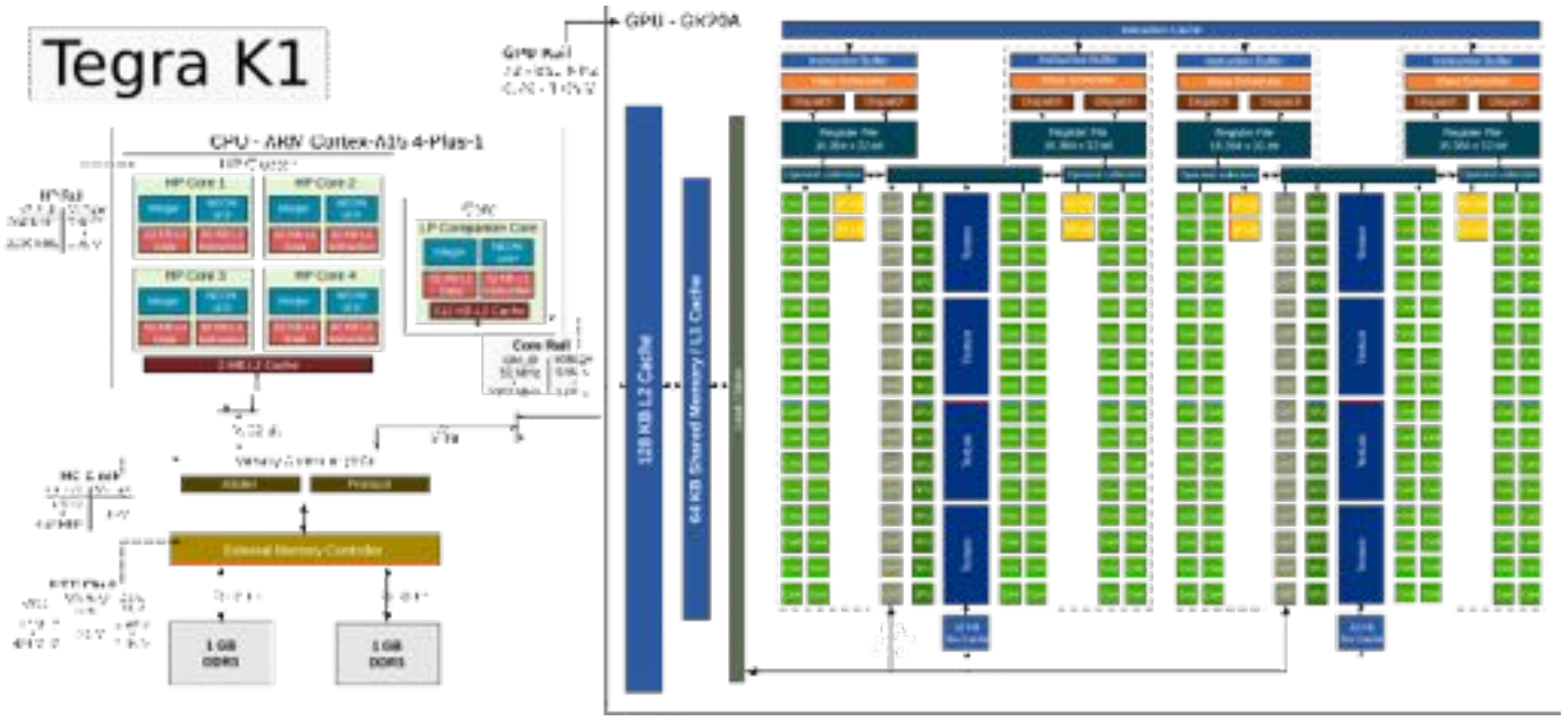
A High Precision Power Model for the Tegra K1 CPU, GPU and RAM

Kristoffer Robin Stokke, Håkon Kvale Stensland, Pål Halvorsen
Department of Informatics, University of Oslo, Norway

[simula . research laboratory]

Power modelling is an important topic in many areas of computing, for example to save energy in texture streaming for gaming[1] or to select efficient H.264 video encoding parameters[2]. However, researchers' view of how hardware consume power is limited. They typically resort to rate-based models to describe the energy consumption of hardware, where power usage is correlated directly with hardware access rates (for example instructions or cache misses per second)[3,4,5,6]. This approach ignores many mechanisms that impact the power usage of a system, such as rail voltages, core- and clock-gating, frequency scaling and variable cost of instruction execution. Because of this, they can mispredict up to 70 % on the Tegra K1. We show that by taking all these factors into account with sufficient hardware knowledge, it is possible to bridge the gap between power usage and software execution to build power models which are over 98 % accurate over all CPU, GPU and memory frequency combinations.



