DOI: https://doi.org/10.36910/6775-2524-0560-2023-53-20
УДК 004.92
Курилко Максим Ігорович, бакалавр
https://orcid.org/0009-0008-3054-7329
Марченко Олександр Іванович, к.т.н., доцент
https://orcid.org/0000-0002-4537-3420
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна

АРХІТЕКТУРА ЕНКОДЕРА ДЛЯ ОПТИМІЗАЦІЇ ПРОЦЕСУ СТИСНЕННЯ РАСТРОВИХ ЗОБРАЖЕНЬ У ФОРМАТ JPEG

Курилко М.І., Марченко О.І. Архітектура енкодера для оптимізації процесу стиснення растрових зображень у формат JPEG. У даній статті запропонована архітектура енкодера зображень у формат JPEG, яка дозволяє оптимізувати процес стиснення зображення, базується на ідеї модульності, використовує конвеєризацію, що дозволяє гнучко налаштовувати процес стиснення під використання доступних апаратних пристроїв для досягнення потрібних вимог по швидкодії та енергоспоживанню пристроїв обробки зображень, що розроблюються.
Ключові слова: стиснення зображень, JPEG енкодер, вбудовані системи, пристрої низького споживання, растрові зображення, апаратне прискорення.

Kurylko M., Marchenko O. Encoder architecture for optimization of raster images compression to the JPEG format. This article proposes an image encoder architecture to the JPEG format, which allows optimization of image compressing process, is based on the concept of modularity, utilizes pipelining, allowing flexible adjustment of the compression process to accommodate available hardware accelerations. This approach aims to meet specific performance and energy consumption requirements in image processing devices.
Keywords: image compression, JPEG encoder, embedded systems, low-power devices, raster graphics, hardware acceleration.

Постановка проблеми. При проектуванні пристроїв обробки зображень виникає необхідність вибору формату зображень та способу їх стиснення для збереження та передачі на інші пристрої. Від правильності такого вибору залежить не тільки якість представлення зображення, але і споживання енергії пристроєм, обсяг необхідної пам'яті та швидкість передачі зображень по мережі. Крім того, такий вибір впливає також на ціну остаточного програмно-апаратного рішення. Для досягнення низького енергоспоживання кінцевими пристроями, для обробки зображень важливо використовувати пристрої низького енергоспоживання (Low Power Devices – LPD), які знаходять застосування практично в усіх галузях виробництва та побуту. Наприклад, пристрої LPD використовуються для рішення задач створення та/або обробки растрових зображень в таких системах як метеостанції, системи відео/фото спостереження, дрони. Різноманіття задач, для рішення яких можуть використовуватися подібні пристрої LPD, і платформ, на базі яких можуть бути створені результуючі системи, а також обмеження на фізичні габарити та ціну пристроїв, призводять до необхідності як підлаштовувати апаратно-програмну базу стиснення зображень під кожну конкретну задачу, так і шукати компроміси між заданими характеристиками цих пристроїв. Вибір формату зображень, архітектури енкодера і способів стиснення зображень є критично важливими для оптимізації процесу стиснення та реалізації ефективних та надійно працюючих пристроїв обробки зображень. Тому розробка архітектури енкодера, яка дозволяє гнучко і оптимізовано підлаштовувати етапи стиснення зображення під задані критерії (швидкодія, енергоефективність, вартість) та використання потрібних апаратних пристроїв є актуальною задачею.

Термінологія.

DCT – Discrete Cosine Transform – вид перетворення Фур'є, що у контексті стиснення зображень дозволяє переходити від просторового до частотного представлення пікселів вхідного зображення.

RGB – колірна модель, у якій колір кодується градаціями складових трьох каналів: червоний, зелений та синій.

MCU – Minimum Coded Unit – ізольований сектор (зазвичай 8x8 пікселів) вхідного зображення, до якого застосовується DCT та деякі інші операції під час стиснення зображення JPEG кодеком.

SIMD – Single Instruction / Multiple Data – властивість процесора виконувати одну інструкцію для паралельної роботи з кількома елементами даних.

SoC – System on a Chip – електронна схема, що має складові та функції цілого пристрою (наприклад комп'ютера) та розміщена на одній інтегральній схемі.

