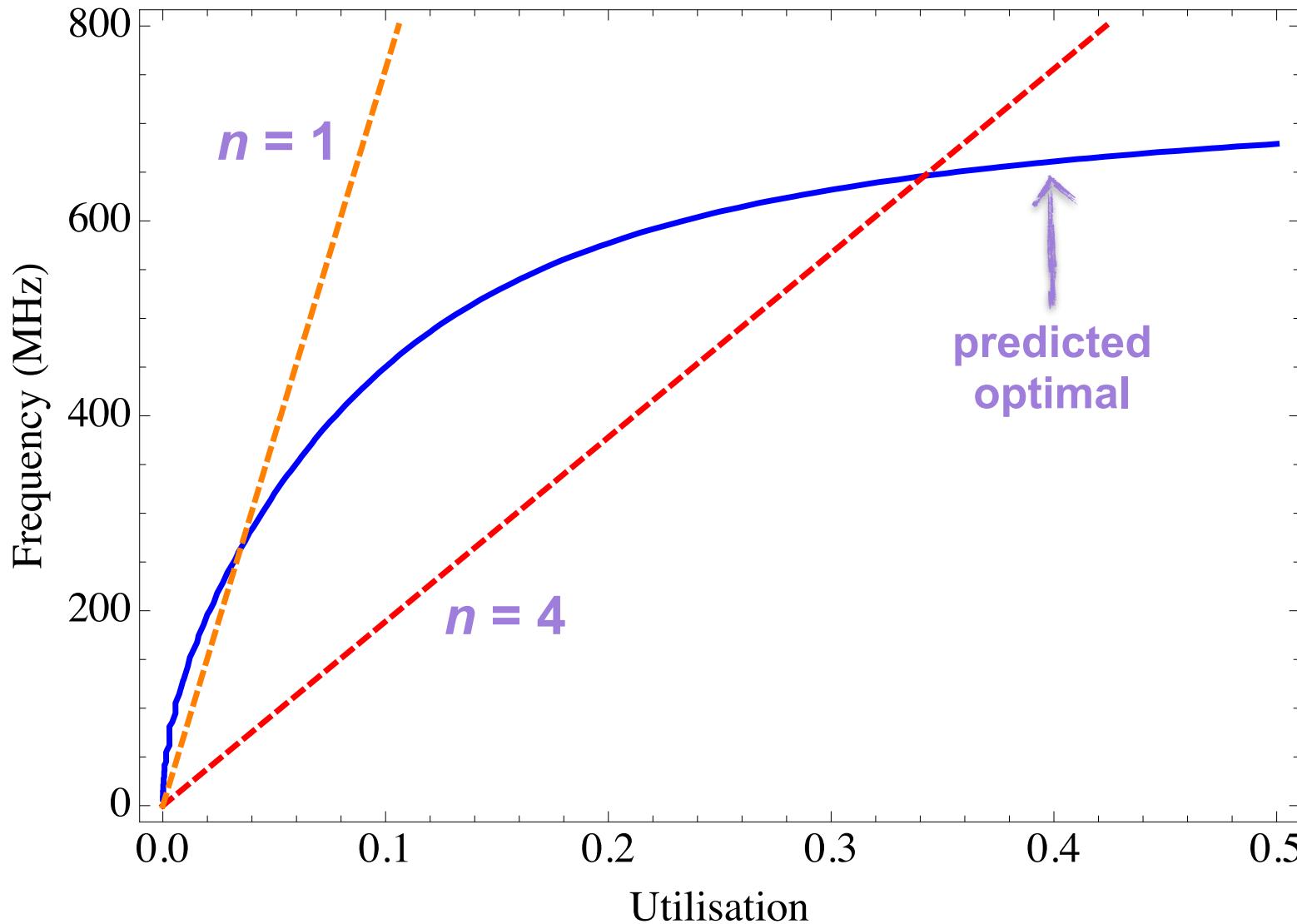
- Assumption: $n \propto 1/f$ (i.e. scalability)
- Then solve $\frac{d}{df}P(f) = 0$