Power Modelling

- Power models are usually built using multivariable linear regression. The difference between them is the underlying analytical assumptions of how hardware consumes energy.
- Rate-based models. Power is correlated with hardware access rates, such as instructions and cache misses per second (ρ_{inst} or ρ_{cache}). Ignores rail voltage and therefore mispredicts up to 70 % on the Tegra K1.

$$P_{total} = \beta_0 + \sum^{\rho \in \mathbb{P}} \beta_\rho \rho \qquad (1)$$

- Modelling switching capacitance αC on each rail R_{hp}, R_{core}, R_{gpu} and R_{mem}. Switching capacitance on rails (for example R_{core} vary with frequencies in other domains such as memory. This is not accounted for in this model, causing it to perform better than rate-based models but still mispredict up to 60 %.

$$P_{total} = \sum^{R \in \mathbb{R}} \alpha C V_R^2 f_R \qquad (2)$$

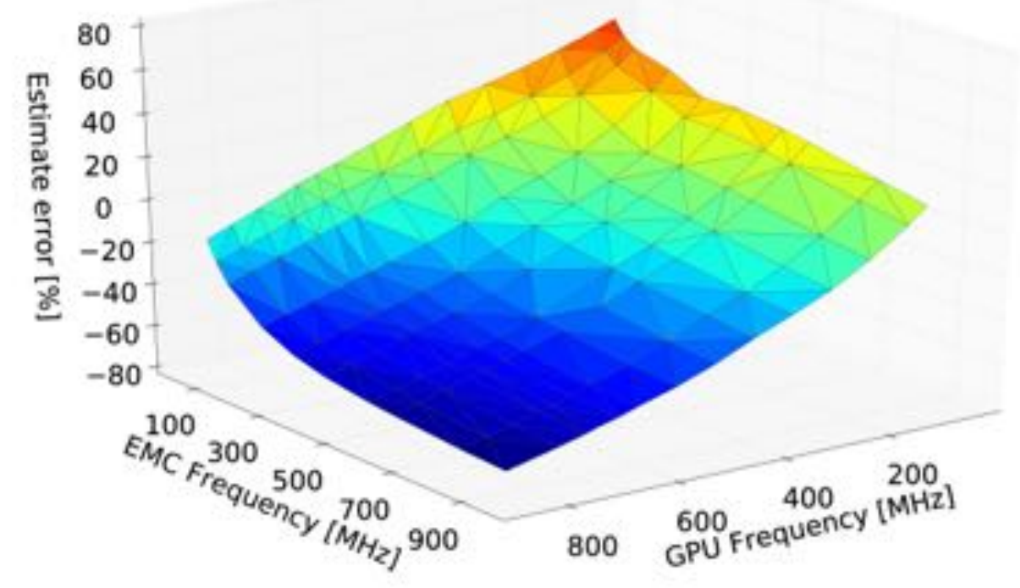
- Our *hybrid approach* combines the strengths of each model, taking into account measured rail voltages using the standard equations for CMOS dynamic and static power. We also model hardware activity on all rails using a diverse set of predictor access rates for the GPU, CPU clusters and memory.
  - Static power on a rail

