УДК 004.4

DOI: 10.18372/2073-4751.70.16841

Бойко Т.П.,
Русанова О.В., к.т.н.,
orcid.org/0000-0003-0145-3012

СПОСІБ ЕНТРОПІЙНОГО КОДУВАННЯ ВІДЕО НА БАЗІ РОЗШИРЕНОГО НАБОРУ ІНСТРУКЦІЙ SIMD AVX-512

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

edutm20@gmail.com

Вступ

Сьогодні спостерігається швидке зростання інтересу до перегляду відео в інтернеті на різноманітних пристроях користувача та значного підвищення вподобань щодо якості відео. Це потребує суттєвих витрат на збереження, обробки відео і як наслідок, може обмежувати пропускну здатність інтернет-мереж. Тому все більше актуальними є питання оптимізації та провадження нових алгоритмів для зростання якості відео, зниження завантаженості серверів компаній та мереж користувачів. Все це зробило необхідністю впровадження нового формату стиснення відео – AV1, що майже на 50% знижує бітовий потік стисненого відео за тієї ж якості. Найбільш проблемною частиною відео-кодеків, що має низькі або вістуні можливості для паралельної обробки, є ентропійне стиснення – що є фінальним узагальненим методом прибирання (стиснення) даних. Сучасні мультипроцесорні системи із традиційною архітектурою зробили можливим створення програмних реалізацій відеокодеків на рівні крупно-зернистого паралелізму, проте це не вплинуло на можливості розпаралелювання ентропійного стиснення.

Альтернативою підвищення продуктивності програм є використання мультипроцесорних систем із підтримкою розширеного набору SIMD-команд, що дозволяють реалізувати розпаралелювання на дрібно-зернистому рівні. У роботі [7] розглянута декодування формату відео AV1 на базі SIMD інструкцій SSE та AVX2. В той же час для розширеного набору інструкцій SIMD AVX-512, що набувають в останній час популярність подібних робіт немає. Щодо останніх, відомі наукові праці, де вивчається вплив роботи розширеного набору інструкцій SIMD AVX-512, але поза межами відео кодування. Так, в роботі [1] Готшлаг та Матіас порівняли роботи коду із використанням розширеного набору інструкцій AVX2 та AVX-512, а в роботі [2] Дж. Тех дослідив AVX-512 в інших частинах кодування відео поза межами ентропійного кодування.

Мета

Метою даної роботи є зменшення часу ентропійного кодування відео з використанням можливостей процесорів із розширеним набором інструкцій типу AVX-512 за рахунок розпаралелювання і використання додаткових SIMD інструкцій у порівнянні з AVX2 та SSE.

Основна частина

У роботі спочатку розглянуто відомий алгоритм ентропійного декодування в репозиторії libaomAV1 [6] для подальшого пояснення його векторизації. Далі описані векторизовані версії алгоритму ентропійного декодування, зокрема версії SIMDAVX2 [1]. Після цього запропонований підхід ентропійного кодування відео з використанням можливостей сучасних процесорних систем з підтримкою набору інструкцій SIMDAVX-512 для вдосконалення існуючих версій алгоритму та проведенийїх порівняльний аналіз. На завершення роботи досліджується продуктивність процесорівSkylake-Хна базі запропонованого алгоритму ентропійного декодування SIMDAVX-512 та зазначені можливі наслідки використання таких інструкцій.

Короткий огляд алгоритму ентропійного декодування в репозиторії AV1 на основі [1, 7] (рис. 1)

