国防科学技术大学
National University of Defense Technology

AMD

# PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration

Bo Su[†] Junli Gu[‡] Li Shen[†] Wei Huang[‡] Joseph L. Greathouse[‡] Zhiying Wang[†]

[†]State Key Laboratory of High Performance Computing
National University of Defense Technology

[‡]AMD Research
Advanced Micro Devices, Inc.

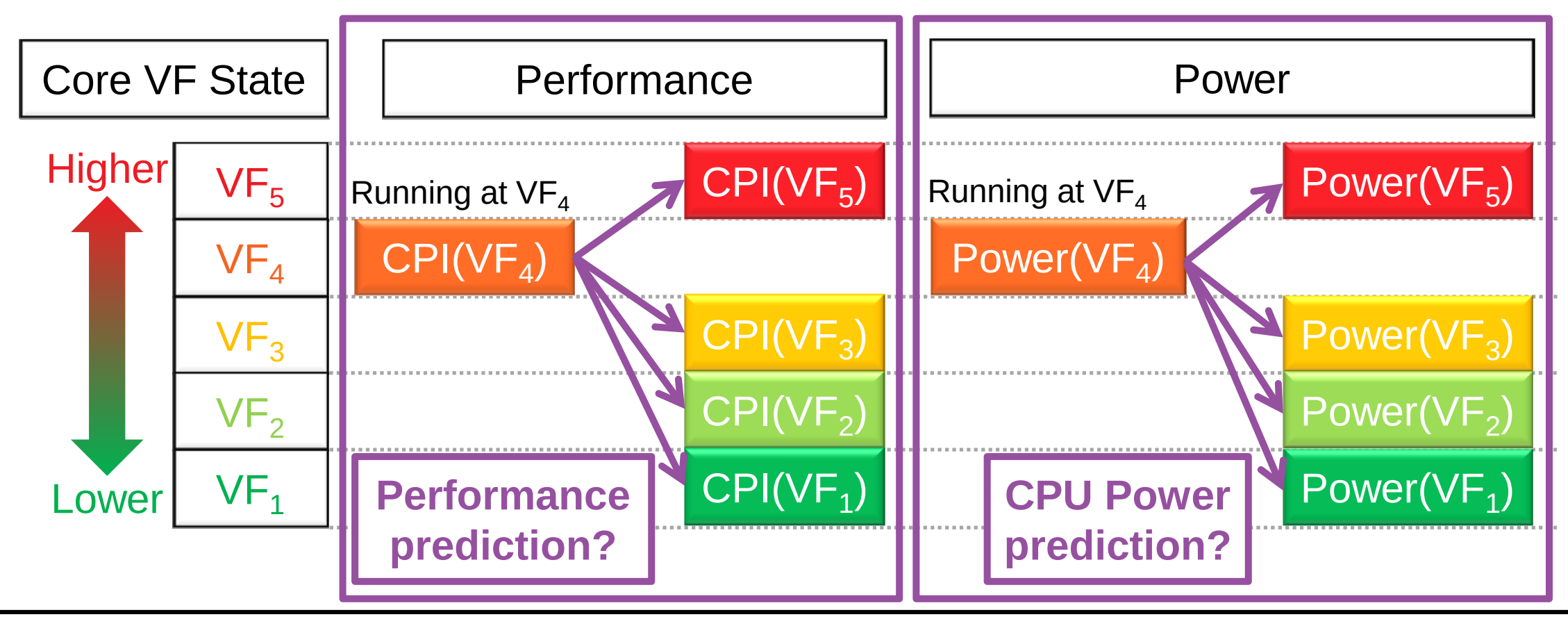
## 1. Problems

- ◆ Dynamic Voltage & Frequency Scaling

  Widely used to boost performance, lower power, and improve energy efficiency.
- ◆ Challenge

  How to predict performance and power across DVFS states?
- ◆ Difficulties brought by modern processors

  Multiple clock domains + Multiple power planes



Core VF State

Higher

Lower

$VF_5$ $VF_4$ $VF_3$ $VF_2$ $VF_1$

Performance

Running at $VF_4$

$CPI(VF_4)$ → $CPI(VF_5)$, $CPI(VF_3)$, $CPI(VF_2)$, $CPI(VF_1)$

**Performance prediction?**

Power

Running at $VF_4$

$Power(VF_4)$ → $Power(VF_5)$, $Power(VF_3)$, $Power(VF_2)$, $Power(VF_1)$

**CPU Power prediction?**

Our work:

Take PMC measurements at one VF state.