$$P_{R,stat} = I_{R,leak} V_R \qquad (3)$$

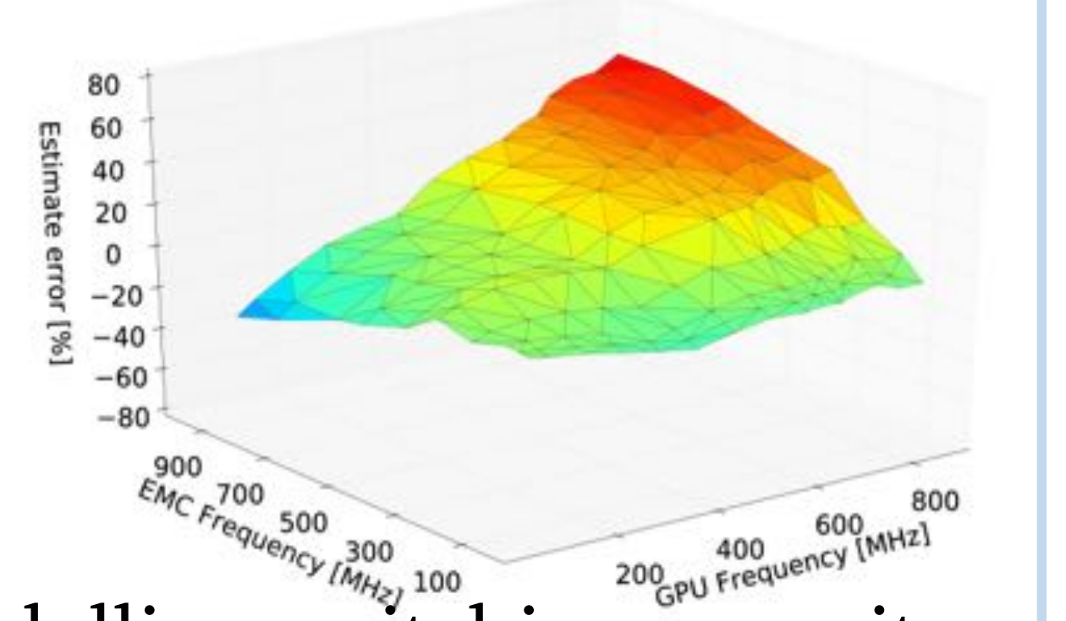
  Dynamic power on a rail

$$P_{R,dyn} = \sum^{\rho \in \mathbb{P}_R} C_{R,\rho} \rho V_R^2 \qquad (4)$$