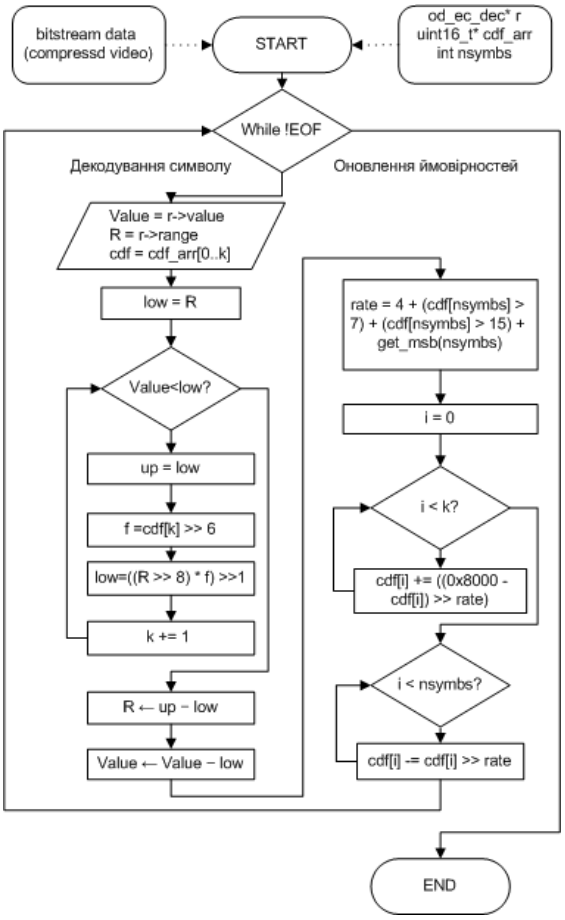

Рис. 1. Flow-chart діаграма ентропійного декодування відео AV1

Логічною частиною кодування відео за «AV1» є ентропійне кодування синтаксичних елементів мультисимвольним методом (арифметичне кодування у M символів, M є [2, 14]). Цей метод є адаптивним – тобто модель кумулятивного розподілу ймовірностей оновлюється для декодуванням кожного наступного символу. Машинне представлення математичної моделі ймовірностей масштабоване із [0..1] на число 2^{15}, тому представляється «M» 15-бітними беззнаковими цілими числами, до того для покращення пропускної здатності процесорної системи, усі операції множення і його результати вміщуються в 16 бітне ціле беззнакове число. Хоча модель ймовірностей має 15-бітну точність, проте в ентропійному кодуванні використовуються старші 9 біт представлення ймовірності [8].У наведеному алгоритмі R – поточна довжина інтервалу арифметичного кодера, а Value – значення кодового рядка. Довжина інтервалу R ділиться на 256 (2^8) перед множенням, щоб значення добутку((R >> 8) × f) не перевищив 16 розрядів.

Після декодування символу оновлюється кумулятивний розподіл ймовірностей. Врахуємо те, що $C_m(n)$ у математичній моделі – раціональне число в інтервалі [0..1], то у моделі машинних обчислень – це число [0..2^{16}-1].

$$c_m(n) = \begin{cases} c_m(n-1)\cdot(1-\alpha), m < k \\ c_m(n-1)\cdot\alpha\big(1-c_m(n-1)\big), m \geq k \end{cases} \quad (1)$$

Дана система рівнянь конвертується в алгоритм оновлення ймовірностей, що представлено на рис. 1 (праворуч).

Векторизація алгоритму

За основу для векторизації взято скалярний алгоритму AV1, який наведено ілюстративно у [8] та версія із використанням набору інструкцій SIMD SSE [1]. У роботі [1] цикл із скалярними змінними на послідовність інструкцій над масивами довжини M, M є [2..16], де 16 – кількість слів у регістрі векторного призначення ymm довжиною 256 біт (табл. 1). Незалежно від змінної вхідної довжини алфавіту M є [2..16], завжди паралельно обчислюється максимальне число – 16 значень без знакових слів. Для адаптації операцій множення над елементами масивів для можливості паралельного обчислення SIMD у межах 16 бітної доріжки (lane) вектора – використовуються інструкції vmulhi [word], [word] (залишає лише старше слово добутку), попередньо зсунувши вправо кожен із множників скалярного алгоритму. Кроки існуючого векторизованого алгоритму [7] на базі декодування символу у вигляді «псевдо-коду» наведено у табл. 1 колонці «AVX2».

