Simulació processador Superescalar: Configuració, Temps i Utilització

Arquitectura de Computadors

Guillermo Pinteño i Vladyslav Lysyy

Índex

Característiques dels cores a estudiar

Es demana estudiar un dels cores de cada un dels següents processadors:

- Intel Core Ultra 9 285K (Arrow Lake core, Lion Cove + Skymont)

- AMD Ryzen 9 9900X (Granite Ridge, Zen 5 core)

Per aconseguir això cal recopilar una sèrie de dades que després s'utilitzaran per fer la simulació amb Simplescalar.

Lion Cove (Intel Core Ultra 9 285K)

Del processador Intel s'ha escollit estudiar un dels core Lion Cove. A continuació les característiques que s'han aconseguit recopilar

1. K-via

En el Lion Cove la quantitat d'instruccions per cicle que pot tractar cada etapa es de **8 instruccion**s [1]. Per tant la k-via del Lion Cove queda com:

Etapa	Nombre d'instruccions
Fetch	8
Decode	8
Issue	8
Commit	8

1. Instruccions que pot tractar cada etapa del Lion Cove.

2. Finestra d'instruccions i cua d'accés

Segons el blog HWCooling [4] i el blog ChipsAndCheese [1] el Lion Cove té un **ROB** (similar a **RUU**) de **576 instruccions**. Per tant utilitzarem aquest valor per RUU del Simplescalar.

Respecte la cua d'accés el Lion Cove té aproximadament 189 entrades per Load i 120 entrades per Store [1]. Com que Simplescalar només permet donar un únic valor de Load/Store Queue (LSQ) s'assumirà la suma d'aquests dos. Per tant, el valor **LSQ** del Lion Cove per aquesta simulació es de **309 entrades**.

3. Cachés

A continuació la configuració de cachés del Lion Cove:

Caches Lion Cove				
Type	Size (KB)	Ways	Sets	Block size (Bytes)
L1 Instructions	64	16	64	64

L0 Data	48	12	64	64
L1 Data	195	12	260	64
Unified L2	2560	10	4096	64

2. Característiques sobre les cachés del Lion Cove (size, ways) [1], (block size) [2]. El nombre de sets és calculat a partir de les altres dades (sets = size (B) / (ways * block_size (B)).

Com es pot observar a la taula 2 el Lion Cove compta amb un "nivell extra" de caché de 195 KB entre la L1 (anomenada per Intel L0) i la L2. Això suposa un problema amb la simulació ja que Simplescalar només suporta dos nivells de cachés. Es per això que s'assumirà un mateix nivell per la caché L0 i L1, per això simplement es farà la suma de la seva capacitat total, mentre que la associativitat es mantindrà igual ja que aquesta es la mateixa per les dues cachés.

Per tant, la taula quedarà de la següent forma:

Caches Lion Cove (modified to Simplescalar use)				
Type	Size (KB)	Ways	Sets	Block size (Bytes)
L1 Instructions	64	16	64	64
L1 Data	243	12	324	64
Unified L2	2560	10	4096	64

3. Característiques de les cachés del Lion Cove amb unificació de la L0 i L1 per la simulació amb Simplescalar.

Respecte l'algorisme de reemplaçament de les cachés del Lion Cove no s'ha aconseguit trobar informació directa. Encara això segons el següent estudi de la Universitat de Saarland [3], Intel opta per utilitzar l'**algorisme LRU** per les seves cachés. Per tant, aquest serà l'algorisme que s'utilitzarà a la simulació de SimpleScalar.

4. Ample de banda i latència de la memòria principal

El Lion Cove té un ample de banda de la DRAM (Memòria principal) de 94,87 GB/s [1]. Aquest valor no és suficient pel simulador Simplescalar, ja que aquest necessita l'ample de banda DRAM expressat en Bytes/cicle. Per tant, sabent que la freqüència base del Lion Cove és de 3,7GHz segons les especificacions de Intel [5], es pot calcular l'ample de banda en Bytes/cicle.

$$ample\ de\ banda\ DRAM\ = \frac{94{,}87 \cdot 10^9\ \frac{B}{s}}{3{,}7 \cdot 10^9\ \frac{1}{s}} = 25{,}64\ \frac{B}{cicle}$$

(Això suposant que Intel especifica GB com 10^9 Bytes, d'igual forma en potències de 2 el resultat quasi no varia)

Segons aquest càlcul el valor d'**ample de banda de DRAM** utilitzat per simular el Lion Cove serà de 25,64 ~ **26 Bytes/cicle**.

D'altra banda, el core compta amb una latència promig de 134,4 ns de DRAM [1]. El simulador Simplescalar necessita la latència mesurada en cicles de rellotge. La freqüència base del Lion Cove és de 3,7GHz, així que es pot calcular la latència en unitats de cicles.

$$latencia = 134 \cdot 10^{-9} s \cdot 3{,}7 \cdot 10^{9}\, s^{-1} = 495{,}8\, cicles$$

Per tant, segons l'aproximació feta el Lion Cove té una **latència amb la DRAM** de 495,8 ~ **496 cicles**.

Per calcular la latència dels següents accessos coneixem que les DRAM DDR5 aconsegueixen carregar els següents blocs de dades de tal forma que després del primer accés el traspàs d'informació només té un cost d'un cicle de DRAM. Coneixent la freqüència del rellotge de la memòria i la freqüència del rellotge del processador, podem calcular quants cicles "ha d'esperar el processador" per obtenir el següent bloc de dades.

El càlcul és el següent:

$$1\ cicle\ RAM \cdot \frac{\frac{1}{4800 \cdot 10^{6}}\, s}{1\ cicle\ RAM} \cdot \frac{1\ cicle\ CPU}{\frac{1}{3{,}7 \cdot 10^{9}}\, s} = 0{,}77083333\ cicles\ CPU$$

Es pot observar que un cicle de RAM correspon a 0.77 cicles de la CPU. Això vol dir que en un cicle de CPU la DRAM no haurà completat un cicle intern, per tant la CPU haurà d'esperar un altre cicle més. Per tant, els **cicles necessaris pels següents accessos seran 2**.

5. Integer ALUs, FP ALUs i nombre de ports d'accés a memoria en 1er nivell de caché

A continuació el nombre d'unitats funcionals (ALUs d'enters i ALUs de coma flotant):

Numeric Type	Simple arithmetic	Multiplication
Integer	3	3
Floating Point	2	2

4. Aquesta informació es troba al esquema proporcionat a la "review" del Lion Cove de ChipsAndCheese [1].

Respecte el nombre de ports d'accés a la memòria de primer nivell, només s'ha trobat informació respecte la caché DL1, però no de la Il1. Aquestes són les característiques de la DL1:

Cache	Read	Write
DL1	3	2

5. Aquesta informació es pot visualitzar al esquema proporcionat a la "review" de LionCove de ChipsAndCheese [1].

Zen 5 core (AMD Ryzen 9 9900X)

Del procesador AMD Ryzen 9 9900X cal estudiar un dels seus cores Zen 5 core. A continuació els paràmetres que s'han aconseguit recopilar:

1. K-via

El Zen 5 core, a diferència del Lion Cove té la peculiaritat de que permet SMT (Simultaneous multithreading [6]. Es per això que té un fetch i un decode amb dos busos (cadascun de 4 instruccions). Com que Simplescalar no permet la simulació de multithreading, s'interpretarà que l'etapa de fetch i decode només compten amb un bus de 4 instruccions cadascún. D'altra banda el issue i el commit si compten amb un bus de 8 instruccions cadascun.

Per tant la k-via del Zen 5 queda com:

Etapa	Nombre d'instruccions
Fetch	4
Decode	4
Issue	8
Commit	8

6. Instruccions que es pot tractar a cada etapa del Zen 5 Core.

2. Finestra d'instruccions i cua d'accés

Segons la review de ChipsAndCheese del Z 5 Core [6], aquest compta amb 448 entrades del ROB pel cas d'un únic fil d'execució al core. Aquest valor es reparteix entre els fils d'execució a major és el nombre d'aquests, ja que el Z 5 Core permet SMT (Simultaneos multithreading). Com que Simplescalar només permet un únic fil d'execució per simulació. Es per tant que el valor que s'agafarà pel **RUU** de Simplescalar serà de **448 entrades**.

La cua d'accés el Zen 5 Core té 104 entrades d'Store [8]. D'altra banda la cua de Load té 202 entrades [6]. Per tant s'assumirà per la simulació de Simplescalar una **LSQ** de **306 entrades** (202+104).

