

Применение процессора NM6403 (L1879BM1) для сжатия изображений.

С.В. Мушкаев, С.В. Ландышев

В данной статье анализируются возможности применения процессора NM6403 в области сжатия статических изображений. Рассматривается алгоритм реализации ДКП (дискретного косинусного преобразования), исследуются вопросы повышения точности вычислений. Приводится краткое описание реализации JPEG кодера на базе процессора NM6403, а также некоторые аспекты применения векторного ядра в задачах кодирования.

Введение

В задачах с достаточно трудоемкими вычислениями всегда возникает вопрос о наиболее эффективном способе их реализации. Для различных архитектур он в каждом случае свой, но общий принцип состоит в минимизации арифметических операций. Для процессора NM6403 это справедливо только от части, так как в ряде случаев полная минимизация вычислений приводит к неэффективности использования векторного узла из-за невозможности распараллелить алгоритм. Таким образом, основная задача заключается в выборе оптимального набора однотипных макроопераций, оперирующих блоками данных, таких как: умножение или суммирование отдельных частей матриц и векторов. К примеру, данный принцип позволяет эффективно организовывать на процессоре NM6403 параллельные вычисления БПФ по основанию 16 или 32 [3].

Алгоритм вычисления двумерного ДКП на процессоре NM6403

Прямое вычисление ДКП 8x8 можно рассматривать как предельный случай, когда дополнительная минимизация уже не требуется. Выделим эти макрооперации, исходя из формулы прямого ДКП.

Двумерное дискретное косинусное преобразование над блоком 8x8 определяется формулой

$$Y(u, v) = \frac{1}{4} w(u)w(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos\left[\frac{\pi(2x+1)u}{16}\right] \cos\left[\frac{\pi(2y+1)v}{16}\right], \text{ где } u, v, x, y = 0, 1, 2, \dots, 7$$

x, y - пространственные координаты пикселей в блоке 8x8 ,

u, v - координаты в частотной области,

$w(u) = 1/\sqrt{2}$ если $u = 0$; 1-в остальных случаях,

$w(v) = 1/\sqrt{2}$ если $v = 0$; 1-в остальных случаях.

В эквивалентном матричном виде формула прямого ДКП записывается в виде:

$$[Y] = [C] \times [X] \times [C^T]$$

[X]- исходная матрица 8x8

[Y]- результирующая матрица 8x8 дискретного косинусного преобразования

[C], [C^T]- матрицы 8x8 коэффициентов преобразования (косинусов) C_{ij}.

Таким образом, вычисление ДКП состоит в перемножении входной матрицы на две матрицы C и C^T. Данное умножение можно осуществить двумя способам:

1. $[Y] = ([C] * [X]) * [C^T]$

2. $[Y] = [C] * ([X] * [C^T])$

Так как вычисления строятся на арифметике с фиксированной точкой, то при реализации ДКП на векторном процессоре данные варианты имеют существенные различия. Выбор наиболее предпочтительного зависит от разрядностей элементов матриц X, C и C^T. Структура векторного множителя такова, что накопление результата при перемножении

двух матриц $[C]$ и $[X]$ происходит в той же разрядной сетке, которую имеет множитель $-[X]$. Поэтому первый вариант накладывает более жесткие ограничения на входные данные. Второй же вариант не привязан к разрядности входных данных и, следовательно, является более универсальным и гибким в выборе разрядности коэффициентов C и C^T , в силу этого ниже будет рассматриваться именно этот вариант.

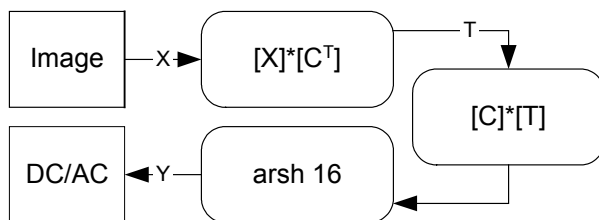
Процесс умножения делится на два этапа:

1. Вычисление промежуточной матрицы $[T]=[X]x[C^T]$
2. Вычисление конечной матрицы $[Y]=[C]x[T]$

Проиллюстрируем одну из наиболее простых и быстрых реализаций алгоритма ДКП для обработки 8-разрядных входных данных. Разрядность матриц C^T и Y удобно взять равной 32 битам, это предотвратит возможность переполнения и упростит последующую за ДКП обработку данных. Точность преобразования зависит от количества значимых бит в элементах матриц C и C^T , как будет показано ниже использование 7 значимых младших бит и знака, обеспечивает приемлемую точность. Следовательно, разрядность матрицы C достаточно взять равной 8 битам.

