EE542
Internet and Cloud Computing
Lecture 21 – Estimating Power with ML

Young Cho
Department of Electrical Engineering
University of Southern California

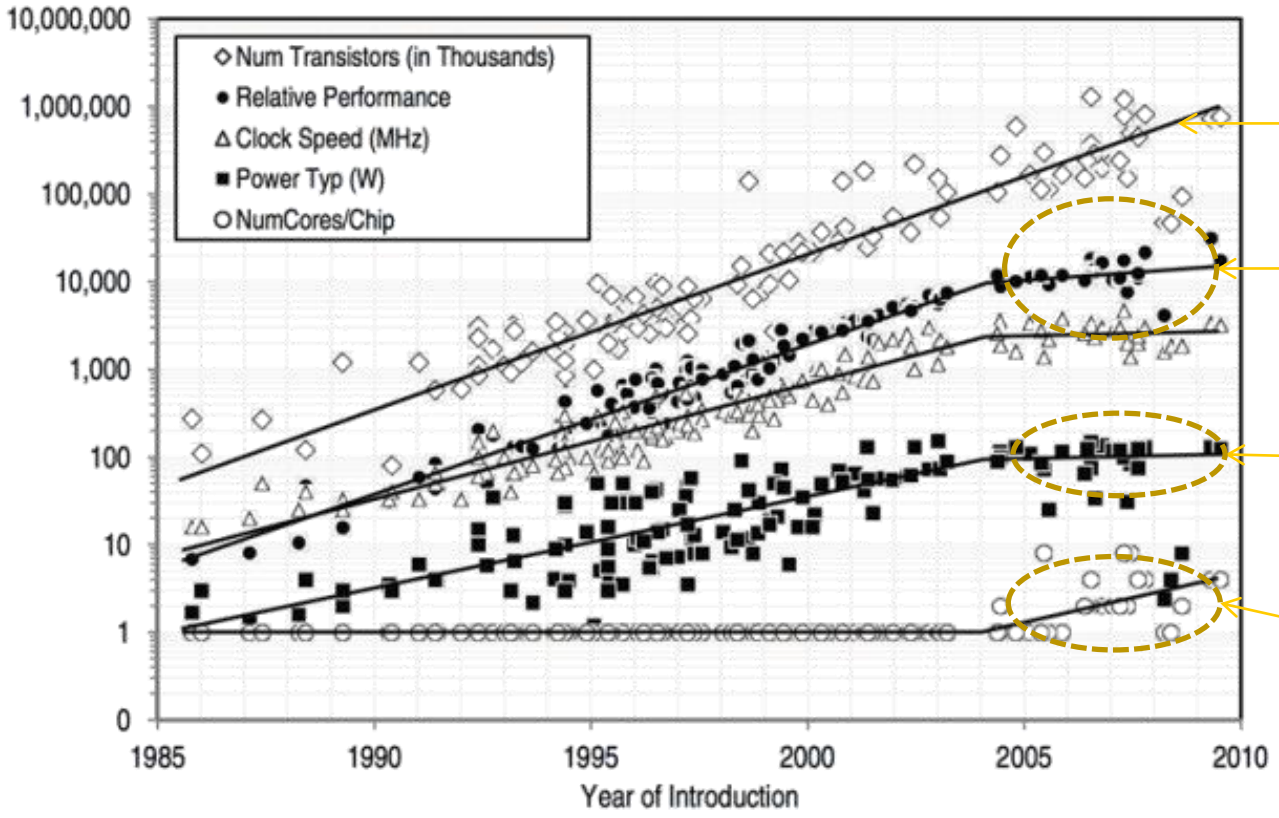
Final Project Progress Demos

- Due Every Mondays 11:59 PM in Nov
- Must Demonstrate Working System
- Incremental Functions Added with Corresponding Results
- System Must Work Every Demo
- Every Team Member Must Contribute in Describing the Demo

Final Project Progress Presentations

- November 18
  - Submit Practice Presentation Slides
- November 20
  - Go over the Practice Presentations
- November 25
  - Submit Final Project Progress Presentation Slides
- November 27, 29
  - Final Project Progress Presentation
  - 7 teams on 27th and 6 teams on 29th
  - Mandatory Attendance
- December 13
  - Final Project Demo Presentation Video

The History



LOGIC DENSITY

PERFORMANCE/POWER

POWER

PROCESSOR CORES

Source: ITRS

The Problem

Voltage scaling as an effective weapon against power reaching its limits !

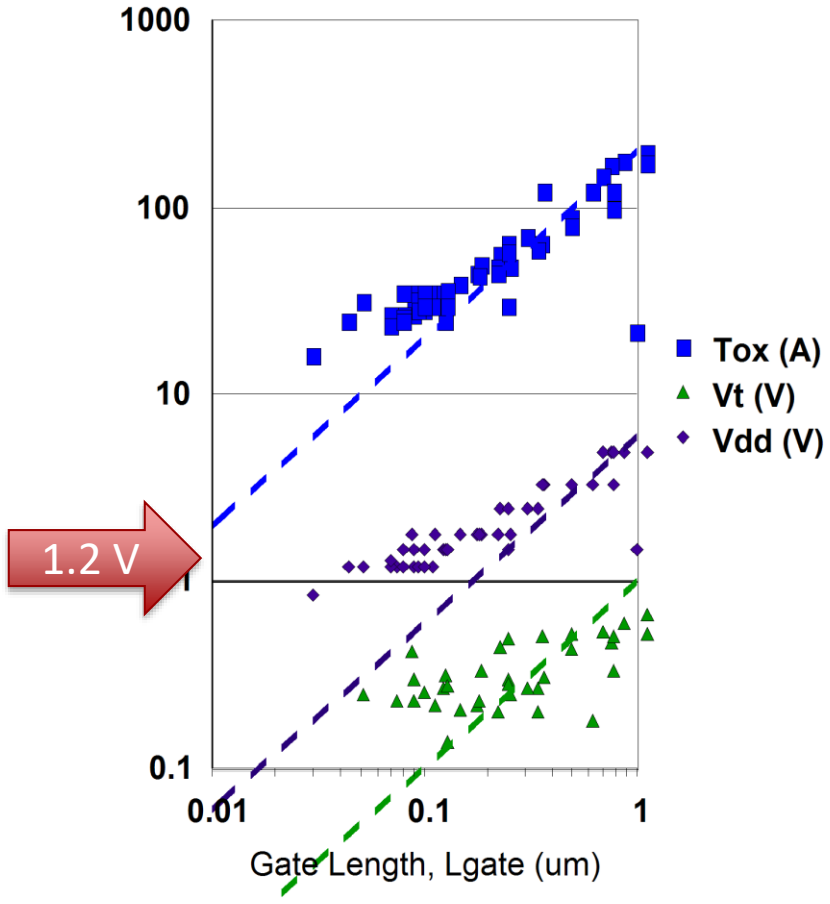
- CMOS at 45 & 22 nm nodes run at 1.2V.

Increased power per unit area

- Thermal Hot spots
- Inefficient Cooling systems

Other Alternative is, **Supply voltage gating**

- Gains diminished by inaccuracies in measurements

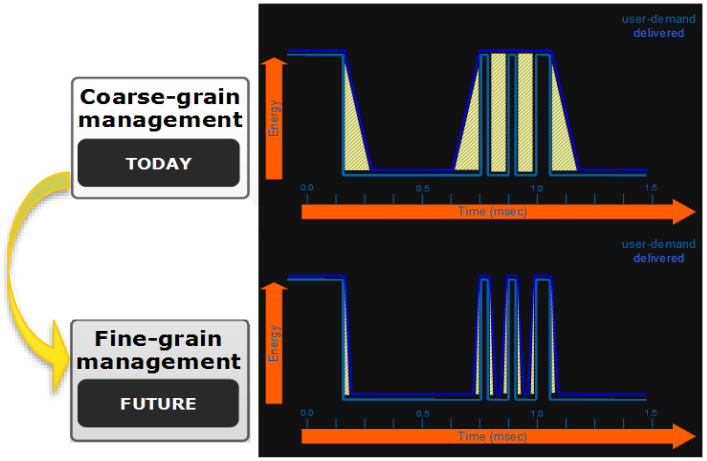


Source: Intel & IBM

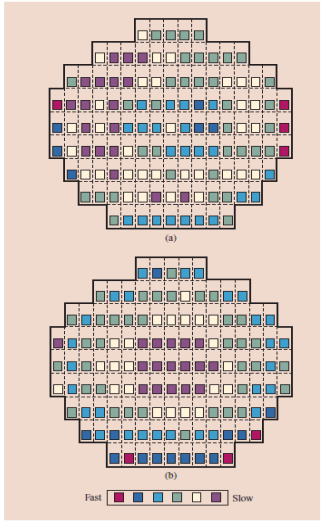
Current Solutions

Current in-situ methods not efficient, cost-effective or scalable

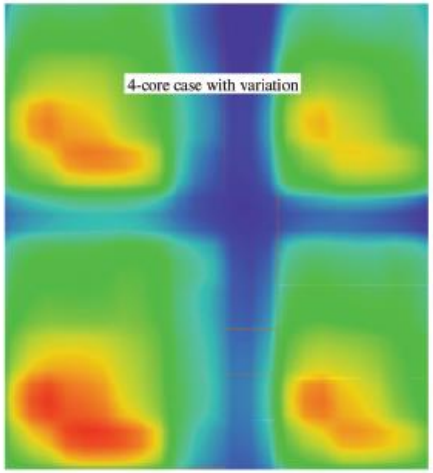
- Mostly analog or mixed signal (high power, area, cost, and poor scalability)
- MIT's on-chip method yields up to 60% power savings, but comes at a huge silicon area cost
- No or Minimal Accounting of PVT Variations
- Result: Less accurate and worse with increasing PVT variations



