AgileWatts

An Energy-Efficient CPU Core Idle-State Architecture for Latency-Sensitive Server Applications

Jawad Haj-Yahya[4]

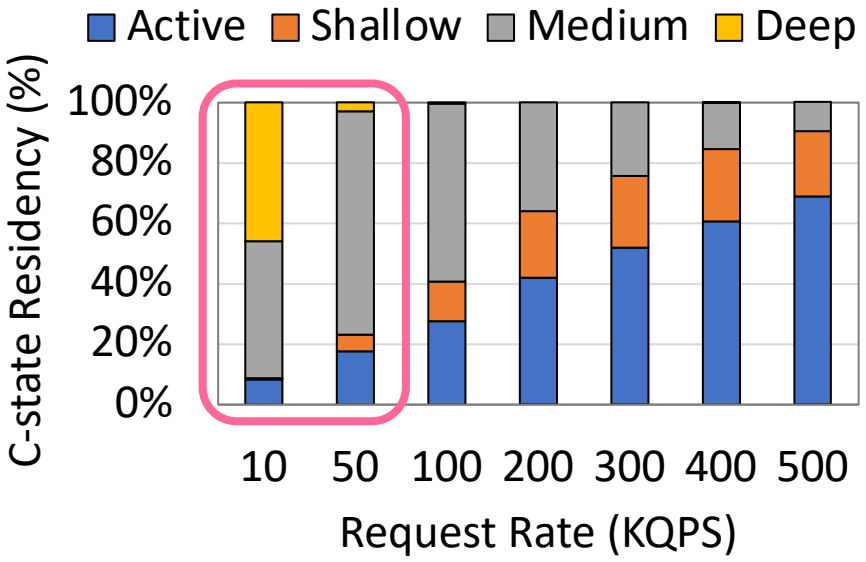
Haris Volos[2] *Davide B. Bartolini*[1] *Georgia Antoniou*[2]

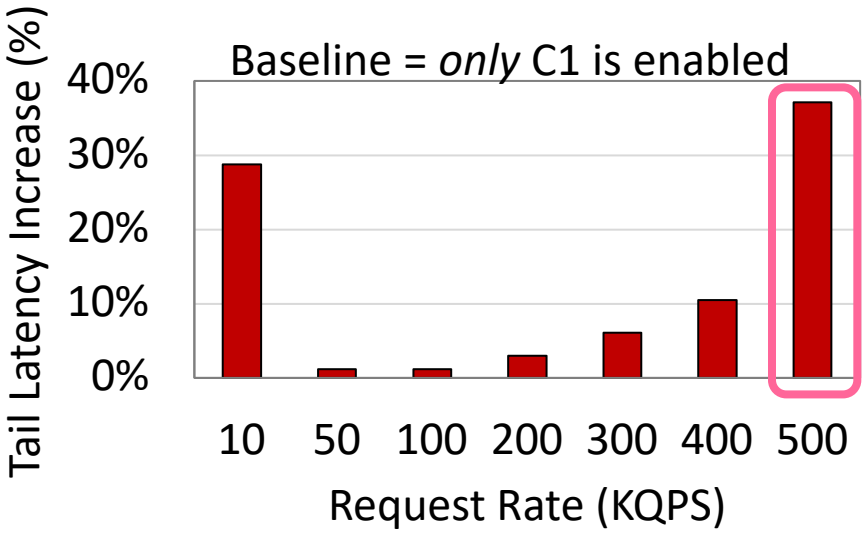
Jeremie S. Kim[3] *Zhe Wang*[1] *Kleovoulos Kalaitzidis*[1] *Tom Rollet*[1]

Zhirui Chen[1] *Ye Geng*[1] *Onur Mutlu*[3] *Yiannakis Sazeides*[2]

1 HUAWEI 2 University of Cyprus 3 ETH zürich 4 RiVOS

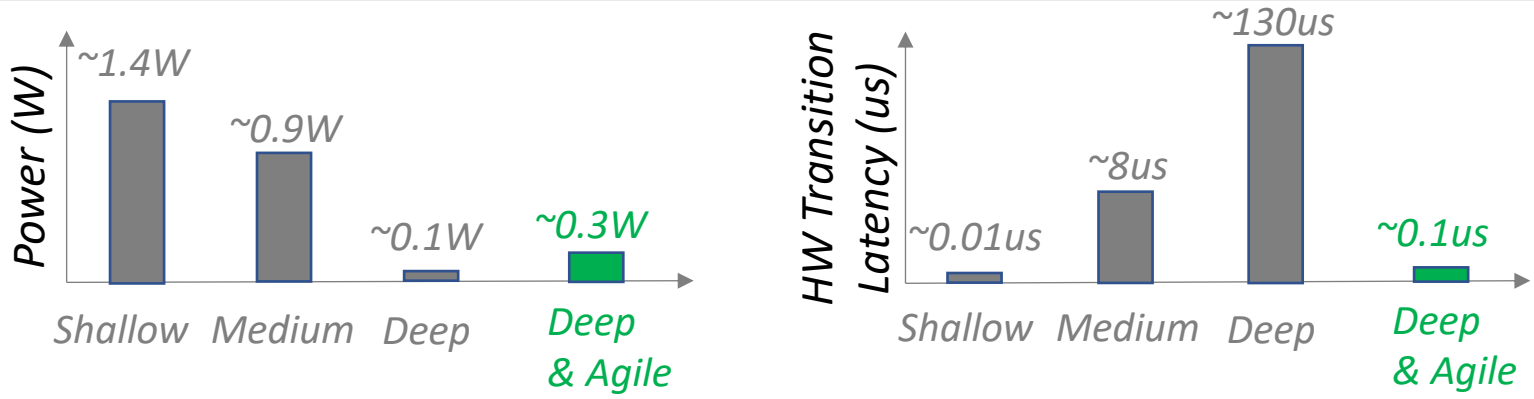
Motivation





- The load of user-facing applications is unpredictable and bursty
  - with microsecond-scale idleness – as known as the *"Killer microseconds" idleness*
- This load behavior prevents CPU cores from entering deep idle states during idle periods, limiting power savings when the CPU core is idle
  - A CPU core running Memcached never enters Deep idle state when running at ≥20% load
- Idle states transition times can increase tail latency significantly
  - The tail latency of Memcached increases by up to 37% when enabling Medium and Deep idle states

Our Goal



- To eliminate the *killer microseconds effect* that prevent servers running latency critical application from entering deep energy saving states, we propose AgileWatts:
  - A Deep & Agile low power state with
  - Nanosecond-scale transition latency

- We design AgileWatts with two design goals in mind:

  1. Drastically reducing the transition latency of deep core idle power states, making deep C-states usable

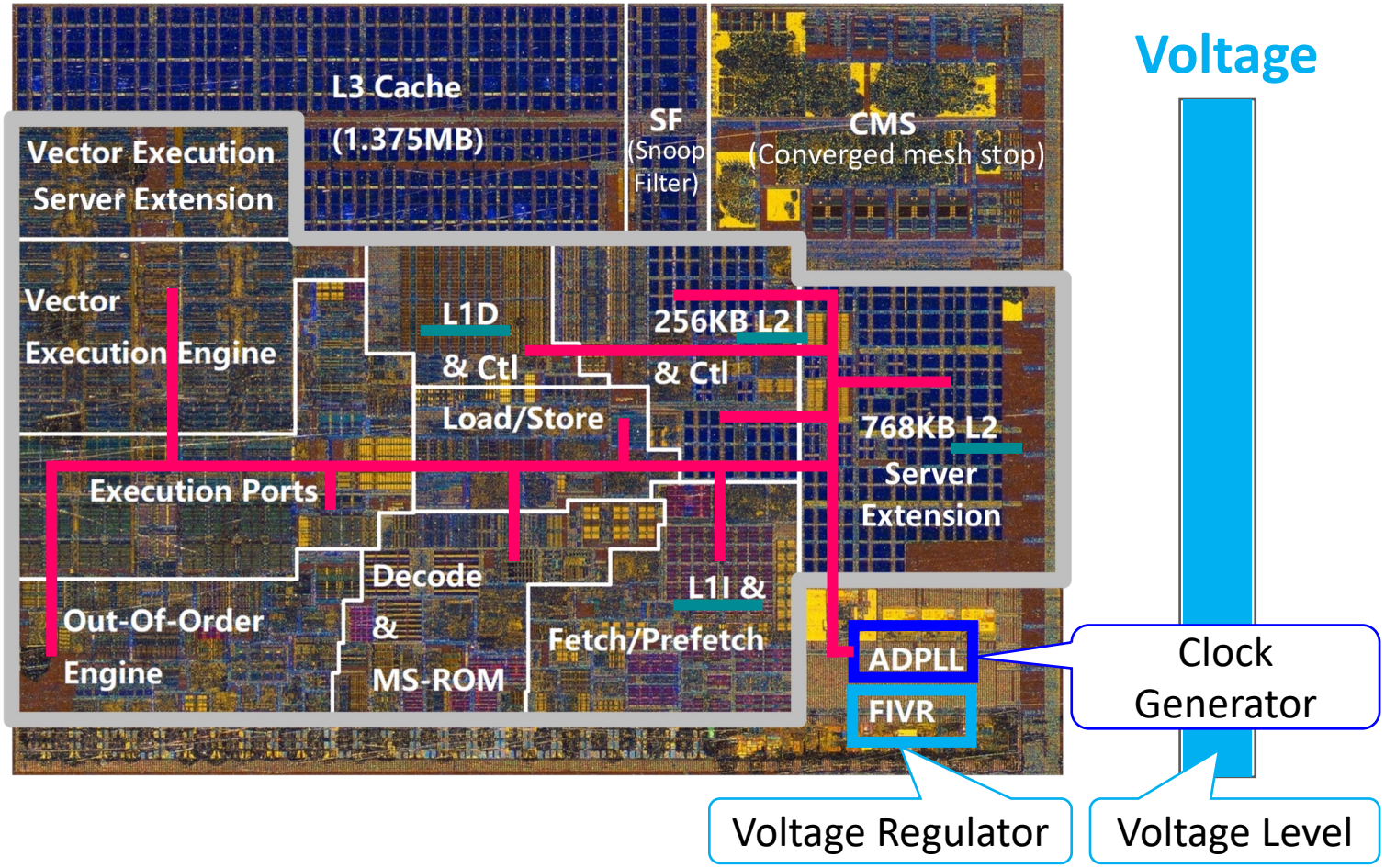
  2. Retaining most of the power savings of deep idle states

Core Idle Power State – Core C-states

- Core C-states are power saving states enable the core to reduce its power consumption during idle periods

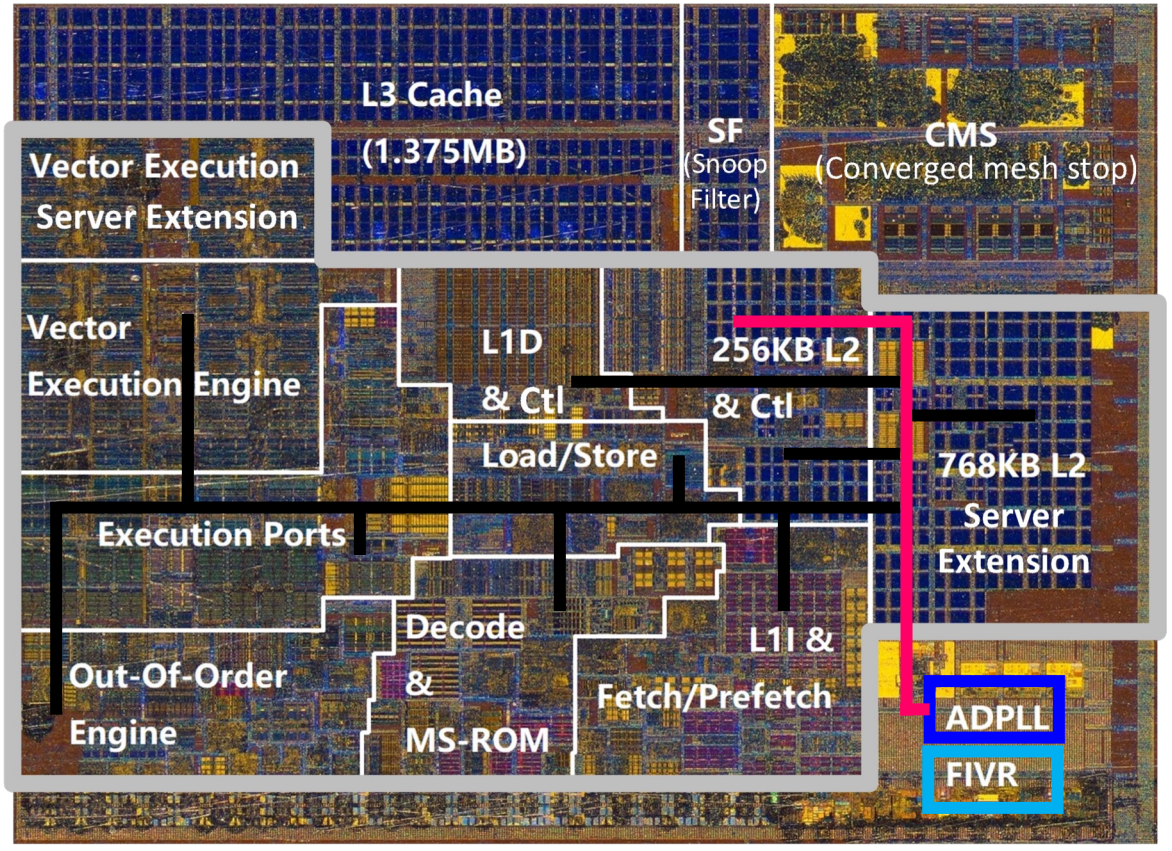
- Intel's Skylake architecture offers four main Core C-state:
  - C0 - Active
  - C1 – Shallow
  - C1E – Medium
  - C6 - Deep

