ISSN 2074-7888, Наукові праці Донецького національного технічного університету, серія Проблеми моделювання та автоматизації проектування, 2011, Випуск 10 (197)

УДК 004.932.2

Ю.І. Жиданова (магістр), **О.В. Самощенко** (канд. техн. наук)
Донецький національний технічний університет
samoshchenko@cs.donntu.edu.ua

ДЕКОДУВАННЯ ЗОБРАЖЕНЬ ФОРМАТУ JPEG ЗАСОБАМИ МОВИ VHDL

Пропонується метод зменшення потоку даних за рахунок апаратної реалізації операцій на FPGA, а також наводяться результати досліджень на прикладі загальновідомого формату JPEG.

Ключові слова: зображення, кодування, апаратна реалізація, FPGA.

Вступ

Вентильні матриці, програмовані електричним полем (Field Programmable Gate Array - FPGA), є одним із різновидів сімейства програмованих логічних інтегральних схем (ПЛІС). Розвиток архітектури FPGA пов'язаний з постійним додаванням пристроїв, що спрощують виконання операцій цифрової обробки сигналів, у тому числі множення [1].

Алгоритм JPEG передбачає розбивку зображення на блоки розміром 8x8 елементів, визначення частотних складових в таких фрагментах зображення, скорочення кількості цих складових (збереження тільки істотних) і, нарешті, їх стиск і запис [2].

Алгоритм JPEG складається з наступних етапів: перехід у колірну схему YCrCb, субдискретизація компонентів кольоровості усередненням груп пікселей, застосування дискретного косинусного перетворення для переходу в частотне представлення (ДКП), квантування, зигзаґ-сканування, стиск. В алгоритмі JPEG найбільш ресурсоємні є операції дискретного косинусного перетворення та переведення зображення із формату RGB у формат YCbCr. Таким чином, апаратною реалізацією замінюються ці операції, а саме за допомогою плат із технологією FPGA.

Особливості FPGA

Особливою характеристикою FPGA є використання розподіленої матриці межз'єднань, що дозволяє створювати кристали логічною ємністю до 10 і більше мільйонів логічних вентилів. У дослідженнях приймала участь плата Spartan 3E FG320 XC3S500E, тактова частота якої дорівнює 100МГц, а період одного такту займає 10нс. Плата SPARTAN 3E дає в розпорядження користувачеві внутрішню пам'ять і помножувачі матриць.

Ядра IDCT та YCbCrRGB відносяться до програмних IP-блоків (IntellectualProperty-cores), тобто блоків, які готові для проектування мікросхем і специфіковані на мові опису апаратури VHDL. Кожне ядро представляє собою окрему апаратну реалізацію деякого алгоритму, що необхідно підключати до проекту MicroBlaze. Фірма Xilinx пропонує це робити наступним чином. Відносно входів (виходів) ядра можливе використання більш ніж двох динамічних входів та більш ніж одного виходу, завдяки забезпеченню до 16 шин інтерфейсу FSL (Fast Simplex Link), тобто односпрямованої шини типу «точка-точка», яка використовується для виконання швидкого зв'язку між будь-якими двома елементами дизайну на FPGA. Користувач може використовувати 8 входів для свого ядра та 8 виходів. На рис.1 показана основна ідея з'єднання спеціалізованого ядра через шинний інтерфейс FSL із MicroBlaze.