Reference : V.De , "Fine grain Power management", Forum talk, ISSCC 2013

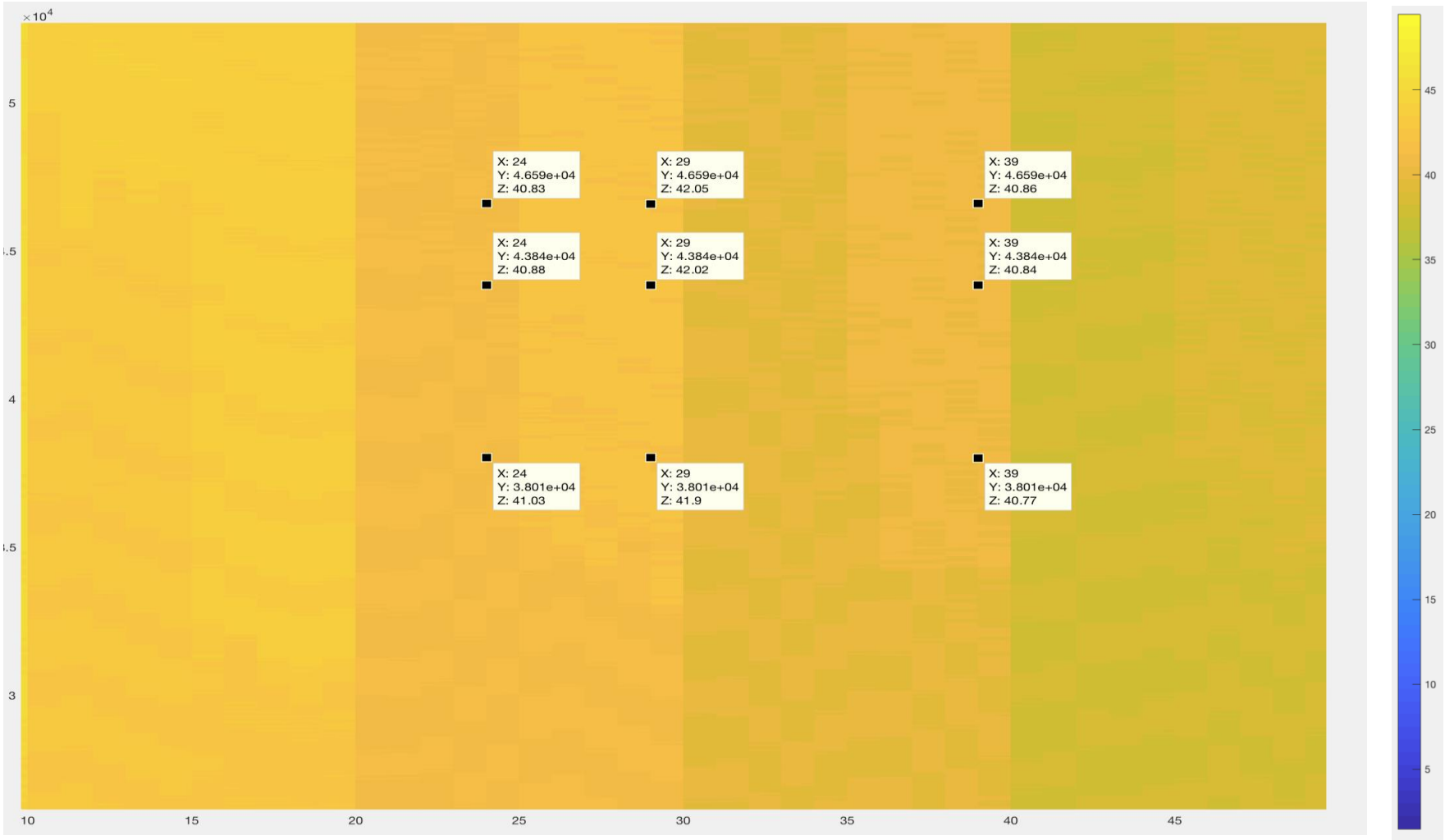


PVT variations



Temperature variations due to packaging

GTX 1070 – 15 Core GPU



Enable 12 best cores

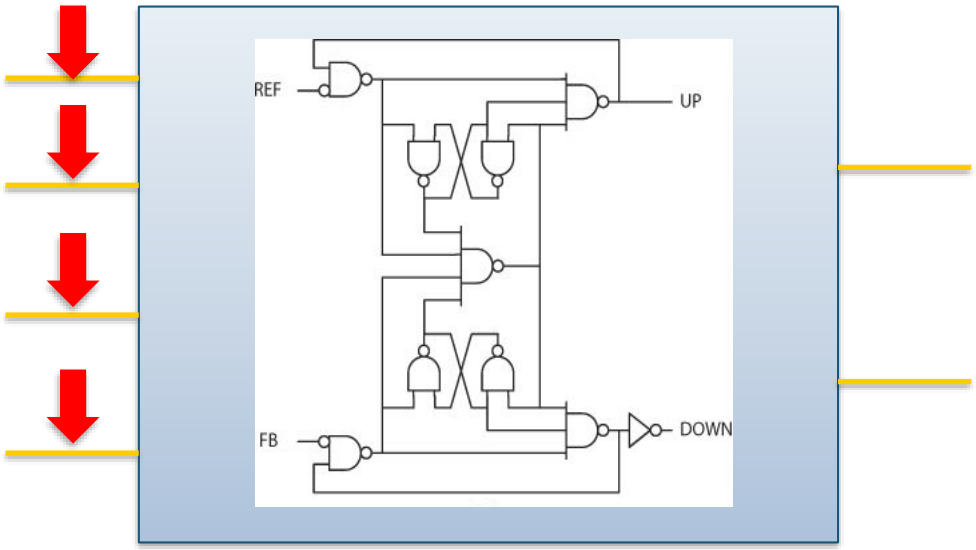
Enable 12 worst cores

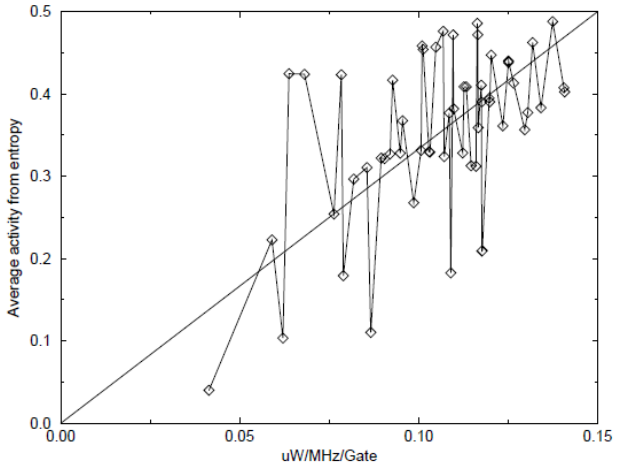
Enable 11 worst cores

Our Work

- Combine Direct measurements and fundamental theory of estimation
- Online linear solver to partition power
- Regressively extract the best model for the device by accurate model collapsing per chip basis.

Background





Comparison with zero-delay power/area simulation results.

$$P_{dyn} = \alpha \,.\, \text{Act}$$

References

- M. Nemani and F. Najm, “Toward a high-level power estimation capability,” IEEE Trans. Computer-Aided Design, vol. 15, no. 6, pp. 588–598, 1996.
- P. E. Landman, J. M. Rabaey, ” Architectural Power Analysis: The Dual Bit Type Method,“ IEEE Transactions on VLSI ,Vol 4 pp. 173-187 , June 1995.

Theory

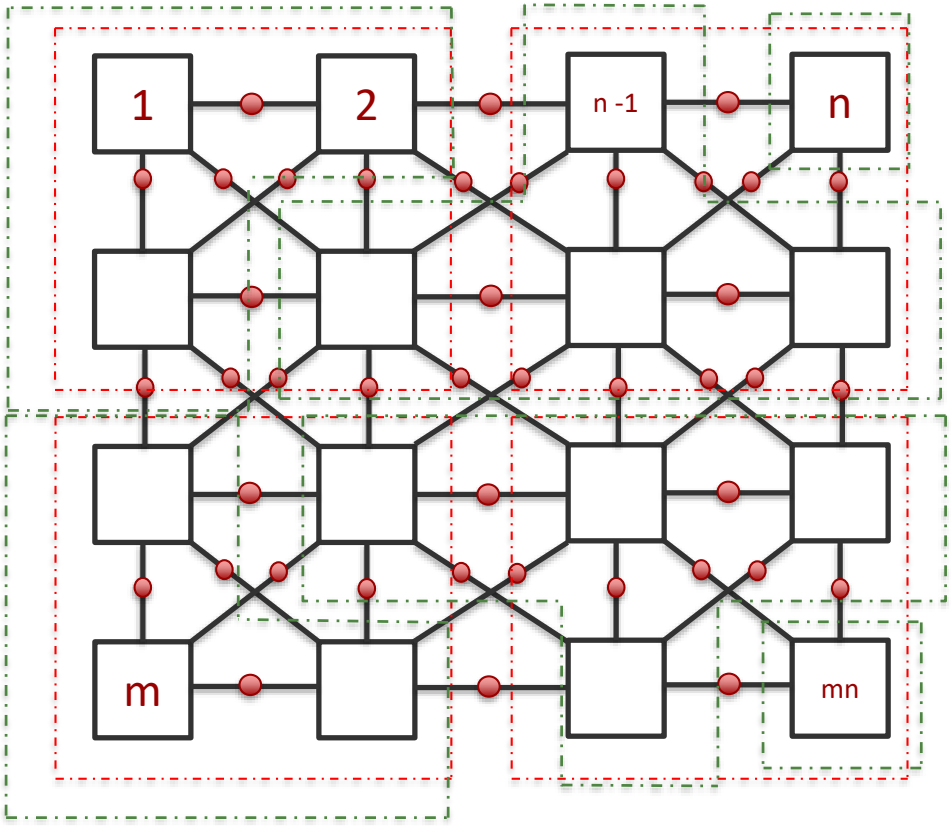
Component C consisting of n wires . Each wire driving a capacitive load of C_i

