Александр Калачев (г. Барнаул)

БОГАТЫЙ НАБОР ДЛЯ НЕОРДИНАРНЫХ ЗАДАЧ: ВОЗМОЖНОСТИ DSP В STM32F4 НА ЯДРЕ CORTEX-M4

*Перед вами стоит задача, требующая отработки решения на основе **широкого динамического диапазона входных параметров?** Необходима **повышенная точность** вычислений? К вашим услугам — возможности **вычислительного процессора 32-разрядного микроконтроллера STM32F4 на ядре Cortex-M4.***

Процессоры с ядрами ARM на данный момент широко применяются в мобильных устройствах и встраиваемых системах различного применения. Архитектура обладает такими привлекательными свойствами, как удобная и эффективная система команд, мощная поддержка при разработке программного обеспечения, высокая энергоэффективность. Одной из последних версий архитектуры ARM является ARM-Cortex-xx, предполагающая варианты для процессоров общего назначения — Cortex-Ax, и для встраиваемых приложений — Cortex-Mx. Функционально архитектуры практически идентичны, за исключением того, что Cortex-A оптимизирована по быстродействию с целью достижения высокой производительности, а Cortex-M — по энергопотреблению, обеспечивая баланс между энергоэффективностью и производительностью.

Требования приложений для встраиваемых систем постоянно повышаются — это и обеспечение пользовательского интерфейса (сенсорный ввод, графический интерфейс), высокие мультимедийные возможности (кодирование/декодирование аудио/видеопотоков), обработка данных (цифровая обработка сигналов, интеллектуальный анализ), и все это при ограниченных энергетических ресурсах. По этой причине микро-