↓

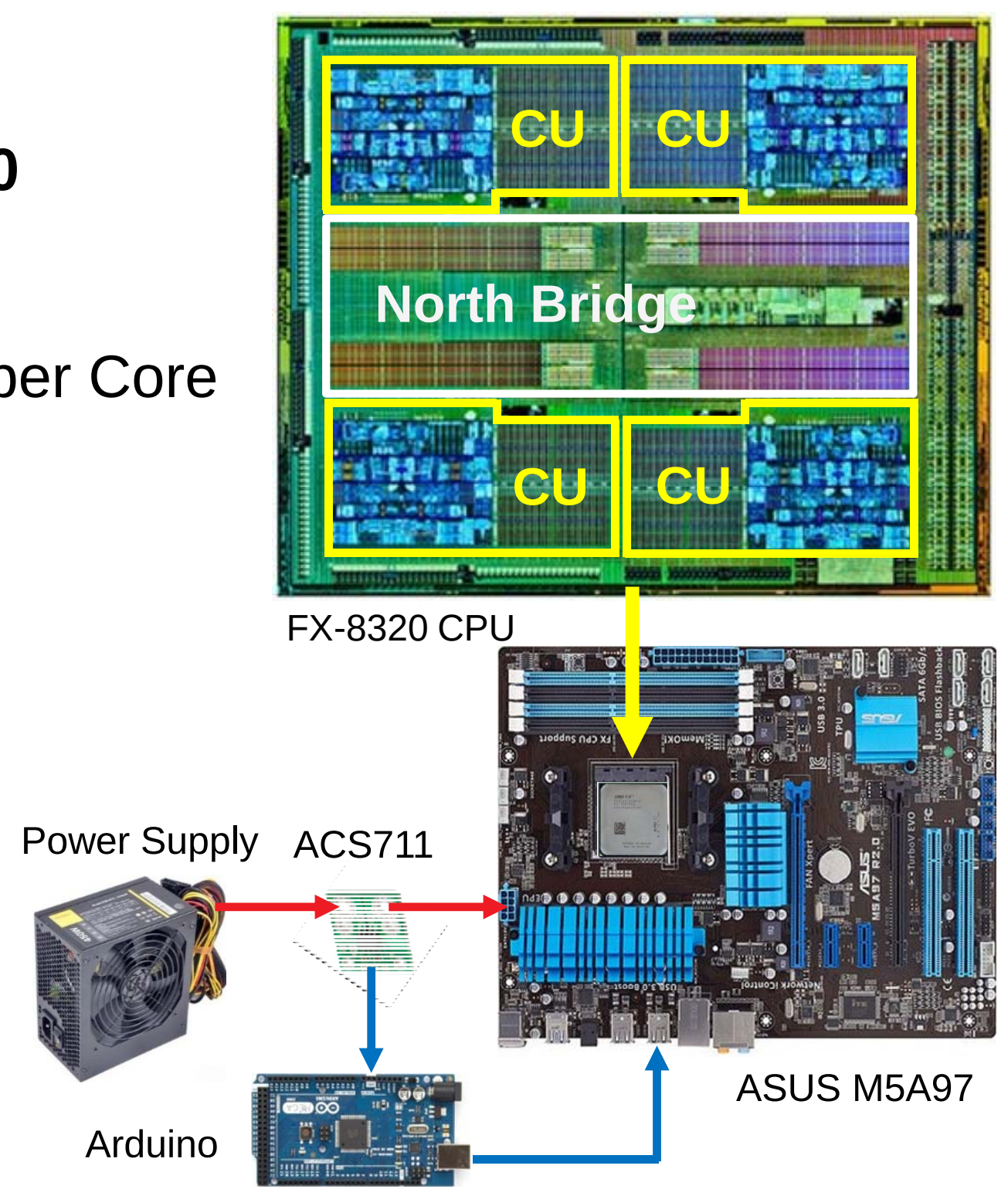
Predict performance and power of other VF states.

## 2. Platform and Tools

### HW Platform

**Processor: AMD FX-8320**
- 4 Compute Units (CU)
  - 2 Cores in one CU
  - PMCs: 6 Counters per Core
  - VF States: 5 States
- North bridge (NB)
  - L3 Cache
  - Memory Controller
  - Shared by all CUs

**Power Measurement**
- Pololu ACS711
  - Current Sensor
- Arduino Mega2560
  - Power Sampler



### SW Tools

**Operating System**
- Ubuntu 12.04 LTS Desktop
- Kernel Version 3.2.0-24

**Tools**
- taskset: A2C Mapping
- msr-tools: PMC Control
- CPUFreq: VF Scaling
- hwmon: Temperature

**Benchmarks**
- SPEC® CPU 2006 v1.2
- PARSEC v2.1
- NPB v3.3.1

## 3. Performance (CPI) Prediction across VF States

**Running at frequency f:**

C(f) = CC(f) + MC(f); (C: total cycles; CC: core cycles; MC: memory stall cycles;)

CPI(f) = CCPI(f) + MCPI(f); (Dividing both sides by instruction numbers)