DSP – Digital Signal Processor – спеціалізований мікропроцесор, призначений для цифрової обробки сигналів (зазвичай, у режимі реального часу).

FPGA – Field-Programmable Gate Array – програмована користувачем вентильна матриця.

Аналіз останніх досліджень і публікацій.

Формат JPEG має наступні важливі властивості:

1. JPEG дозволяє досягти балансу між якістю зображення та розміром файлу завдяки методу стиснення з втратами, що дозволяє отримувати файл з меншим розміром при збереженні прийнятної якості зображення.

2. JPEG дозволяє гнучко налаштовувати параметри стиснення, такі як ступінь стиснення, таблиці кодування, параметри кольорових каналів та інші, що дає можливість оптимізувати стиснення зображення для конкретних потреб та вимог.

3. JPEG підтримується більшістю переглядачів зображень, веб-браузерів та операційних систем, що забезпечує широкі можливості для відображення та використання зображень.

4. JPEG підтримується добре відомим та широко використаним стандартом, який проходив ретельне тестування та оптимізацію протягом багатьох років, що гарантує його стабільність та надійність.

5. Існує багато інструментів, бібліотек та ресурсів, які спрощують роботу з форматом JPEG, що сприяє ефективній реалізації та адаптації алгоритмів стиснення для конкретних потреб.

6. Стандарт JPEG має міжнародний статус, що дозволяє використовувати його у різних додатках та сервісах без обмежень.

У роботі [1] описано і зроблено порівняння деяких архітектур побудови JPEG енкодерів. В цій статті пропонується варіант реалізації найшвидшого енкодера, але такі характеристики розробки, як ціна пристрою та енергоспоживання, залишаються поза межами розгляду в ній. У роботі [2] описаний спосіб реалізації DCT етапу стиснення на спеціалізованому SoC, проте спосіб комунікації із головним процесором та його програмна частина залишаються невизначеними. У роботі [3] міститься детальний опис всіх етапів JPEG стиснення, а також описані варіанти оптимізації кожного з етапів стиснення за допомогою використання DSP процесорів, проте в цій роботі не наведені отримані характеристики швидкодії та енергоспоживання. У роботах [4], [5], [6] описані варіанти побудови JPEG енкодерів, які базуються на використанні виключно FPGA, проте наведені в них результати практичних експериментів показують зависоке енергоспоживання таких пристроїв, що призводить до недоцільності використання такого підходу для розробки пристроїв та систем обробки зображень, де економне споживання енергії є ключовим критерієм.

Звідси витікає необхідність пошуку розробки нових архітектур JPEG енкодерів, які будуть ґрунтуватися на гібридних програмно-апаратних рішеннях з використанням LPD пристроїв, що дозволить знаходити компроміси при оптимізації процесу стиснення зображень за такими критеріями як швидкодія, вартість та енергоспоживання отриманих JPEG енкодерів.

Перед подальшим аналізом існуючих архітектур енкодерів розглянемо етапи кодування (стиснення) у форматі JPEG (рис. 1).

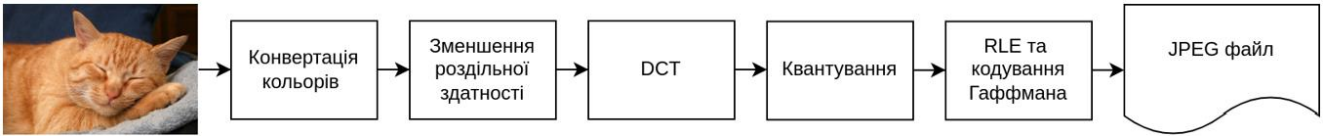

Рис. 1. Етапи кодування (стиснення) у форматі JPEG

Стандарт JPEG [7] регламентує послідовне використання кількох алгоритмів стиснення, приймаючи на вхід початкове растрове зображення (послідовний потік пікселів, де інформація про кожен піксель має фіксований розмір та формат). Стандарт JPEG також тісно пов'язаний із JFIF