Векторизовний алгоритм [1] оновлення ймовірностей замість двох циклів по умові використовує паралельні обчислення із маскою. Спочатку створюється маска на основі декодованого символу (одиниці – значення слів доріжок від нуля

до декодованого символу – 2^{16}-1, всі доріжки векторного регістру – 0). Далі за одну SIMD-інструкцію від маски віднімається значення ймовірностей і отримуємо ті ж результати 0x80000 – cdf[i], і є [0..k-1],. та (–cdf[i]), і є [k..N]. Маска використовується із попереднього кроку 1 (алгоритм декодування символу). Кроки векторизованого алгоритму у представленні «псевдо-коду» наведено у табл. 1 колонка «AVX2» нижче.

Опис запропонованого підходу ентропійного кодування для AVX-512

Недоліками алгоритму ентропійного декодування [1] є те що він, по-перше, неефективно використовує значення векторних регістрів, що залишились при декодуванні символу у функції оновлення кумулятивних ймовірностей, а по-друге, зберігає доріжки (lanes) векторного регістру ітеративно (в залежності від M), за рахунок скалярних інструкцій, оскільки в наборах SIMDSSE та AVX2 відсутні інструкції для збереження змінної кількості доріжок (слів) із векторного регістру [9]. Це так звана проблема лівостороннього пакування – коли потрібно зберегти в локацію пам'яті доріжки із векторного регістру, номери яких – змінні типу run-time [9]. Тому обмежуючись набором інструкцій SIMD SSE або AVX2 ці проблеми неможливо вирішити ефективно:можливо зробити спеціалізації функції ентропійного декодування для кожного із варіантів M [2..14], оскільки масиви кумулятивного розподілу зберігаються в памяті із вирівнюванням 2 байти, запис всього значення векторного регістру призведе до їх перезапису і як наслідок, некоректної роботи ентропійного декодування. Така можливість лише мінімізує кількість ітеративних збережень, але не прибере їх повністю. Вирішення цієї проблеми (лівостороннього пакування) можлива із набором інструкцій SIMD AVX-512.

У версії функції ентропійного кодування із набором розширених інструкцій AVX-512 використовується k-регістр спеціального призначення довжиною 32 біт, який доступний тільки в мікроархітектурах із підтримкою розширеного набору інструкцій AVX-512 (Skylake-X, Icelake, Tigerlake). В цей регістр зберігається бітова маска під час оновлення ймовірностей кумулятивного розподілу (табл. 1). За допомогою інструкції із набору розширень SIMD AVX-512 vpmaskstore {k}, ymm0 можна одразу порівняти та зберегти тільки ті 16-бітні беззнакові значення доріжок (lanes) із ymm0, для яких номер слова у векторного регістрі відповідає відповідному біту у регістрі спеціального призначення {k}.У табл. 1 наведені набори інструкцій для скалярної версії, векторизованої для SSE/AVX2та запропонованої авторами для AVX-512.

Таблиця 1. Оновлення ймовірностей ентропійного кодування на різних мікроархітектурах, послідовність необхідних інструкцій функції

Скалярна	**Набір розширень SIMD: SSE/AVX2**	**Набір розширень SIMD: AVX-512**
for i = 0; i < Value; i++ do	for i = 0; i < N-1; i++ do	cdf ← mask_store(cdf[i], {k_{reg}})
cdf[i] ← c_k (1 - α),	cdf[i] ← sub[i]	
for i; i < N; i++ do	end for	
cdf[i] ← c_k * α (1 – c_k)		
end for		

Крім того, у алгоритмі декодування символу за допомогою набору AVX-512 можна зменшити кількість інструкцій (табл. 2). Наприклад, якщо у векторизованій версії AVX2 для порівняння значення двох векторних регістрів ymm і збереження цього результату у вигляді маски в регістрі загального призначення потрібно дві інструкції (порівняння векторних регістрів, збереження старших бітів із векторного регістру в регістр загального призначення), то для AVX512 – всього одна.

Це також досягається із застосуванням агрегованої інструкції, яка порівнює значення векторних регістрів і зберігає старші біти доріжок (lanes) у k-регістр.