C0 (Active) Core C-state



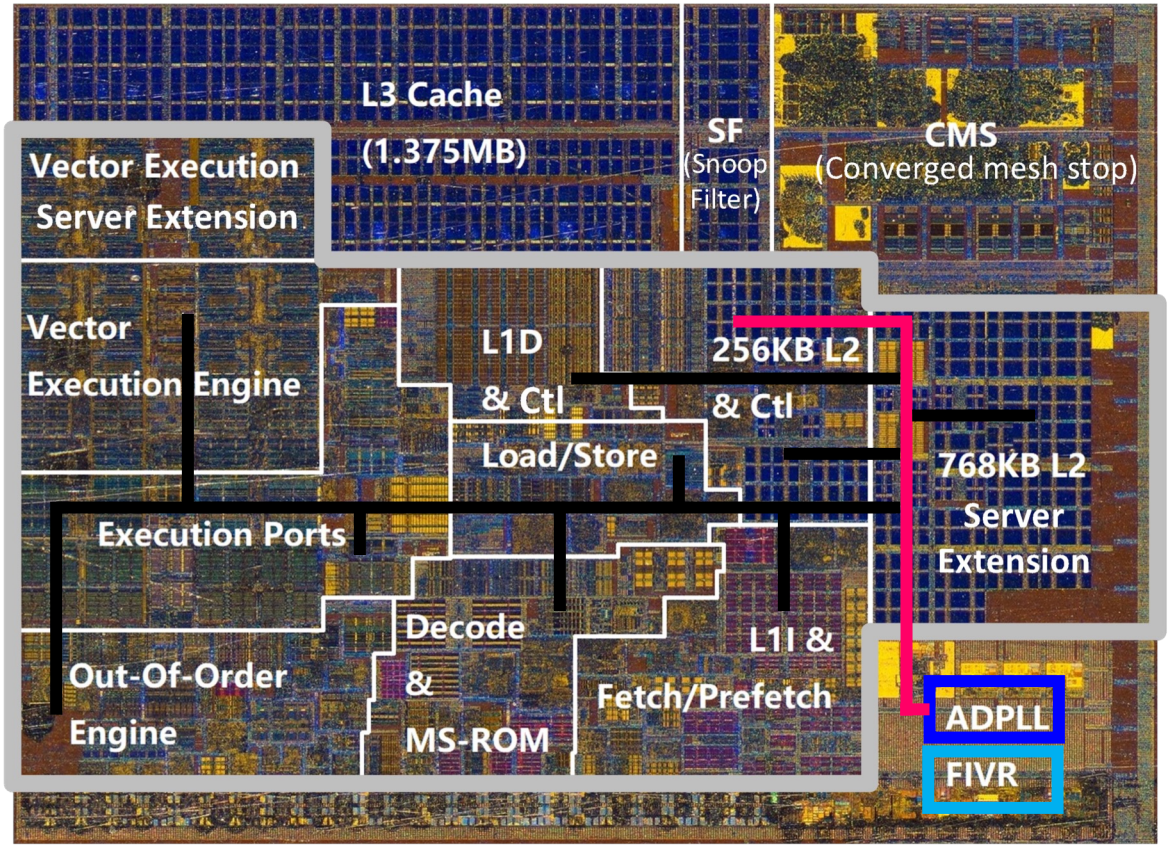
C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C0	Running	On	Coherent	Nominal	Maintained

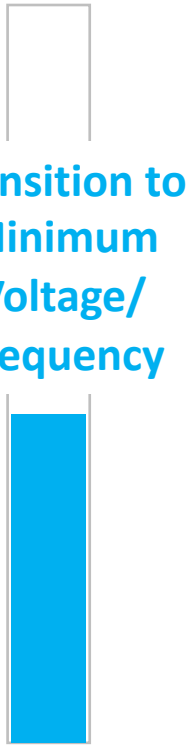
C1 (Shallow) Core C-states



C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C1	Most Stopped	On	Coherent	Nominal	Maintained

C1E (Medium) Core C-state

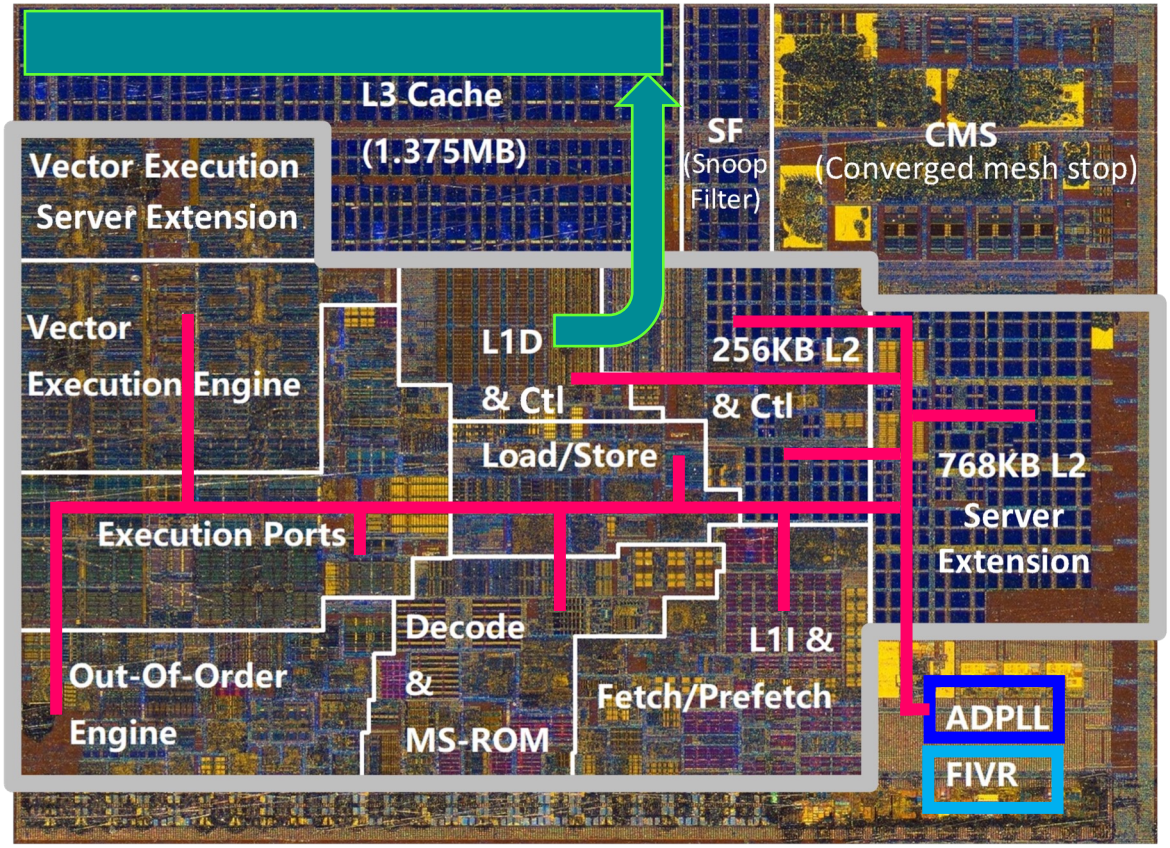




C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C1E	Most Stopped	On	Coherent	Min V/F	Maintained

C6 (Deep) Core C-state

Flush L1/L2 Caches

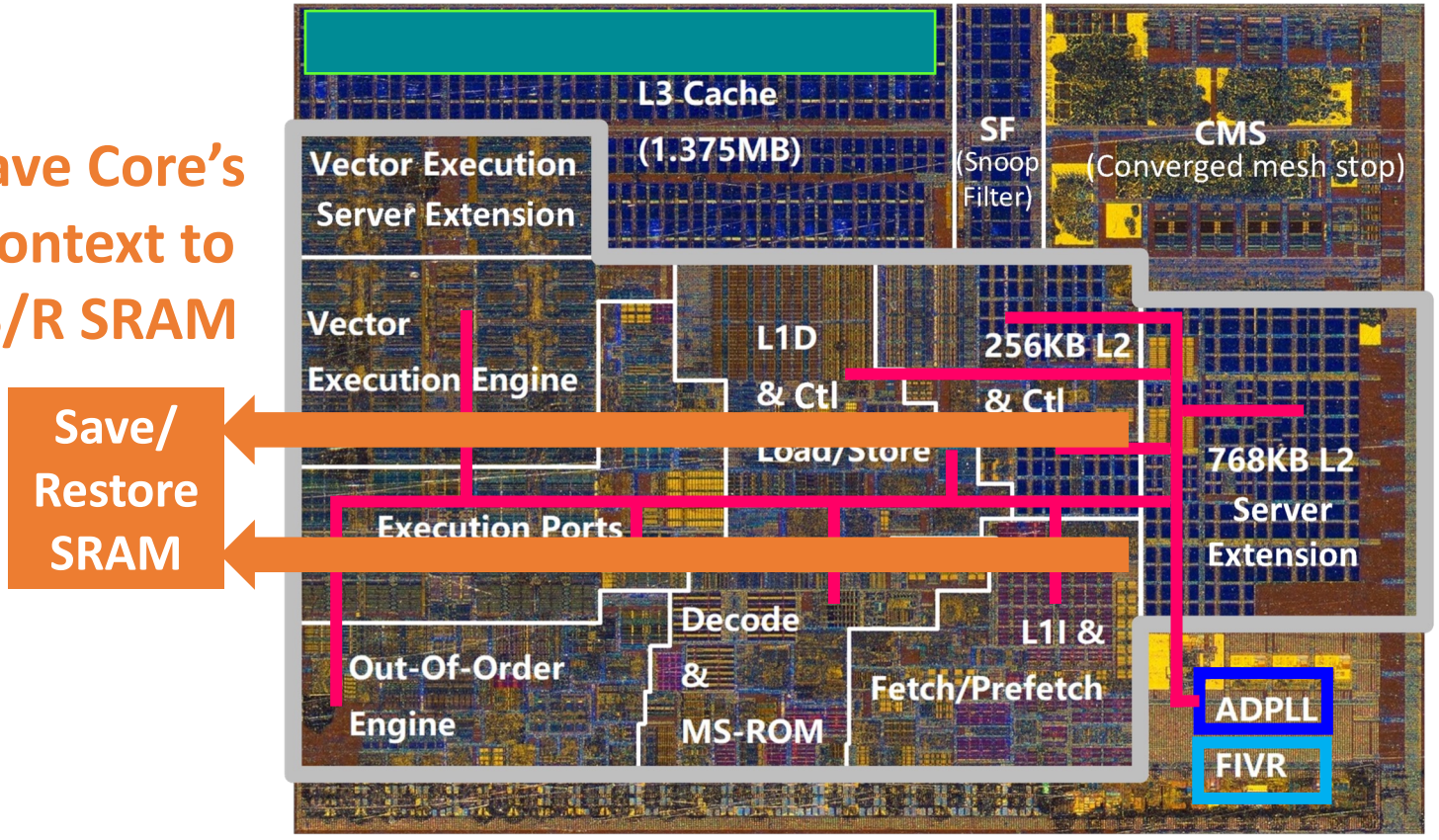


Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6	Running	On	Flushed	Nominal	Maintained