Заметим, что все элементы матриц интерпретируются как знаковые, поэтому диапазон входных элементов X_{ij} лежит в пределах $-128..127$, что не исключает возможность обработки и обычных изображений с 256 градациями серого.

Коэффициенты преобразования с плавающей точкой (индекс fp) $C_{fp}(i,j)$ лежат в диапазоне $(-0.491 ..+0.491)$, перевод их в формат с фиксированной точкой (индекс fix) осуществляется по формуле $C_{fix}(i,j)=round(C_{fp}(i,j)*2^8)$, где 2^8 - масштабирующий множитель, $round()$ - функция округления до ближайшего целого. Использование одинакового масштабирующего множителя - 2^8 как для матрицы $[C]$, так и для $[C^T]$ позволяет накапливать результат в 32-х разрядной сетке матриц $[T]$ и $[Y]$ без переполнения и использовать единый нормирующий сдвиг на 16 бит вправо ($arsh 16$). То есть, вся процедура вычисления ДКП состоит из двух этапов умножения и нормирующего сдвига:



В развернутом виде последовательность умножения матриц и внутреннее разбиение упакованных данных показано на рис.1 (жирными линиями выделены границы 64 разрядных слов, тонкими - разбиение на составляющие элементы).

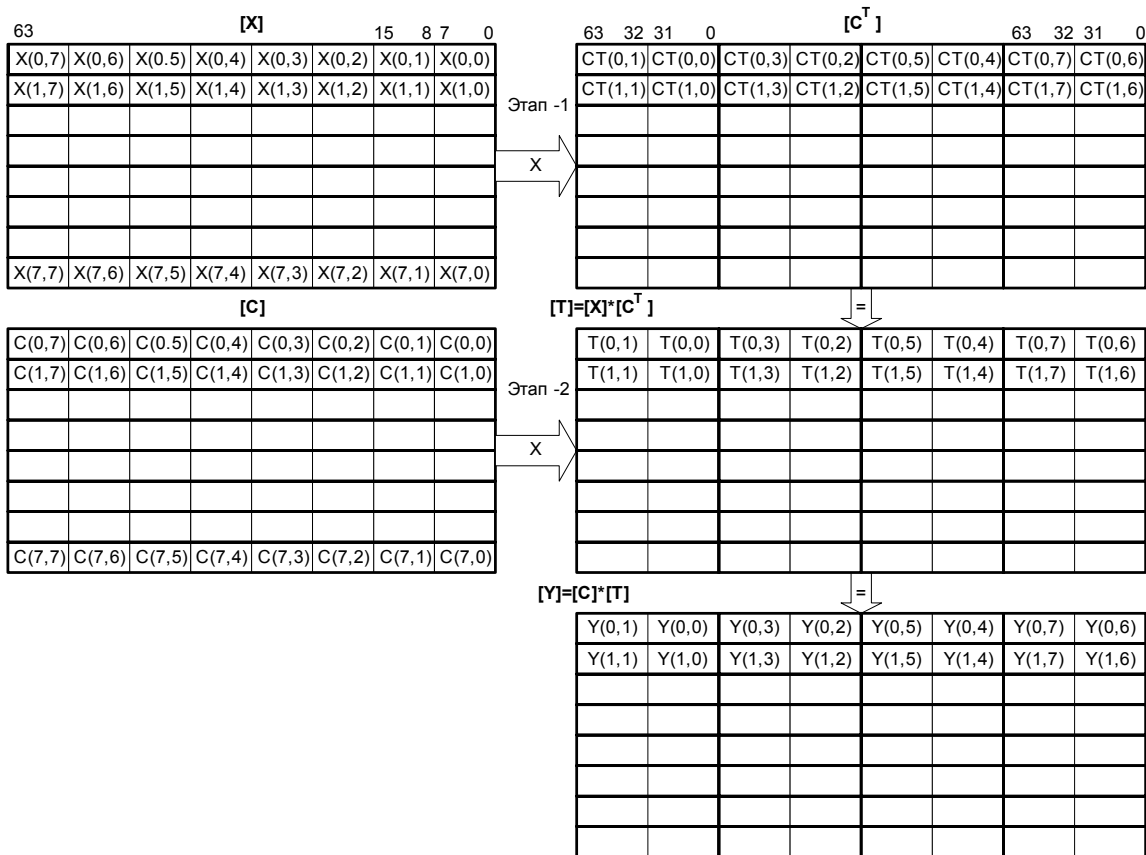


Рис. 1 Последовательность умножения матриц при вычислении ДКП над блоком 8x8 и внутреннее разбиение упакованных данных.