Таблиця 2. Алгоритм ентропійного кодування символу у AV1 на різних мікроархітектурах, перелік необхідних інструкцій

Скалярна	**Набір розширень SIMD: SSE/AVX2**	**Набір розширень SIMD: AVX-512**
low ← R	low[0..15] ← R	low[0..15] ← R
for k = 1; Value < low; k = k + 1 **do**	f[0..15] ← saturate(2^9 − (c[0..15] >> 6))	f[0..15] ← saturate(2^9 − (c[0..15] >> 6))
up ← low	low[0..15] ← ((R >> 8) × f[0..15]) >> 1	low[0..15] ← ((R >> 8) × f[0..15]) >> 1
f ← 2^9 − (c_k>> 6)	cmp[0..15] ← low[0..15] >up[0..15] ? 0 : -1	$\mathbf{k_{reg}}$ **mask** (16 bits) ← low[0..15] >up[0..15] ? 0 : -1
low ← ((R >> 8) × f) >> 1	sub[0..15] ← up[0..15] - low[0..15]	sub[0..15] ← up[0..15] -low[0..15]
end for	**mask** ← MSB(cmp[0..15])>>1	-
R ← up – low	k ← TCZ(mask)	k ← TCZ(k_{reg})
Value ← Value – low	R ← sub[k]	R ← sub[k]

Використання інструкцій AVX-512 дозволяє зменшити кількість інструкцій для процесора та не використовувати зайві регістри для збереження проміжного результату як у частині алгоритму ентропійного кодування відео, що відповідає за декодування символу, так і в частині, що відповідає за оновлення ймовірностей.

Із табл. 2 бачимо, що найбільш неефективна (вимагає більше інструкцій та часу на виконання) версія коду скалярна, потім версія для SSE чи AVX2, найбільш ефективна – версія із AVX-512. Можна припустити, що при великих значення символу, що декодується (може бути в межах інтервалу [0..15] чим ближче до верхньої межі інтервалу, тим більше буде відносна продуктивність коду із використанням розширеного набору інструкцій AVX-512, і навпаки – скалярна версія буде швидша на коротких інтервалах якщо декодоване значення буде близько 0. Тому слід досліджувати роботу ентропійного кодування на необхідних користувачу вхідних відеофайлах, та збирати таку статистику і вже потім вирішувати, яку саме версію алгоритму краще використовувати.

Дослідження роботи процесорів під час ентропійного декодування

Дослідження проведено на процесорі мікроархітектури IntelSkyalake. Використовувались збірки трьох версій декодеру AV1 із репозиторію libaom на базі AVX-512, AVX2 та C, в усіх версіях декодеру були вимкнені (закоментовані) усі інші частини відео-кодування окрім ентропійного. Вхідними параметрами для декодеру є набір з декількох відео-фалів (табл. 3), що закодовані різними параметрами та в різній роздільній здатності. Результатами дослідження є середньозважена кількість кадрів на секунду (fps) для різних збірки декодеру: на базі коду нової версії на базі AVX-512, що пропонується, та існуючих версій коду на мові C та на базі SIMDAVX2 (табл. 3).

Версія AVX-512 має перевагу над скалярною та SIMD SSE/AVX2. Із табл. 3 можна побачити, що час виконання залежить від довжини вхідного масиву cdf (рис. 1) та його значень. Середнє прискорення по набору відео під час тестування складало ~4%. Версія на базі SIMD AVX-512 працює швидше ніж існуючі алгоритми на базі SIMD AVX2 у будь яких випадках, а середнє прискорення складає ~2%. Нерівномірність результатів можна пояснити можливою присутністю інших процесів у системі й інших модулів відео кодування, їх впливом на тактову частоту процесору та переривання, які вони можуть створювати.