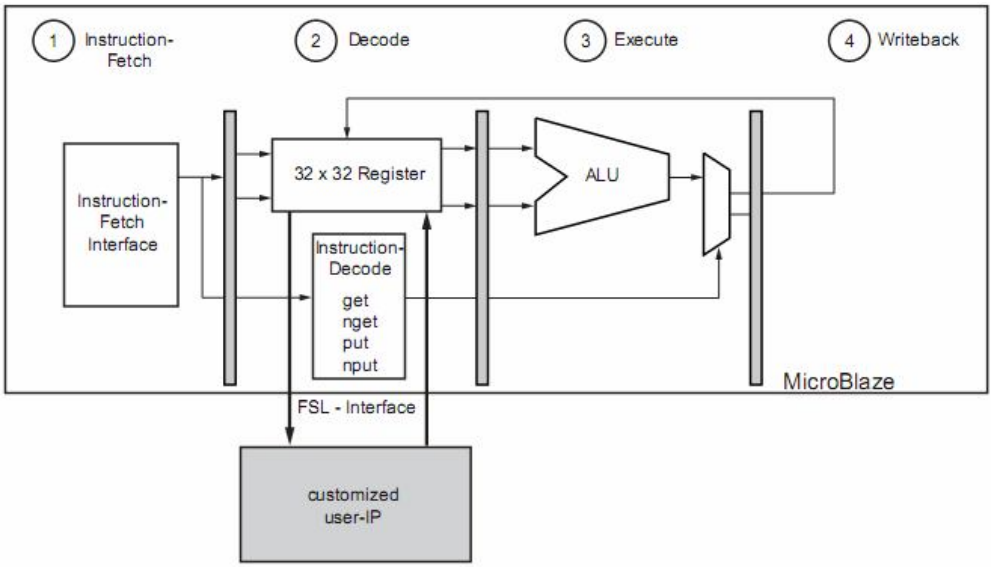

Рисунок 1 – Підключення спеціалізованого ядра через шину FSL до MicroBlaze

Максимальна частота роботи спеціалізованого ядра не буде зменшувати частоту роботи MicroBlaze, тому що воно незалежне і не впливає на внутрішню архітектуру MicroBlaze. Якщо, наприклад, робота ядра займає 100 тактів, щоб обчислити якийсь результат, то MicroBlaze може виконати тим часом інший код програми і не повинен чекати 100 тактів. Інтеграція апаратних частин у програмне забезпечення не потребує вбудованого спеціального коду, тому що інтерфейс FSL визначає макроси для роботи з ядром – відіслати та прийняти дані [3].

Опис роботи ядра IDCT

Загальну схему ядра IDCT приведено на рис.2.

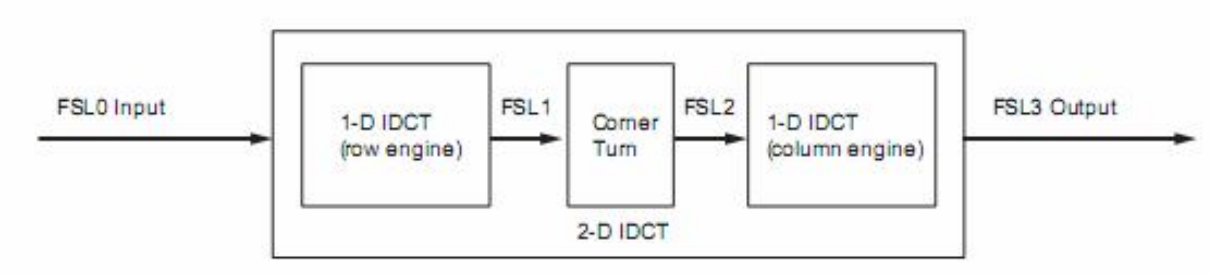

Рисунок 2 – Загальна схема роботи ядра IDCT

Ядро є одномірним зворотним дискретним косинусним перетворенням. Але його просто використовувати для двомірного зворотнього ДКП, виходячи з того, що складається воно із двох частин одномірного зворотнього ДКП та блоку транспонування матриці.

Для інтеграції ядра з програмним забезпеченням використовується програма на мові С. Програма проста і виконує запис деяких даних у ядро та зчитує їх назад. Блок даних, що буде записаний у ядро, складається із 8 вхідних даних. Перед тим як наступний блок даних буде записаний у ядро, MicroBlaze чекає на результуючий блок. Очевидно, вихідний блок даних буде складатися із 8 вихідних значень із IDCT. Для запису даних у ядро використовується спеціально визначена функція.

Спочатку програма записує 8 значень із пам'яті до шини FSL. Ядро отримує коефіцієнти ДКП та повертає частину зображення у блоках, які мають розмір 8x8 чи 16x16. Ядро виконує зворотне ДКП над вхідними блоками за допомогою одномірного перетворення. Коли результат готовий, MicroBlaze зчитує данні (8 слів) із шини FSL.

Ядро IDCT спілкується із зовнішнім світом за допомогою шини FSL. Інтерфейс шини є 32-розрядним. Цей канал може використовуватися для прийому (передачі) даних чи управляючих слів. Окремі біти інформують про тип інформації, яка передається. Продуктивність шини може сягати 300 МБ/сек.

На рис.3 показано вбудований проект MicroBlaze із ядром IDCT та деякою периферією OPB.