На первом этапе умножения вычисляется промежуточная матрица $[T] = [X] * [C^T]$. Коэффициенты $[C^T]$ загружаются в матрицу весовых коэффициентов по две колонки, а восемь векторов-строк матрицы $[X]$ последовательно поступают на вход векторного умножителя (см. Рис. 1). За каждый такт происходит умножение одного вектора-строки матрицы $[X]$ на загруженную матрицу весовых коэффициентов, т.е. находится пара чисел матрицы $[T]$: $T(i,j)$ и $T(i,j+1)$ (Рис. 2). Таким образом, полное перемножение двух матриц 8x8 требует 4 перезагрузки матрицы коэффициентов и происходит за 32 шага умножения. Учитывая, что коэффициенты матрицы $[C^T]$ постоянны, то при работе со всем изображением можно так распределить вычисления, что они не потребуют постоянной перезагрузки рабочей матрицы и тогда среднее время вычисления одного элемента матрицы $[T]$ составит примерно 0.5 такта. Эффективность умножения также достигается за счет использованием SIMD инструкции - гер 32, выполняющей умножение блоками по 32 вектора.

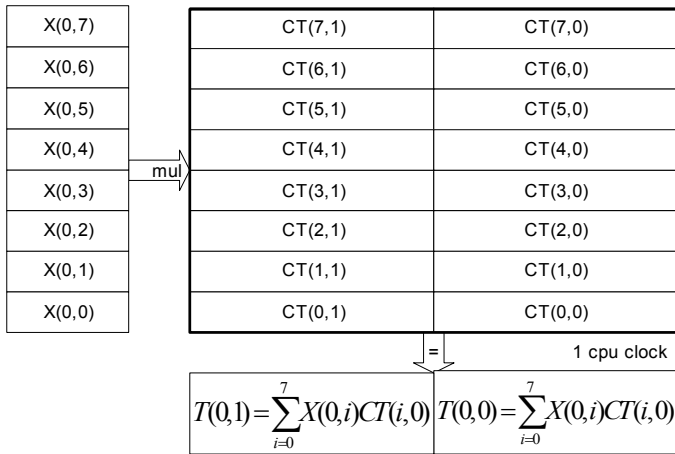


Рис. 2 Схема умножения нулевой строки матрицы [X] на два левых столбца матрицы [C^T]. Этап 1.

На втором этапе находится матрица [Y]=[C]*[T]. Элементы промежуточной матрицы [T] загружаются в матрицу весовых коэффициентов также по две колонки, а вектора-строки матрицы [C] последовательно поступают на вход векторного умножителя (см. Рис. 3). Процесс умножения такой же как и на первом этапе за тем исключением, что на данной стадии возможно оперировать блоками только по восемь 64-разрядных слов (инструкция гер 8), после чего матрицу весов необходимо перегружать. Так как перегрузка матрицы занимает 32 такта и на фоне 8 умножений не удастся полностью совместить эти операции, то каждый элемент матрицы Y(i,j) в среднем находится в 4 раза дольше, т.е. примерно за 2 такта.

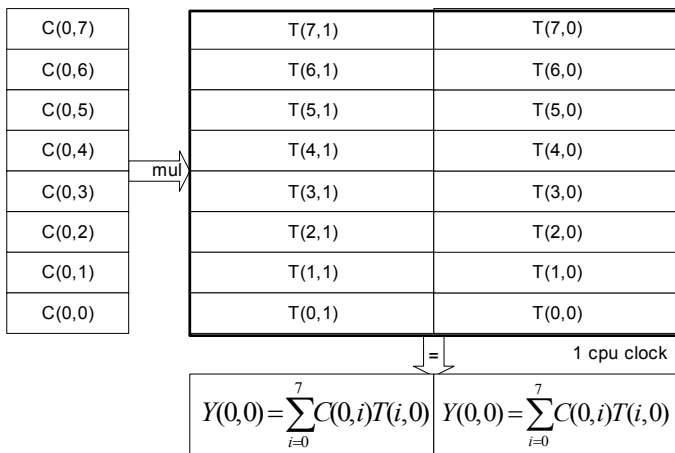


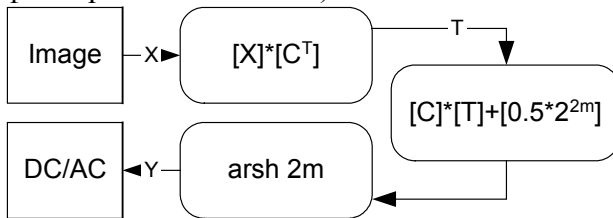
Рис. 3 Схема одного шага умножения нулевой строки матрицы [C] на два левых столбца матрицы [T]. Этап 2.