Signal Transitions at each line is represented as tr_i and Supply voltage as V_{DD}

$$P_{nom} = \sum_{i=1}^{n}(tr_i \,.\, c_i . V_{DD}^2) = \sum_{i=1}^{n}(tr_i \,.\, W_i)$$

$$|\boldsymbol{P_{nom}} - \boldsymbol{P_{est}}| = \boldsymbol{\varepsilon}$$

$\boldsymbol{\varepsilon} = 0$, *Ideal Conditions*
$\boldsymbol{\varepsilon} \neq 0$, *Under PVT variations and measurement noise*



$$P_{est,1} = \sum_{l=1}^{m}(tr_l \,.\, W_l)$$

$$P_{est,2} = \sum_{q=1}^{p}(tr_q \,.\, W_q)$$

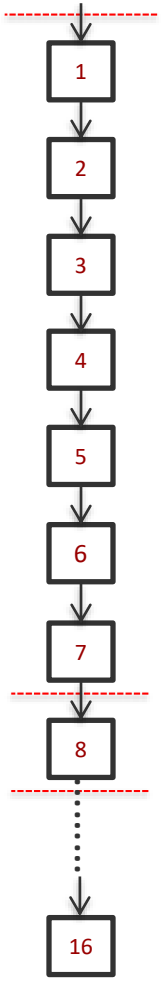
$$where, n > m > p$$

$$|\boldsymbol{P_{nom}} - \boldsymbol{P_{est,1}}| = \boldsymbol{\varepsilon_1}$$

$$|\boldsymbol{P_{nom}} - \boldsymbol{P_{est,2}}| = \boldsymbol{\varepsilon_2}$$

$$\boldsymbol{\varepsilon_2} \geq \boldsymbol{\varepsilon_1}$$

Evaluating logic cuts to extract desired instrumentation point



$P_{nom} = \sum_{i=1}^{n}(tr_i \,.\, W_i)$, Where n = all lines in C

$P_{est,1} = \sum_{p=m}(tr_p \,.\, W_p)$, Where m = inputs of C.

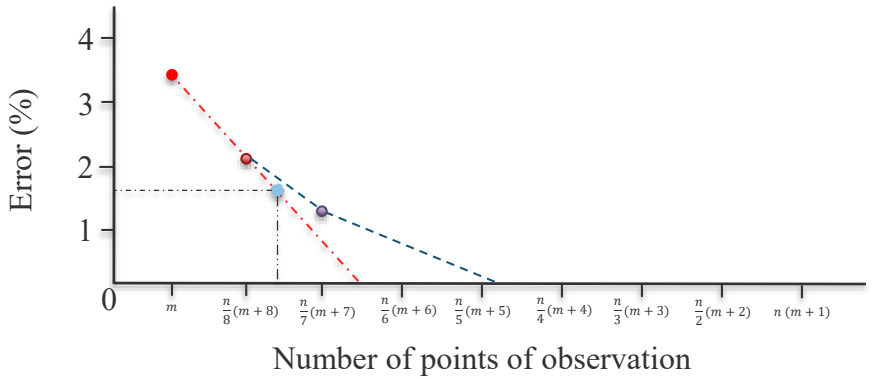
$\left|P_{nom} - P_{est,1}\right| = \varepsilon_1$

$if \;\; \max(\varepsilon_1) > \mathrm{E}$,

$f(x) = \cup_{k=1}^{j} V_k$, Such that $f(V_k) = f(V_{k+1}) \;\; \forall \; k+1 \leq j$

$P_{est,2} = \sum_{p=a}(tr_p \,.\, W_p)$, Where a = inputs of cuts.

$\left|P_{nom} - P_{est,2}\right| = \varepsilon_2$



Using Online Solver for Weights

Counter values	Power Values	Residue
28 1 0 3 1 1 2 26 1	0.3063	0
3 0 1 3 1 0 2 3 1	0.0717	0
3 0 0 2 0 0 2 3 1	0.0078	0
5 1 0 5 1 1 2 5 1	0.1025	0
5 0 1 5 1 0 2 5 1	0.0933	0
5 0 0 4 0 0 2 5 1	0.0039	0
7 1 0 7 1 1 2 7 1	0.1265	0
7 0 1 7 1 0 2 7 1	0.1429	0
7 0 0 6 0 0 2 7 1	0.0973	8.05E-07
9 1 0 9 1 1 2 9 1	0.1409	9.01E-07
9 0 1 9 1 0 2 9 1	0.1453	9.02E-07
9 0 0 8 0 0 2 9 1	0.0179	9.02E-07
11 1 0 11 1 1 2 11 1	0.1747	9.16E-07
10 0 1 10 1 0 2 10 1	0.0012	9.18E-07
11 0 0 11 0 0 2 11 1	0.1695	9.23E-07
13 1 0 13 1 1 2 13 1	0.2305	9.24E-07
13 0 1 13 1 0 2 13 1	0.2165	9.78E-07
12 0 0 12 0 0 2 12 1	0.0039	9.88E-07
15 1 0 15 1 1 2 15 1	0.2233	9.08E-07
15 0 0 15 0 1 2 15 1	0.2683	9.18E-07
16 0 0 17 1 0 2 16 1	0.2347	9.8E-07

W1	W2	W3	W4	W5	W6	W7	WS
0.000463369	0.000203485	0.000208883	3.94E-05	0.000166	0.000171	5.54E-05	0.000883
0.000924628	0.000317377	0.00023828	0.000078	0.00023	0.00028	7.93E-05	0.001344
0.000917426	0.00031614	0.000237881	7.74E-05	0.00023	0.00028	7.91E-05	0.00135
0.000921373	0.000317757	0.00023748	7.78E-05	0.00023	0.000279	7.94E-05	0.001346
0.000957923	0.000330416	0.000240154	8.08E-05	0.000234	0.000291	8.16E-05	0.001396
0.000957177	0.000330682	0.000240756	8.07E-05	0.000234	0.000291	8.15E-05	0.001397
0.000964848	0.00033605	0.000241628	8.14E-05	0.000237	0.000295	8.24E-05	0.001415
0.000963136	0.000335876	0.000242144	8.13E-05	0.000237	0.000295	8.22E-05	0.001417
0.001044859	0.00037248	2.51E-04	8.83E-05	0.000253	3.24E-04	8.69E-05	0.001543
0.001045028	0.000372572	2.51E-04	8.83E-05	0.000253	3.24E-04	8.69E-05	0.001543
0.001044964	0.000372552	2.51E-04	8.83E-05	0.000253	3.24E-04	8.69E-05	0.001543
0.001045086	0.000372471	2.51E-04	8.83E-05	0.000253	3.24E-04	8.69E-05	0.001543
0.001045088	0.000372474	2.51E-04	8.83E-05	0.000253	3.24E-04	8.69E-05	0.001543

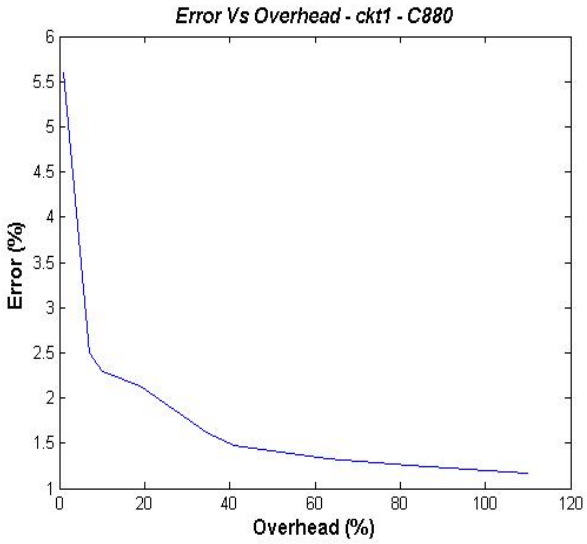
$$P_{total,t} = \sum_{i=1}^{k} (a_i .\ P_{dyn}(I_i)) + P_{St}(\sum_{i=1}^{k} I_i) + Residue$$

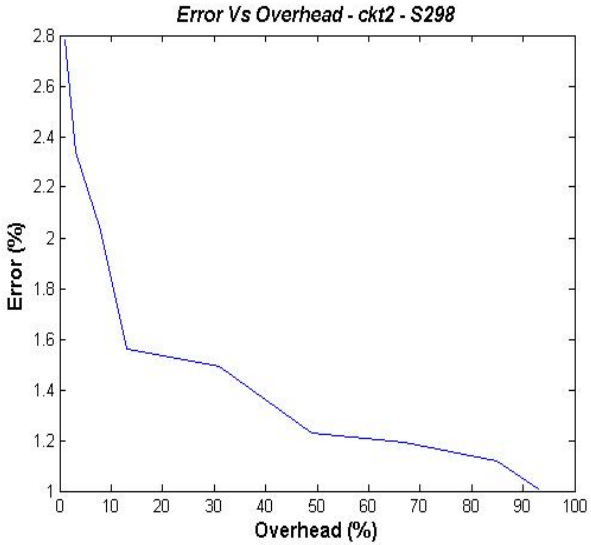
$$P_{Total} = (\sum_{j=1}^{n} W_j a_j) + (1 * W_S) + Residue$$

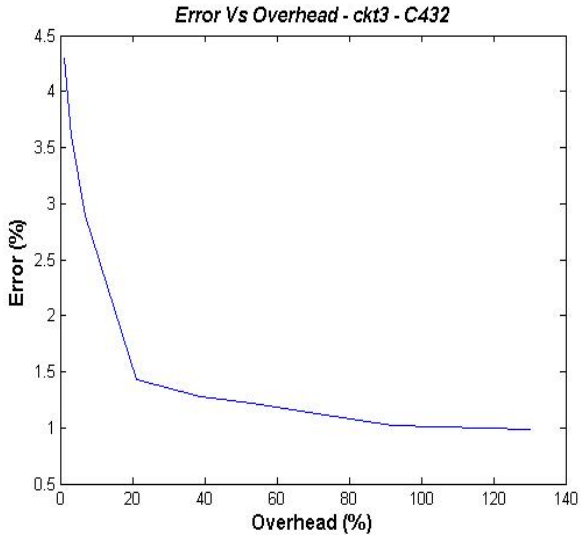
Generalized Recursive information form of estimation,