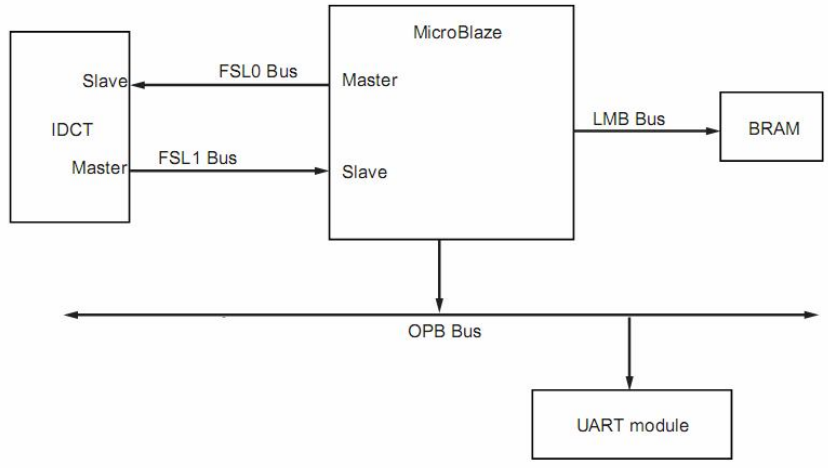

Рисунок 3 – Підключення ядра зворотнього ДКП до процесора

Вся вбудована система складається із самого MicroBlaze, двох шин FSL, ядра IDCT, вбудованої шини OPB, модуль комунікації (UART module) та вбудований модуль па'мяті RAM, в якому розміщується програма.

На рис.4 показана детальніша схема підключення ядра IDCT до MicroBlaze. Управляючі сигнали виходять із інтерфейсу шини FSL. Зовнішні сигнали, такі як глобальний системний годинник чи системний скид можуть бути легко інтегровані.

Це ядро повністю залежне від платформи виконання. Пристрій вимагає 4 IOBs, 4 елемента MULT18x18, 4 RAMB16s і близько 1300 слайсів; вбудований процесор працює на частоті 100 МГц.

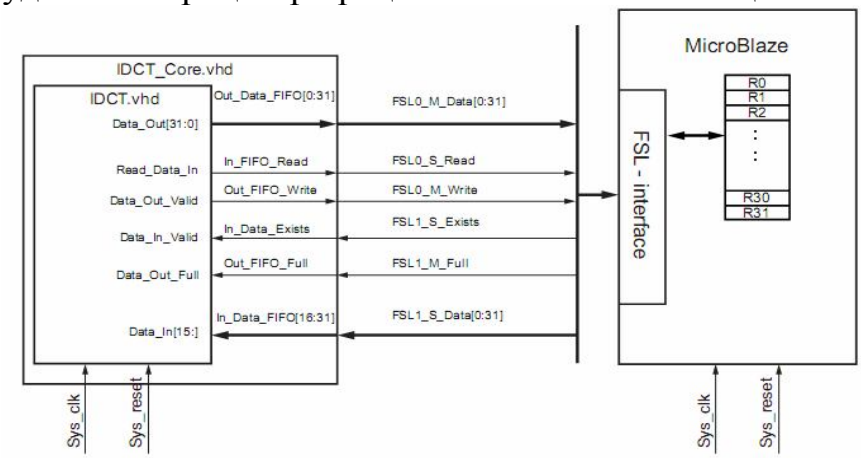

Рисунок 4 – Детальна схема підключення ядра IDCT через шину FSL до MicroBlaze.

Логічні блоки, що конфігуруються (КЛБ), представляють собою основні логічні ресурси для виконання комбінаційних схем. Кожен КЛБ ПЛІС сімейства Spartan-3E складається з чотирьох секцій, згрупованих у пари (рис.5). Ліва пара називається slicem (слайд) і містить повнофункціональні логічні генератори. Пара slicel може реалізувати тільки логіку [4].

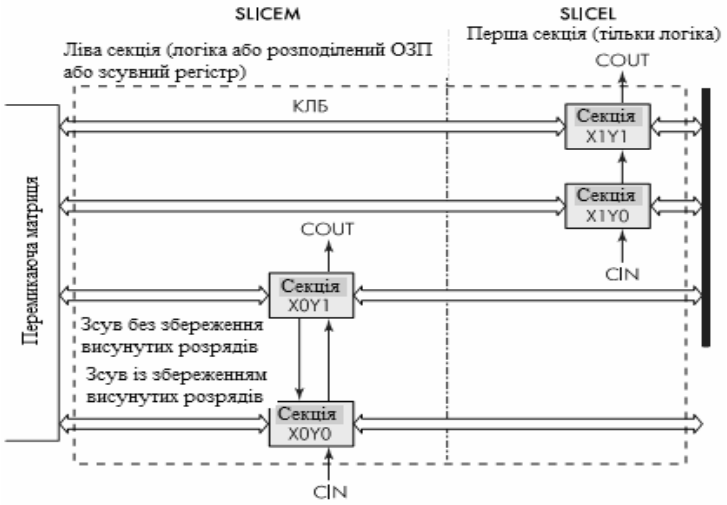

Рисунок 5 – КЛБ ПЛІС сімейства Spartan-3E

Опис ядра YCbCr2RGB

Ядро представляє себою VHDL код, що визначає оптимізовану структуру конвертора, використовуючи тільки п'ять мультиплікаторів для здійснення YCrCb до RGB-перетворення. Нижче представлені формули у вигляді їх реалізації в самому ядрі [5].