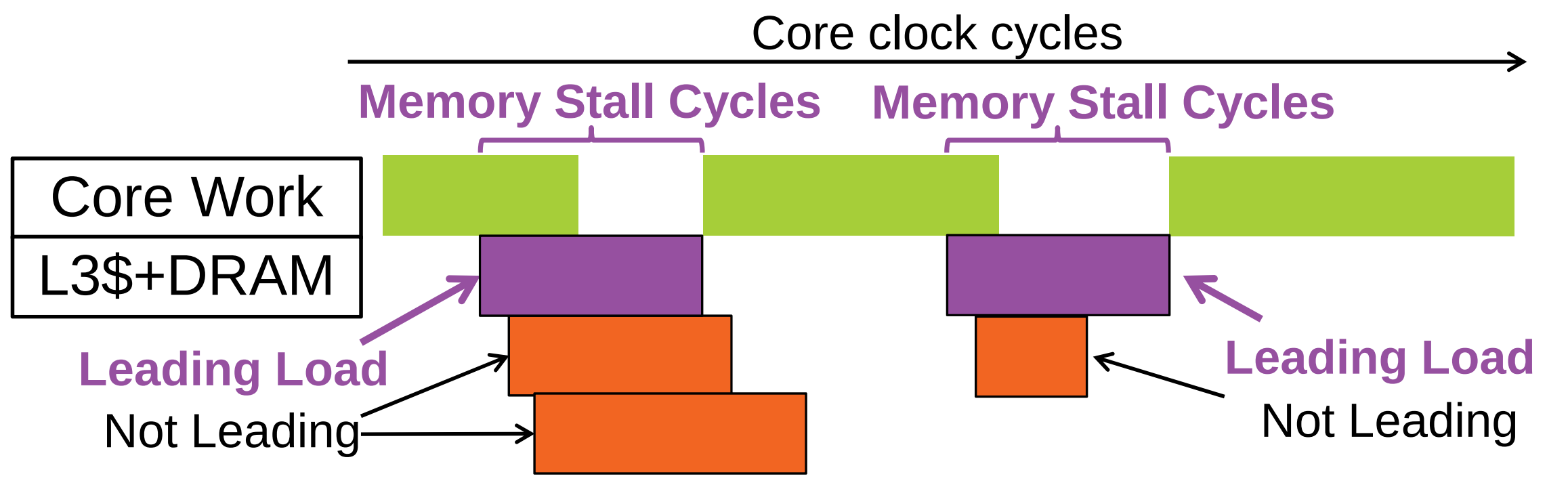
**If runs at frequency f':**

CPI(f') = CCPI(f) + MCPI(f) * ( f' / f ); (CCPI stays the same; MCPI scales with frequency)

Q: How to predict CPI(f')?

A: Get C(f) and MC(f) => Key: MC(f) => "Memory Stall Cycles = Cycles with **Leading Loads**"

The first non-prefetch load to leave a core while there is no other active leading load from that core



Core clock cycles

Memory Stall Cycles

Memory Stall Cycles

Core Work

L3$+DRAM

**Leading Load**

Not Leading

**Leading Load**

Not Leading

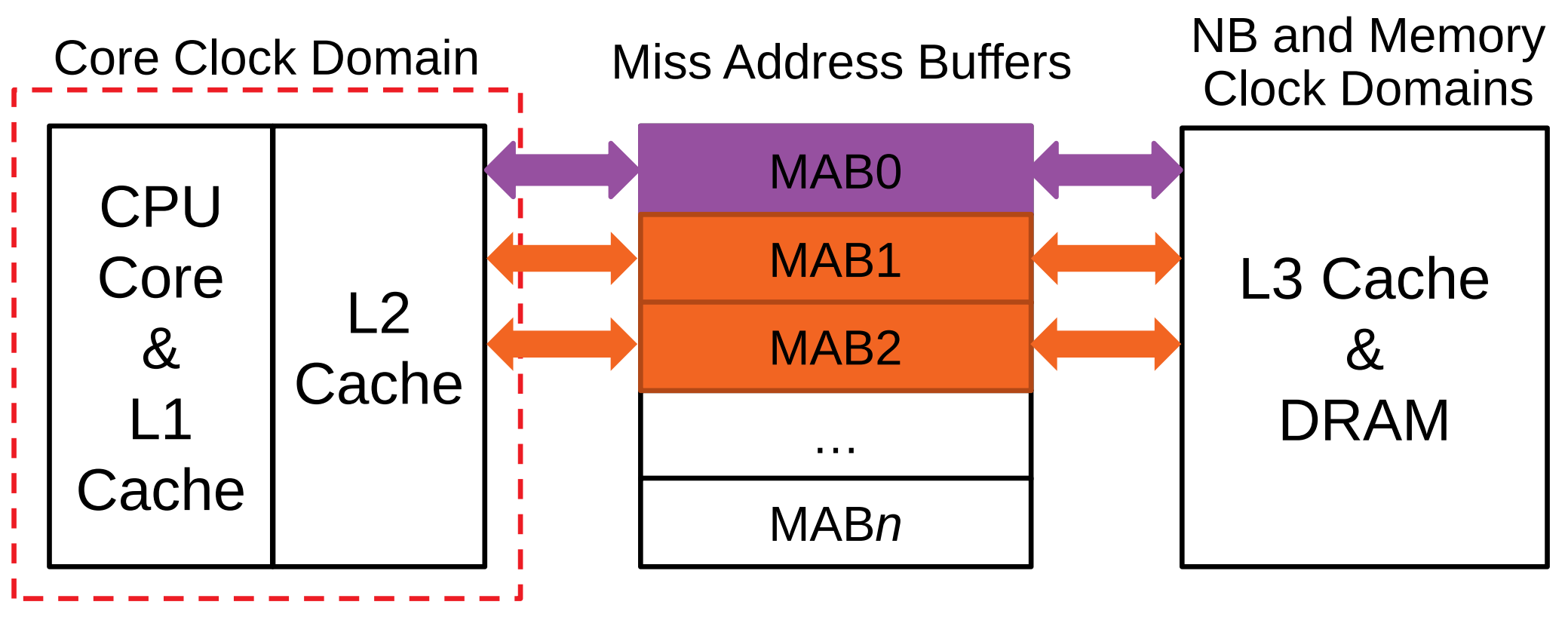
On Simulator:
- CF'10, Keramidas+
- TOC'11, Eyerman+
- IGCC'11, Rountree+

We demonstrate in commercial processors:
- USENIX-ATC'14, Su+

LL-MAB: Implement Leading Loads model on AMD processor through L2$ Miss Address Buffer

MAB: also known as MSHR (Miss Status Handling Register)
In MAB, buffers have fixed priority (MAB0 > MAB1 > … > MAB*n*).
L2$ misses always enter an empty entry with the highest priority.

Leading Loads always enter MAB0 !