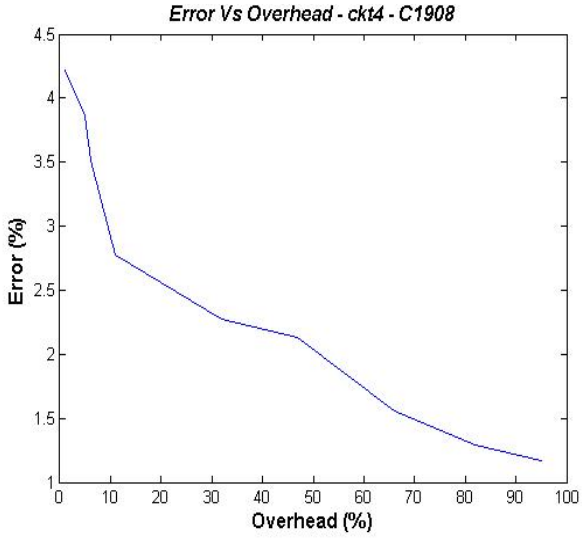
$$Z(k+1) = H(k+1).W + V(k+1)$$

Initial Results



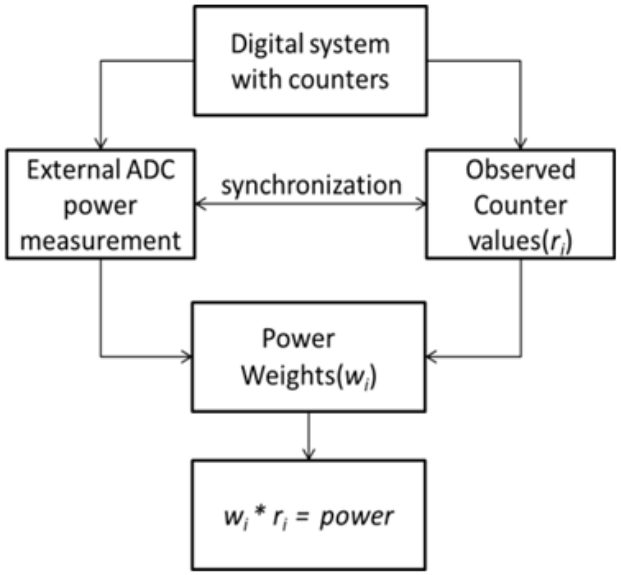




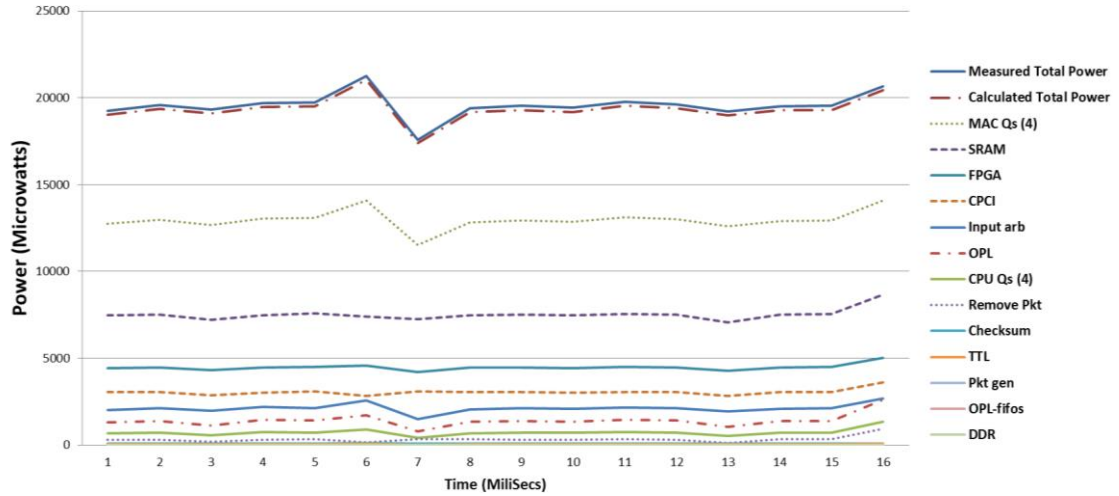


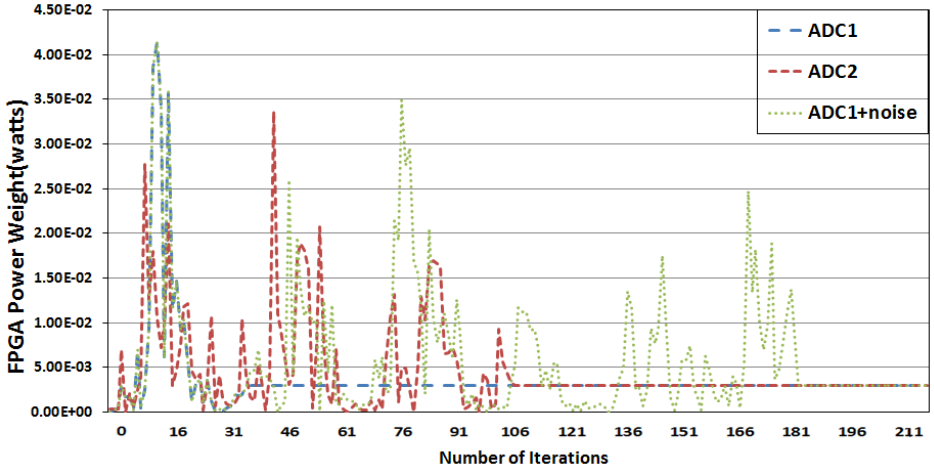
Reduction of methodology to practice

- NetFPGA : Open source hardware with all the networking appliances built in.
- The goal was to identify the power used by different subcomponents of the board. Eg: Eth Phy , SRAM, DDR



Initial Hardware Results





Maximum deviation less than 3% from ADC

Reduce Overhead without losing accuracy

Problem : How do I reduce the instrumentation overhead and yet maintain accuracy ?

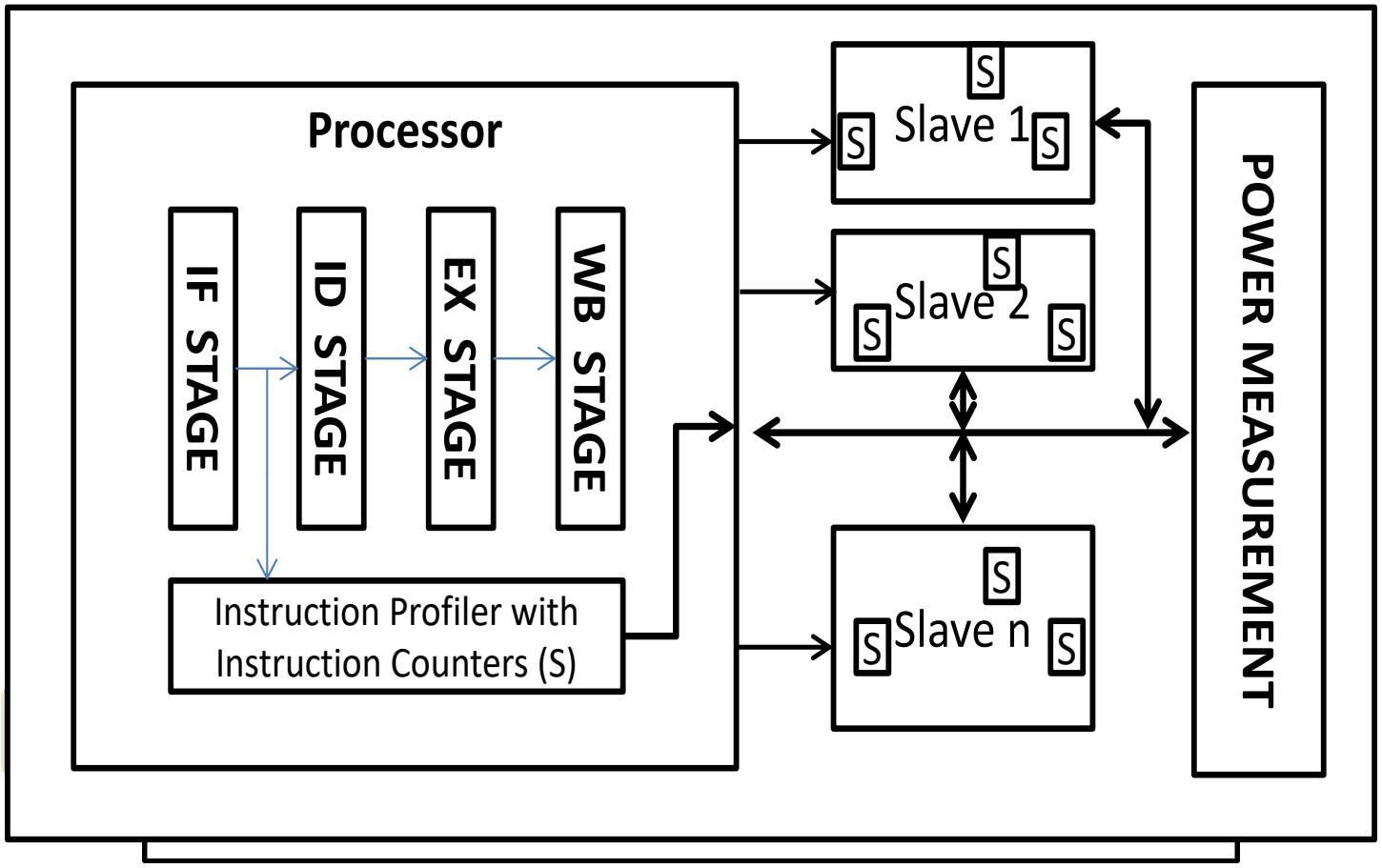
Solution :

- What is the proxy for activity within CPU ?

  Answer : Instructions

- Can I use these instructions to measure power accurately ?