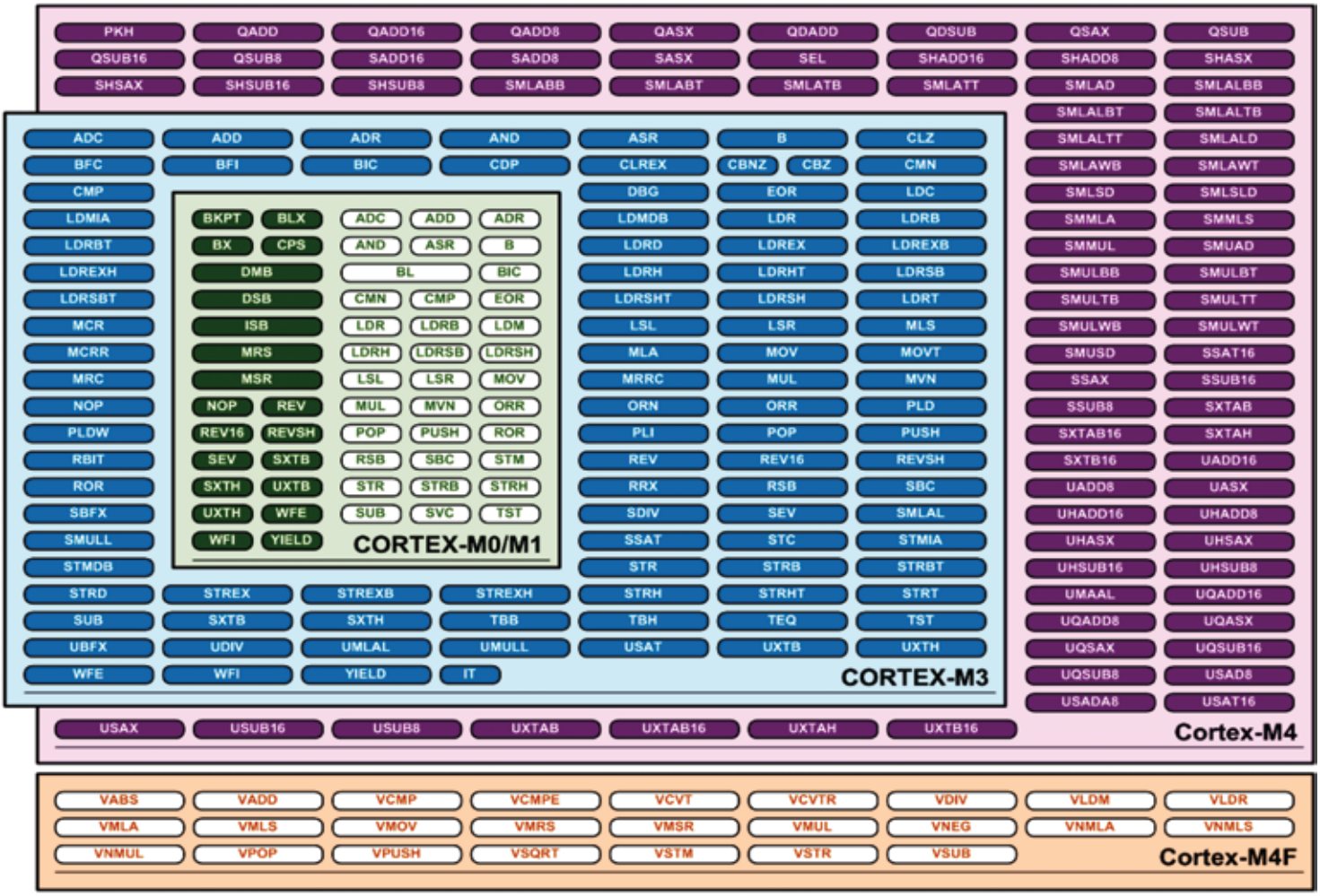

Рис. 1. **Системы команд процессорных ядер архитектур Cortex-Mx**

контроллеры на базе архитектуры ARM-Cortex-Mx набирают популярность.

Общая характеристика микроконтроллеров серии SMT32 F4

32-разрядные микроконтроллеры STMicroelectronics серии STM32 F4 принадлежат к наиболее мощной версии архитектуры ARM-Cortex-M – ARM-Cortex-M4F [1, 2]. Это позволяет им сочетать в себе возможности:

- микроконтроллеров общего назначения:
– богатый набор периферии;
– низкое энергопотребление;
– производительность реального времени;
- цифрового сигнального процессора:
– оптимизированный набор команд, большинство из которых выполняются за один цикл;
– SIMD инструкции;
– схема параллельного сдвига регистров;
- вычислительного процессора:
– блок вычислений в формате с плавающей точкой;
– одинарная точность;
– простота программирования;
– широкий диапазон обрабатываемых данных.

Сравнение систем команд процессорных ядер семейства Cortex-Mx представлено на рисунке 1.

Основные характеристики ядра Cortex-M4:

- архитектура ARMv7-ME (полная совместимость с Cortex-M3);
- модуль MAC (*Single-cycle multiply-accumulate*)
- оптимизированные инструкции SIMD (*Single instruction multiple data*)
- инструкции с насыщением (*Saturating*)
- модуль Floating-Point Unit одинарной точности (FPU) (версия Cortex-M4F);
- аппаратное деление (2…12 циклов).

DSP-возможности STM32 F4

Несмотря на все разнообразие задач цифровой обработки сигналов (ЦОС), во многих случаях все сводится к компрессии/декомпрессии сигнала, набору фильтров и ряду ортогональных преобразований [4].

Фильтры с конечной импульсной характеристикой (КИХ-фильтры) находят применение в:

- системах передачи данных и телекоммуникации;
- подавлении эха;
- сглаживании данных.

Типовой вид КИХ-фильтра:

$$y[n]=\sum_{k=0}^{N-1}h[k]x[n-k];$$

где h[k] – коэффициенты фильтра, x[n-k] – n-k-й отсчет входного сигнала, N-порядок фильтра.

Фильтры с бесконечной импульсной характеристикой (БИХ-фильтры), они же – рекурсивные фильтры, применяются для:

- обработки звука;
- при управлении двигателями.

В формировании выходных отсчетов сигнала рекурсивного фильтра кроме отсчетов самого сигнала принимают участие выходные отсчеты самого фильтра:

$$y[n]=b_0x[n]+b_1x[n-1]+b_2x[n-2]+a_1y[n-1]+a_2y[n-2];$$

где ai, bi – коэффициенты фильтра, x[i] – отсчеты входного сигнала, y[i] – выходные отсчеты фильтра.

Быстрые преобразования сигналов, в частности быстрое Фурье-преобразование:

- сжатие звука;
- системы коммуникаций с расширением спектра;
- удаление шумов.

Типовой операцией практически для всех быстрых преобразований является т.н. «бабочка», например, для преобразования Фурье:

$$Y[k_1]=X[k_1]+X[k_2]$$
$$Y[k_2]=(X[k_1]-X[k_2])e^{-j\omega}$$

Как видно, одной из основных операций является умножение с накоплением – MAC, которая в зависимости от конкретной задачи может быть 8-, 16- или 32- битным числом.