C6 (Deep) Core C-state

Save Core's Context to S/R SRAM



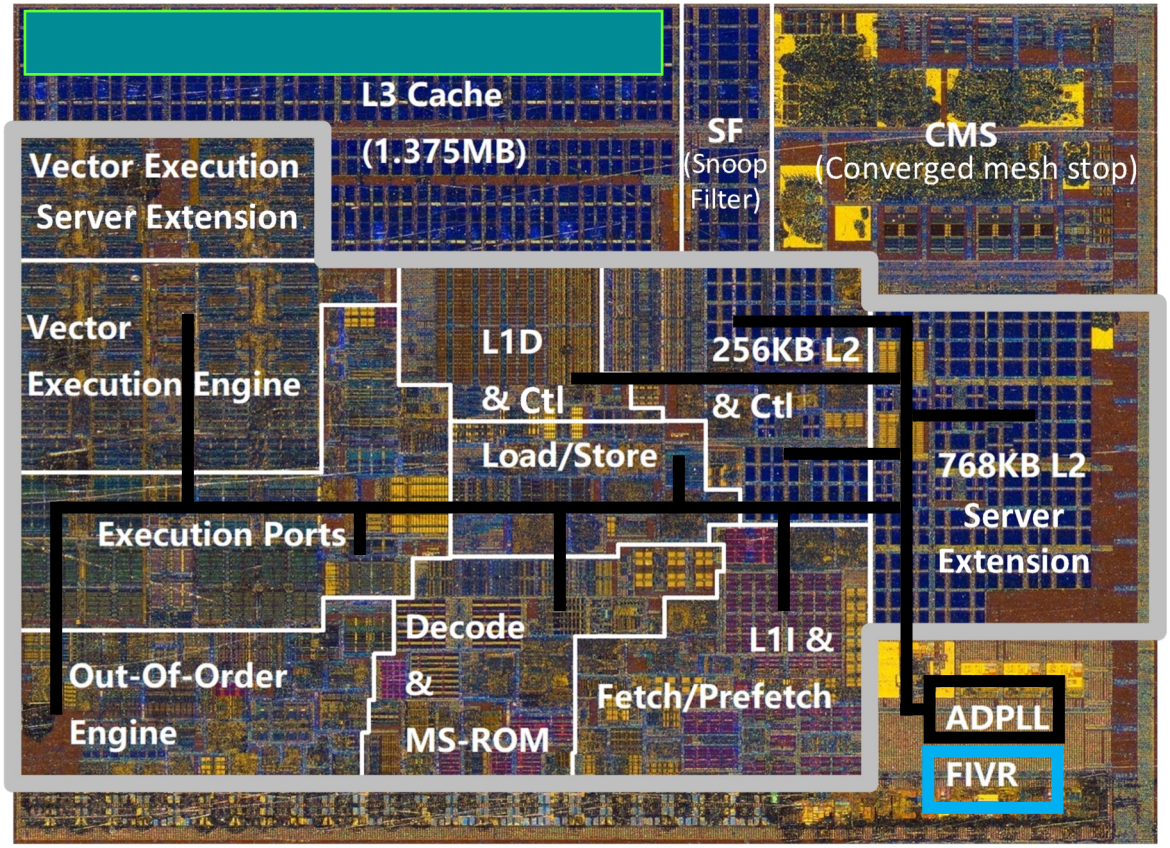
Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6	Running	On	Flushed	Nominal	S/R SRAM

C6 (Deep) Core C-state

Turn-off the clocks and PLL

Save/ Restore SRAM



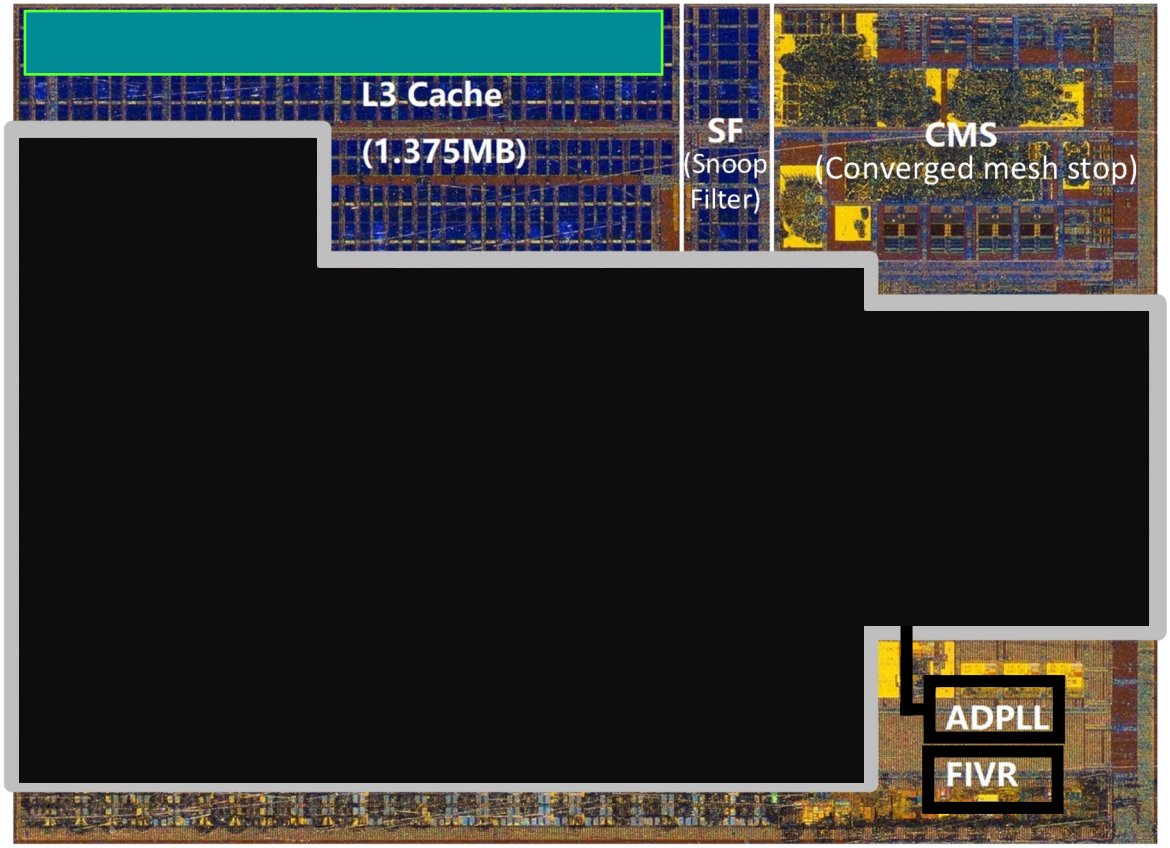
Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6	Stopped	off	Flushed	Nominal	S/R SRAM

C6 (Deep) Core C-state

Turn-off the voltage

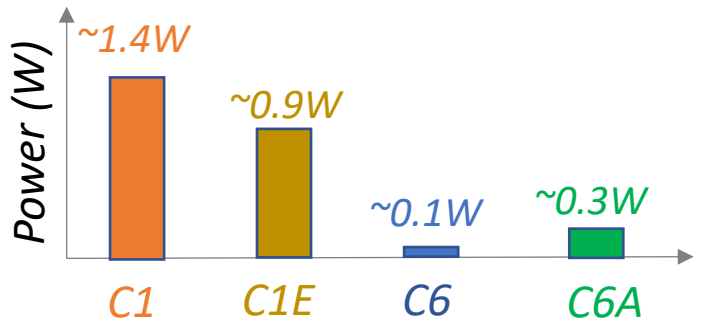
Save/ Restore SRAM

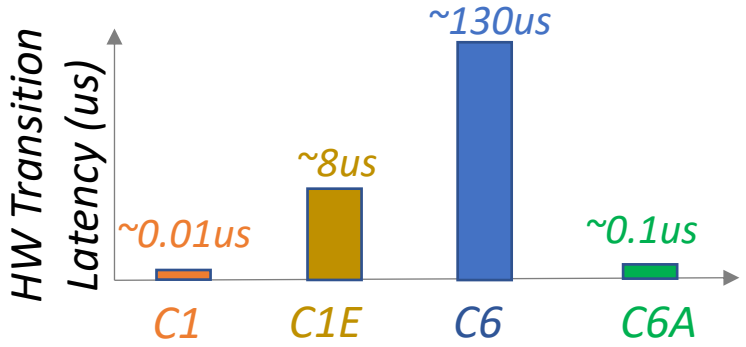


Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6	Stopped	off	Flushed	Shut-off	S/R SRAM

C-states Power/Latency Tradeoff





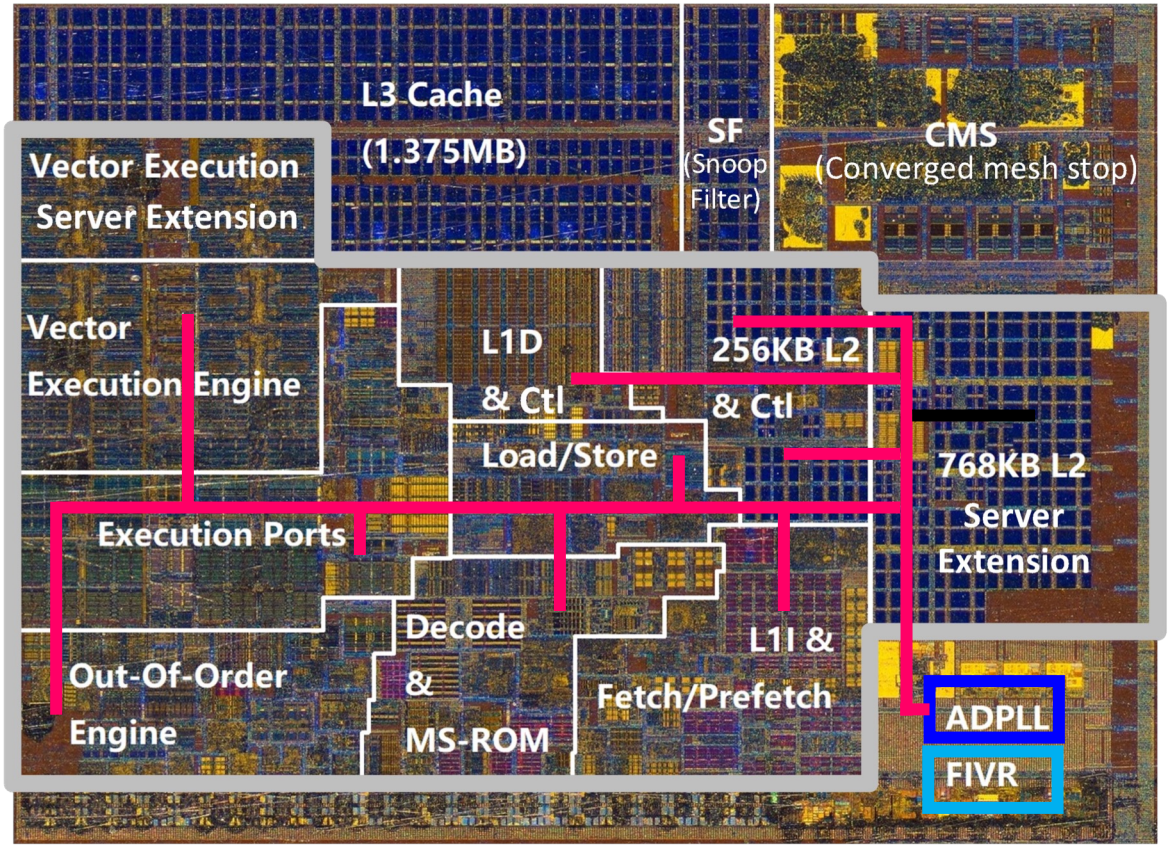
- C-states provide power and performance tradeoff
- The higher the C-state number
  - The lower its power consumption
  - But the longer the transition latency
- During a C-state transition a core cannot be utilized
  - Degrading the performance of latency-sensitive applications
- AgileWatts tackles this problem with C6A (C6 Agile) C-state
  - Deep idle C-state with a nanosecond-scale transition latency

AgileWatts

- We achieve AgileWatts goals with three key components:

1. Units Fast Power-Gating
2. Cache Coherence and Sleep Mode
3. C6A Power Management Flow

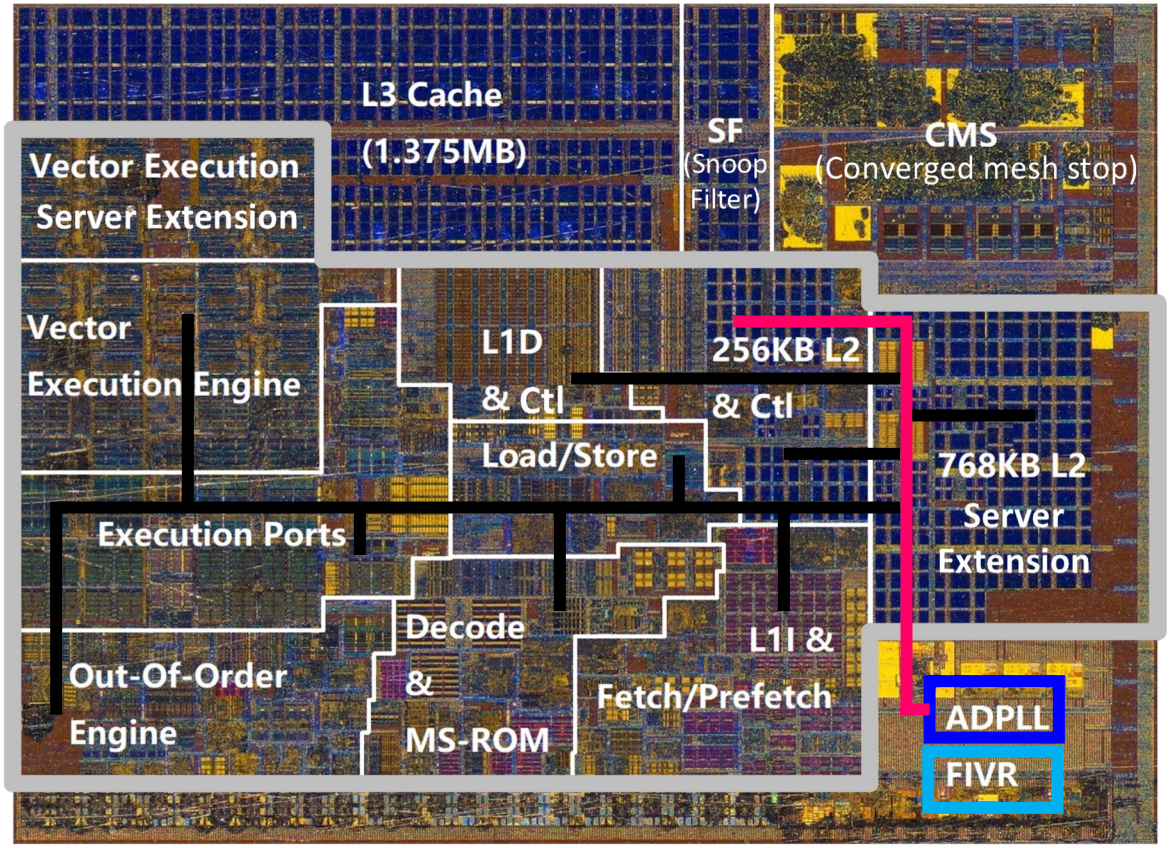
C6A (Deep & Agile) Core C-state



C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6A	Running	On	Coherent	Nominal	Maintained