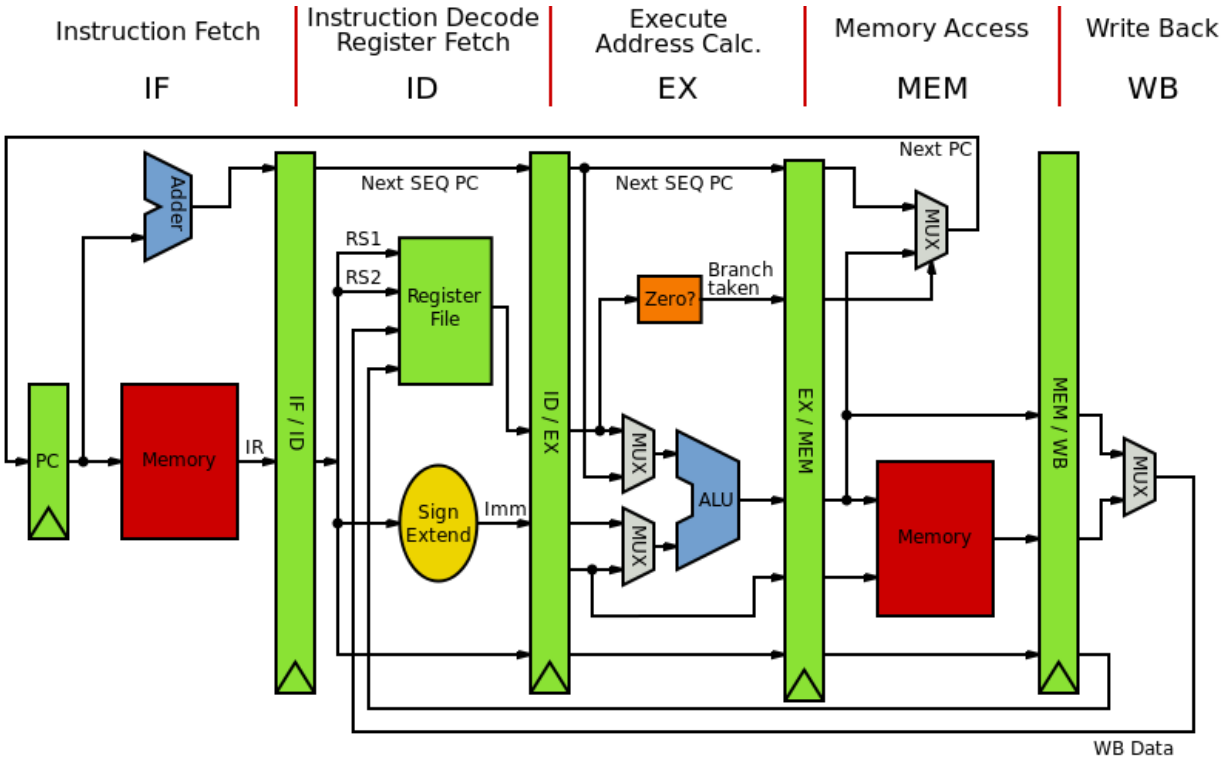
Accounting Activity of SoC



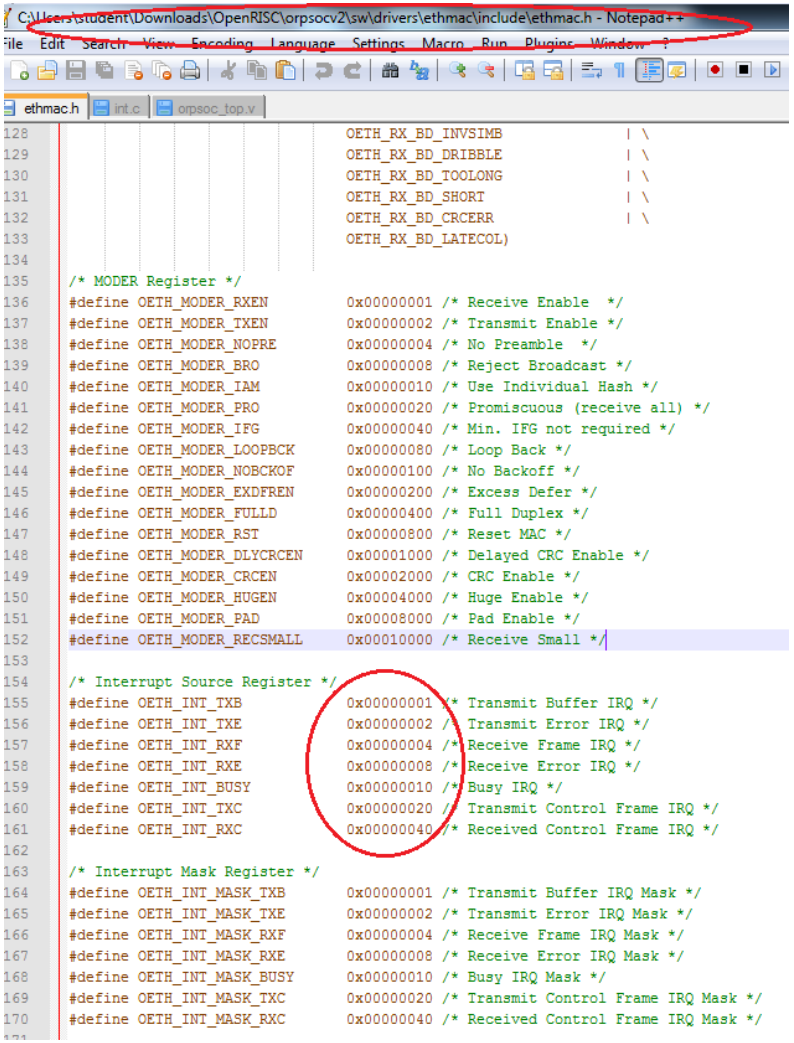
Summarizing the Activity as temporal and spatial activity
Summarizing The spatial activity

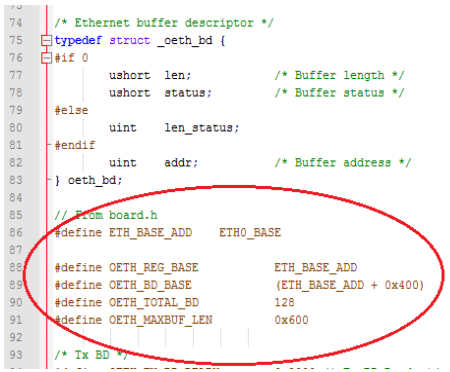
Online Estimation of Energy Per Instruction of a processor

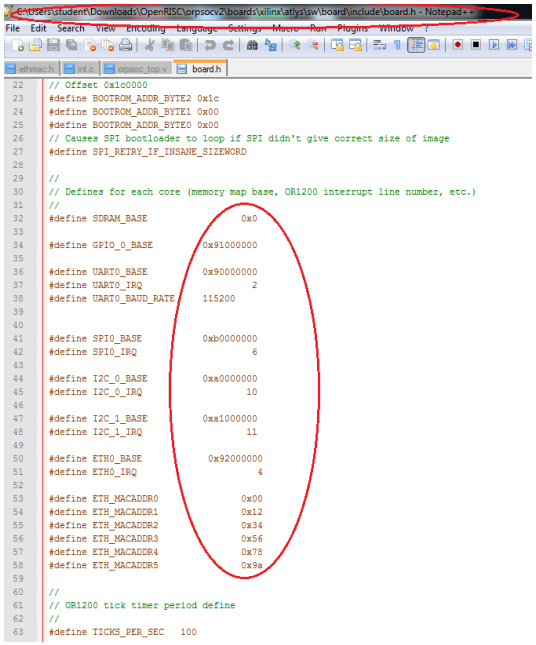
- Weights represent contribution of instructions to power.



Integrating Peripheral Sensors of an SoC into Instruction profiler

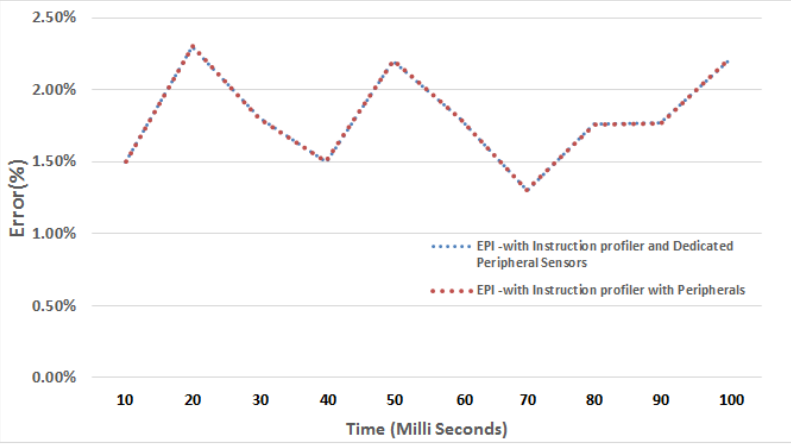


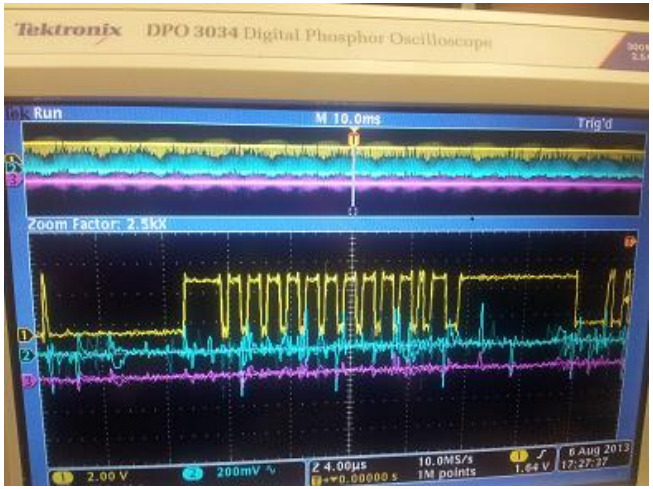




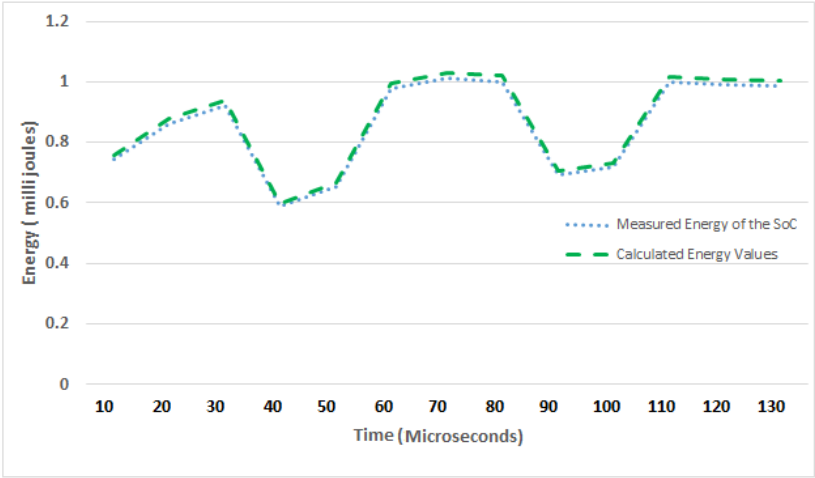
- Peripherals are accessed by load or stores to a pre-defined memory address.
- The hardware Profiler matches these instructions and addresses to account for the peripherals.