Модуль умножения MAC в STM32 F4 способен за один такт выполнять инструкции умножения или умножения с накоплением (таблица 1) [3]:

- знаковое/беззнаковое умножение;
- знаковое/беззнаковое умножение с накоплением;

Таблица 1. **Операции модуля MAC STM32 F4**

Операция	Инструкции ядра
16 x 16 = 32	SMULBB, SMULBT, SMULTB, SMULTT
16 x 16 + 32 = 32	SMLABB, SMLABT, SMLATB, SMLATT
16 x 16 + 64 = 64	SMLALBB, SMLALBT, SMLALTB, SMLALTT
16 x 32 = 32	SMULWB, SMULWT
(16 x 32) + 32 = 32	SMLAWB, SMLAWT
32 x 32 = 32	MUL
32 ± (32 x 32) = 32	MLA, MLS
32 x 32 = 64	SMULL, UMULL
(32 x 32) + 64 = 64	SMLAL, UMLAL
(32 x 32) + 32 + 32 = 64	UMAAL
32 ± (32 x 32) = 32 (старшие разряды)	SMMLA, SMMLAR, SMMLS, SMMLSR
(32 x 32) = 32 (старшие разряды)	SMMUL, SMMULR
SIMD MAC	
Сложение/вычитание результатов произведений полуслов операндов (16 x 16) ± (16 x 16) = 32	SMUAD, SMUADX, SMUSD, SMUSDX
Сложение/вычитание результатов произведений полуслов операндов с последующим сложением с 32-битным предыдущим результатом (16 x 16) ± (16 x 16) + 32 = 32	SMLAD, SMLADX, SMLSD, SMLSDX
Сложение/вычитание результатов произведений полуслов операндов с последующим сложением с 32-битным предыдущим результатом (16 x 16) ± (16 x 16) + 64 = 64	SMLALD, SMLALDX, SMLSLD, SMLSLDX

Таблица 2. **SIMD инструкции STM32 F4**

Операция	Инструкции ядра
Знаковое сложение/вычитание полуслов или байт	SADD16, SADD8, SSUB16, SSUB8
Беззнаковое сложение/вычитание полуслов или байт	UADD16, UADD8, USUB16, USUB8
Знаковое сложение/вычитание полуслов или байт с последующим сдвигом результата на один бит вправо (деление полуслов/байт результата на два)	SHADD16, SHADD8, SHSUB16, SHSUB8
Беззнаковое сложение/вычитание полуслов или байт с последующим сдвигом результата на один бит вправо (деление полуслов/байт результата на два)	UHADD16, UHADD8, UHSUB16, UHSUB8
Вычисление суммы старшего и младшего полуслов первого и второго операнда и помещение её в старшее полуслово результата, вычисление разности старшего и младшего полуслов первого и второго операнда и помещение её в младшее полуслово результата (вычисления идут либо с учетом знаков полуслов, либо без учета)	SASX, UASX,
То же, но с последующим делением полуслов результата на два (сдвиг вправо полуслов на 1 разряд), операция знаковая	SHASX
Вычисление разности старшего и младшего полуслов первого и второго операнда и помещение её в старшее полуслово результата, вычисление суммы старшего и младшего полуслов первого и второго операнда и помещение её в младшее полуслово результата (вычисления идут либо с учетом знаков полуслов, либо без учета)	SSAX, USAX
То же, но с последующим делением полуслов результата на два (сдвиг вправо полуслов на 1 разряд), операция знаковая	SHSAX
Вычисление разностей байт операндов с последующим сложением абсолютных разностей	USAD8
Вычисление разностей байт операндов с последующим сложением абсолютных разностей, сложение полученного результата с ранее вычисленным значением	USADA8

• знаковое/беззнаковое умножение с накоплением с 64-битным результатом;

Группа команд MAC включает в себя:

- 4 операции 16-битного MAC;
- 2 операции 32-битного MAC;
- 7 операций с 8-битными данными.

Все команды, перечисленные в таблице 1, выполняются за один цикл. Одно это уже ускоряет выполнение программ цифровой обработки сигналов. Дальнейшее ускорение может быть достигнуто программной оптимизацией за счет применения циклических буферов данных, развертывания циклов, кеширования промежуточных переменных.

Для задач, оперирующих с 8- или 16-битными данными возможно применение SIMD инструкций (таблица 2), выполняющих действия с отдельными полусловами (16-разрядные значения) или байтами 32-разрядных регистров (т.н. пакетированные данные).