Determine offline

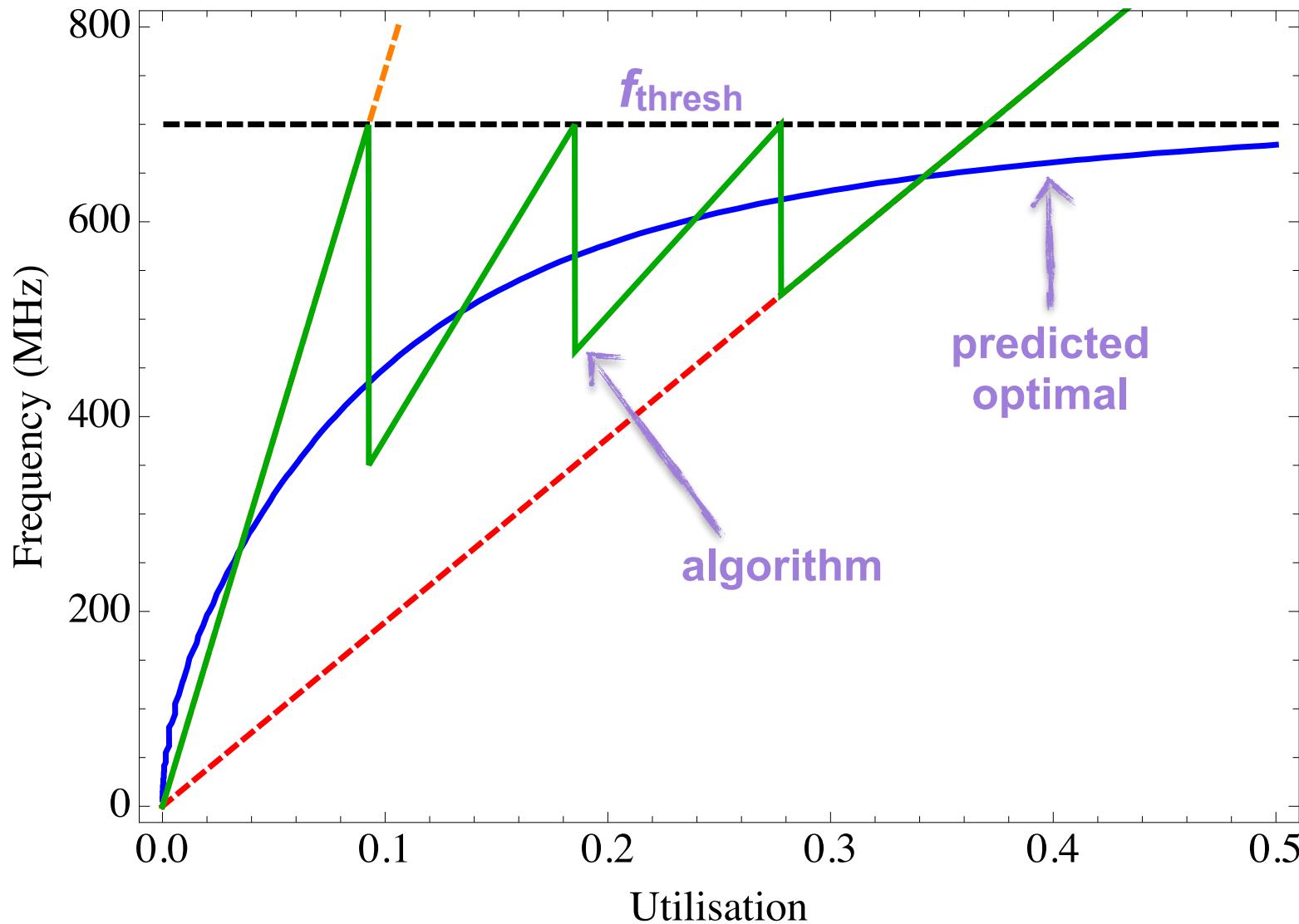
Known online

$$P'_{\text{CPU}} = P'_{\text{uncore}} + \gamma u \left(2C_{\text{eff}} VV' + \frac{P'_{\text{static}} f - P_{\text{static}}}{f^2} \right)$$


Analysis



Analysis



- **medusa**: Linux offline-aware governor
- keep utilisation < 100%
- parameterised by f_{thresh}
 - maintain $f < f_{\text{thresh}}$ by onlining cores as required
- practical considerations
 - available parallelism
 - changing workload
 - predicting effect of OP change
 - limiting OP switching