3. Cachés

A continuació la configuració de cachés del Z 5 core:

Caches Z 5 core				
Type	Size (KB)	Ways	Sets	Block size (Bytes)
L1 Instructions	32	8	64	64
L1 Data	48	12	64	64
Unified L2	1024	16	1024	64

7. Característiques sobre les cachés del Z 5 core (size, ways) [6], (block size) [7]. El nombre de sets és calculat a partir de les altres dades (sets = size (B) / (ways * block_size (B)).

A diferència del Lion Cove, el Z 5 Core no té una caché intermitja entre la L1 i la L2, per tant es pot fer la simulació exacta de la configuració de cachés original del Z 5 Core.

Respecte el algorisme de reemplaçament de les cachés del Z 5 Core, podem suposar que utilitza un PLRU (Pseudo LRU) [10]. Però per la simulació de Simplescalar s'utilitzarà l'**algorisme LRU.**

4. Ample de banda i latència de la memòria principal

Segons una diapositiva de AMD a la conferència Hot Chips [9], l'**ample de banda** d'accés a la **DRAM** és de **32 Bytes/cicle** segons. Per tant, aquest és el valor que s'utilitzarà a la simulació amb Simplescalar.

No s'ha trobat el valor exacte de latència d'accés a DRAM, però es sap que la latència de L3 miss és de 79,08 ns [6]. Sabent que la latència de DRAM és la de un L3 miss + overhead del controlador de memòria i bus (i aquest es mímim), s'agafarà aquest valor del L3 miss per aproximar la latència DRAM.

Per tant, per la simulació es proposa una latència de DRAM de 79,08 ns. Però Simplescalar necessita el valor de latència en cicles. Per tant, prenent una freqüència base de rellotge de 4,4 GHz. La latència de la DRAM es calcula de la següent forma:

$$latencia = 79{,}08 \cdot 10^{-9} s \cdot 4{,}4 \cdot 10^{9} \frac{1}{s} = 347{,}95\ cicles$$

Per tant, el valor de **latència de primer accés a la DRAM** per la simulació Simplescalar serà de 347,95 ~ **348 cicles**.

D'altra banda, com el mida de línia de la L3 és de 64 Bytes [7], mentre que el bus entre L3 y DRAM és de 32 Bytes [6], calen dos accessos a memòria per obtenir el valor de la DRAM. El primer accés té una latència de 348 cicles com s'ha mencionat abans, però al segon accés la DRAM ja té preparats els següents 32 Bytes. Es per això que la latència a partir del segon accés es molt menor a la del primer.

A l'igual que amb el Lion Cove, per aconseguir la latència dels següents accesos al primer cal calcular-ho sabent la freqüència de rellotge de la DRAM DDR5 i el rellotge del processador.

El càlcul és el següent:

$$1\ cicle\ RAM \cdot \frac{\frac{1}{4800 \cdot 10^6}s}{1\ cicle\ RAM} \cdot \frac{1\ cicle\ CPU}{\frac{1}{4{,}4 \cdot 10^9}s} = 0{,}91\ cicles\ CPU$$

Es pot observar que un cicle de RAM correspon a 0,91 cicles de la CPU. Això vol dir que en un cicle de CPU la DRAM no haurà completat un cicle intern, per tant la CPU haurà d'esperar un altre cicle més. Per tant, els **cicles necessaris pels següents accessos seran 2**.

5. Integer ALUs, FP ALUs i nombre de ports d'accés a memoria en 1er nivell de caché

A continuació el nombre d'unitats funcionals (ALUs d'enters i ALUs de coma flotant):

Numeric Type	Simple arithmetic	Multiplication
Integer	3	3
Floating Point	2	2

8. Nombre d'ALUs del Lion Cove. Aquesta informació es troba al esquema proporcionat a la "review" del Zen 5 Core de ChipsAndCheese [6].

Respecte el nombre de ports d'accés a la memòria de primer nivell, només s'ha trobat informació respecte la caché DL1, però no de la Il1. Aquestes són les característiques de la DL1:

Cache	Read	Write
DL1	4	2
Il1	(Proablement 2)	?

9. Nombre de ports de la memòria de primer nivell. Es pot deduir que la caché Il1 té 2 ports d'accés per la lectura ja que el Zen 5 Core té 2 busos en paral·lel per l'etapa de fetch, ja que permet el SMT (encara així aquesta és una deducció molt especulativa).

Mesures de rendiment alternatives

A més de les característiques concretes dels processadors, podem comparar aquests a través de les proves de rendiment que s'han fet amb certs benchmarks:

Benchmark	Lion Cove (Intel Core Ultra 9 285 K)	Zen 5 (AMD Ryzen 9 9900X)
passmark [11][12]	5095 (single thread) 67643 (multithread)	4673 (single thread) 54604 (multithread)
cinebench [13][14]	2409 (single thread) 43264 (multithread)	2256 (single thread) 32930 (multithread)
3DMark [15][16]	54475	46190

Aquestes proves no donen un valor amb unitats, si no que utilitzen unes puntuacions que serveixen per comparar entre processadors. Si s'observen els resultats d'aquests tres benchmarks, es pot veure

que el Intel Core Ultra 9 285K té en la majoria de proves una major puntuació que el AMD Ryzen 9 9900X. Aquesta diferència és mínima, per tant es pot concloure que aquests dos processadors tenen rendiments similars.

Simulació dels cores

Per a poder simular en Simplescalar els dos cores, cal adaptar (a més de les adaptacions ja fetes) les dades als requisits del simulador. Concretament Simplescalar no permet introduir valors de configuració que no siguin potència de 2 (en execpció de la latència). Per tant, cal arrodonir els paràmetres obtinguts a la potència de 2 més propera:

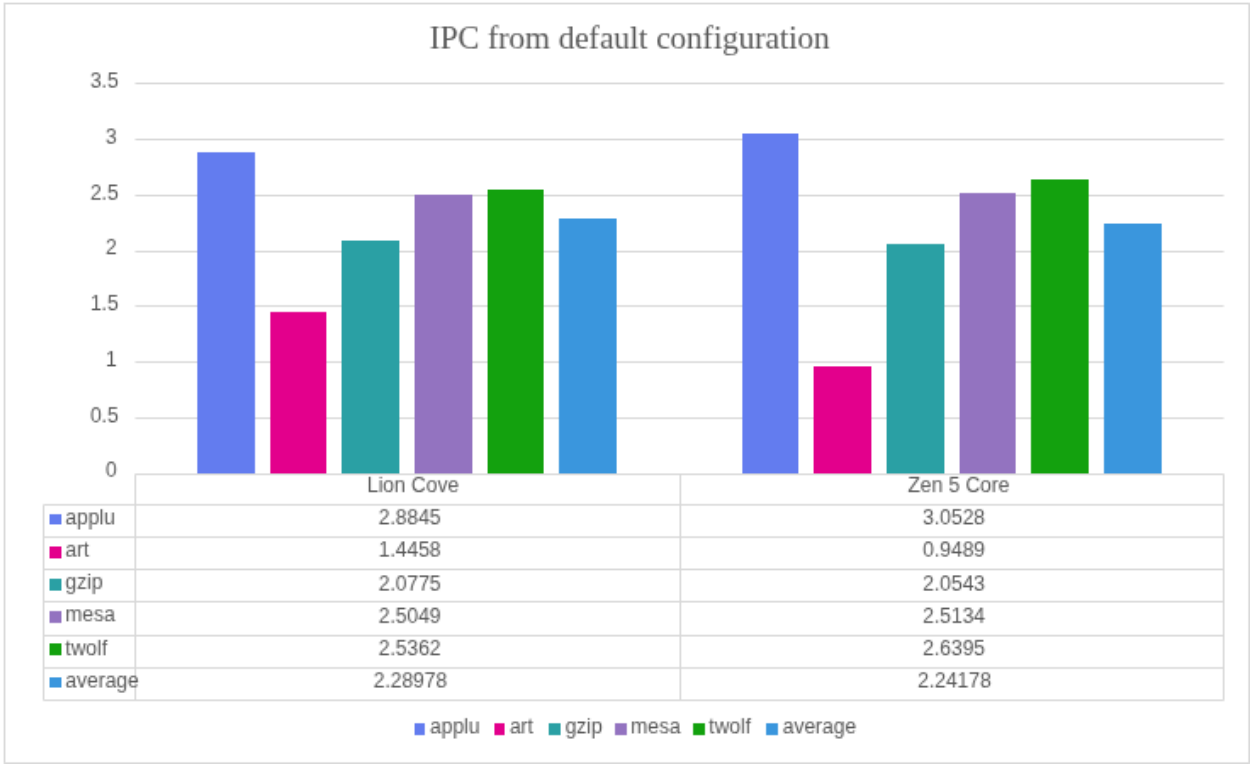
Adaptació valors a paràmetres de Simplescalar			
Paràmetre	**Valors originals**	**Paràmetre Simplescalar**	**Arguments Simplescalar**
Lion Cove			
k-via fetch	8 instruccions	-fetch:ifqsize	8
k-via decode	8 instruccions	-decode:width	8
k-via issue	8 instruccions	-issue:width	8
k-via commit	8 instruccions	-commit:width	8
ruu	576 instruccions	-ruu:size	512
lsq	309	-lsq:size	256
cache IL1	64:64:16:LRU	-cache:il1	il1:64:64:16:l
cache DL1	324:64:12:LRU	-cache:dl1	dl1:256:64:16:l
cache L2 (Unified)	4096:64:10:LRU	-cache:il2 -cache:dl2	dl2 ul2:4096:64:8:l
ample de banda DRAM	26 bytes/cicle	-mem:width	32
latència DRAM	496 cicles / 2 cicles	-mem:lat	496 / 2 cicles
ALUs Integer Simple	3	-res:ialu	3
ALUs Integer Multiply	3	-res:imult	3
ALUs Float Simple	2	-res:fpalu	2
ALUs Float Multiply	2	-res: fpmult	2
Zen 5 Core			
k-via fetch	4 instruccions	-fetch:ifqsize	4
k-via decode	4 instruccions	-decode:width	4
k-via issue	8 instruccions	-issue:width	8
k-via commit	8 instruccions	-commit:width	8
ruu	448 instruccions	-ruu:size	512
lsq	306	-lsq:size	256
cache IL1	64:64:8:LRU	-cache:il1	il1:64:64:16:l