Core Clock Domain

Miss Address Buffers

NB and Memory Clock Domains

CPU Core & L1 Cache

L2 Cache

MAB0

MAB1

MAB2

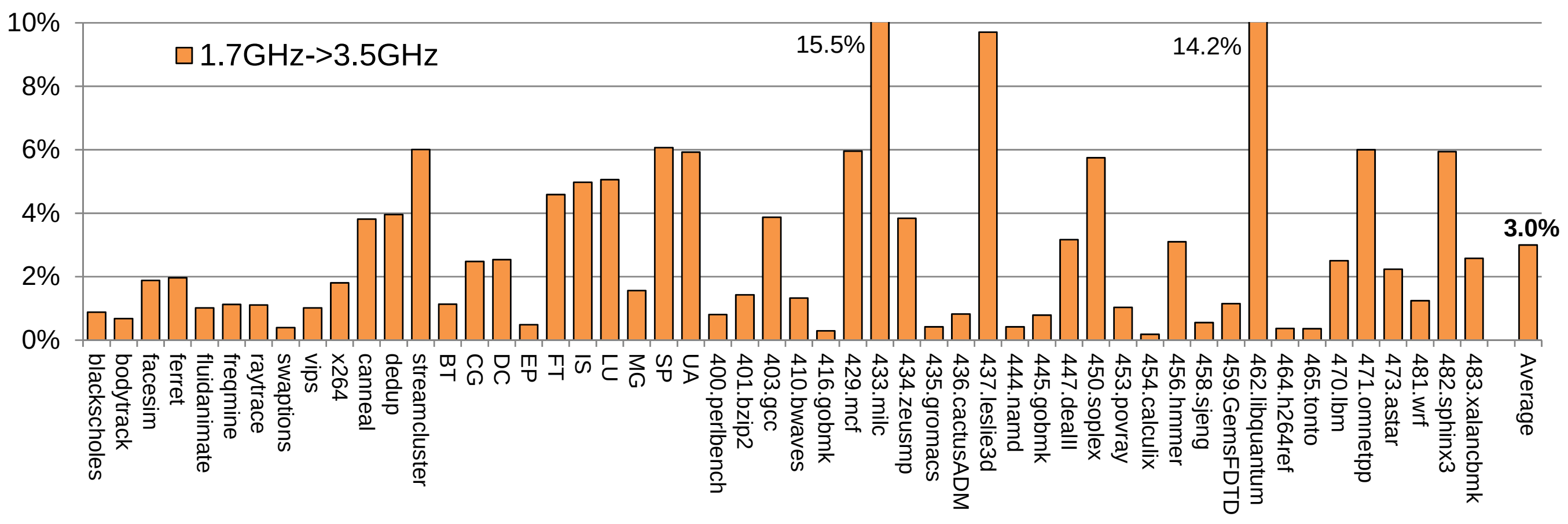
…

MAB*n*

L3 Cache & DRAM

**Use 3 HW events**:
E1: Total Cycle Number
E2: Instruction Number
E3: Memory Stall Cycle Number

| # | Code | Event Name |
|---|---|---|
| E1 | 0x76 | CPU_Clocks_not_Halted |
| E2 | 0xC0 | Retired_Instructions |
| **E3** | **0x69** | **MAB0_Wait_Cycles** |