В итоге, суммарное расчетное время умножения [Y]=[C]×[X]×[C^T] составляет 64*0.5+64*2=160 тактов. Реальная цифра составляет около 190 тактов. К общему времени следует также добавить 32 такта, необходимых для нормализующего сдвига, однако эту операцию можно выполнить параллельно с процедурами, следующими за ДКП, такими как: квантование или Z-упорядочение. В этом случае среднее время обработки одного пикселя изображения можно считать равное 3 тактам.

Точность алгоритмов ДКП преобразования.

При выборе способа реализации того и или иного алгоритма приходится одновременно учитывать несколько факторов: входной и выходной диапазон данных, точность получаемых результатов и общая производительность. Все они взаимно исключают друг друга, но как показал анализ, правильный подбор коэффициентов и незначительная вариация вычислений может существенно повысить точность без потери общей производительности.

Точность вычисления ДКП зависит от формата представления коэффициентов преобразования C_{ij} в числах с фиксированной точкой, а также от способа конечной обработки результатов. Формат представления коэффициентов определяется: разрядностью ячеек, масштабирующим коэффициентом- M и способом округления - $C_{fix} = \text{round}(C_{fp} * M)$. Конечная обработка состоит так же в округлении результатов умножения при отбрасывании дробной части. В случае ДКП коэффициенты преобразования C_{ij} лежат в диапазоне $(-0.491 .. +0.491)$, поэтому можно применять масштабирующий коэффициент - $M = 2^m$, где m - эффективная разрядность ячеек (число значимых бит и знак). Тип округления в обоих случаях используется до ближайшего целого, так как эта схема показывает наилучшие результаты и на векторном ядре может быть реализована параллельно с операцией умножения путем прибавления к произведению $[C] * [T]$ константы $1/2 * 2^{2m}$ (число 0.5 с фиксированной точкой).



На основе данной схемы был проведен анализ зависимости точности ДКП преобразования от разрядности косинусных коэффициентов - m . Под разрядностью в данном контексте понимается количество значимых бит со знаком в n -битовом слове ($n \geq m$). Для простоты коэффициенты обеих матриц $[C]$ и $[C^T]$ вычислялись с одинаковым масштабирующим коэффициентом 2^m , а разрядная сетка для накопления результатов предполагалась достаточно широкой, чтобы исключить возможность переполнения.

Тестирование проводилось по всем блокам 8×8 изображения "Lena" размером 256×256 . Точность оценивалось по сходству результатов ДКП полученных на основе арифметики с фиксированной и плавающей точкой. Сравнение проводилось по нескольким критериям, ниже приводятся две наиболее показательные оценки:

$$1. \text{RMSE}(X) = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x_{fix}(i) - x_{fp}(i))^2} \quad - \text{среднеквадратичная ошибка величины } X \text{ по}$$

всему кадру. В качестве X подставлялись коэффициенты ДКП преобразования: DC (показатель средней яркости блока) и AC (значения пространственных частот). Раздельный анализ этих коэффициентов упрощает интерпретацию результатов.

$$2. \text{PSNR}(X) = 20 \cdot \log \left(\frac{255}{\text{RMSE}(X)} \right) \quad - \text{отношение сигнал/шум между исходным кадром и}$$

восстановленным изображением с помощью обратного ДКП по схеме:

Исх. кадр \Rightarrow ДКП_{fix} \Rightarrow ОДКП_{fp} \Rightarrow Восст. кадр

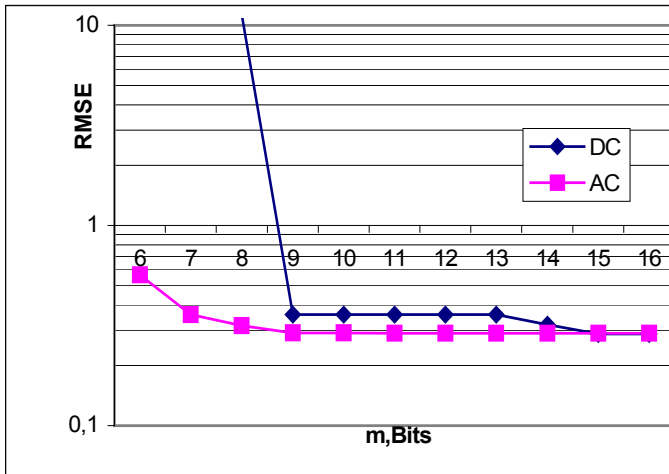


Рис. 4 Зависимость среднеквадратичной ошибки DC/AC коэффициентов от разрядности-*m* коэффициентов $C(i,j)$