Конвертор YCbCr2RGB підтримує наступні перетворення:

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} CA & (1-CA-CB) & CB \\ CC(-CA) & CC(CA+CB-1) & CC(1-CB) \\ CD(1-CA) & CD(CA+CB-1) & CD(-CB) \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_Y \\ O_C \\ O_C \end{bmatrix}$$

де CC та CD дозволяють динамічне стиснення діапазону для B-Y та R-Y, а константи OY та OC представляють зміщення для CB та CR. Коефіцієнти CC та CD дозволяють скоротити динамічний діапазон компонент кольору.

Трансформація із моделі YCbCr у RGB визначена у наступному виразі.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/CD \\ 1 & \dfrac{-CB}{CC(1-CA-CB)} & \dfrac{-CA}{CD(1-CA-CB)} \\ 1 & 1/CC & 0 \end{bmatrix} \begin{bmatrix} Y-O_Y \\ C_B-O_C \\ C_R-O_C \end{bmatrix}$$

У цьому виразі є наступні нетривіальні константні множення коефіцієнтів. Коефіцієнти квантування:

$$ACOEF = \left[2^{CWIDTH-2} \frac{1}{CD} \right],$$

$$BCOEF = \left[2^{CWIDTH-2} \frac{-CB}{CC(1-CA-CB)} \right],$$

$$CCOEF = \left[2^{CWIDTH-2} \frac{-CA}{CD(1-CA-CB)} \right],$$

$$DCOEF = \left[2^{CWIDTH-2} \frac{1}{CC} \right],$$

де [] означає округлення до найближчого цілого; CWIDTH - точність коефіцієнтів квантування.

Операція зсуву виконується перед множенням матриць. Щоб використати переваги суматорів у DSP48, процес обчислення виконується навпаки: спочатку множення потім зсув. Значення коефіцієнтів зсуву регулюються заново:

$$ROFFSET = RND\_CONST - ACOEF * COFFSET + YOFFSET * 2^{CWIDTH-2}$$

$$GOFFSET = RND\_CONST - (BCOEF + CCOEF) * COFFSET + YOFFSET * 2^{CWIDTH-2}$$

$$BOFFSET = RND\_CONST - DCOEF * COFFSET + YOFFSET * 2^{CWIDTH-2}$$

Тоді перетворення із моделі YCbCr у RGB визначається так:

$$R = [(ACOEF * C_R + ROFFSET) / 2^{CWIDTH-2}]_{OWIDTH} + Y,$$
$$G = [(BCOEF * C_B + CCOEF * C_R + GOFFSET) / 2^{CWIDTH-2}]_{OWIDTH} + Y,$$
$$B = [(DCOEF * C_B + BOFFSET) / 2^{CWIDTH-2}]_{OWIDTH} + Y,$$

де [ ]k означає округлення до k бітів. Округлення виконується додаванням 0.5*LSB (LSB=2- INBITS, де INBITS – точність вхідного RGB зображення) та усіканням даних. Константа округлення, $RND\_CONST = 2^{CWIDTH-3}$, додається до ROFFSET, GOFFSET та BOFFSET.

Попередні вирази відносяться до випадків необмеженої довжини додавальників. Практично:

$$RND\_CONST = 2^{MWIDTH-OWIDTH-2-FAMILY\_HAS\_MAC},$$
$$SCALE\_M = 2^{IWIDTH+CWIDTH-MWIDTH+FAMILY\_HAS\_MAC},$$
$$ROFFSET = RND\_CONST - [(ACOEF * COFFSET + YOFFSET * 2^{CWIDTH-2}) / SCALE\_M],$$
$$GOFFSET = RND\_CONST - [(BCOEF + CCOEF) * COFFSET + YOFFSET * 2^{CWIDTH-2}) / SCALE\_M],$$
$$BOFFSET = RND\_CONST - [(DCOEF * COFFSET + YOFFSET * 2^{CWIDTH-2}) / SCALE\_M].$$

На рис.6 представлена архітектура реалізації описаних вище виразів.