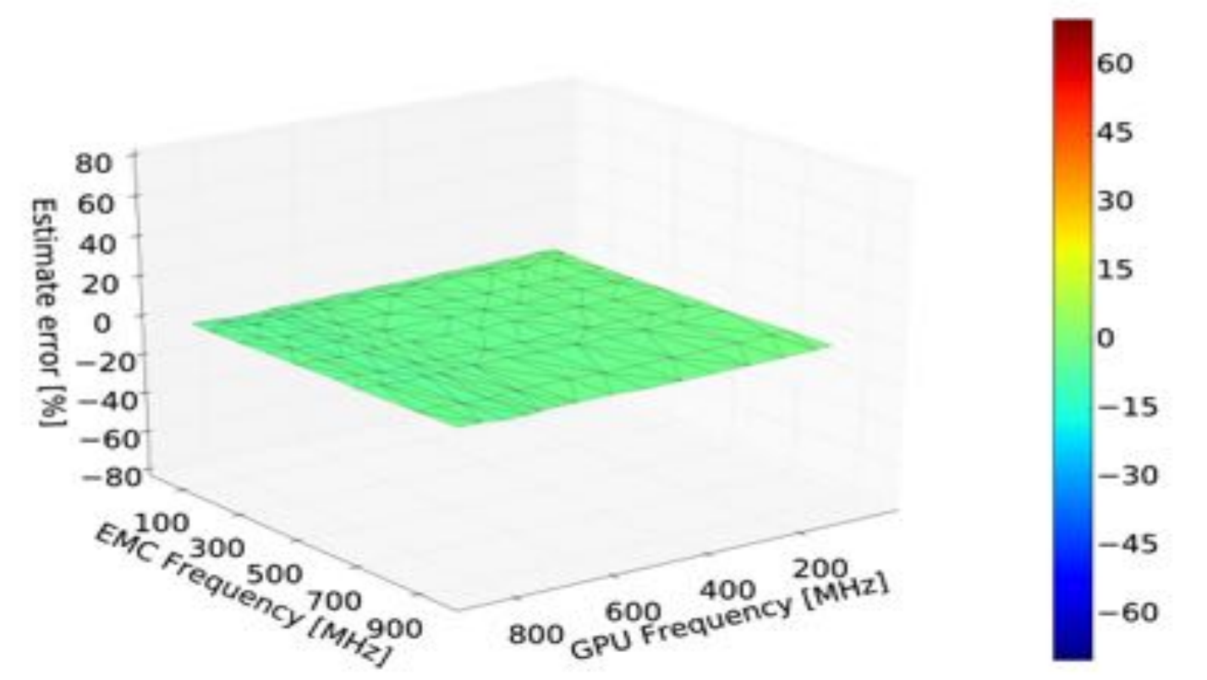
Comparison of Methods



Rate-based model error.



Modelling switching capacitance.



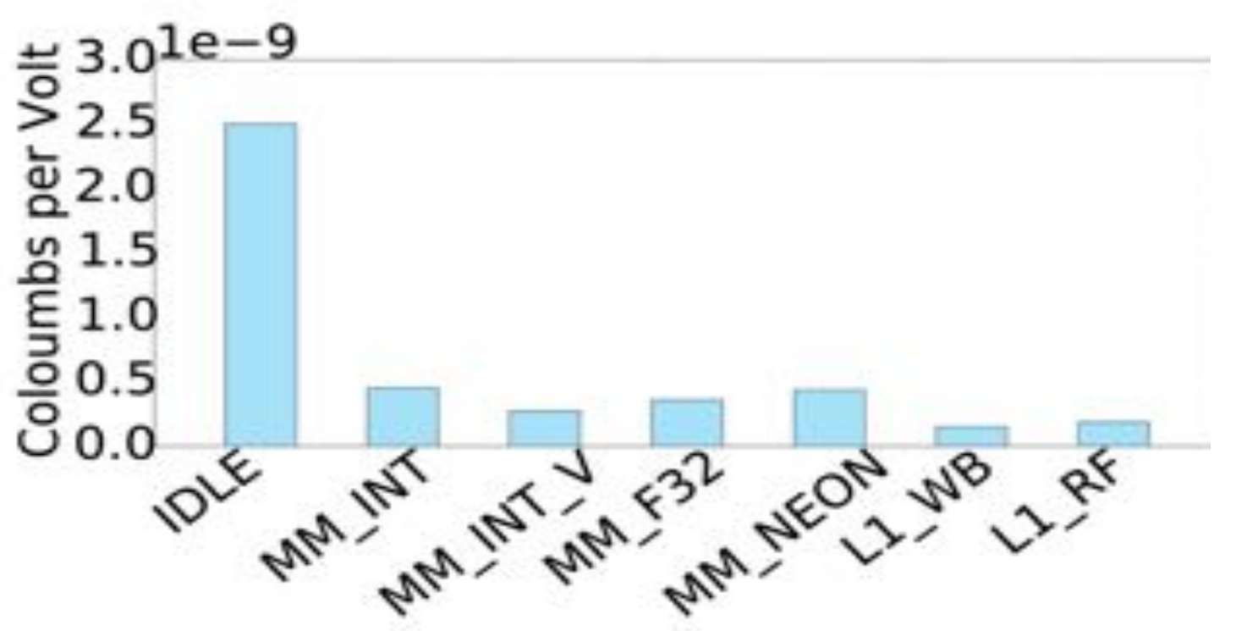
Our hybrid model (motion estimation).