стандартом [8][9], що регламентує структуру та формат вихідного файлу. На першому кроці, у разі необхідності, зображення перетворюється з RGB (червоний, зелений, синій) колірного простору в YCbCr (YUV) колірний простір, де Y відповідає за яскравість, а Cb та Cr за кольорові відтінки. Цей крок може бути пропущений, якщо кольоровий простір вхідного зображення співпадає з необхідним. Конвертація кольорів необхідна для виконання наступного кроку, на якому зменшується дискретизації (Downsampling) зображення. Оскільки людське око є недостатньо чутливим до всіх відтінків кольору, інформація про кольорові компоненти (Cb і Cr) може бути зменшена (відкинута деяка частина цієї інформації), що веде до зниження розміру даних без значних втрат візуальної якості. Зазвичай, Cb та Cr компоненти зменшуються вдвічі у порівнянні з вхідним зображенням. Наступним кроком для кожного з трьох колірних каналів виконується DCT (Discrete Cosine Transform). Канал поділяється на блоки (зазвичай 8x8 пікселів), які називають MCU (мінімальна одиниця кодування) і на кожному з цих блоків застосовується перетворення, яке виконує перехід від просторового представлення даних про пікселі у частотне. Це дозволяє групувати інформацію про зображення та виділяти важливі частотні компоненти. Наступним кроком є квантування (Quantization). На цьому кроці визначається яка частотна інформація буде видалена, а яка буде залишена. Цей процес зменшує точність частотних коефіцієнтів, отриманих з DCT. Високочастотні коефіцієнти, як правило, зменшуються більше, ніж низькочастотні, оскільки зміни на високих частотах менш помітні для людського ока. Це один з ключових кроків, де відбувається відкидання певної частини даних, що і забезпечує власне стиснення зображення. Після квантування дані серіалізуються методом зигзагоподібного сканування блоків (це підвищує ефективність подальшого кодування) і потім стискаються за допомогою безвтратних алгоритмів стиснення, таких як RLE (Run-Length Encoding), який кодує послідовності однакових елементів як один елемент і його довжину, та кодування Гаффмана, яке присвоює коротші коди елементам, які зустрічаються частіше. Під час декодування зображення всі кроки виконуються у зворотному порядку.

Тепер, для кращого розуміння запропонованої нижче архітектури, розглянемо архітектури існуючих енкодерів (рис. 2).