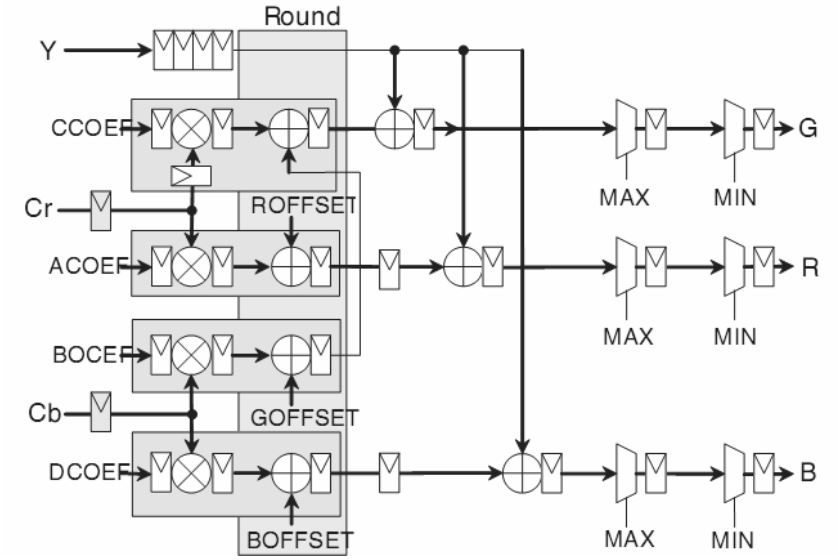

Рисунок 6 –Архітектура конвертора із моделі YCbCr у RGB

Враховуючи округлення\квантування, схема реалізує наступні вирази:

$$R = [C_R * ACOEF + ROFFSET]_{OWIDTH} + Y,$$
$$G = [C_B * BCOEF + C_R * CCOEF + ROFFSET]_{OWIDTH} + Y,$$
$$B = [C_B * DCOEF + BOFFSET]_{OWIDTH} + Y,$$

де [ ]k означає округлення до k бітів.

Сірі блоки представляють логічні блоки, які виконуються за допомогою блоків DSP, коли вони доступні у пристрої.

Програмна частина алгоритму декодування

Програма алгоритму декодування зображень розроблена на мові С. Вона складається із декількох частин та повністю повторює усі кроки алгоритму (рис.7).

```
void axta(double* DCT, SWORD* X, double* tDCT, BYTE* result, BYTE
dimens)
{
	double s = 0;
	double* temp = (double*)malloc(sizeof(double)*dimens*dimens);
	// temp = tDCT * X

	for (unsigned char i = 0; i < dimens; i++)
	{
		for (unsigned char j = 0; j < dimens; j++)
		{
			s = 0;
			for (unsigned char k = 0; k < dimens; k++)
			{
				s += tDCT[i*dimens+k] * X[k*dimens+j];
			}
			temp[i*dimens+j] = s;
		}
	}
	// res = temp * A
	for (unsigned char i = 0; i < dimens; i++)
	{
		for (unsigned char j = 0; j < dimens; j++)
		{
			s = 0;
			for (unsigned char k = 0; k < dimens; k++)
			{
				s += temp[i*dimens+k] * DCT[k*dimens+j];
			}
		result[i*dimens+j] = s;
		}
	}
free((void*)temp);
}
void convert_8x8_YCbCr_to_RGB_tab(BYTE *Y, BYTE *Cb, BYTE *Cr, BYTE
*tab, DWORD im_loc, DWORD X_image_bytes, BYTE *im_buffer)
// Function (such as optimization) can be called without parameters Cb, Cr
{
```

```
DWORD x,y;
BYTE nr, im_nr;
BYTE Y_val,Cb_val,Cr_val;
BYTE *ibuffer = im_buffer + im_loc;
nr=0;
for (y=0;y<8;y++)
 {
      im_nr=0;
      for (x=0;x<8;x++)
        {
         Y_val=Y[nr];
         Cb_val=Cb[tab[nr]]; Cr_val=Cr[tab[nr]];
         ibuffer[im_nr++] = rlimit_table[Y_val + Cb_tab[Cb_val]]; //B
         ibuffer[im_nr++] = rlimit_table[Y_val +
Cr_Cb_green_tab[WORD_hi_lo(Cr_val,Cb_val)]]; //G
         ibuffer[im_nr++] = rlimit_table[Y_val + Cr_tab[Cr_val]]; // R
         nr++; im_nr++;
        }
      ibuffer+=X_image_bytes;
 }
}
```

Рисунок 7 – Програма алгоритму декодування

Спочатку зчитується файл із розширенням jpg у буфер. Із буфера у циклі вибирається заголовок файлу, де зберігається інформація про зображення, таблиці Хаффмана, таблиці квантування та інформація стосовно децимації компонент кольоровості. Вибір відбувається по відповідним до інформації маркерам. Далі розраховується кількість блоків розміром 8x8, по яких виконується декодування таблиць Хаффмана, зворотне ДКП для кожної компоненти Y, Cb, Cr та приведення зображення до моделі RGB. Вхідними даними програми є зображення у форматі jpg.