C6A (Deep & Agile) Core C-state

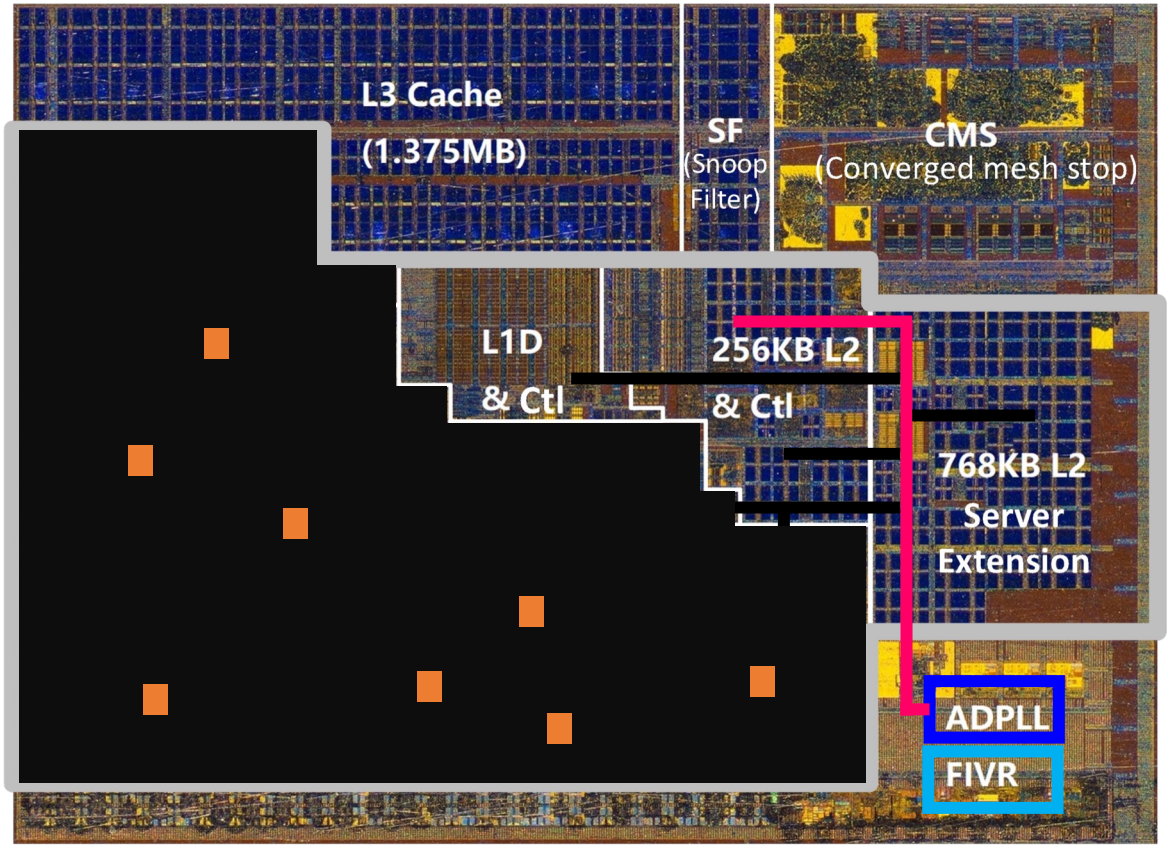
- **Clock-gate** most of the clock and keep the PLL On



C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6A	Most Stopped	On	Coherent	Nominal	Maintained

C6A (Deep & Agile) Core C-state

- **Power-gates** most of the core units
- Retaining the **context in place**

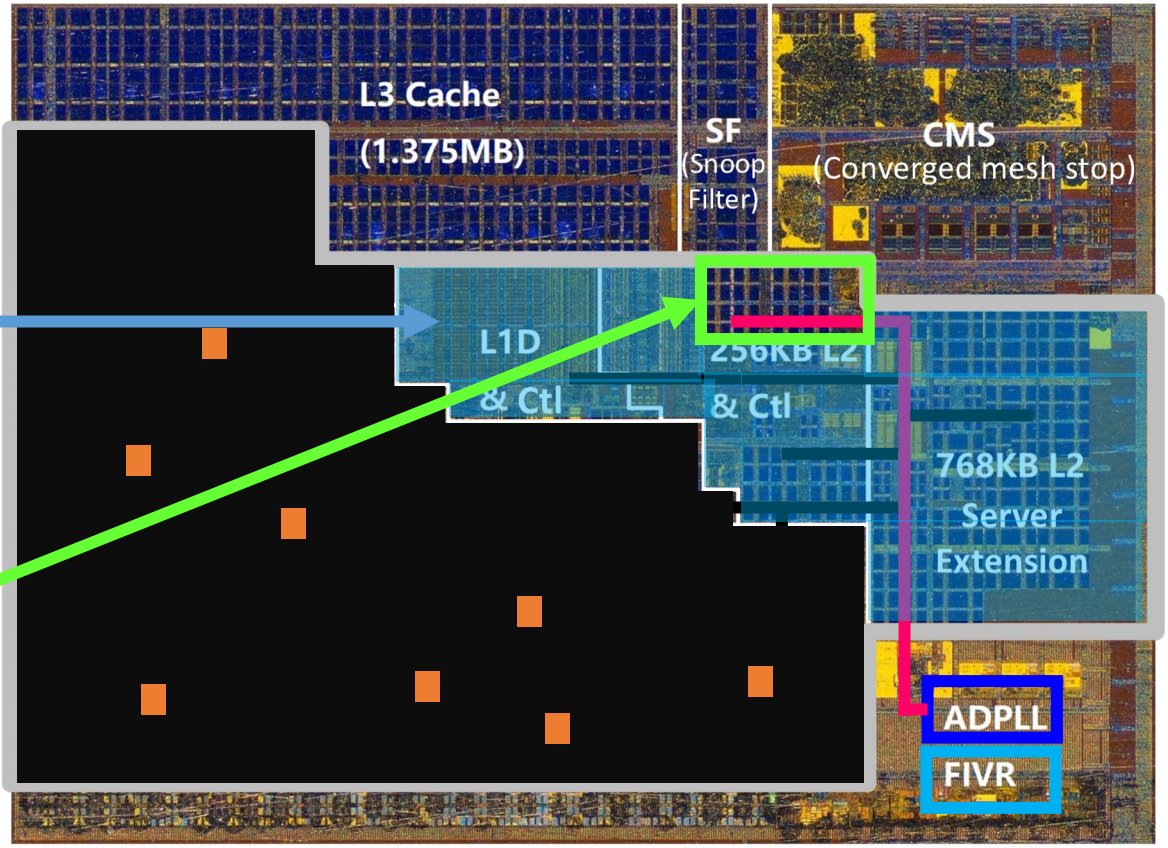


Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6A	Most Stopped	On	Coherent	Most PG	Maintained

C6A (Deep & Agile) Core C-state

- Reduce the caches supply voltage using **sleep transistors**
- Retain **a logic active** to wake-up the caches to respond to snoop requests



Voltage

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6A	Most Stopped	On	Coherent	PG/Ret/Nom	Maintained

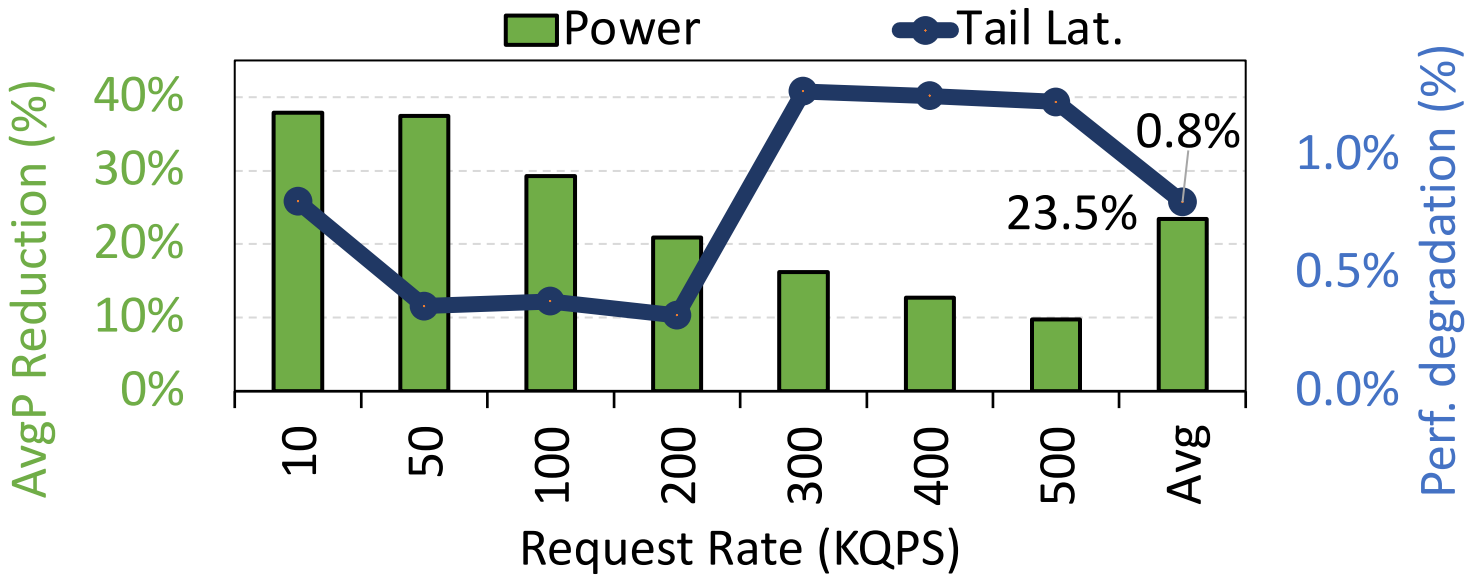
Evaluation Methodology

- **Power and Performance Model:** to evaluate AgileWatts, we use an accurate analytical power model, calibrated against an Intel Skylake server and considering the performance penalty

- **Power and C-state Residency Measurements:** We measure C-state residency and number of transitions using processor's residency reporting counters
  - We use the RAPL interface to measure power consumption

- **Workloads:** We evaluate AgileWatts using three latency-critical workloads: Memcached, Apache Kafka, and MySQL

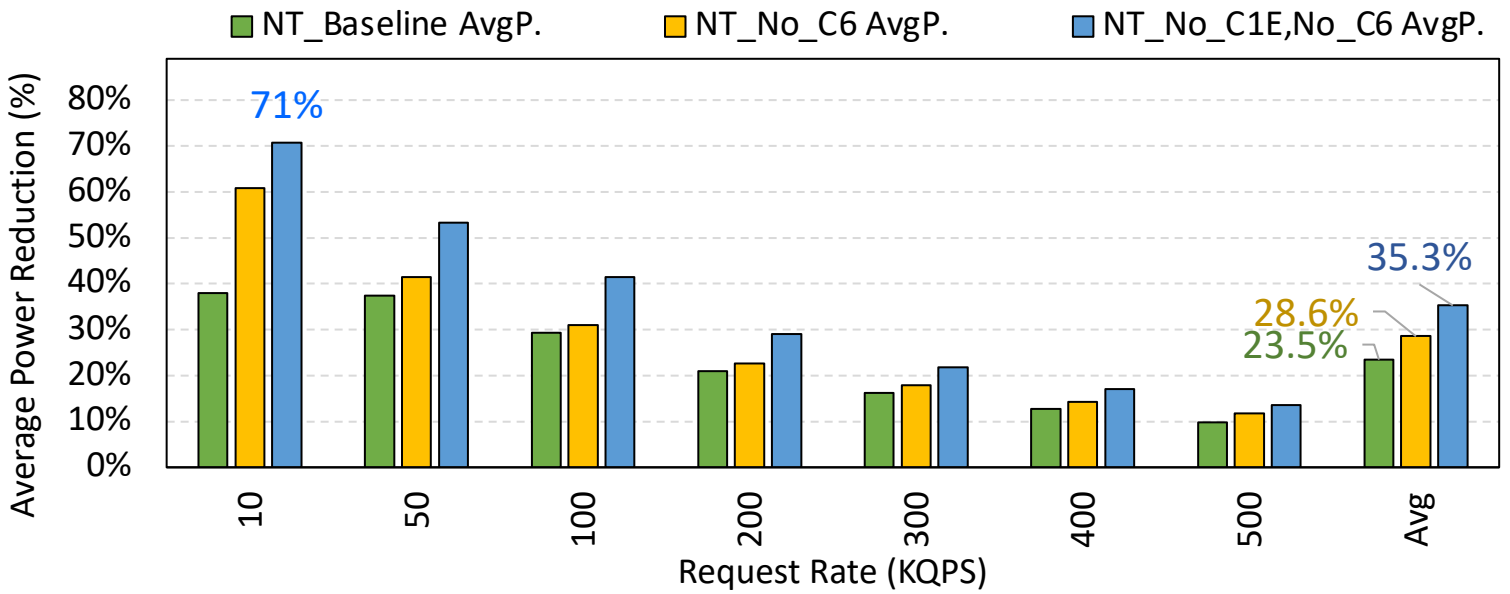
Power Savings and Overhead at Varying Load Levels