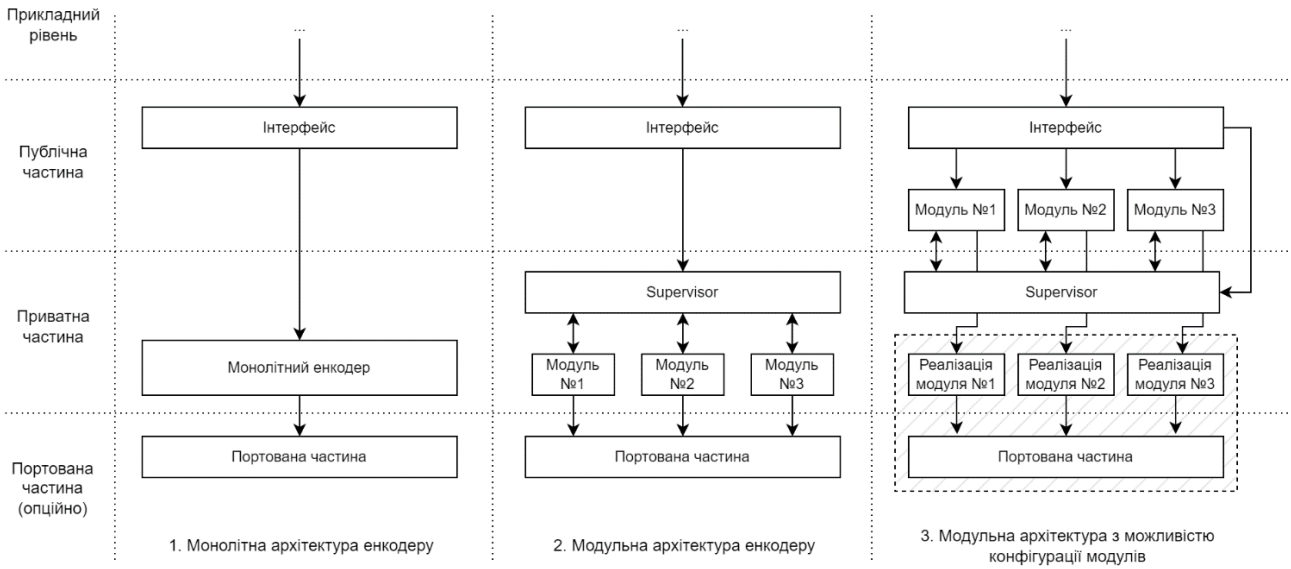

Рис. 2. Архітектури енкодерів

1. Енкодер з монолітною архітектурою. Таку архітектуру мають, як правило, старіші або більш прості енкодери, які є цілісними програмними рішеннями, що виконують всі кроки JPEG стиснення в одному модулі. Інтерфейс таких енкодерів надає доступ лише до високорівневих функцій керування та до базової конфігурації. Такі енкодери можуть бути швидшими за інші, оскільки мають тіснішу інтеграцію компонентів, але вони є менш гнучкими та їх важче підтримувати, розширювати та оптимізовувати під конкретну платформу. Прикладами енкодерів із монолітною архітектурою є JPEGENC [10] та jpge.cpp [11].

2. Енкодер з модульною архітектурою. Архітектура такого енкодера складається з множини модулів, кожен з яких виконує свою частину процесу кодування. Такі енкодери, як правило, мають також окремий модуль, що виконує функції супервайзера, який містить інформацію про поточний стан процесу стиснення, забезпечує міжмодульну взаємодію та може додатково проводити конвертацію даних до більш зручного для користувача представлення. Інтерфейс таких енкодерів надає доступ лише до високорівневих функцій без можливості конфігурації чи модифікації модулів. Прикладами енкодерів з модульною архітектурою є libjpeg [12] та turbo-jpeg [13].
3. Енкодер з модульною архітектурою і можливістю конфігурування модулів. Архітектура таких енкодерів також є модульною, але, на відміну від попередньої архітектури, дозволяє індивідуальну конфігурацію кожного модуля за допомогою публічного API. Така архітектура є найбільш гнучкою та може включати зміну параметрів кодування, повну заміну реалізації модулів (ця частина позначена на рисунку штрихованим заповненням), та навіть динамічне налаштування процесу кодування. Модуль-супервайзер, як і в попередньому випадку, відповідає за міжмодульну взаємодію.

В порівнянні з монолітною архітектурою, обидві модульні архітектури забезпечують більшу гнучкість і зручність для виконання оптимізації енкодера при переході на нову апаратну платформу, оскільки кожен модуль можна розробляти, тестувати та замінювати незалежно від інших. Крім того, модульна реалізація, як правило, не має значного впливу на швидкодію енкодера та на обсяг необхідної для його роботи пам'яті. Зазначені вище властивості модульної архітектури з можливістю конфігурування модулів визначають її найкращу пристосованість до виконання оптимізації процесу стиснення зображення з орієнтацією на різні критерії оптимізації та використання різних апаратних пристроїв, зокрема LPD пристроїв.

Запропонована архітектура енкодера.

Запропонована архітектура (рис. 3) побудована за модульним принципом з можливістю конфігурації модулів за допомогою публічного API і відрізняється від існуючих використанням оптимізованого набору із 6 основних та 4 допоміжних модулів, має гнучку систему міжмодульної синхронізації та підтримує конвеєризацію виконання етапів кодування (стиснення). Кожен основний модуль приймає дані від попереднього модуля, виконує конкретний етап JPEG кодування та передає результат своєї роботи наступному.