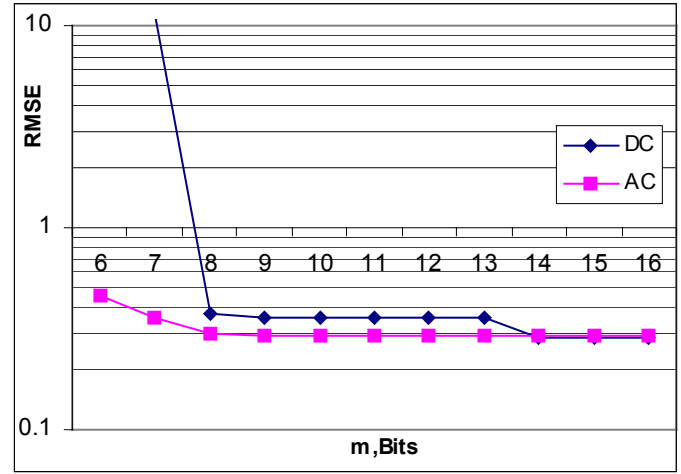


Рис. 5 Зависимость среднеквадратичной ошибки DC/AC коэффициентов от разрядности - *m* с коррекцией коэффициентов $C(i,j)$

Как видно из Рис. 4, при переходе от 8-разрядной точности коэффициентов преобразования к 9-разрядной наблюдается резкий перепад в точности DC коэффициентов.

Увеличение точности обусловлено особенностью записи коэффициентов C_{ij} , участвующих в вычислении DC, в формате с фиксированной точкой. Эти коэффициенты расположены в первом столбце матрицы $[C^T]$ и первой строке матрицы $[C]$, все они равны числу 0,35355339059327. В формате с фиксированной точкой это число записывается в двоичном виде как: 0.0101 1010 1000 0010 0111... Видно, что после 9 знака после запятой следует 5 нулей, т.е. точность записи числа 0.35355 с помощью 9 бит (0101 1010 1b) будет такая же, как и с помощью 14 бит(0101 1010 1000 00b). В случае 8 бит ошибка связана с неопределенностью округления в большую или меньшую сторону, т.е. округление 8 бита до числа (0101 1010) практически также правомерно, как и до (0101 1011). Очевидно, что, используя только один способ округления, полученные коэффициенты C_{ij} будут вносить некоторую систематическую ошибку в DC. Этот недостаток можно устранить, получая коэффициенты с помощью вероятностного округления, что равносильно поочередному округлению $C^T(i, 0)$ и $C(0, j)$ то в большую, то в меньшую сторону. Как видно из графиков (на Рис. 5, Рис. 8), в результате такой коррекции резко возрастает точность ДКП преобразования на разрядностях меньших 9.

В ряде задач часто требуется обрабатывать беззнаковые данные. В этих случаях дополнительное увеличение точности (см. Рис.7, Рис. 8) можно достичь, если предварительно перевести входные значения из диапазона $[0..2^m-1]$ в диапазон $[-2^{m-1}.. +2^{m-1}-1]$, вычитая из входных данных число 2^{m-1} . Для обычных изображений это соответствует смещению диапазона $[0..255]$ в диапазон $[-128..127]$. В итоге результат ДКП преобразования будет отличаться от обычного варианта только DC коэффициентом на некоторую константу. После коррекции DC элемента результат полностью приводится к нормальному виду. Заметим, что механизм такого смещения заложен только в стандарте JPEG. Схема, приведенная на Рис. 6, позволяет обрабатывать любые 8-разрядные изображения и в других стандартах: MPEG, H.263 и пр.