Initial Results on EPI





- Using the partitioning tool , we needed 150% additional hardware. We now have reduced it to less than 1% of the total SoC



Drawbacks of EPI

- Instruction is conceptual and are not mapped into physical component.
  - Hard to use it to assist thermal management

Accurate Sub-circuit Physical level power measurement in Processors/SoC

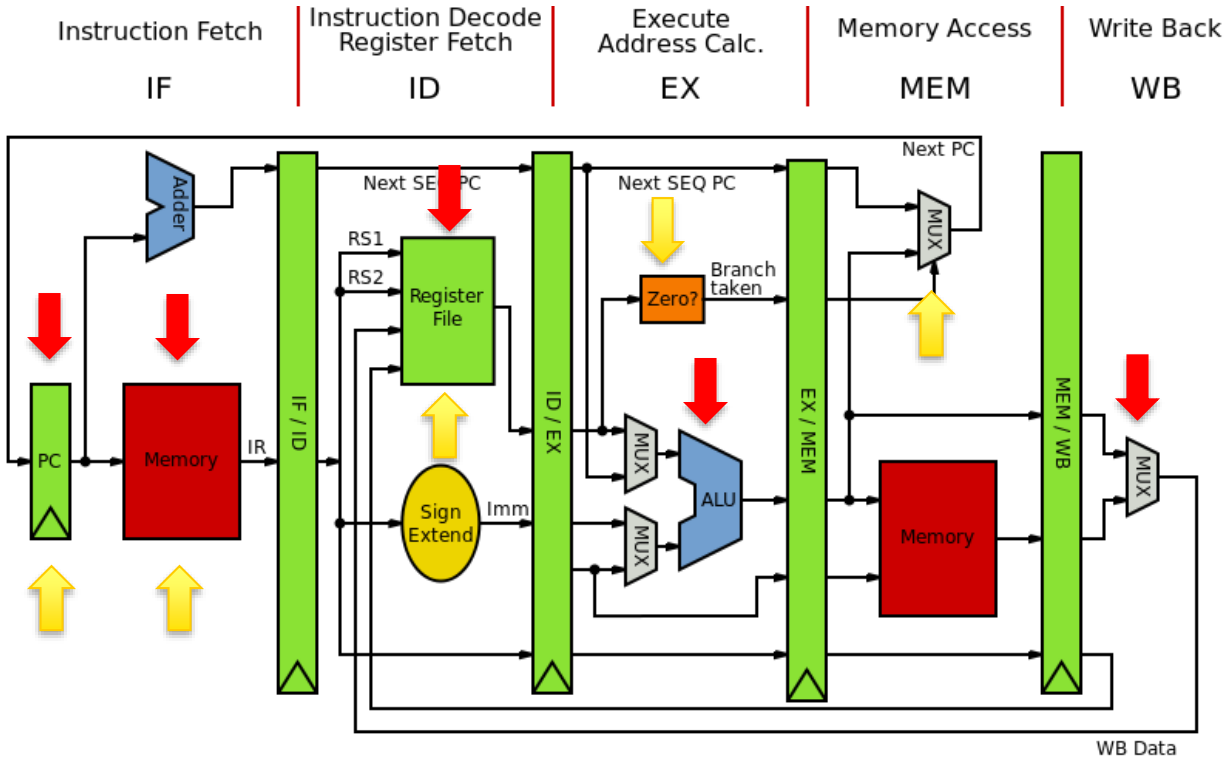
Problem : How do I measure physical level sub-component power accurately without instrumenting additional counters?

Solution :

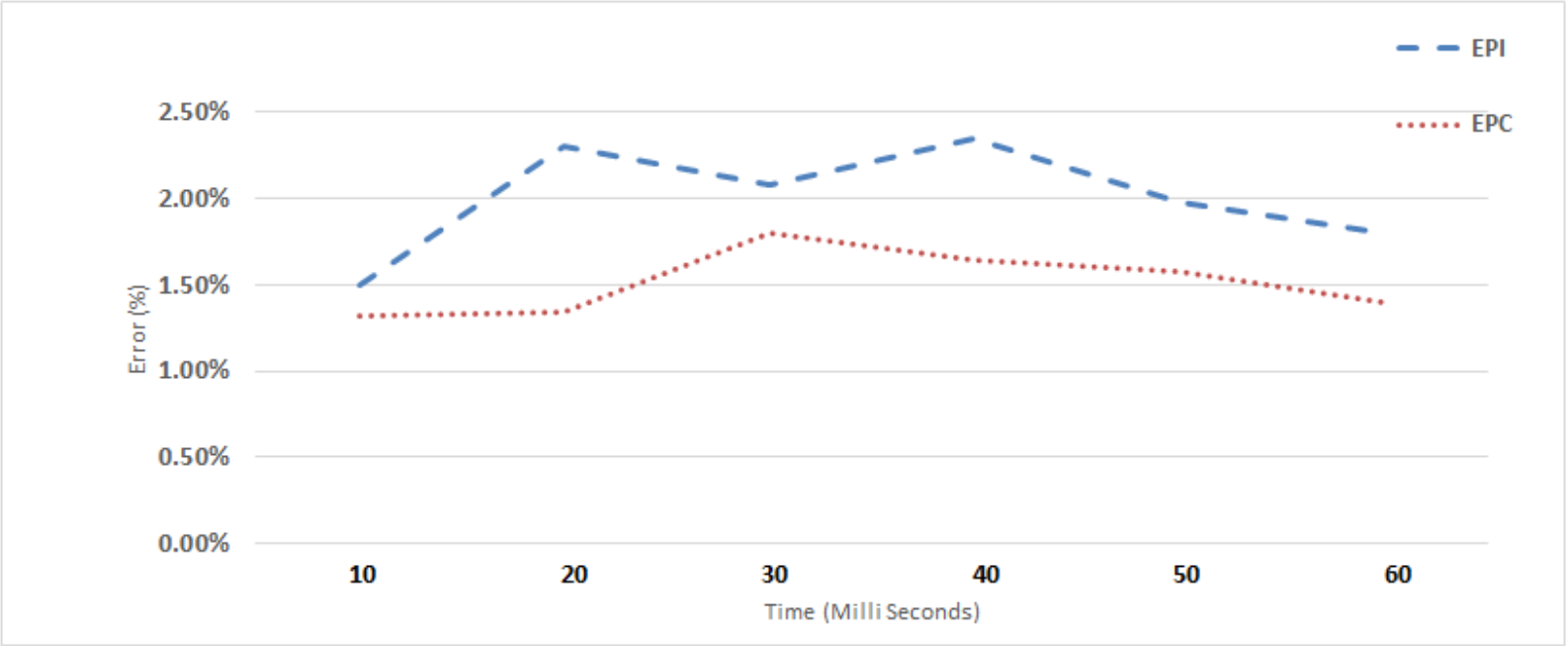
- We explored the possibility by looking into processor itself
- By doing three levels of manual mapping and we were able to map 80 instructions to 55 components.

Energy Per Component of a Processor

- Perform analysis on processor instruction execution paths.
  - Find the number of components at the lowest granularity that gives same accuracy.