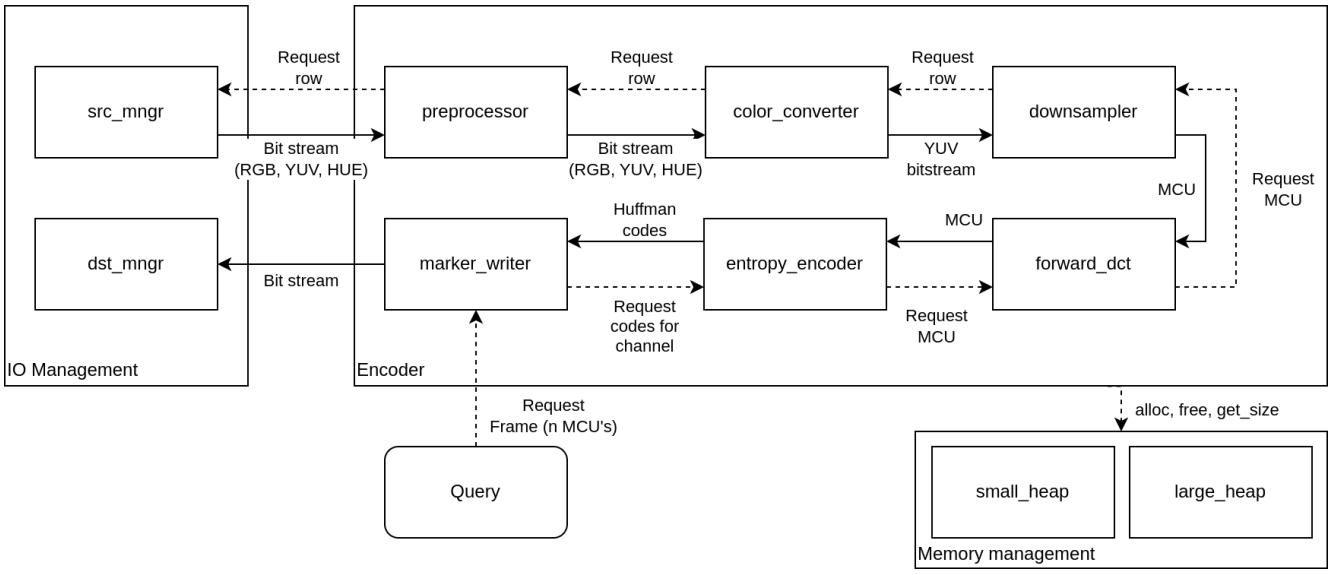

Рис. 3. Запропонована архітектура JPEG енкодера

- preprocessor – модуль, що виконує розрахунок або вибір коефіцієнтів для стиснення, зміну формату вхідних пікселів тощо;
- marker_writer – модуль, що виконує форматування вихідного файлу згідно з JFIF стандартом;

- color_converter – модуль, що виконує конвертацію кольорів із вхідного RGB колірного простору в YCbCr колірний простір;
- downsampler – модуль, що виконує зменшення розміру кольорових компонентів та поділ зображення на MCU;
- forward_dct – модуль, що виконує DCT над кожним MCU;
- entropy_encoder – модуль, що виконує квантування коефіцієнтів, кодування Гаффмана та кодування довжин серій нулів (RLE).

Призначення чотирьох додаткових модулів запропонованої архітектури:

- модулі small_heap та large_heap реалізують інтерфейс для всіх операцій виділення та переміщення пам'яті, що використовується поточним енкодером; small_heap використовується для операцій з пам'яттю менше 1 Кб, наприклад, операцій з локальними змінними модулів, хеш-таблицями, а large_heap використовується для операцій з пам'яттю більше 1 Кб, наприклад, операцій з буферами каналів, таблицями коефіцієнтів, тощо;
- модуль src_mngr використовується для доступу та позиціювання у вхідних даних та містить інформацію про кількість доступних байт для читання;
- модуль dst_mngr використовується для запису та позиціювання у вихідних даних, та містить інформацію про загальну кількість записаних даних та кількість байтів, що доступні для запису.

Розглянемо можливості виконання оптимізації процесу кодування (стиснення) зображень формату JPEG на основі запропонованої архітектури. Ця архітектура відрізняється від існуючих рішень, меншою кількістю та повною ізольованістю модулів один від одного, оскільки взаємодія між ними відбувається тільки через модуль-супервайзер. Крім того, в цій архітектурі було вирішено об'єднати деякі етапи процесу JPEG кодування (наприклад, в модулях downsampler та entropy_encoder), на яких вирішуються підзадачі, що потребують щільної взаємодії між собою з передачею великої кількості даних між ними. Тому, реалізація цих підзадач окремими модулями виглядає недоцільною. Таким чином, запропонована архітектура, з однієї сторони, зменшує навантаження на модуль-супервайзер, оскільки модулів менше, а з другої сторони, має достатню кількість модулів для виконання оптимізації окремих етапів стиснення зображення незалежно від інших етапів.