Тестування ядер

Множення блоку коефіцієнтів 8x8 на матрицю косинусів розміром 8x8 відбувається за 1261 тактів процесора. Таким чином, результат роботи апаратного ядра IDCT дорівнює 12мкс, що у 2250 разів швидше, ніж швидкість системи Matlab. Необхідно зазначити, що апаратний модуль повністю залежний від платформи виконання, тобто при використанні більш швидкої та потужної плати можна отримати ще кращі результати.

На рис.8 наведено часову діаграму роботи ядра, а саме розрахунок одного коефіцієнту.

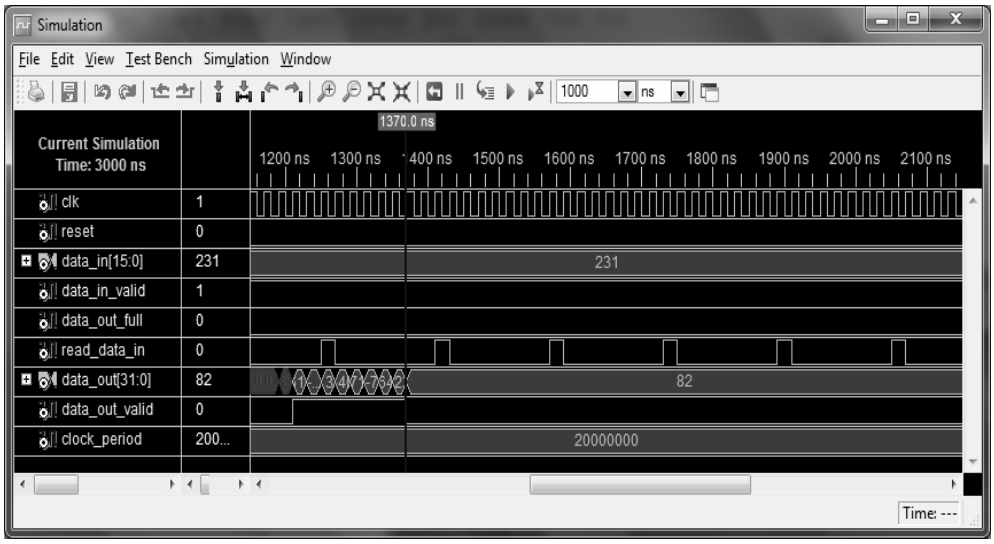

Рисунок 8 – Часова діаграма роботи ядра IDCT

Сигнал data_out_valid у положенні ‘0’ вказує наявність правильного вихідного результату. На рис.9 показано діаграму часу роботи зворотнього ДКП.

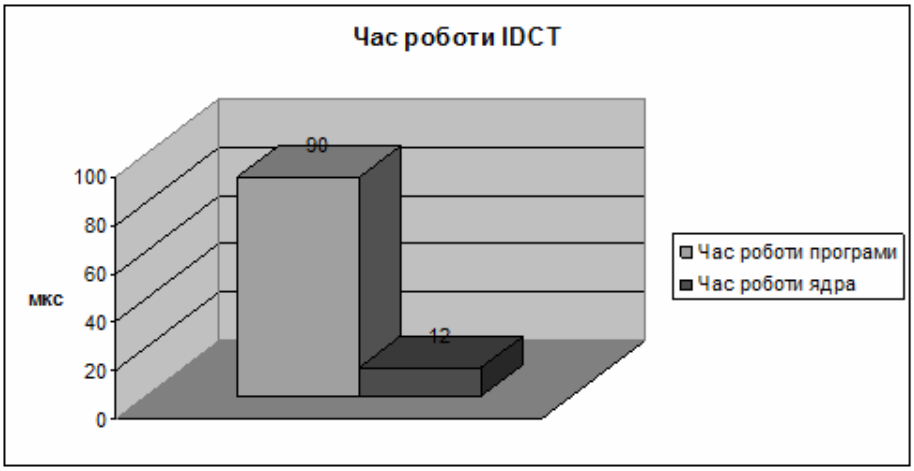

Рисунок 9 - Результат роботи ядра IDCT

Для тестування ядра конвертора YCbCr2RGB розроблено тест на мові VHDL. Часова діаграма показує отриманий результат (рис.10).

```
wait until rising_edge(clk);
                ce <= '1';
                sclr <= '0';
                y <= "00010000";
               cb <= "10000000";
               cr <= "10000000";
```

Проект ядра займає 182 слайси, 151 LUT, 57 IOB, 4 MULT18X18SIO.