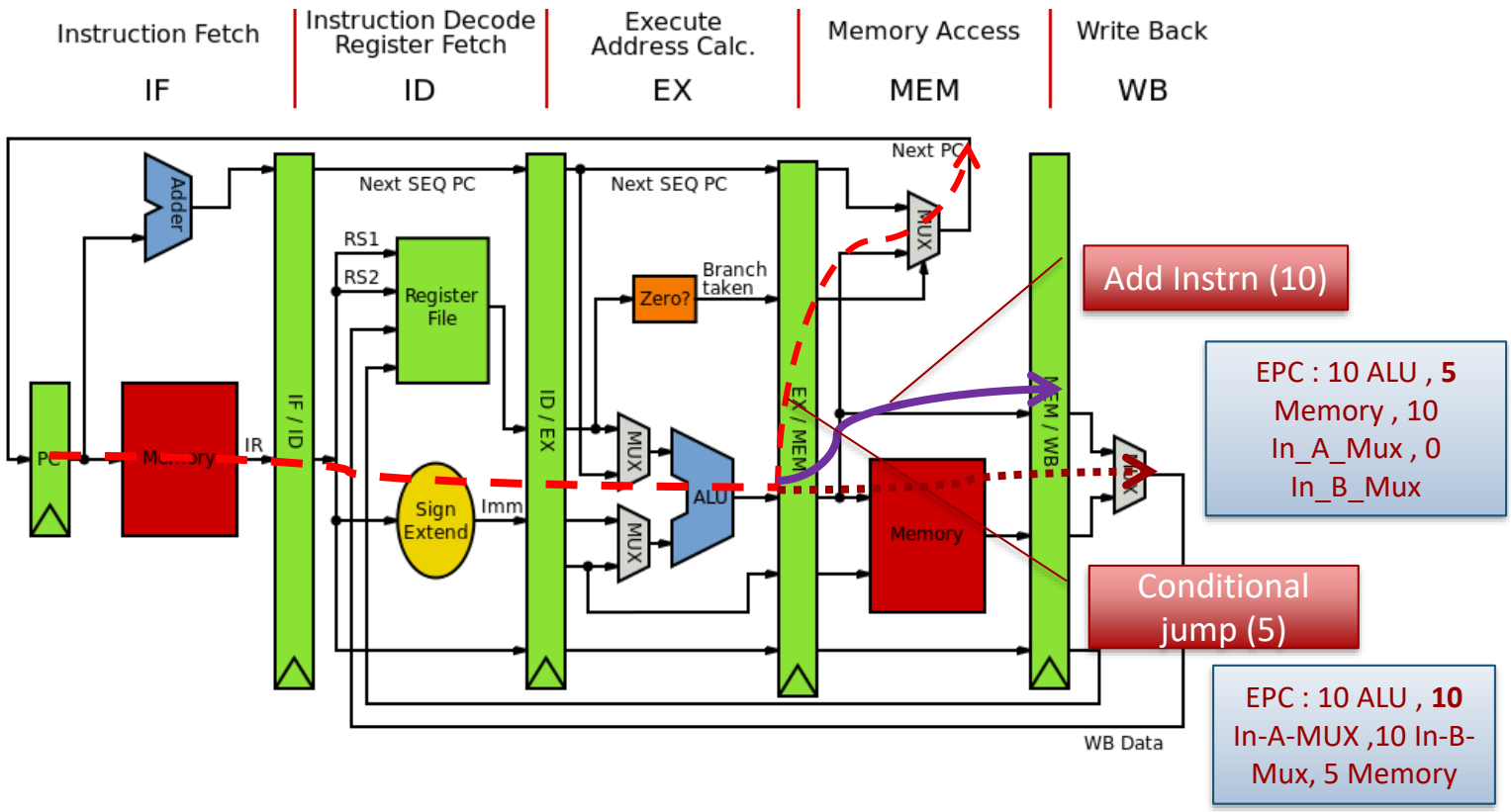


EPC results



- Reduced 80 instruction counters to 55 Component counters
- Replaced 70 dedicated peripheral counters with 12 counters in the instruction profiler

Comparison of EPI and EPC



- EPI Produces an average weights associated with switching at the mux and memory.
- EPC accounts for these switching – hence more accurate

Drawbacks of EPC

- It is we difficult to extract different layers of components for EPC for a very complex design.
- Instruction overlaps not considered.

Improving Accuracy of estimation without increasing overhead

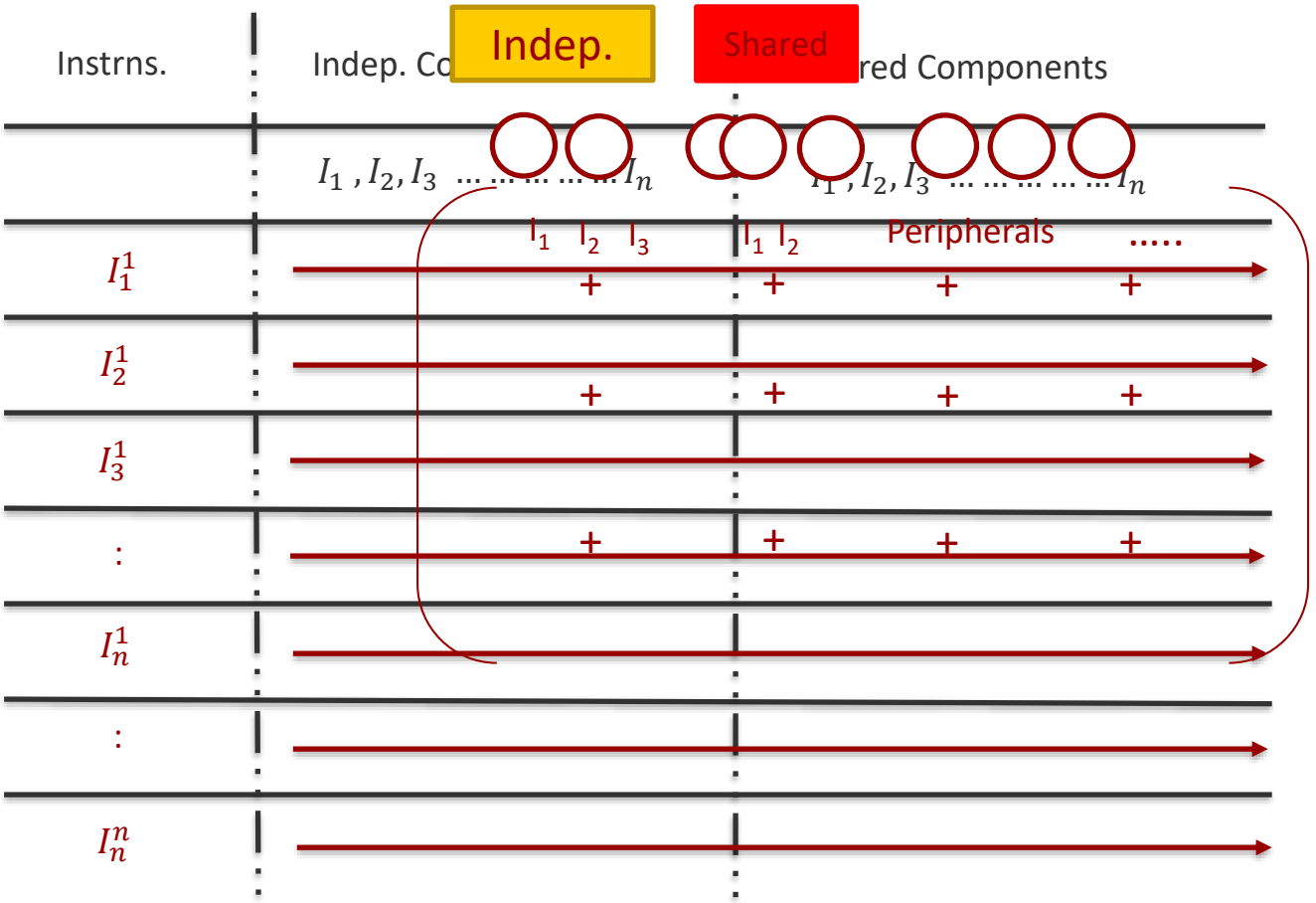
Problem : How do I improve the accuracy with the same overhead?

- How do I account for various execution overlaps ?

  Ans : Implicit enumeration of shared components

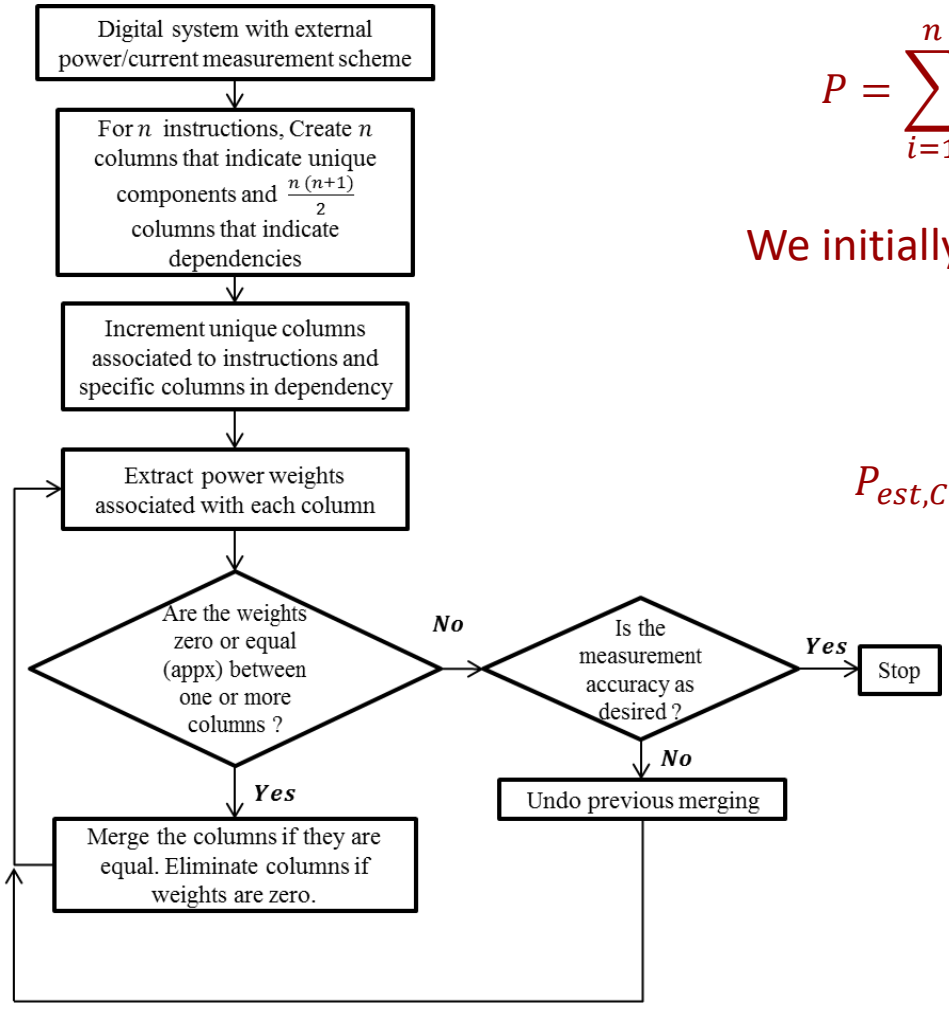
- Can I still use instruction counts to measure power accurately ?