Кожен модуль виконує покладені на нього функції процесу стиснення зображення або за запитом від наступного модулю, тобто виконується так-звана ланцюгова синхронізація між модулями, або без запитів у режимі реального часу. Наприклад, у разі використання ланцюгової синхронізації, ініціюючим сигналом модуля marker_writer буде запит від користувача на виконання деякої частини роботи по стисненню вхідного зображення. В режимі реального часу розподіл даних між модулями та контроль за нестачею або надлишком даних (overrun, underrun) виконує супервайзер, і використання цього режиму дозволяє будувати гнучкі системи із асинхронною роботою деяких модулів. Запропонована архітектура надає можливість організації роботи енкодера у режимі реального часу, чим відрізняється від існуючих, в яких пропонується лише ланцюгова синхронізація [12]. Крім того, в запропонованій реалізації є також можливість вказувати кількість піксельних рядків зображення, обробку яких потрібно виконати за один виклик функцій обробки, на відміну від існуючих рішень, в яких оброблюється тільки одразу все зображення в цілому [11].

Якщо при реалізації роботи модулів використовувати багатопоточність або декілька процесорів, то можна конвеєризувати процеси стиснення декількох зображень по етапах обробки. Конвеєризація дозволяє на кожному з етапів кодування починати обробку наступного зображення як тільки обробка попереднього зображення на цьому етапі буде закінчена, що значно прискорює процес стиснення зображень. Приклад такої конвеєризації обробки чотирьох зображень на різних етапах стиснення показаний на рис. 4.

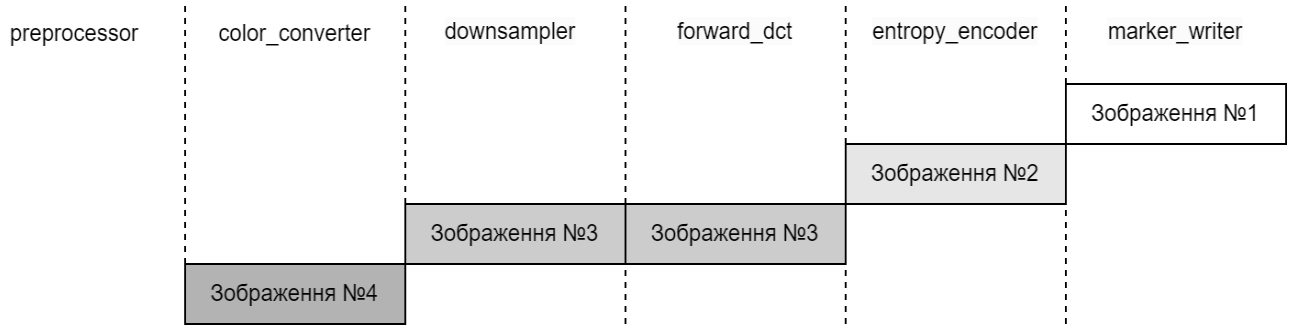

Рис. 4. Приклад конвеєризації стиснення декількох зображень

Оскільки в існуючих архітектурних рішеннях конвеєризація при стисненні одночасно декількох зображень не використовується, то можливість використання такої конвеєризації у запропонованій архітектурі є одним із потужних аспектів, які вона надає для оптимізації стиснення зображень порівняно з існуючими рішеннями.

Реалізація енкодера згідно із запропонованою архітектурою має також і найпростіший режим виконання процесу стиснення зображення, у якому всі модулі реалізовані програмно за класичними алгоритмами (наприклад, AAN для DCT [14]). Цей режим працює за умовчанням.

Висновки

Запропонована архітектура енкодера надає можливість розробнику реалізувати свої власні модулі замість стандартних модулів з метою оптимізації процесу стиснення растрових зображень у формат JPEG. Реалізація оптимізованих модулів може бути як чисто програмною, так і на основі спеціалізованих апаратних пристроїв, в тому числі й пристроїв LPD, а також з використанням конвеєризації обробки декількох зображень на різних етапах процесу стиснення.

