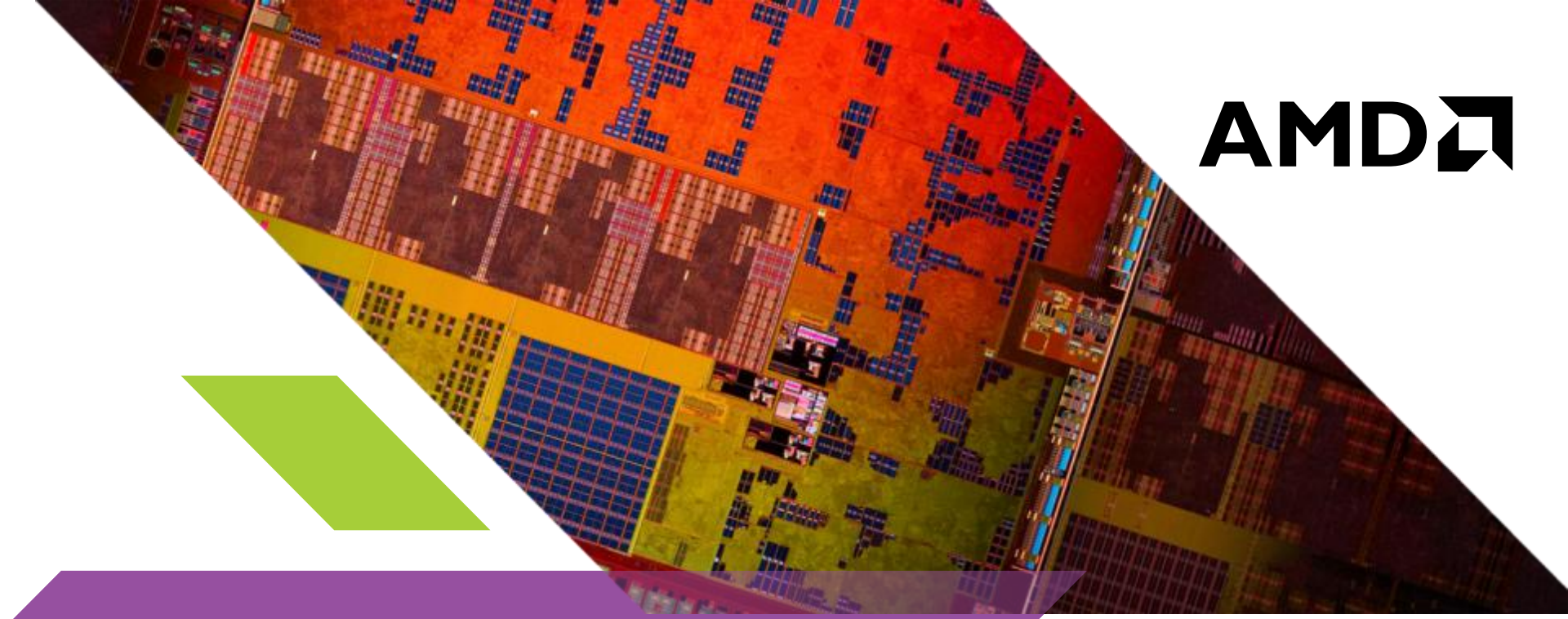


PPEP: ONLINE PERFORMANCE, POWER, AND ENERGY PREDICTION FRAMEWORK

BO SU[†] JUNLI GU[‡] LI SHEN[†] WEI HUANG[‡]
JOSEPH L. GREATHOUSE[‡] ZHIYING WANG[†]

[†]NUDT [‡]AMD RESEARCH

DECEMBER 17, 2014

國防科學技術大學
National University of Defense Technology

BACKGROUND

- Dynamic Voltage and Frequency Scaling (DVFS)
  - Widely employed
    - Boost performance, lower power, and improve energy efficiency

BACKGROUND

AMD

- Dynamic Voltage and Frequency Scaling (DVFS)
  - Widely employed
    - Boost performance, lower power, and improve energy efficiency
  - Good DVFS decisions require
    - **Accurate performance & power predictions across Voltage Frequency (VF) states**

BACKGROUND

AMD

- Dynamic Voltage and Frequency Scaling (DVFS)
  - Widely employed
    - Boost performance, lower power, and improve energy efficiency
  - Good DVFS decisions require
    - **Accurate performance & power predictions across Voltage Frequency (VF) states**

- Challenges Brought by Modern Processors
  - Cores and north bridge
    - Different clock domains and power planes
  - Off-chip Memory
    - Own clock domain

TWO QUESTIONS

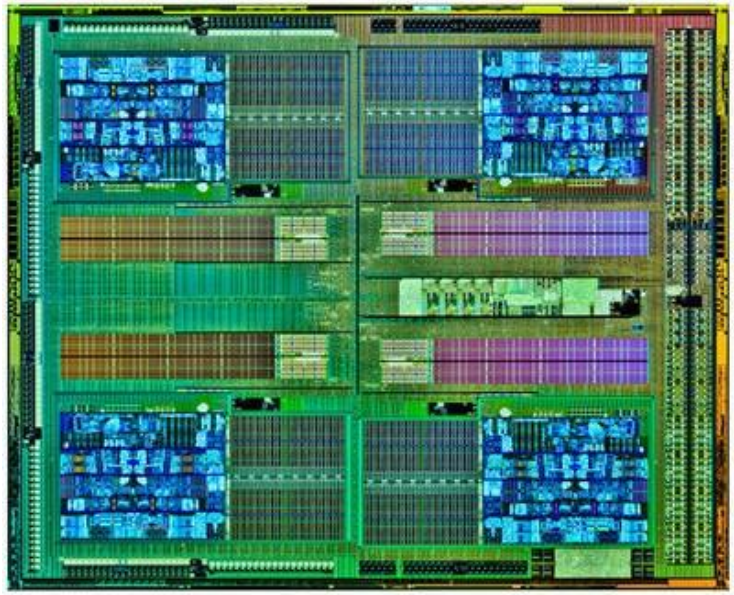
AMD

Q1: How does the application's performance change with the VF state?

Q2: How does the application's power change with the VF state?

EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

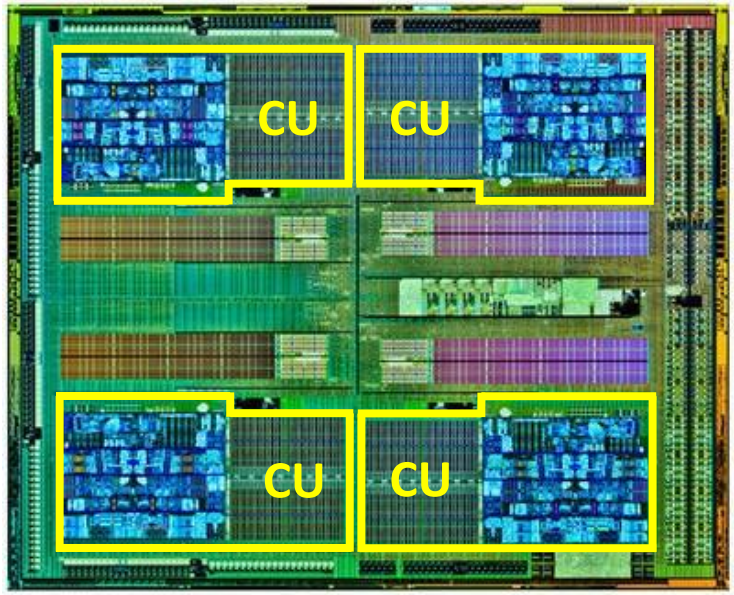
- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h “Piledriver”



FX-8320 CPU

EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

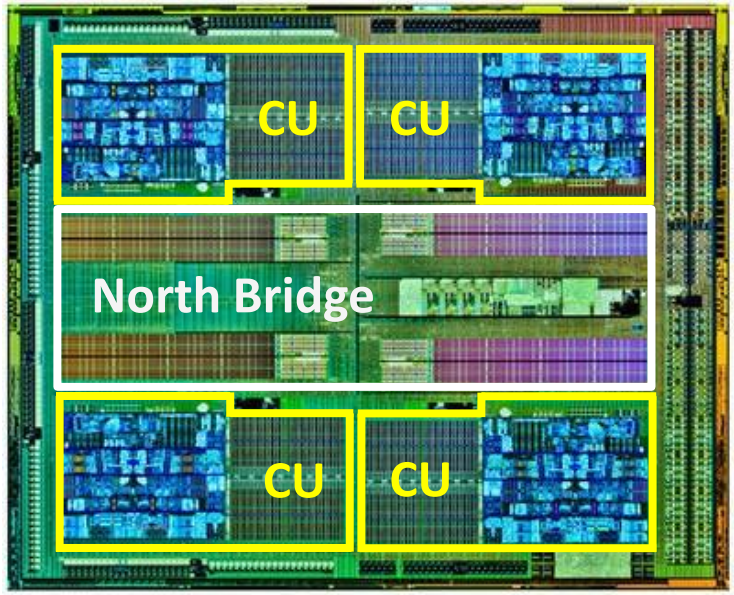
- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h "Piledriver"
    - 4 CUs (Compute Units)
      - 2 Cores in a CU
      - 5 VF (Voltage-Frequency) states



FX-8320 CPU

EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

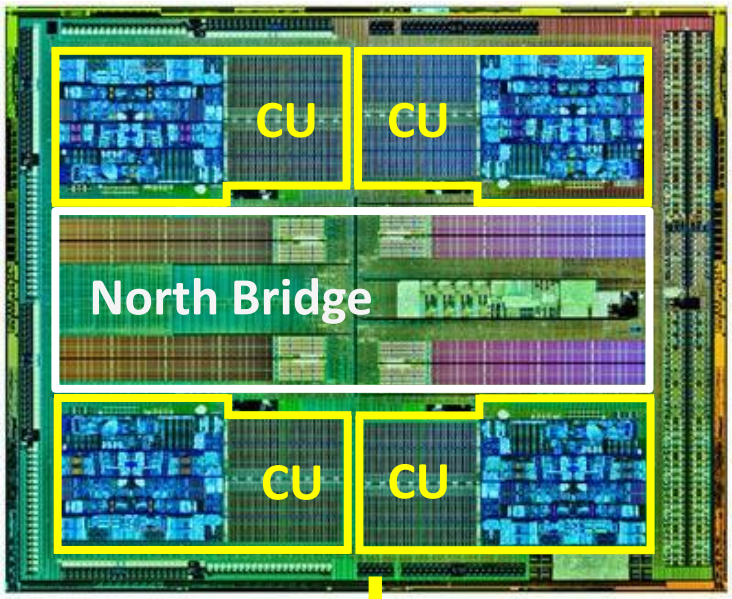
- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h "Piledriver"
    - 4 CUs (Compute Units)
      - 2 Cores in a CU
      - 5 VF (Voltage-Frequency) states
    - The NB (North Bridge)
      - Shared by all CUs
      - L3$ and memory controller



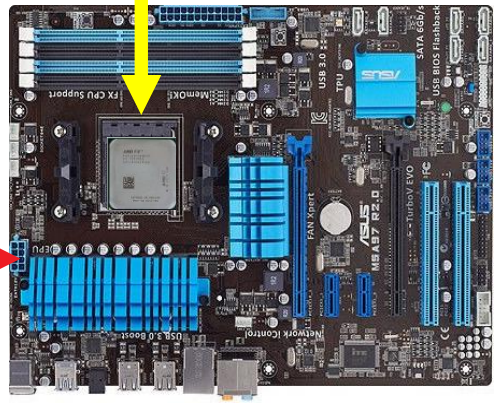
FX-8320 CPU

EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h "Piledriver"
    - 4 CUs (Compute Units)
      - 2 Cores in a CU
      - 5 VF (Voltage-Frequency) states
    - The NB (North Bridge)
      - Shared by all CUs
      - L3$ and memory controller
- ***Power Measurement***
  - *Pololu ACS711:* Current sensor



FX-8320 CPU

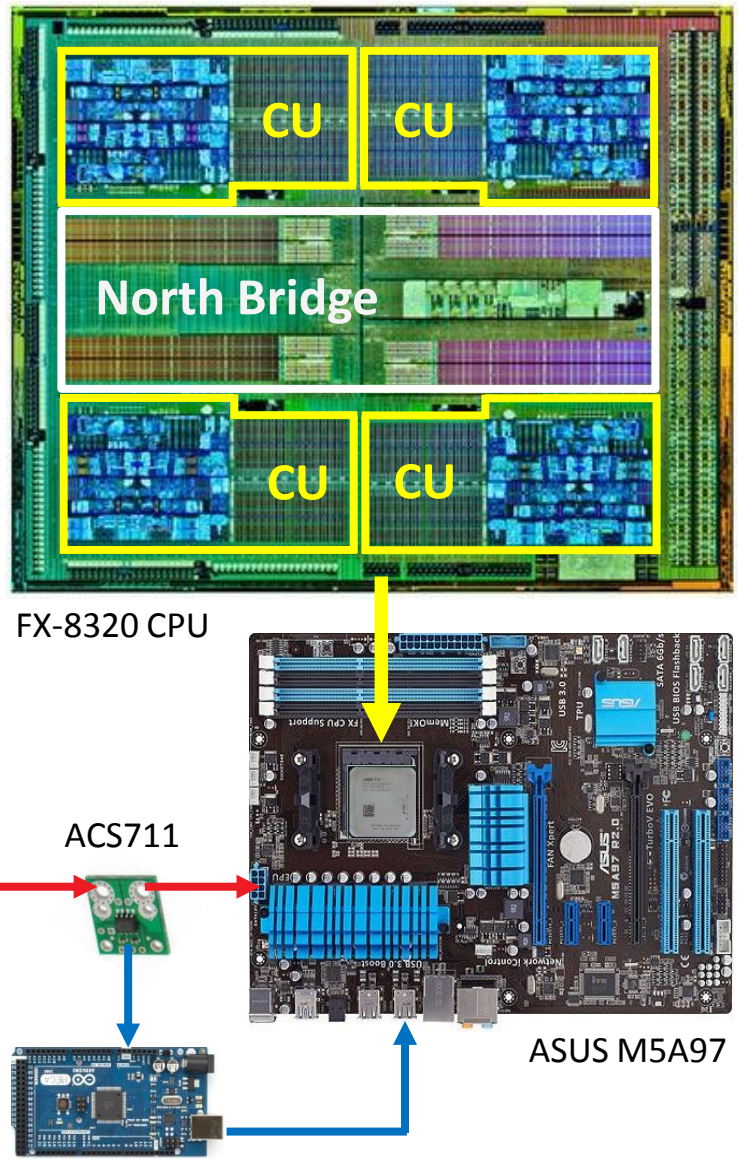

ASUS M5A97

Power Supply

ACS711

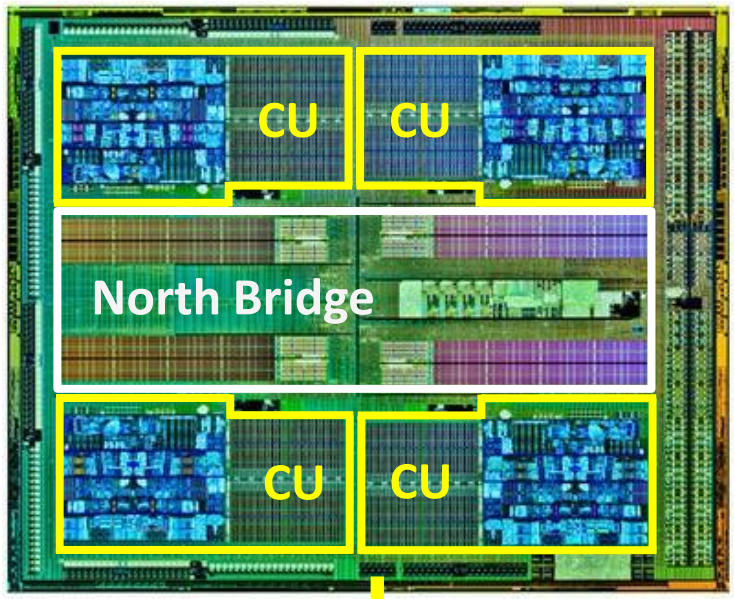
EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h "Piledriver"
    - 4 CUs (Compute Units)
      - 2 Cores in a CU
      - 5 VF (Voltage-Frequency) states
    - The NB (North Bridge)
      - Shared by all CUs
      - L3$ and memory controller
- ***Power Measurement***
  - *Pololu ACS711:* Current sensor
  - *Arduino:* Power reading

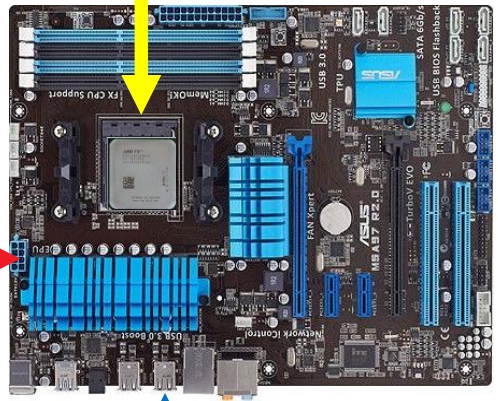


EXPERIMENTAL SETUP
PLATFORM & BENCHMARKS

- ***Processor: AMD FX-8320***
  - 2nd Generation Family 15h "Piledriver"
    - 4 CUs (Compute Units)
      - 2 Cores in a CU
      - 5 VF (Voltage-Frequency) states
    - The NB (North Bridge)
      - Shared by all CUs
      - L3$ and memory controller
- ***Power Measurement***
  - *Pololu ACS711:* Current sensor
  - *Arduino:* Power reading
- ***Benchmark Suites***
  - SPEC® CPU 2006, PARSEC, NAS Parallel Benchmarks



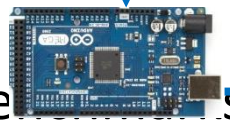
FX-8320 CPU



ASUS M5A97

Power Supply

ACS711



Arduino

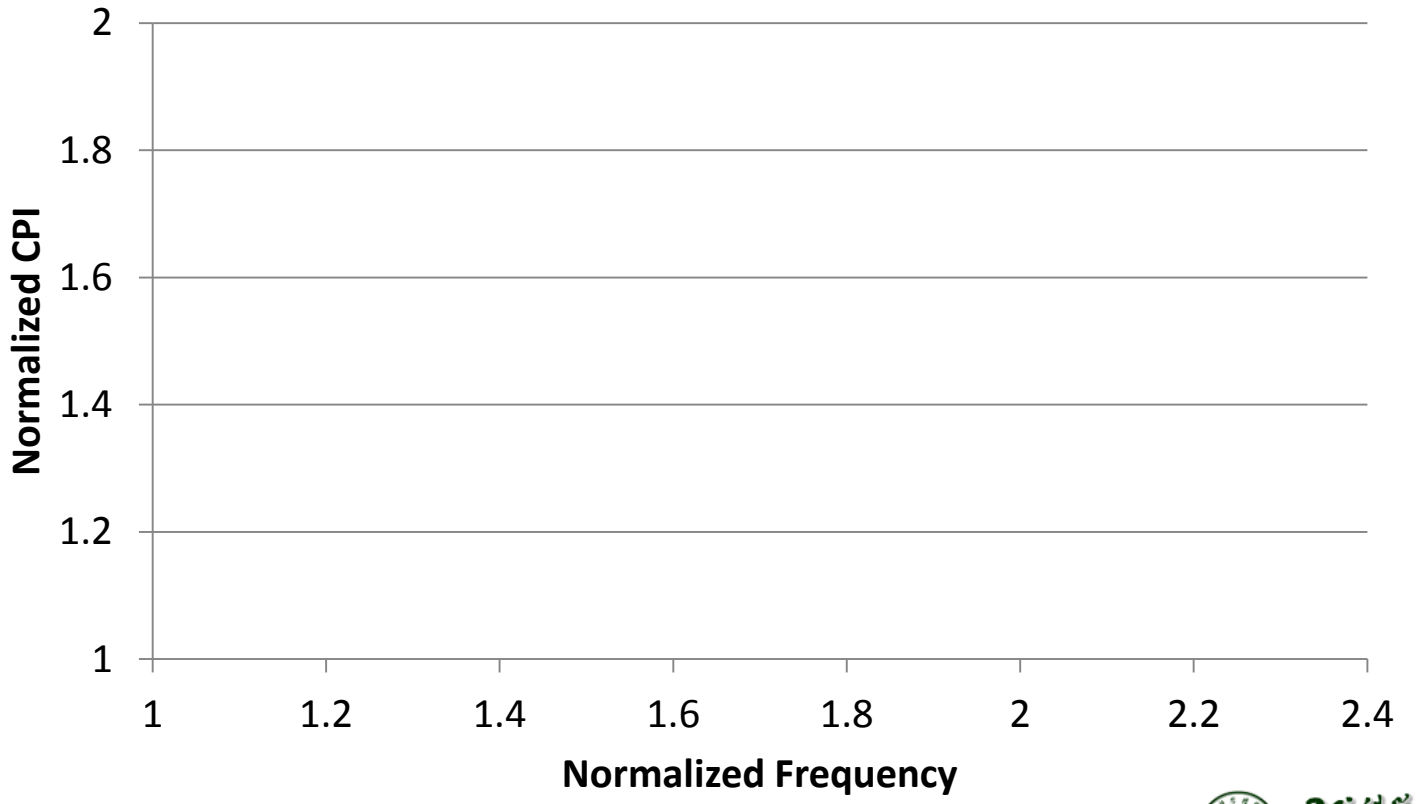
OUTLINE

AMD

- Background
- Experimental Setup
- **Performance Prediction Across DVFS States**
- Power Prediction Across DVFS States
- PPEP Framework
- Conclusion

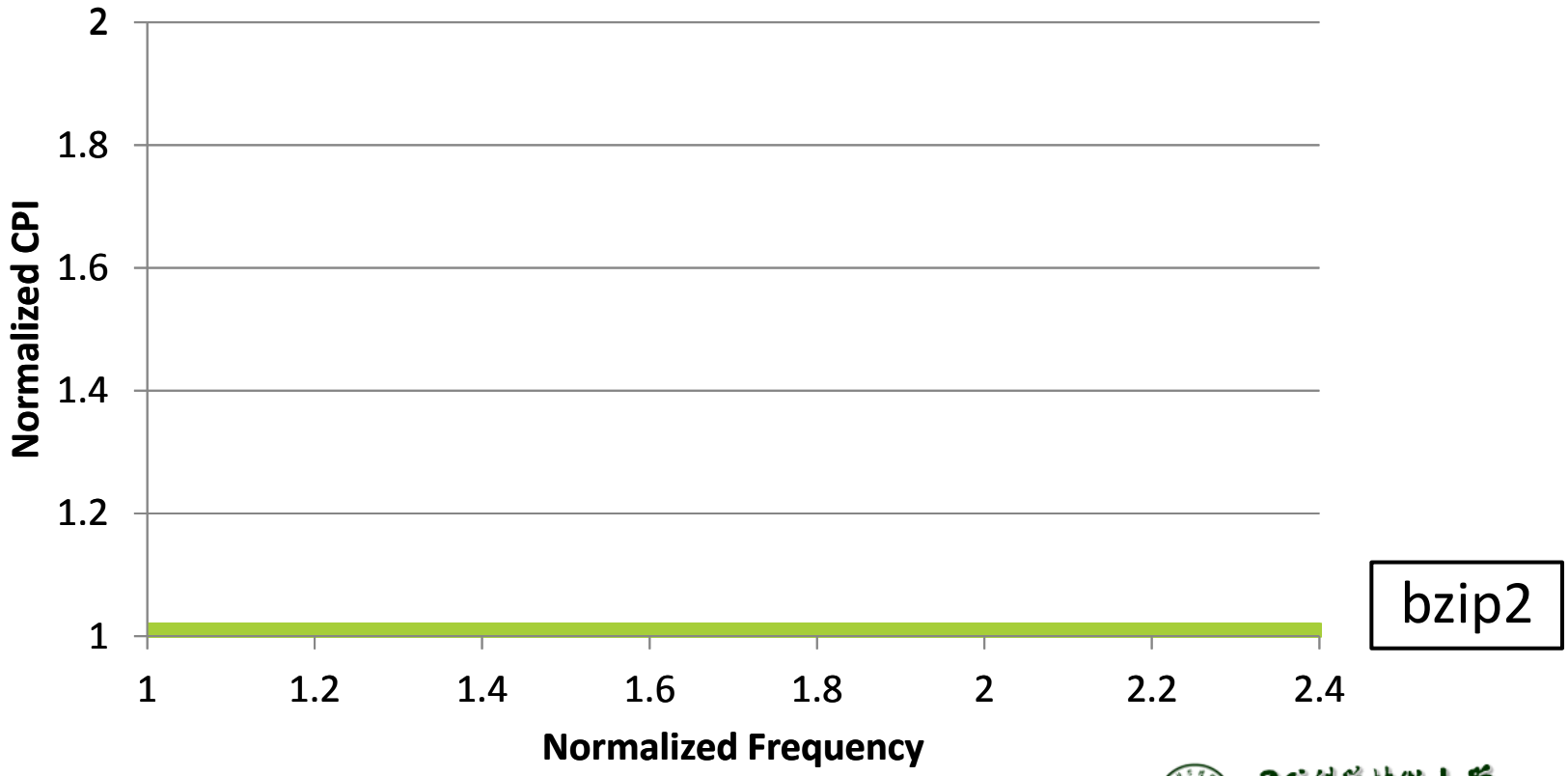
Q1: How does the application's performance change with the VF state?

HOW CPI CHANGES WITH FREQUENCY
(LOWER IS BETTER)



HOW CPI CHANGES WITH FREQUENCY
(LOWER IS BETTER)

Estimate #1 CPI does not change when frequency increases.

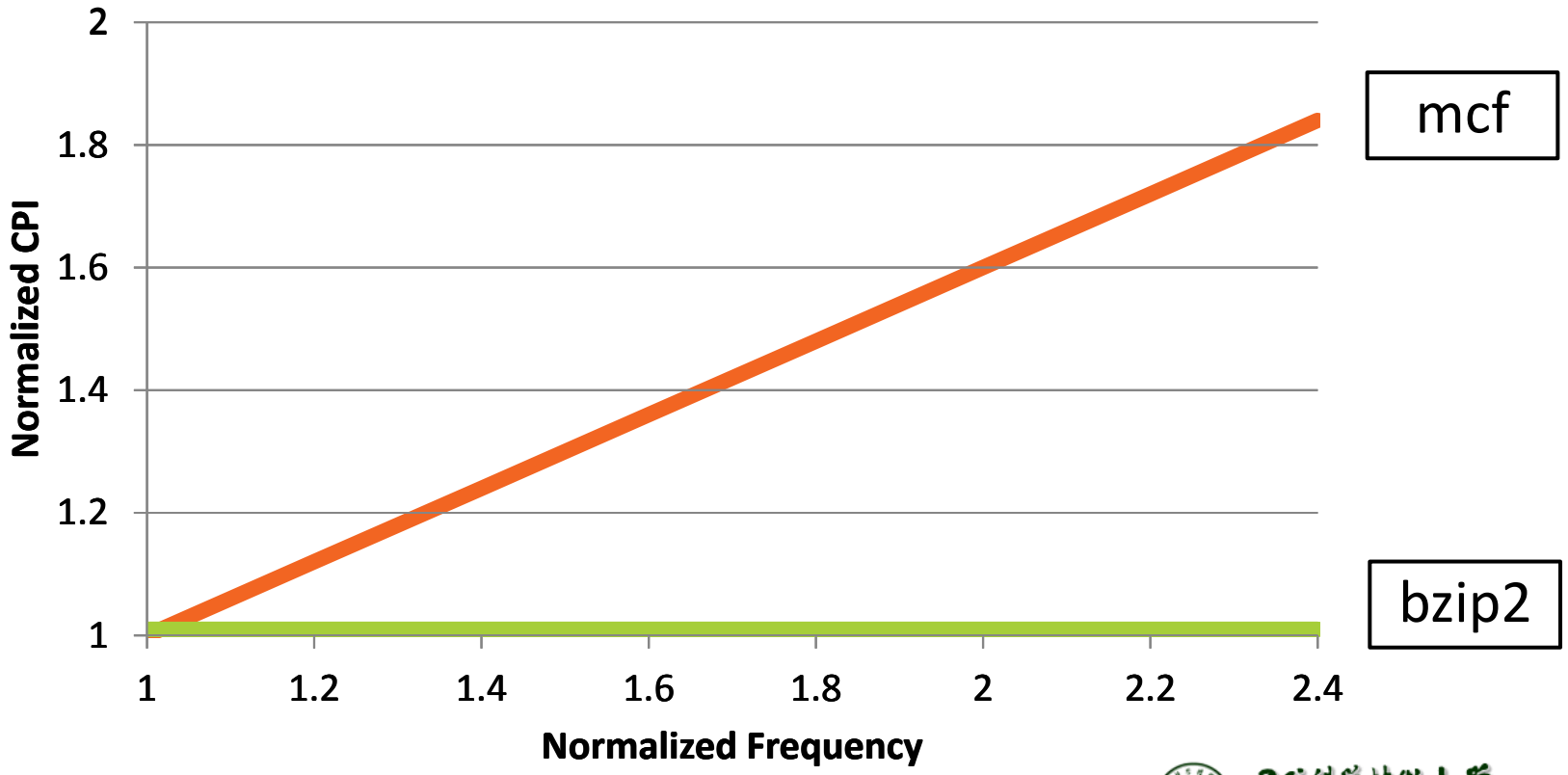


National University of Defense Technology

HOW CPI CHANGES WITH FREQUENCY
(LOWER IS BETTER)

AMD

Estimate #1	CPI does not change when frequency increases.
Estimate #2	CPI also increases when frequency increases.

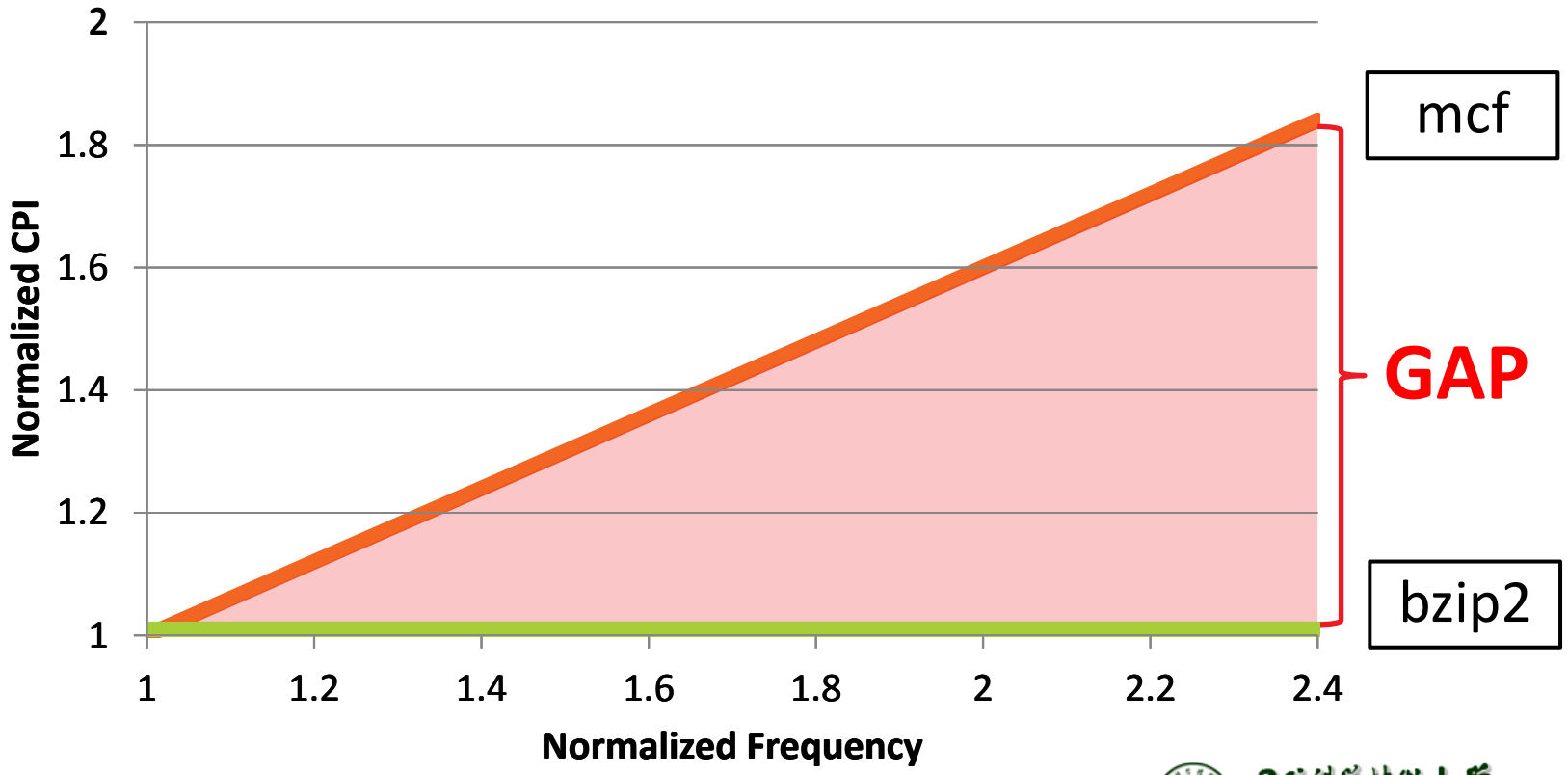


國防科學技術大學
National University of Defense Technology

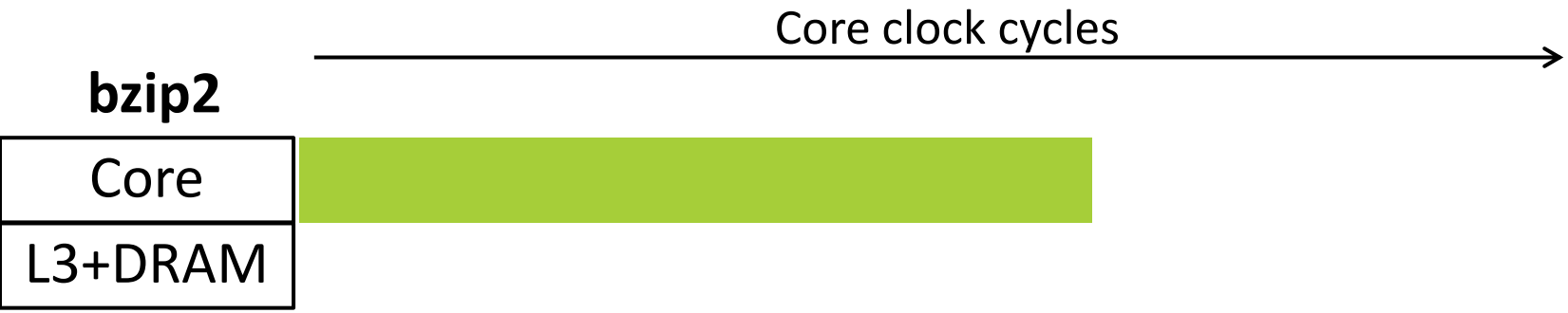
HOW CPI CHANGES WITH FREQUENCY
(LOWER IS BETTER)

AMD

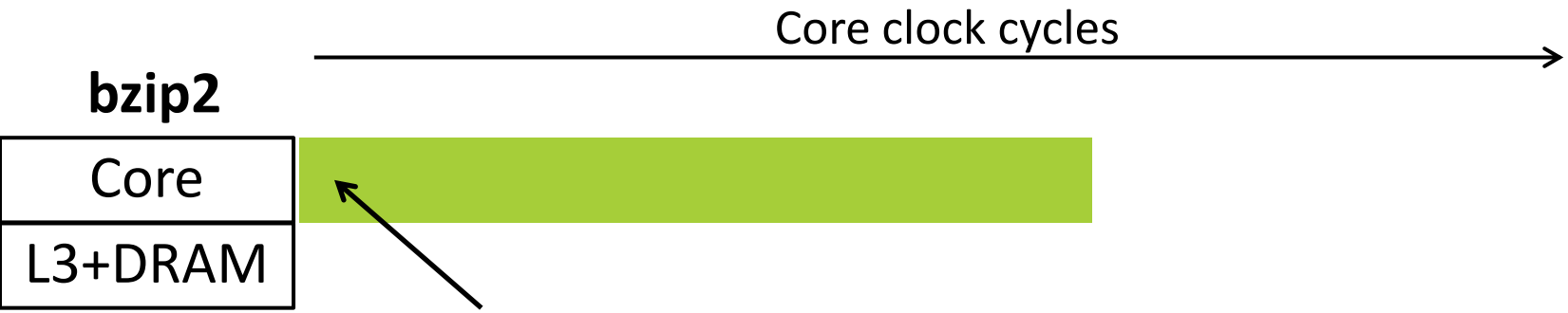
Estimate #1	CPI does not change when frequency increases.
Estimate #2	CPI also increases when frequency increases.



WHAT MAKES THE CPI DIFFERENCE?



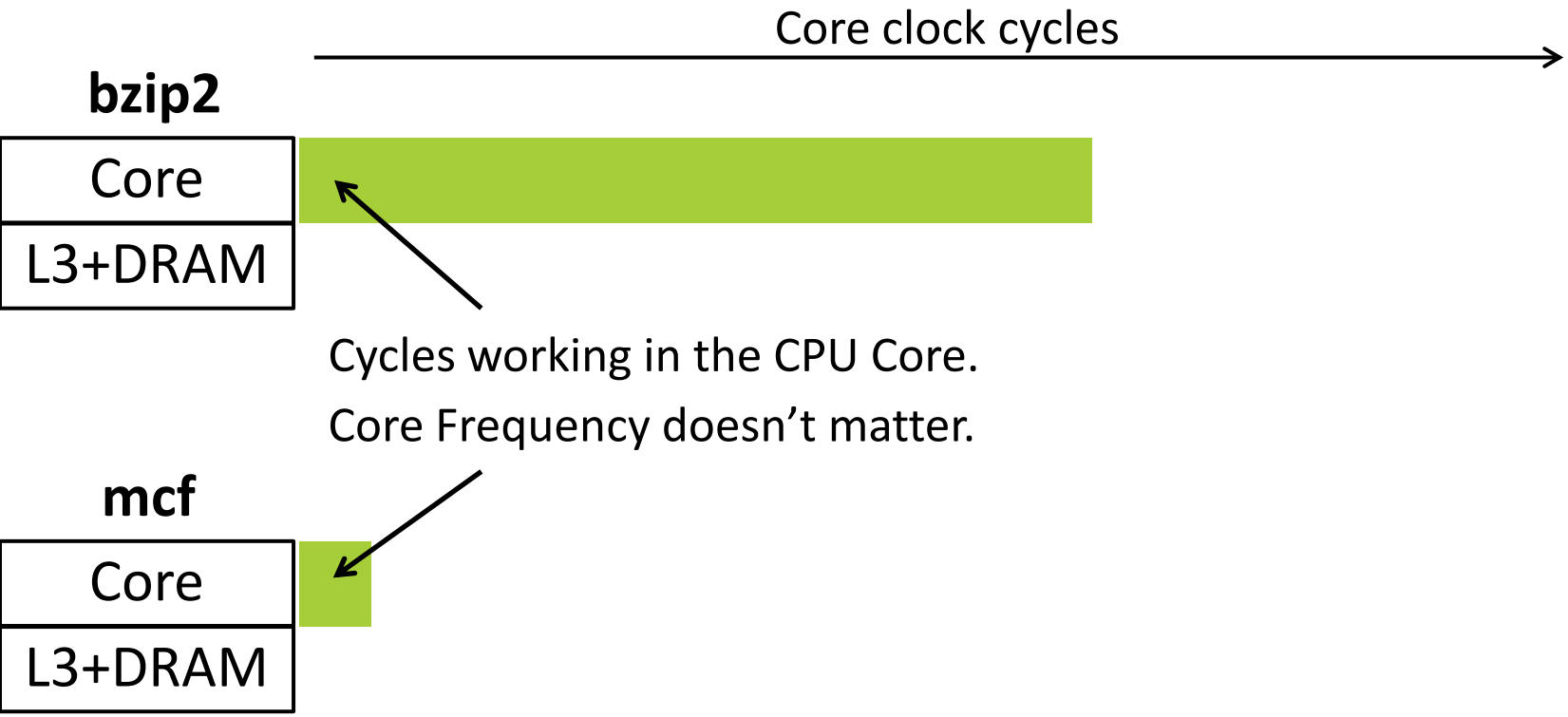
WHAT MAKES THE CPI DIFFERENCE?