cache DL1	64:64:12:LRU	-cache:dl1	dl1:64:64:16:l
cache L2 (Unified)	1024:64:16:LRU	-cache:il2 -cache:dl2	dl2 ul2:1024:64:16:l
ample de banda DRAM	32 bytes/cicle	-mem:width	32
latència DRAM	348 cicles / 2 cicles	-mem:lat	348 / 2 cicles
ALUs Integer Simple	3	-res:ialu	3
ALUs Integer Multiply	3	-res:imult	3
ALUs Float Simple	2	-res:fpalu	2
ALUs Float Multiply	2	-res: fpmult	2

10. Correcció de les dades per la simulació.

Resultats de la simulació

A continuació el resultat de la simulació del Lion Cove i del Zen 5 Core a partir de la configuració mencionada a la taula anterior:



11. Resultat simulació amb configuració original dels cores.

Podem veure primerament que l'impacte dels benchmarks és molt similar entre els dos cores. Cap dels cores destaca en un tipus d'us sobre l'altre. Encara així hi ha una diferència entre el IPC al executar el benchmark "applu". Es podria supossar que aquesta diferència és causada per la diferent configuració de cachés entre els cores, però realment la diferència és donada per la diferent latència de la DRAM en la simulació d'aquests (496 cicles al Lion Cove i 348 cicles al Zen 5 Core). Cal remarcar que realment la diferència de latència entre el Lion Cove i el Zen 5 no és tan gran en

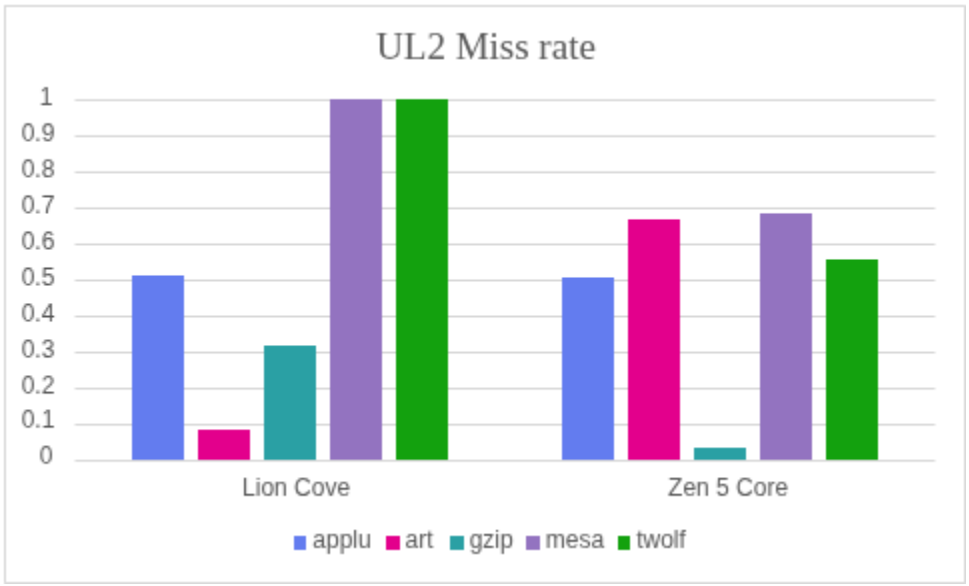
la configuració real, però la limitació de Simplescalar en el valor dels paràmetres ha fet que aquesta diferència sigui major. Així que es pot supossar que en una simulació amb valors exactes els IPCs dels dos cores serien molt més similars.

Concretament, el benchmark "applu" és l'únic que es veu força influenciat per aquesta diferència de latència de DRAM. Això és degut probablement al fet de que "applu" treballa amb un gran volum de dades que sobrepassen la capacitat de les cachés, fent així que l'accés a la DRAM sigui més habitual que als altres benchmarks.

Per arribar a aquesta conclusió s'ha configurat el Lion Cove amb les mateixes característiques de cachés que el Zen 5 Core, però amb les corresponents latències DRAM de cadascun.

D'altra banda també existeix una diferència entre els IPCs al executar el benchmark "art". Es pot observar que a diferència del benchmark "applu", en el Zen 5 Core el IPC és menor que al Lion Cove. Primerament es pot descartar la possibilitat de que estigui afectat per la latència de la DRAM, ja que contradiria la hipòtesis anterior. Per tant, probablement aquesta diferència vingui donada per la diferència de tamay de les cachés, concretament la caché de segon nivell unificada del Lion Cove té més capacitat total que la del Zen 5.

Si comparem les dades de miss rate entre el Lion Cove i el Zen 5 Core s'obté el següent:



12. Miss rate de la simulació amb configuració original.

Es pot observar al gràfic de la imatge 12 que la diferència és significativa en l'execució del benchmark "art". Això pot indicar que la localitat temporal d'instruccions o dades d'aquest benchmark és molt alta però la configuració de la L2 del Zen 5 Core no l'està aprofitant.

Per la resta de benchmarks els IPCs dels dos processadors són molt similars, això indica que encara tindre certes diferències, aquest dos cores tenen capacitats d'execució quasi idèntiques. Això es pot veure fàcilment si examinem la mitjana dels IPCs dels dos cores és pràcticament la mateixa.

Millores dels cores

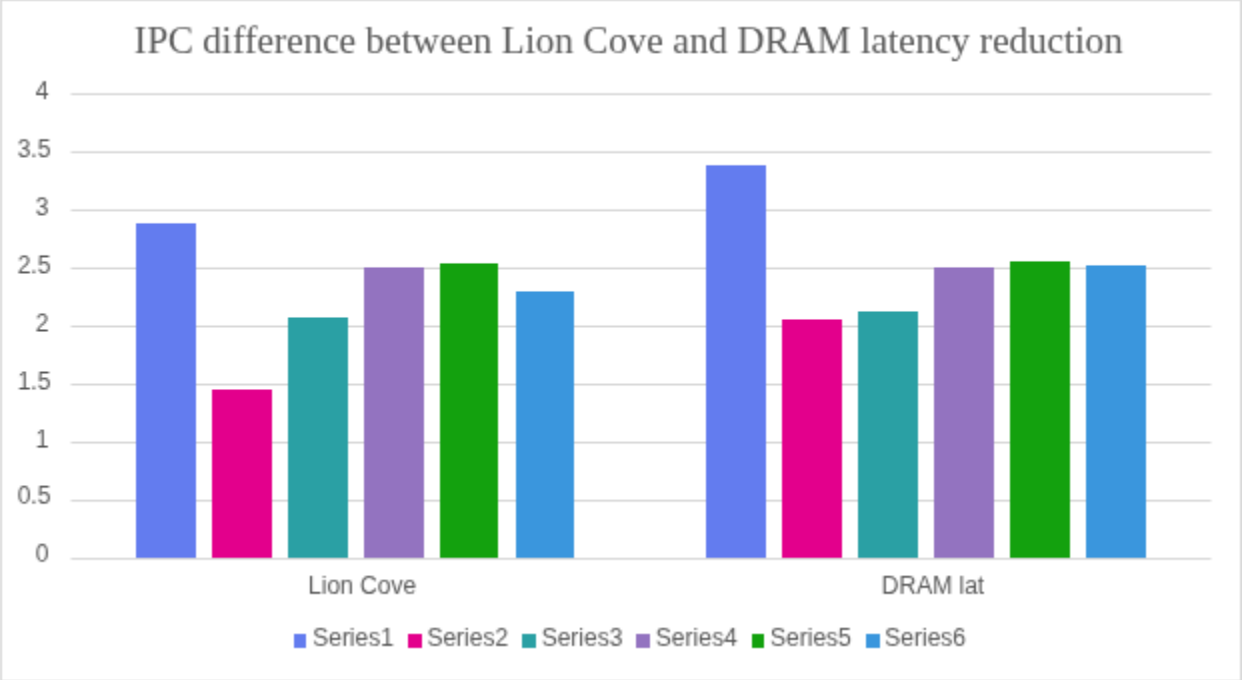
A continuació es faran una sèrie de propostes de millora dels cores.