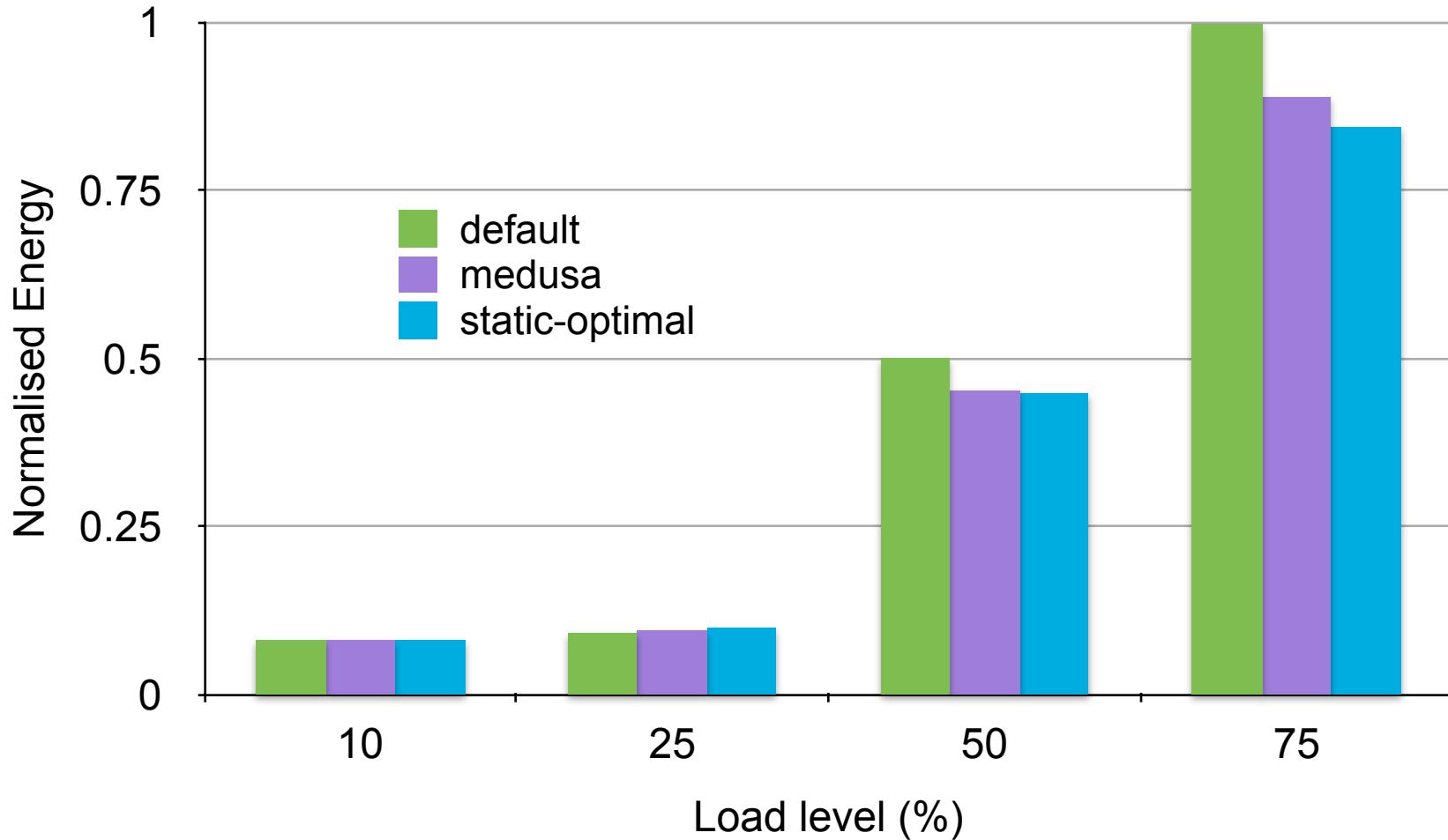
Determining f_{thresh}



- Low static power
 - zero!
 - always online cores before increasing f
- Otherwise
 - experimental
 - analytical