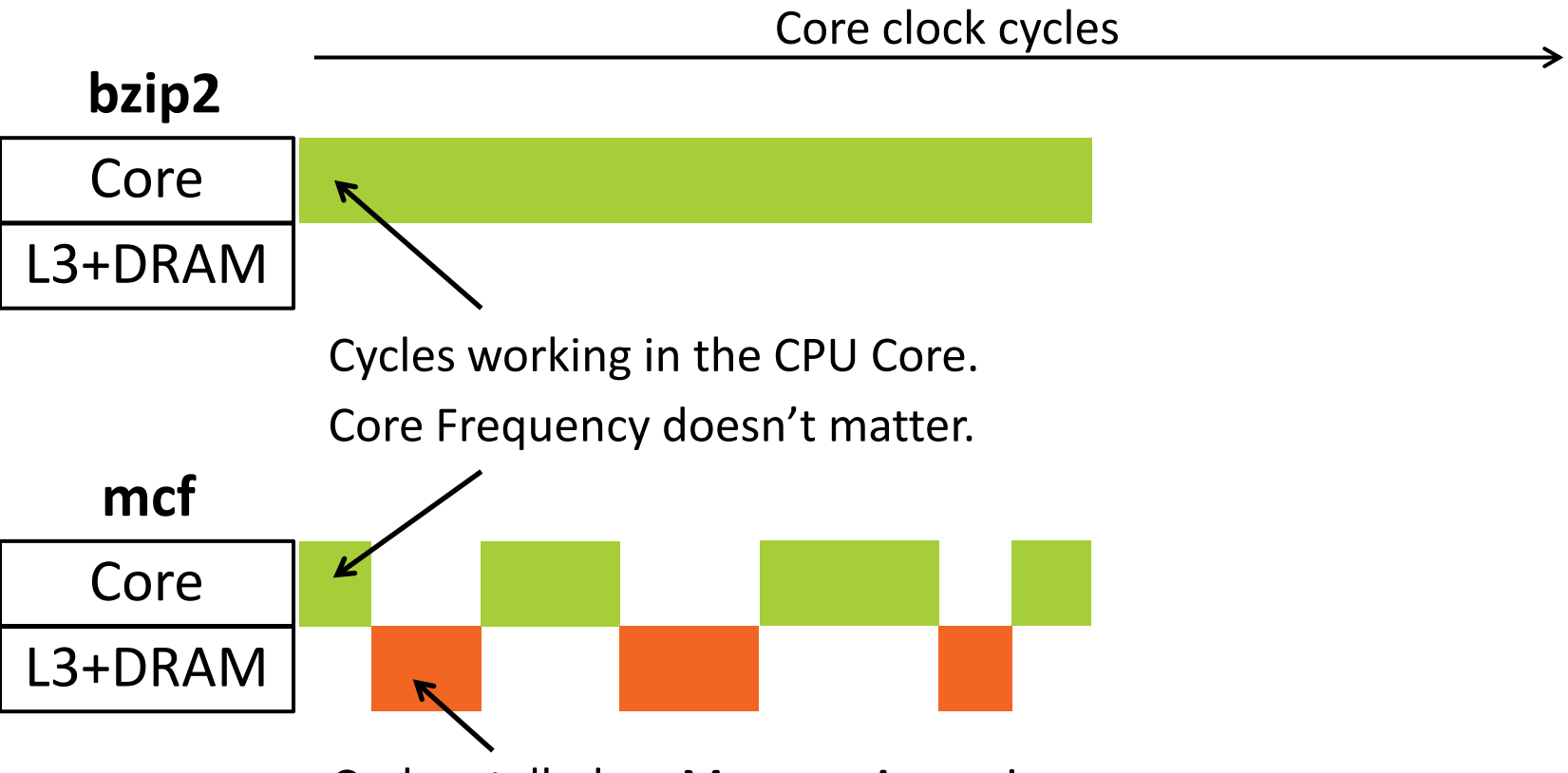
Cycles working in the CPU Core.

Core Frequency doesn't matter.

WHAT MAKES THE CPI DIFFERENCE?



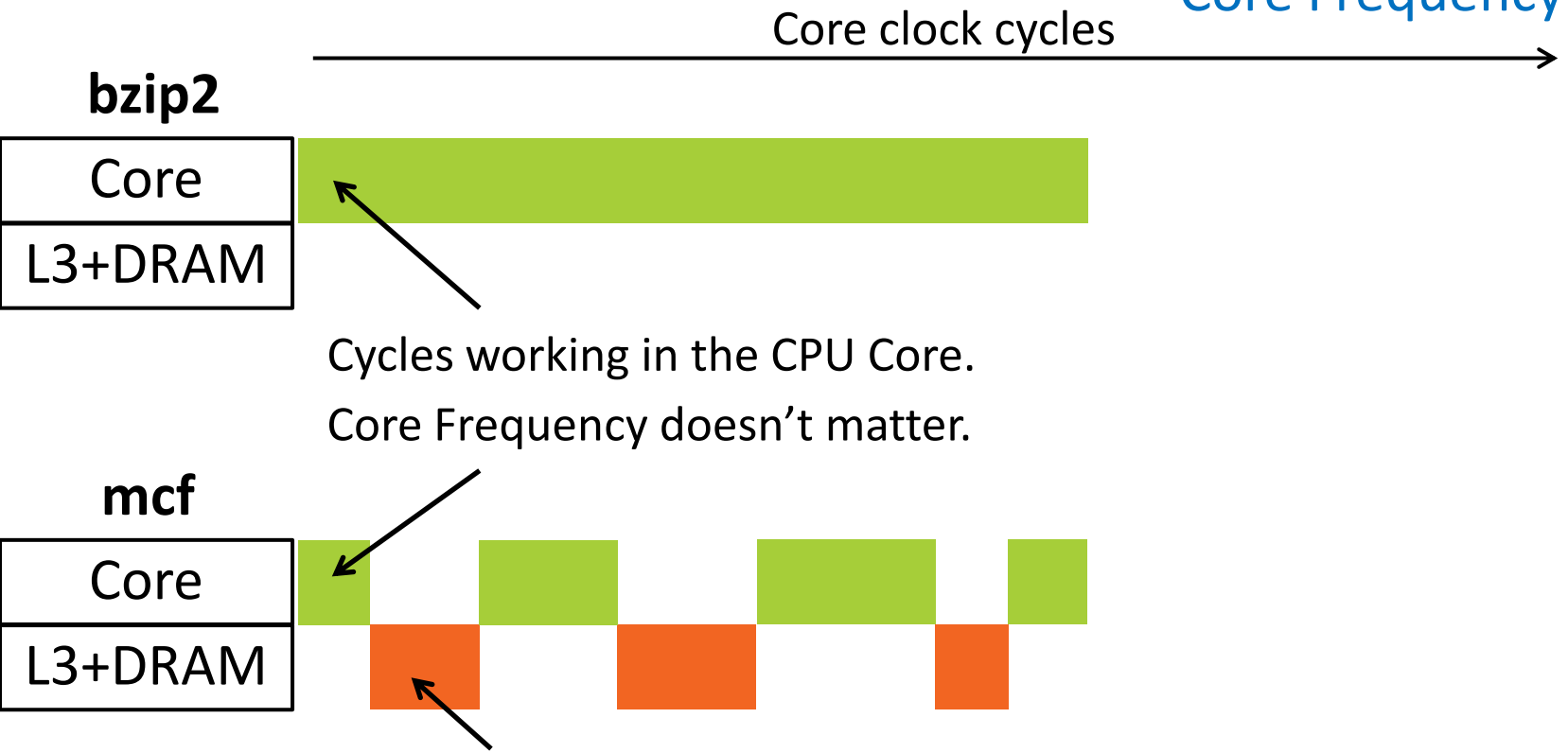
WHAT MAKES THE CPI DIFFERENCE?



WHAT MAKES THE CPI DIFFERENCE?

AMD

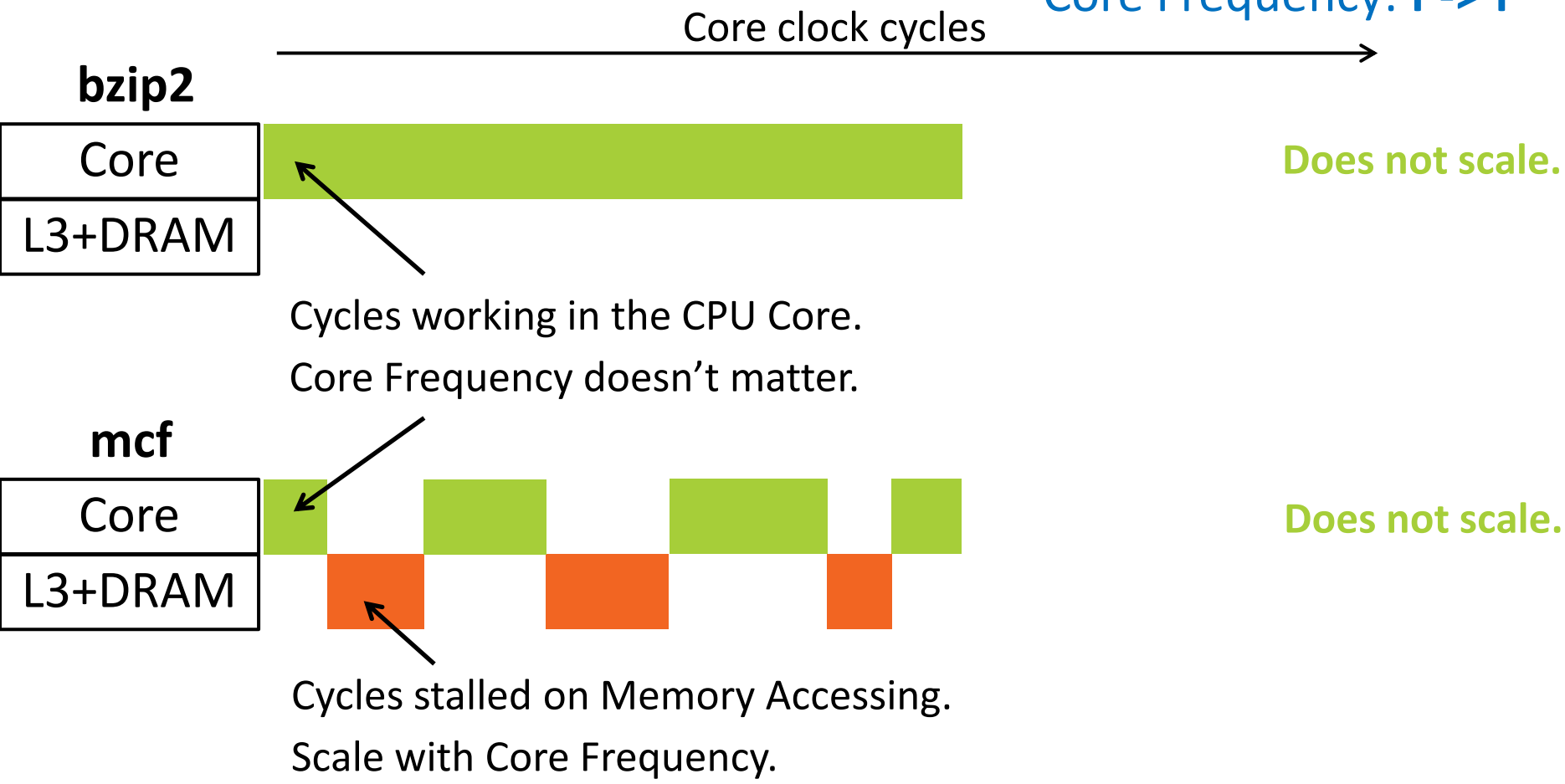
Core Frequency: **f -> f'**



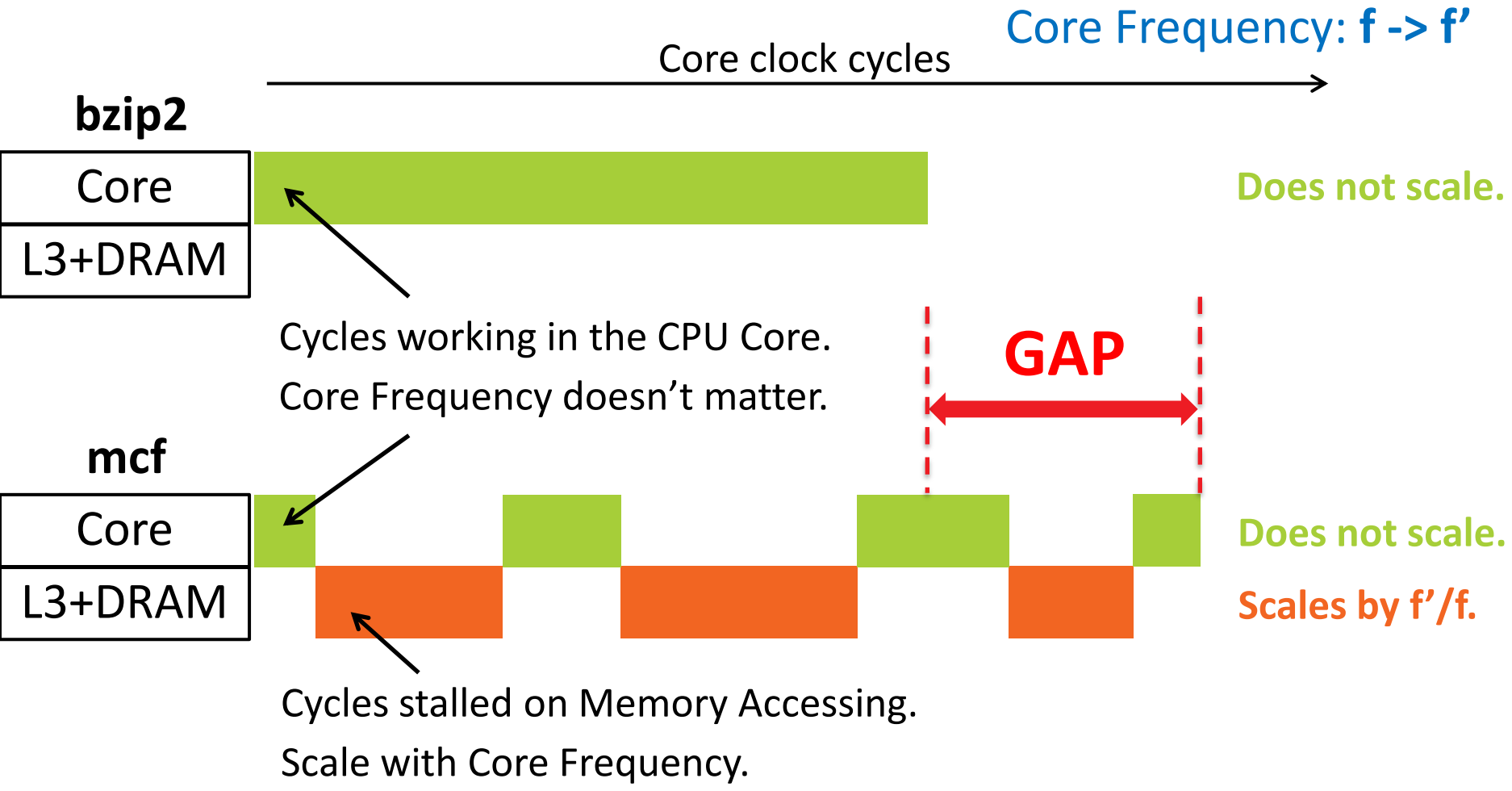
WHAT MAKES THE CPI DIFFERENCE?

AMD

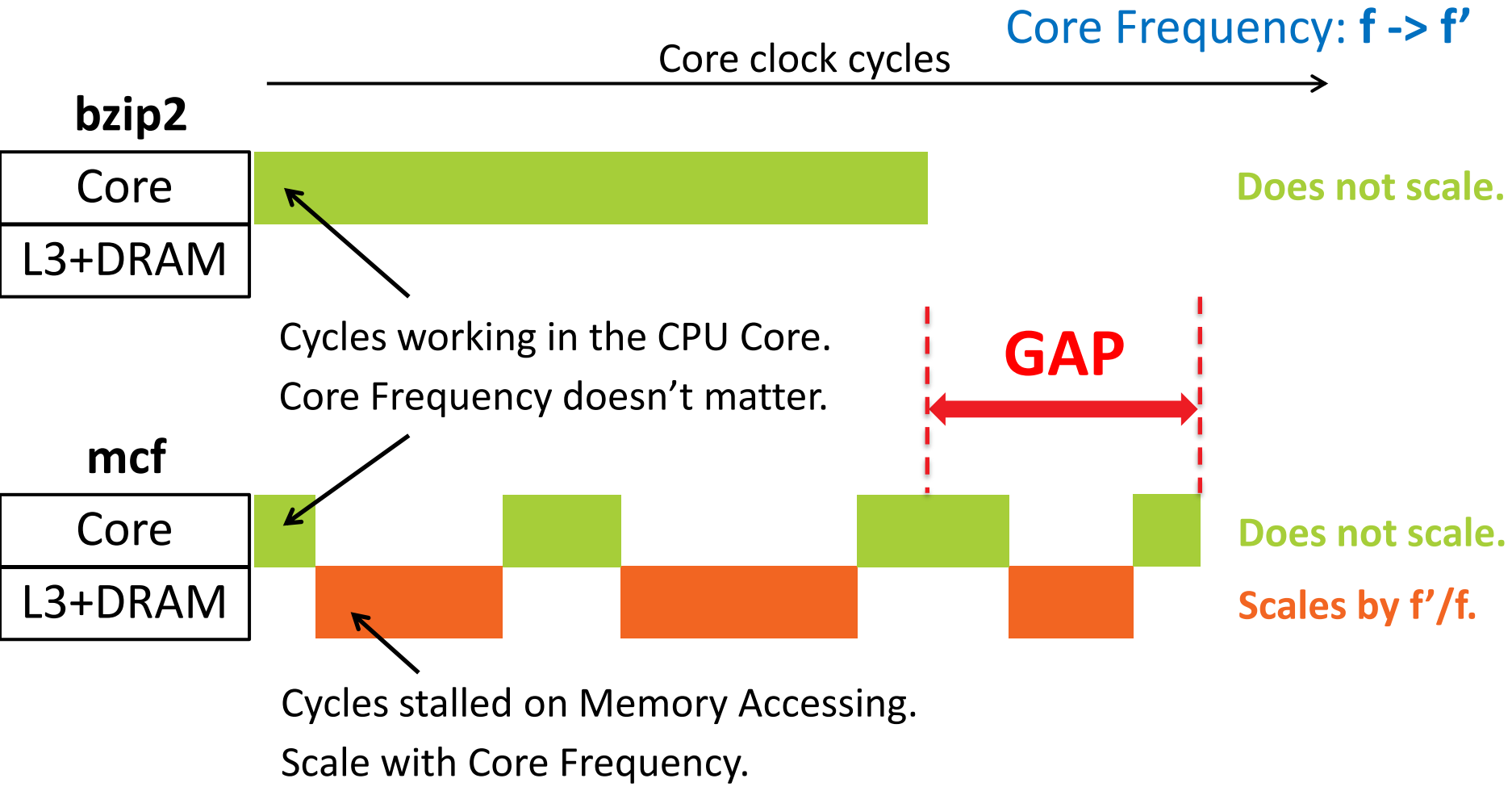
Core Frequency: **f -> f'**



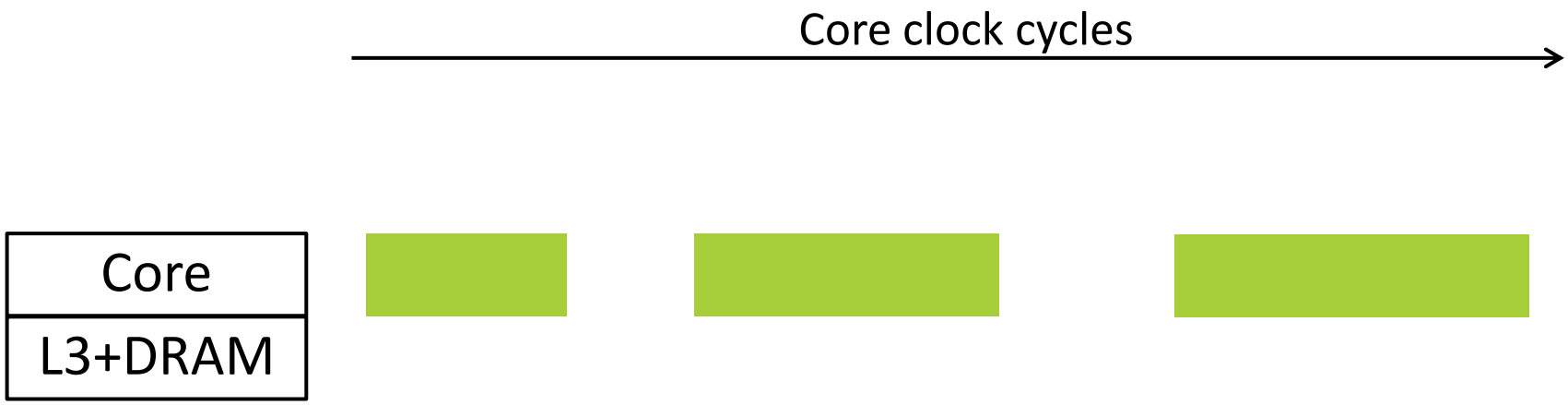
WHAT MAKES THE CPI DIFFERENCE?

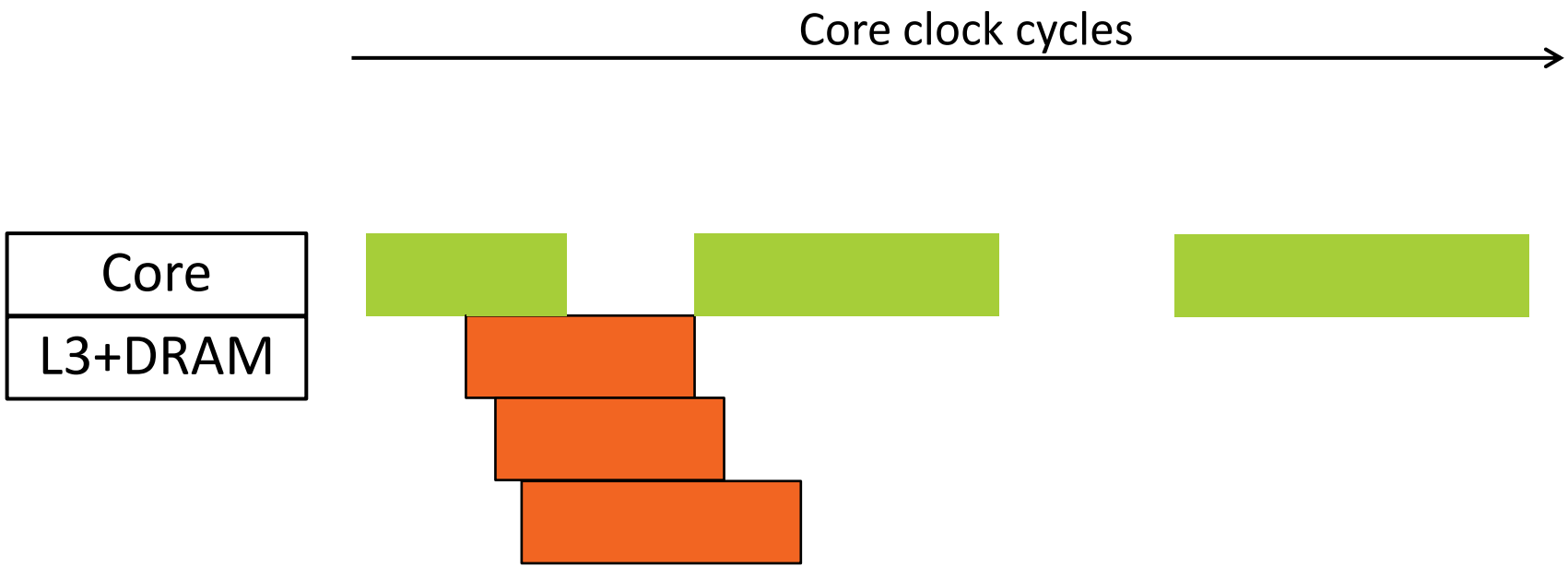


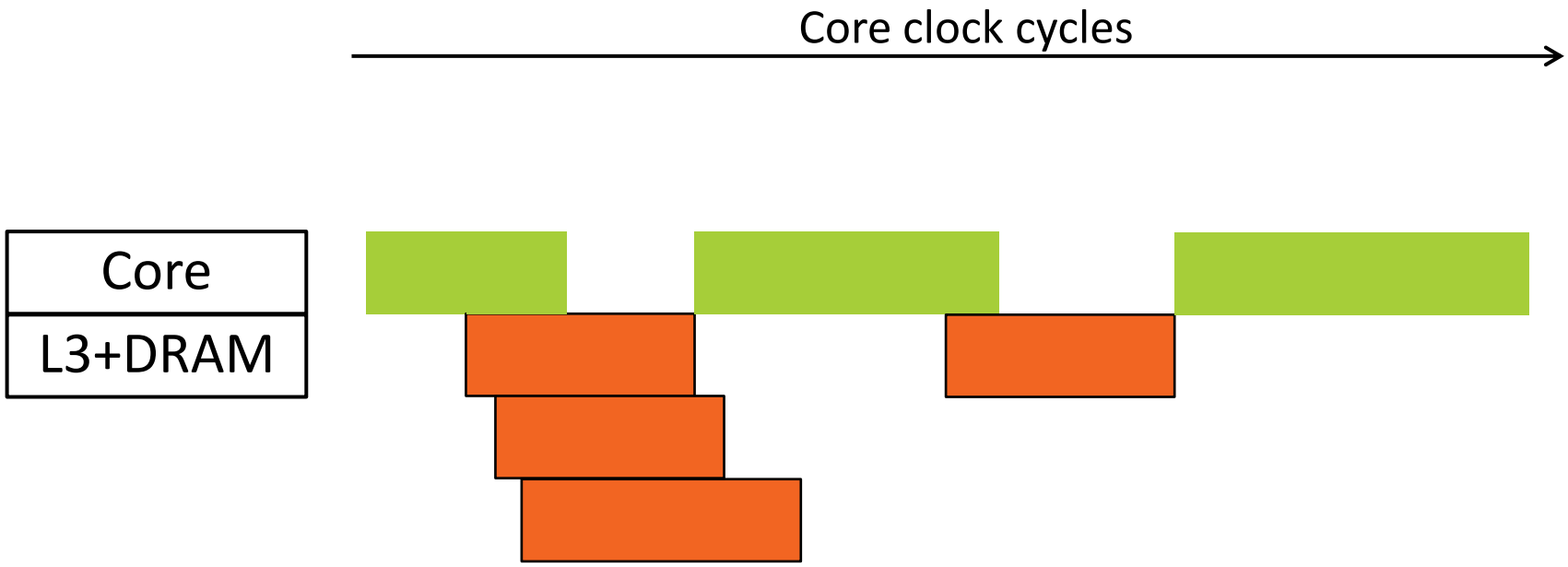
WHAT MAKES THE CPI DIFFERENCE?

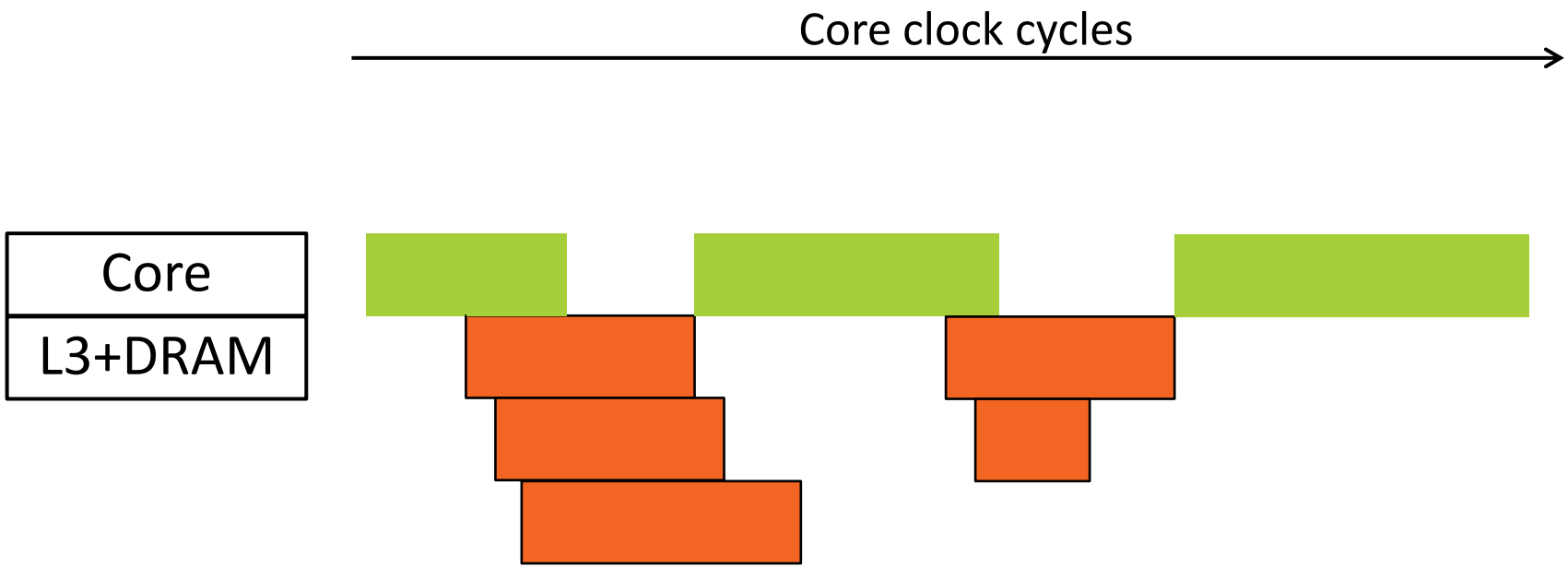


Key: Memory Stall Cycles estimation





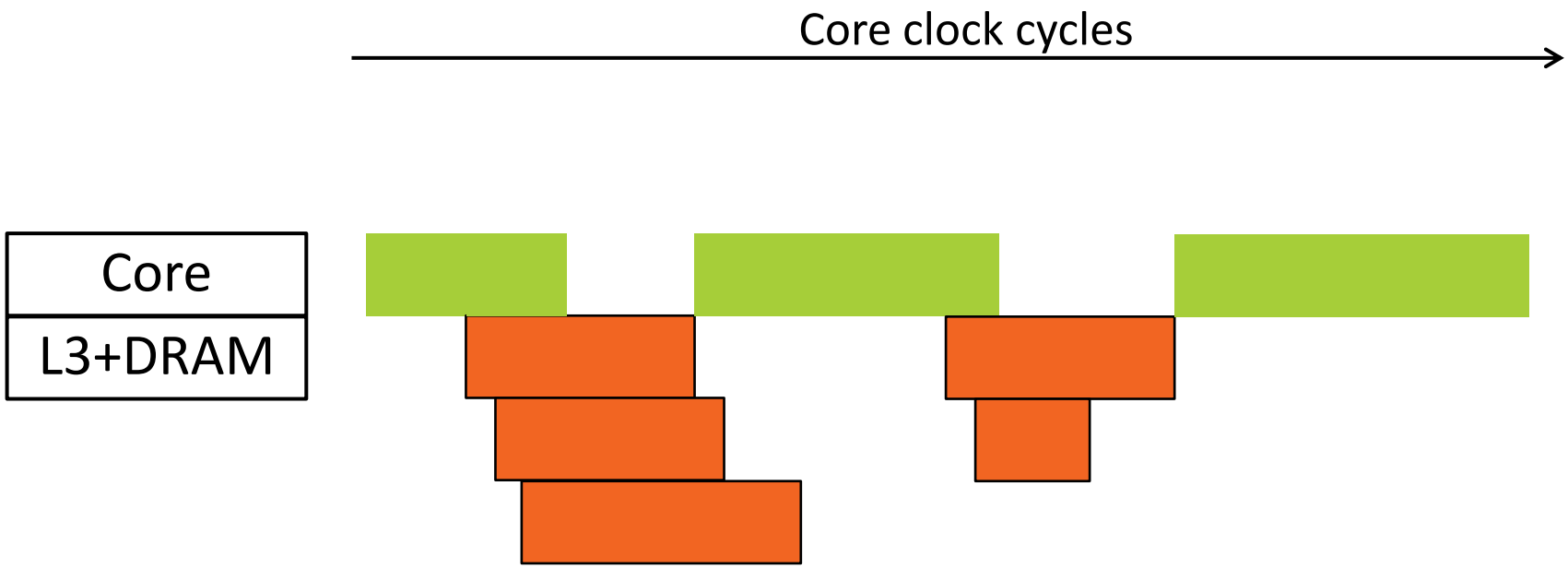




“LEADING LOADS” MEMORY TIME ESTIMATION

AMD

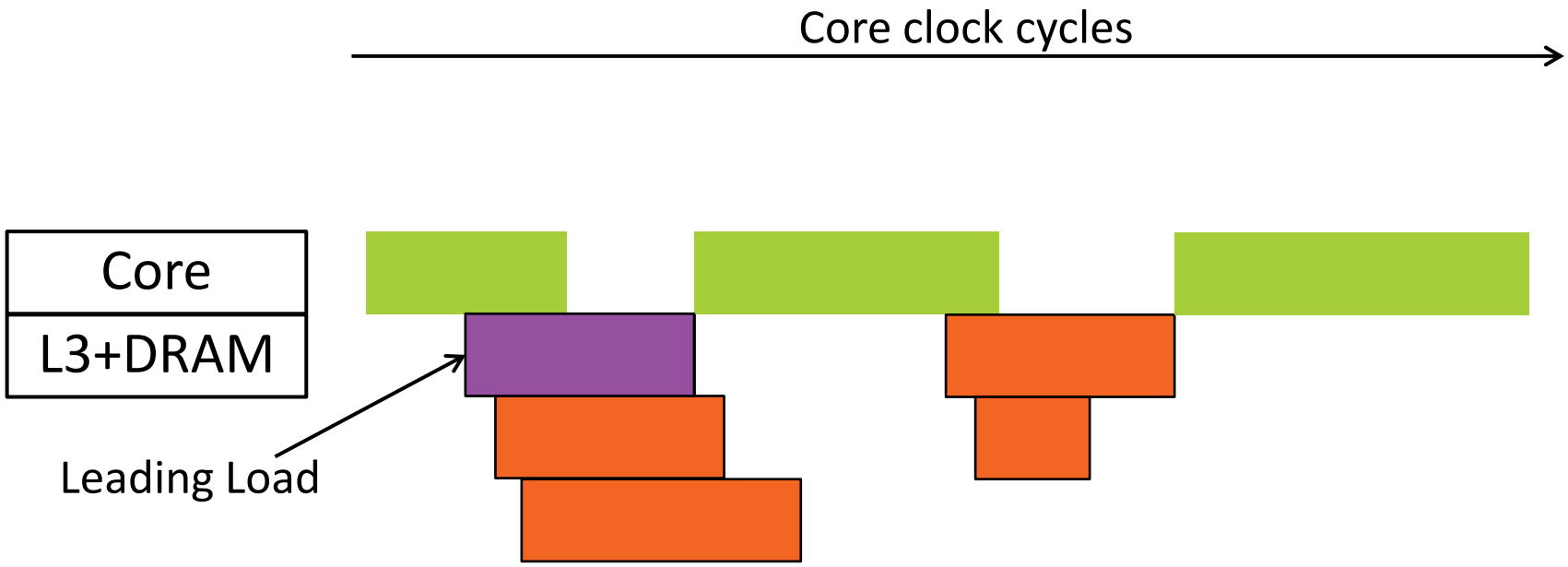
[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



“LEADING LOADS” MEMORY TIME ESTIMATION

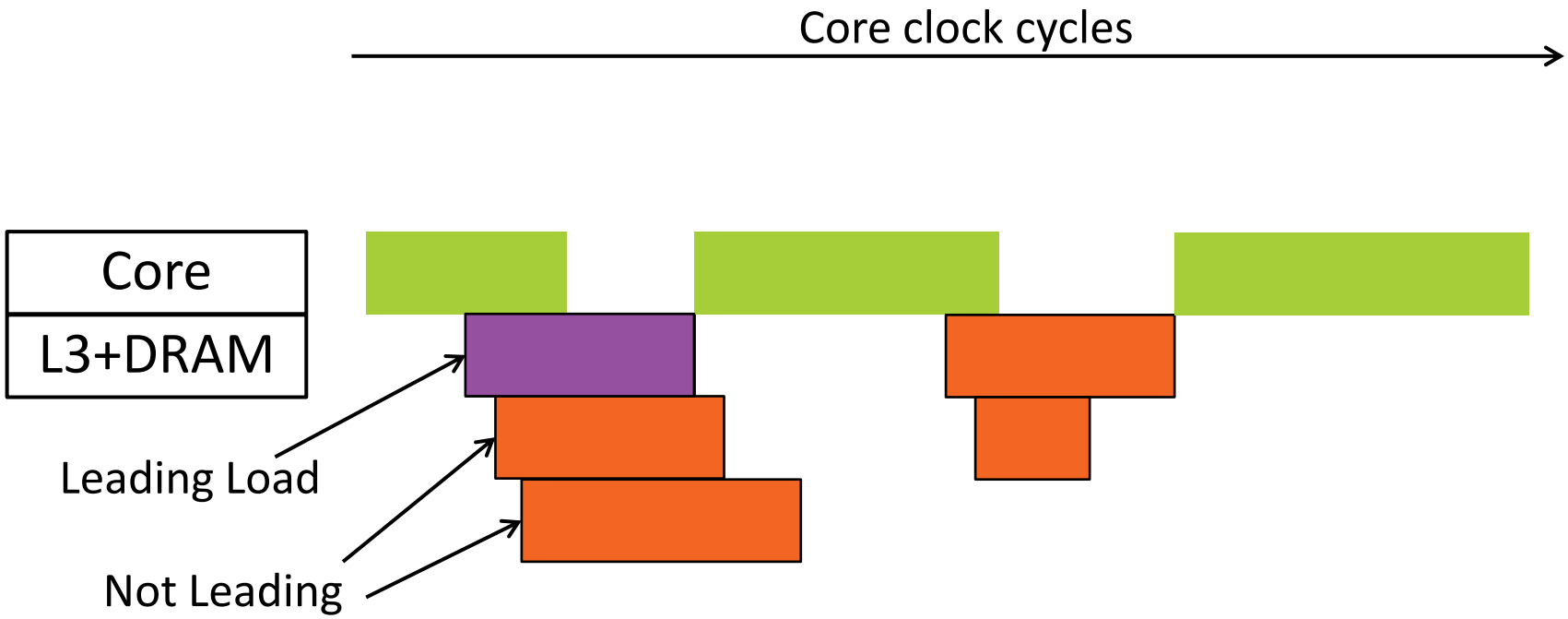
AMD

[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



“LEADING LOADS” MEMORY TIME ESTIMATION

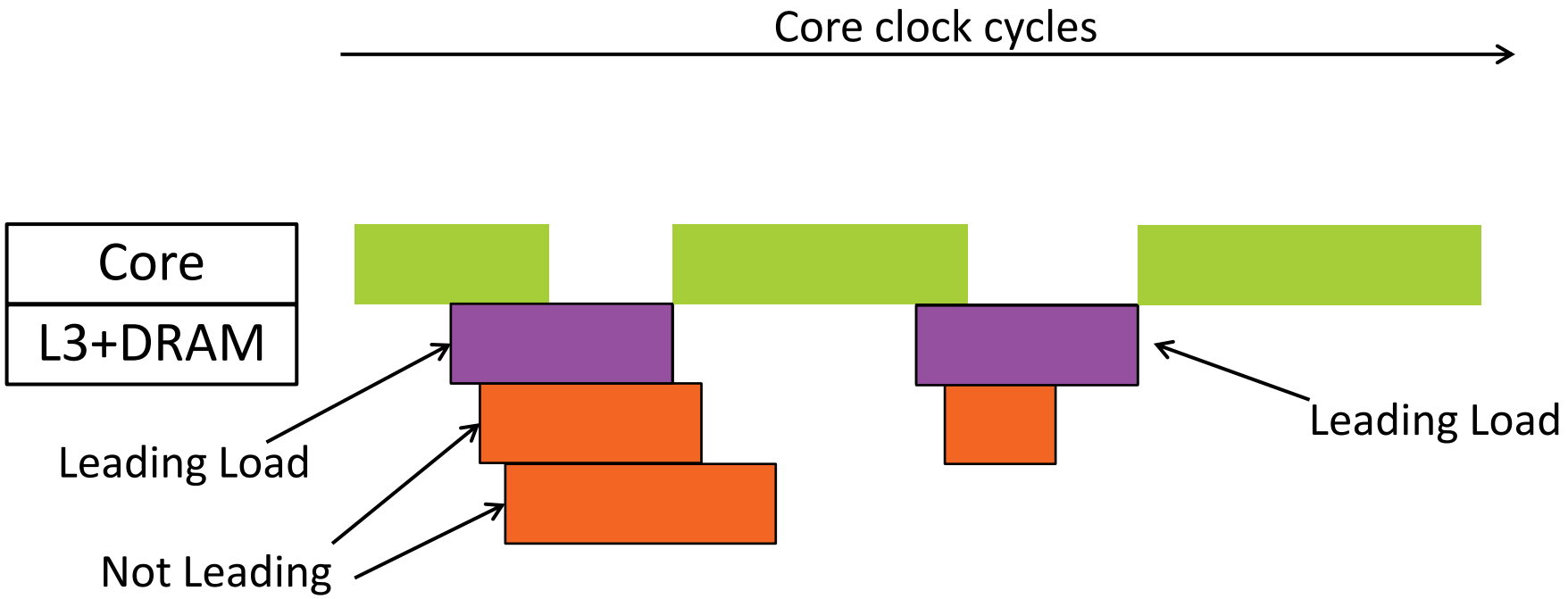
[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