Lion Cove

Primera millora

A partir de l'anàlisis del apartat anterior, una primera millora al Lion Cove seria reduir el seu temps d'accés a DRAM, ja que s'ha pogut observar que aquest afecta considerablement als programes amb una quantitat de dades que superin la capacitat de les cachés.

Per tant, la **primera millora** del Lion Cove serà reduir la seva **latència a DRAM a la meitat.**

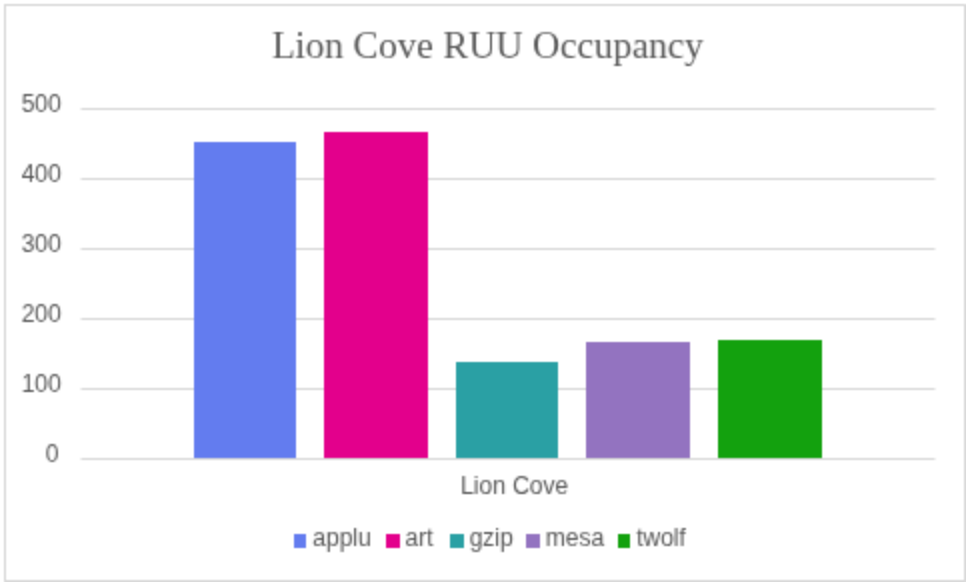


13. Comparació IPC entre Lion Cove i millora en la latencia de la DRAM.

Es pot observar al gràfic 13 que aquesta reducció de la latència ha estat una petita millora en la mitjana dels IPCs del Lion Cove.

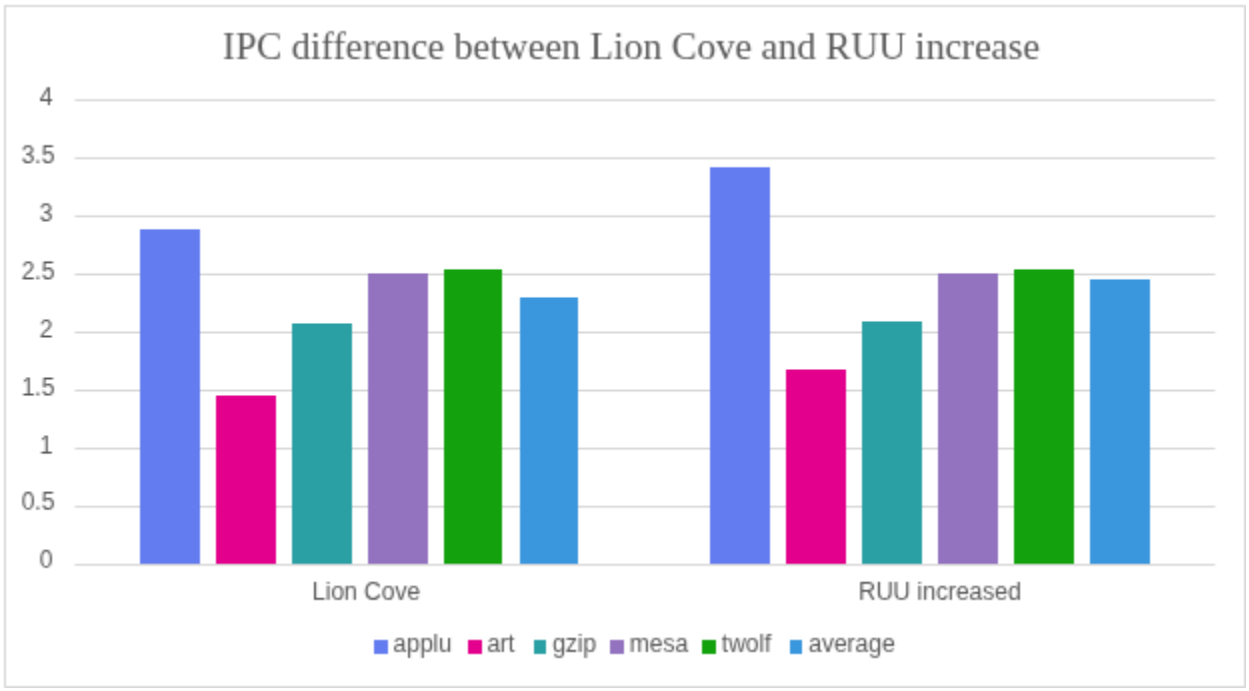
Segona millora

D'altra banda, una posible **segona millora del Lion Cove** és l'**augment de la seva finestra d'instruccions**. Això degut a que si observem l'ocupació de la RUU, hi ha benchmarks que mostren una ocupació promig que s'apropa a les entrades màximes de la RUU. Per tant, augmentant el seu tamany disminuirà la probabilitat de completar la RUU al màxim, augmentat el IPC ja que quan la RUU és completa, el processador ha de parar d'introduir noves instruccions fins que no hi hagi posicions lliures a la RUU.



14. Ocupació de la RUU en l'execució del benchmark amb el Lion Cove.

Es pot observar a la imatge 14 que, efectivament, hi ha dos benchmarks amb una alta ocupació de la RUU del Lion Cove. Per tant un augment de la finestra d'instruccions suposaria una millora del IPC.