- AgileWatts reduces the average power consumption by up to 38% at low load
  - At high load, AgileWatts still provides 10% power savings

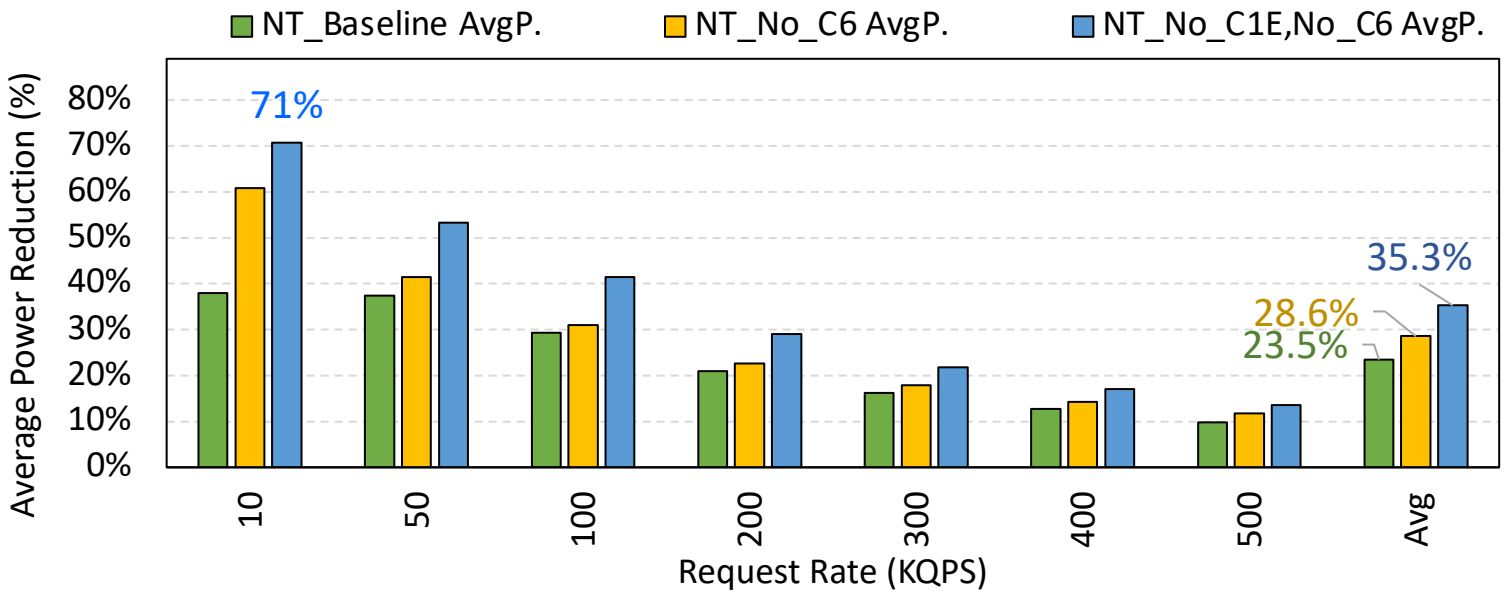
- AgileWatts has less than 1.3% impact on tail latency (server side)

Commonly used Configurations



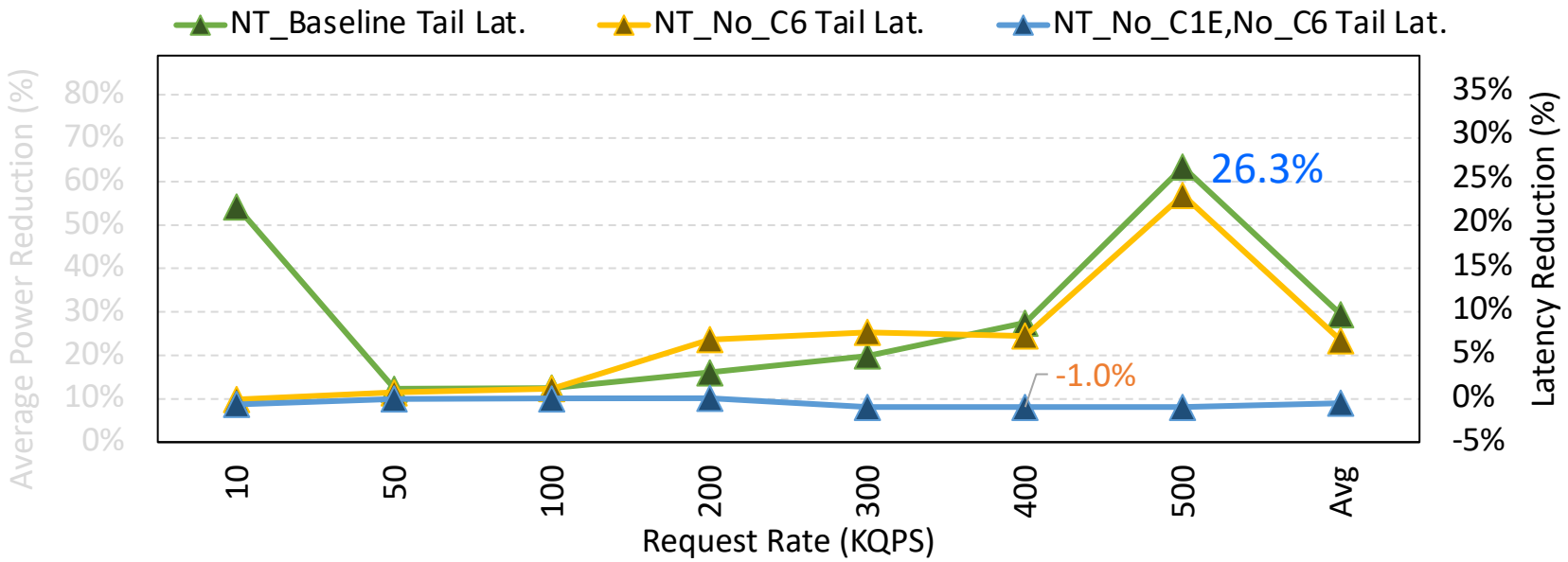
- Server vendors provide recommended system configurations, such as disabling certain C-states to increase system performance or disabling Turbo Boost to reduce power consumption
- We analyze three common configurations that *successively* disable Turbo, C6, and C1E in the baseline configuration (P-states disabled and Turbo and C-states enabled)
  1. Turbo disabled (NT_Baseline),
  2. Turbo and C6 disabled (NT_No_C6), and
  3. Turbo, C6 and C1E disabled (NT_No_C6,No_C1E)

Commonly used Configurations



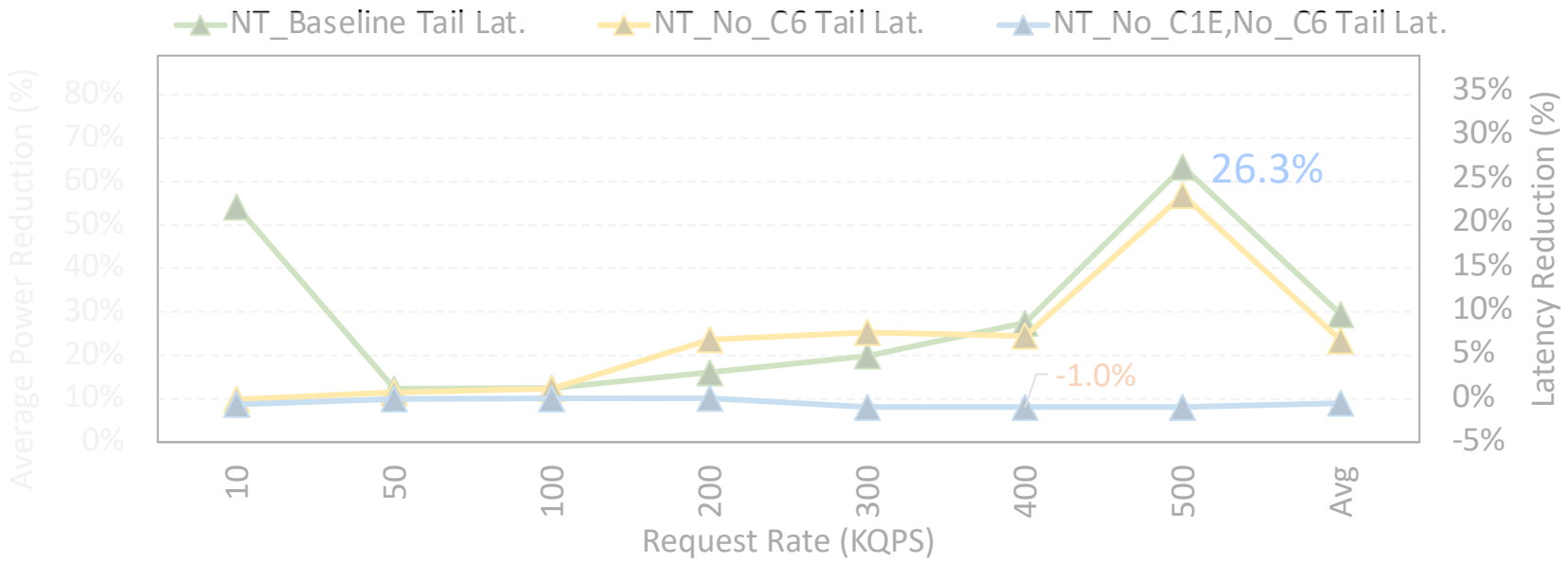
- AgileWatts reduces average power consumption (AvgP) up to 71% against all three tuned configurations
- The reason is that, in these workloads AgileWatts replaces the time that other configurations spend in C1 with C6A C-state, which has much lower power

Commonly used Configurations



- AgileWatts improves the performance by up to 26.3% compared to NT_Baseline and NT_No_C6 configurations
- AgileWatts only degrading the performance by up to 1% compared to NT_No_C6,No_C1E configuration

Commonly used Configurations



AgileWatts provides average and tail latencies comparable to the tuned configurations that disable some power management features, while reducing power significantly

Other Details in the Paper

- Implementation and hardware cost
- Idle power analysis
- C6A and C6AE latency
- Performance penalty
- Staggered unit wake-up
- Design complexity and effort
- Power savings and overhead at varying load levels
- Analysis of turbo boost performance improvement

Conclusion

- We introduced AgileWatts, a new deep C-state architecture that drastically reduces the transition latency, making deep C-states usable
- We showed that AgileWatts architecture can be achieved by leveraging agile power management techniques while retaining most of the power savings of existing deep idle states
- AgileWatts reduces the energy consumption of Memcached by up to 71% with up to 1% performance degradation
- AgileWatts is an effective approach to improving energy consumption of servers running latency-critical applications by enabling the server's processor to enter deep energy-saving states during short idle periods with negligible performance impact
- We hope our work paves the way for improving the energy proportional of datacenter servers

AgileWatts

An Energy-Efficient CPU Core Idle-State Architecture for Latency-Sensitive Server Applications

Jawad Haj-Yahya[4]

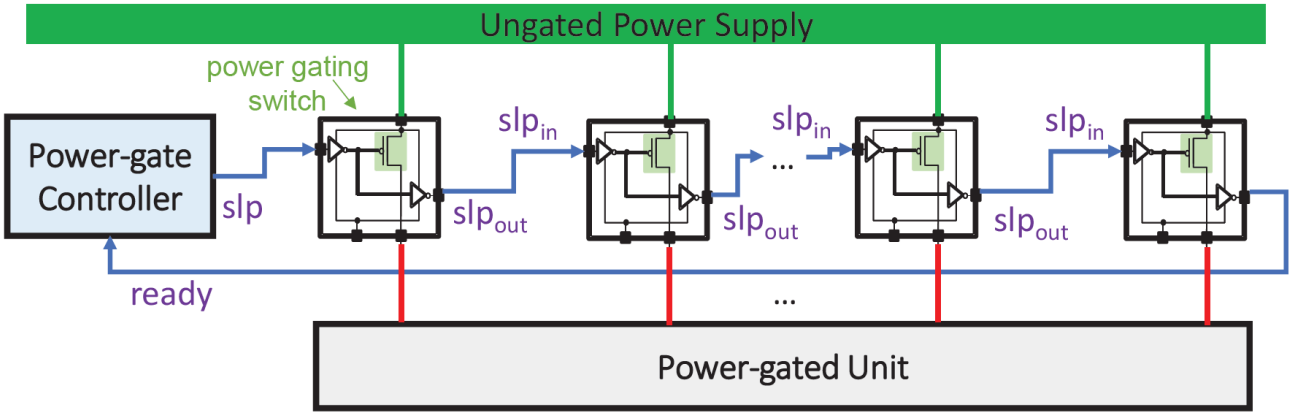
Haris Volos[2] *Davide B. Bartolini*[1] *Georgia Antoniou*[2]

Jeremie S. Kim[3] *Zhe Wang*[1] *Kleovoulos Kalaitzidis*[1] *Tom Rollet*[1]