“LEADING LOADS” MEMORY TIME ESTIMATION

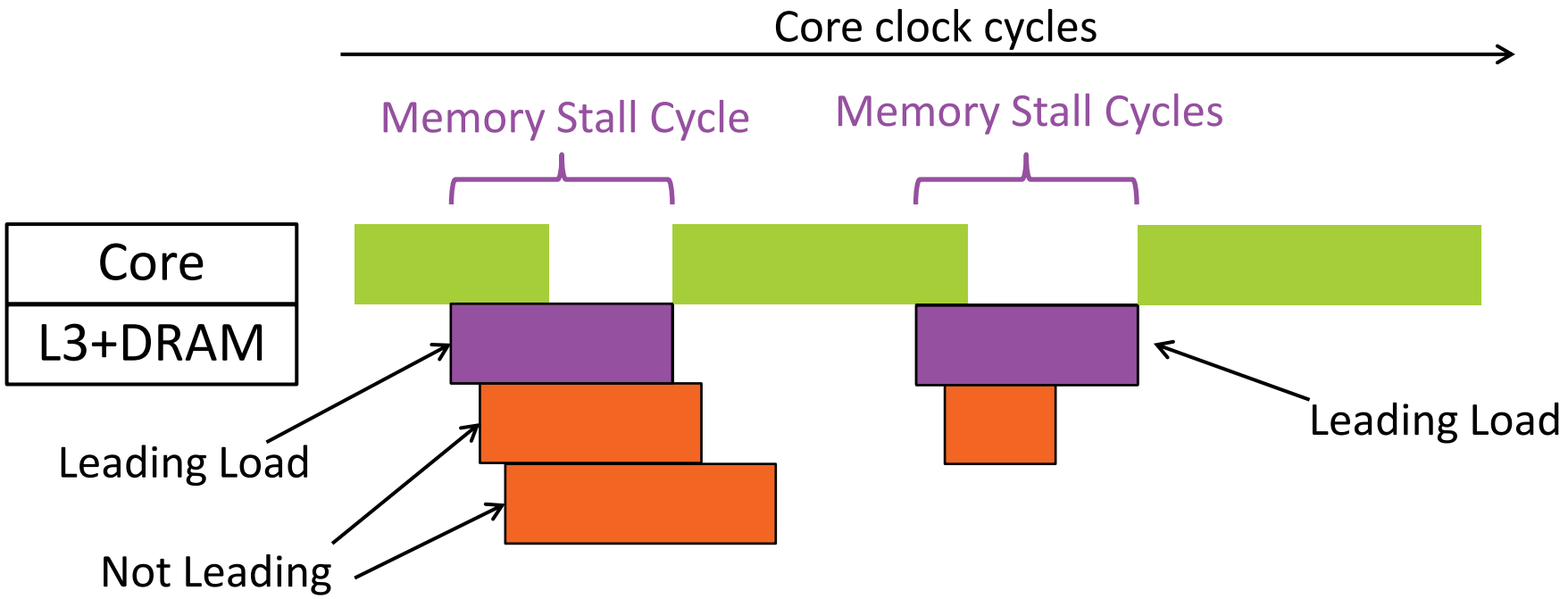
AMD

[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



“LEADING LOADS” MEMORY TIME ESTIMATION

[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



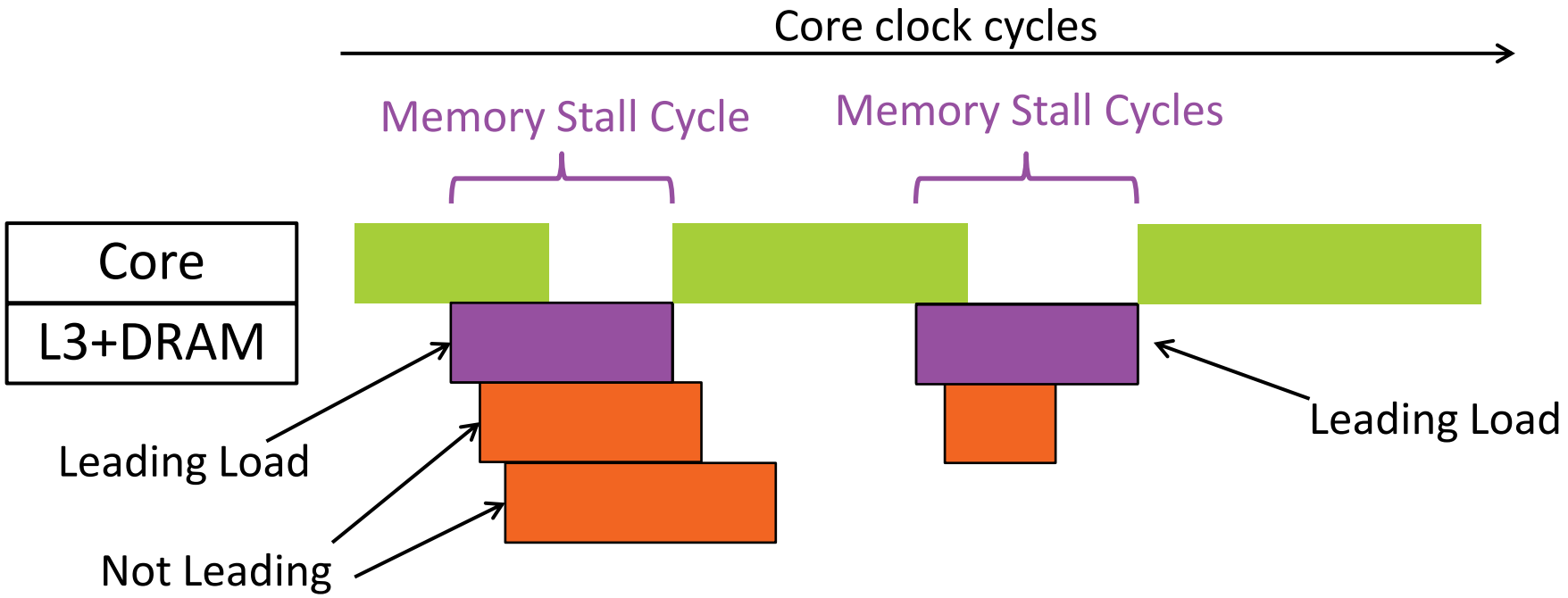
Leading Loads Model:

Memory stall cycles = cycles that leading loads are active

“LEADING LOADS” MEMORY TIME ESTIMATION

AMD

[CF’10 Keramidas+], [TOC’10 Eyerman+], and [IGCC’11 Rountree+]



Leading Loads Model:

Memory stall cycles = cycles that leading loads are active

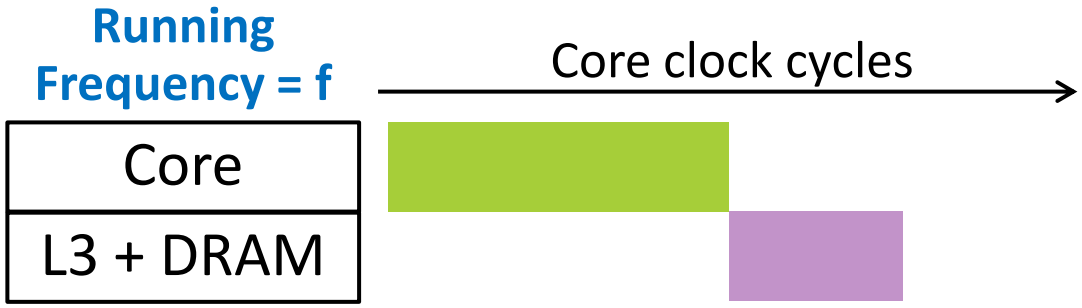
AMD hardware has MAB0 performance counter which can estimate Leading Loads cycles. [USENIX-ATC’14 Su+]

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

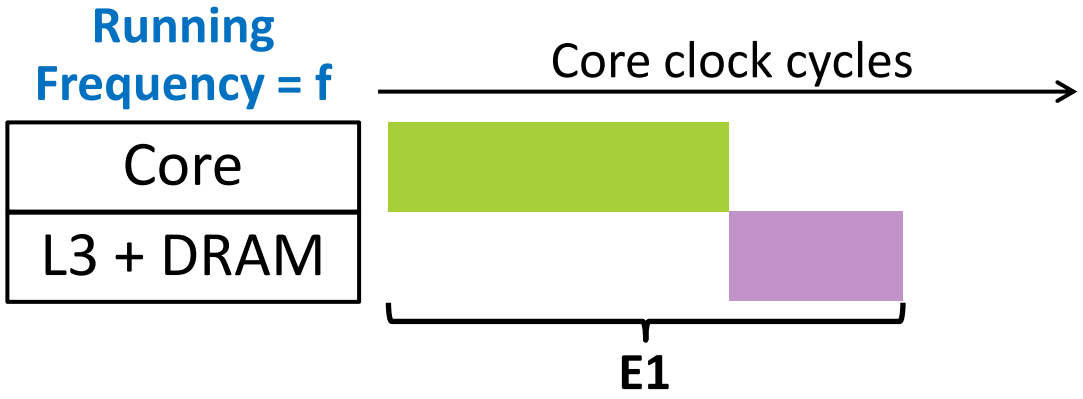


CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

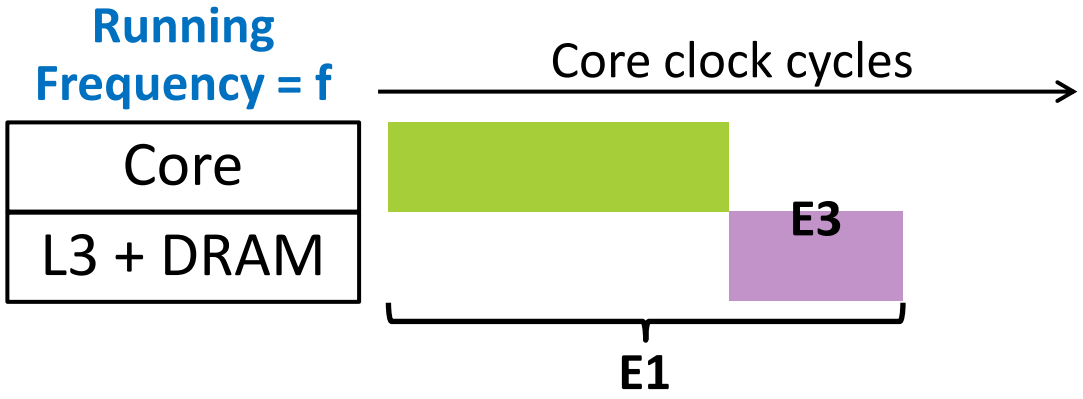


CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

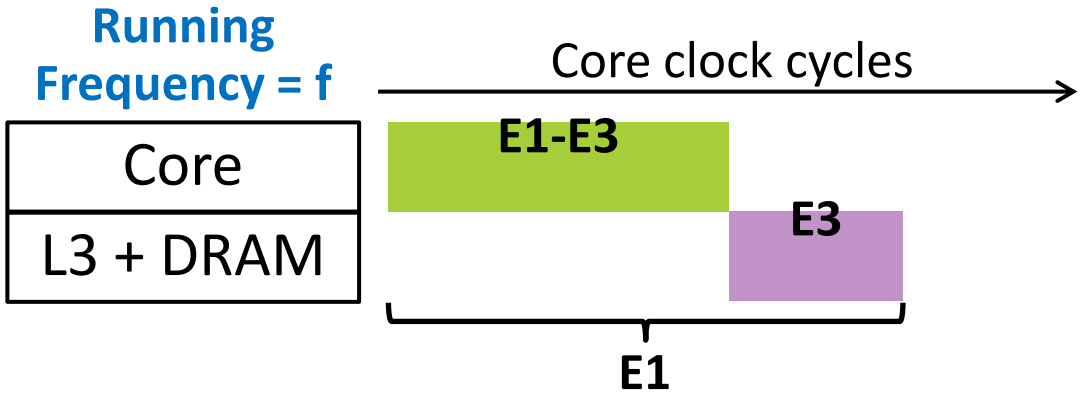


CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

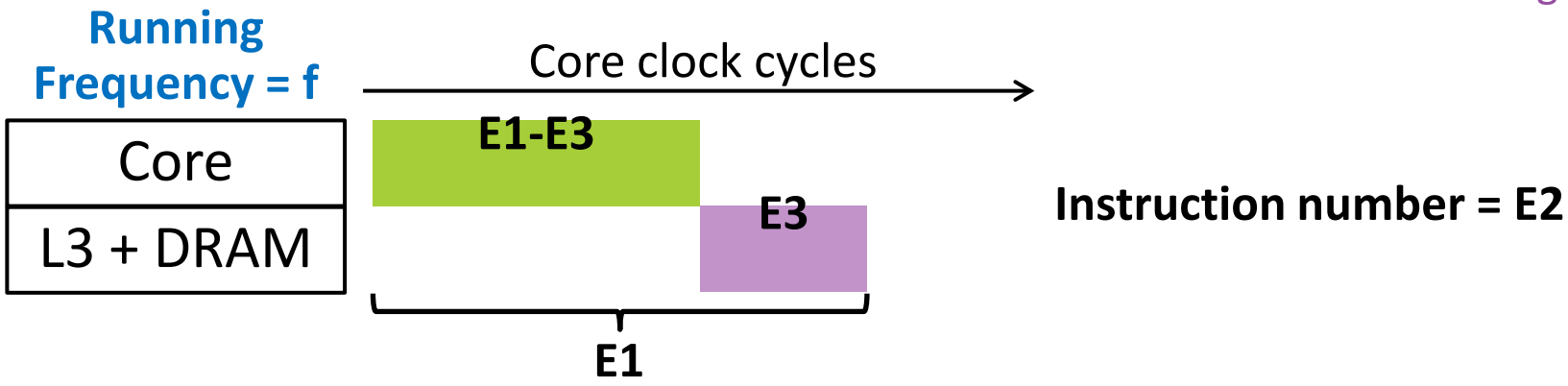


CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



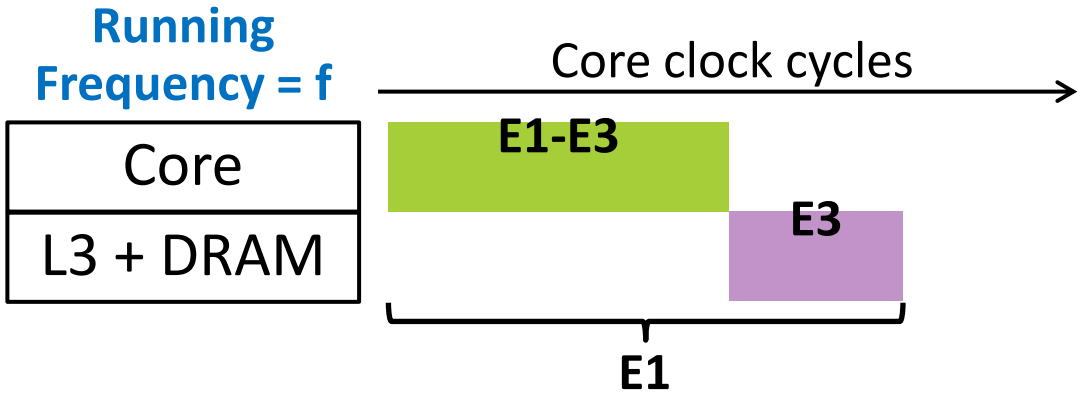
Instruction number = E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

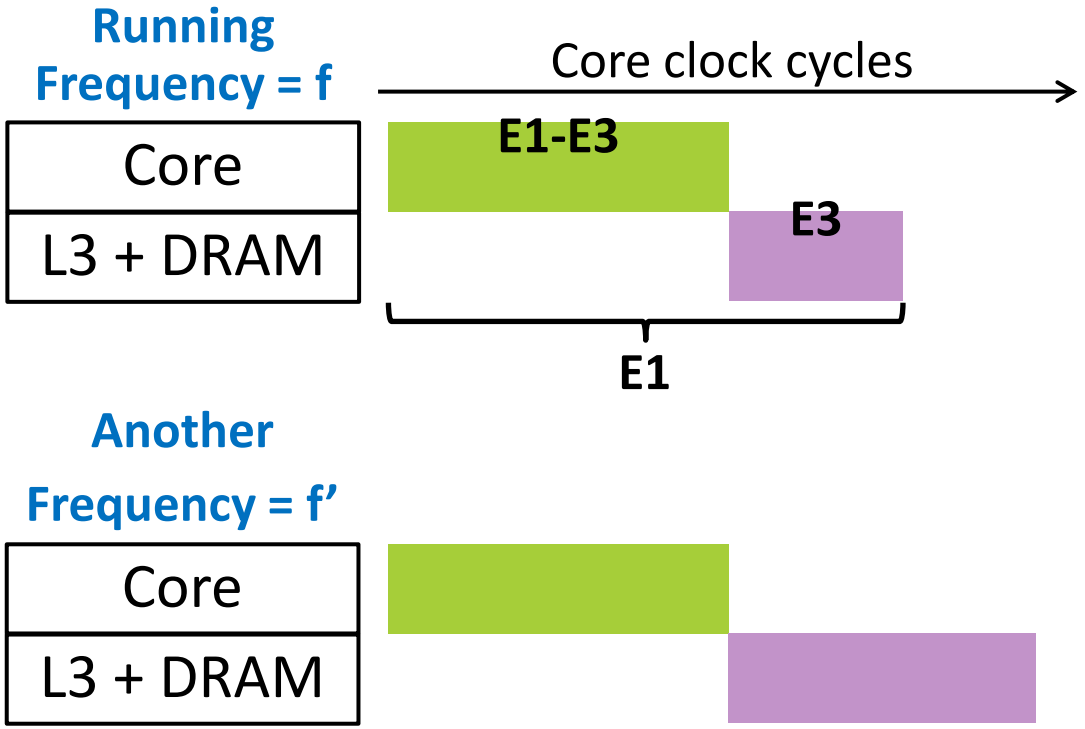
CPI(f) = E1/E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

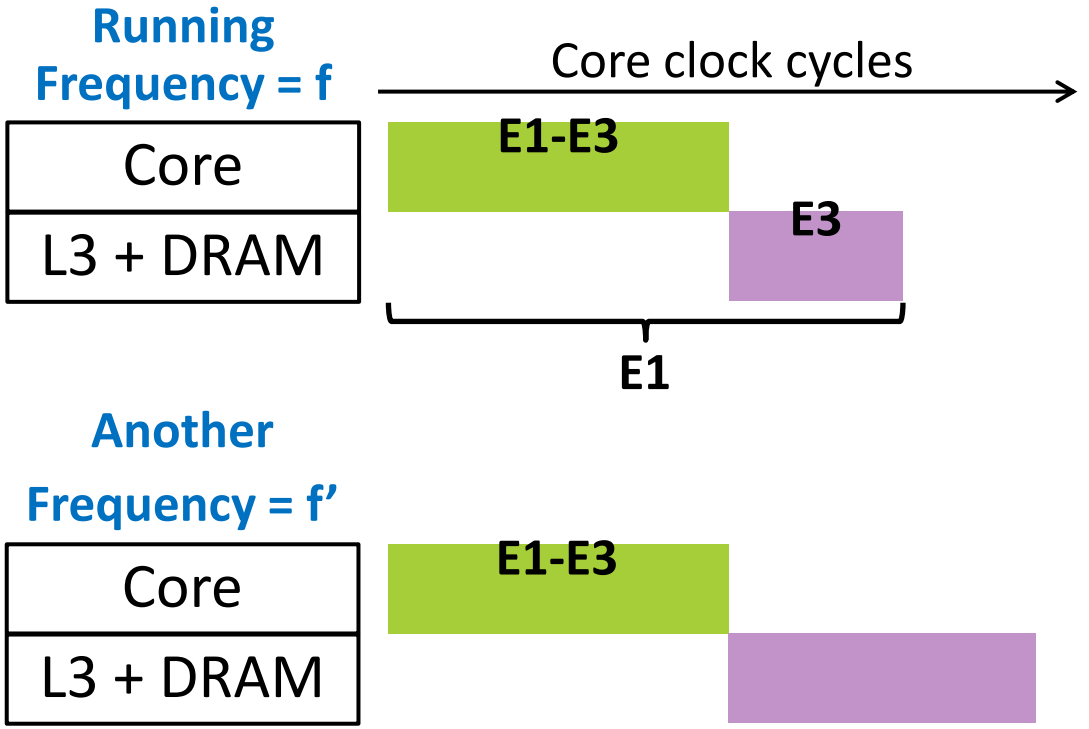
CPI(f) = E1/E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

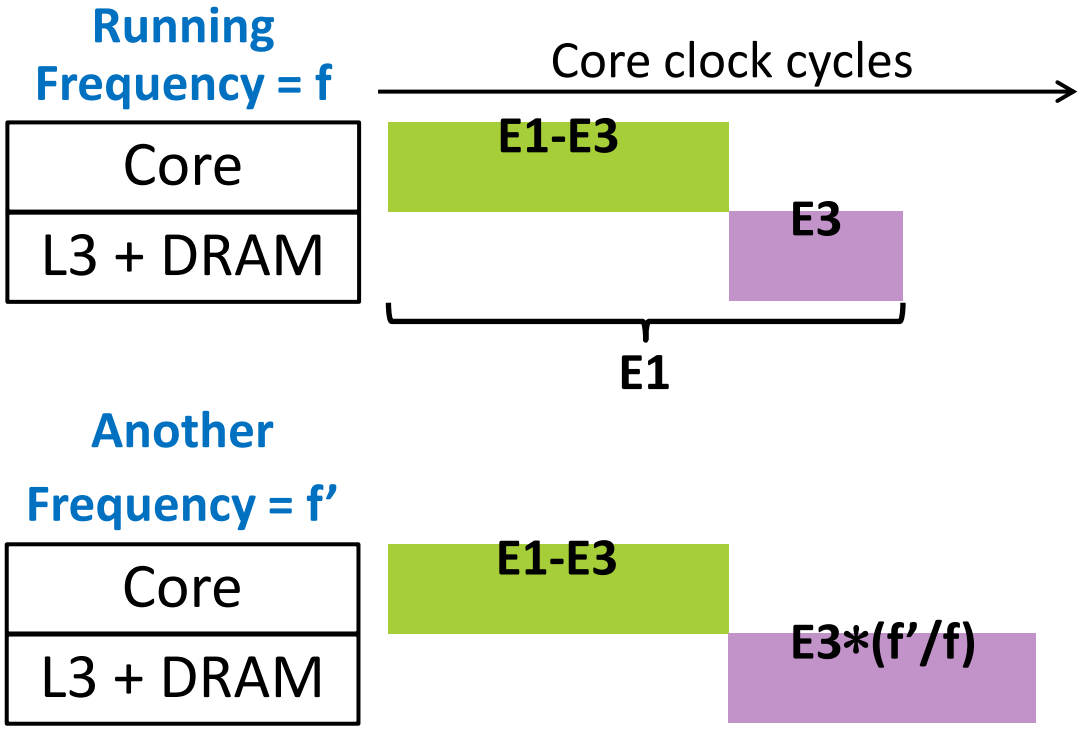
CPI(f) = E1/E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

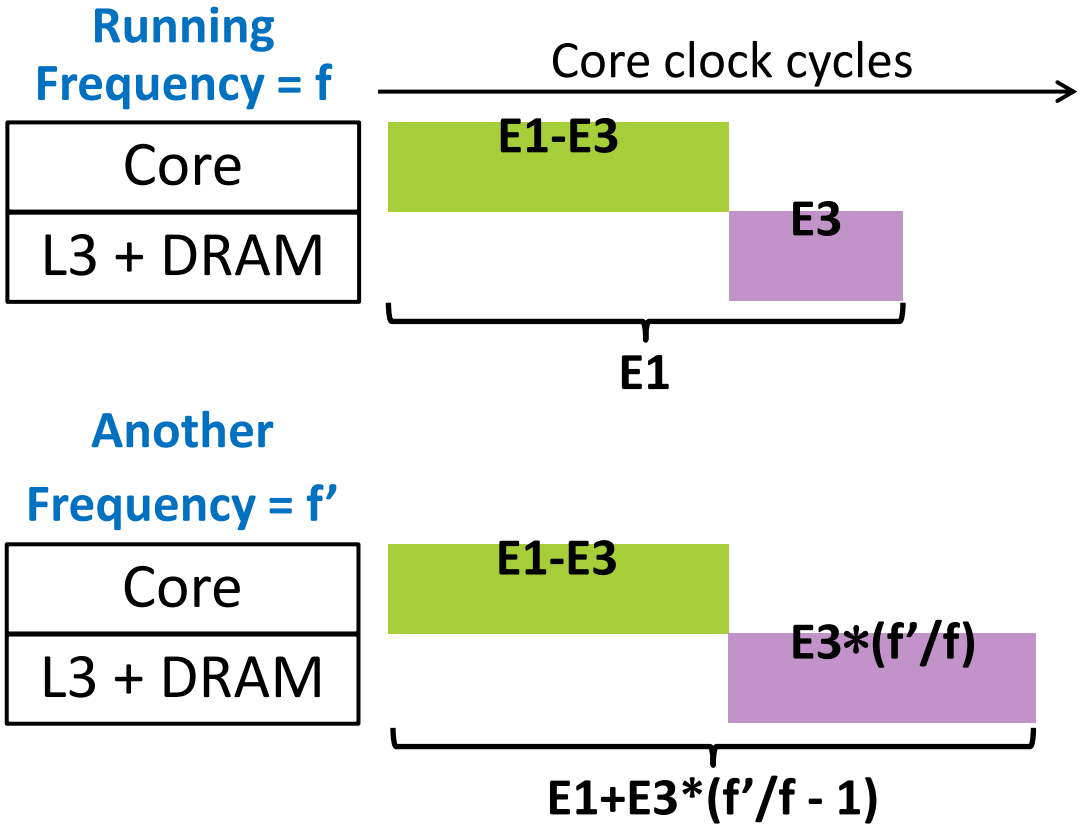
CPI(f) = E1/E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

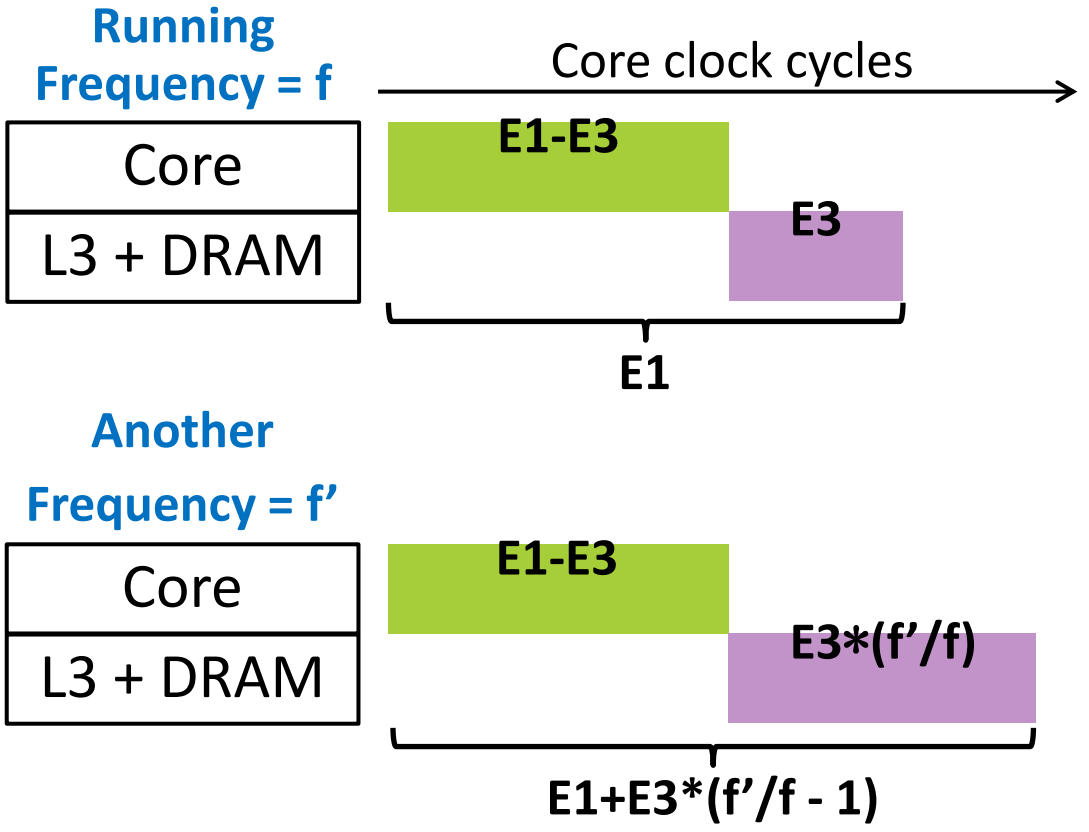
CPI(f) = E1/E2

CPI PREDICTOR

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

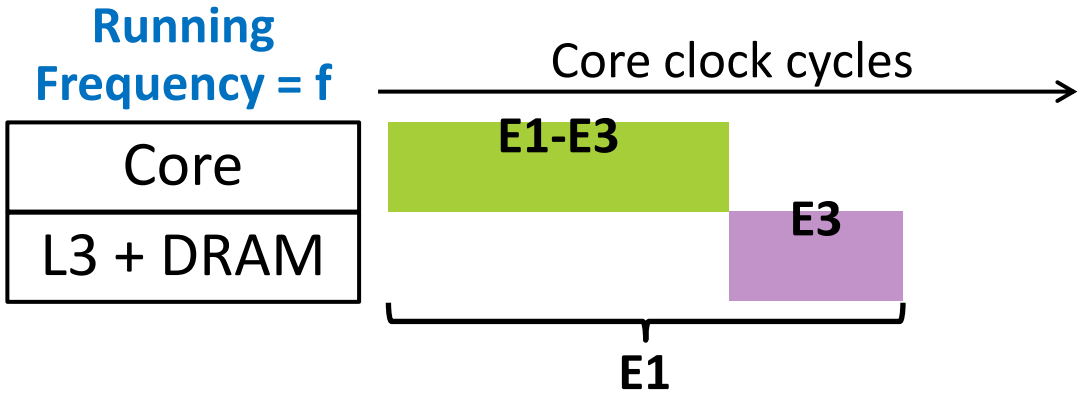


CPI PREDICTOR

- 3 HW events

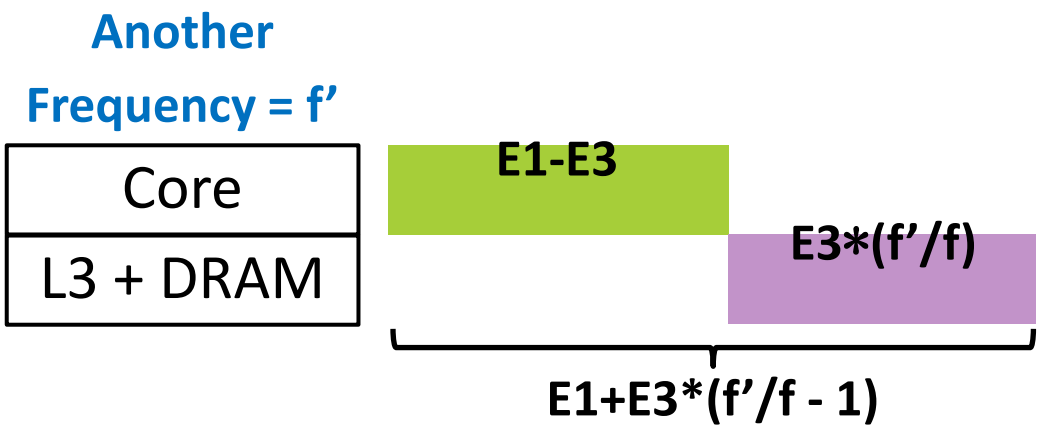
#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

CPI(f) = E1/E2



Instruction number = E2

CPI(f') = (E1+E3*(f'/f - 1))/E2

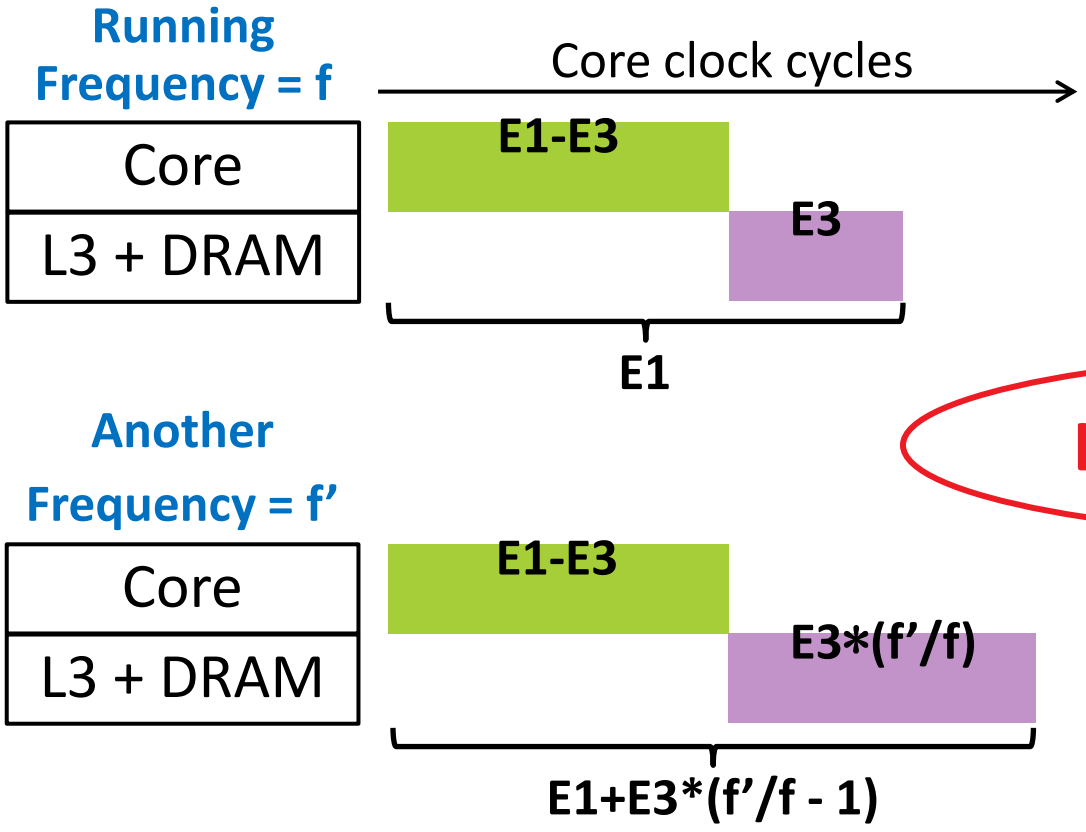
CPI PREDICTOR

AMD

- 3 HW events

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles



Instruction number = E2

CPI(f) = E1/E2

LL-MAB CPI Predictor

Instruction number = E2

CPI(f') = (E1+E3*(f'/f - 1))/E2

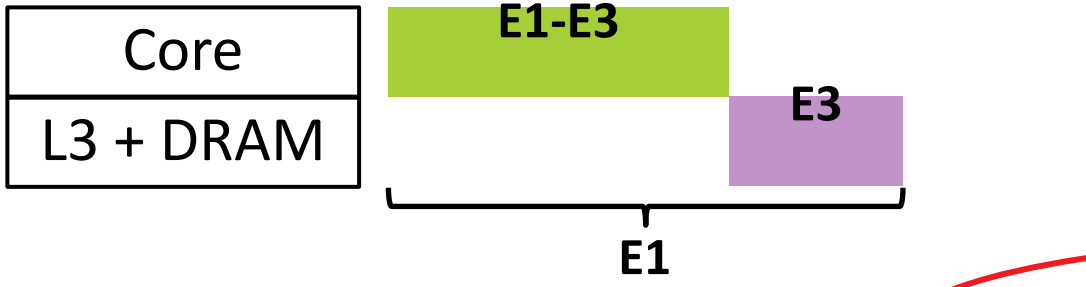
CPI PREDICTOR

- 3 HW events
- Step=200ms

#	Code	Event Name
E1	0x76	CPU_Clocks_not_Halted
E2	0xc0	Retired_Instructions
E3	0x69	MAB0_Wait_Cycles

MAB based Leading Loads Cycles

Running Frequency = f

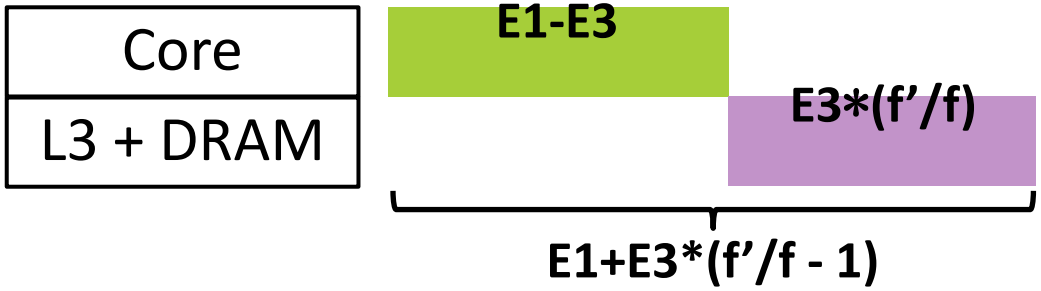


Instruction number = E2

CPI(f) = E1/E2

LL-MAB CPI Predictor

Another Frequency = f'

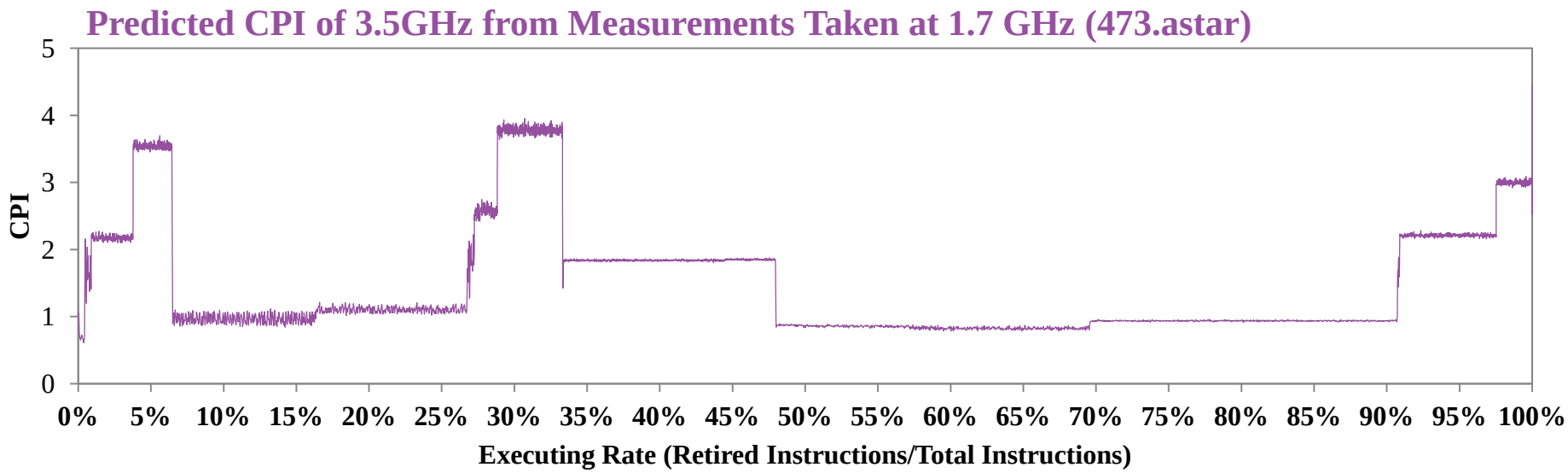


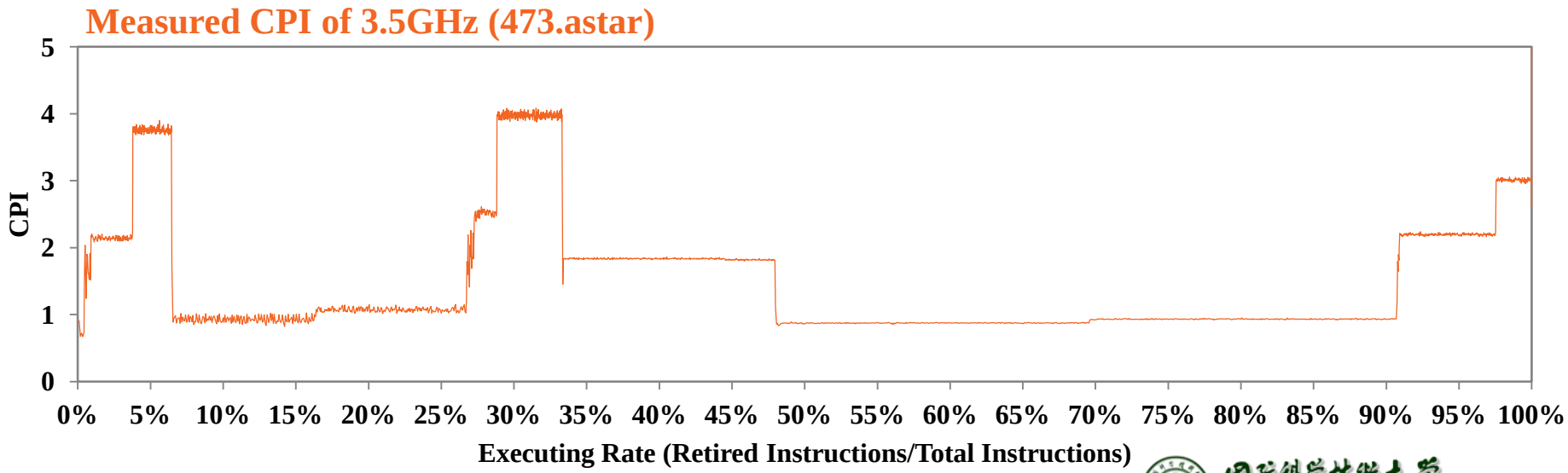
Instruction number = E2

CPI(f') = (E1+E3*(f'/f - 1))/E2

PREDICTED CPI CURVE V.S. MEASURED CPI CURVE

1.7GHZ (VF2) -> 3.5GHZ (VF5)