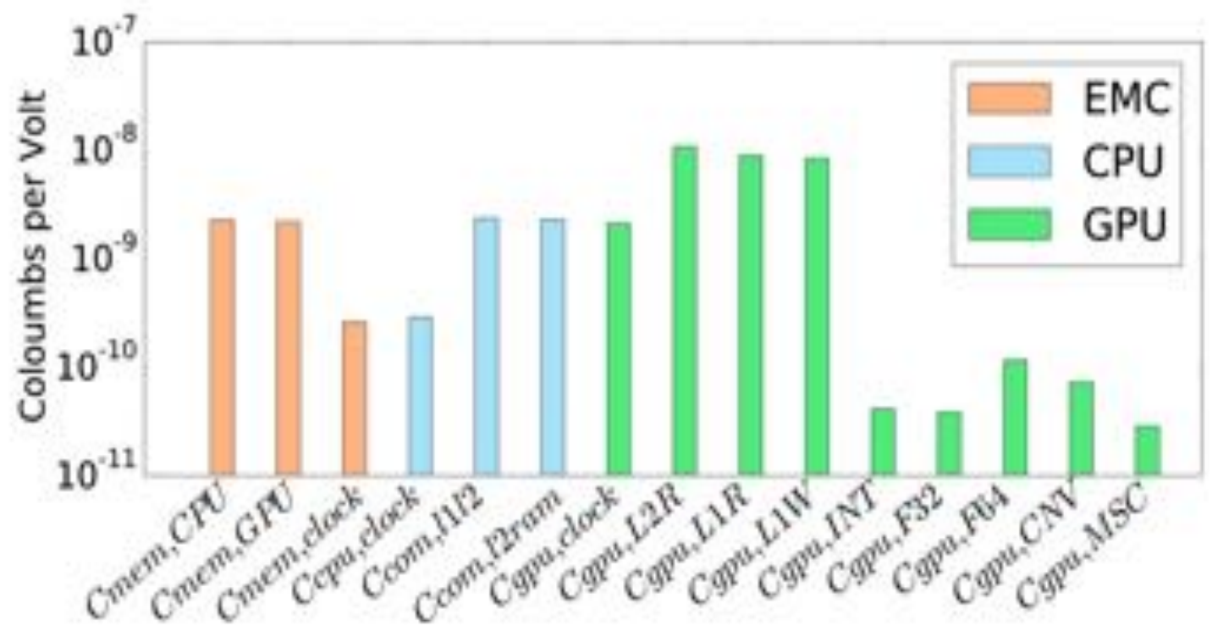
Model Predictors and Construction

Rail	Predictor	Description	Coefficient	Value
GPU	V_{gpu}	GPU voltage	$I_{gpu,leak}$	0.274
	$\rho_{gpu,clock}$	Total clock cycles per second	$C_{gpu,clock}$	$2.10\frac{nC}{V}$
	$\rho_{gpu,L2R}$	L2 cache 32B reads per second	$C_{gpu,L2R}$	$10.79\frac{nC}{V}$
	$\rho_{gpu,L1R}$	L1 cache 4B reads per second	$C_{gpu,L1R}$	$8.90\frac{nC}{V}$
	$\rho_{gpu,L1W}$	L1 cache 4B writes per second	$C_{gpu,L1W}$	$8.43\frac{nC}{V}$
	$\rho_{gpu,INT}$	Integer instructions per second	$C_{gpu,INT}$	$41.11\frac{pC}{V}$
	$\rho_{gpu,F32}$	Float (32-bit) instructions per second	$C_{gpu,F32}$	$38.15\frac{pC}{V}$
	$\rho_{gpu,F64}$	Float (64-bit) instructions per second	$C_{gpu,F64}$	$115.33\frac{pC}{V}$
	$\rho_{gpu,CNV}$	Conversion instructions per second	$C_{gpu,CNV}$	$72.42\frac{pC}{V}$
	$\rho_{gpu,MSC}$	Miscellaneous instructions per second	$C_{gpu,MSC}$	$28.36\frac{pC}{V}$
Memory	$\rho_{mem,clock}$	Total clock cycles per second	$C_{mem,clock}$	$258.06\frac{pC}{V}$
	$\beta_{mem,204}$	Power offset at 204 MHz	$P_{mem,204}$	$-0.03W$
	$\beta_{mem,300}$	Power offset at 300 MHz	$P_{mem,300}$	$0.05W$
	$\rho_{mem,CPU}$	CPU busy memory cycles per second	$C_{mem,cpu}$	$2.25\frac{nC}{V}$
	$\rho_{mem,OTH}$	Other (GPU) busy memory cycles per second	$C_{mem,oth}$	$2.17\frac{nC}{V}$
Core	V_{core}	Core rail voltage (always powered)	$I_{core,leak}$	$633.7mA$
	$\rho_{core,clk}$	Active clock cycles per second (LP core)	$C_{core,clk}$	$301.49\frac{pC}{V}$
HP	V_{hp}	HP rail voltage (when powered)	$I_{hp,leak}$	$59.8mA$
	$\rho_{hp,clk1}$	Active clock cycles per second (first core)	$C_{hp,clk1}$	$395.65\frac{pC}{V}$
	$\rho_{hp,clk2}$	Active clock cycles per second (second core)	$C_{hp,clk2}$	$270.40\frac{pC}{V}$
	$\rho_{hp,clk3}$	Active clock cycles per second (third core)	$C_{hp,clk3}$	$261.89\frac{pC}{V}$
	$\rho_{hp,clk4}$	Active clock cycles per second (fourth core)	$C_{hp,clk4}$	$213.95\frac{pC}{V}$
HP+Core	$V_{com,online}$	Rail voltage when any core is online (not gated)	$I_{cpu,leak}$	$24.00mA$
	$\rho_{com,l1l2}$	Cache maintenance, L1 and L2	$C_{com,l1l2}$	$2.35\frac{nC}{V}$
	$\rho_{com,l2ram}$	Cache maintenance, L2 and RAM	$C_{com,l2ram}$	$2.29\frac{nC}{V}$
	$\rho_{com,ips}$	Instructions per second (workload-specific)	$C_{com,ips}$	N/A
Other	P_{base}	Base power	-	$0.78W$

Model is trained by stressing the various architectural units (for example integer, floating point, cache) using specialised benchmarks. All benchmarks are run over all possible combinations of frequencies, cores and cluster. Predictor sources:

- Voltage measurements
- CUPTI for GPU hardware performance counters
- PERF for CPU hardware performance counters
- Modified kernel sources to trace per-core clock- and power-gating