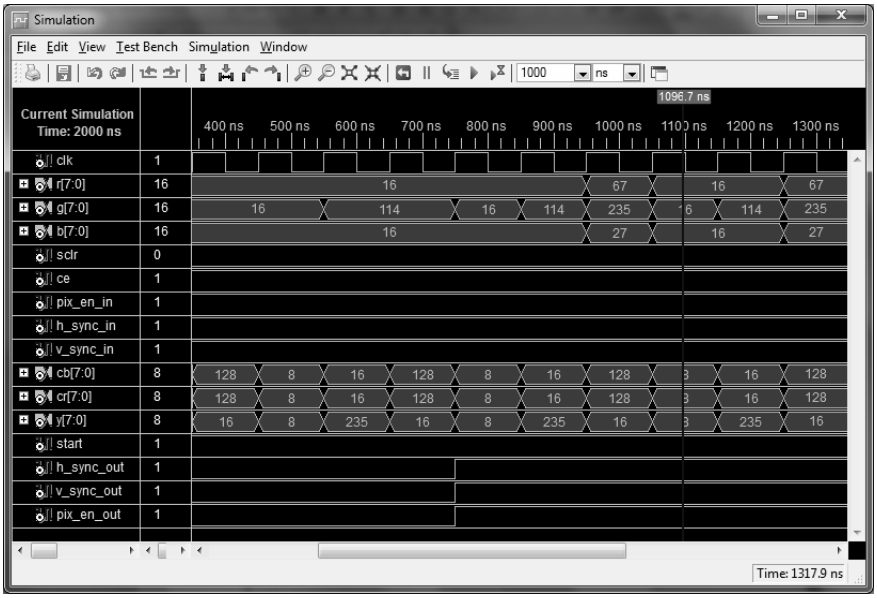

Рисунок 10 – Часова діаграма роботи ядра конвертора YCbCr2RGB

Час обробки однієї матриці коефіцієнтів займає 0,64мкс. На рис.11 показана діаграма часу роботи конвертору YCbCr2RGB.

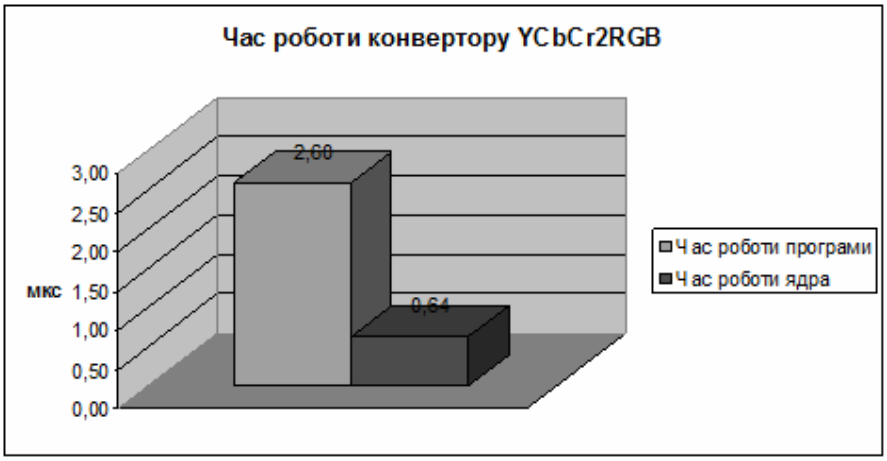

Рисунок 11 - Результат роботи конвертору

На рис.12 показано приклади двох стислих зображень і відповідний їм розмір файлу. Зображення практично ідентичні. Порівняння результатів у системі Matlab вказує на таку максимально можливу різницю по каналах: R =12; G = 5; B = 13.

Різність обчислювалась наступним чином:

```
jpeg_image = imread('c:/temp/matlab/cat.jpg','jpg');
bmp_image = imread('c:/temp/matlab/image.bmp','bmp');
dif_image = abs(jpeg_image - bmp_image);
imshow(dif_image);
max(max(dif_image))
ans(:,:,1) =  12
ans(:,:,2) =   5
ans(:,:,3) =  13
```