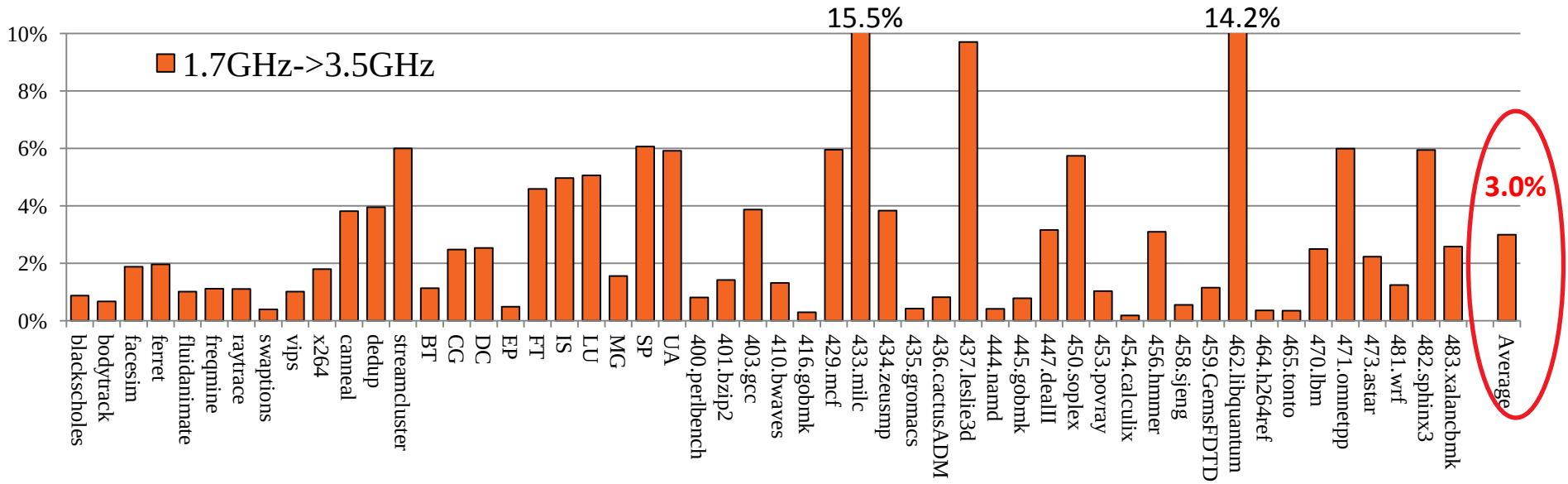




National University of Defense Technology

CPI PREDICTION ERROR

- FX-8320 Processor
- All 52 benchmarks in SPEC, PARSEC, and NPB
- 1.7GHz (VF2) -> 3.5GHz (VF5)



OUTLINE

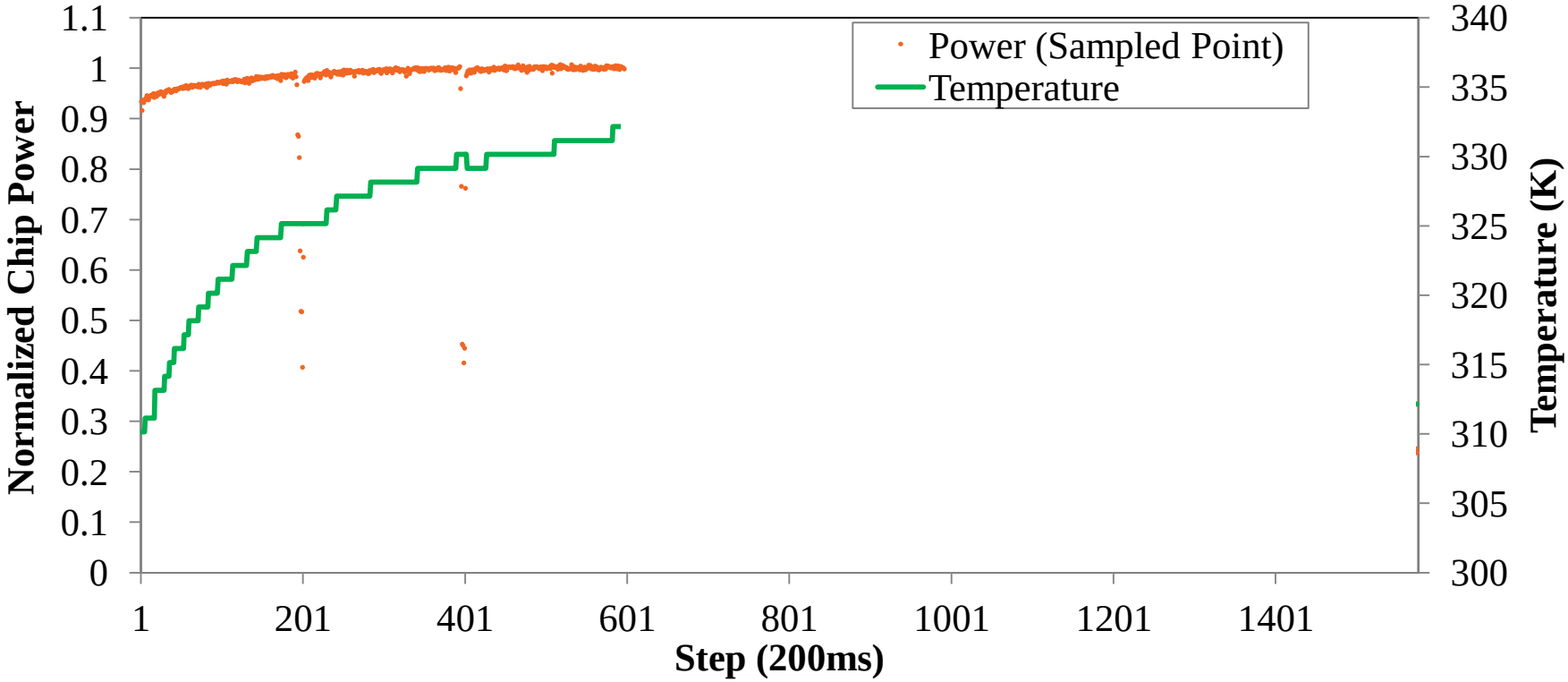
AMD

- Background
- Experimental Methodology
- Performance Prediction Across DVFS States
- **Power Prediction Across DVFS States**
- PPEP Framework
- Conclusion

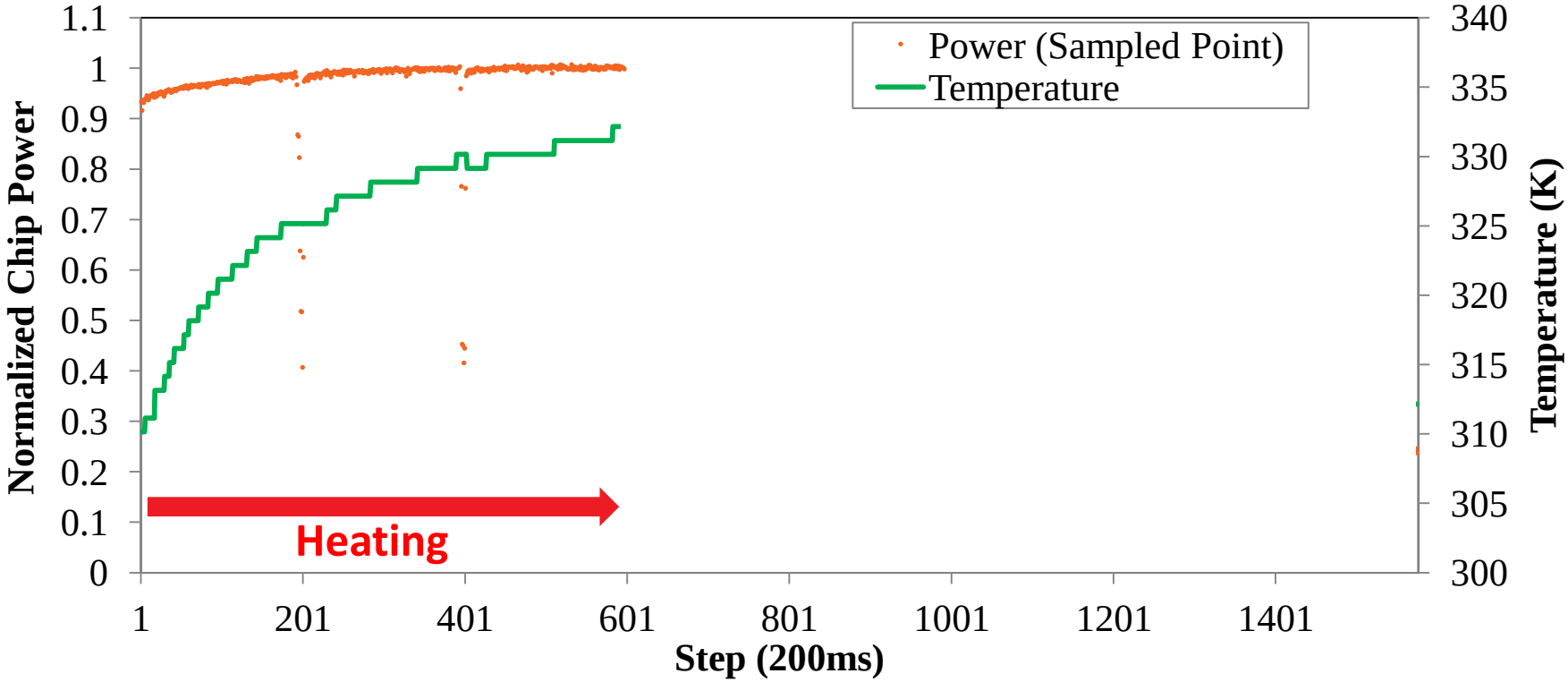
Q2: How does the application's power change with the VF state?

IDLE POWER MODEL + DYNAMIC POWER MODEL

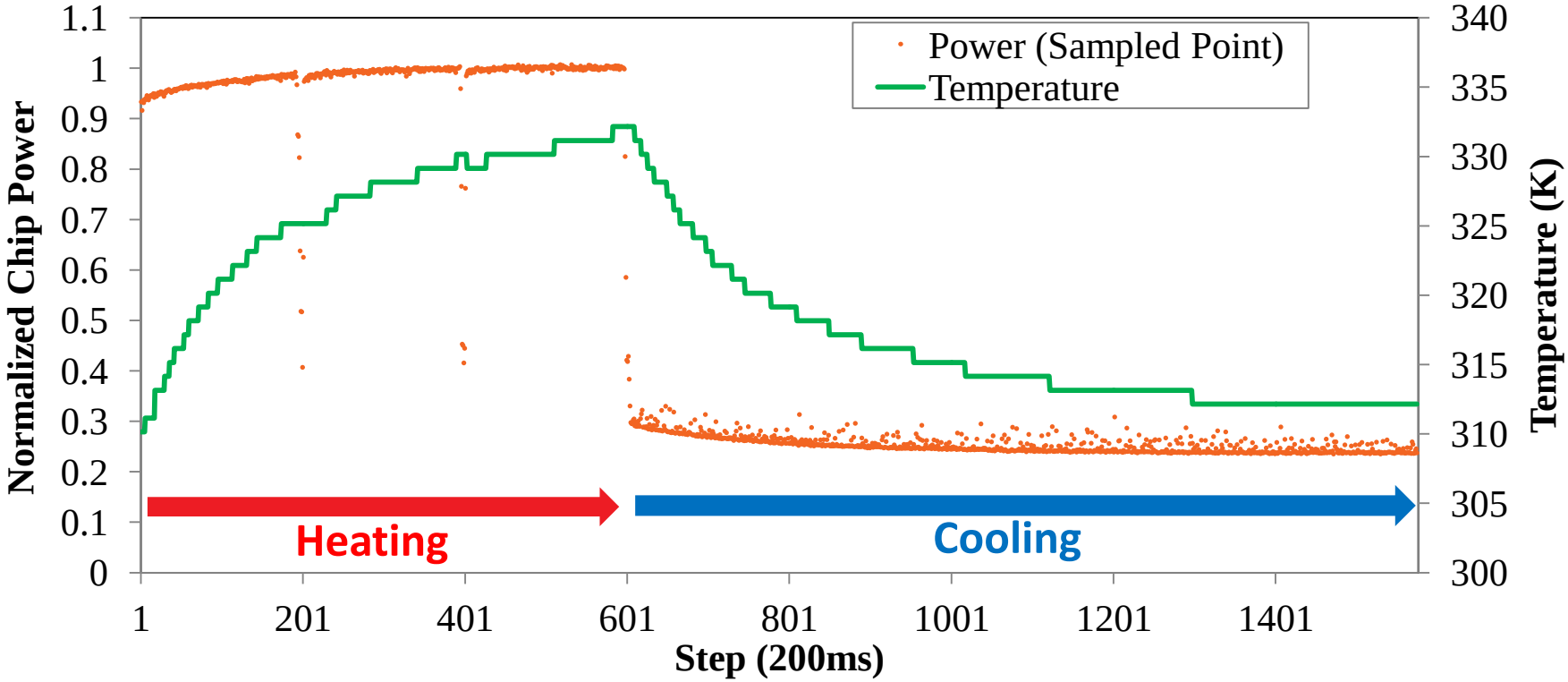
IDLE POWER MODEL + DYNAMIC POWER MODEL



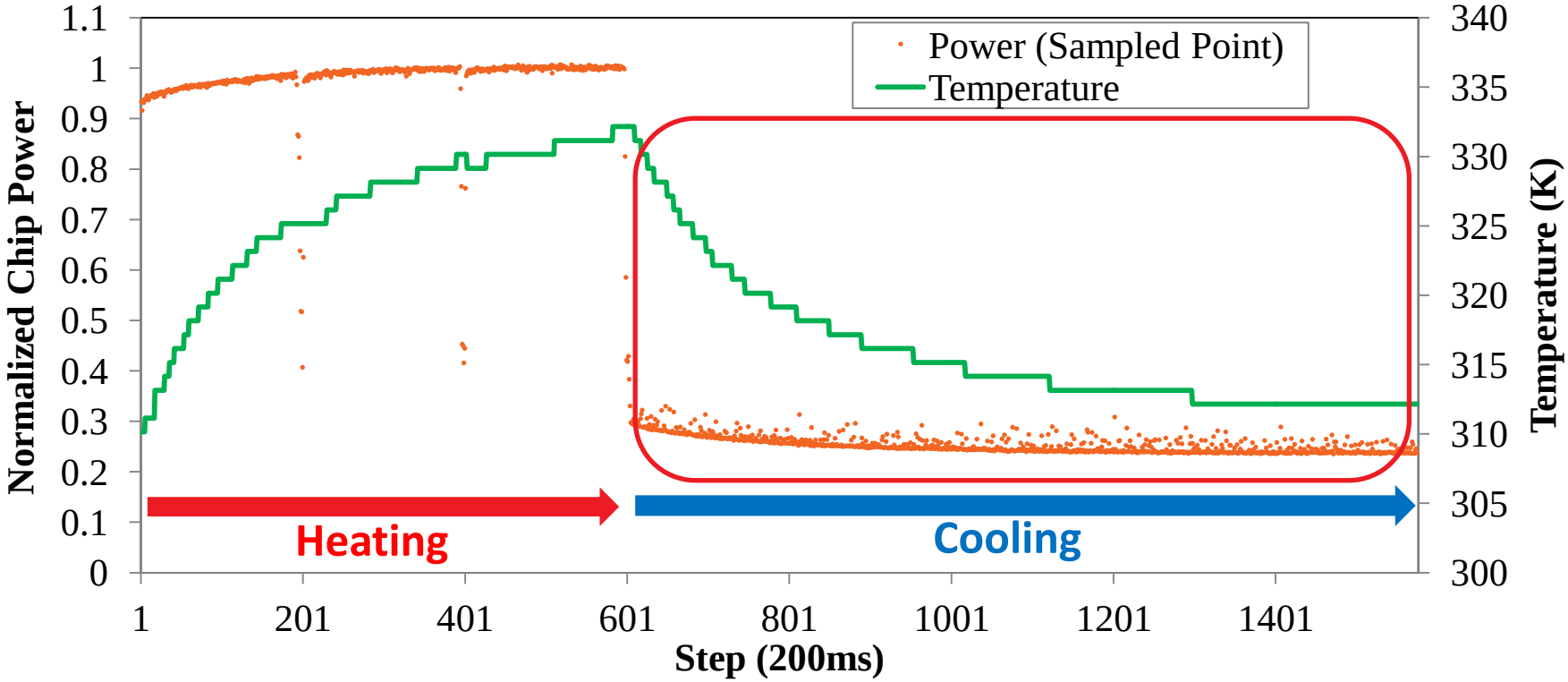
IDLE POWER MODEL + DYNAMIC POWER MODEL



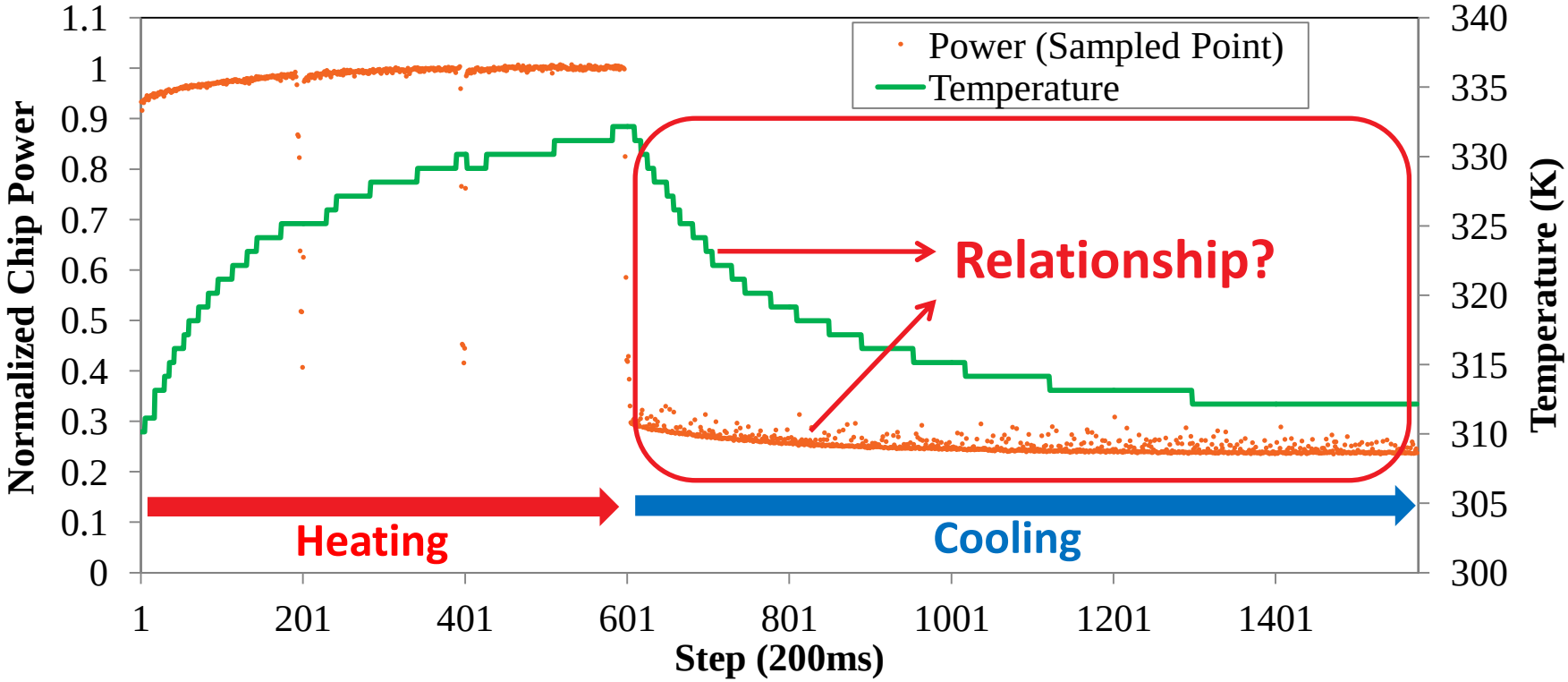
IDLE POWER MODEL + DYNAMIC POWER MODEL



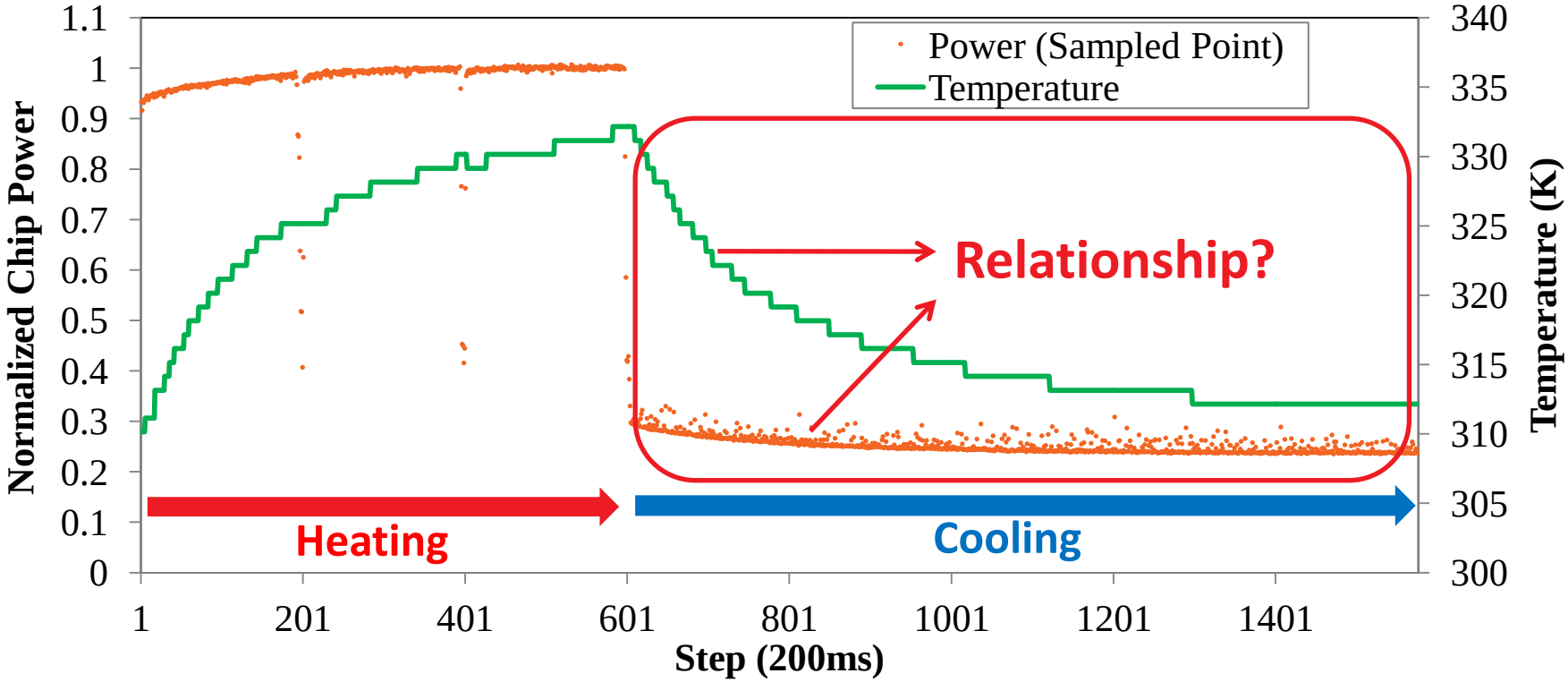
IDLE POWER MODEL + DYNAMIC POWER MODEL



IDLE POWER MODEL + DYNAMIC POWER MODEL



IDLE POWER MODEL + DYNAMIC POWER MODEL



VF State	Voltage(V)	Frequency(GHz)	Error of P_{idle}
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%**

IDLE POWER MODEL + DYNAMIC POWER MODEL

- Chip Dynamic Power: $P_{dyn}(VF_n) = P_{chip} - P_{idle}(VF_n)$
- Use 9 HW Events to Build the Dynamic Power Model

IDLE POWER MODEL + DYNAMIC POWER MODEL

AMD

- Chip Dynamic Power: $P_{dyn}(VF_n) = P_{chip} - P_{idle}(VF_n)$
- Use 9 HW Events to Build the Dynamic Power Model

#	Event	Code
E4	Retired_uOPs	0xc1
E5	FPU_Pipe_Assignment	0x00
E6	L1I$_Fetches	0x80
E7	L1D$_Accesses	0x40
E8	L2$_Requests	0x7d
E9	Retired_Branches	0xc2
E10	Retired_MisBranches	0xc3
E11	L2$_Misses	0x7e
E12	Dispatch_Stalls	0xd1

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

(PS= Per-Second Count Value)

POWER MODEL VALIDATION
EXPERIMENT

- Benchmark Combination
  - Totally 152 combinations
  - Thread number: 1, 2, 3, and 4
    - SPEC: single-/multi- programed
    - PARSEC: single-/multi- threaded
    - NPB: single-/multi- threaded

Number	SPEC	PARSEC	NPB	Total
1-thread	29	13	10	52
2-thread	15	13	10	38
3-thread	10	12	10	32
4-thread	7	13	10	30
Total	61	51	40	**152**

POWER MODEL VALIDATION
EXPERIMENT

AMD

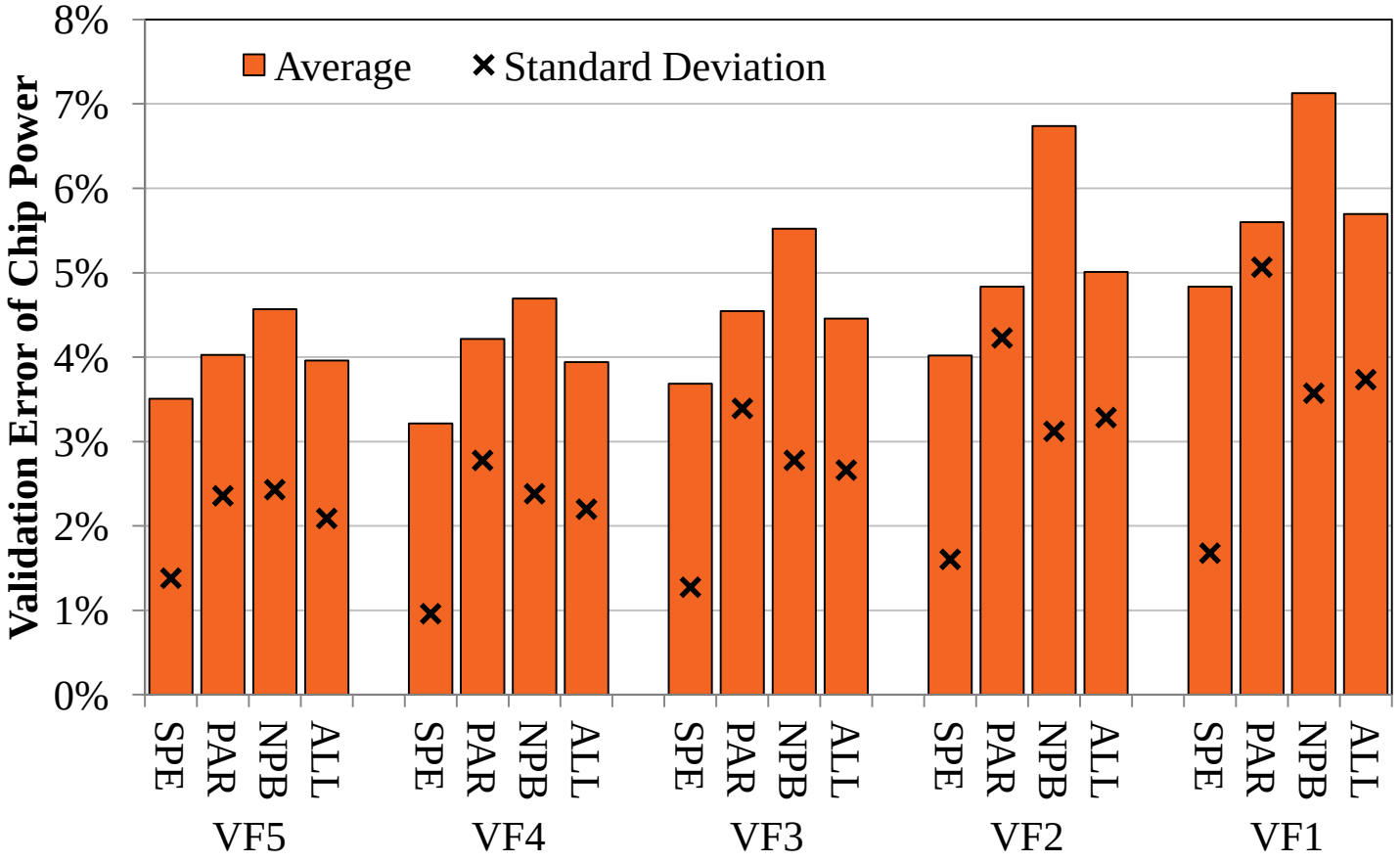
- Benchmark Combination
  - Totally 152 combinations
  - Thread number: 1, 2, 3, and 4
    - SPEC: single-/multi- programed
    - PARSEC: single-/multi- threaded
    - NPB: single-/multi- threaded

Number	SPEC	PARSEC	NPB	Total
1-thread	29	13	10	52
2-thread	15	13	10	38
3-thread	10	12	10	32
4-thread	7	13	10	30
Total	61	51	40	**152**

- 4-fold cross validation
  - Divide 152 benchmark combination in to 4 sets
  - Train on all combinations of 3 sets
  - Test on the remaining set

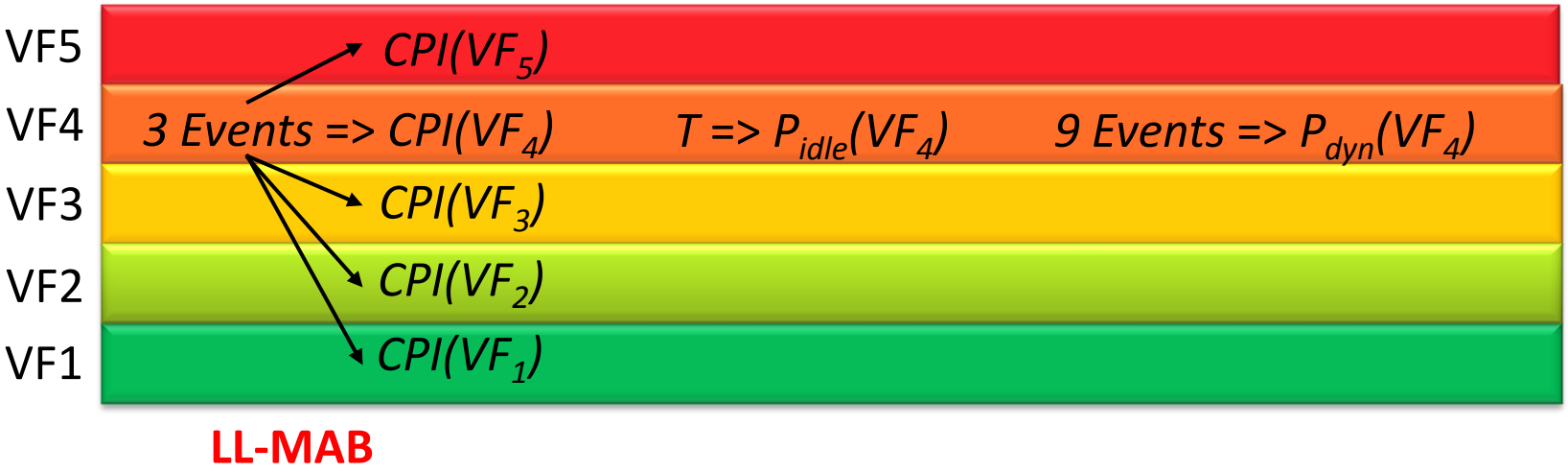
POWER MODEL VALIDATION
(LOWER IS BETTER)

- Error: 4.6%
- Standard Deviation: 2.8%



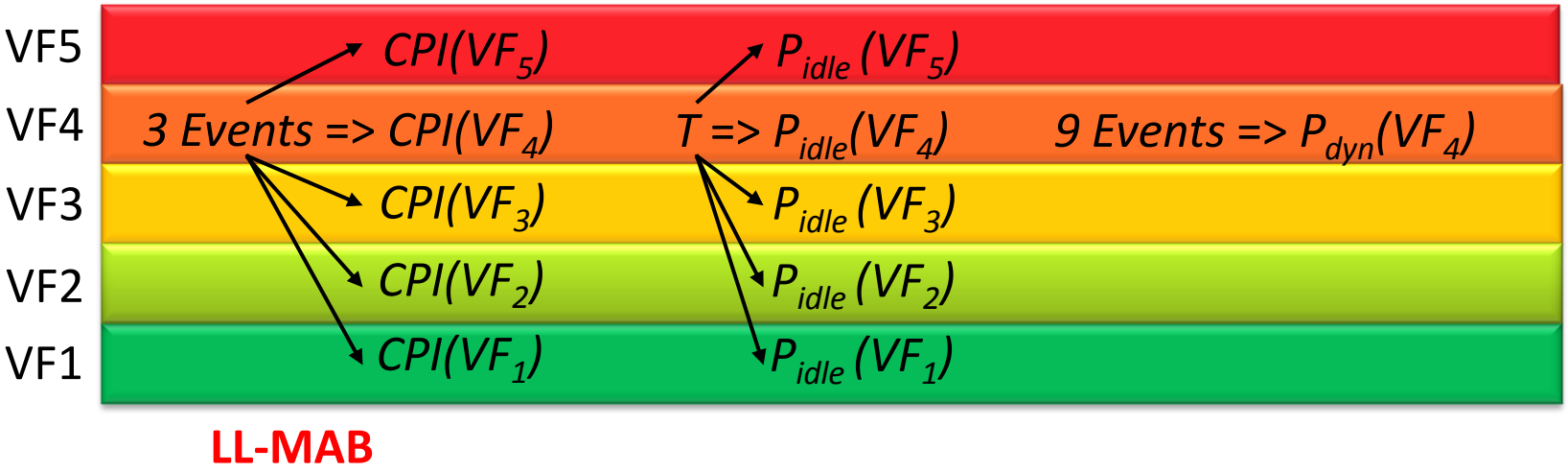
National University of Defense Technology

PREDICTING POWER ACROSS VF STATES



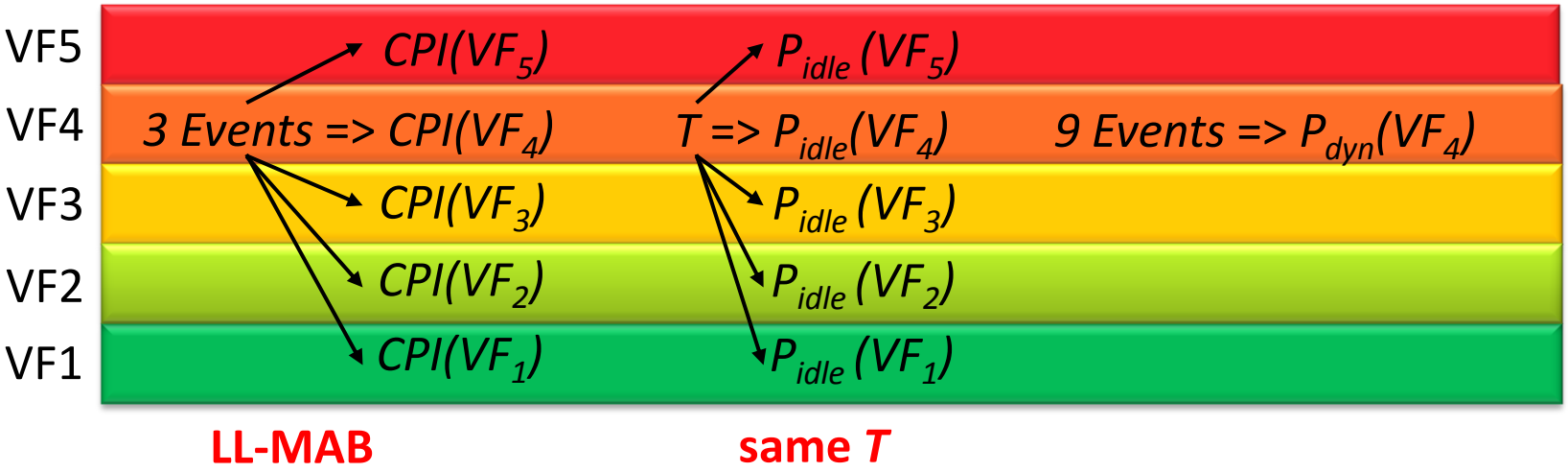
- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction

PREDICTING POWER ACROSS VF STATES



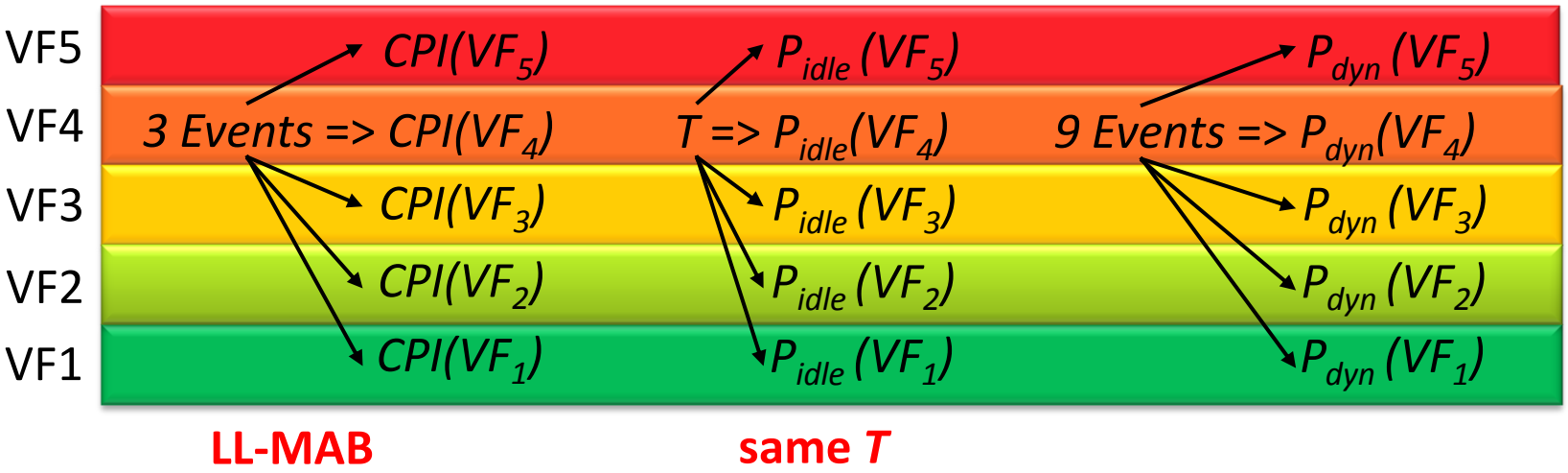
- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction

PREDICTING POWER ACROSS VF STATES



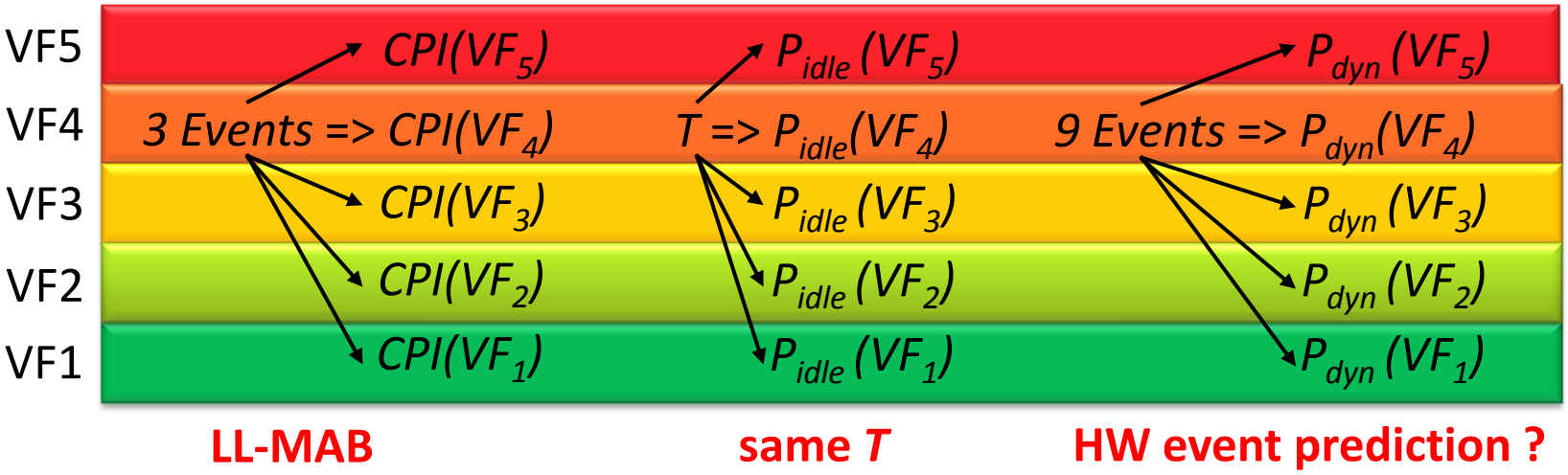
- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction

PREDICTING POWER ACROSS VF STATES

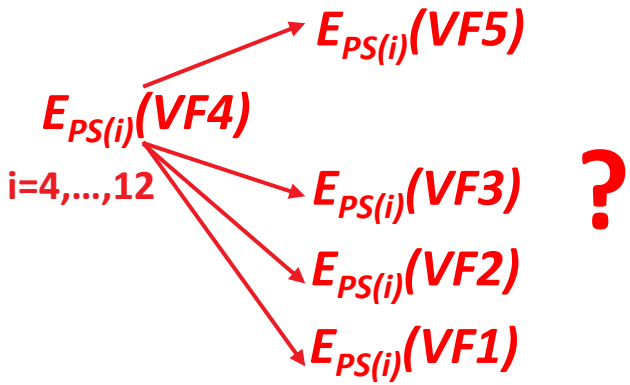


- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction

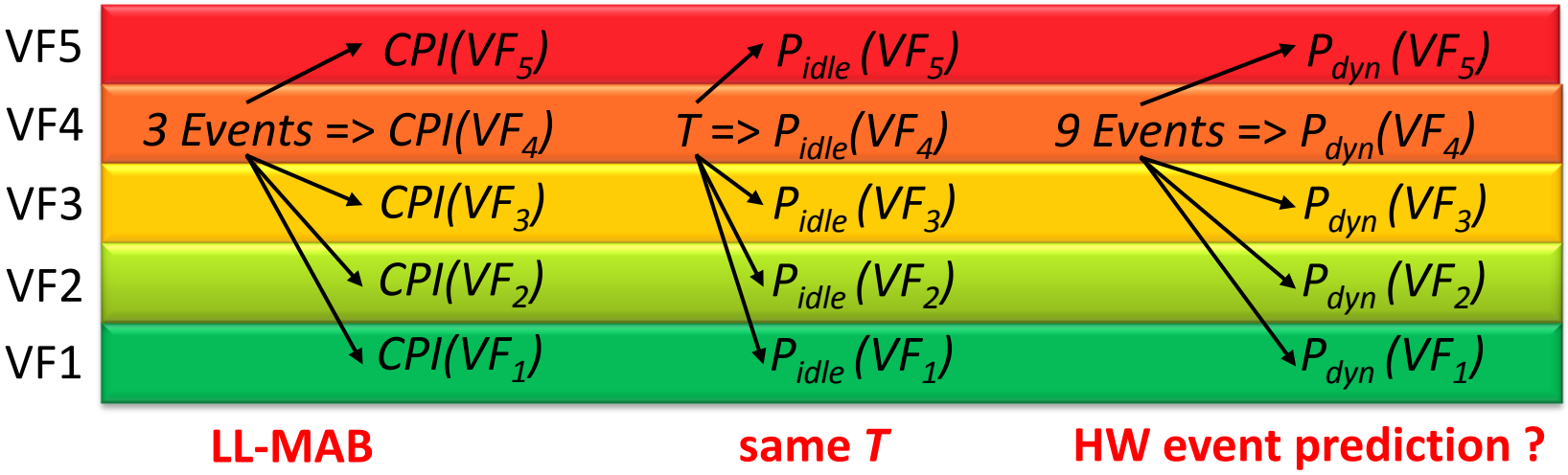
PREDICTING POWER ACROSS VF STATES



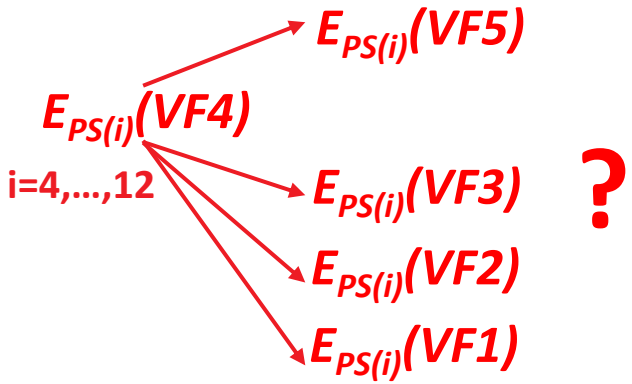
- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction



PREDICTING POWER ACROSS VF STATES



- Predicting CPI Across VF States
  - LL-MAB Predictor
- Predicting P_{idle} Across VF States
  - Use the same *Temperature* (approximation)
- Predicting P_{dyn} Across VF States
  - HW events prediction



2 observations + LL-MAB CPI Predictor

OBSERVATION 1 + OBSERVATION 2

- At any given point in the execution of a program, core-private HW event counts per instruction are independent of VF state.
- => To execute the same section of instructions in a program, the activities of core-private resources are independent of VF state.