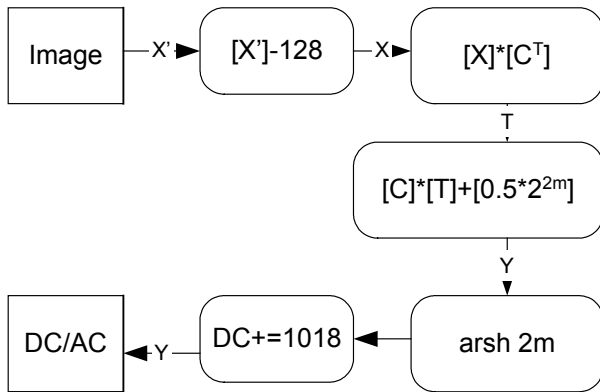


Рис. 6 Альтернативная схема вычисления ДКП с DC-смещением

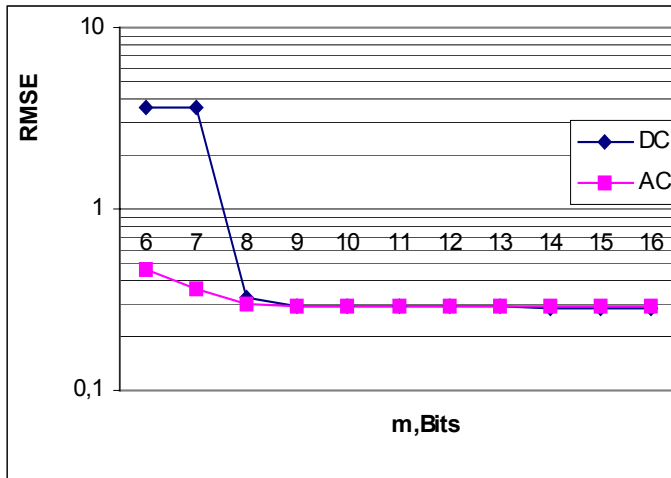


Рис. 7 Зависимость среднеквадратичной ошибки DC/AC коэффициентов от разрядности - m с коррекцией коэффициентов $C(i,j)$ и DC смещением

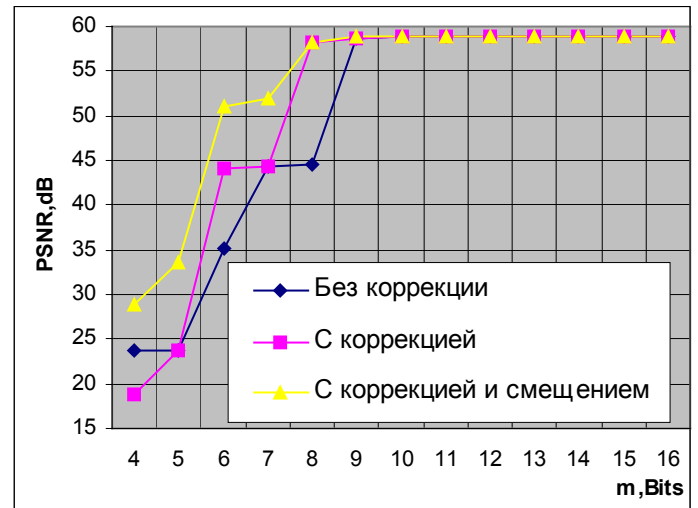


Рис. 8 Зависимость отношения сигнал/шум восстановленного изображения после ДКП от разрядности коэффициентов $C(i,j)$

На практике схема, приведенная на рис.6, с применением механизма DC-смещения позволяет без переполнения использовать более точное 12-битовое представление коэффициентов матрицы $[C^T]$. О точности такой схемы можно судить, сравнивая результаты прямого и обратного ДКП с плавающей и фиксированной точкой (см. табл.1). В таблице приводится зависимость отношения сигнал/шум восстановленного изображения "Lena 256x256" от коэффициента квантования Q .

Q	PSNR, dB		
	ДКП _{fix} -ОДКП _{fix}	ДКП _{fix} -ОДКП _{fp}	ДКП _{fp} -ОДКП _{fp}
1	52.84	58.65	58.9
2	48.41	49.63	49.65
4	45.74	46.38	46.39
8	42.13	42.37	42.37
16	38.03	38.12	38.12
32	33.78	33.82	37.82
64	29.80	29.84	29.84

Таблица 1. Зависимость отношения сигнал/шум восстановленного изображения от коэффициента квантования Q .

Краткое описание реализации JPEG кодера на процессоре NM6403.