Таблиця 3. Декодер на базі різних версій алгоритму ентропійного декодування на деяких відео файлах формату AV1 (кадрів/секунду – framepersecond–вище – краще)

Назва файлу	Середнє значення символу	AVX-512, спроба 1	AVX-512, спроба 2	AVX-512, спроба 3	AVX2 спроба 1	AVX2 спроба 2	AVX2 спроба 3	C99 спроба 1	C99 спроба 2	C99 спроба 3	AVX-512 середнє	AVX2 середнє	C99 середнє
Big_Buck_Bunny_1080_10s_30MB.webm	5.90	20.60	20.72	20.74	18.80	18.61	18.63	18.44	18.74	18.45	20.69	18.68	18.54
CityHall_1920x1080.webm	4.76	57.03	56.92	57.48	56.81	57.17	57.39	56.65	55.96	56.18	57.14	57.12	56.26
CityHall_3840x2160.webm	4.75	19.08	19.02	18.95	18.70	19.10	19.06	19.06	19.10	19.06	19.02	18.95	19.07
Sparks (1000 кадрів)	3.89	94.99	93.41	93.88	91.62	90.19	92.23	91.27	92.79	93.78	94.09	91.35	92.61
Chimera-AV1-10bit-1920x1080-6191kbps.ivf (300 кадрів)	5.28	55.57	54.36	55.03	53.72	54.12	53.73	52.64	51.76	54.92	54.99	53.86	53.11

Під час тестування вищезгаданих версій алгоритму ентропійного кодування при декодуванні двох відео CityHall_3840x2160.webm та CityHall_3840x2160.webm не було досягнуто суттєвого прискорення, що пояснюється саме малим середнім значенням декодованих символів ~ 4.

Перевагою скалярного алгоритму в деяких відео є його робота в умовах коли усі символи, що будуть декодовані, є відносно невеликим числом (0..4). Це важливо, коли кількість ітерацій та номер ітерації циклу на якій буде знайдено число, що задовольняє умові алгоритму на кроці 4, не має ніякої залежності – воно може бути будь-яким. Можна припустити, що при такій невеликій кількості ітерацій циклу сучасні процесори не задіють механізм передбачення розгалужень, а мала кількість тактів циклу, який не завантажує конвеєр на повну не суттєво знизить IPC функції ентропійного декодування od_ec_decode_cdf_q15 (назва функції із реалізації алгоритму ентропійного кодування в libaom). І навпаки, при великій кількості ітерацій циклу, така не прогнозованість створює додаткове навантаження на механізм передбачень процесорної системи. В такій ситуації слід переконфігурувати роботу алгоритму ентропійного кодування відео у енкодері для максимізації утилізації можливостей архітектури під час декодування відео (це проблематика іншого дослідження, що не порушується в даній роботі).

Перевагою алгоритму декодування на базі AVX512 відносно AVX2 є абсолютною у всіх випадках, оскільки кількість менша інструкцій при використанні алгоритму на базі SIMDAVX-512 суттєво менша (табл. 2). Перевагою AVX-512 відносно скалярного алгоритму є також кількість менша інструкцій навіть із врахуванням довжини та ширини конвеєра – якщо умовно приблизно поділити кількість операцій на висоту суперскалярного конвеєра 5, то кількість операцій буде менша.

Недоліки мікроархітектури із розширеним набором інструкцій AVX512 та шляхи їх вирішення

Під час виконання інструкцій із розширеного набору SIMDAVX-512 можливе зниження тактової частоти процесорів Intel. Частота відновлюється через певний час (1000, 2000 мікросекунд в залежності від інструкції) після виконання останньої секції коду із такими інструкціями для недопущення надмірної кількості змін частот (коливань). Через суттєвий час дії сповільнення частоти, використання інструкцій розширеного набору AVX-512 може сповільнити виконання інших інструкцій у цій секції коду та роботу процесора в цілому. Наприклад, продуктивність коду веб-серверів знижується на 10%, якщо бібліотека SSL використовує інструкції AVX-512 [9]. Коливання частоти процесорної системи не можливо прогнозувати під час

розробки програмного забезпечення, оскільки вплив використання інструкцій AVX-512 на продуктивність коду залежить від конкретного робочого навантаження.

Великі векторні інструкції SIMD можуть споживати більше енергії на ядро, причому збільшення потужності зазвичай пропорційне збільшенню продуктивності. Отже, використання більшої потужності зазвичай призводить до більшої продуктивності.