#	Event Name
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses

OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, core-private HW event counts per instruction are independent of VF state.
- => To execute the same section of instructions in a program, the activities of core-private resources are independent of VF state.

#	Event Name	Error
E4	Retired_uOPs	0.6%
E5	FPU_Pipe_Assignment	0.9%
E6	L1I$_Fetches	0.7%
E7	L1D$_Accesses	0.7%
E8	L2$_Requests	5.0%
E9	Retired_Branches	0.7%
E10	Retired_MisBranches	1.3%
E11	L2$_Misses	4.0%

On FX-8320, between 3.5GHz (VF5) and 1.7GHz (VF2)

OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event
E1	CPU_Clock_not_Halted
E12	Dispatch_Stalls

OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)

OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)

- Cycles retiring 4 instructions
- Cycles retiring 3 instructions
- Cycles retiring 2 instructions
- Cycles retiring 1 instruction
- Cycles without retiring

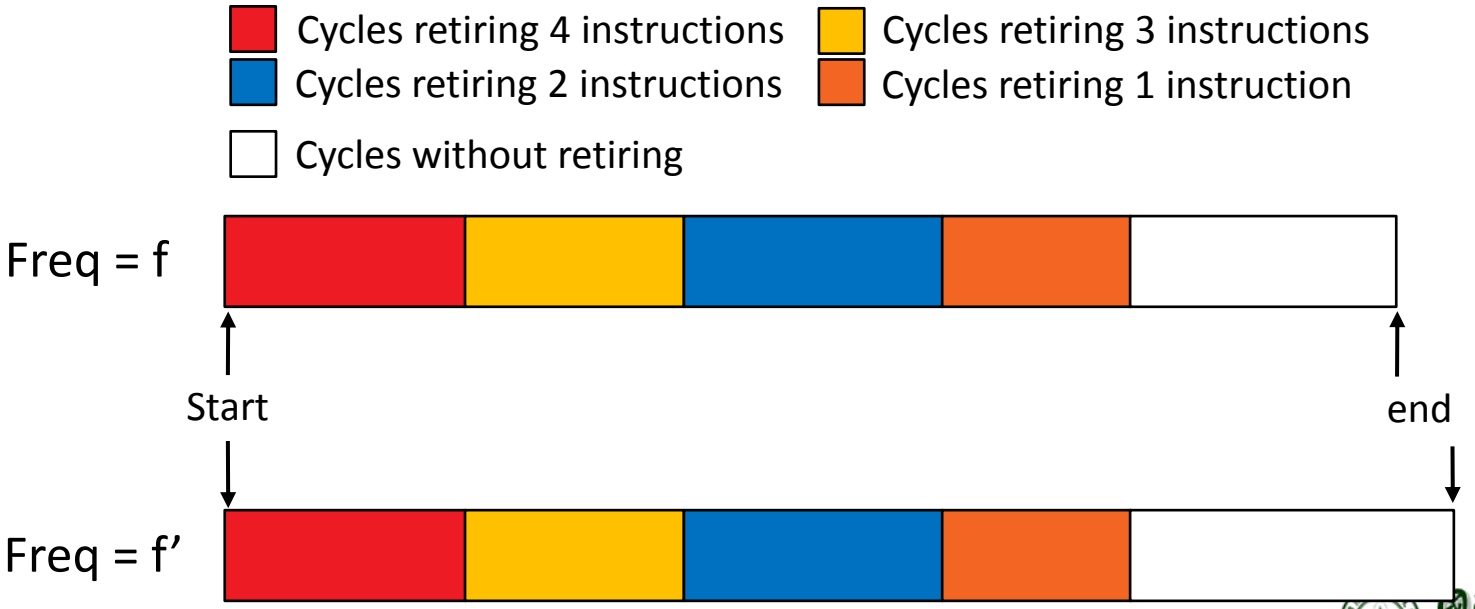
OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)



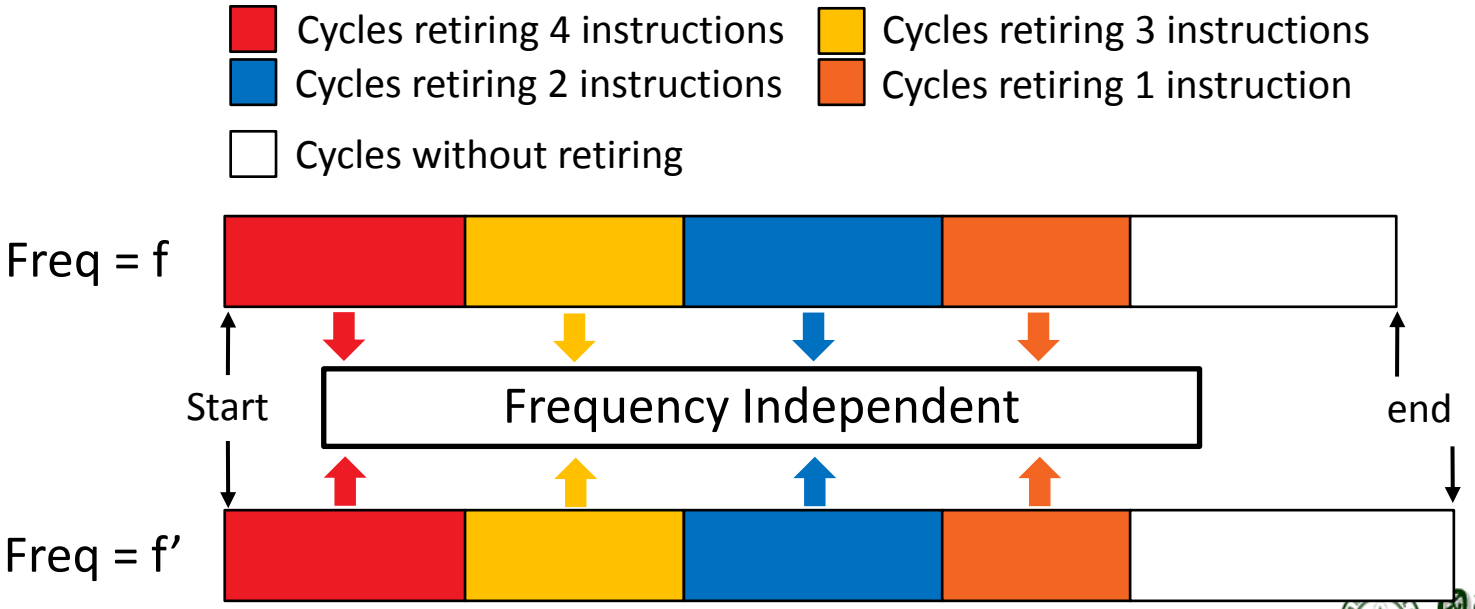
OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)

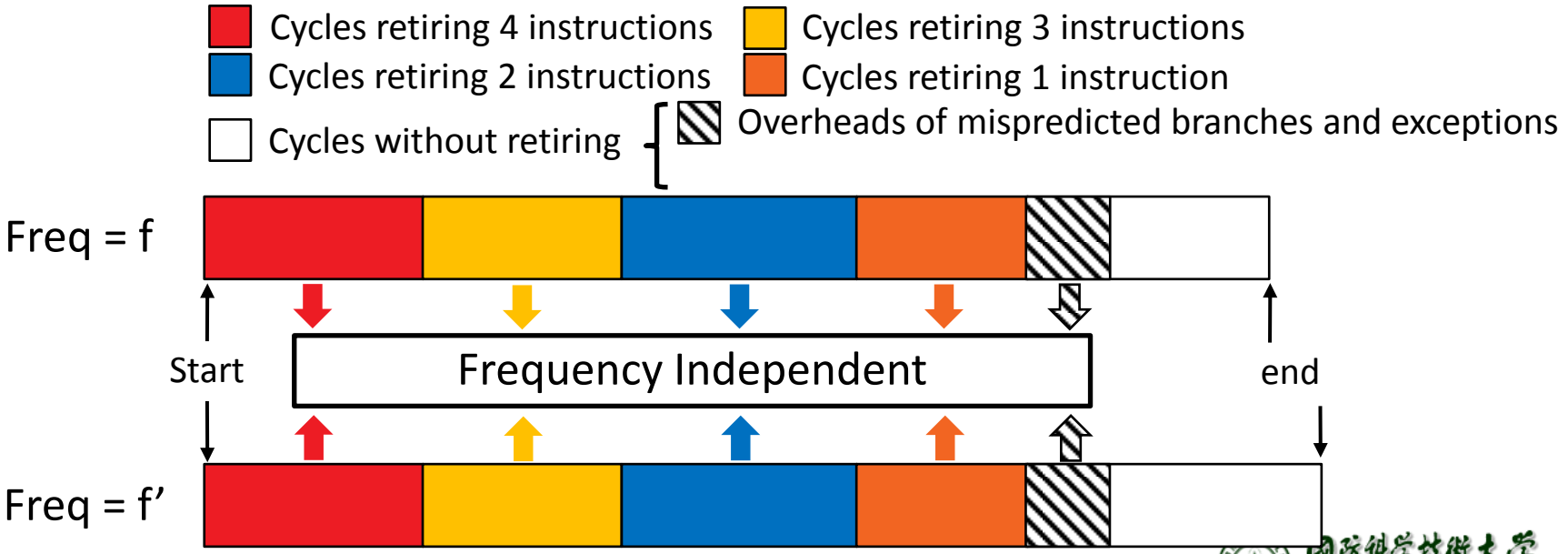


OBSERVATION 1 + OBSERVATION 2

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)



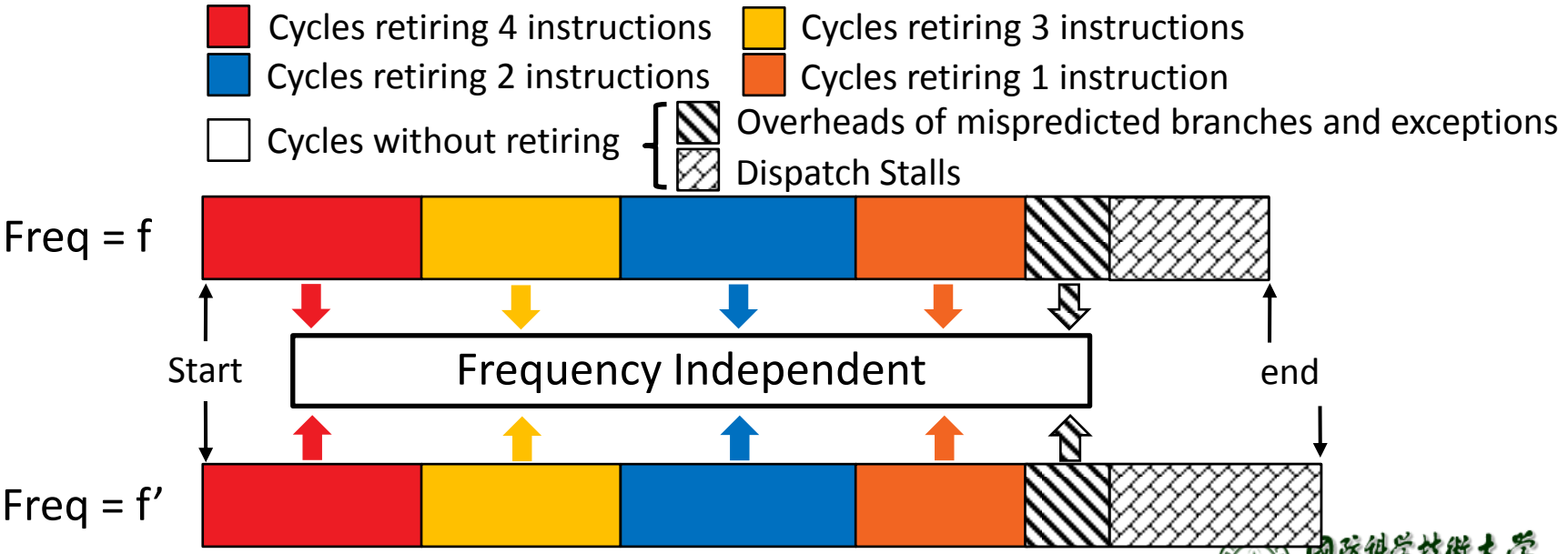
OBSERVATION 1 + OBSERVATION 2

AMD

- At any given point in the execution of a program, C_{PI} - $DispatchStall_{PI}$ is independent of VF state. (*PI=Per Instruction*)
- => To execute the same section of instructions, *E1-E12* is independent of VF state.

#	Event	Gap Error
E1	CPU_Clock_not_Halted	1.7%
E12	Dispatch_Stalls	

On FX-8320, between3.5GHz (VF5) and 1.7GHz (VF2)



國防科学技術大学 National University of Defense Technology

DYNAMIC POWER PREDICTION – FLOW

AMD

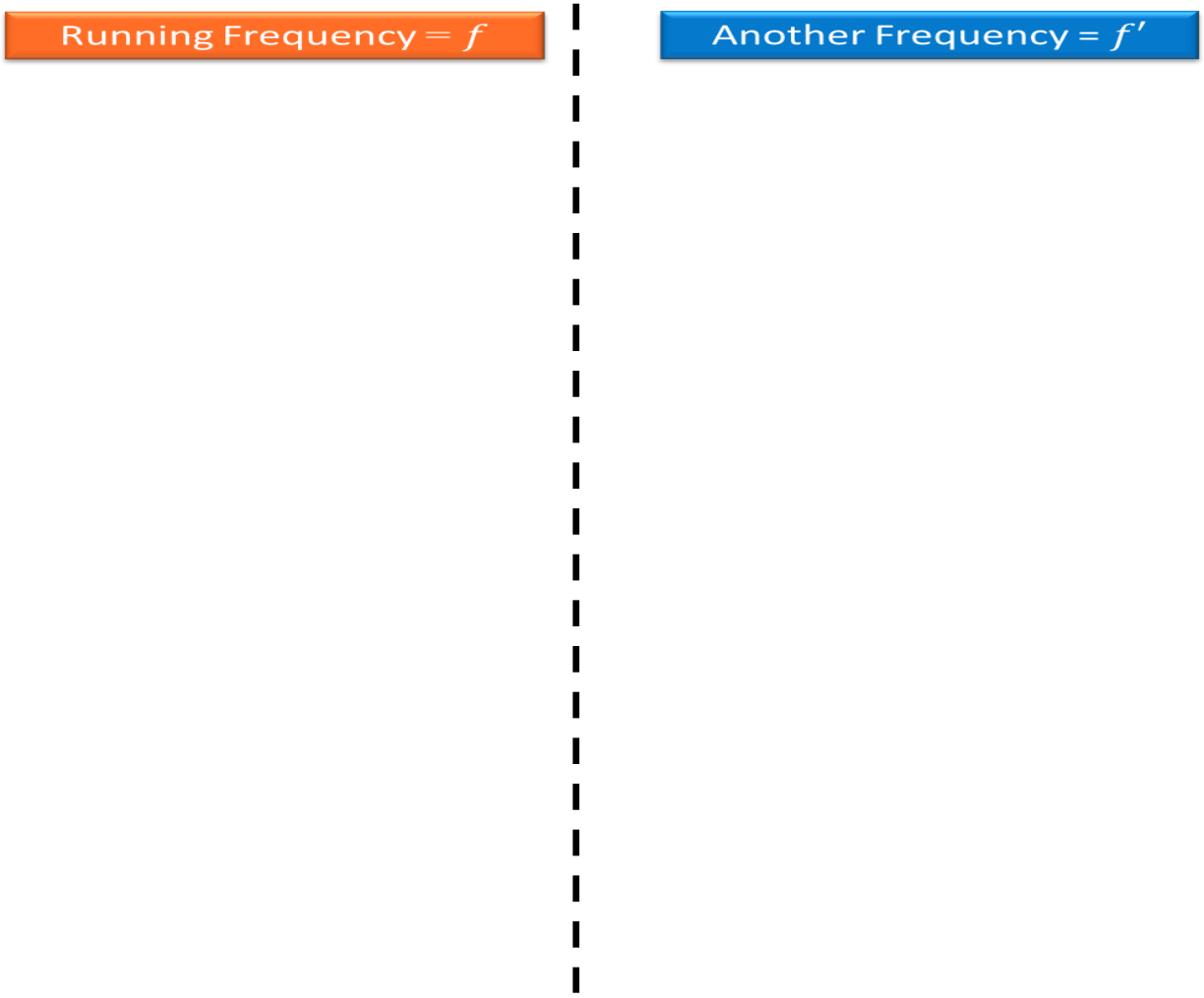
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)



DYNAMIC POWER PREDICTION – FLOW

AMD

HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)

Running Frequency = f

From PMC

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

$E2_{PS}(f)$

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

Another Frequency = f'

DYNAMIC POWER PREDICTION – FLOW

AMD

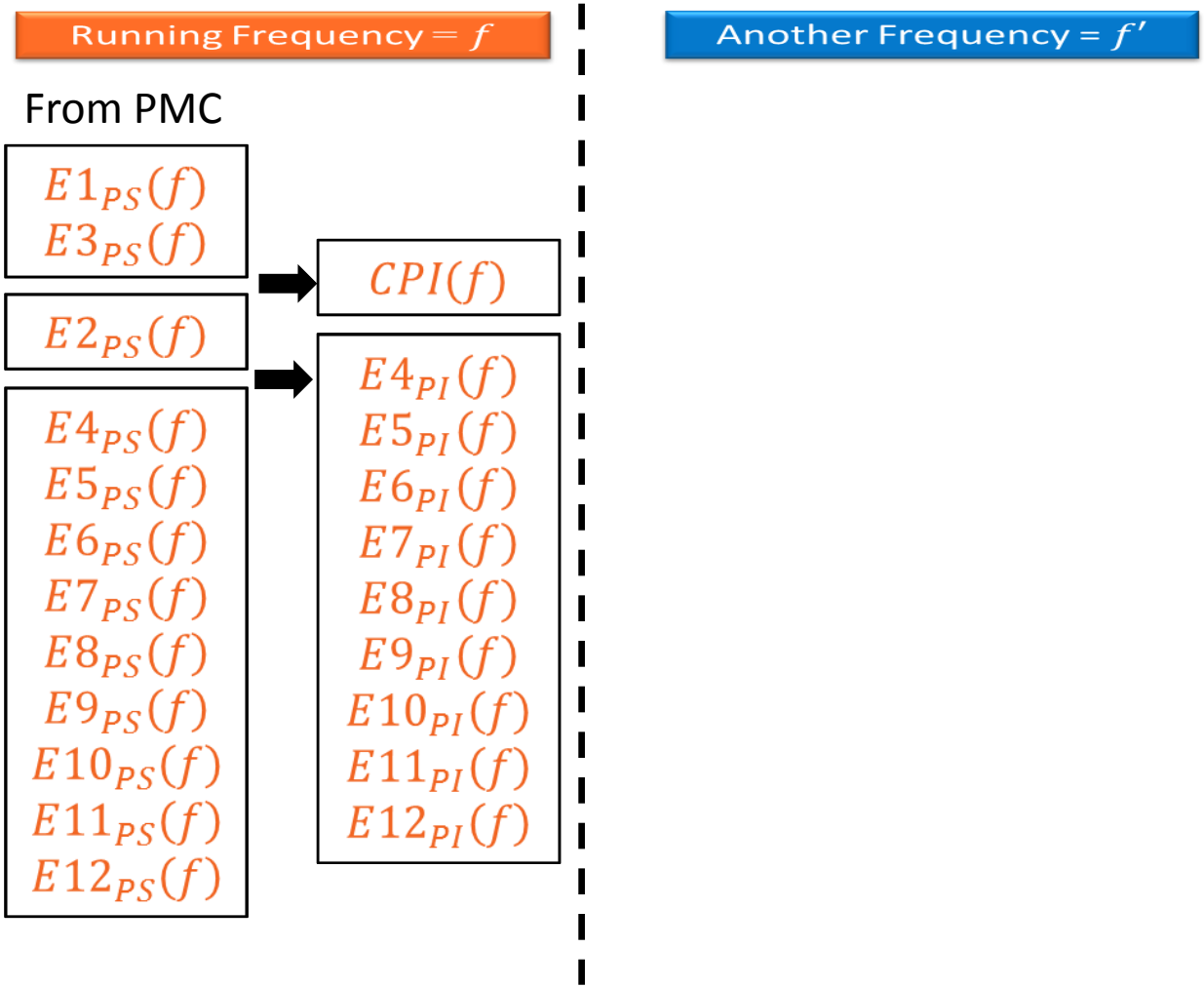
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)



DYNAMIC POWER PREDICTION – FLOW

AMD

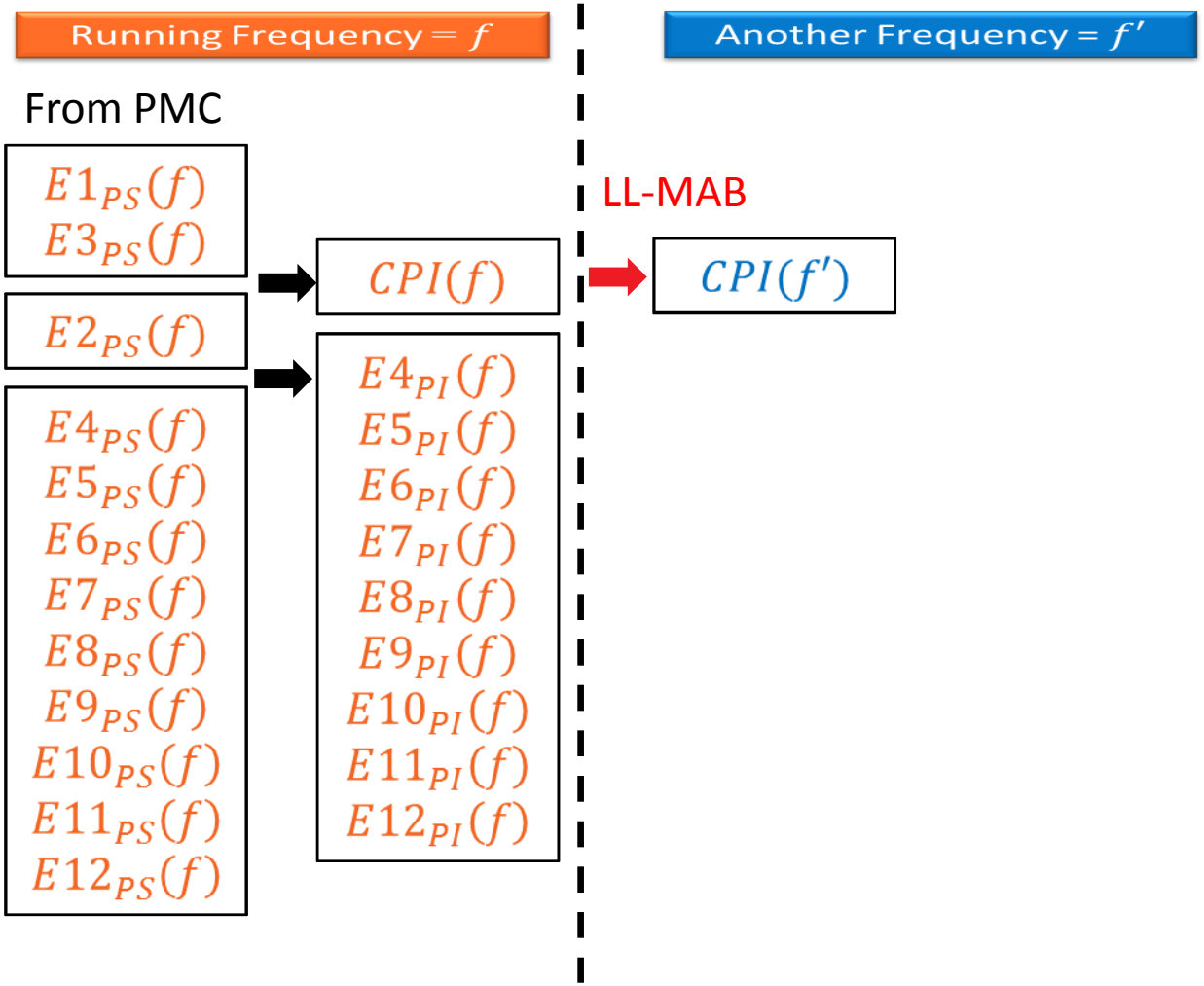
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)



DYNAMIC POWER PREDICTION – FLOW

AMD

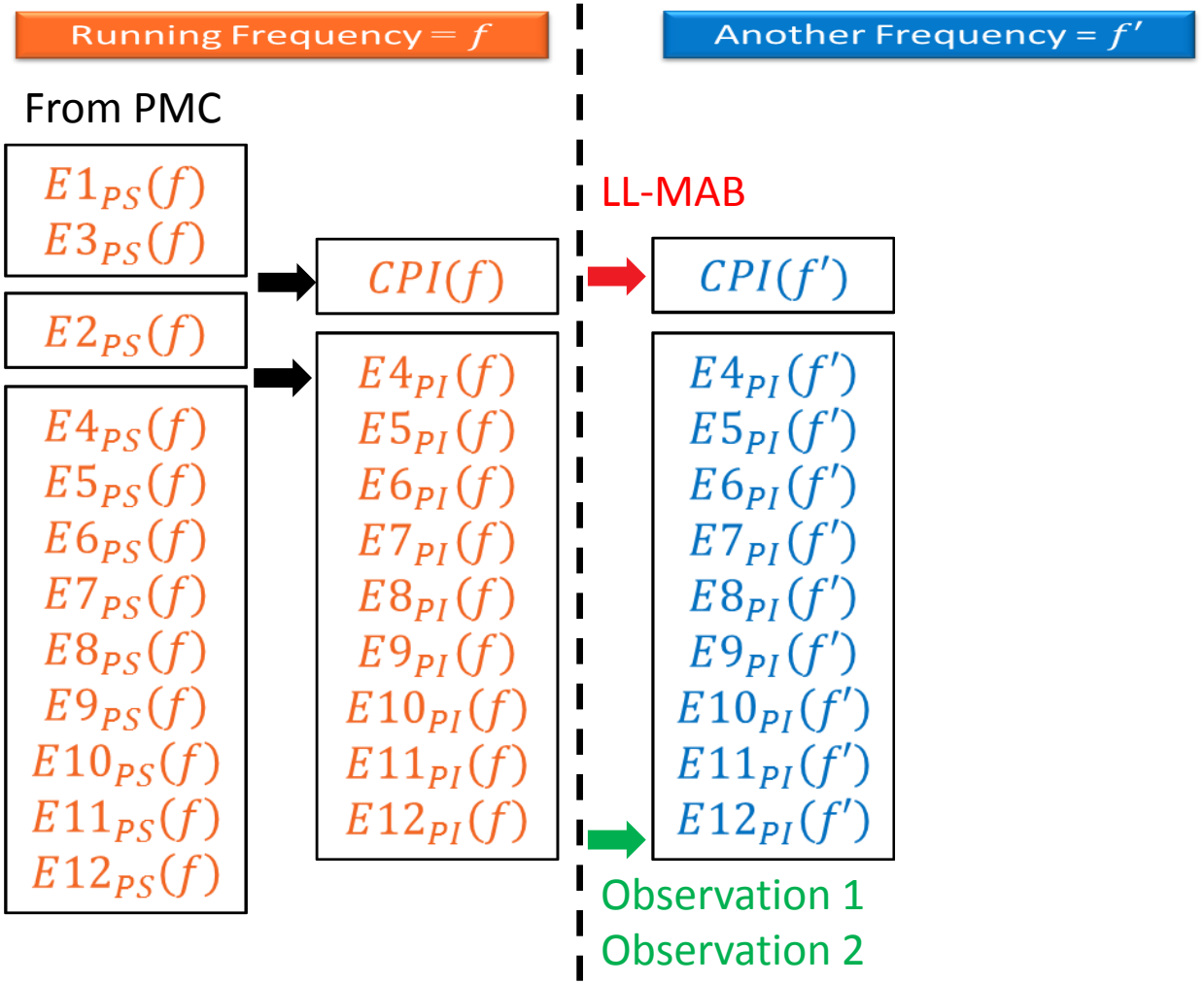
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)



DYNAMIC POWER PREDICTION – FLOW

AMD

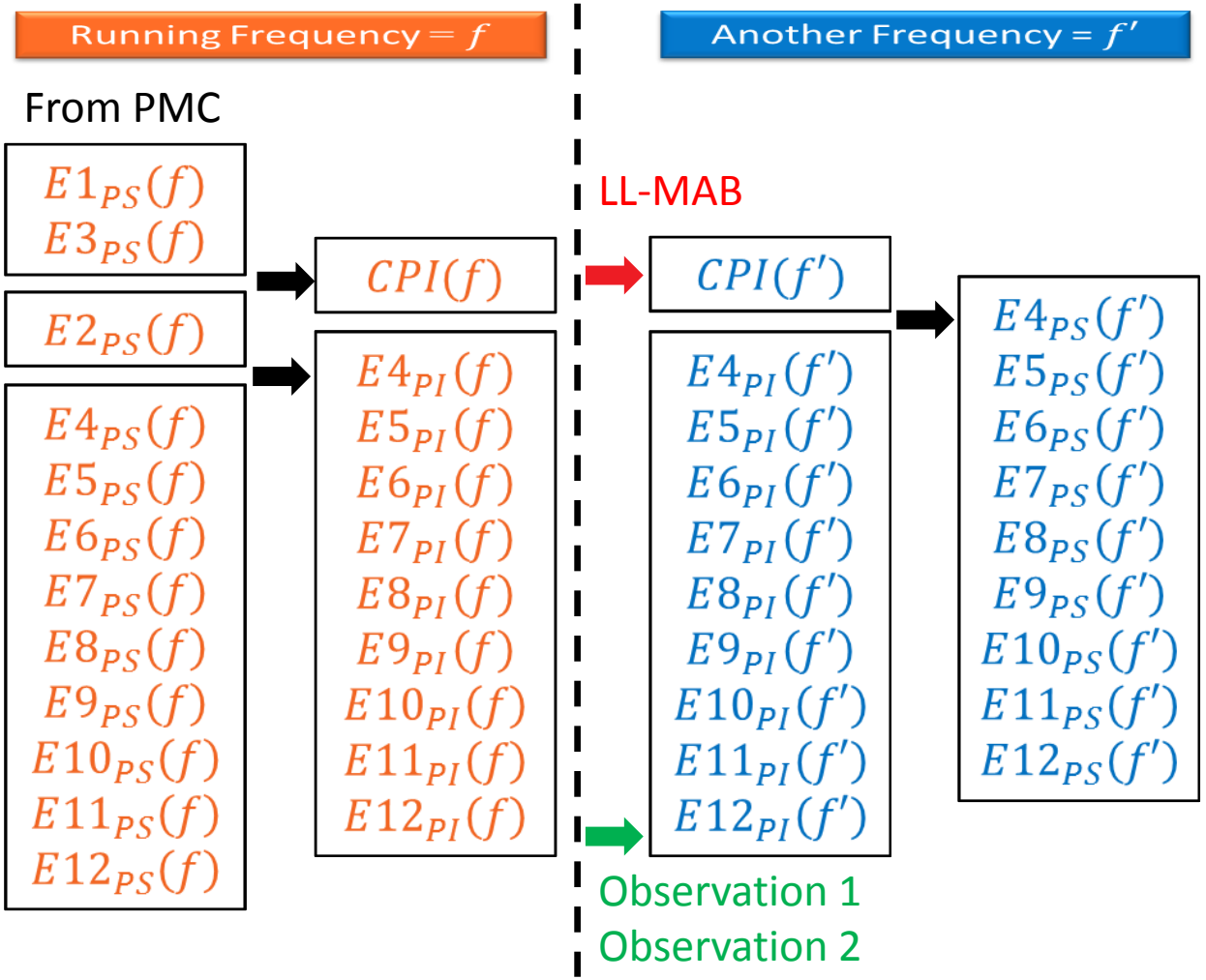
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

Prediction Flow (Bridge: LL-MAB + 2 OBs)



DYNAMIC POWER PREDICTION – FLOW

AMD

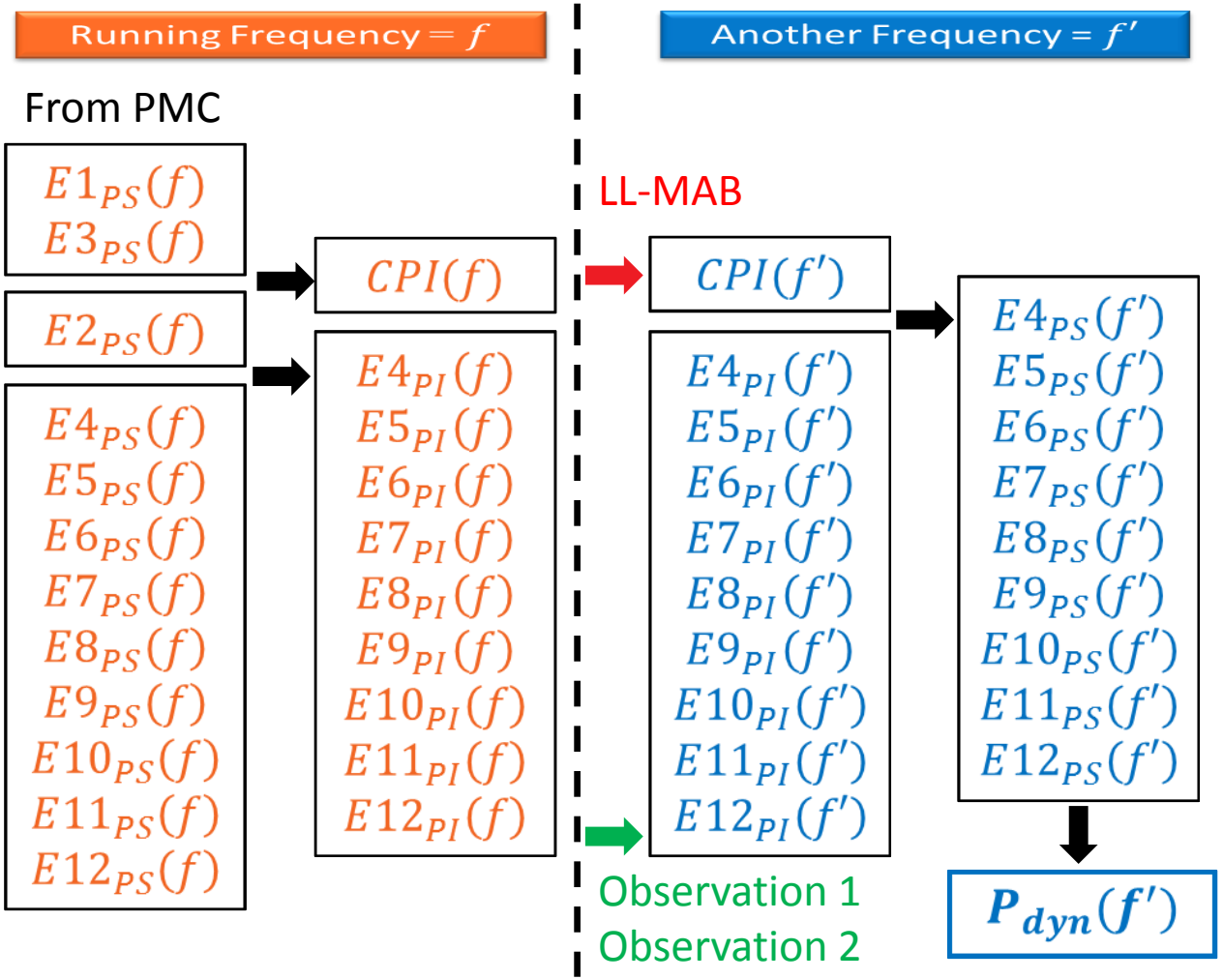
HW Events

#	Event
E1	CPU_Clock_not_Halted
E2	Retired_Instructions
E3	MAB0_Wait_Cycles
E4	Retired_uOPs
E5	FPU_Pipe_Assignment
E6	L1I$_Fetches
E7	L1D$_Accesses
E8	L2$_Requests
E9	Retired_Branches
E10	Retired_MisBranches
E11	L2$_Misses
E12	Dispatch_Stalls

PS=per-second

PI=per-instruction

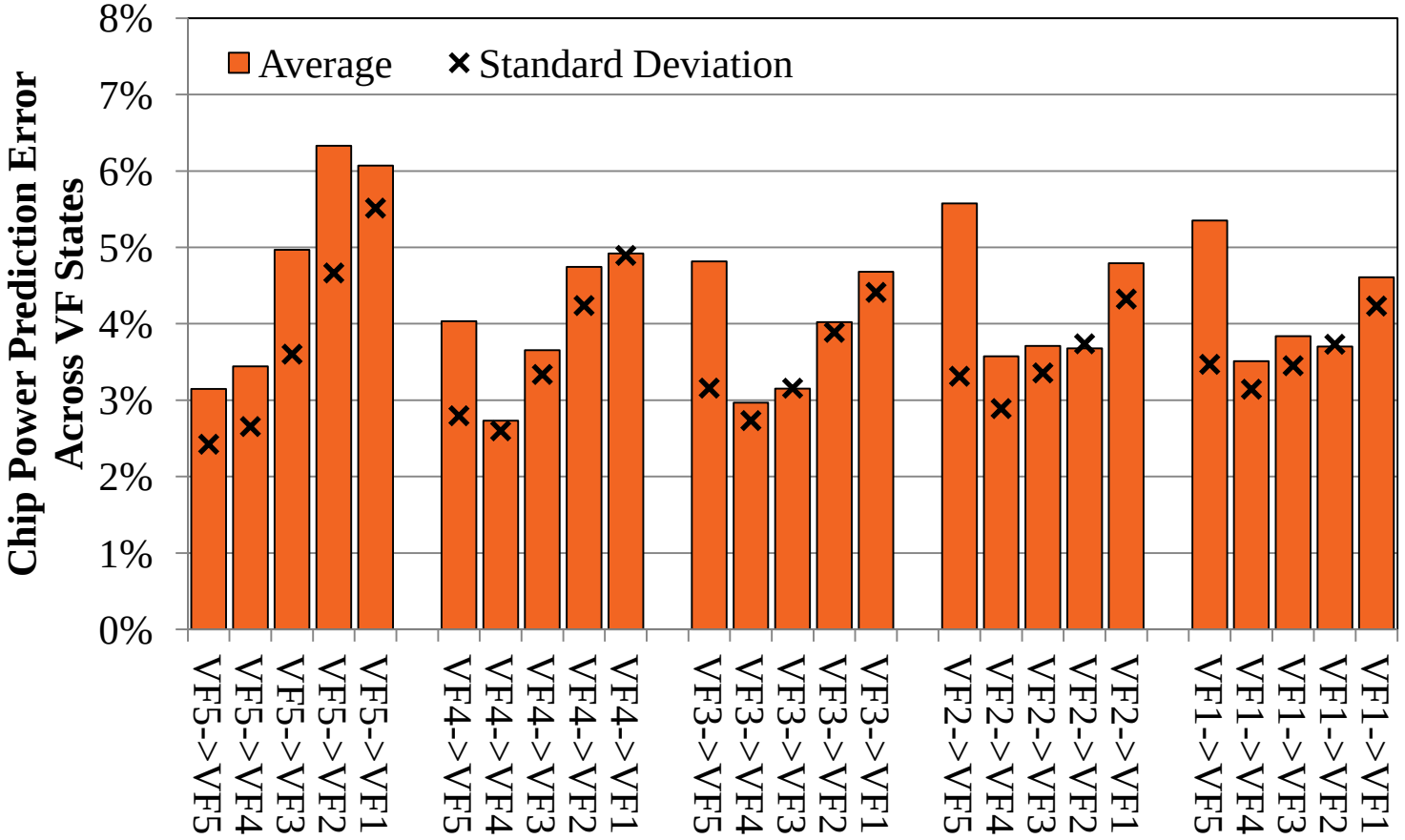
Prediction Flow (Bridge: LL-MAB + 2 OBs)



PREDICTING ERROR OF AVERAGE CHIP POWER

LOWER IS BETTER

- Error: 4.2%
- Standard Deviation: 3.6%



OUTLINE

- [illegible]
- Pe[illegible]
- Power [illegible]
- [illegible]
- C[illegible]

Q1: How does the application's performance change with the VF state?