Zhirui Chen[1] *Ye Geng*[1] *Onur Mutlu*[3] *Yiannakis Sazeides*[2]

1 HUAWEI 2 University of Cyprus 3 ETH zürich 4 RiVOS

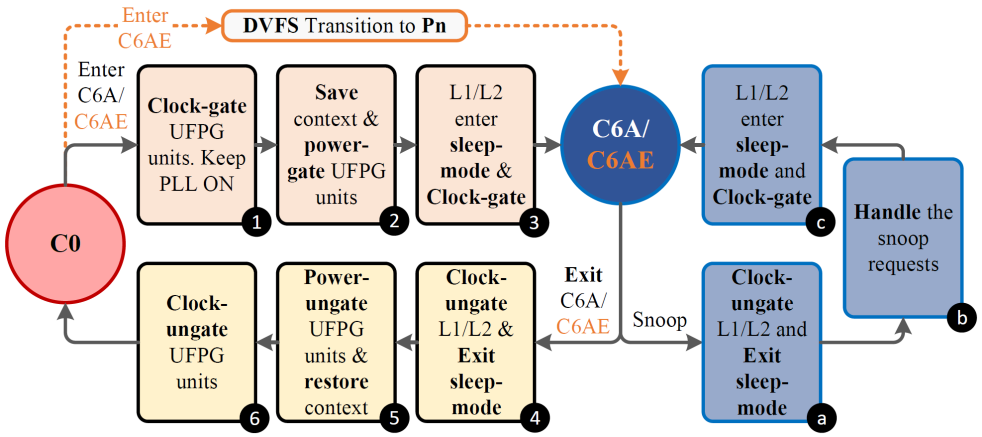
Power-gating



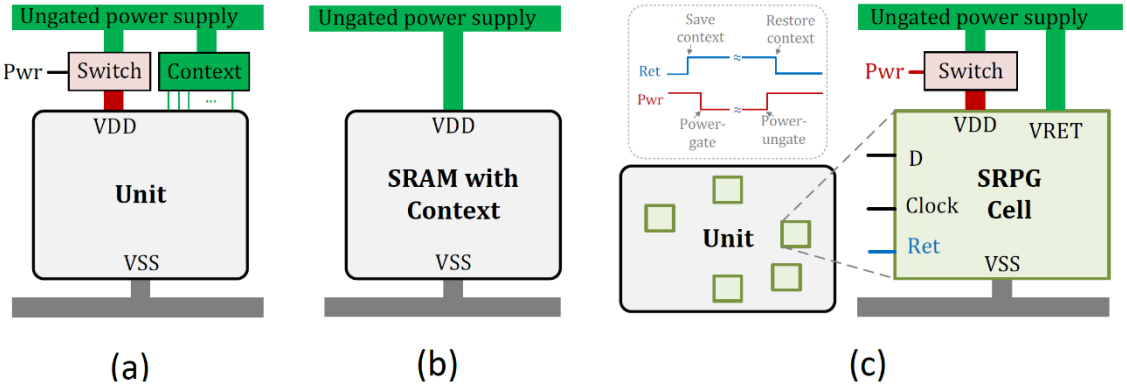
- Power-gating is a technique that is used to eliminate leakage of idle circuits
- Typically, the wake-up latency from a power-gated state requires a few to tens of cycles
- To reduce the worst-case peak in-rush current when waking up a power-gate:
  - A power-gate controller uses a staggered wake-up technique (shown in the figure)
  - It turns on different power-gate switch cells in stages to limit the current spike
  - To do so, the slp_{in} and slp_{out} of the switch cells are daisy-chained
  - The controller issues a signal to the first slp_{in}, and it receives an acknowledgement (*ready*) from the last slp_{out} , indicating that the power-gate is fully conducting
- Modern processors implement the staggering technique
  - The Intel Skylake core staggers the wake up of the AVX power-gates over 15ns

3. C6A Power Management Flow

- AgileWatts flow orchestrates the transitioning between the C0 and C6A C-states and handles coherence traffic while in C6A state
- The entry flow:
  - Clock-gates the gated-domain (UFPG) and keeps the core phase-locked loop (PLL) powered-on
  - When entering C6AE (C6A Enhanced), it initiates a non-blocking transition to Pn – the P-state with lowest frequency and voltage
  - Saves (in place) the gated-domain's context and shuts down its power
  - Sets the private caches into sleep mode and shuts down their clock
- When a snoop request arrives, the flow:
  - Clock-ungates the private cache domain and contextually adjusts its supply voltage to exit sleep mode
  - When all outstanding snoop requests are serviced, the flow rolls back the changes in reverse order and brings the core back into full C6A (or C6AE) state



Retain the Context in Place



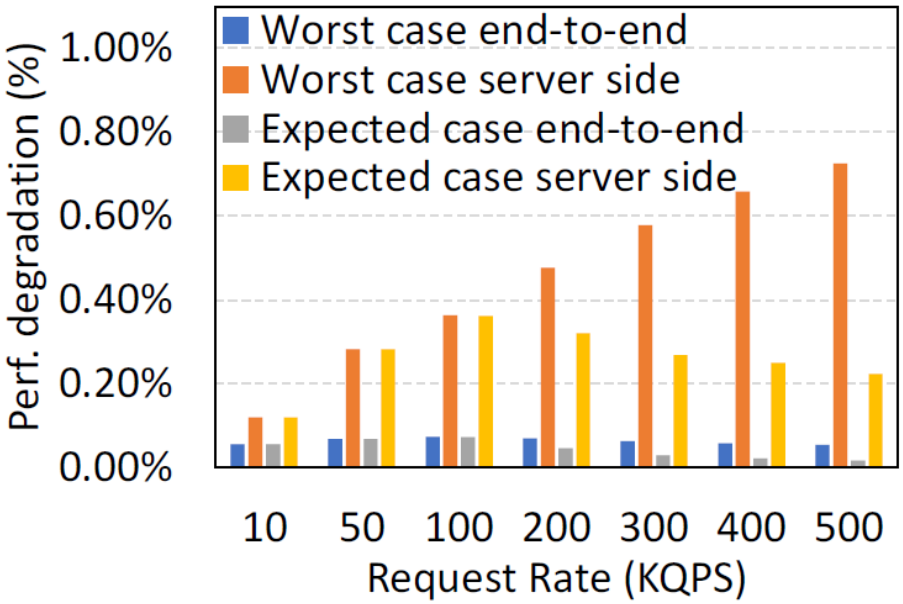
- AgileWatts leverages three techniques to efficiently retain the context during C6A

  a) Placing Unit Context in the Ungated Domain

  b) Place SRAM Context in Ungated Power Supply

  c) State Retention Power Gates (SRPGs)

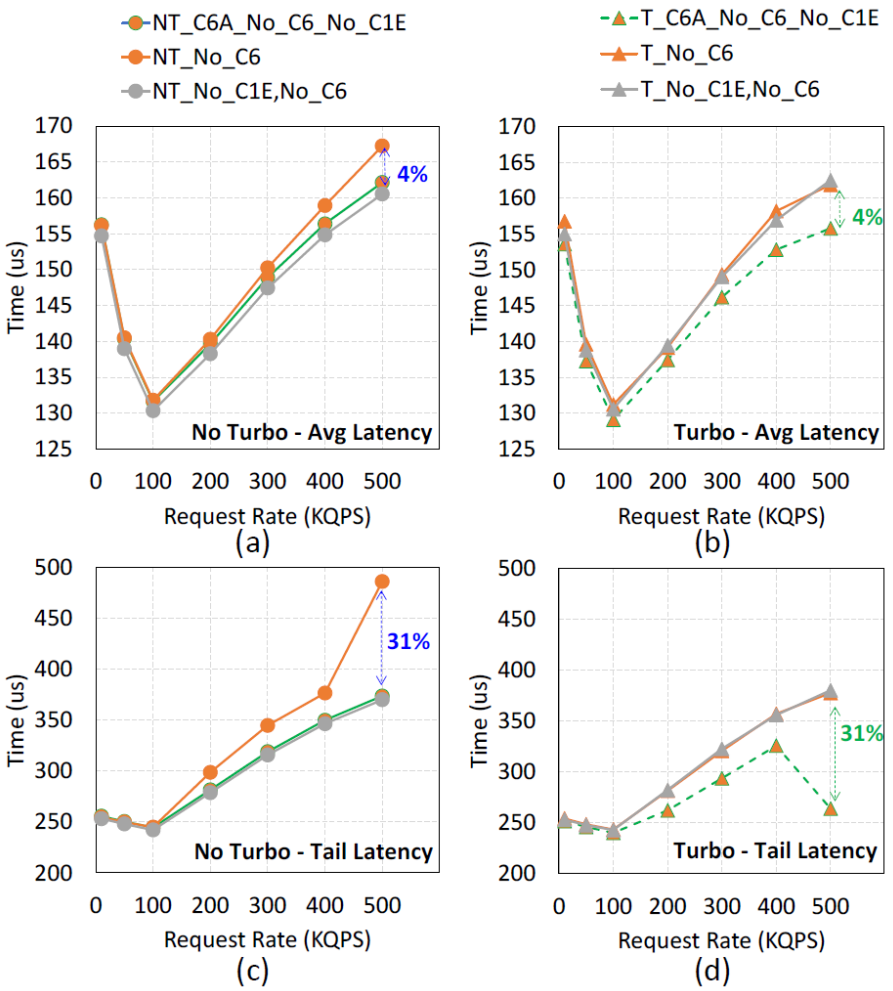
Power Savings and Overhead at Varying Load Levels



- Further analyzes of AgileWatts impact on average response time is shown
- We consider end-to-end (including network latency measured at 117us) and server-side response time for two cases:
  - The worst case, where we assume a C-state transition for each query and
  - the expected case, with the actual C-state transitions observed in the baseline
- As expected, the gap between the worst and the expected case is larger at high load,
  - since multiple queries are serviced within the same active period.
- We observe that the degradation of the end-to-end response time (i.e., by client) is negligible because the (non-changing) network latency dominates the overall response time

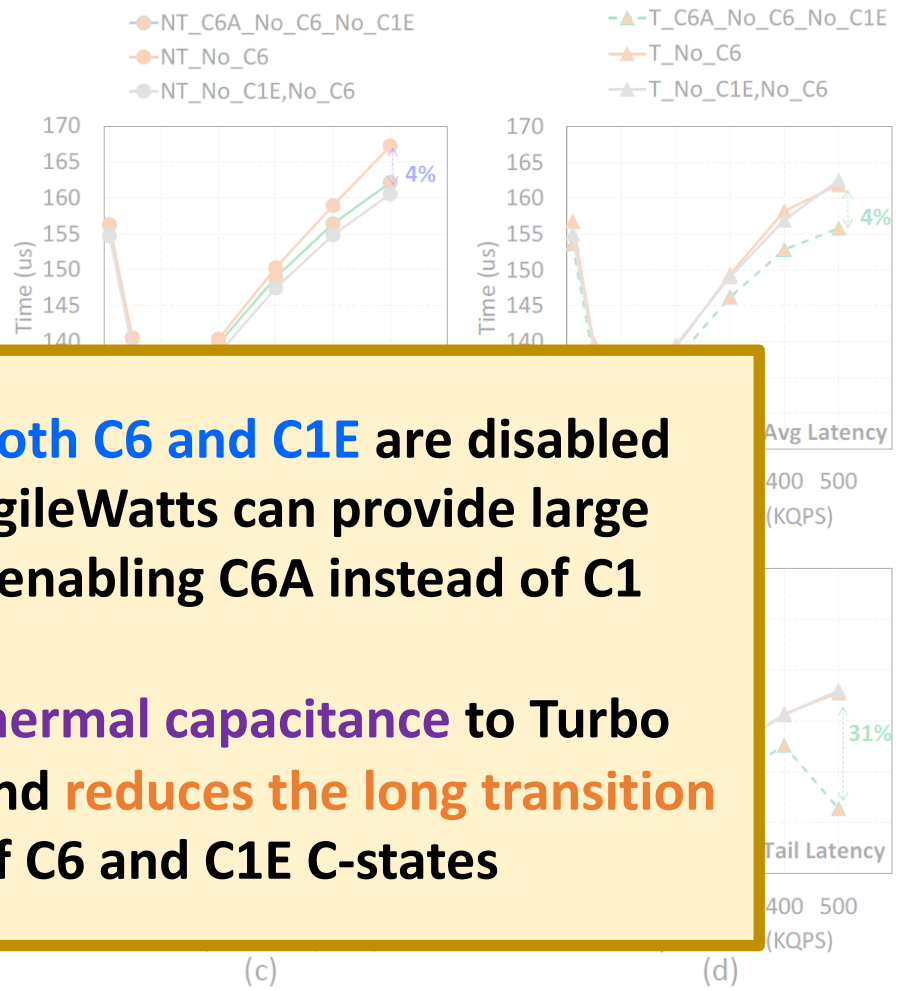
Analysis of Turbo Boost Performance Improvement

- Average and tail latency at different request rates (QPS) for four configurations that show the effect of idle power state on Turbo performance:
  - Turbo/No-Turbo & C6 disabled (T_No_C6/NT_No_C6),
  - Turbo/No-Turbo & C6 & C1E disabled (T_No_C6,No_C1E/NT_No_C6,No_C1E), compared
- With AgileWatts: Turbo/No-Turbo & C6A enabled & C6 & C1E disabled (T_C6A_No_C6_No_C1E/NT_C6A_No_C6_No_C1E)
- Figures (a,c) show that the configuration with Turbo and C6 disabled (i.e., NT_No_C6) increases the average/tail latency performance by up to 4%/31% over the same configuration with C1E disabled (i.e., NT_No_C6,No_C1E)
- Figures (c,d) show that enabling Turbo while disabling C1E (i.e., T_No_C6,No_C1E) does not improve performance over the same configuration with Turbo disabled (i.e., NT_No_C6,No_C1E)
- Figures (b,d) show that with Turbo enabled, only disabling C6 (i.e., T_No_C6) has the same performance as additionally disabling C1E (i.e., T_No_C6,No_C1E)
  - The reason is that in the T_No_C6 configuration, the transition overhead of C1E on average/tail latency offsets any thermal capacitance gains and ensuing performance gains from Turbo.