CPI Prediction:
On FX-8320 CPU
1.7GHz => 3.5GHz

Avg Error: 3.0%
Avg STD: 3.2%

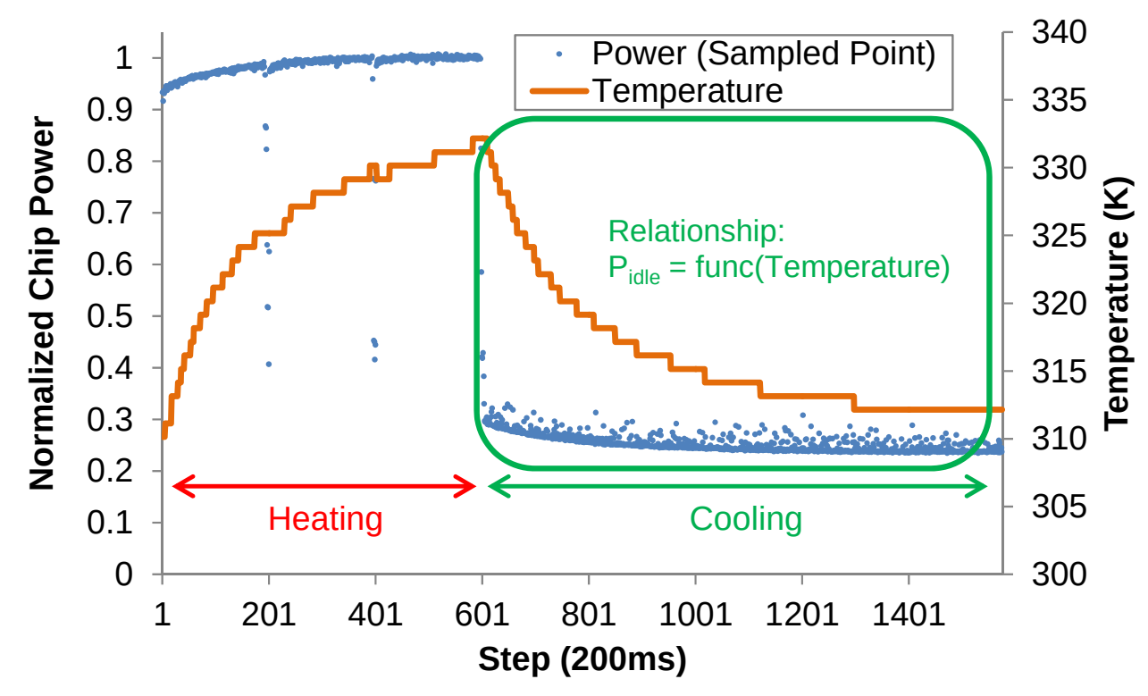
## 4. Power Prediction across VF States

**Building the power model: $P_{chip} = P_{idle} + P_{dyn}$ ($P_{idle}$ = Idle Power, $P_{dyn}$ = Dynamic Power)**

**Temperature aware idle power model**

Chip idle state: no benchmark is running

$P_{idle} \propto$ Temperature, Voltage



$$P_{idle}(VF_n) = W_{idle1} * \text{Temperature} + W_{idle0}$$
$$W_{idle1} = a_3V_n^3 + a_2V_n^2 + a_1V_n + a_0$$
$$W_{idle0} = b_3V_n^3 + b_2V_n^2 + b_1V_n + b_0$$

**PMC based dynamic power model**

$P_{dyn}$ is related with CPU HW events

$P_{dyn} \propto$ Activity of 9 HW Events

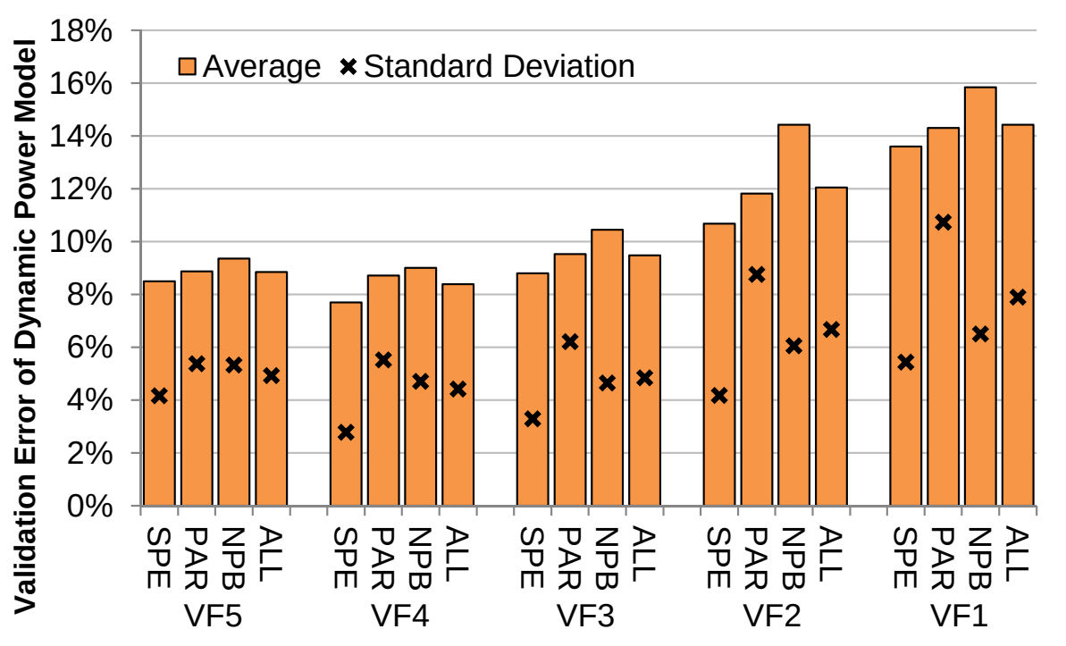
| # | Code | Event Name |
|---|---|---|
| E4 | 0xc1 | Retired_uOP |
| E5 | 0x00 | FPU_Pipe_Assignment |
| E6 | 0x80 | Instruction_Cache_Fetches |
| E7 | 0x40 | Data_Cache_Accesses |
| E8 | 0x7d | Request_to_L2_Cache |
| E9 | 0xc2 | Retired_Branch_Instructions |
| E10 | 0xc3 | Retired_Mispredicted_Branch_Instructions |
| E11 | 0x7e | L2_Cache_Misses |
| E12 | 0xd1 | Dispatch_Stalls |

$$P_{dyn}(VF_n) = \sum_{c=0}^{CoreNum-1}\left(\sum_{i=4}^{10}\left(\left(\frac{V_n}{V_5}\right)^{\alpha}\times W_{dyn}(i)\times E_{PS}(i)\right)+\sum_{i=11}^{12}\left(W_{dyn}(i)\times E_{PS}(i)\right)\right)$$