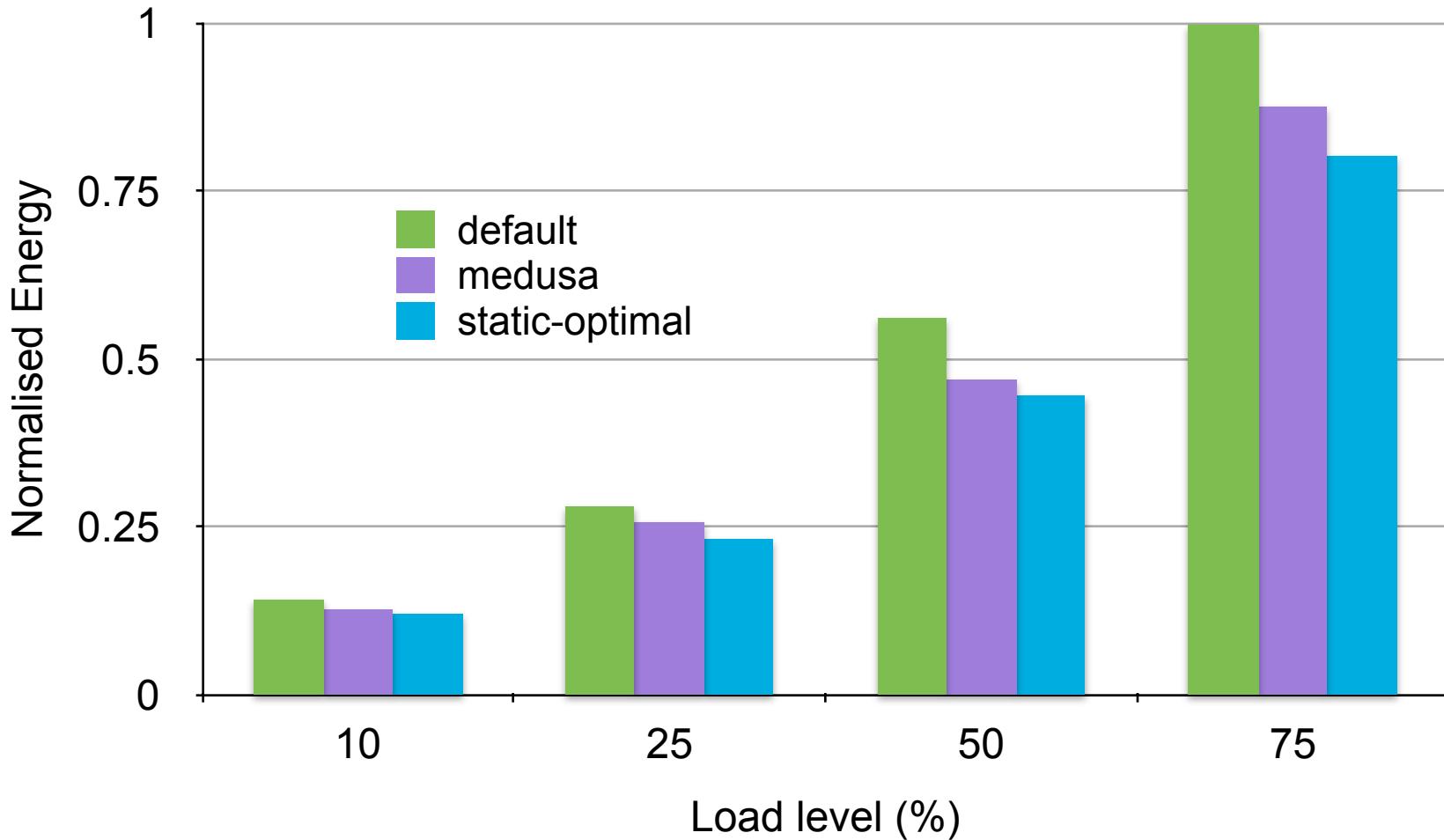


Exynos: medusa loadgen