В рамках даної архітектури енкодера інтеграція апаратних пристроїв відбувається досить легко та швидко, що забезпечує гарний рівень оптимізації кінцевого пристрою, в якому виконується обробка зображень, за необхідним співвідношенням показників швидкості кодування, енергоспоживання та вартості отриманого пристрою. Зокрема, якщо основним критерієм оптимізації є енергоспоживання, то на таких етапах стиснення, як дискретне косинусне перетворення або конвертація кольорів, доцільно використовувати пристрої LPD. Ще однією важливою властивістю запропонованої архітектури є те, що завдяки можливості гнучкого конфігурування через публічний API, вона дозволяє простіше виконувати роботу з різними версіями проєкту, надає можливість виправляти помилки одразу для всіх проектів та підвищує переносимість проектів, оскільки API залишається загальним для всіх проектів. Крім того, використання такої архітектури зменшує загальний обсяг виконуваної роботи та понижує рівень кваліфікації розробників, який є необхідним для виконання оптимізації процесу стиснення зображення, оскільки модуль-супервайзер може виконувати значну кількість допоміжної роботи, такої як конвертація даних, розподілення задач, кешування, фокусуючи увагу розробників саме на процесі оптимізації.

Список бібліографічного опису

1. Kazuo Sakiyama. Finding the best System Design Flow for a High-Speed JPEG Encoder [Електронний ресурс] / Kazuo Sakiyama, Patrick R. Schaumont, Ingrid M. Verbauwhede. – 2003. – Режим доступу до ресурсу: https://web.archive.org/web/20170705130129/http://www.seas.ucla.edu/%7Eingrid/Publications/2003aspdac.pdf
2. Bryan Chan Jia Ching. Implementation of an 8x8 Discrete Cosine Transform on Programmable System-on-chip [Електронний ресурс] / Bryan Chan Jia Ching, Ab Al-Hadi Ab Rahman, Nabihah Ahmad. – 2018. – Режим доступу до ресурсу: https://iopscience.iop.org/article/10.1088/1742-6596/1049/1/012084/pdf
3. Mikael Andersson. Parallel JPEG Processing with a Hardware Accelerated DSP Processor [Електронний ресурс] / Mikael Andersson, Per Karlstr¨om. – 2004. – Режим доступу до ресурсу: https://www.diva-portal.org/smash/get/diva2:19951/FULLTEXT01.pdf
4. Tian-Yang Li. An FPGA-based JPEG preprocessing accelerator for imageclassification [Електронний ресурс] / Tian-Yang Li, Fan Zhang, Wei Guo, Jian-Liang Shen, Ming-Qian Sun. – 2022. – Режим доступу до ресурсу: https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/tje2.12174
5. Scavongelli C. FPGA implementation of JPEG encoder architectures for wireless networks [Електронний ресурс] / C. Scavongelli, M. Conti . – 2017. – Режим доступу до ресурсу: https://jes-eurasipjournals.springeropen.com/articles/10.1186/s13639-016-0047-5

6. George Gabriel Kyrtsakas. An FPGA Implementation of a Custom JPEG Image Decoder SoC Module [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://scholar.uwindsor.ca/cgi/viewcontent.cgi?referer=&httpsredir=1&article=6947&context=etd
7. ISO/IEC 10918-1:1994 - Digital compression and coding of continuous-tone still images: Requirements and guidelines. [Електронний ресурс] – 1994. – Режим доступу до ресурсу: https://www.iso.org/standard/18902.html
8. Eric Hamilton. JPEG File Interchange Format. Version 1.02 [Електронний ресурс]. – 1992. – Режим доступу до ресурсу: https://www.w3.org/Graphics/JPEG/jfif3.pdf
9. ISO/IEC 10918-5:2013 - Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF). [Електронний ресурс] – 2013. – Режим доступу до ресурсу: https://www.iso.org/ru/standard/54989.html
10. Larry Bank. Arduino JPEG encoder [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://github.com/bitbank2/JPEGENC
11. Rich Geldreich. C++ JPEG compression/fuzzed low-RAM JPEG decompression codec [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://github.com/richgel999/jpeg-compressor
12. libjpeg веб-сайт [Електронний ресурс]. – 1991. – Режим доступу до ресурсу: https://libjpeg.sourceforge.net/
13. libjpeg-turbo веб-сайт [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://libjpeg-turbo.org/
14. Yukihiro Arai. A Fast DCT-SQ Scheme for Images / Yukihiro Arai, Takeshi Agui and Masayuki Nakajima. IEICE TRANSACTIONS. – 1988. Vol. E7. No. 11. – с. 1095-1097.