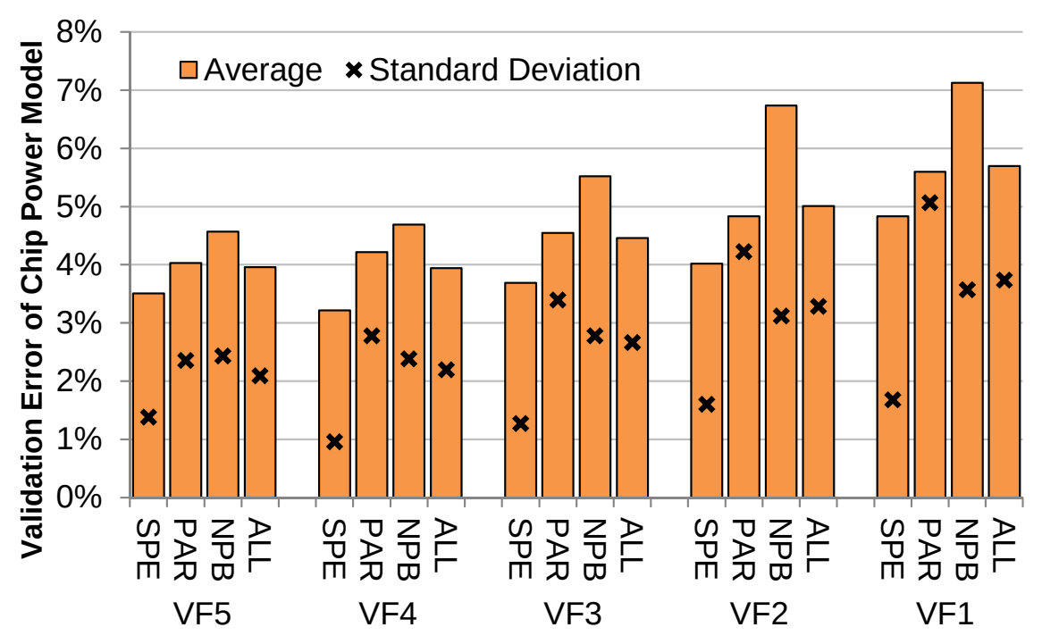
(PS: Per-Second value; c: core #; i: event #)

| VF State | Voltage (V) | Frequency (GHz) | *$P_{idle}$ Error* |
|---|---|---|---|
| VF5 | 1.320 | 3.5 | 2.4% |
| VF4 | 1.242 | 2.9 | 2.6% |
| VF3 | 1.128 | 2.3 | 3.6% |
| VF2 | 1.008 | 1.7 | 2.8% |
| VF1 | 0.888 | 1.4 | 2.7% |

$P_{idle}$: 2.8% error



$P_{dyn}$: 10.6% error, 5.8% standard deviation



$P_{chip}$: 4.6% error, 2.8% standard deviation

### Power Prediction across VF States

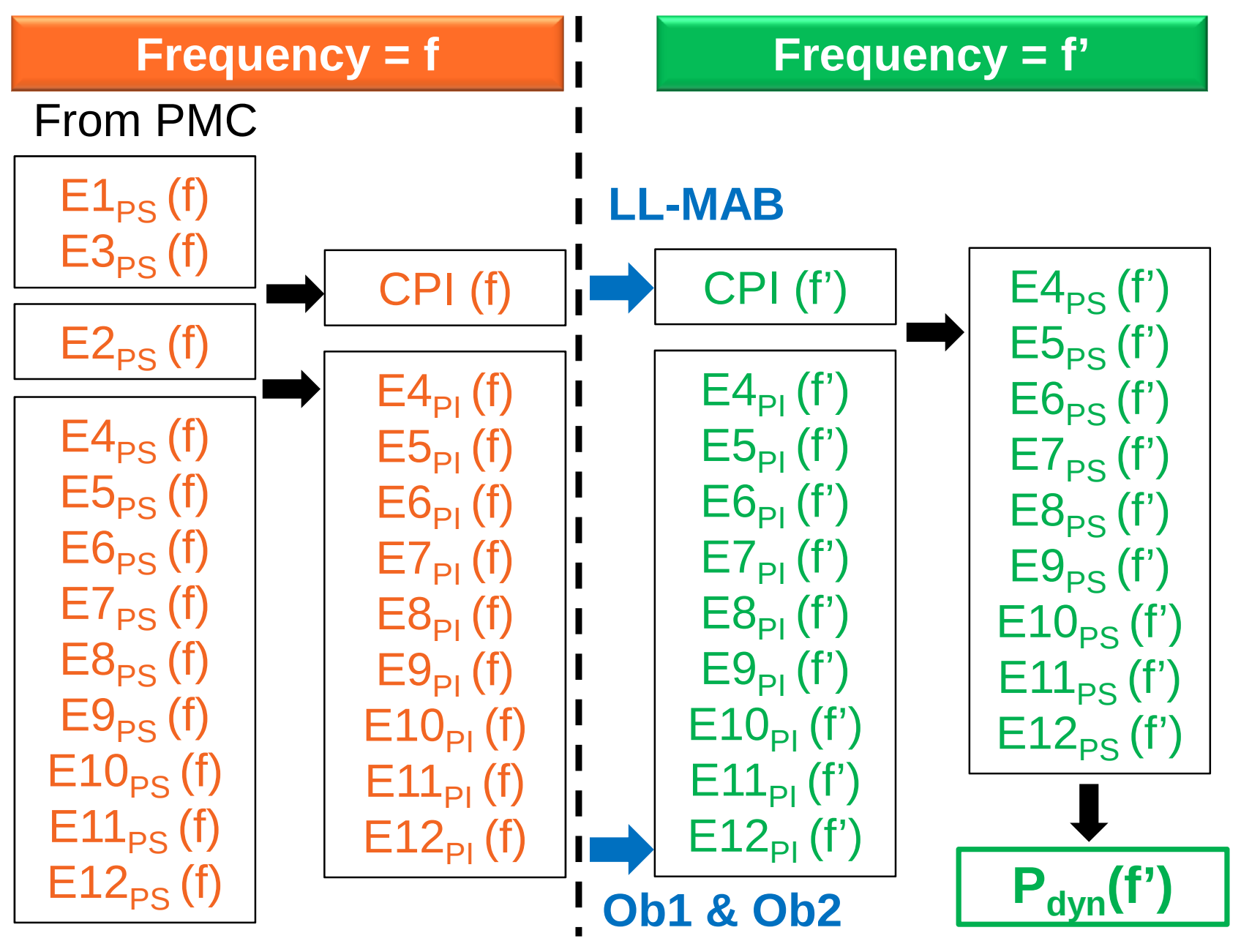
**$P_{idle}$**: Use the current temperature of VF states (A simplified approximation).