Вибрана частота залежить від кількох факторів, включаючи суміш інструкцій, що виконуються на ядрі, і кількість інших ядер, які виконують подібну суміш інструкцій. Суміш інструкцій складається з типу, ширини та кількості інструкцій, включаючи векторні інструкції, які виконуються протягом певного періоду часу. Параметри цих факторів різні на різних мікроархітектурах процесорів Intel.

На максимальну досяжну частоту ядра Intel® Turbo Boost (P0n) впливає категорія інструкцій, що виконуються на цьому ядрі, і кількість ядер, що поділяють цю категорію. Чим більше активних ядер у спільній категорії, тим нижча частота цих ядер. Ці категорії називаються рівнями потужності, описані в роботі [6] та наведені у табл. 4.

Таблиця 4. Максимальні рівні частоти ядра процесорів Intel

Рівень споживання	Частота	Набір інструкцій SIMD
0 Intel® AVX2 Light ((«Легкі AVX2»)	Максимальна	Скалярні, SSE: усі цілочисельні інструкції та з плаваючою крапкою, AVX2: цілочисельні інструкції,крім для AVX2: цілочисельне множення й fused-multiply-add (FMA) інструкції
1 Intel® AVX2 Heavy («Важкі AVX2») та AVX-512 Light («Легкі AVX-512»)	Залежить від покоління та серійного номеру процесору	AVX2 цілочисельне множення, FMA, інструкції чисел з плаваючою крапкою AVX-512 цілочисельні інструкції, крім AVX-512 цілочисельного множення та FMA інструкцій
2 Intel® AVX512 Heavy («Важкі AVX512»)	Знижена частота відносно до Рівня споживання 1	AVX-512 цілочисельне множення, FMA, інструкції чисел з плаваючою крапкою

У процесорі Intel® Xeon® Scalable є три рівні потужності, які ЦП може застосовувати в сценаріях без обмеження потужності. Коли потужність обмежена, мікропроцесор також може регулювати частоту ядер, щоб залишатися в межах заявленого ліміту TDP (Thermal Design Power).

Якщо виконується цілочисельна інструкція Intel® AVX-512, через лічильники продуктивності процесор "розуміє", що потрібен новий рівень потужності AVX-512 Light, що призводить до блокування ядра на 10-20 мкс, доки не буде застосована нова частота, інструкції AVX-512 виконуються з перервами до t+2000 мкс, після чого робоче навантаження процесору повертається до встановленого за замовчуванням, ядро знову блокується на 10-20 мкс, і переводить частоту у рівень за замовчуванням. *Лічильники продуктивності ядра Skylake-X містять подію під назвою "CORE_POWER" (EventSelect=0x28) із полями Umask для вибору "LVL[012]_TURBO_LICENSE". Ці лічильники підраховують кількість циклів, що відбуваються на кожному рівні частоти. Подія продуктивності CORE POWER.LVL[012]_TURBO LICENSE відповідно для скалярного коду [.LVL0], коду AVX2 [.LVL1] і коду AVX-512[.LVL2]. Лічильники містять подію CORE POWER.THROTTLE, що рахує цикли зі продуктивністю, що знизилась під час запиту (блокування) ліцензії на потужність [2].

Готшлаг М., Белоза Ф. запропонували один із підходів для вирішення проблеми зниження частот: розподіл коду і подальша спеціалізація ядра на виконання або скалярного, або коду із розширеними інструкціями набору AVX-512 [4]. Частота, однак, зменшується, лише якщо

принаймні приблизно одна інструкція відповідного типу виконується за цикл [5] або якщо виконується досить щільна суміш інструкцій з двох різних категорій [5]. Це зниження частоти впливає на кожне ядро окремо в залежності від навантаження на конкретне ядро [6].

Усі використані в даному алгоритмі інструкції із набору SIMDAVX-512 відноситься до CPU_POWER.LVL1_LICENSE (табл. 4), тому не створюють запити на зміну частоти у відмінні від інструкцій з набору SIMD AVX2 (код буде працювати на частоті процесора на 10% нижчій у порівняні із скалярною – при цьому блокування ядра буде здійснено один раз на початку декодування відео (чим менше разів – тим краще, блокування ядра триває 20 мкс, і встановлюється на період 2000 мкс, час блокування ядра поновлюється без затримки на повторне блокування до його закінчення – тобто алгоритм ентропійного кодування постійно поновлює рахунок).