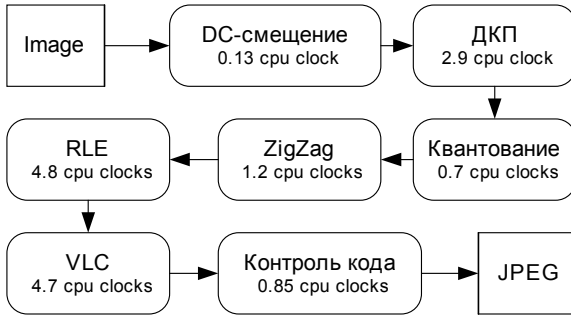


Рис. 9 Блок-схема JPEG кодирования

На Рис. 9 приведена блок-схема JPEG-кодирования и относительное распределение процессорного времени между блоками (для NM6403). Для каждого блока указано усредненное время обработки (в процессорных тактах) одного пикселя изображения "Lena 256x256". Для последних трех блоков временные характеристики достаточно условны, так как они сильно зависят от таблиц квантования и самого изображения. Суммарное среднее время обработки одного пикселя изображения составляет около 15-16 тактов, что соответствует (при тактовой частоте NM6403 - 40 МГц) 25 кадрам в секунду для монохромных CIF изображений.

Вся предварительная обработка, состоящая из операций: DC-смещения, ДКП, квантования, Z-упорядочения и насыщения, достаточно хорошо векторизуется, так как эти процедуры не зависят от входных данных. Остальные операции связаны с кодированием (RLE) и упаковкой данных (VLC). В них присутствуют команды условного перехода и работа с кодами переменной длины, поэтому полностью векторная обработка данных становится невозможной. Однако в некоторых процедурах часть работы можно переложить на векторный узел с целью упрощения и ускорения последующей обработки на скалярном ядре. В частности, в процедурах RLE и контроле кода с помощью векторной предобработки удастся значительно сократить объем скалярных операций, а количество условных переходов свести к минимуму.

В обоих случаях суть векторной предобработки заключается в составлении битового вектора по некоторой анализируемой последовательности из 64-х элементов. Битовый вектор - это 64-разрядное слово, каждый бит которого определяется булевой функцией от соответствующего элемента последовательности: $b_i = f(x_i)$. В качестве простейших булевых функций на NM6403 возможно использовать следующие варианты:

$$f(x_i) = \begin{cases} 0, x_i \geq 0 \\ 1, x_i < 0 \end{cases}, f(x_i) = \begin{cases} 1, x_i \geq 0 \\ 0, x_i < 0 \end{cases}, f(x_i) = \begin{cases} 0, x_i = 0 \\ 1, x_i > 0 \end{cases}, f(x_i) = \begin{cases} 0, x_i - \text{четный} \\ 1, x_i - \text{нечетный} \end{cases} \text{ и т.д.}$$

Данные функции выполняются на векторном узле, поэтому за такт в битовом векторе можно накапливать по 2-4-8-16... бит в зависимости от разрядности элементов x_i . В сочетании с другими логическими и арифметическими функциями можно реализовать и более сложные варианты, например: $f(x_i) = \begin{cases} 0, x_i = C \\ 1, x_i \neq C \end{cases}$

В частности, в процедуре контроля кода, следующей после упаковки кодов Хаффмана (процедура VLC) в бинарный поток, производится проверка на наличие в потоке символа FF. По стандарту JPEG число FF является зарезервированным служебным полем и в случае его вхождения должно быть заменено на FF00. Анализируя битовый вектор, полученный по

функции $f(x_i) = \begin{cases} 1, x_i = FF \\ 0, x_i \neq FF \end{cases}$, можно сравнительно легко определить наличие или

местоположение слова FF. Относительно быстрое получение битового вектора на процессоре NM6403 (8 - 64 такта в зависимости от разрядности x_i и сложности $f(x)$) позволяет эффективно применять данный механизм для задач поиска элемента в больших массивах.

Другой вариант использования битового вектора применяется в процедуре группового кодирования (RLE). Задача RLE состоит в поиске цепочек одинаковых элементов в некоторой последовательности и замене их на пары <счетчик повторений, значение>, что в результате уменьшает избыточность данных. При обработке изображений ищутся цепочки нулей в блоках по 64 элемента и заменяются на пары <RUN,LEVEL>, где RUN-длина цепочки (от 0 до 15) , LEVEL- следующий за цепочкой ненулевой элемент или нуль если длина цепочки превышает 15.

Проиллюстрируем нахождение таких цепочек и их длину на примере.