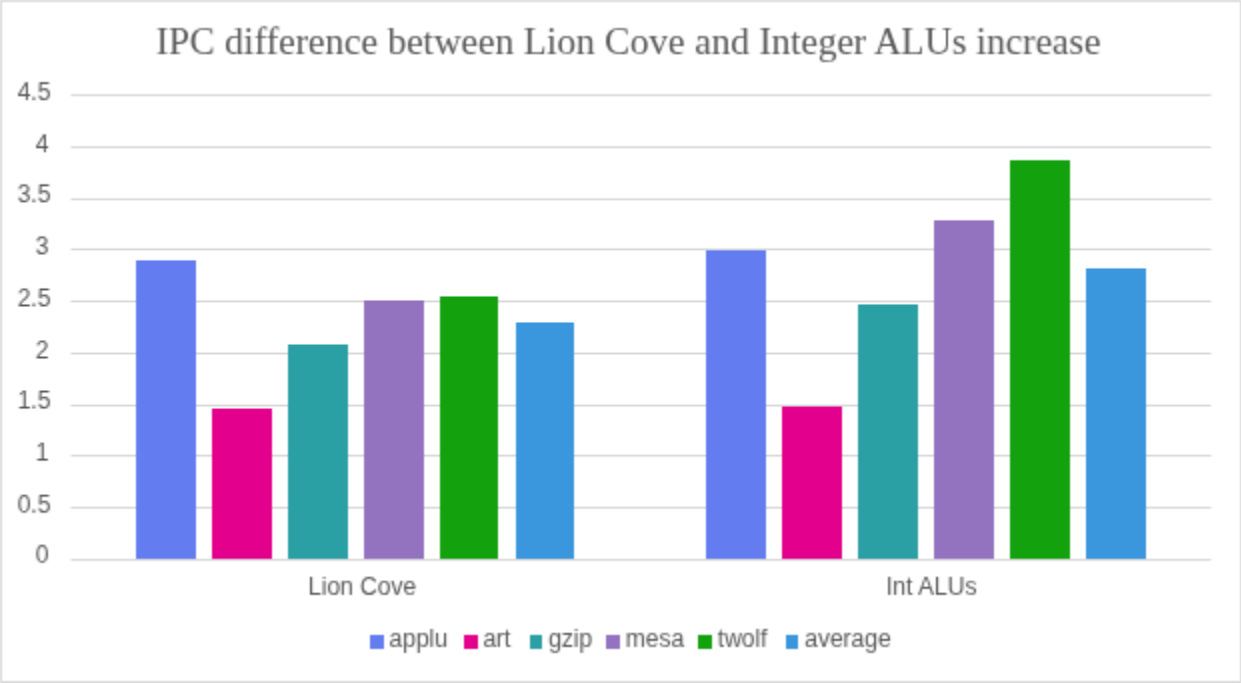


15. Comparativa Lion Cove amb versió modificada amb major finestra d'instruccions.

Es pot observar que sí hi ha hagut una millora dels IPCs en l'execució dels benchmarks que més ocupació de la RUU tenien.

Tercera millora

Per últim, una **tercera millora del Lion Cove** podria ser **augmentar la quantitat d'ALUs** que disposa. Concretament les ALUs d'enters, ja que les operacions aritmetiques d'enters són les més utilitzades en la majoria de programes comuns. Els càlculs aritmètics amb coma flotant són propis d'àmbits específics (gràfics, càlcul científic, etc), però aquests tipus de programes tenen l'alternativa d'utilitzar GPUs pels seus càlculs.



16. Comparació del IPC del Lion Cove i la seva versió modificada amb major nombre d'ALUs d'enters.

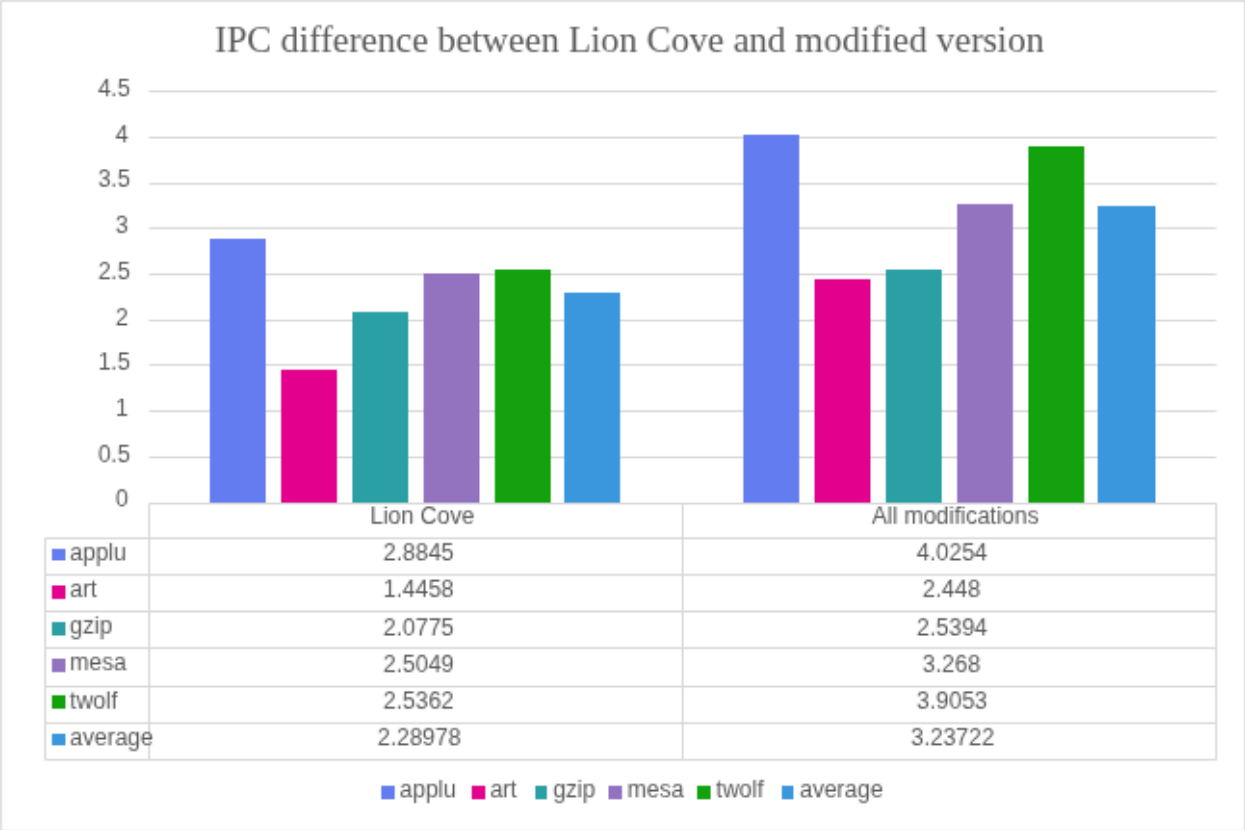
Es pot observar al gràfic 16 que l'agument de la quantitat d'ALUs suposa una millora del IPC en la majoria dels benchmarks (encara que aquestos treballin en coma flotant). Això és degut a que les operacions aritmètiques d'enters són també necessàries en instruccions de control com salts i comparacions. Per tant, un augment del nombre d'ALUs permet una major simultaneitat en l'execució d'instruccions comunes a tots els programes (independentment dels tipus de dades que tractin).

Implementació de totes les millores

Fins ara s'ha comparat el Lion Cove original amb cadascuna de les modificacions. En aquest apartat s'aplicaran totes les modificacions en la mateixa simulació per veure l'impacte d'aquestes tres alhora. A continació les modificacions que s'han aplicat:

Paràmetre	Lion Cove original	Lion Cove modificat
latència DRAM	496 / 2	248 / 2
finestra d'instruccions	512	1024
nombre ALUs d'enters	3 simple / 3 multiplicació	5 simple / 5 multiplicació

17. Comparació de configuracions entre Lion Cove original i modificat.



18. Comparació IPC Lion Cove amb versió millorada.

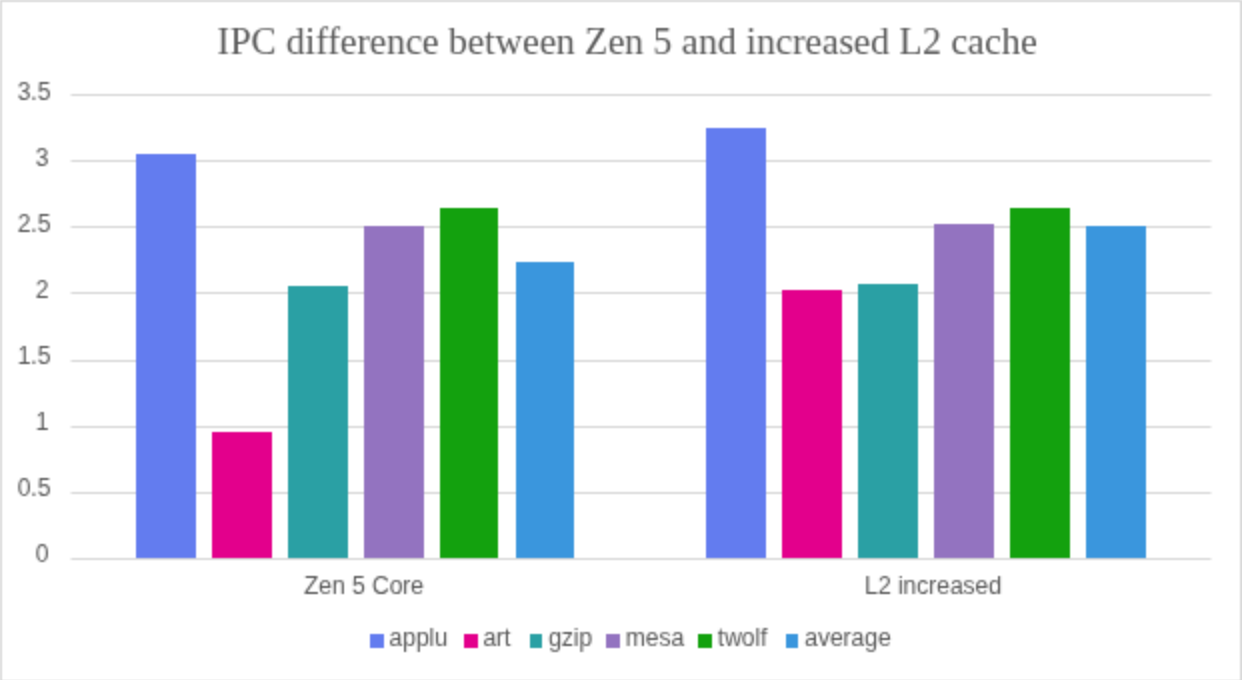
Es pot observar al gràfic 18 que al aplicar les tres modificacions a la configuració del Lion Cove hi ha hagut un augment en el IPC de l'execució de tots els benchmarks.