**$P_{dyn}$**: Need the HW Events Predictor (2 Experimental Observations + LL-MAB CPI Predictor)

Observation 1: At any given point in the execution of a program, core-private hardware event counts per-instruction (PI) are independent of VF state.

Observation 2: At any given point in the execution of a program, *CPI-DispatchStallsPI* is independent of VF state.

**HW Events and $P_{dyn}$ Prediction Flow**



**Frequency = f**

From PMC

$E1_{PS}$ (f)
$E3_{PS}$ (f)

$E2_{PS}$ (f)

$E4_{PS}$ (f)
$E5_{PS}$ (f)
$E6_{PS}$ (f)
$E7_{PS}$ (f)
$E8_{PS}$ (f)
$E9_{PS}$ (f)
$E10_{PS}$ (f)
$E11_{PS}$ (f)
$E12_{PS}$ (f)

CPI (f)

$E4_{PI}$ (f)
$E5_{PI}$ (f)
$E6_{PI}$ (f)
$E7_{PI}$ (f)
$E8_{PI}$ (f)
$E9_{PI}$ (f)
$E10_{PI}$ (f)
$E11_{PI}$ (f)
$E12_{PI}$ (f)

**Frequency = f'**

**LL-MAB**

CPI (f')

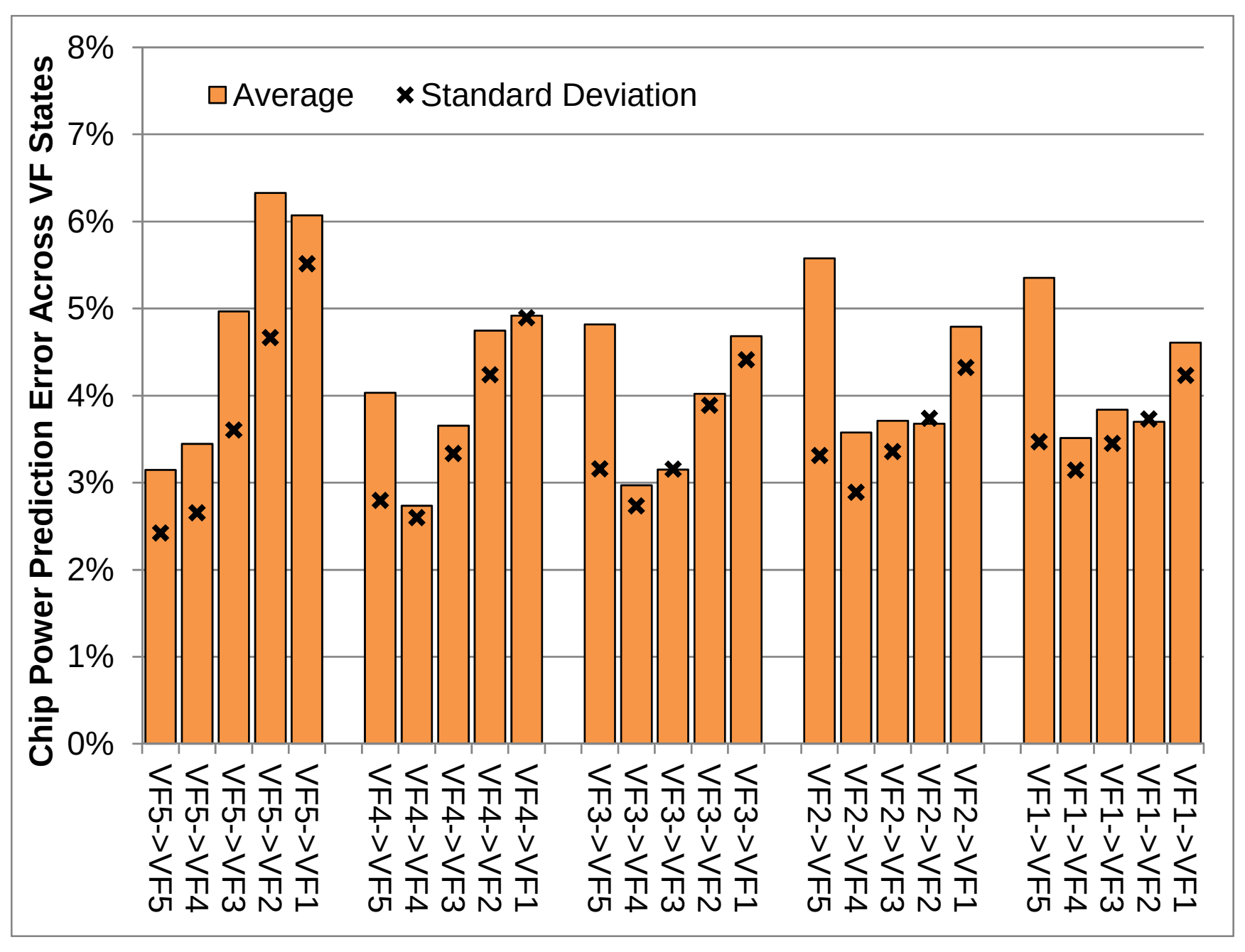
$E4_{PI}$ (f')
$E5_{PI}$ (f')
$E6_{PI}$ (f')
$E7_{PI}$ (f')
$E8_{PI}$ (f')
$E9_{PI}$ (f')
$E10_{PI}$ (f')
$E11_{PI}$ (f')
$E12_{PI}$ (f')