Automated Energy per component – The concept



Increment counters such that ,
Independent counters =
Instruction Count and
Shared components =
Increment every time
associated instruction appears

Automated Energy per component



$$P = \sum_{i=1}^{n} a_i w_i + \sum_{j=1}^{\frac{n(n+1)}{2}} a_j w_j + \sum_{k=1}^{m} a_k w_k + P_{static} + R$$

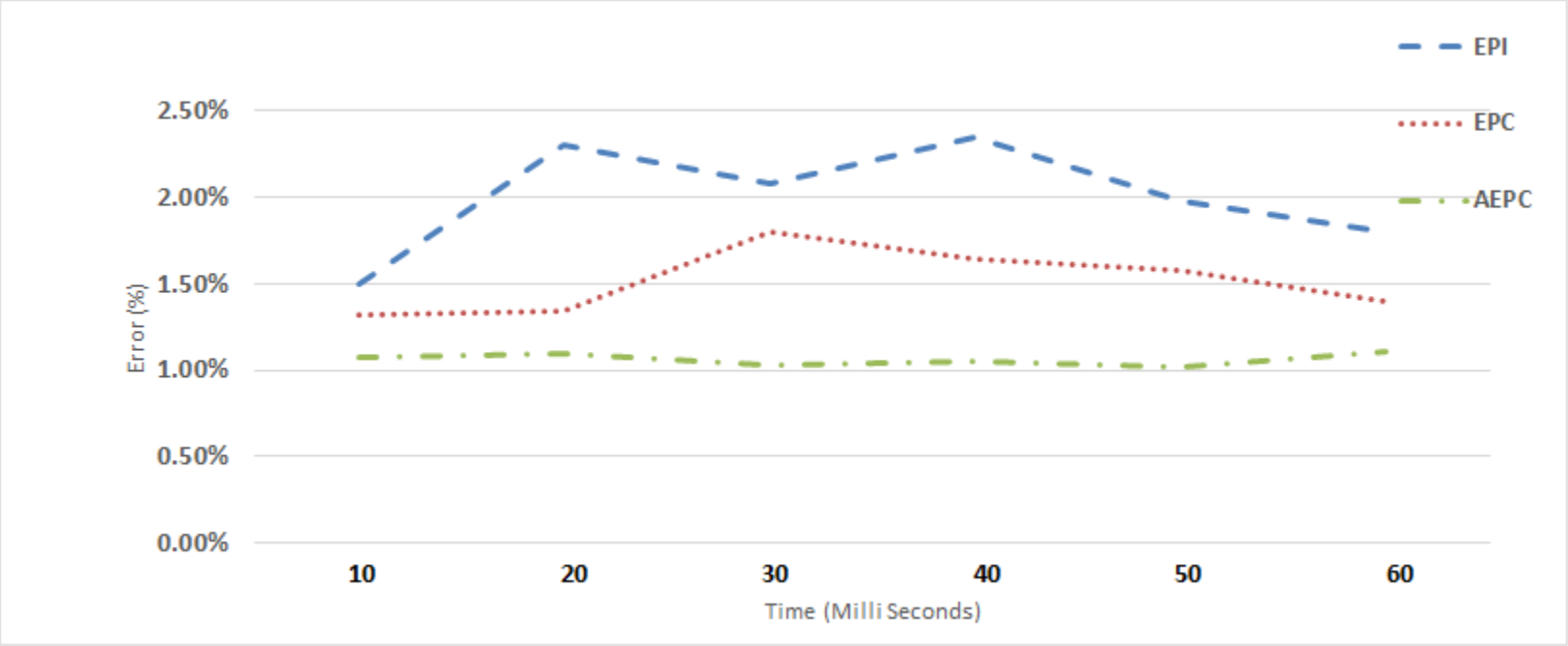
We initially assume that , $\exists\ C_z$, $such\ that$, $C_z \in I_x \cap I_y$

$$P_{est,C1} = \sum_{p=u} (tr_p \,.\, W_p)$$

- = 0 , C_z does not exist (W = 0)
- = $P_{est,Cv}$, *if Cz is equally shared between multiple instructions*
- Unique

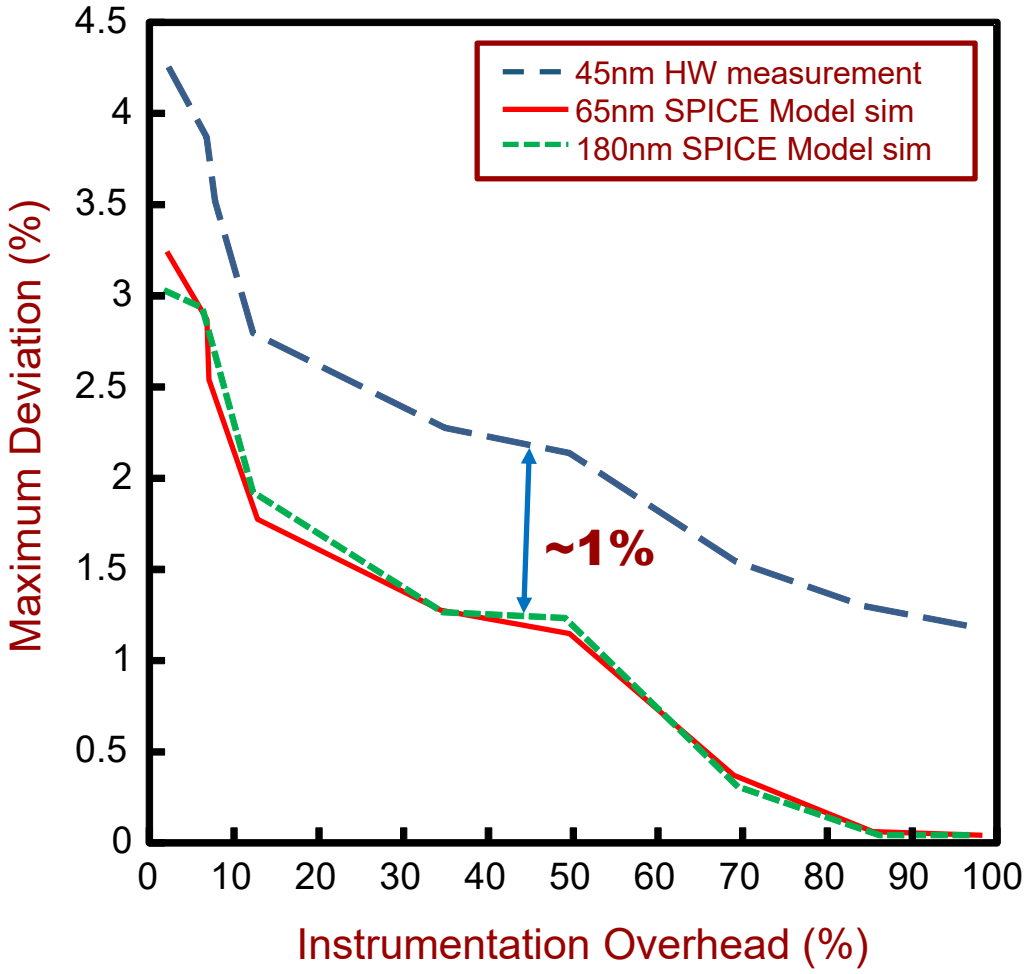
The weights are means to perform implicit enumeration of component sharing.

AEPC Results



We compare the weights of components from EPC than those of AEPC.
(based on the sharing information in AEPC).
For ALU and register file , the difference between power estimated by AEPC and EPC < 0.5%

Comparison of simulator and hardware results



Accelerated Accurate Simulator

Circuit	Number of Gates	Comm. SPICE (ref)		Comm. Accelerated. SPICE			Our method		
		Simulation Time (s)	Memory (MB)	Simulation Time (s)	Memory (MB)	Accuracy (%)	Simulation Time (s)	Memory (MB)	Accuracy (%)
C432	163	20412.69	927.24	2886.72	274.93	99.971	24.59	7.90	99.999
S298	240	23790.45	1082.26	3364.45	320.42	99.971	30.07	8.20	99.999
C880	383	14760.83	670.90	2087.52	199.81	99.971	11.82	6.07	99.999
C1908	2092	172808.22	7904.09	24439.01	2326.92	98.150	286.46	32.00	99.999
Microctr.	50000	310320.00	14000.00	43887.00	4000.00	98.003	9092.40	49.00	99.999

Results on Simulation of various benchmark circuits with Comm. SPICE, Accl. SPICE and our method

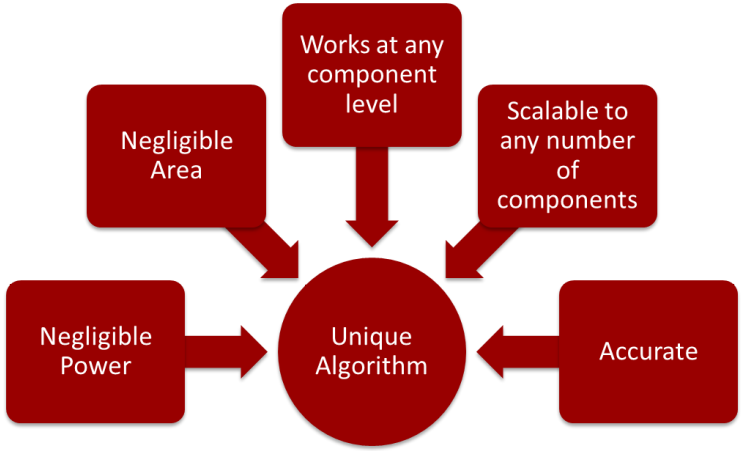
277 x over reference for the same accuracy

Accuracy (%)	Raw Convergence				Convergence with seed value				Speedup in conv. using Accl. Sim for training		Comparison of Weights from Comm. SPICE and Accl. SPICE
	Comm. SPICE		Accl. Sim		Comm. SPICE		Accl. Sim		Raw	Seeded	
99.999	Samples	Time (mins)	Samples	Time (mins)	Samples	Time (mins)	Samples	Time (mins)			Same Values
	19055	953.50	19070	121.19	66	3.12	102	0.59	7.88x	5.29x	
99.7	4120	210.00	4157	30.35	63	3.10	89	0.53	6.92x	5.85x	Same Values
98	2230	122.00	2283	18.01	55	3.02	71	0.47	6.77x	6.43x	Same Values
97	1775	69.00	1810	14.51	43	1.56	57	0.38	4.76x	4.10x	Same Values

Results on rate of convergence with Comm. SPICE, Accl. SPICE and our method

Summary

Goal : To develop an accurate method to measure the power consumption of components on-chip which is fine-grained, and low overhead.



Method : By observing a summary of internal circuit activity along with our online solver and extract the best model for the device by accurate model collapsing

Result : A Power measurement system that is :

- Accurate
- Fine granularity
- Low overhead