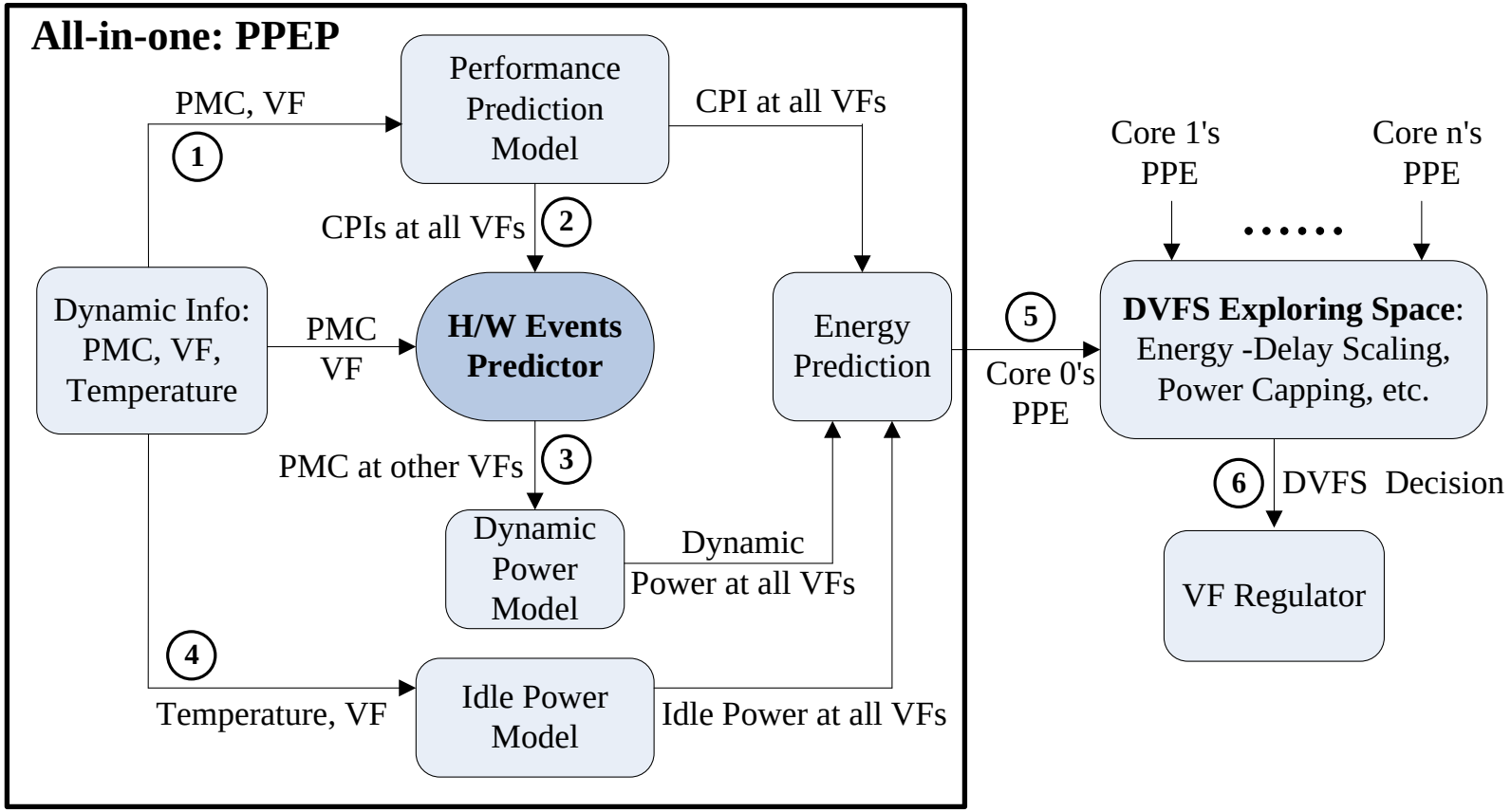
Q2: How does the application's power change with the VF state?

OUTLINE

- Background
- Experimental Methodology
- Performance Prediction Across DVFS States
- Power Prediction Across DVFS States
- **PPEP Framework**
- Conclusion

PPEP FRAMEWORK

- PMC based User Daemon
- Periodically Running



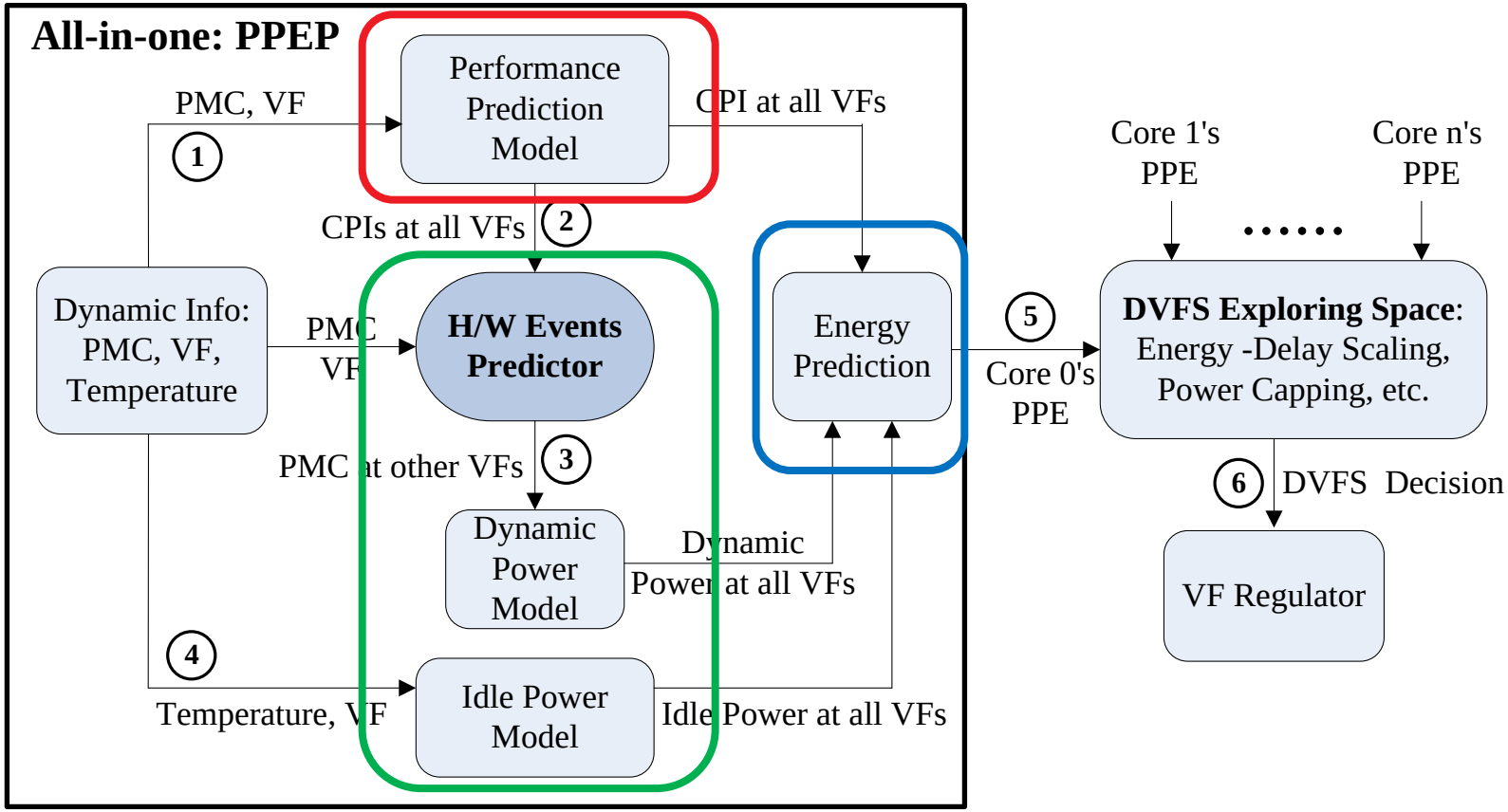
PPEP FRAMEWORK

Online
Performance
Power
Energy

Prediction

Commercial
AMD
Processors

- PMC based User Daemon
- Periodically Running



CONCLUSION

AMD

On Commercial Processor

- Demonstrated an across-VF CPI predictor “LL-MAB”
  - According to the *Leading Loads* theory
- Demonstrated an across-VF power predictor
  - Through the “LL-MAB” CPI predictor and 2 observations
- Combining them together: PPEP Framework
  - Supplies performance, power, and energy information
  - SW method w/o requiring HW or OS modification

THANK YOU!

AMD

- Questions

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

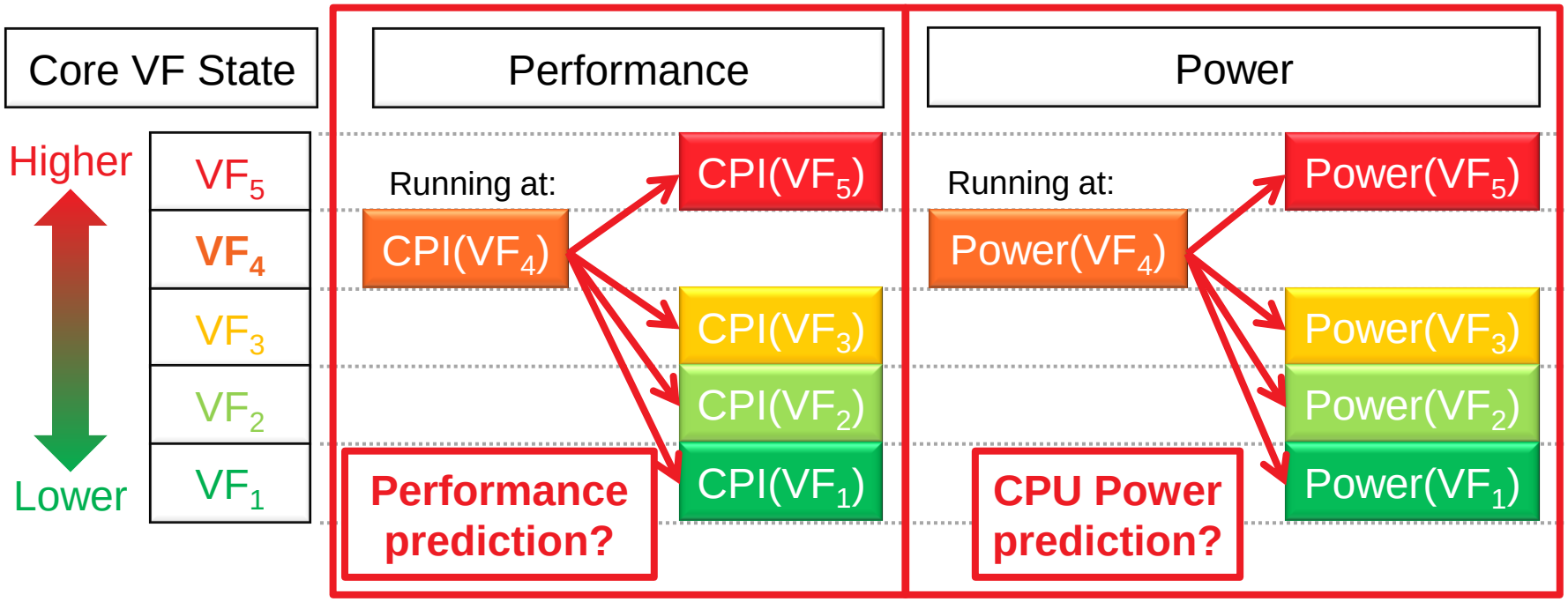
ATTRIBUTION

Backup Slides

PERFORMANCE & POWER PREDICTION

AMD

- Running VF state: VF4
- Predicting performance, power of other VF states



國防科學技術大學 National University of Defense Technology

EXPERIMENTAL METHODOLOGY

SOFTWARE TOOLS

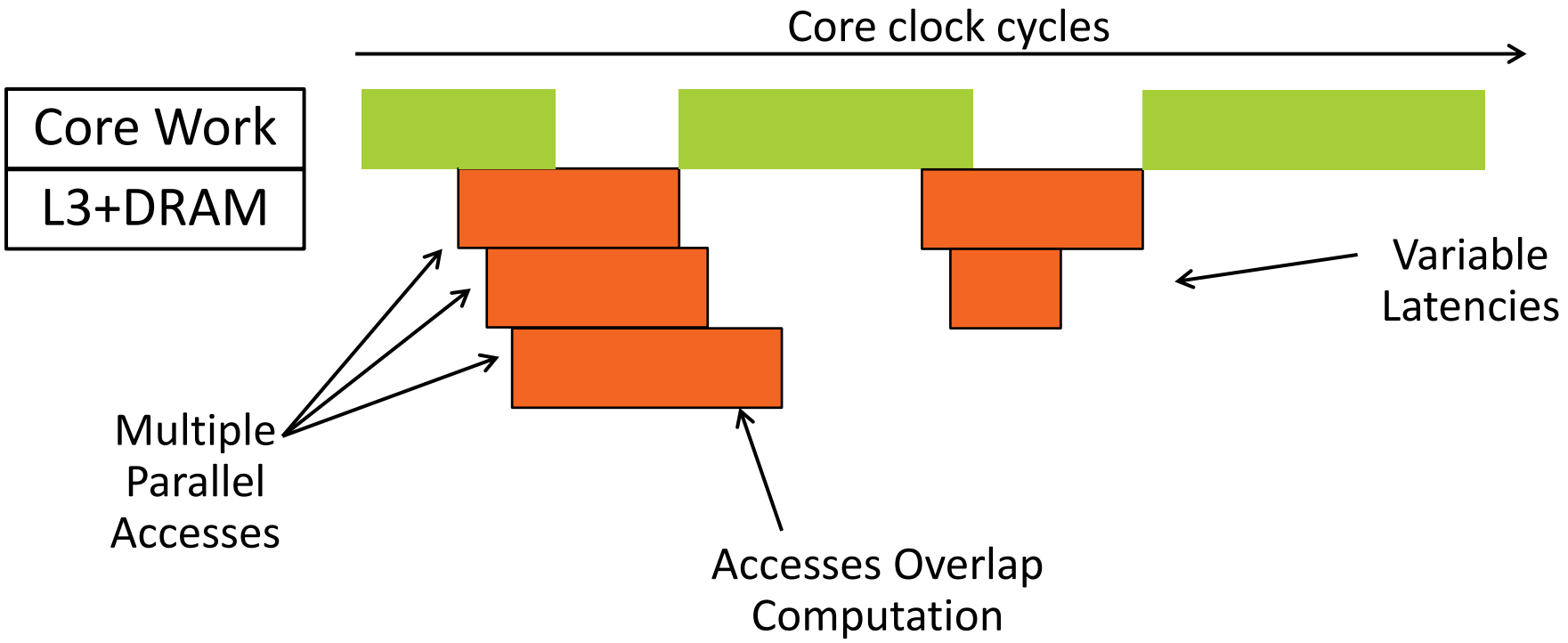
- Operating System
  - ***Ubuntu 12.04 LTS Desktop*** (kernel version *3.2.0-24*)
- Tools
  - ***taskset****:* A2C mapping
  - ***msr-tools****:* PMC control
  - ***CPUFreq*** userspace governor: VF Scaling
  - ***hwmon*** tree in sysfs: temperature measurement
- Benchmark Suites
  - ***SPEC® CPU 2006*** v1.2 (29 benchmarks)
  - ***PARSEC*** v2.1 (13 benchmarks)
  - ***NAS Parallel Benchmarks*** v3.3.1 (10 benchmarks)

HOW TO ESTIMATE "MEMORY STALL CYCLES"?

MODERN CORES MAKE THIS DIFFICULT

- Memory level parallelism
- Accesses overlap computation
- Variable latencies

memory access counts do not estimate memory time accurately

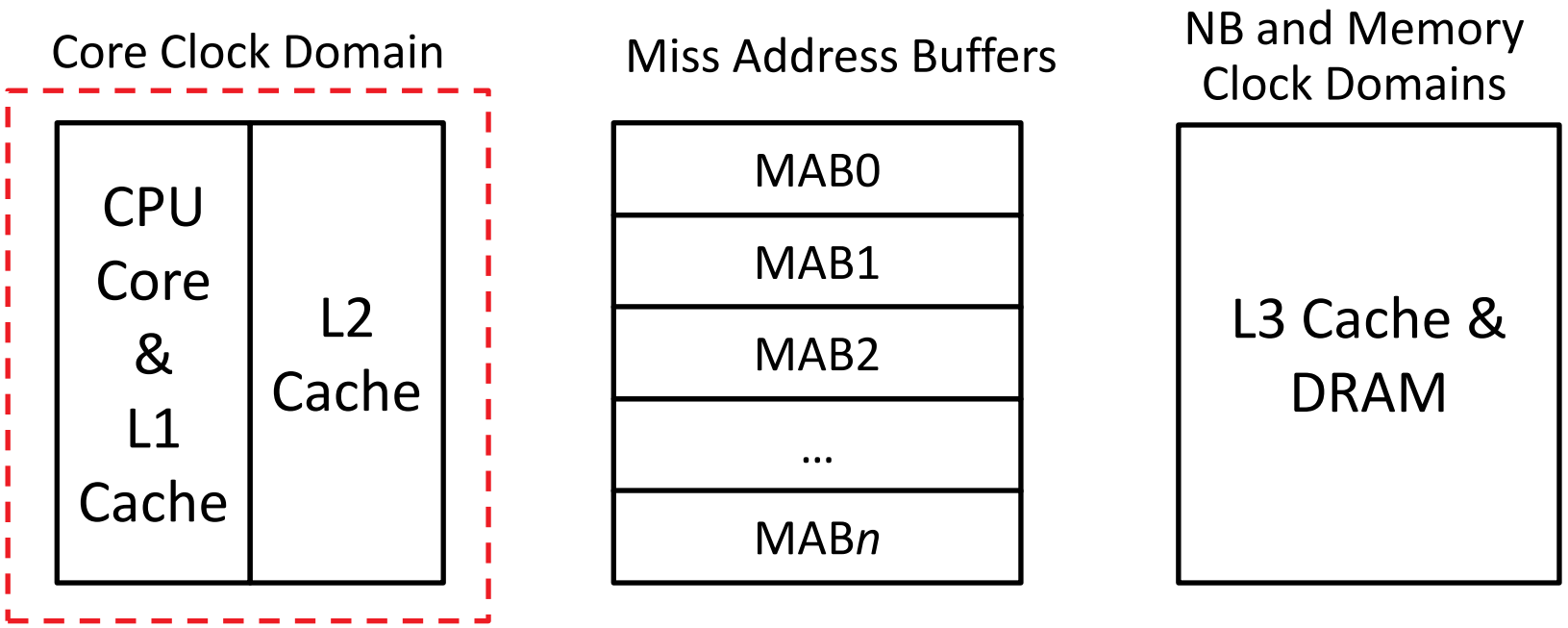


LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



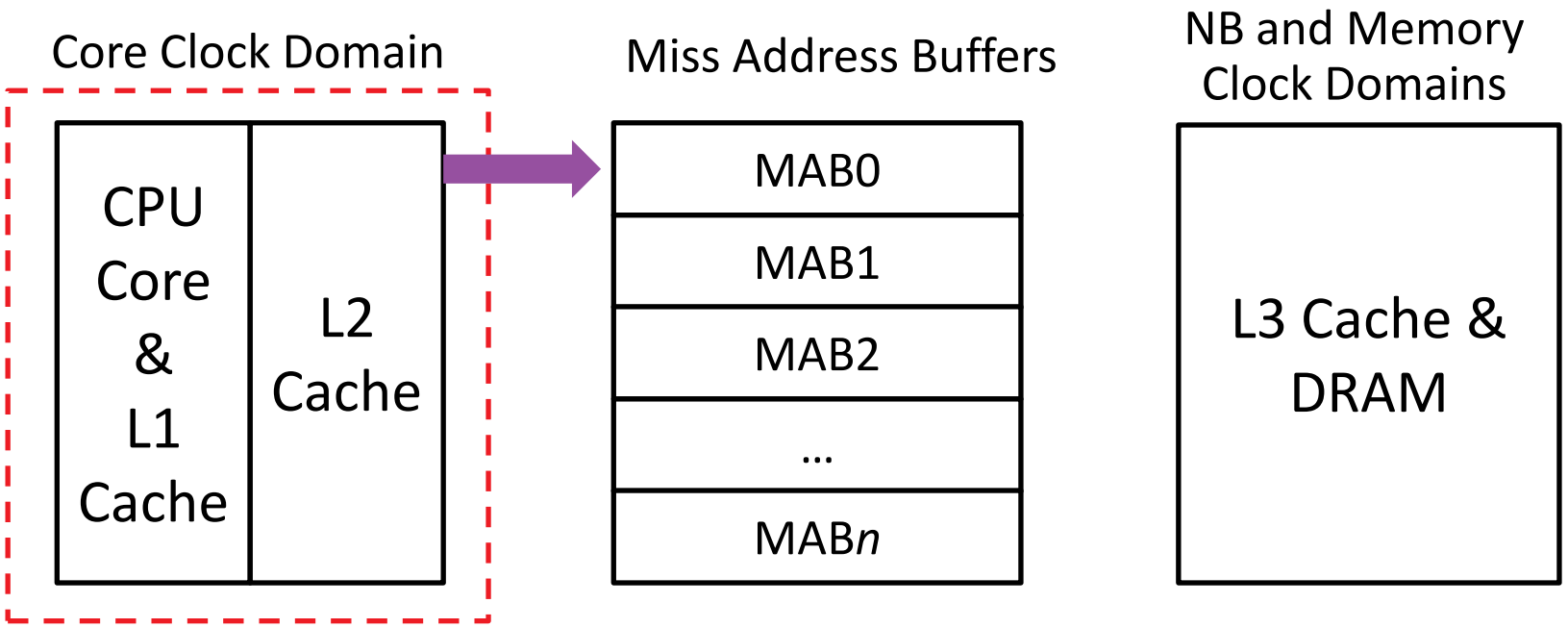
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



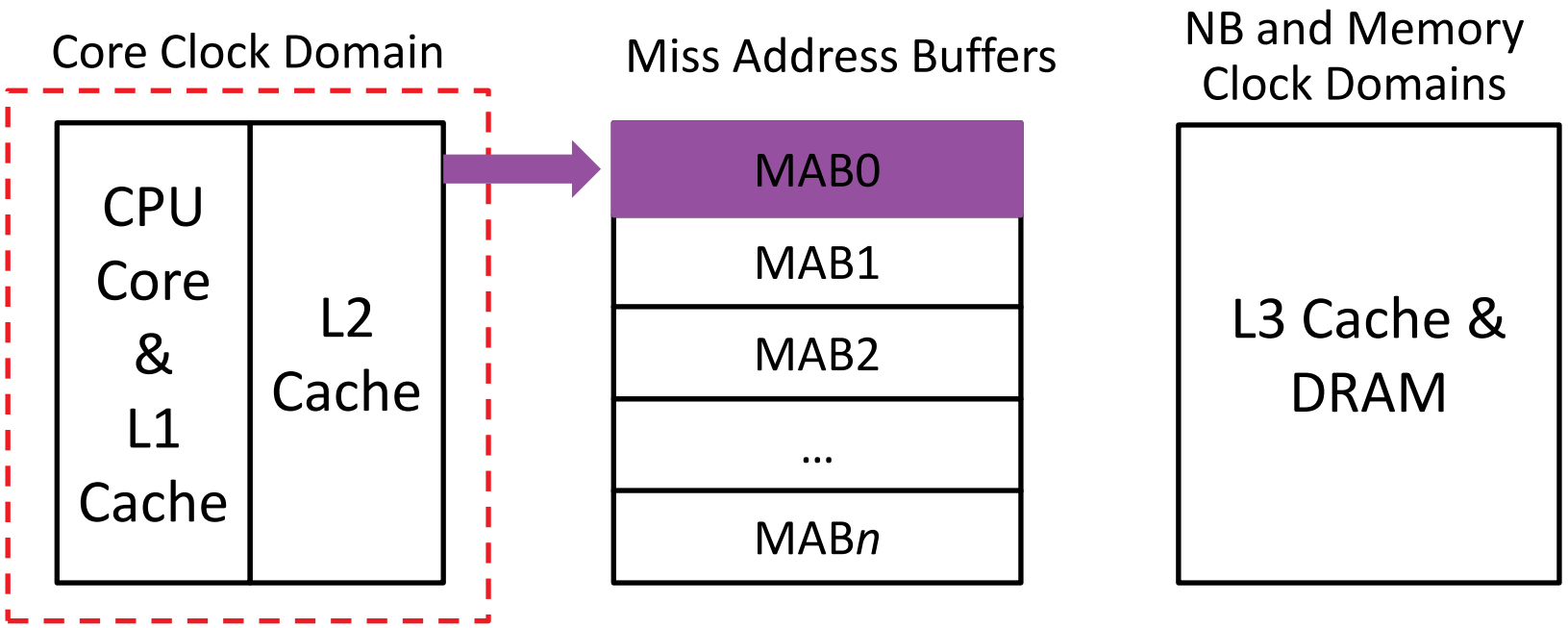
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



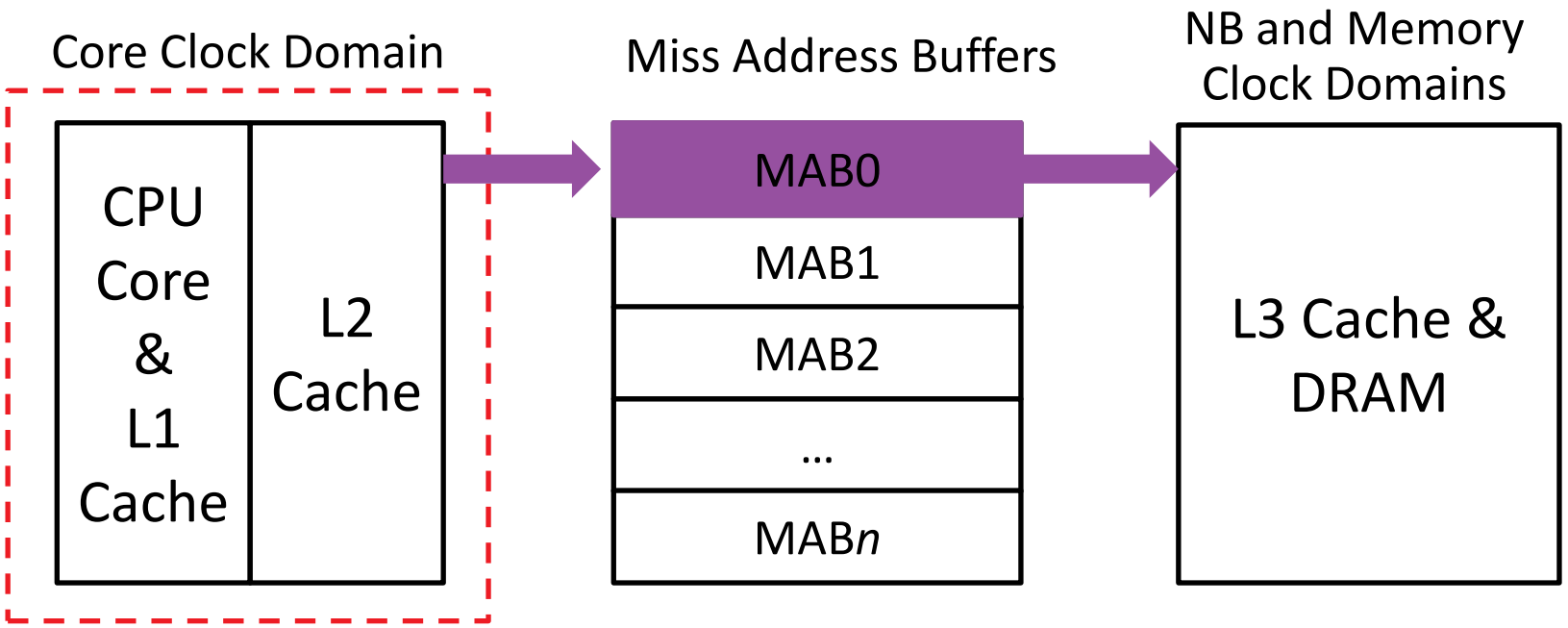
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



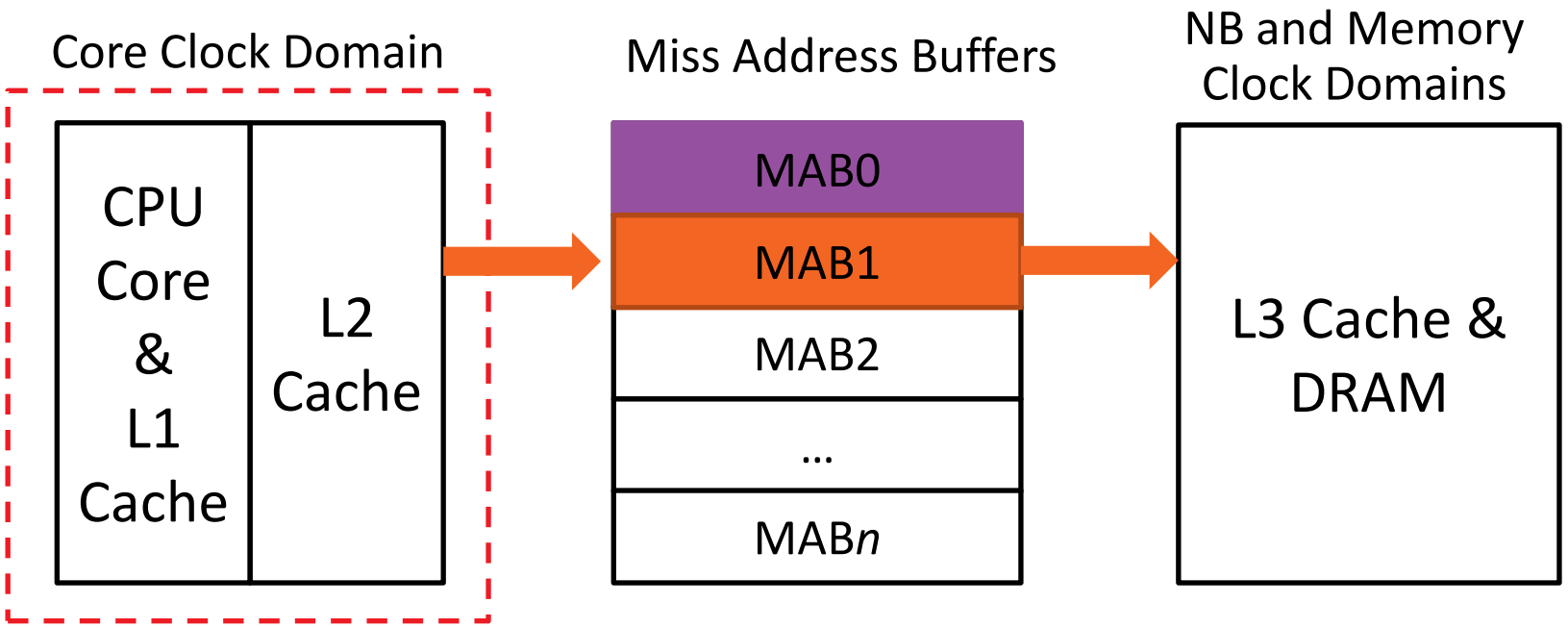
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



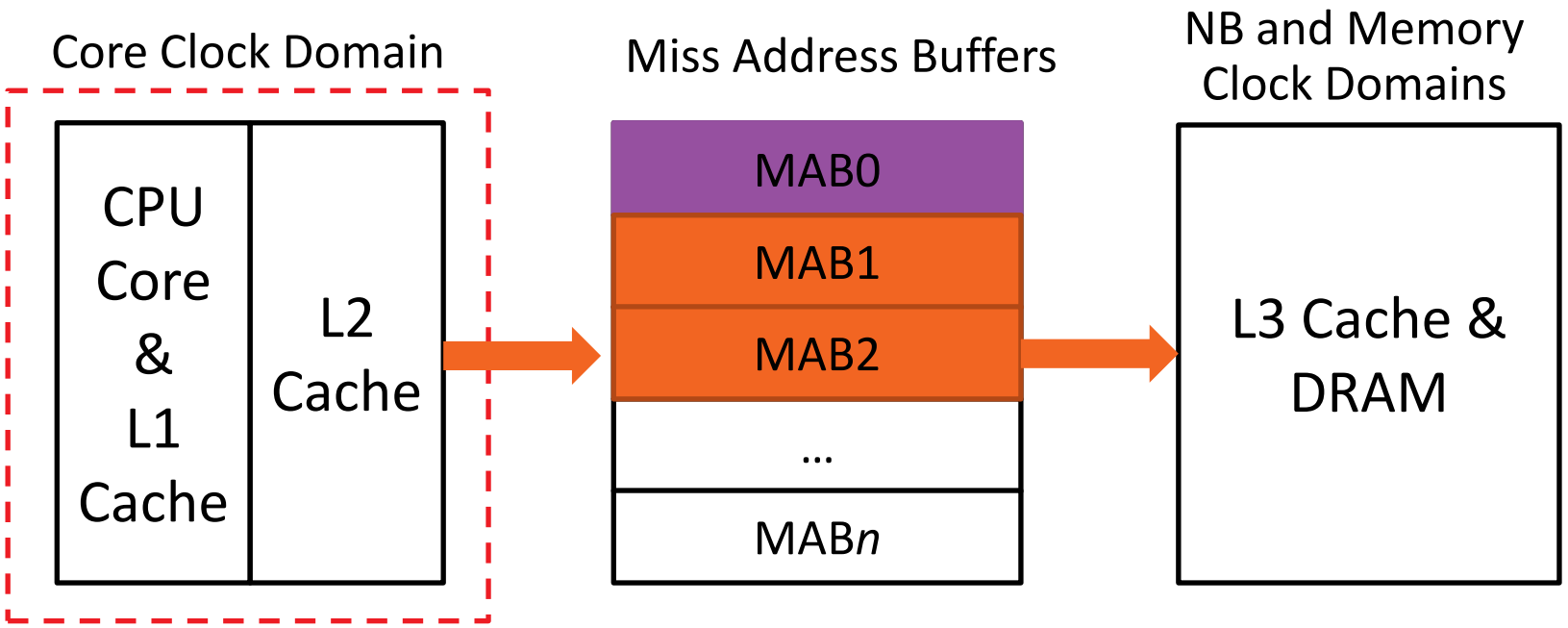
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



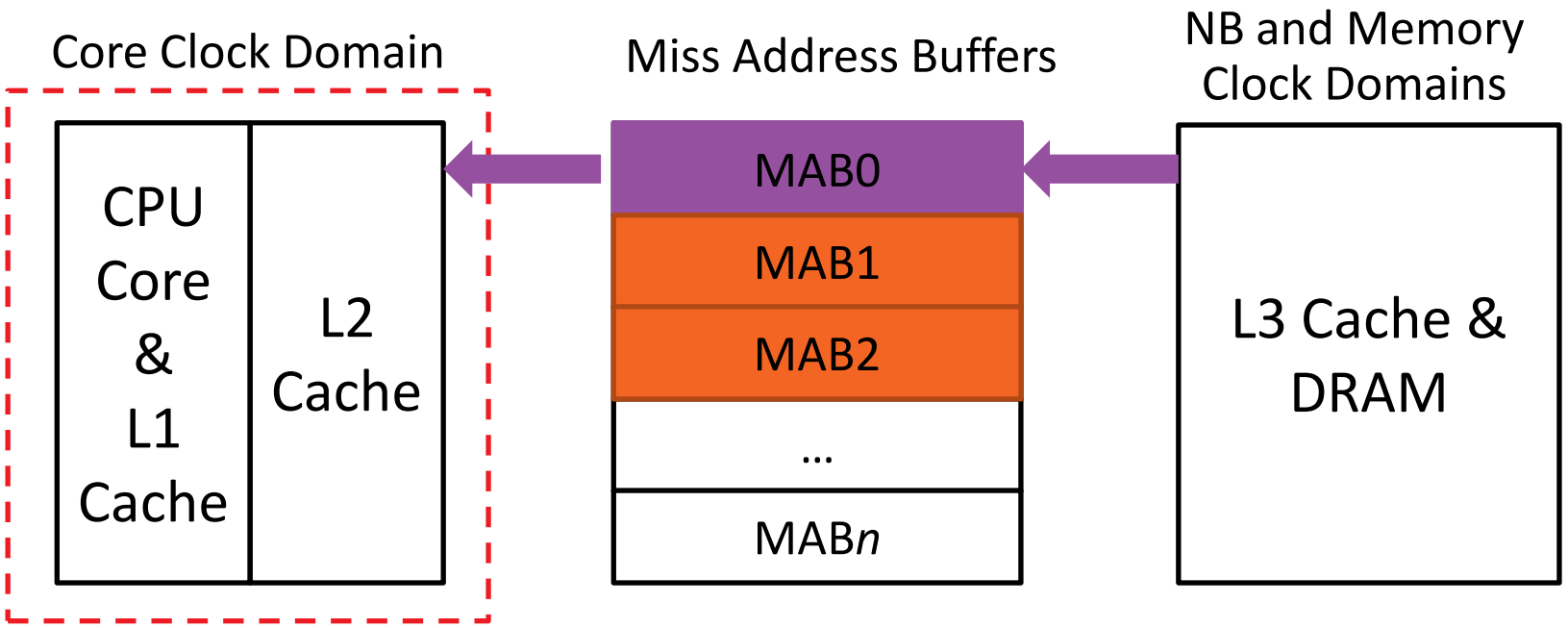
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



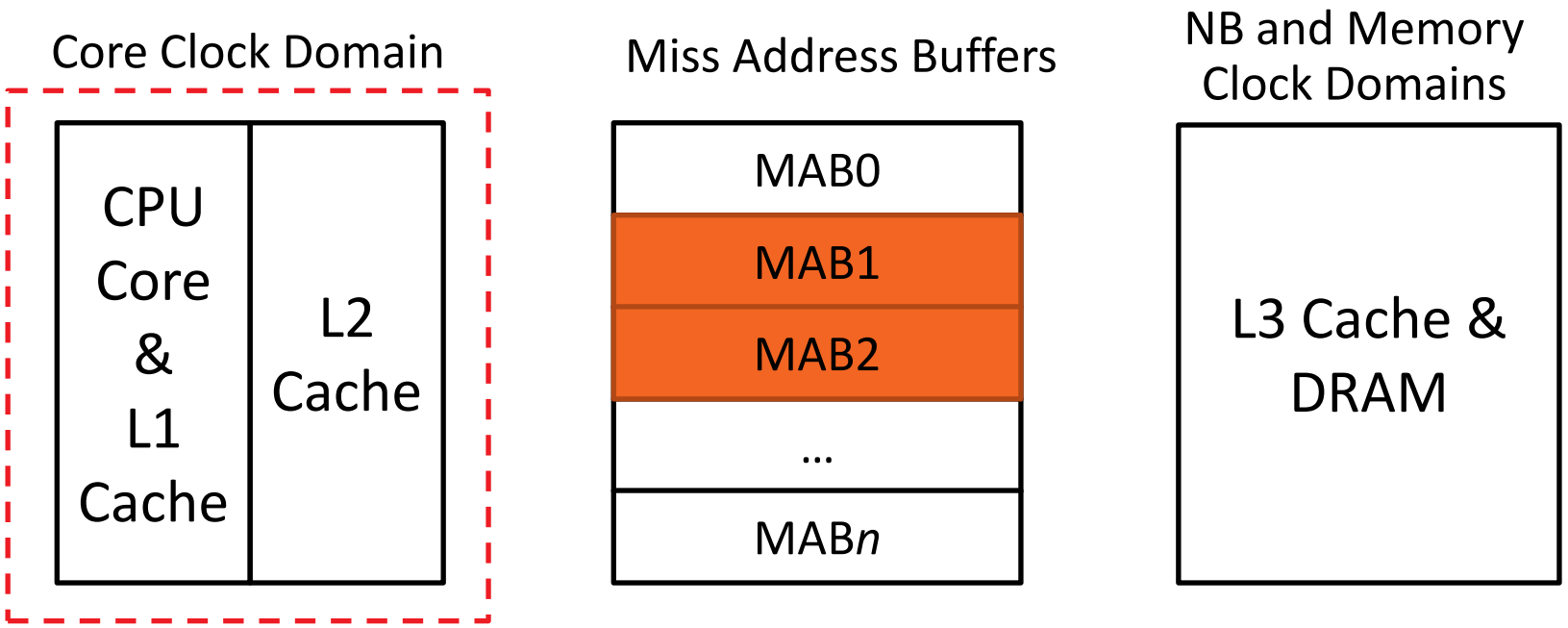
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory



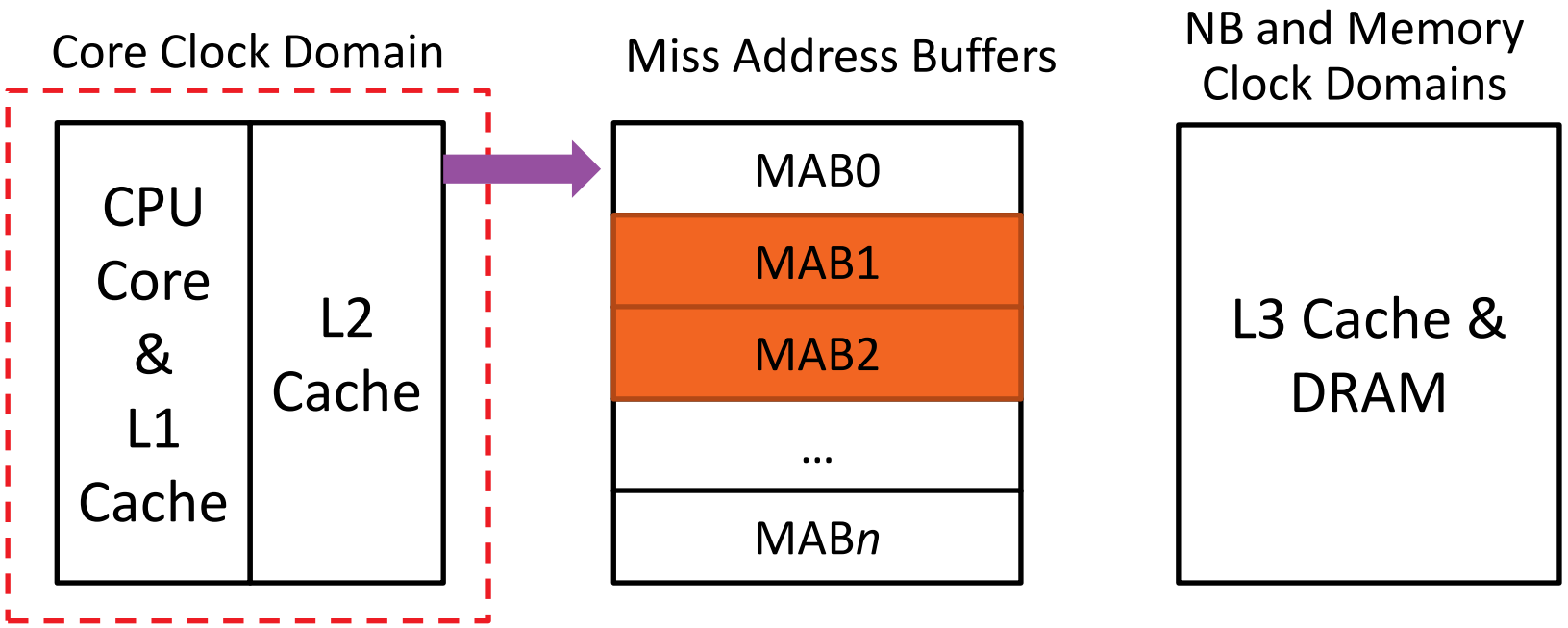
Performance event 0x69 allows SW to count # of cycles with filled MABs

LEADING LOADS ON AMD PROCESSORS

AMD

L2 cache misses held in Miss Address Buffer (MAB)

- MAB entries have a static priority (e.g. MAB0 is highest priority)
- Highest priority empty MAB holds the miss until it returns from memory

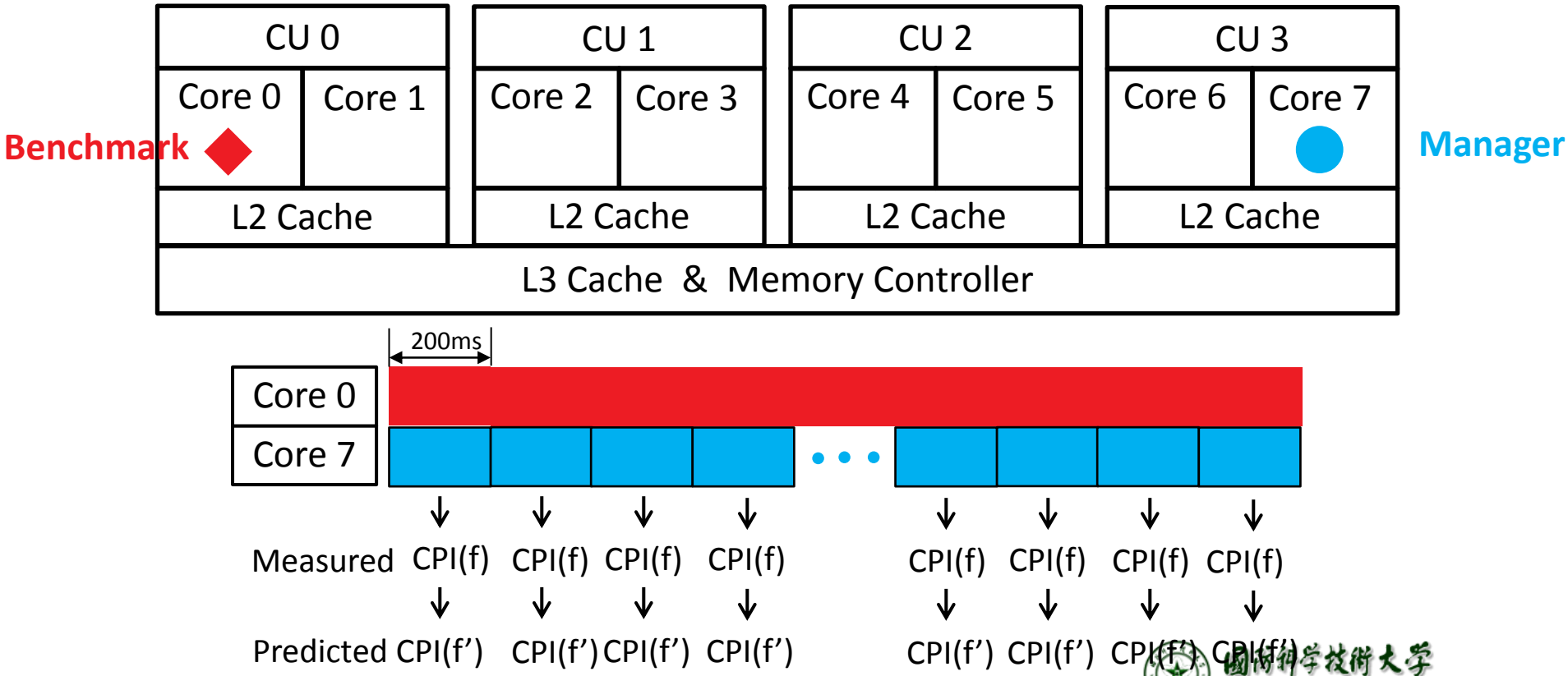


Performance event 0x69 allows SW to count # of cycles with filled MABs

MAB BASED LEADING LOADS MODEL: LL-MAB

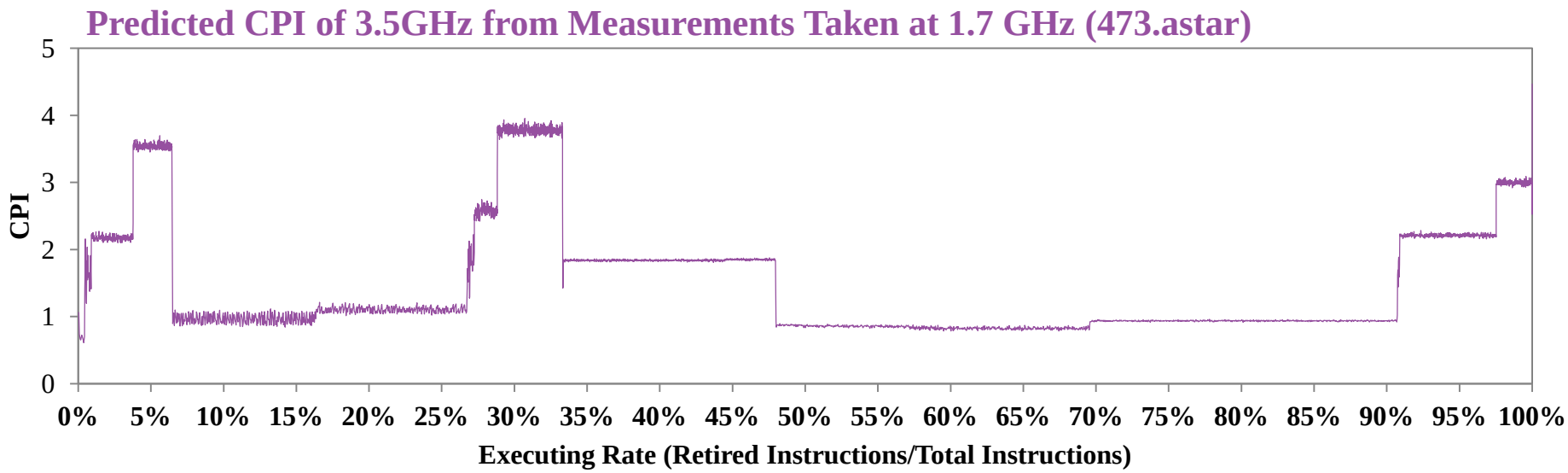
EXPERIMENT

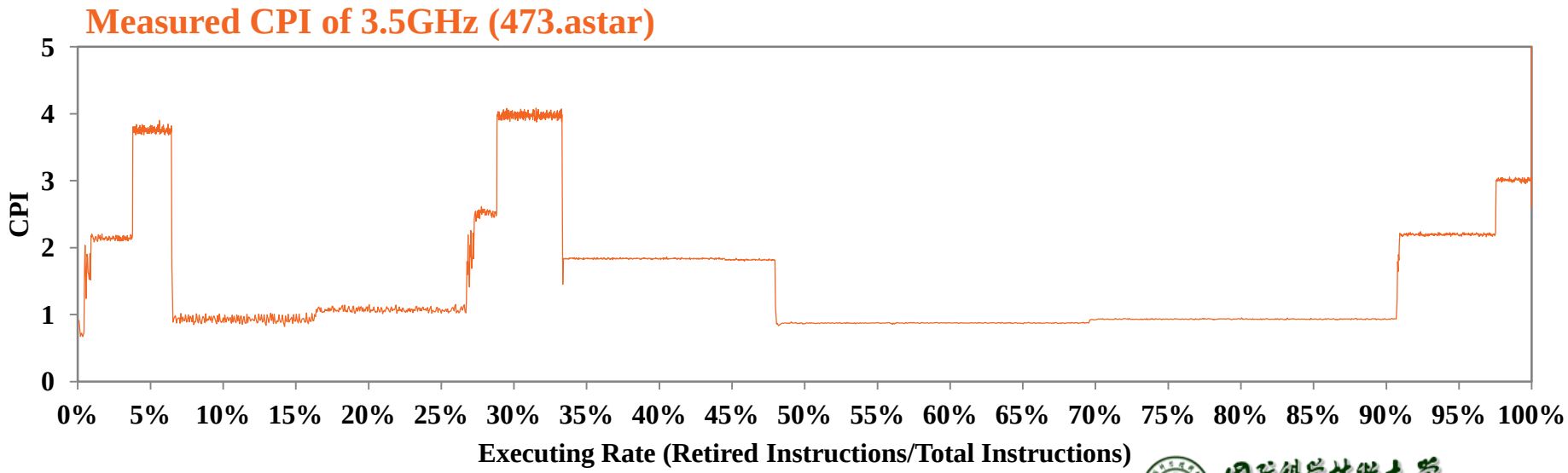
- Benchmark
  - Running on core 0.
- Manager
  - Running on Core 7. Collects the counts every 200ms.



PREDICTED CPI V.S. MEASURED CPI

1.7GHZ -> 3.5GHZ

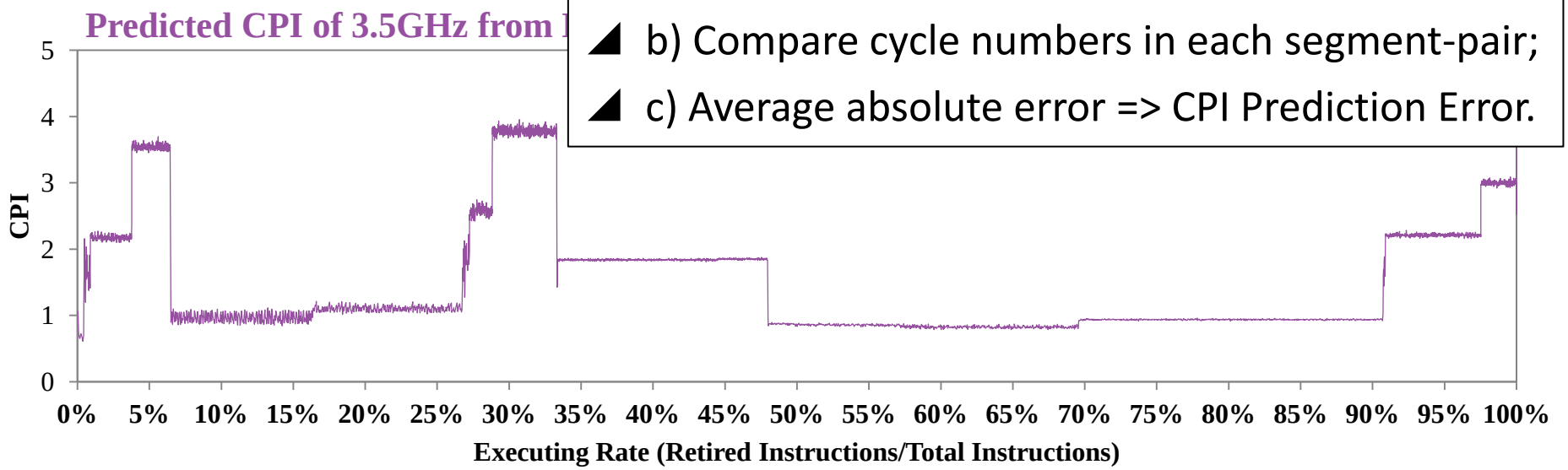


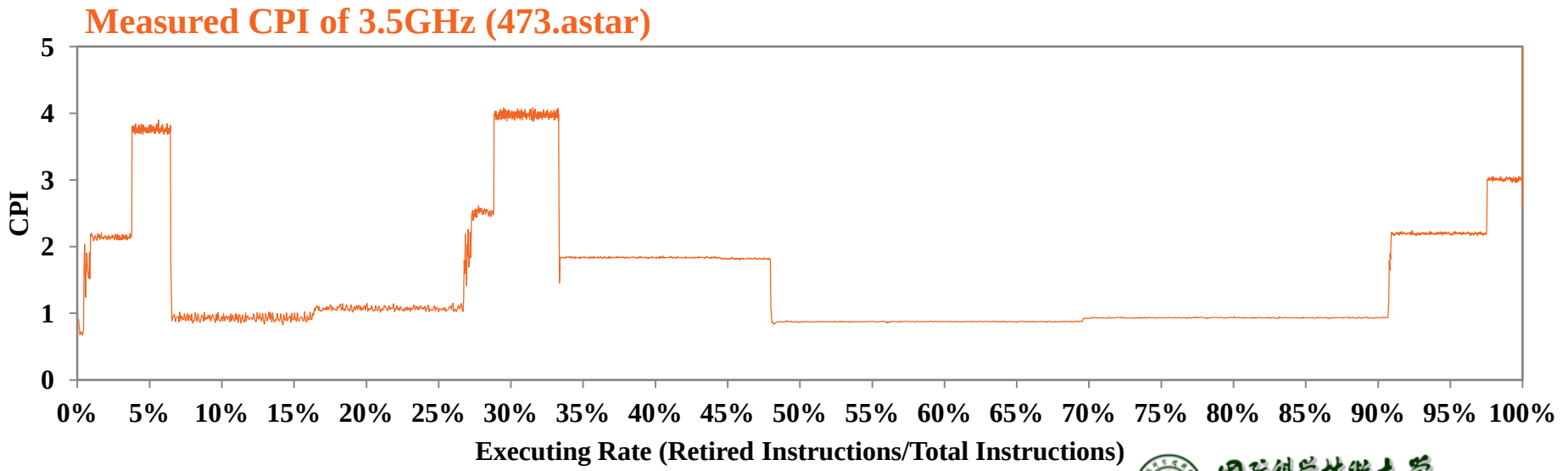


PREDICTED CPI V.S. MEASURED CPI

1.7GHZ -> 3.5GHZ

- a) Divide each curve into 20 segments;
- b) Compare cycle numbers in each segment-pair;
- c) Average absolute error => CPI Prediction Error.

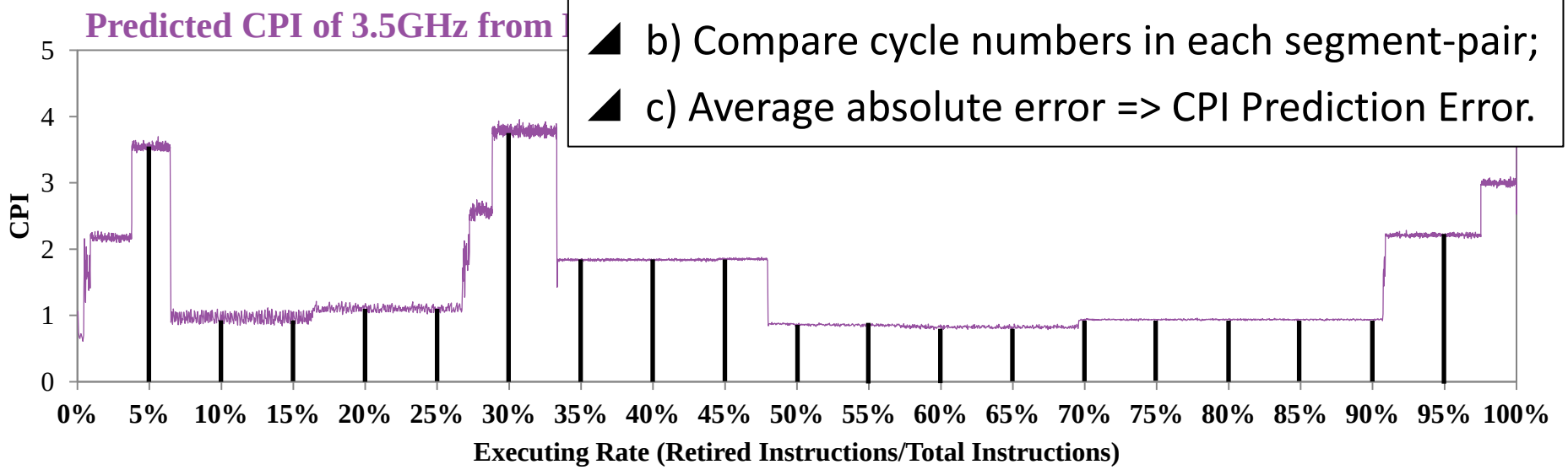


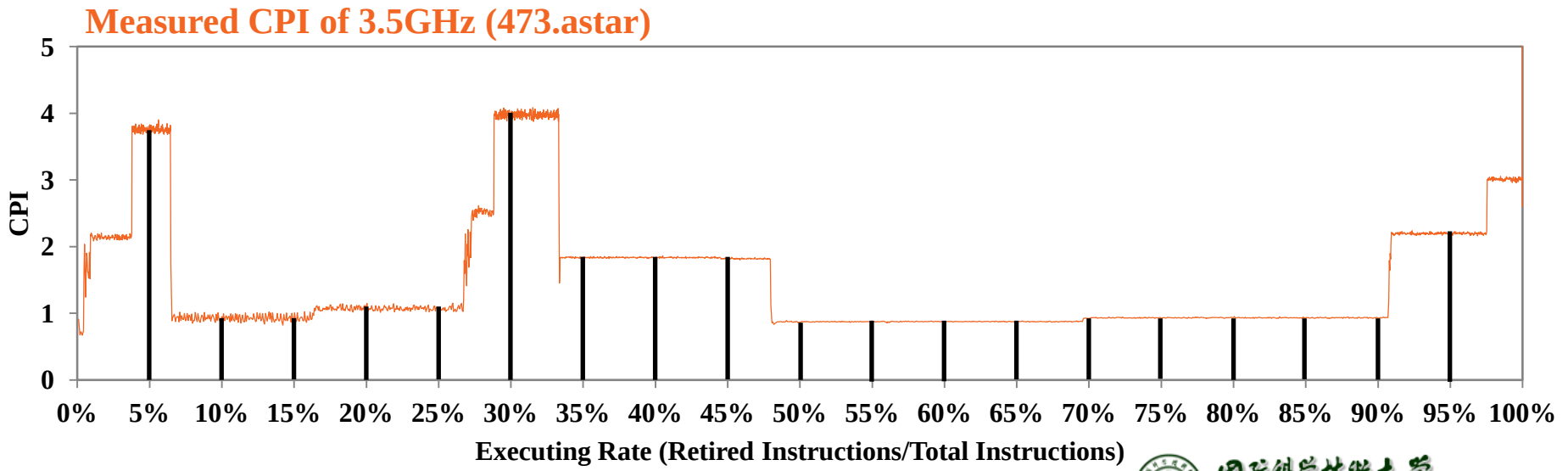


PREDICTED CPI V.S. MEASURED CPI

1.7GHZ -> 3.5GHZ

- a) Divide each curve into 20 segments;
- b) Compare cycle numbers in each segment-pair;
- c) Average absolute error => CPI Prediction Error.

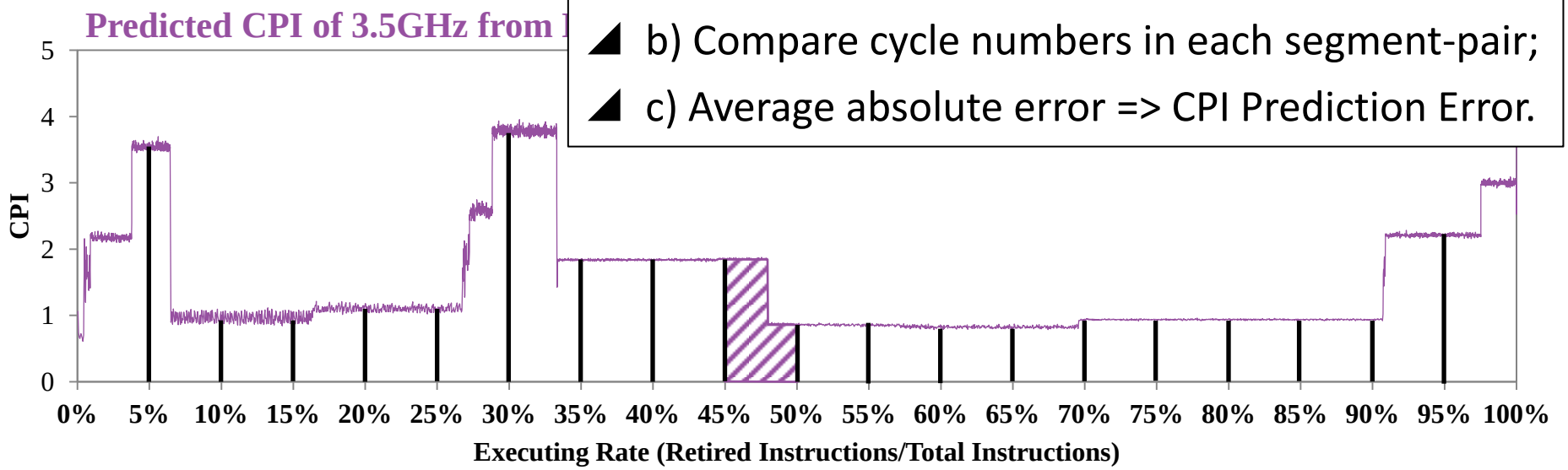


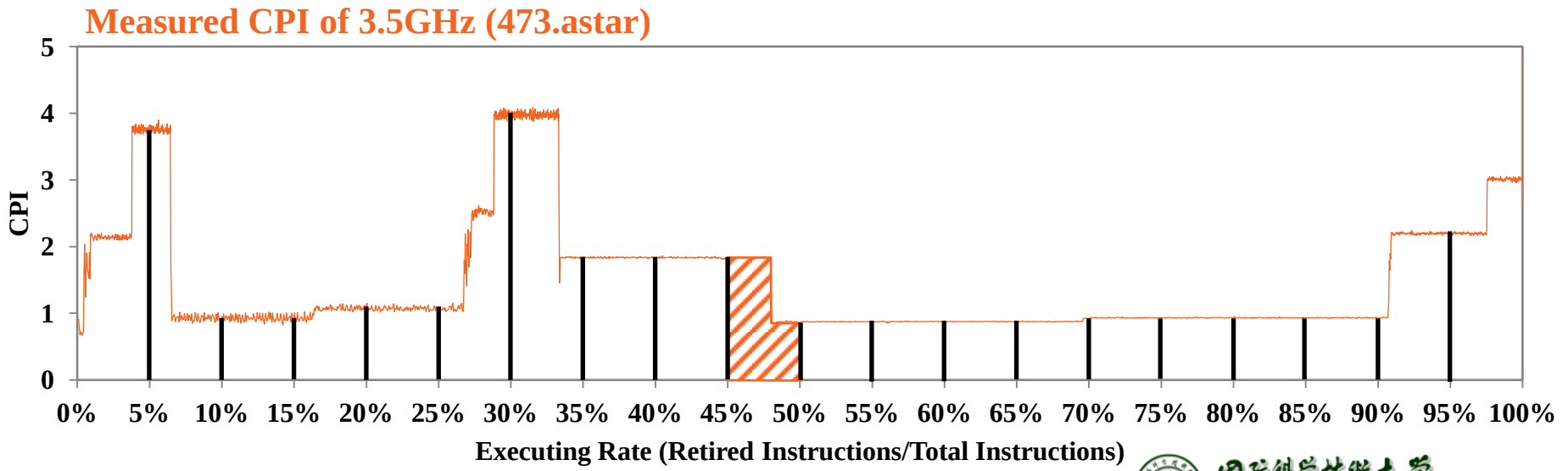


PREDICTED CPI V.S. MEASURED CPI

1.7GHZ -> 3.5GHZ

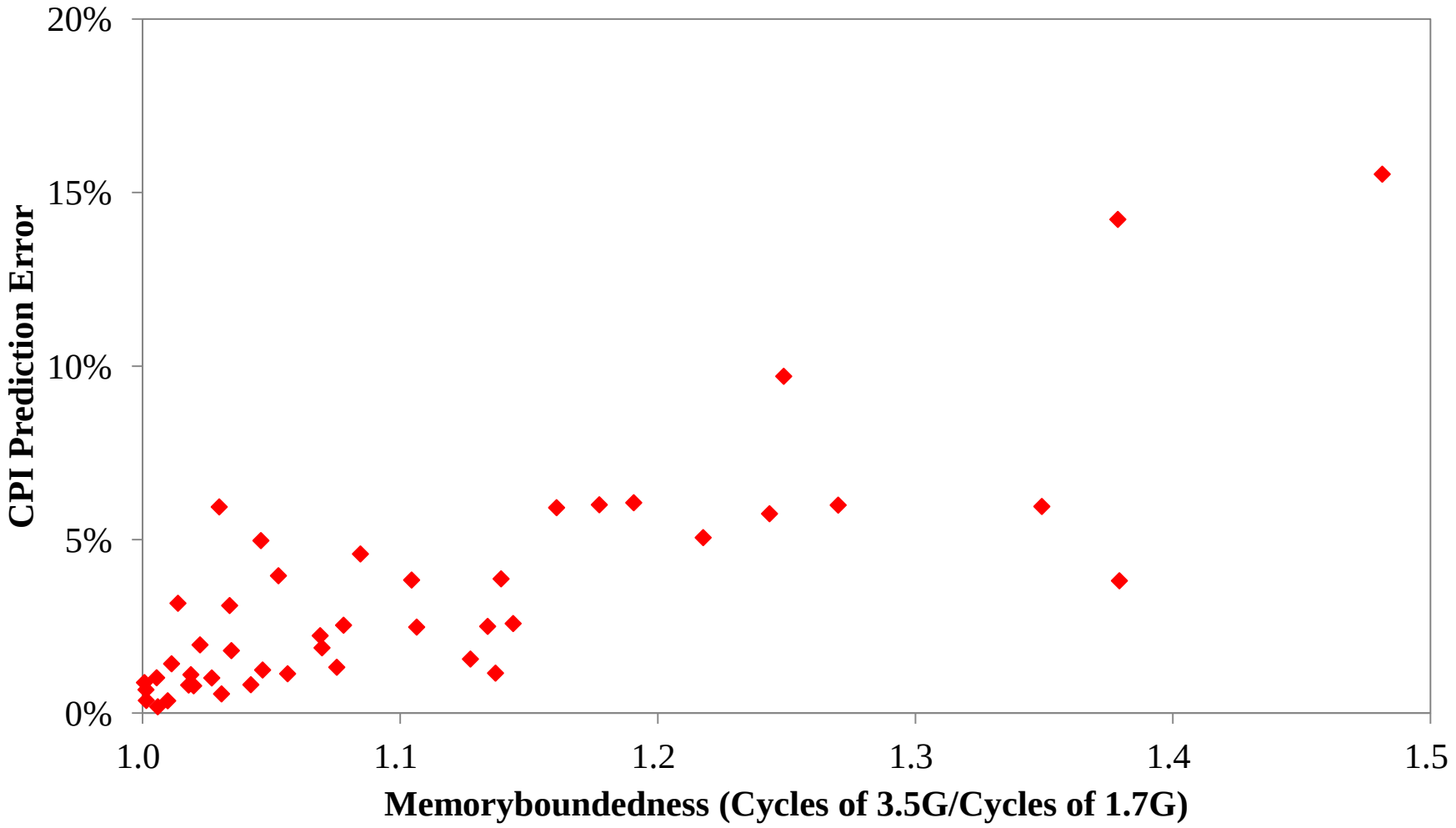
- a) Divide each curve into 20 segments;
- b) Compare cycle numbers in each segment-pair;
- c) Average absolute error => CPI Prediction Error.





CPI PREDICTION ERROR V.S. MEMORYBOUNDEDNESS

1.7GHZ -> 3.5GHZ



CHIP IDLE POWER MODEL

MODEL AND ERROR

AMD

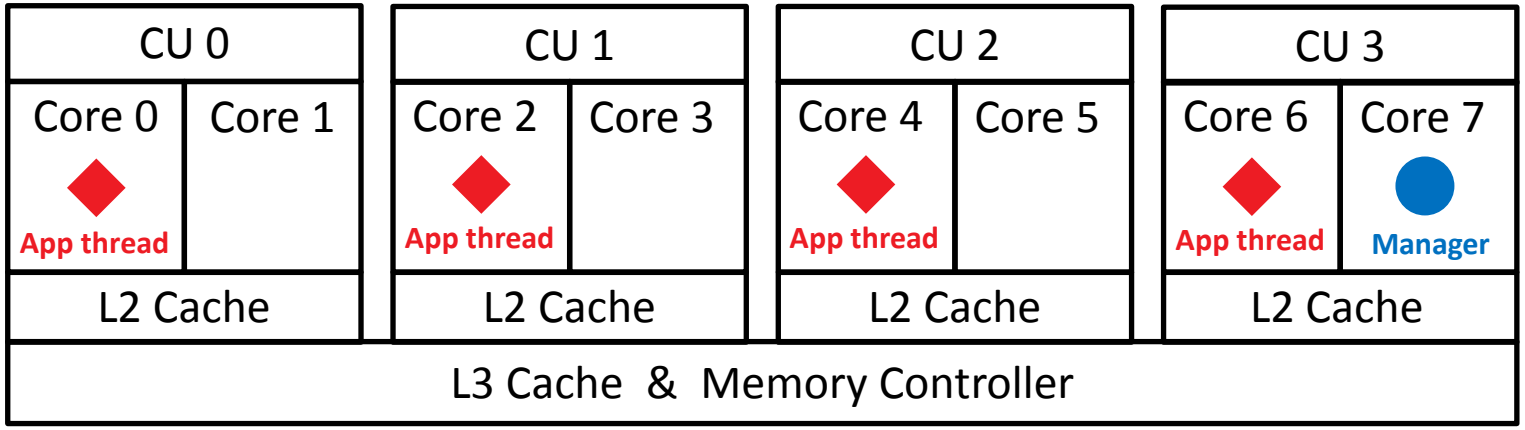
- Idle Power Estimation Model
  - $P_{idle}(VF_n) = W_{idle1}(VF_n) \times Temperature + W_{idle0}(VF_n)$
  - $W_{idle1}(VF_n) = a_3 V_n^3 + a_2 V_n^2 + a_1 V_n + a_0$
  - $W_{idle0}(VF_n) = b_3 V_n^3 + b_2 V_n^2 + b_1 V_n + b_0$

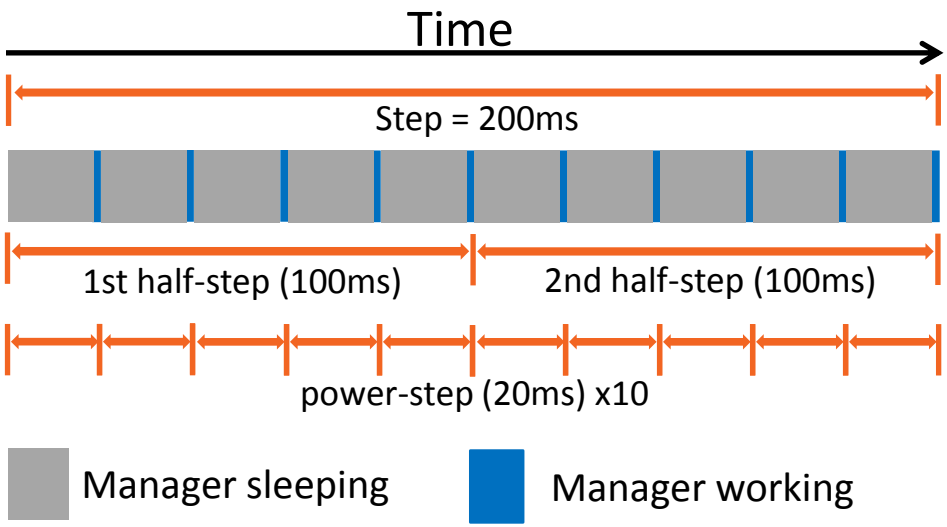
CHIP DYNAMIC POWER MODEL

EXPERIMENT

AMD

App threads mapping & Manager



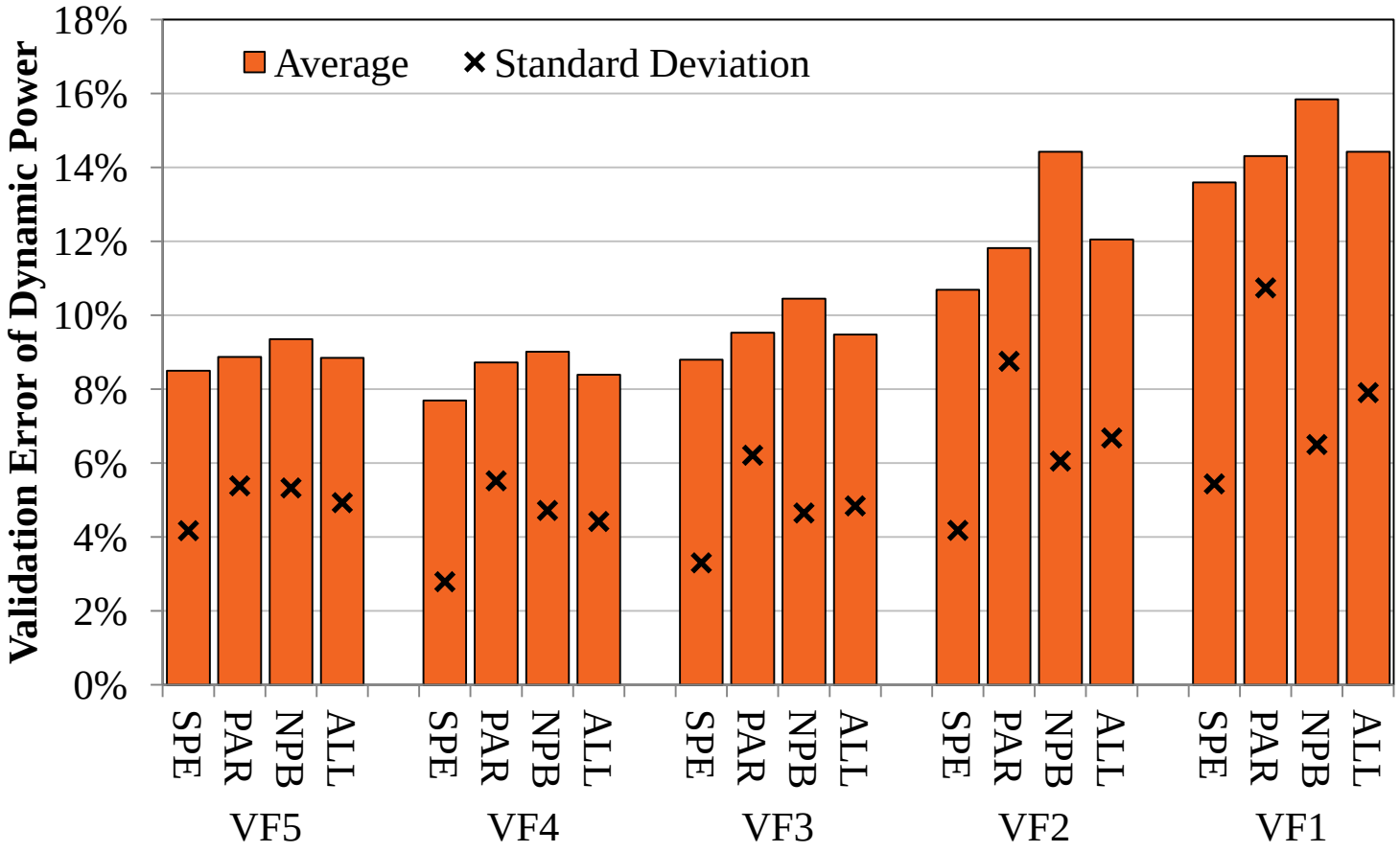


- Step (200ms)
  - Count events on each core
  - Get P_{chip} and Temperature
- 2 Half-steps (100ms each)
  - Time-multiplexing the counters
  - Count 6 events on each core
- 10 Power-steps (20ms each)
  - Read power from microcontroller
  - P_{chip} is the average power

FITTING ERROR: DYNAMIC POWER MODEL

(LOWER IS BETTER)