Si observem exclusivament el increment del IPC mitjà, aquest ha augmentat un 41% respecte el valor original. Aquest és un augment considerable, però probablement aplicar aquestes modificacions al Lion Cove suposarien un gran augment de la complexitat del hardware i del consum que no factibles.

Zen 5 Core

Primera millora

Com s'ha pogut observar a l'apartat anterior, la configuració de caché L2 del Zen 5 Core afecta a programes com "art" que tenen una localitat temporal de dades o instruccions molt elevada. Per tant, una **primera millora del Zen 5 Core** serà l'augment del **doble de la seva capacitat total de caché L2**.

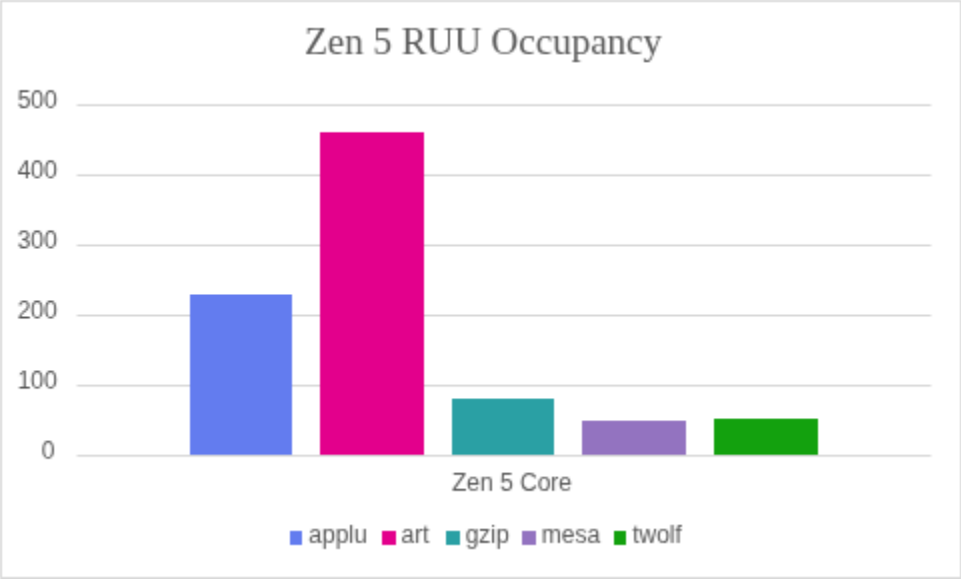


19. Comparació IPCs entre Zen 5 Core i la versió amb major capacitat total de caché L2.

Es pot observar que aquesta millora afecta positivament al benchmark "art", ja que es redueixen els L2 miss.

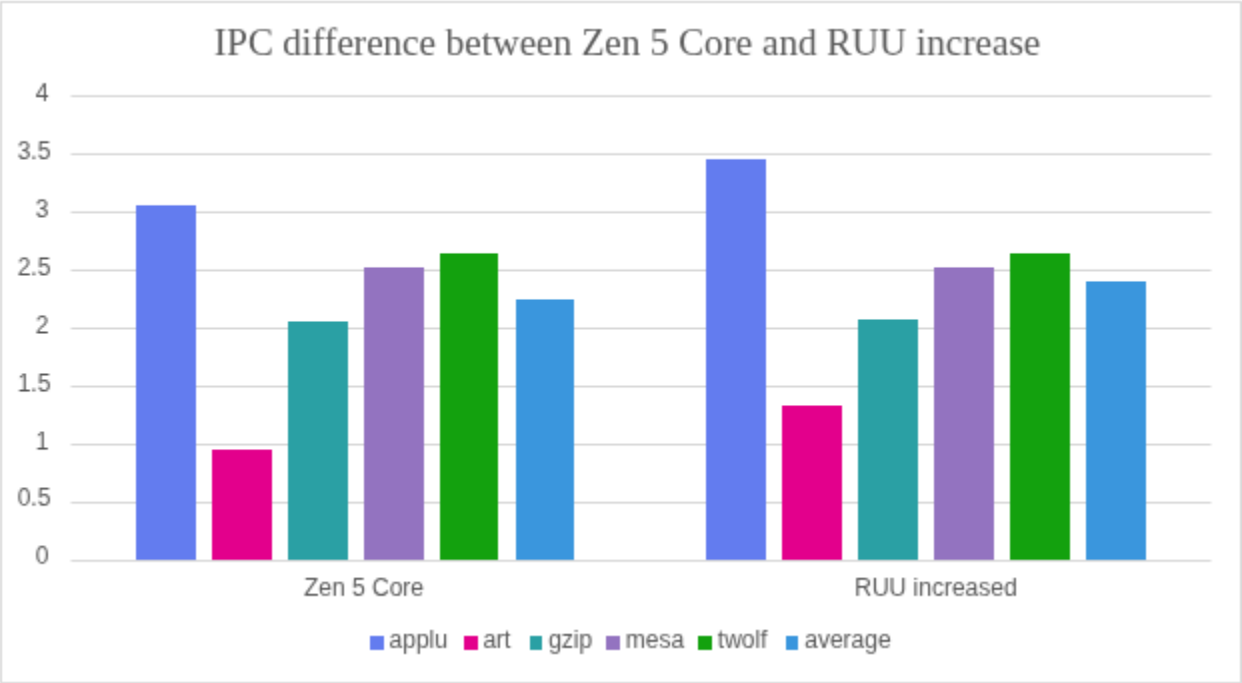
Segona millora

Al igual que amb el Lion Cove, una **segona millora** del Zen 5 podria ser **augmentar la seva finestra d'instruccions**, ja que hi ha benchmarks que tenen un nivell d'ocupació molt alt.



20. Ocupació promig de la RUU del Zen 5.

Como indica el gràfic 20 el benchmark "art" té un alt nivell d'ocupació, per tant augmentar el doble la finestra d'instrucions augmentaria el IPC en l'execució de programes com "art".