Допустим, после операций ДКП, квантования и Z-упорядочения образовалась некоторая последовательность X, состоящая, к примеру, преимущественно из нулей, и пусть 11 ненулевых элементов (a,b,c,d,e,f,g,h,i,j,k) находятся на позициях 0,2,4,5,6,13,19,25,26,34,53 соответственно:

X= 0000000000k000000000000000000000j0000000ih00000g00000f000000edc0b0a
 $x_{63} \dots \dots x_1, x_0$

Сформируем битовой вектор по формуле $b_i = f(x_i) = \begin{cases} 0, x_i = 0 \\ 1, x_i \neq 0 \end{cases}, i=0..63$

B= 000000000010000000000000000010000001100000100000100000011110101
 $b_{63} \dots \dots b_1, b_0$

Инвертируем битовый вектор: Y[00]= Not(B)

Y[00]= 1111111111011111111111111111111101111111001111101111101111110001010

Выполним итеративный цикл от i=1 до 15

Y[i] = (Y[i-1]>>1) AND (Y[i-1])

Y[01]= 011111111101111111111111111111110111111100111110111110111111000101
 Y[02]= 001111111100111111111111111111110011111100011110011110011111000000
 Y[03]= 000111111100011111111111111111110001111100001110001110001111000000
 Y[04]= 000011111100001111111111111111110000111100000110000110000111000000
 Y[05]= 000001111100000111111111111111110000011100000010000010000011000000
 Y[06]= 00000011110000001111111111111111000000110000000000000000001000000
 Y[07]= 0000000111100000001111111111111110000000100000000000000000000000
 Y[08]= 0000000011100000000111111111111111000000000000000000000000000000
 Y[09]= 0000000001100000000011111111111111000000000000000000000000000000
 Y[10]= 0000000000100000000001111111111111000000000000000000000000000000
 Y[11]= 0000000000000000000000011111111111000000000000000000000000000000
 Y[12]= 0000000000000000000000000111111111100000000000000000000000000000
 Y[13]= 0000000000000000000000000001111111000000000000000000000000000000
 Y[14]= 0000000000000000000000000000011111000000000000000000000000000000
 Y[15]= 0000000000000000000000000000000111100000000000000000000000000000

Просуммируем биты в каждой колонке и получим вектор S из 64 элементов:

S= 0123456789A0123456789ABCDEFGHIFFF0123456700123450123450123450123456000101

X= 0000000000k000000000000000000000j0000000ih00000g00000f000000edc0b0a

Каждая полученная сумма s_i показывает количество подряд стоящих нулей слева от числа x_i , а значение суммы, увеличенное на единицу, указывает на относительное положение

следующего элемента - LEVEL. Извлекая из вектора S смещения s_{i+1} и совершая по ним переходы параллельно в векторе S и X, получим искомый набор <RUN,LEVEL>:
<a,?><b,1><c,1><d,0><e,0><f,6><g,5><h,5><i,0><j,7><0,15><a,2><?,10>

Заключение

Анализ вычисления ДКП на арифметике с фиксированной точкой показал, что 8-разрядное представление коэффициентов C_{ij} по точности практически не уступает ДКП с плавающей точкой - соотношение сигнал/шум восстановленного изображения составляет около 58.2dB (Рис. 8), а при разрядностях выше 14 точность достигает предела - PSNR=58.9dB. При той же схеме вычислений комбинирование разных разрядностей коэффициентов [C] и [C^T], например 8 и 12 соответственно, делает отличие от плавающей точки еще меньшим - PSNR=58.6dB, а при уровне квантования более 4 оно исчезает полностью. Отличительная особенность прямого способа вычисления ДКП от быстрых алгоритмов с фиксированной точкой заключается в минимальном количестве этапов умножения, что с одной стороны приводит к меньшему накоплению ошибок, а с другой - позволяет строить вычисления на векторном процессоре.

Продемонстрирована применимость векторного узла процессора NM6403 в задачах кодирования данных. Гибкая программируемая структура векторного сопроцессора и возможность оперировать данными произвольной разрядности позволяет эффективно реализовывать различные схемы работы с битовыми векторами, распараллеливая тем самым сложно векторизуемые алгоритмы.

Литература

1. НТЦ Модуль "Процессор NM6403. Введение в архитектуру"
2. С. Кун. Матричные процессоры на СБИС. М.:Мир, 1991
3. Кашкаров В., Мушкаев С. "Организация параллельных вычислений в алгоритмах БПФ на процессоре NM6403" // Цифровая обработка сигналов, 2001 № 1)
4. Кашкаров В."Организация параллельных вычислений преобразования Адамара на процессоре NM6403 (J11879VM1)" // Цифровая обработка сигналов, 2001 № 3).