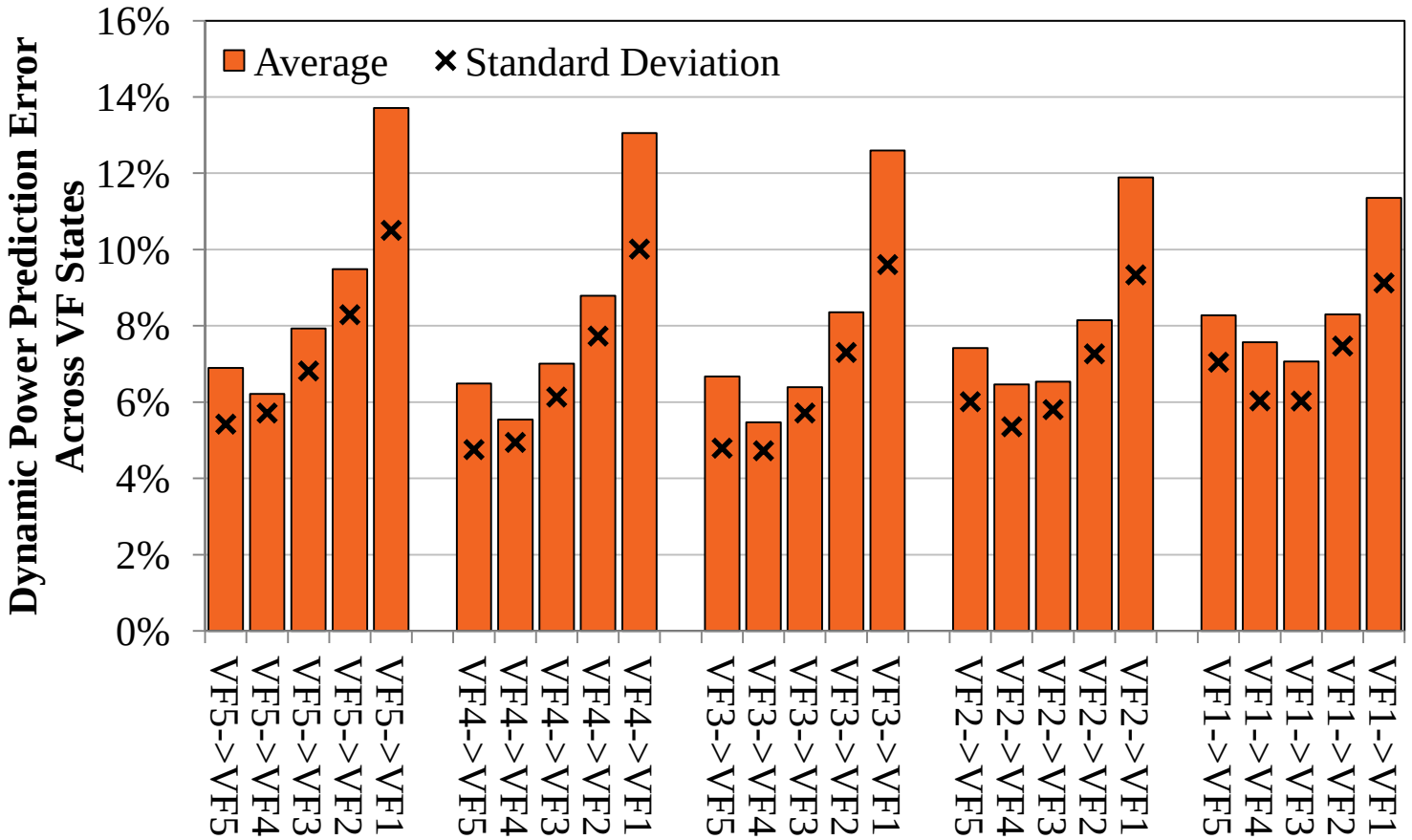
- Error: 10.6%
- Standard Deviation: 5.8%



PREDICTING ERROR: AVERAGE P_{dyn}

(LOWER IS BETTER)

- Error: 8.3%
- Standard Deviation: 6.9%



THE IMPACT OF POWER GATING

AMD

- Getting per-core power
  - P_{dyn} ---- has already split to cores naturally
  - P_{idle} ---- what is the quotas for each core and the north bridge?
- Splitting the P_{idel}
  - With the help of *Power Gating* feature of the processor
- Enabling Power Gating (PG) in FX-8320
  - Per-CU Power Gating
  - CU: Both cores in a CU are idle => CU is power gated
  - NB: All CUs are idle => NB is power gated

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

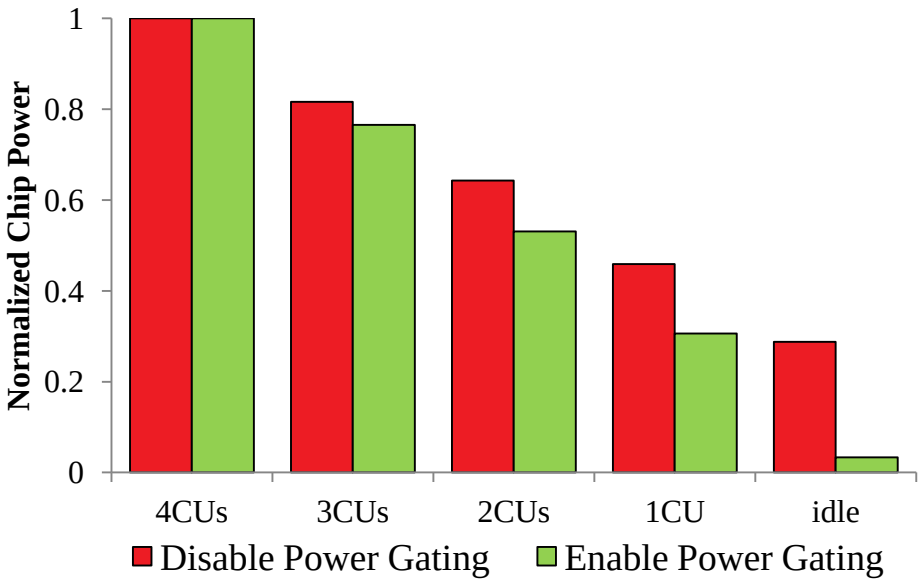
A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

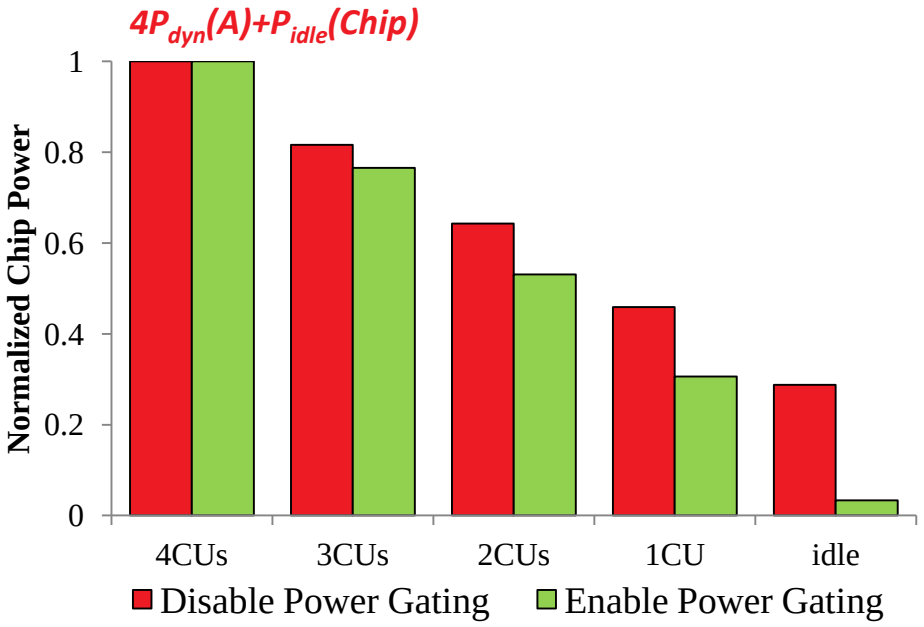


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

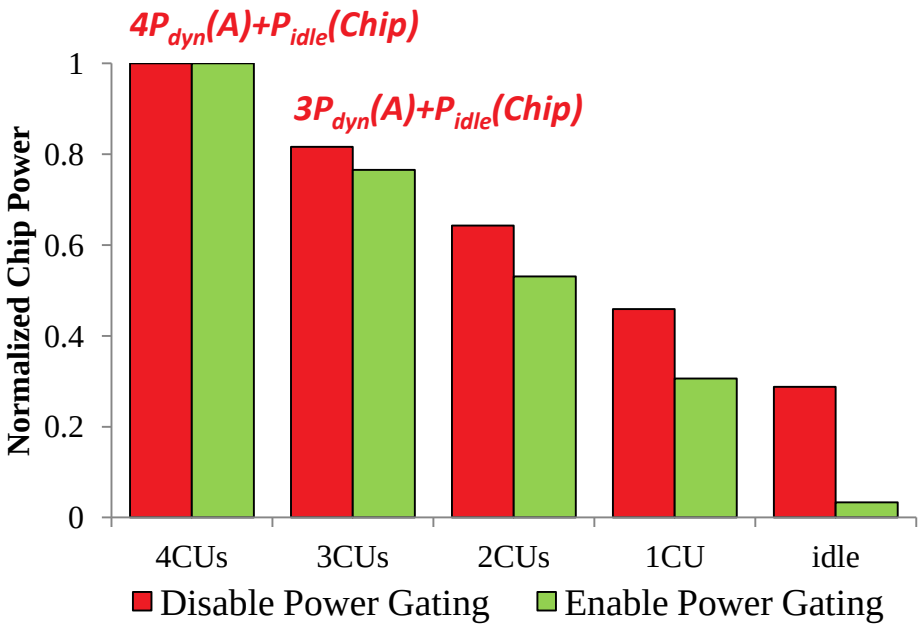


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

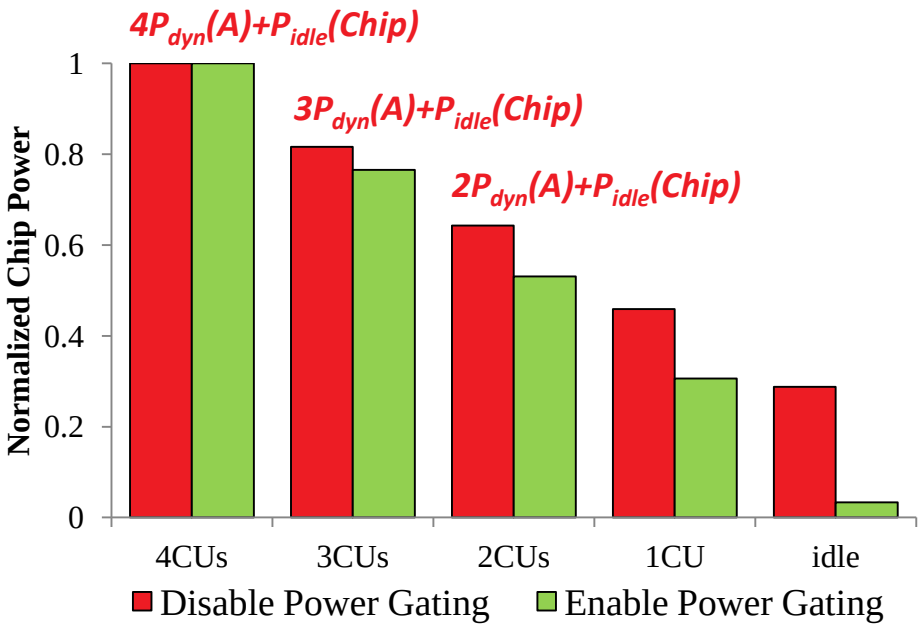


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

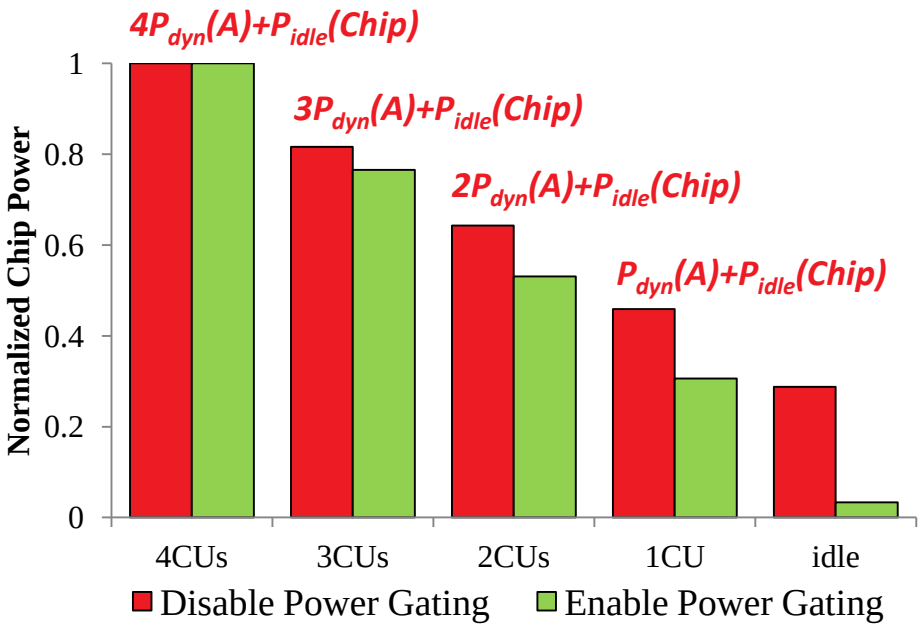


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

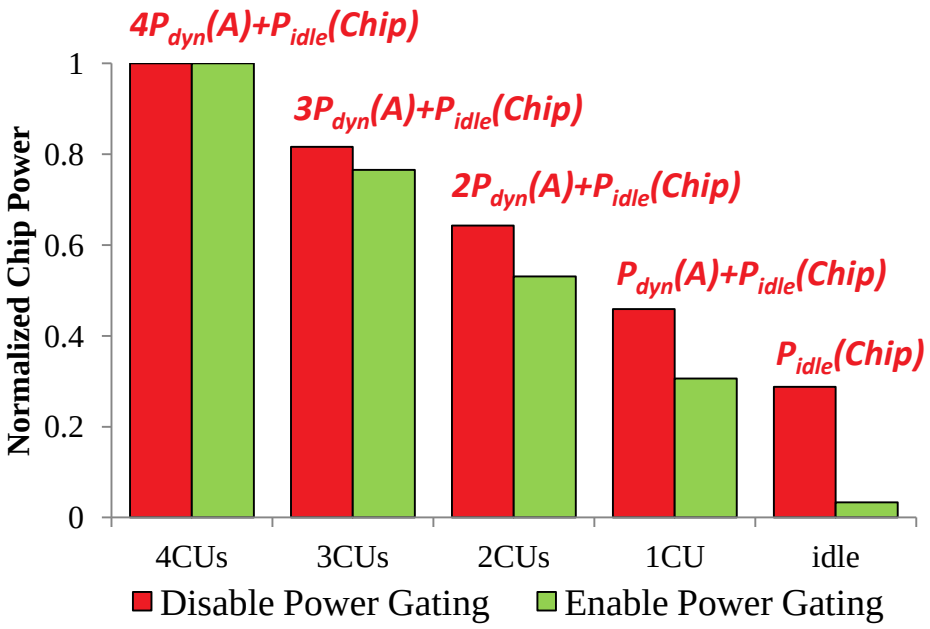


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

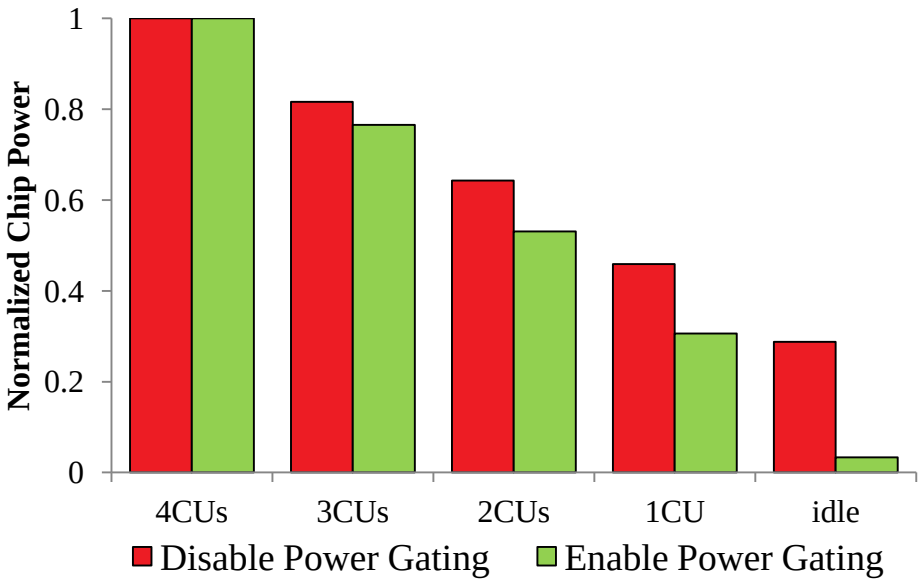


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

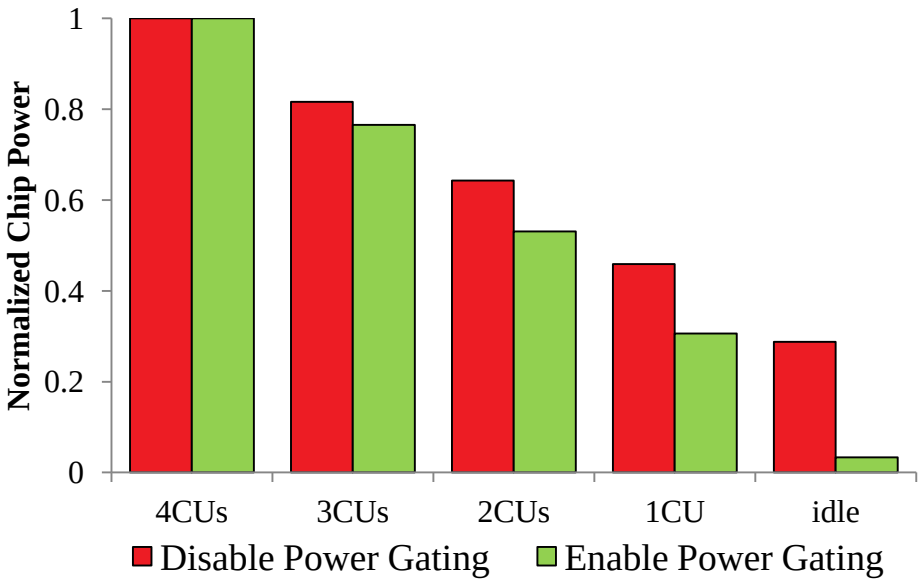


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

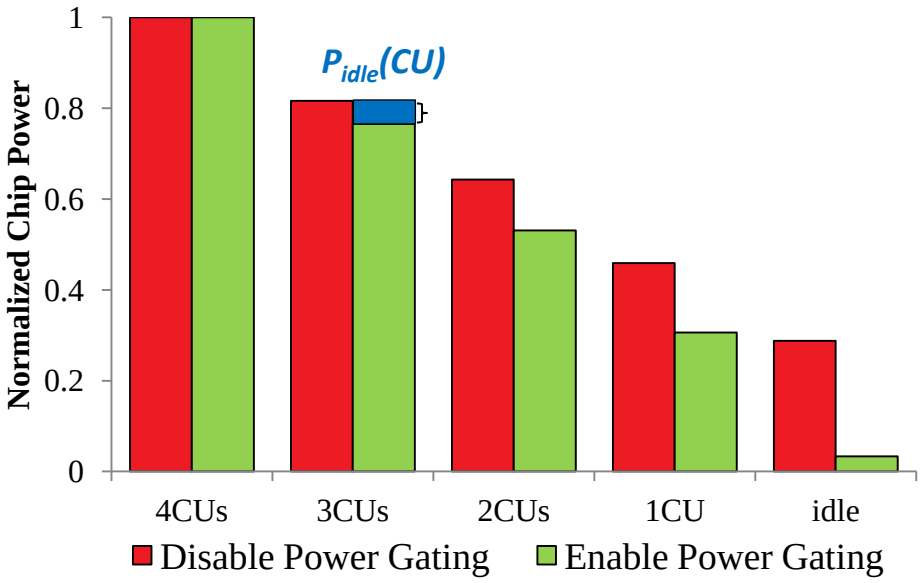


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

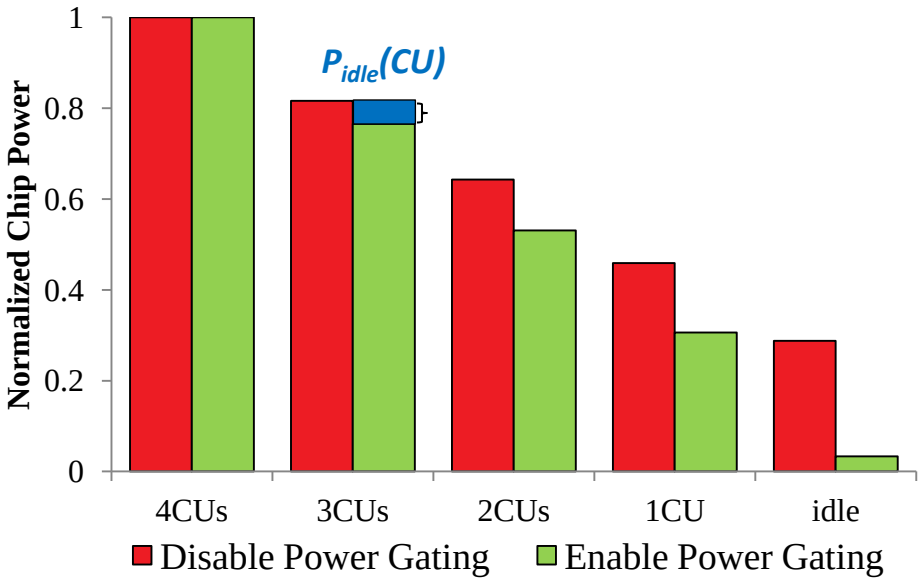


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

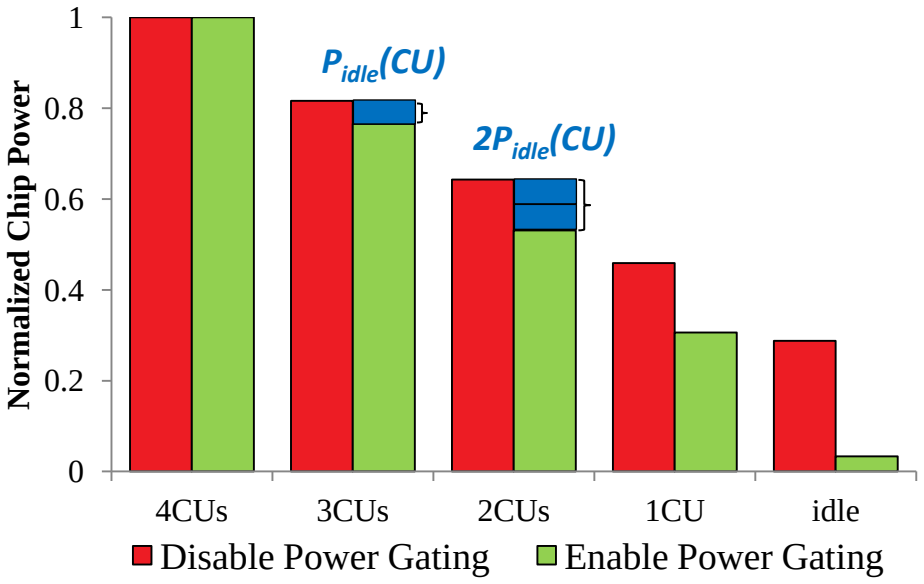


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

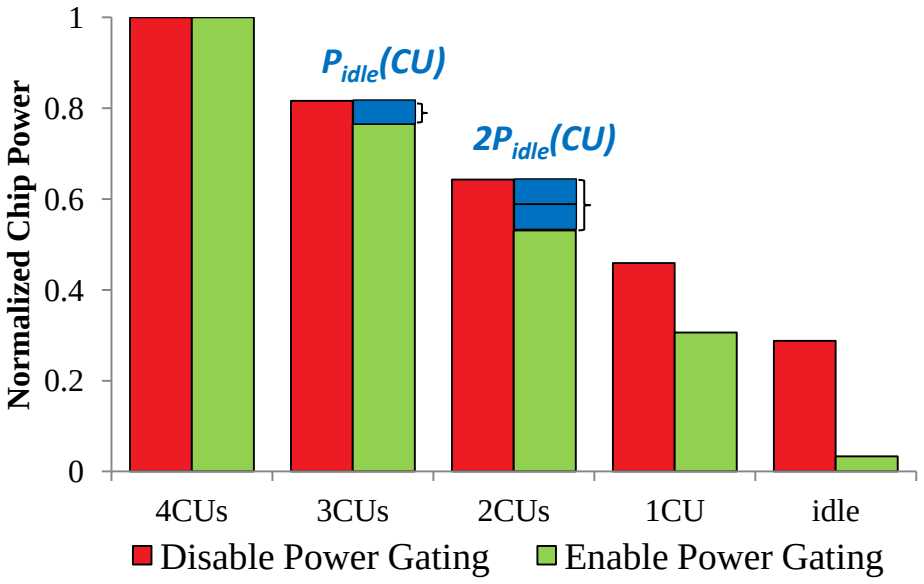


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

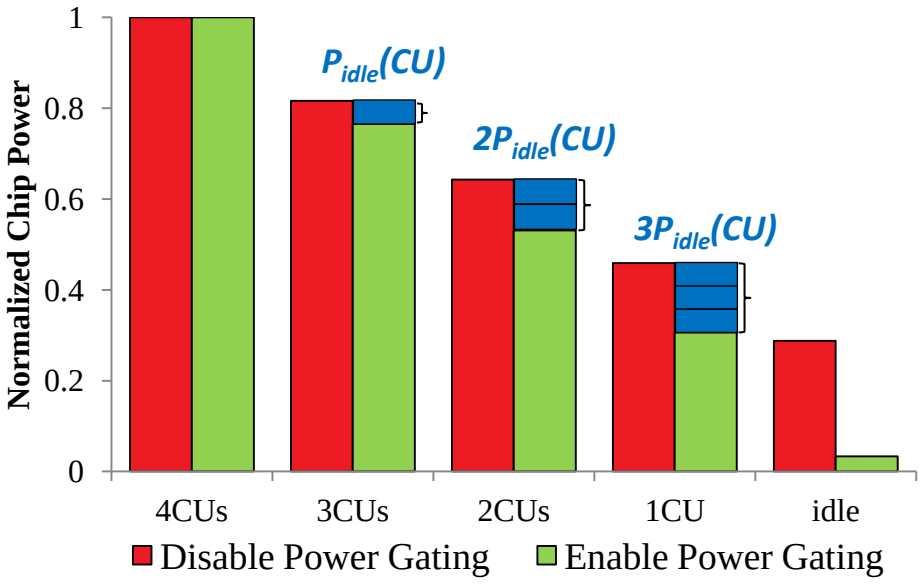


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

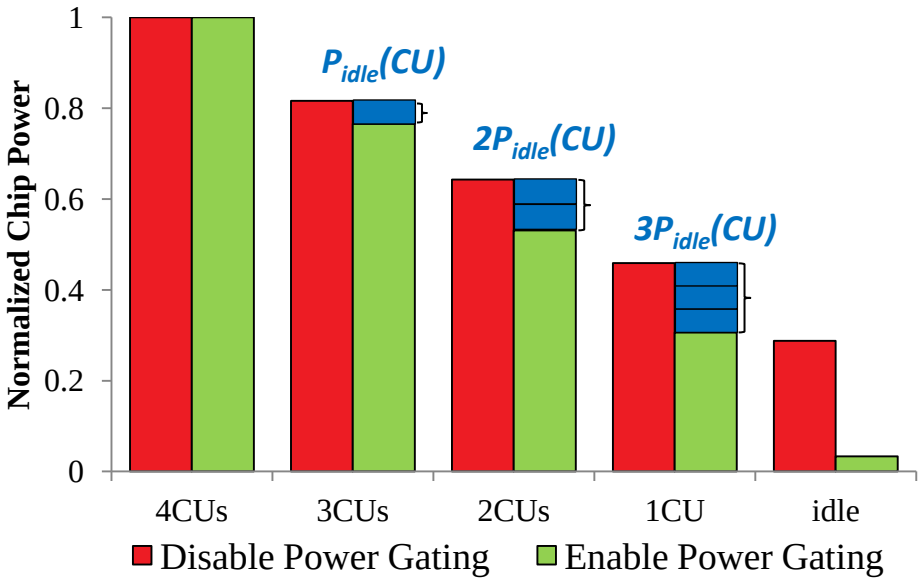


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle



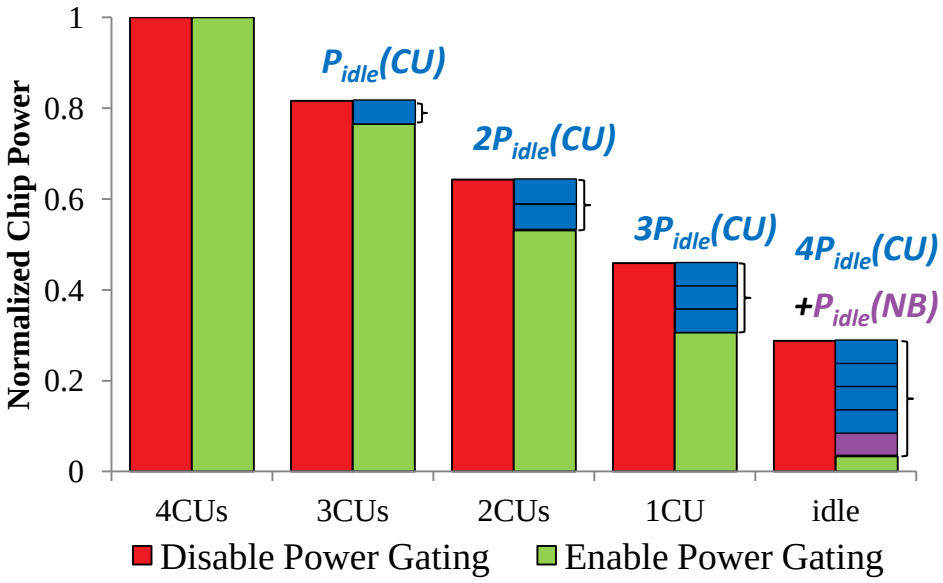
國防科學技術大學 National University of Defense Technology

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

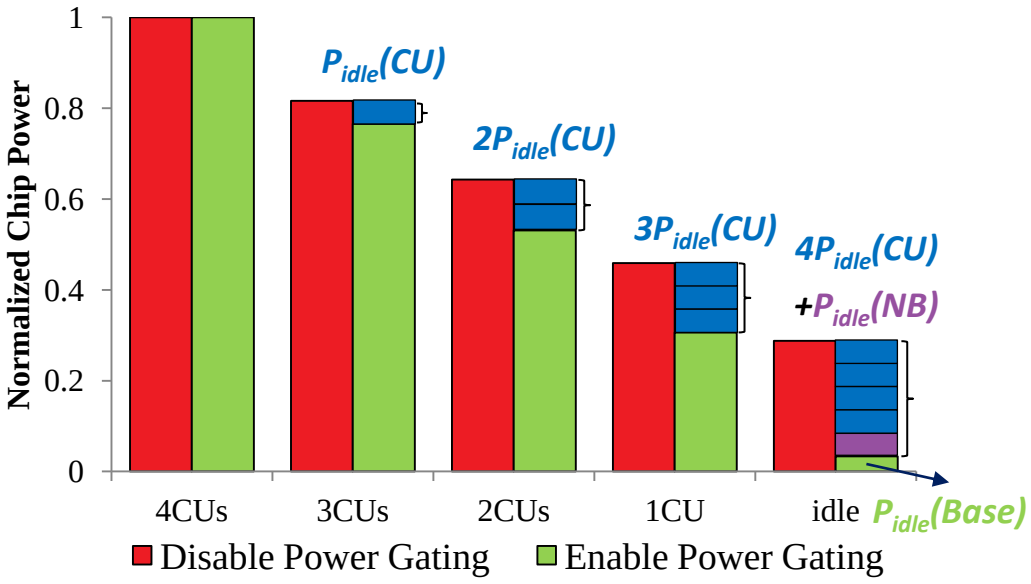


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: “A”
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of “A” times the dynamic power of “A” by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle

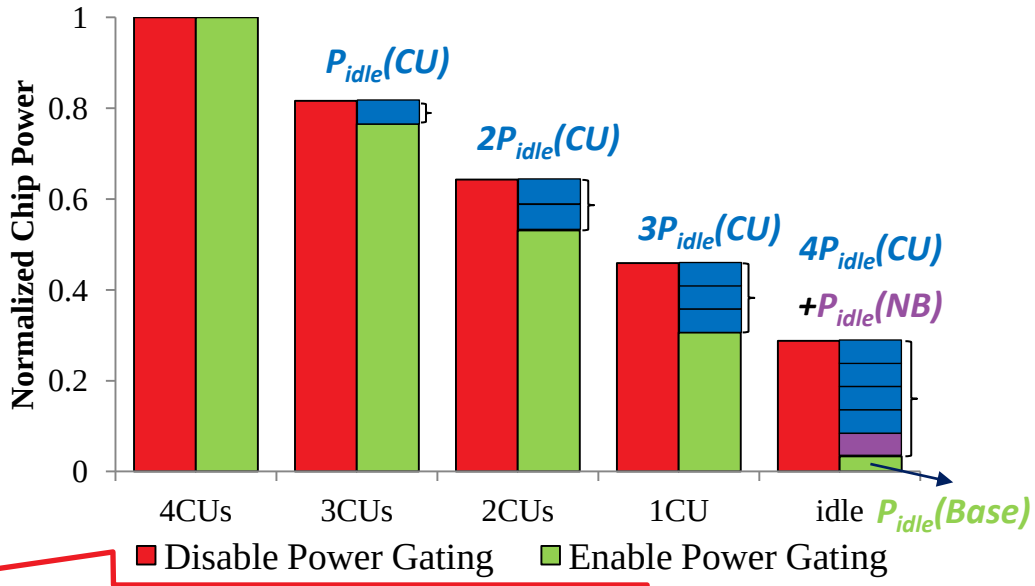


THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: "A"
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of "A" times the dynamic power of "A" by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle



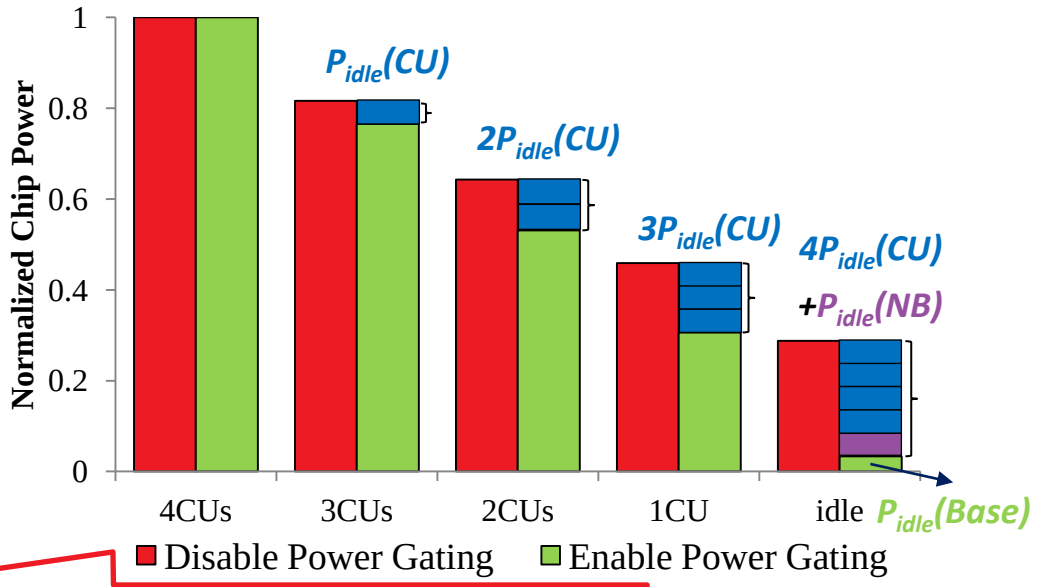
Get $P_{idle}(CU)$, $P_{idle}(NB)$, $P_{idle}(Base)$

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: "A"
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of "A" times the dynamic power of "A" by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle



n = busy cores in Chip

m = busy cores in CU

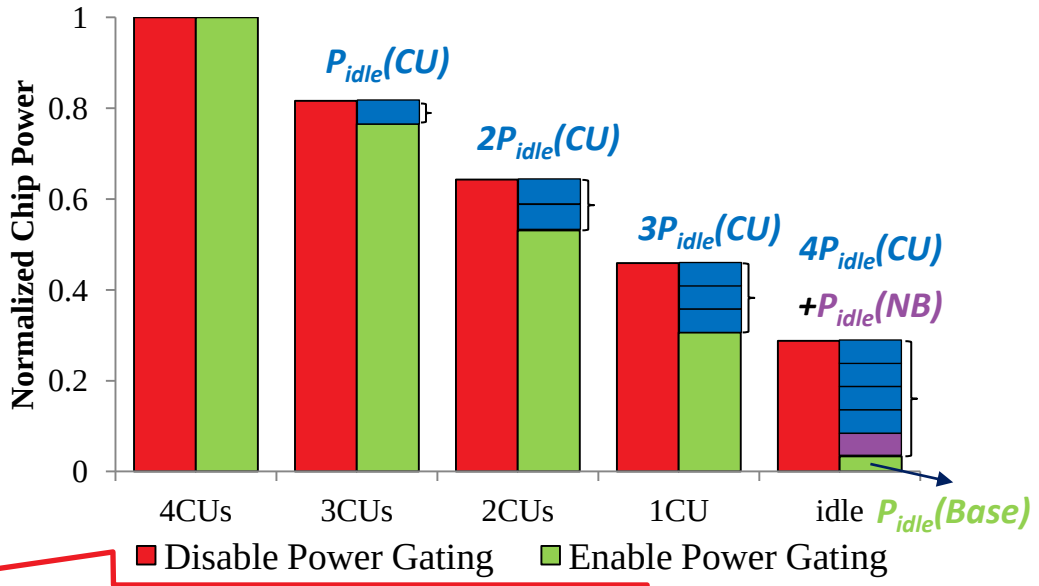
Get $P_{idle}(CU)$, $P_{idle}(NB)$, $P_{idle}(Base)$

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: "A"
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of "A" times the dynamic power of "A" by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle



n = busy cores in Chip

m = busy cores in CU

Get $P_{idle}(CU)$, $P_{idle}(NB)$, $P_{idle}(Base)$

When Power Gating is disabled:

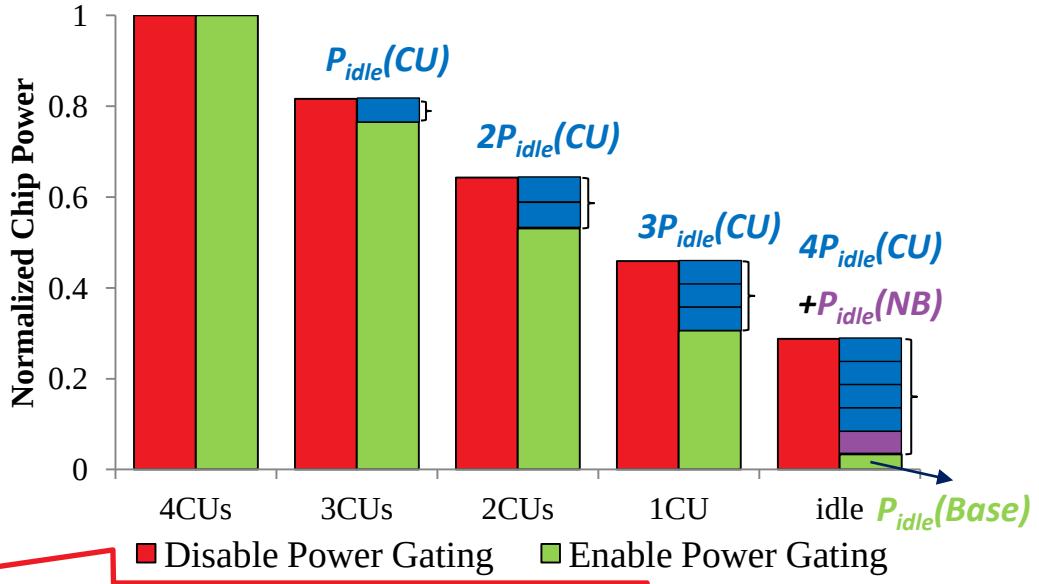
$$P_{idle}(core) = \frac{4P_{idle}(CU) + P_{idle}(NB) + P_{idle}(Base)}{n}$$

THE IMPACT OF POWER GATING

AMD

- Micro Benchmark: "A"
  - Dataset is fitted in L1 data cache => No NB activity (dynamic power)
  - N instances of "A" times the dynamic power of "A" by N

A2C Mapping	CU0 C0, C1	CU1 C2, C3	CU2 C4, C5	CU3 C6, C7
4CUs	A, -	A, -	A, -	A, -
3CUs	A, -	A, -	A, -	Idle
2CUs	A, -	A, -	Idle	Idle
1CU	A, -	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Idle



n = busy cores in Chip

m = busy cores in CU

Get $P_{idle}(CU)$, $P_{idle}(NB)$, $P_{idle}(Base)$

When Power Gating is disabled:

$$P_{idle}(core) = \frac{4P_{idle}(CU) + P_{idle}(NB) + P_{idle}(Base)}{n}$$

When Power Gating is enabled:

$$P_{idle}(core) = \frac{P_{idle}(CU)}{m} + \frac{P_{idle}(NB) + P_{idle}(Base)}{n}$$

PPEP FRAMEWORK
COMPUTATION COMPLEXITY

AMD

- Computation complexity on each core

For the current VF state	ADD	MUL	DIV
CPI calculation			1
Power calculation	18	29	
Total	18	29	1

For every other VF state	ADD	MUL	DIV
CPI prediction	4	1	11
Power prediction	18	39	1
Total	22	40	12

- If core number is ***c***, VF number is ***k***

OP	Number	Example: FX-8320, c=8, k=5
ADD	c*(18+22*(k-1))	848
MUL	c*(29+40*(k-1))	1512
DIV	c*(1+12*(k-1))	392

<3k FPOPs Per-step

(computing flow w/o optimized)

PPEP FRAMEWORK
RUNNING OVERHEAD

AMD

Measured Running Overhead

Overhead of PPEP (FX-8320, VF5)	Step = 200ms	Step = 20ms
Power	Negligible	1.3W
Performance (App and PPEP is not attached on a same core)	Negligible	Negligible
Performance (App and PPEP is attached on a same core)	1.6% slowdown	5.8% slowdown

PPEP USAGE EXAMPLE: ONE-STEP POWER CAPPING

AMD

- PPEP could rapidly reach the cap