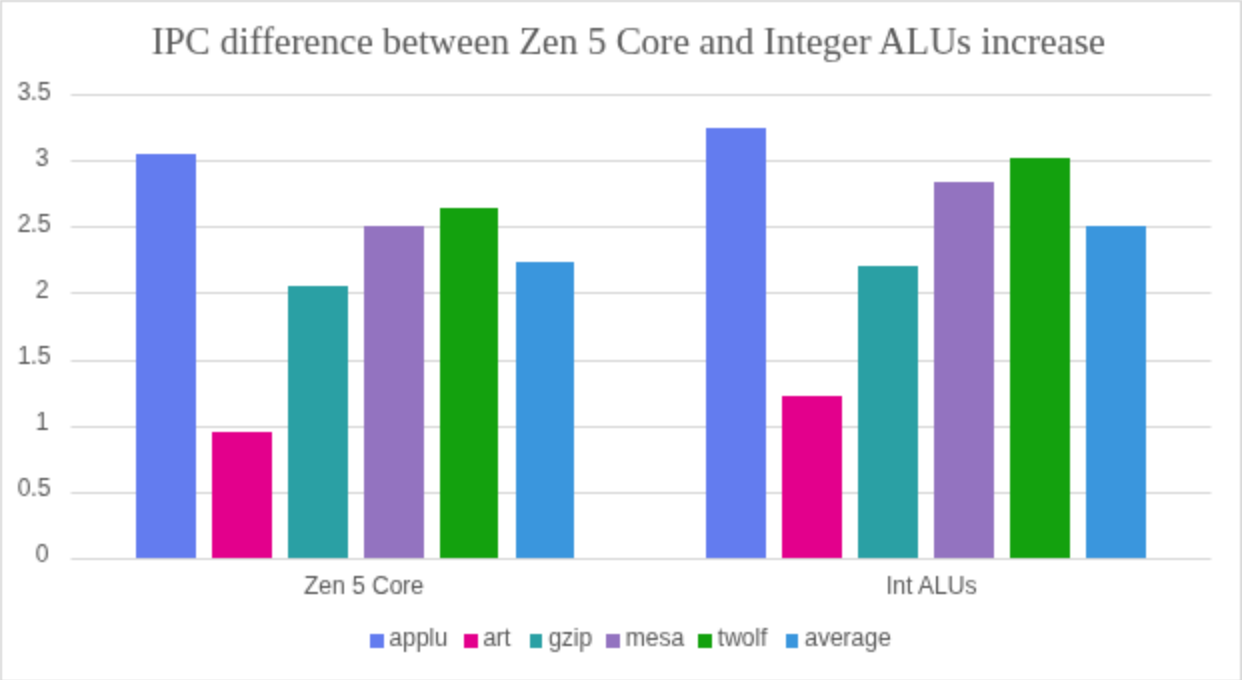


21. Comparació Zen 5 Core amb versió modificada amb major finestra d'instruccions.

Es pot veure al gràfic 21 que, encara que el canvi es mínim, hi ha una millora dels IPCs en l'execució dels benchmarks "applu" i "art", que són aquells que major ocupació de la RUU tenien.

Tercera millora

Per últim, a l'igual que al Lion Cove, una **tercera millora del Zen 5 Core** podria ser **augmentar el nombre d'ALUs de càlcul enter**. A continuació la comparació:



22. Comparació del IPC del Zen 5 Core i la seva versió modificada amb major nombre d'ALUs d'enters.

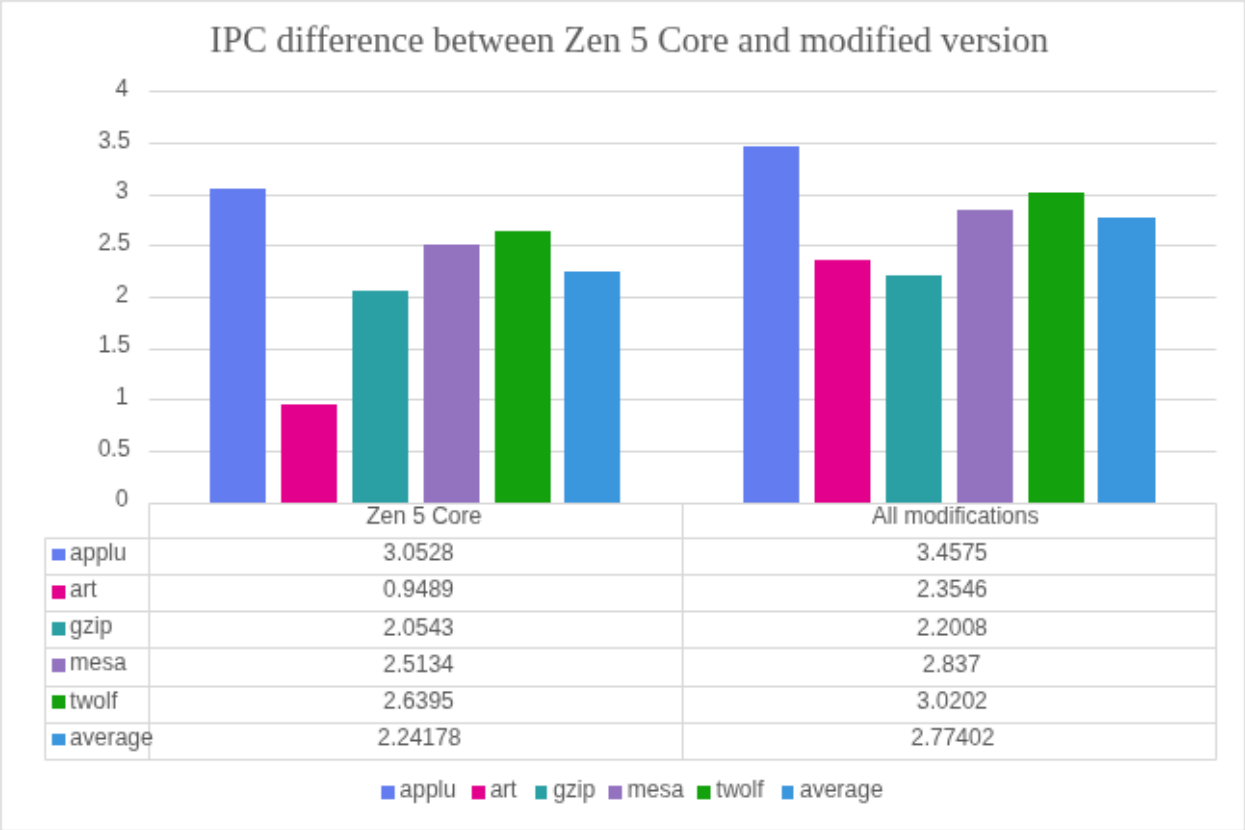
Es pot observar que hi ha un augment del IPC a la majoria de benchmarks, independentment del tipus de dades que tractin. Això degut al mateix motiu explicat en la tercera modificació del Lion Cove.

Implementació de totes les millores

Fins ara s'ha comparat el Zen 5 Core original amb cadascuna de les modificacions. En aquest apartat s'aplicaran totes les modificacions en la mateixa simulació per veure l'impacte d'aquestes tres alhora. A continació les modificacions que s'han aplicat:

Paràmetre	Zen 5 Core original	Zen 5 Core modificat
configuració L2	1024 sets / 64 tamany bloc / 16 associativitat	2048 sets / 64 tamany bloc / 16 associativitat
finestra d'instruccions	512	1024
nombre ALUs d'enters	3 simple / 3 multiplicació	5 simple / 5 multiplicació

23. Comparació de configuració entre Zen 5 original i modificat.



	Zen 5 Core	All modifications
applu	3.0528	3.4575
art	0.9489	2.3546
gzip	2.0543	2.2008
mesa	2.5134	2.837
twolf	2.6395	3.0202
average	2.24178	2.77402

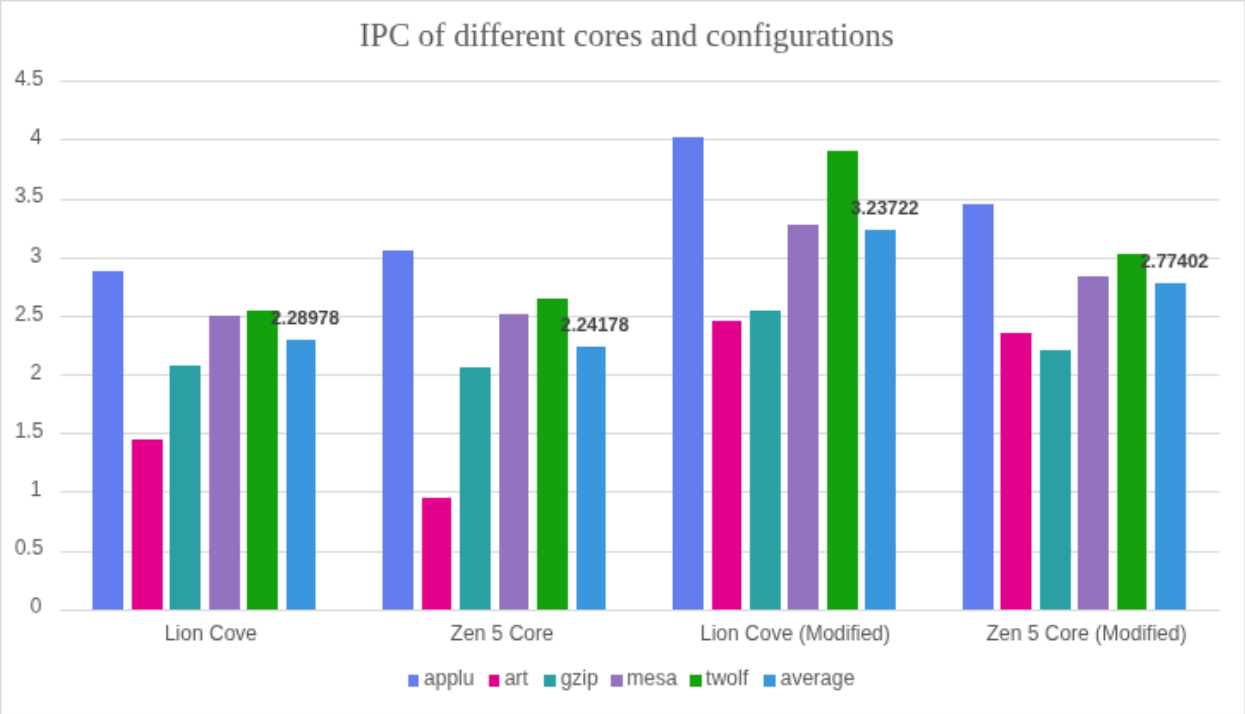
24. Comparació IPC Zen 5 Core amb versió modificada.