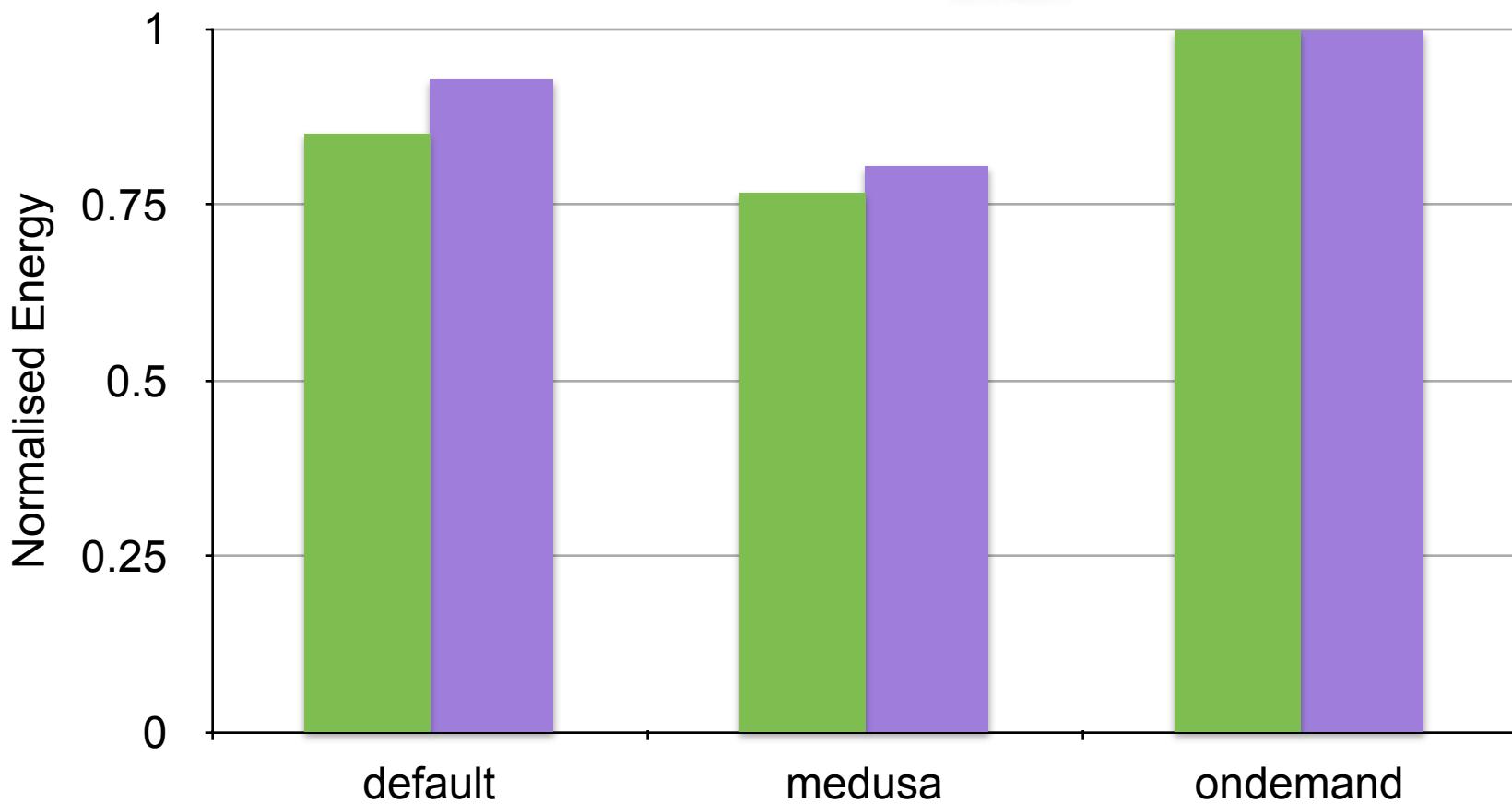
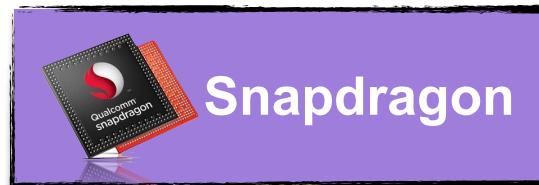