Однак, під час тестування на мікроархітектурі Skylake-X процесорної системи Intel Xeon 3 Gold система працювала на частоті 2,0 ГГц, у порівнянні із 2,0 ГГц при виконанні скалярного коду (без змін). Це пояснюється тим, що, по-перше, майже всі інші функції відеокодеку AV1 із функцій не пов'язаних з ентропійного кодуванням за замовчуванням працюють із застосуванням розширеного набору інструкцій AVX2 (оскільки цей SIMD набір підтримують майже всі сучасні процесорні системи), по-друге – такі інструкції мають рівень потужності як «легкі» інструкції AVX-512. Час декодування відео збірки на основі спеціалізованої версії коду із SIMD інструкціями для, наприклад, функцій дискретного косинусного перетворення, навіть на зниженій частоті у декілька разів менше за скалярний (дані функції займають відносно більше процесорного часу під час декодування відео).

Висновки

За допомогою розширеного набору інструкцій AVX-512 можна зменшити кількість процесорного часу та кількість інструкцій програми ентропійного декодування відео AV1 відносно існуючих версій на базі SIMDAVX2 та скалярною версії. Перевага над використанням AVX2 є стійкою, а перевага над скалярною версією залежить від величини декодованого символу та розміру вхідного масиву ймовірностей до функції ентропійного декодування, але в середньому склала ~4%.

На різних процесорах, як правило в залежності від покоління та випуску мікроархітектури, можуть бути присутні ефекти зниження продуктивності коду через зменшення тактової частоти роботи усієї мультипроцесорної системи. Тому варто роздподіляти код між різними ядрами в залежності від використання інструкцій, що належать до «важких» та «легких» груп споживання потужності.

Під час тестування на мікроархітектурі Skylake-Хентропійного кодування відео AV1 ефект зниження частоти відсутній – оскільки багато коду у відеокодеку AV1 примусово працює із застосуванням розширеного набору інструкцій AVX2, що мають таку ж потужність як і «легкі» інструкції AVX-512, що не призводить до змін та зниження тактової частоти процесорної системи.

Перспективами подальших досліджень є програмна реалізація інших модулів (частин) відео-кодеку AV1 із застосуванням розширеного набору інструкцій AVX-512 та дослідження їх роботи.

Література

1. *Videolan.* Репозиторій dav1d. [Електронний ресурс]. – Режим доступу: https://code.videolan.org/videolan/dav1d.

2. *Gottschlag, Mathias et al*. Fair Scheduling for AVX2 and AVX-512 Workloads. USENIX Annual Technical Conference (2021). – P. 745-758.

3. *Teh, J*. Hadarmard transform and sum of absolute difference improvement on high efficiency video coding using intel advanced vector extension-512. – 2018. – 22 p.

4. *Gottschlag, Mathias and Frank Bellosa*. Mechanism to Mitigate AVX-Induced Frequency Reduction. – 2018. – 12 p.

5. *Lemire D.*. Avx-512: when and how to use these new instructions. [Електронний

ресурс]. – Режим доступу: https://lemire.me /blog/2018/09/07/avx512-when-and-how-to-use-these-newinstructions/.

6. *Kinsella R., MacNamara C., G. Tkachuk G.* TECHNOLOGY GUIDE Intel Corporation. Intel® AVX-512 – Instruction Set for Packet Processing. [Електронний ресурс]. – Режим доступу: https://builders. intel.com/docs/networkbuilders/intel-avx-512-instruction-set-for-packet-processing-technology-guide-1617440657.pdf

7. *Google.* Репозиторій libaom. [Електронний ресурс]. – Режим доступу: https://aomedia.googlesource.com/aom/

8. *Han, Jingning, Bohan Li, Debargha Mukherjee, Chiang Ching-Han, Cheng Chen, Hui Su, Sarah Parker, Urvang Joshi, Yue-Meng Chen, Yunqing Wang, Paul Wilkins, Yaowu Xu and Jim Bankoski*. A Technical Overview of AV1. *Proceedings of the IEEE* 109 (2021). – P. 1435-1462.