Analysis of Turbo Boost Performance Improvement

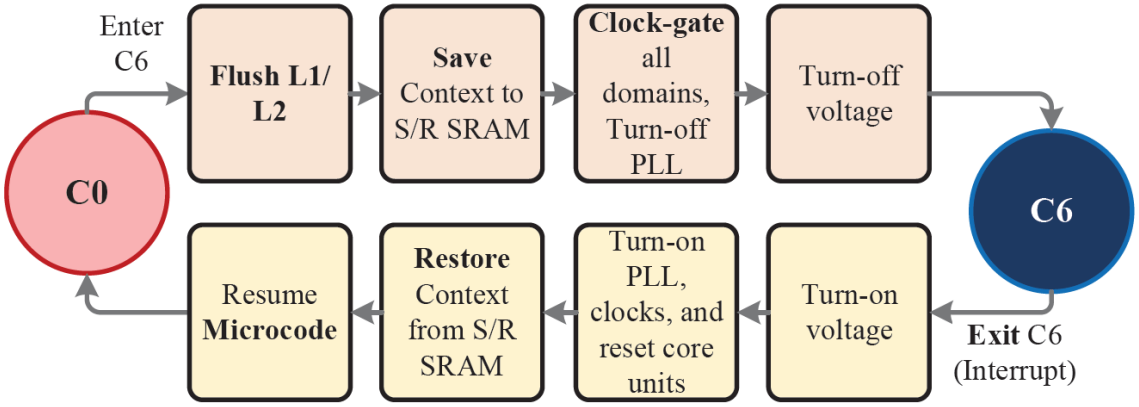
- Average and tail latency at different request rates (QPS) for four configurations that show the effect of idle power state on Turbo performance:
  - Turbo/No-Turbo & C6 disabled (T_No_C6/NT_No_C6),
  - Turbo/No-Turbo & C6 & C1E disabled (T_No_C6,No_C1E/NT_No_C6,No_C1E), compared
- With AgileWatts: Turbo/No-Turbo & C6A enabled & C6 & C1E disabled (T_C6A_N
- Figures (Turbo an the avera 4%/31% disabled
- Figures (disabling improve with Turb
- Figures (disabling performa T_No_C6,No_C1E)
  - The reason is that in the T_No_C6 configuration, the transition overhead of C1E on average/tail latency offsets any thermal capacitance gains and ensuing performance gains from Turbo.



In a configuration where both C6 and C1E are disabled while Turbo is enabled, AgileWatts can provide large performance benefits by enabling C6A instead of C1

Doing so provides larger thermal capacitance to Turbo compared to enabling C1E and reduces the long transition latency overhead of C6 and C1E C-states

Core C6 C-state Entry and Exit Flow



- C6 entry flushes privates caches, saves context, and turns off the PLL and voltage
  - Entry latency is dominated by the L1/L2 cache flush
- C6 exit is simply the reverse process of the entry flow
  - Exit latency is dominated by the hardware wake-up and state restoration

Conclusion

Problem: the irregular request patterns and tight latency requirements of user-facing applications render ineffective existing energy-conserving techniques when the server processor is idle due to the long transition time from a deep idle power state

Goal: directly address the root cause of the inefficiency, namely the high transition latency, to mitigate the energy inefficiency of modern datacenter server processors

Mechanism: AgileWatts, a new deep C-state architecture that drastically reduces the transition latency by leveraging three key power management techniques

- Uses medium-grained power-gates distributed across the core and maintains context in place
- Keeps private caches (i.e., L1/L2) and minimal control logic for cache coherence power-ungated
- Clock-gates the core components and clock distribution while keeping the clock generator on

Evaluation: we evaluate AgileWatts on the Intel Skylake server processor using variety of workloads, AgileWatts:

- Reduces the energy consumption of Memcached by up to 71% with up to 1% performance degradation
- Shows similar trends for other evaluated services such as MySQL and Kafka
- The new C-states, C6A is up to 900× faster than the deepest existing idle state C6

Conclusion: AgileWatts is an effective approach to improving energy consumption of servers running latency critical applications by enabling server's processor to enter deep energy saving states during short idle periods with negligible performance impact

Core Idle Power State – Core C-states

C-state	Transition time	Residency time	Power per core
C0 (P1)	N/A	N/A	~4W
C0 (Pn)	N/A	N/A	~1W
C1 (P1)	2µs	2µs	1.44W
C6A (P1)	2µs	2µs	~0.3W
C1E (Pn)	10µs	20µs	0.88W
C6AE (Pn)	10µs	20µs	~0.23W
C6	133µs	600µs	~0.1W

Table 1

C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C0	Running	On	Coherent	Active	Maintained
C1	Stopped	On	Coherent	Active	Maintained
C6A	Stopped	On	Coherent	PG/Ret/Active	In-place S/R
C1E	Stopped	On	Coherent	Min V/F	Maintained
C6AE	Stopped	On	Coherent	PG/Ret/Min V/F	In-place S/R
C6	Stopped	Off	Flushed	Shut-off	S/R SRAM

Table 2

- Core C-states are power saving states enable cores to reduce their power consumption during idle periods
- Modern processors support various C-states, for example, Intel's Skylake architecture offers the following four: C0, C1, C1E, C6
  - The higher the C-state number the lower its power consumption but the longer the transition time
- Table 2 describes the microarchitectural state for each core C-state and our new proposed idle states
  - C6A and C6AE (which replace C1 and C1E)
- While C-states reduce power consumption, during the entry-to and exit-from a C-state a core cannot be utilized
  - For example, it is estimated that C6 requires 133µs transition time
  - As a result, entry-exit latencies can degrade the performance of services that have microseconds processing latency, such as in user-facing applications

Motivation (II)

- Since the deepest C-state is C6 (Table 1), we estimate an upper bound of the average power (AvgP) savings for the ideal case of a deep idle state using equation 1 with:
  - The latency of C1 (2µs) and
  - The power of C6 (0.1W) per core
- R_{Ci} denotes the residency at power state Ci
  - i.e., the fraction of time a core spends in state Ci
- P_{Ci} denotes the average core power in state Ci
- Referring to our workload examples and using C-states power from Table 1:
  - Memcached at 20% load (i.e., R_{C0}= 20%, R_{C1}= 80%, R_{C6}= 0%) the potential power savings is **55%**

C-state	Transition time	Residency time	Power per core
C0 (*P1*)	N/A	N/A	~4W
C0 (*Pn*)	N/A	N/A	~1W
C1 (*P1*)	2µs	2µs	1.44W
C1E (*Pn*)	10µs	20µs	0.88W
C6	133µs	600µs	~0.1W

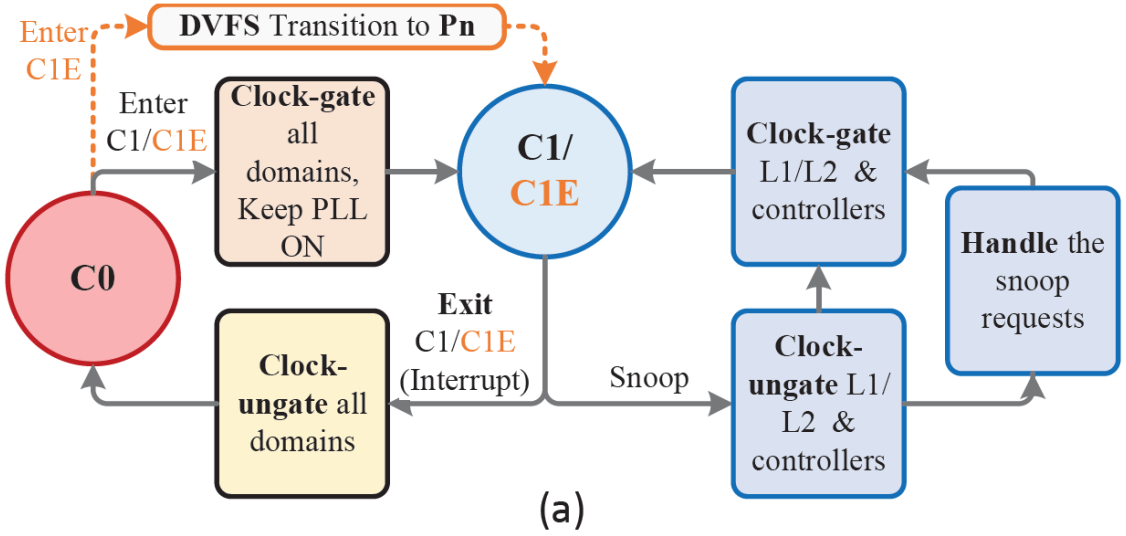
Table 1

$$AvgP_{baseline} = \sum_{i\in\{0,1,6\}} (R_{C_i} \times P_{C_i})$$
$$AvgP_{savings} = R_{C_1} \times (P_{C_1} - P_{C_6})$$
$$AvgP_{savings\%} = (AvgP_{savings} / AvgP_{baseline}) \times 100 \quad (1)$$

Equation 1

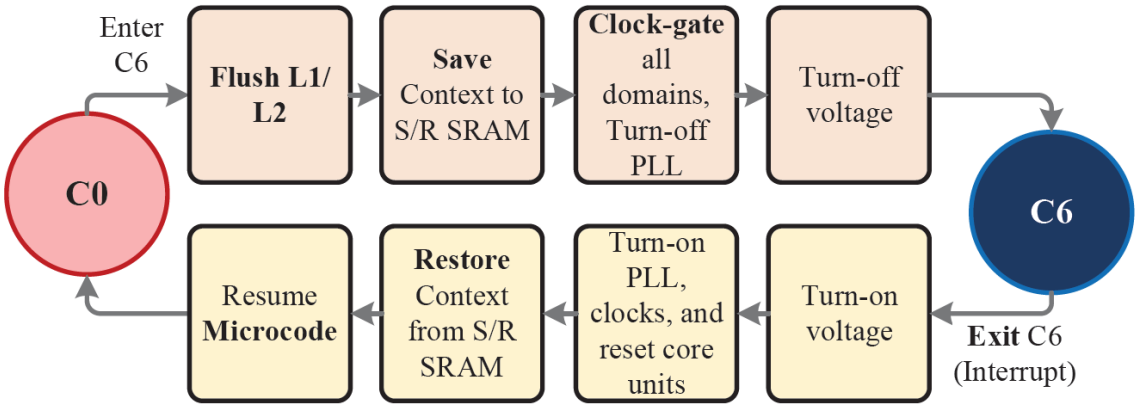
Core C1 C-state Entry and Exit Flow

- C1 entry mainly clock-gates all core domains and keeps PLLs ON
- C1E further reduces the voltage and frequency
- C1 exit Clock-ungates all domains
- During C1/C1E the core responses to snoop requests



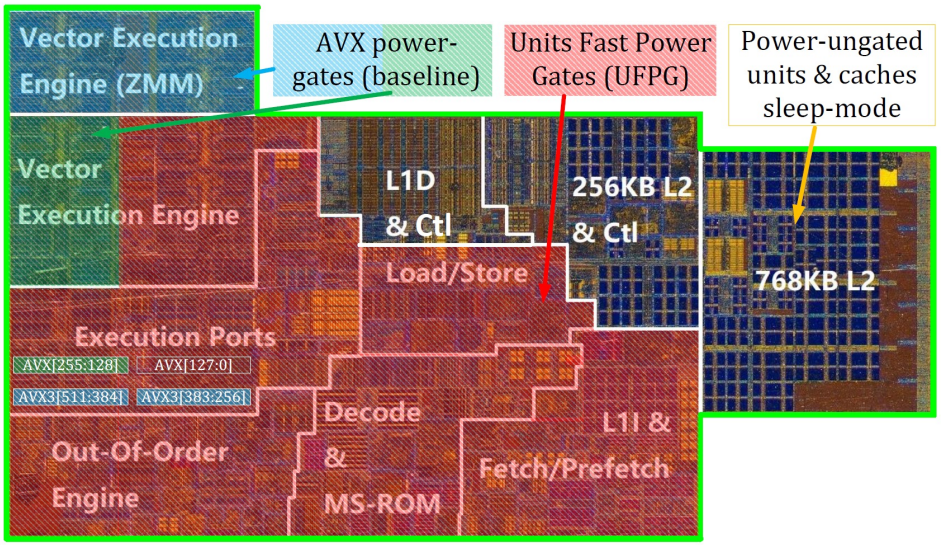
Core C6 C-state Entry and Exit Flow

- C6 entry flushes privates caches, saves context, and turns off the PLL and voltage
- C6 exit is simply the reverse process of the entry flow
- C6 entry latency is dominated by the L1/L2 cache flush
  - This flush time varies depending on 1) the number of dirty lines and 2) the core frequency when entering C6
  - Flushing a 50% dirty cache at 800MHz can take ~75µs
- C6 exit latency is significantly faster (~30µs)
  - ~10µs for hardware wake-up (power-ungating, PLL relock, reset, and fuse propagation)
  - State and microcode restoration take 20µs