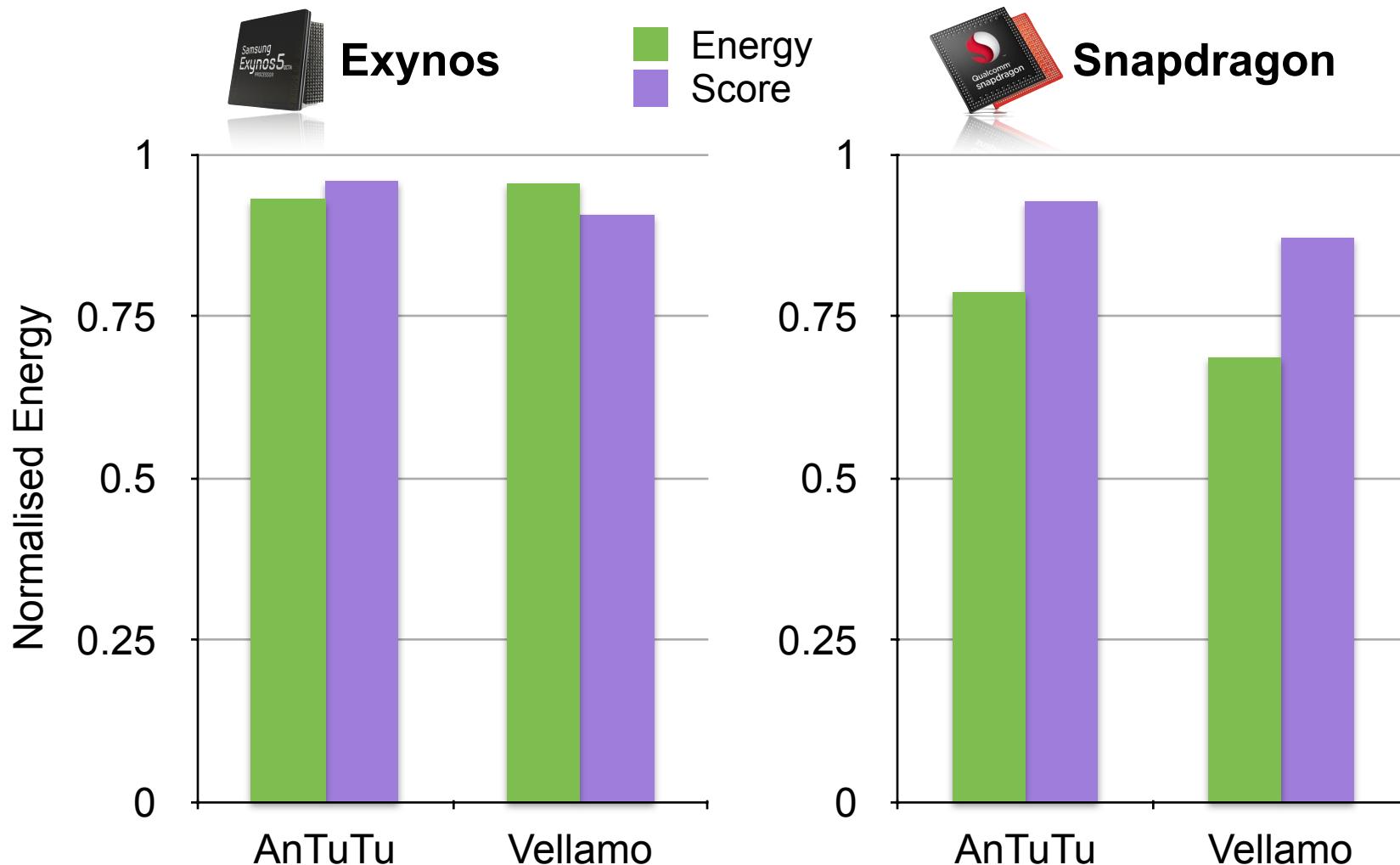
Snapdragon: medusa loadgen



medusa H.264 playback



medusa benchmarks



Summary

- Online cores for more efficient f
- Low P_{static} :
 - P independent of n
 - increase n before f
- High P_{static} :
 - increase f to a point (f_{thresh})
 - then prefer to increase n