9. *Stackoverflow*. AVX2 what is the most efficient way to pack left based on a mask. [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions /36932240/avx2-what-is-the-most-efficient-way-to-pack-left-based-on-a-mask

Бойко Т.П., Русанова О.В.

СПОСІБ ЕНТРОПІЙНОГО КОДУВАННЯ ВІДЕО НА БАЗІ РОЗШИРЕНОГО НАБОРУ ІНСТРУКЦІЙ SIMD AVX-512

Метою даної роботи є зменшення часу ентропійного кодування відео з використанням можливостей процесорів із розширеним набором інструкцій типу AVX-512 за рахунок розпаралелювання і використання додаткових SIMD інструкцій у порівнянні з AVX2 та SSE.

У роботі досліджуються алгоритм ентропійного декодування відео AV1, як існуюча скалярна версія, так і векторизована версія на базі SIMD SSE та AVX2. Проаналізовані недоліки наведених алгоритмів, які призводять до зайвих витрат часу процесорів і як наслідок до зниження їх продуктивності.

Дані версії не використовують усі можливостей сучасної мікроархітектури із підтримкою набору SIMD AVX-512. У роботі показано, що за допомогою набору інструкцій SIMD AVX-512 можна ефективно вирішити проблеми, зокрема лівостороннього пакування, та зменшити кількість векторних та загального призначення регістрів, що використовує програма, уникнути зайвих спеціалізацій функцій, що потребує додаткових витрат ресурсів, а саме стек-памяті.

Наведені позитивні результати, отримані під час тестування запропонованого способу ентропійного декодування на базі SIMD AVX-512 у порівнянні зі скалярною та векторизованою версіями SIMD SSE та AVX2. Розглянуті недоліки мікроархітектури із розширеним набором інструкцій AVX512 та шляхи їх вирішення. Вказано, що наведені недоліки можливо виправити при виборі правильних параметрів ентропійного енкодування. Відносне прискорення частини ентропійного декодування, що відповідає за оновлення ймовірностей працює значно швидше, оскільки не залежить від значень декодованого символу.

Ключові слова: *SIMD, AV1, AVX-512, ентропійне кодування, відео, кодек, стиснення відео.*

Boyko T.P., Rusanova O.V.

A WAY OF ENTROPY VIDEO CODING BASED ON EXTENDED INSTRUCTION SET SIMD AVX-512

The purpose of this work is to reduce the time of entropy coding of video using the capabilities of processors with an extended instruction set of the AVX-512 type due to parallelization and the use of additional SIMD instructions compared to AVX2 and SSE. The paper investigates the AV1 video entropy decoding algorithm, both the existing scalar version and the vectorized version based on SIMD SSE and AVX2. The disadvantages of the above algorithms are analyzed, which lead to additional run time of processors and, as a result, to a decrease of their performance.

These versions do not utilize all the capabilities of the modern microarchitecture with support for the AVX-512 SIMD set. In this paper we show that by means of the SIMD AVX-512 instruction set, it is possible to effectively solve problems, in particular, left-hand packing, and reduce the number of vector and general-purpose registers used by the program, avoid unnecessary specialization of functions, which requires additional resource consumption, for instance, stack memory.

The positive results obtained during testing of the proposed method of entropy decoding based on SIMD AVX-512 in comparison with the scalar and vectorized versions of SIMD SSE and AVX2 are given. Disadvantages of microarchitecture with extended AVX512 instruction set and ways to solve them are considered. It is indicated that the above shortcomings can be corrected by choosing the correct entropy encoding parameters. The relative acceleration of the part of entropy decoding, which is responsible for updating the probabilities, works much faster, because it does not depend on the values of the decoded symbol.

Keywords: *SIMD, AV1, AVX-512, entropy coding, video, CODEC, video compression*.