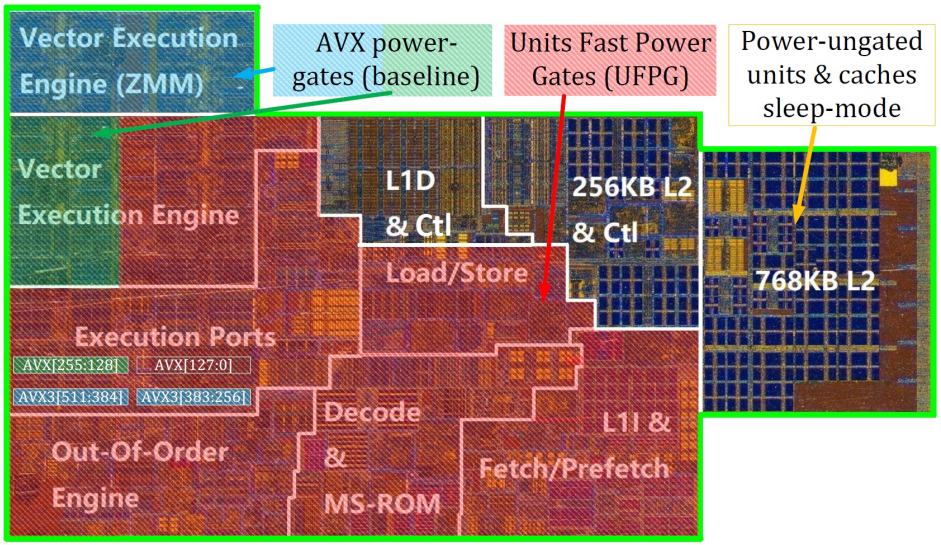
(b)

1. Units Fast Power-Gating



- Units Fast Power-Gating is a low-latency power-gating (PG) architecture that
  - Shuts off most of the core units while retaining the context in place
  - thus, enabling a transition latency of tens of nanoseconds
- Conventional context retention techniques sequentially save/restore the context to/from an external SRAM before/after power-gating/un-gating
  - This process adds several microseconds to the entry/exit latency
  - AgileWatts retains the context in place, completely removing save/restore latency overhead at a very small additional idle power cost

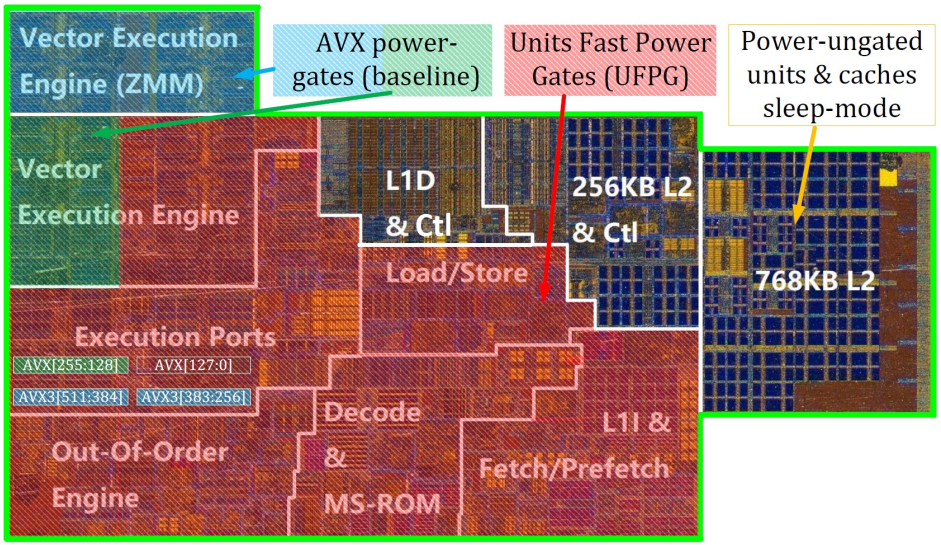
1. Units Fast Power-Gating



- Units Fast Power-Gating is a low-latency power-gating (PG) architecture that
  - Shuts off most of the core units while retaining the context in place
  - thus, enabling a transition latency of tens of nanoseconds

- AgileWatts retains the CPU context in place, completely removing save/restore latency overhead at a very small additional idle power cost

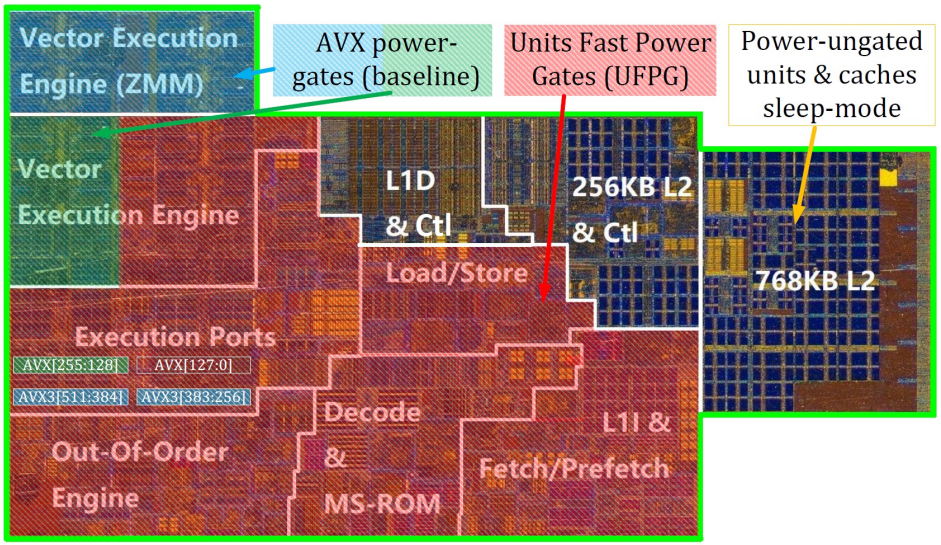
2. Caches Coherency and Sleep Mode (I)



- To avoid the high latency to flush private caches to power-gate them, AgileWatts keeps them power-ungated when transitioning to C6A

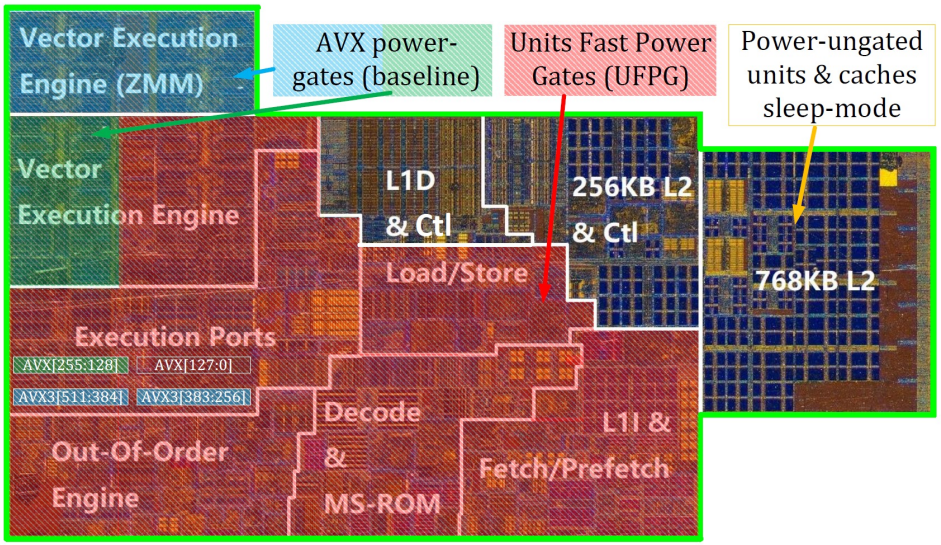
- This has two design implications:
  - AgileWatts needs to employ other power-saving techniques to reduce the power of the caches
  - A core in C6A state still needs to serve coherence requests (i.e., snoops)

2. Cache Coherence and Sleep Mode (II)



- AgileWatts employs two key techniques to reduce the power consumption of the power-ungated private cache domain:
  1. Unless a coherency request is being served, AgileWatts keeps this domain clock-gated to save its dynamic power
  2. AgileWatts leverages the cache sleep-mode technique, which adds sleep transistors to the SRAM array of private caches.
     - These sleep transistors reduce the SRAM array's supply voltage to the lowest level that can safely retain the SRAM content while significantly reducing leakage power

2. Caches Coherency and Sleep Mode (III)

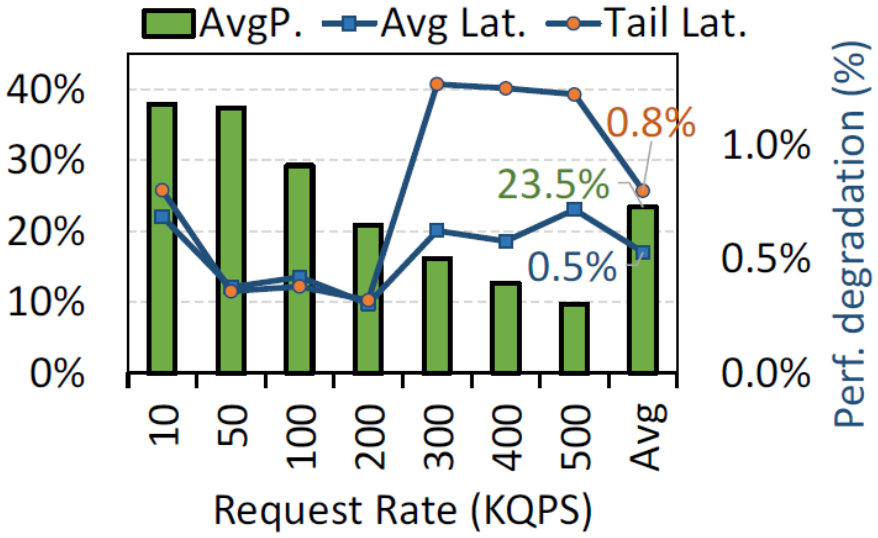


- Since private caches are not flushed when a core enters C6A, AgileWatts must allow the core to respond to snoop requests

- AgileWatts keeps the logic required to handle cache snoops in the power-ungated (but clock-gated) domain together with the private caches

- As soon as this logic detects incoming snoop traffic
  - it temporarily increases the SRAM array voltage through the sleep transistors and reactivates the clock of the private caches for the time required to respond to snoop requests

Power Savings and Overhead at Varying Load Levels



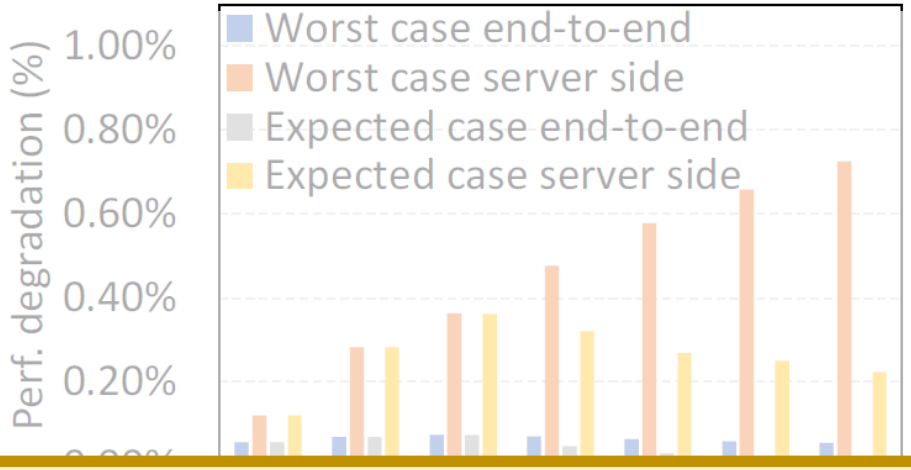
- AgileWatts reduces the average power consumption by up to 38% at low load, with less than 1% impact on both average and tail latency

- At high load, AgileWatts still provides 10% power savings, with less than 1.3% impact on tail latency

Other Details in the Paper

- Implementation and hardware cost
- Idle Power Analysis
- C6A and C6AE Latency
- Performance Penalty
- Staggered Unit Wake-up
- Design Complexity and Effort
- Power Savings and Overhead at Varying Load Levels
- Analysis of Turbo Boost Performance Improvement

Power Savings and Overhead at Varying Load Levels



AgileWatts significantly improves core average power consumption of the Memcached service across load levels with minimal performance overhead over the baseline

- Fur
- We ide
  res
  -
  - the expected case, with the actual C-state transitions observed in the baseline
- As expected, the gap between the worst and the expected case is larger at high load,
  - since multiple queries are serviced within the same active period.
- We observe that the degradation of the end-to-end response time (i.e., by client) is negligible because the (non-changing) network latency dominates the overall response time