Summary



Download medusa at

ssrg.nicta.com.au/projects/energy-management



NICTA

Unifying DVFS and Offlining in Mobile Multicores

Aaron Carroll
Gernot Heiser



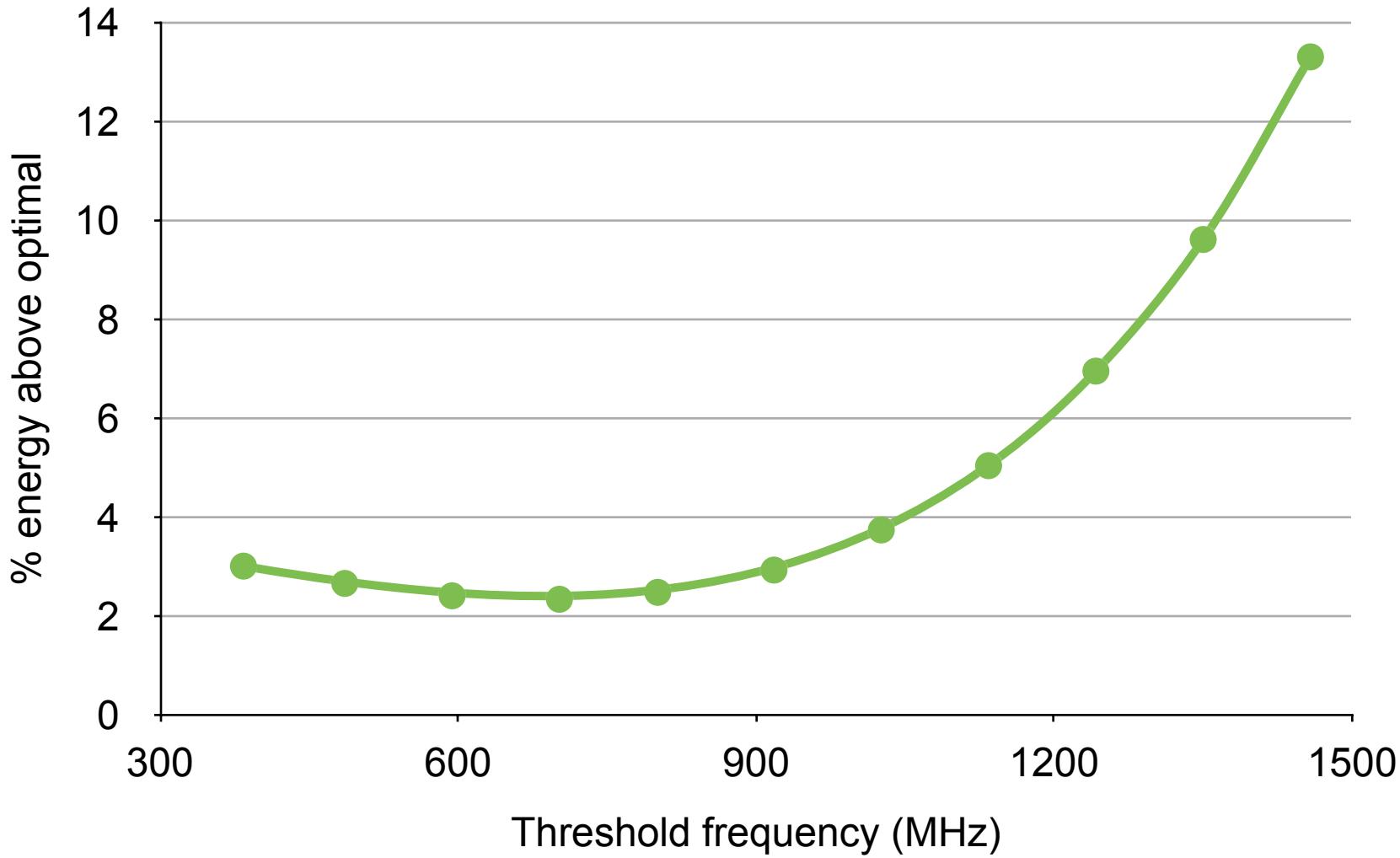
NICTA Funding and Supporting Members and Partners