References

1. Kazuo Sakiyama , Patrick R. Schaumont and Ingrid M. Verbauwhede. *Finding the best System Design Flow for a High-Speed JPEG Encoder*. 2003. – Available at: https://web.archive.org/web/20170705130129/http://www.seas.ucla.edu/%7Eingrid/Publications/2003aspdac.pdf [Accessed 1 Dec. 2023].
2. Bryan Chan Jia Ching, Ab Al-Hadi Ab Rahman and Nabihah Ahmad. *Implementation of an 8x8 Discrete Cosine Transform on Programmable System-on-chip*. 2018. – Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1049/1/012084/pdf [Accessed 1 Dec. 2023].
3. Andersson, M. (2004). *Parallel JPEG Processing with a Hardware Accelerated DSP Processor Examensarbete utfört i Datorteknik vid Tekniska Högskolan i Linköping av*. [online] Available at: https://www.diva-portal.org/smash/get/diva2:19951/FULLTEXT01.pdf [Accessed 1 Dec. 2023].
4. Li, T., Zhang, F., Guo, W., Shen, J. and Sun, M. An FPGA-based JPEG preprocessing accelerator for image classification. *The Journal of Engineering*, 2022(9), pp.919–927. – Available at: doi:https://doi.org/10.1049/tje2.12174 [Accessed 1 Dec. 2023].
5. Cristiano Scavongelli and Conti, M. FPGA implementation of JPEG encoder architectures for wireless networks. *EURASIP Journal on Embedded Systems*, 2017(1). – Available at: doi:https://doi.org/10.1186/s13639-016-0047-5 [Accessed
1 Dec. 2023].
6. Kyrtsakas, G. *An FPGA Implementation of a Custom JPEG Image Decoder SoC An FPGA Implementation of a Custom JPEG Image Decoder SoC Module Module*. 2017. – Available at: https://scholar.uwindsor.ca/cgi/viewcontent.cgi?referer=&httpsredir=1&article=6947&context=etd [Accessed 1 Dec. 2023].
7. *ISO/IEC 10918-1:1994*. ISO. Available at: https://www.iso.org/standard/18902.html [Accessed 1 Dec. 2023].
8. Hamilton, E. *JPEG File Interchange Format*. 1992. – Available at: https://www.w3.org/Graphics/JPEG/jfif3.pdf [Accessed 1 Dec. 2023].
9. *ISO/IEC 10918-5:2013*. ISO. Available at: https://www.iso.org/ru/standard/54989.html [Accessed 1 Dec. 2023].
10. Bank, L. *bitbank2/JPEGENC*. GitHub. 2023. – Available at: https://github.com/bitbank2/JPEGENC [Accessed 1 Dec. 2023].
11. Geldreich, R. *jpeg-compressor*. GitHub. 2023. – Available at: https://github.com/richgel999/jpeg-compressor [Accessed 1 Dec. 2023].
12. libjpeg.sourceforge.net. (n.d.). *libjpeg*. – Available at: https://libjpeg.sourceforge.net/ [Accessed 1 Dec. 2023].
13. libjpeg-turbo.org. (n.d.). *libjpeg-turbo*. – Available at: https://libjpeg-turbo.org/ [Accessed 1 Dec. 2023].
14. Yukihiro Arai. A Fast DCT-SQ Scheme for Images / Yukihiro Arai, Takeshi Agui and Masayuki Nakajima. IEICE TRANSACTIONS. – 1988. Vol. E7. No. 11. – pp. 1095-1097.