**Ob1 & Ob2**

$E4_{PS}$ (f')
$E5_{PS}$ (f')
$E6_{PS}$ (f')
$E7_{PS}$ (f')
$E8_{PS}$ (f')
$E9_{PS}$ (f')
$E10_{PS}$ (f')
$E11_{PS}$ (f')
$E12_{PS}$ (f')

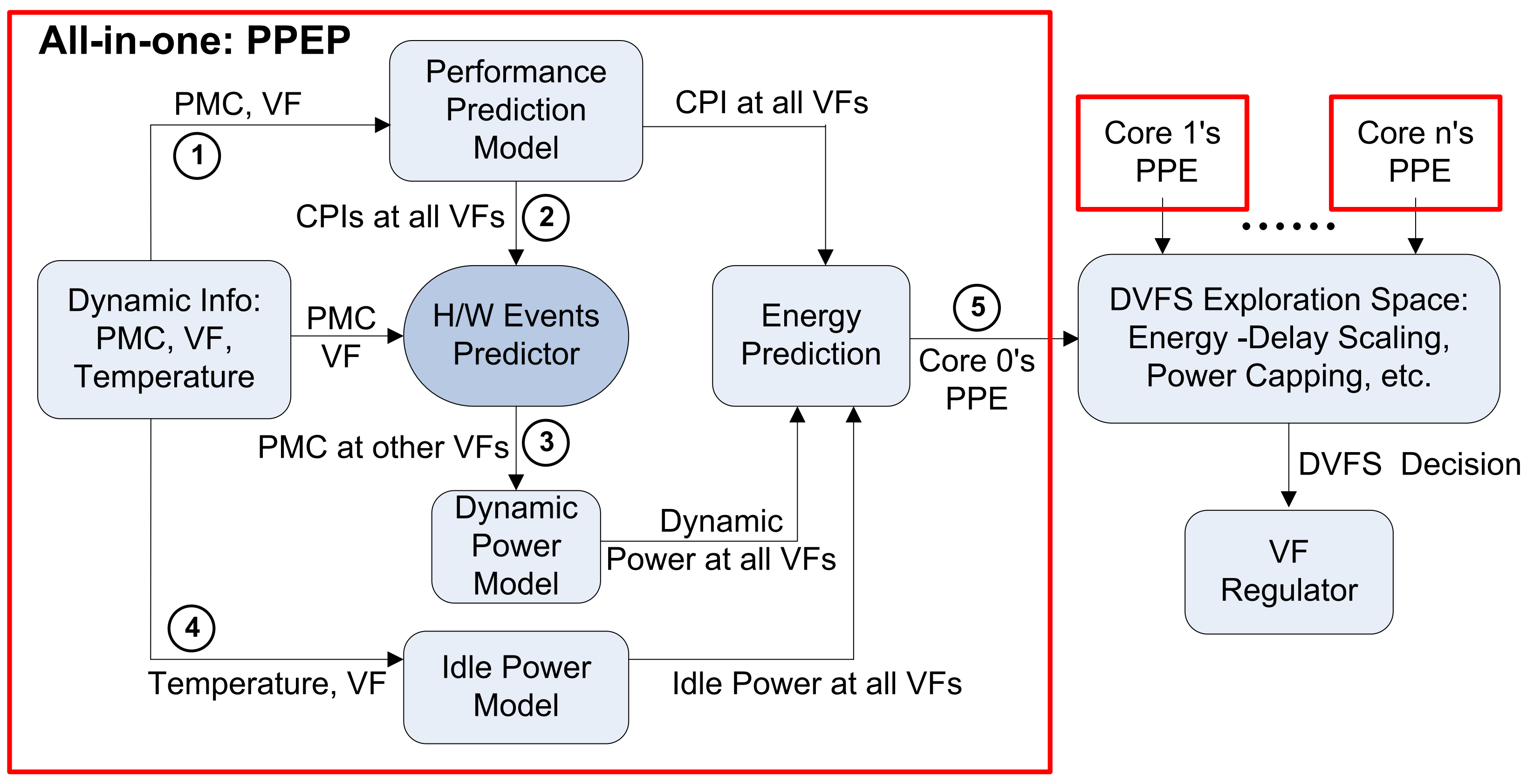
**$P_{dyn}$(f')**

**Power Prediction Error (Average $P_{chip}$)**



## 5. Putting them together: PPEP

After distributing $P_{idle}$ and $P_{dyn}$ to each core:



**All-in-one: PPEP**

Performance Prediction Model

H/W Events Predictor

Dynamic Info: PMC, VF, Temperature

Dynamic Power Model

Idle Power Model

Energy Prediction

① PMC, VF

CPI at all VFs

② CPIs at all VFs

PMC VF

③ PMC at other VFs

Dynamic Power at all VFs

④ Temperature, VF

Idle Power at all VFs

⑤ Core 0's PPE

Core 1's PPE

……

Core n's PPE

DVFS Exploration Space: Energy -Delay Scaling, Power Capping, etc.

DVFS Decision

VF Regulator

## 6. Conclusion

On AMD Commercial Processors:

- ◆ Implemented an across-VF CPI predictor "LL-MAB"

  Following the Leading Loads theory, 3.2% CPI prediction error
- ◆ Implemented an across-VF power predictor

  Driven by the LL-MAB CPI predictor, 4.2% power prediction error
- ◆ Combining them together: PPEP

  Supply online PPE information of each VF states

  Software method w/o requiring hardware or operating system modifications