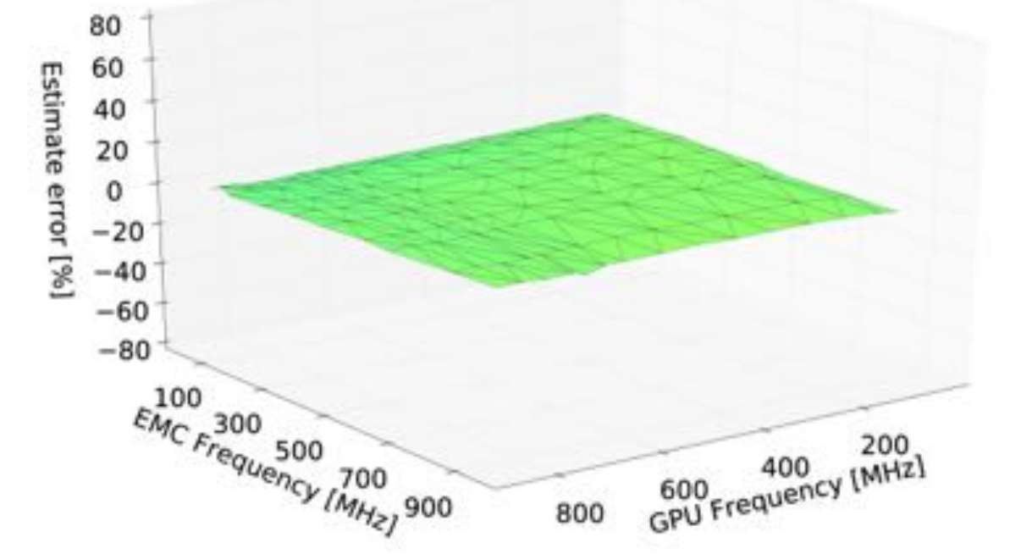




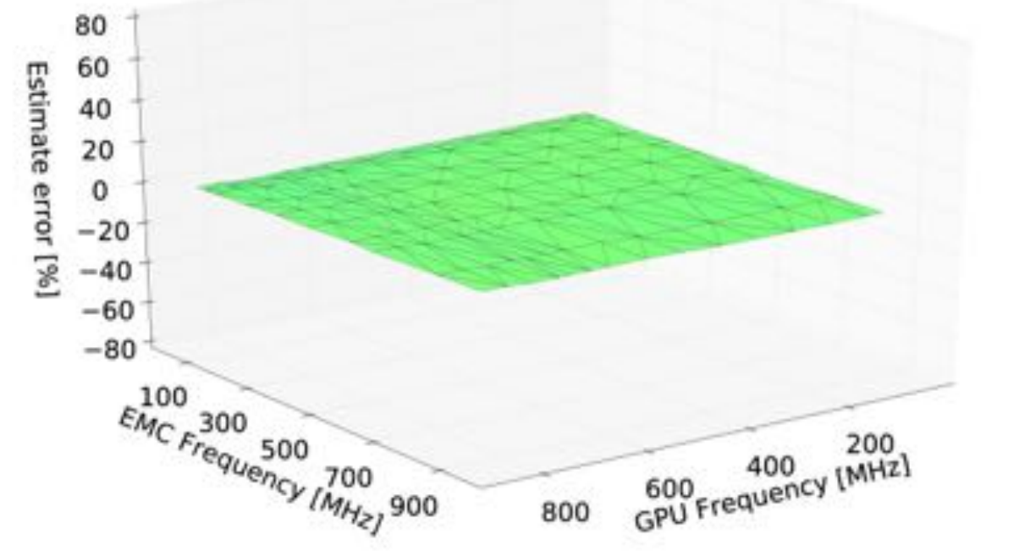
Benchmark	Description	Components / Instructions under explicit stress										
		CPU	RAM (CPU)	GPU	RAM (GPU)	L2	L1	INT	F32	F64	Conv.	Misc.
Idle CPU	GPU off, CPU in idle state.	✓										
CPU-workload	GPU off, CPU processing.	✓	✓									
Idle GPU	GPU on and idle, CPU in idle state.	✓		✓								
L2 Read	Stresses L2 cache reads only.	✓		✓		✓						
L1 Read	Stresses L1 cache reads.	✓	✓	✓			✓					
L1 Write	Stresses L1 cache writes.	✓	✓	✓			✓					
RAM	Stresses RAM activity (GPU EMC).	✓		✓	✓							
Integer	Stresses integer arithmetic unit.	✓		✓		✓		✓				
Float32	Stresses floating point unit.	✓		✓		✓		✓	✓			
Float64	Stresses floating point unit.	✓		✓		✓		✓		✓		
Control	Stresses conversion instructions.	✓		✓		✓		✓			✓	
Misc	Stresses miscellaneous instructions.	✓		✓		✓		✓				✓

Evaluation

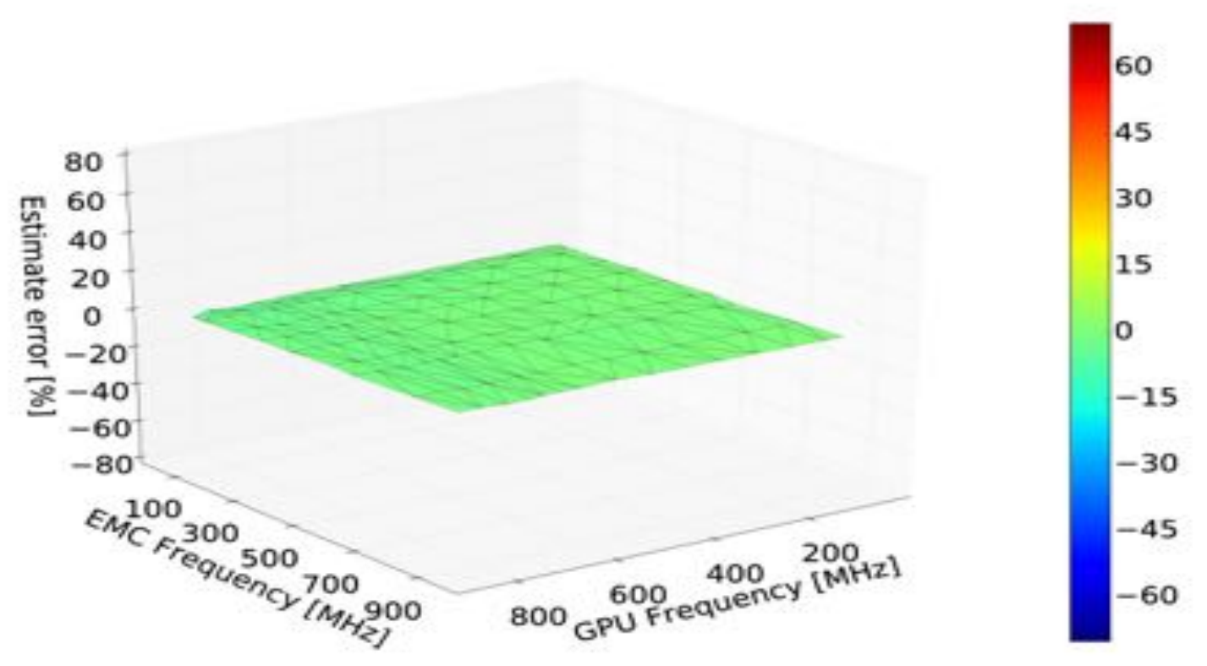
GPU Model Error for Video Processing Filters