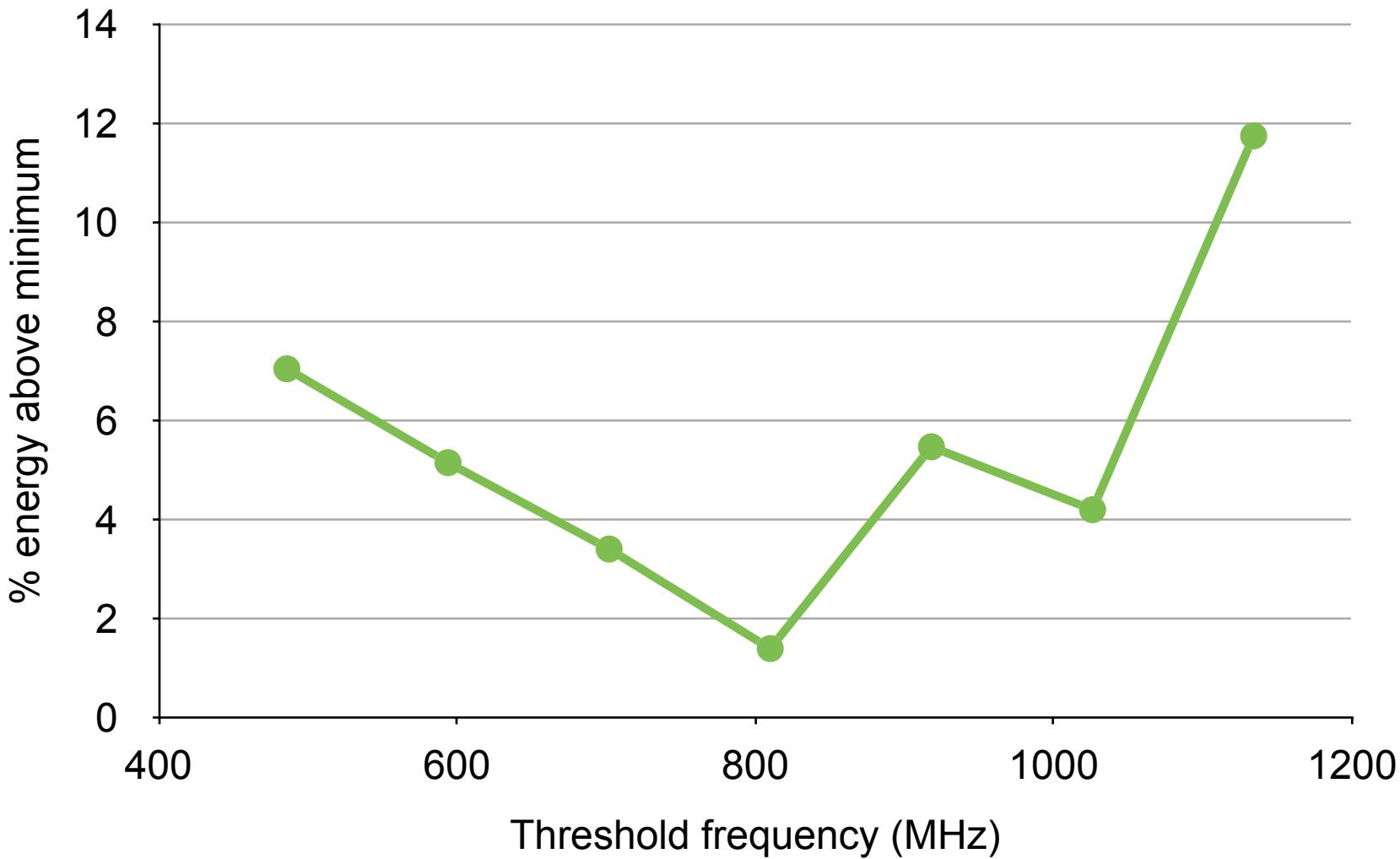
OP transition cost

	 Exynos	 Snapdragon		
	t (ms)	P (mW)	t (ms)	P (mW)
online/offline	22.6	357	11.7	454
online/offline	16.5	1305	4.09	1926
$f_{\min} \rightarrow$	2.10	1033	11.1	420
$f_{\min} \rightarrow$	0.39	320	6.05	317
$f_{\max} \rightarrow$	1.20	1400	0.24	838

f_{thresh} prediction



f_{thresh} sensitivity



P_{static} variation