Es pot apreciar al gràfic 24 que al aplicar les tres modificacions a la configuració del Zen 5 Core hi ha hagut un augment en el IPC de l'execució de tots els benchmarks. El benchmark que més s'ha beneficiat ha sigut "art", però la resta també han obtingut beneficis relatius.

Concretament, si observem únicament la mitjana, les modificacions han comportat una millora del 23% sobre el valor mitjà original. No és una millora molt elevada, per tant es pot suposar que no es factible l'augment del hardware (i per tant del consum) per tal d'aconseguir aquest petit increment del IPC.

Comparativa dels dos cores

A continuació es farà una comparativa entre la millora de cadascun dels cores. A continuació els IPCs obtinguts de cadascuna de les versions dels cores:



25. IPCs dels dos cores originales i les seves versions modificades.

Es pot observar a la imatge 25 que en tots dos casos les millores dels cores han suposat un increement en el IPC promig. Encara així, es pot observar que el Lion Cove ha tingut un increment major (del 41%), mentre que el Zen 5 Core ha tingut un increment del 23%. El factor principal d'aquesta diferència ha estat el menor k-via del fetch i decode del Zen 5 Core, degut a que aquest compta amb SMT (Simultaneous Multithreading), i s'ha decidit simular només un fil d'execució. Aquesta diferència limita la capacitat de millora del Zen 5 respecte el Lion Cove.

Es podria haver proposat com a millora del Zen 5 l'augment de la k-via del fetch i decode, però s'ha considerat que això canviaria radicalment les decisions de disseny originals respecte la possibilitat de SMT del Zen 5. Justament l'intenció d'aquest anàlisis es contemplar les diferències entre el Lion Cove i el Zen 5. A més, això aproparia massa la configuració del Zen 5 a la del Lion Cove, cosa que no permetria fer un anàlisis de les diferències proper a la realitat.

Un altre factor important que ha limitat la comparativa entre els dos cores ha estat la limitació de Simplescalar d'haver d'utilitzar paràmetres potències de 2. Valors que eren similars però no iguals entre els cores s'han hagut de simular amb valors iguals, això també ha comportat una menor diferència de comportament entre aquests.

Comentari de text

"Compared to Arrow Lake, AMD's Ryzen 9 9900X has very uniform clock speeds within a CCD. All cores on the first CCD can run at 5.6 GHz, while all cores on the second CCD run at 5.35 GHz. Arrow Lake can clock higher at 5.7 GHz if you pick the right core, but AMD and Intel are running their high performance cores at very similar clocks this generation. Even though more of AMD's cores can reach the highest clock speeds, operating systems will have to categorize cores by their expected performance class to optimize low threaded workloads."

En aquest fragment de text es pot observar la tendència de les empreses fabricadores de processadors en trobar l'equilibri entre rendiment i consum. Si no importés el consum el més adient seria tindre tots els cores amb freqüències de rellotge altes, ja que així s'augmentaria la velocitat de processament de qualsevol procés. Però el consum és un factor important, sobretot en dispositius portàtils on hi ha una dependència d'una bateria.

Es pot observar que el processador d'Intel opta més per aquesta separació entre cores d'alt rendiment i cores més eficients. Això permet que els programes que compten amb poca paralel·lització s'executin en cores més potents, mentre que aquells programes més paral·lelitzats utilitzin els cores més eficients. D'aquesta manera no es perd velocitat de processament i es redueix el consum.

D'altra banda el processador d'AMD té una freqüència de rellotge molt similar en tots els cores, això suposa un major consum ja que impossibilita la capacitat de repartir els programes segons les necessitats específiques de cadascun (com s'ha explicat en el cas d'Intel).

Davant d'aquesta nova tendència, els encarregats de que sigui efectiva són els sistemes operatius, ja que són aquests els que han de conèixer les capacitats de cada core i han de decidir l'assignació que es fa de cada programa a aquests. Per tant, es pot veure que per que aquesta millora del hardware sigui eficaç, cal una implementació software que tingui en compte la nova dinàmica.

Referències

Intel Lion Cove

[1] Chips and Cheese, "Lion Cove: Intel's P-Core Roars" 2024. [Online]. Disponible: https://chipsandcheese.com/p/lion-cove-intels-p-core-roars

[2] Techbloat "Intel CPU Cache Line Size" 2025. [Online]. Disponible: https://www.techbloat.com/intel-cpu-cache-line-size.html

[3] "Reverse Engineering of Cache Replacement Policies in Intel Microprocessors and Their Evaluation", Andreas Abel and Jan Reineke, Department of Computer Science, Saarland University. [Online]. Disponible: https://embedded.cs.uni-saarland.de/publications/ISPASS14.pdf

[4] HWCooling, "Intel's new P-Core: Lion Cove is the biggest change since Nehalem", 2024. [Online]. Disponible: https://www.hwcooling.net/en/intels-new-p-core-lion-cove-is-the-biggest-change-since-nehalem/

[5] Intel, "Procesador Intel® Core™ Ultra 9 285K". [Online]. Disponible: https://www.intel.la/content/www/xl/es/products/sku/241060/intel-core-ultra-9-processor-285k-36m-cache-up-to-5-70-ghz/specifications.html

Zen 5 Core

[6] Chips and Cheese, "AMD's Ryzen 9950X: Zen 5 on Desktop", 2024. [Online]. Disponible: https://chipsandcheese.com/p/amds-ryzen-9950x-zen-5-on-desktop

[7] Daniel Lemire's blog, "Measuring the size of the cache line empirically", 2023. [Online]. Disponible: https://lemire.me/blog/2023/12/12/measuring-the-size-of-the-cache-line-empirically/

[8] Chips and Cheese "Discussing AMD's Zen 5 at Hot Chips 2024", 2024. [Online]. Disponible: https://chipsandcheese.com/p/discussing-amds-zen-5-at-hot-chips-2024

[9] Hot Chips 2024, "Next Generation "Zen 5" "Core, 2024. [Online]. Disponible: https://hc2024.hotchips.org/assets/program/conference/day2/24_HC2024.AMD.Cohen.Subramony.final.pdf

[10] Wikipedia, " Cache replacement policies". [Online]. Disponible: https://en.wikipedia.org/wiki/Cache_replacement_policies

Benchmarks

[11] PassMark, "Intel Core Ultra 9 285K". [Online]. Disponible: https://www.cpubenchmark.net/cpu.php?id=6296&cpu=Intel+Core+Ultra+9+285K

[12] PassMark, " AMD Ryzen 9 9900X". [Online]. Disponible: https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+9+9900X&id=6171

[13] overclocking.com, "Review: Intel Core Ultra 9 285K and Ultra 5 245K". [Online]. Disponible: https://en.overclocking.com/review-intel-core-ultra-9-285k-and-ultra-5-245k/7/

[14] overclocking.com, "Review : AMD Ryzen 9 9900X". [Online]. Disponible: https://en.overclocking.com/review-amd-ryzen-9-9900x/4/

[15] overclocking.com, " Review: Intel Core Ultra 9 285K and Ultra 5 245K". [Online]. Disponible: https://en.overclocking.com/review-intel-core-ultra-9-285k-and-ultra-5-245k/11/

[16] overclocking.com, " Review : AMD Ryzen 9 9900X". [Online]. Disponible: https://en.overclocking.com/review-amd-ryzen-9-9900x/5/