Debarreling

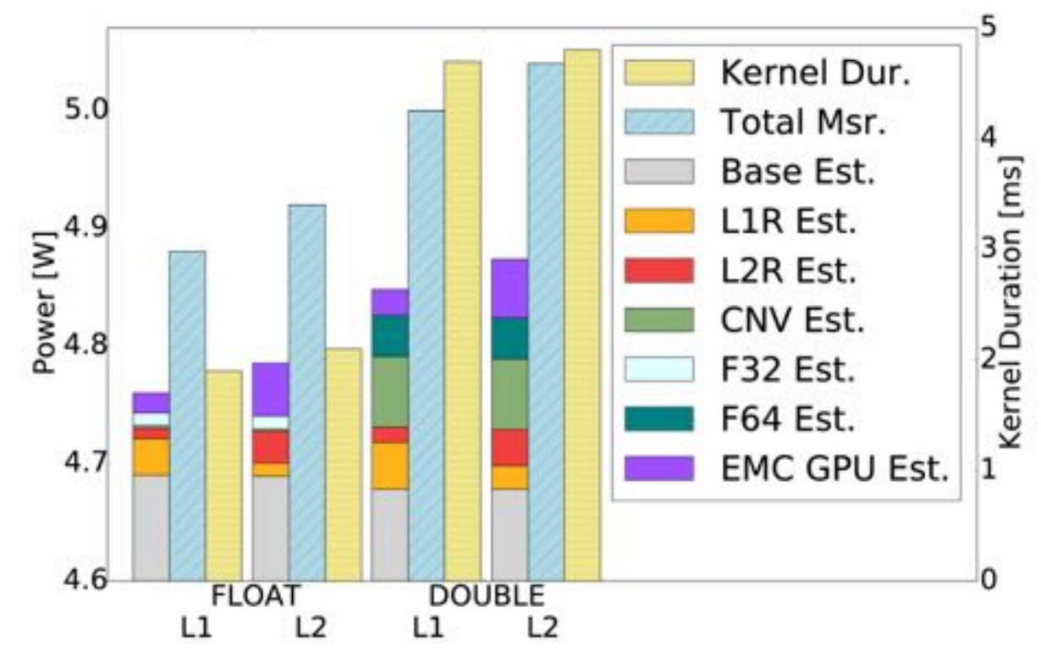


DCT

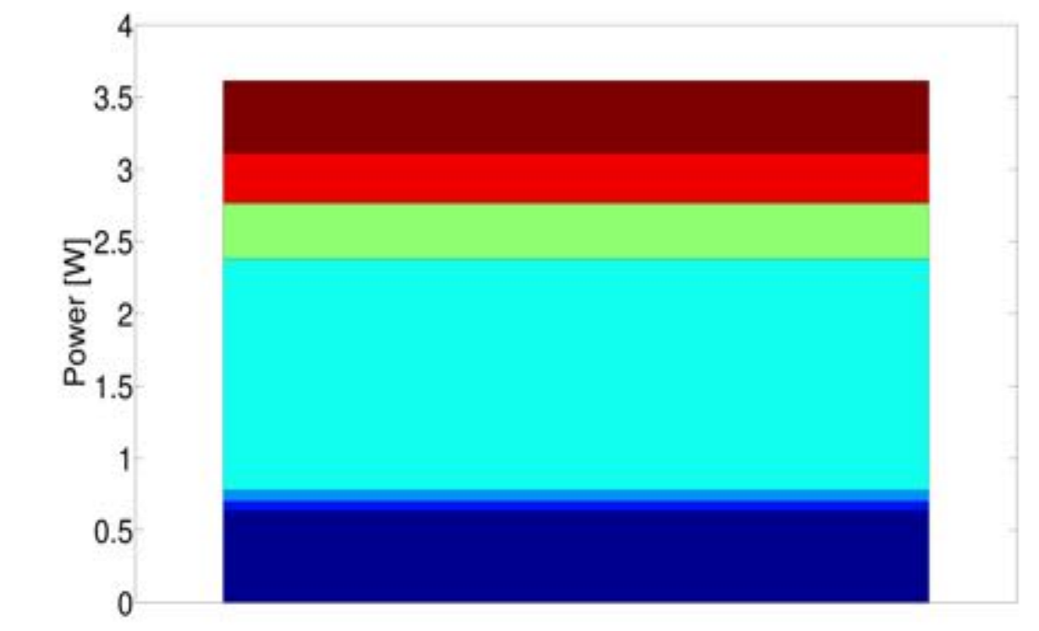


Rotation

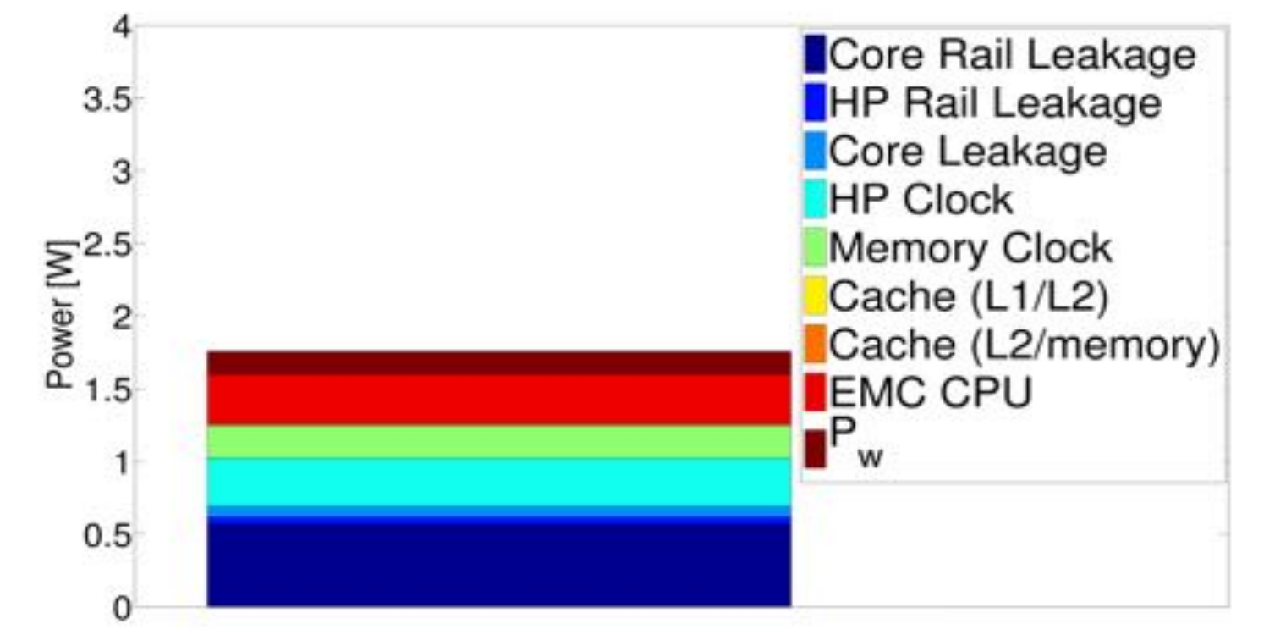
Optimising for Power by Exploiting Non-Coherent Cache, Shorter Datatypes and Frequency



DCT



Max CPU/mem frequency



Memory frequency = 528 MHz
CPU frequency = 564 MHz

Concluding Remarks

We have shown that it is possible to build high-precision power models for the Tegra K1's GPU, CPU and memory by incorporating voltage measurements and an extensive amount of hardware activity predictors into the model. The model is more than 98 % accurate, and reveals how the Tegra K1 draws power under the influence of software. Using the model, we are also able to optimise power usage of video processing filters by utilising non-coherent caches, smaller datatypes and lowering operating frequencies without compromising performance.

References

[1] M. Hosseini, J. Peters, and S. Shirmohammadi. Energy Budget Compliant Adaptive 3D Texture Streaming in Mobile Games. In Proc of MMSys, pages 1–11, 2013.

[2] Y. O. Sharrab and N. J. Sarhan. Aggregate Power Consumption Modeling of Live Video Streaming Systems. In Proc of MMSys, pages 60–71. ACM, 2013.

[3] M. Hosseini, J. Peters, and S. Shirmohammadi. Energy Budget Compliant Adaptive 3D Texture Streaming in Mobile Games. In Proc of MMSys, pages 1–11, 2013.

[4] M. Dong and L. Zhong. Self Constructive High Rate System Energy Modeling for Battery Powered Mobile Systems. In Proc of MobiSys, pages 335–348. ACM, 2011.

[5] F. Xu, Y. Liu, Q. Li, and Y. Zhang. V edge: Fast Self Constructive Power Modeling of Smartphones Based on Battery Voltage Dynamics. In Proc of NDSI, pages 43–55, 2013.

[6] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In Proc of CODES+ISSS, pages 105–114, 2010.