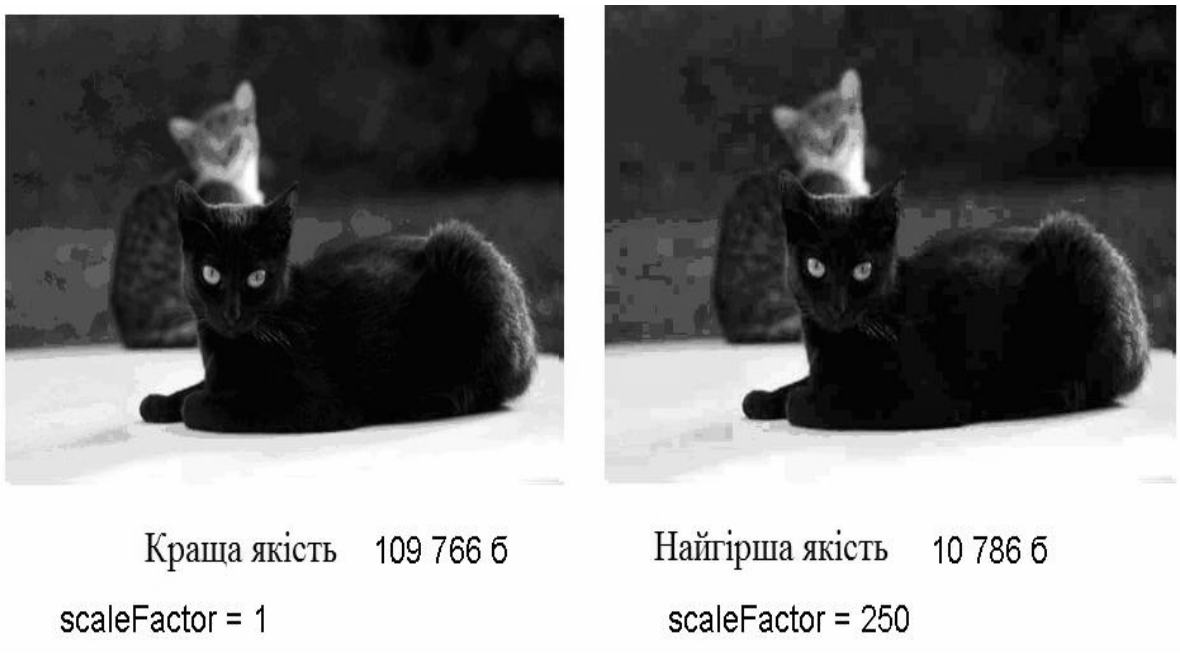

Рисунок 12 - Стиснені зображення із різним фактором якості

Висновки

Аналіз ефективності показав, що апаратна реалізація вирішує питання швидкодії і дає приріст 87% в операції зворотнього ДКП і 75% - в операції перекладу з моделі YCbCr в RGB.

Алгоритм JPEG використано лише як приклад. Запропонована методика може бути використана для будь-яких алгоритмів, що здійснюють компресію шляхом втрати якості.

Список літератури

1. Кузелин М.О. Современные семейства ПЛИС фирмы Xilinx : справочное пособие / М.О.Кузелин, Д.А. Кнышев, В.Ю.Зотов. – 440с.

2. Авторская научная библиотека. Стандарт сжатия JPEG [Електронний ресурс]. – Режим доступу: http://www.sernam.ru/cod_15.php

3. Hans-Peter Rosinger Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel [Електронний ресурс] / Hans-Peter Rosinger. – Режим доступу: http://www.xilinx.com/support/documentation/application_notes/xapp529.pdf

4. Spartan-3E FPGA Family: Complete Data. Preliminary Product Specification Sheet. - [Електронний ресурс]. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds635.pdf

5. Gabor Szedo Color-Space Converter: YCrCb to RGB / Gabor Szedo: [Електронний ресурс].– Режим доступу: http://www.xilinx.com/support/documentation/application_notes/xapp931.pdf

Надійшла до редакції 05.10.2011.
Рецензент: канд. техн. наук, Зори С.А.

ISSN 2074-7888, Наукові праці Донецького національного технічного університету, серія Проблеми моделювання та автоматизації проектування, 2011, Випуск 10 (197)

Ю.И. Жиданова, **А.В. Самощенко**
Донецкий национальный технический университет

Декодирование изображений формата JPEG средствами языка VHDL. Предлагается метод уменьшения потока данных за счет аппаратной реализации операций на FPGA, а также приводятся результаты исследований на примере общеизвестного формата JPEG.
Ключевые слова: изображения, кодирование, аппаратная реализация, FPGA.

Yu.I. Zhidanova, A.V. Samochshenko
Donetsk National Technical University

Decoding JPEG Images by Means of VHDL Language. A method is proposed to reduce the flow of data through the hardware implementation of operations on the FPGA, as well as the results of investigations on the example of a well-known format of JPEG.
Keywords: image encoding, hardware implementation, FPGA.