Минимизируется количество инструкций загрузки/хранения для обмена между памятью и банком регистров. Выполняются две или четыре транзакции данных за один цикл (соответственно для 16- или 8-разрядных данных), если нет необходимости в 32-бит данных. Ядро поддерживает группу инструкций для упаковки/распаковки данных. В группу SIMD-инструкций входят также несколько инструкций MAC-модуля.

Набор инструкций позволяет эффективно использовать параллелизм целого ряда алгоритмов ЦОС на уровне данных или на уровне операций.

Рассмотрим несколько примеров.

1) КИХ-фильтр

Типовой С-код, реализующий фильтр:

```
  void fir(q31_t *in, q31_t *out, q31_t *coeffs,
int *stateIndexPtr,
       int filtLen, int blockSize)
  {
    int sample;
    int k;
    q31_t sum;
    int stateIndex = *stateIndexPtr;
    for(sample=0; sample < blockSize; sample++)
      {
        state[stateIndex++] = in[sample];
        sum=0;
        for(k=0;k<filtLen;k++)
    {
      sum += coeffs[k] * state[stateIndex];
      stateIndex--;
      if (stateIndex < 0)
        {
          stateIndex = filtLen-1;
        }
    }
        out[sample]=sum;
      }
      *stateIndexPtr = stateIndex;
  }
```

Входной сигнал обрабатывается блоками, внутренний цикл (выделен жирным шрифтом) состоит из двух операций выборки из памяти, MAC, модификации указателей с использованием циклической адресации.

```
  sample = blockSize/4;
    do
    {
      sum0 = sum1 = sum2 = sum3 = 0;
      statePtr = stateBasePtr;
      coeffPtr = (q31_t *)(S->coeffs);
      x0 = *(q31_t *)(statePtr++);
      x1 = *(q31_t *)(statePtr++);
      i = numTaps>>2;
      do
      {
             c0 = *(coeffPtr++);
             x2 = *(q31_t *)(statePtr++);
             x3 = *(q31_t *)(statePtr++);
             sum0 = __SMLALD(x0, c0, sum0);
             sum1 = __SMLALD(x1, c0, sum1);
             sum2 = __SMLALD(x2, c0, sum2);
             sum3 = __SMLALD(x3, c0, sum3);
             c0 = *(coeffPtr++);
             x0 = *(q31_t *)(statePtr++);
             x1 = *(q31_t *)(statePtr++);
             sum0 = __SMLALD(x0, c0, sum0);
             sum1 = __SMLALD(x1, c0, sum1);
             sum2 = __SMLALD (x2, c0, sum2);
             sum3 = __SMLALD (x3, c0, sum3);

      } while(--i);
      *pDst++ = (q15_t) (sum0>>15);
      *pDst++ = (q15_t) (sum1>>15);
      *pDst++ = (q15_t) (sum2>>15);
```

```
    *pDst++ = (q15_t) (sum3>>15);
    stateBasePtr= stateBasePtr + 4;
  } while(--sample);
```

Используется развертывание цикла (внутренний цикл, в результате чего количество итераций внешнего цикла снизилось в 4 раза), отсчеты и коэффициенты фильтра кешируются в регистрах, применяется циклическая адресация буфера.

Внутренний цикл выполняется за 26 машинных циклов и содержит 16 16-битных MAC-операций – примерно 1.625 машинных цикла на один шаг вычисления фильтра (не оптимизированный код обеспечивает 12 машинных циклов на вычисление шаг фильтра). Для сравнения: в специализированных цифровых сигнальных процессорах (ЦСП), вычисление шага КИХ-фильтра выполняется за один машинный цикл.

2) БИХ-фильтр

Внутренний цикл вычислений БИХ-фильтра второго порядка содержит 16 машинных циклов:

```
xN = *x++;
yN = xN * b0;
yN += xNm1 * b1;
yN += xNm2 * b2;
yN -= yNm1 * a1;
yN -= yNm2 * a2;
*y++ = yN;
xNm2 = xNm1;
xNm1 = xN;
yNm2 = yNm1;
yNm1 = yN;
// Decrement loop counter
// Branch
```

3) Быстрые алгоритмы преобразования сигналов

Аналогично, одновременное вычисление суммы и разности операндов может ускорить вычисление «бабочки» при вычислении преобразования Фурье. Для 16-разрядных данных возможно и применение инструкции двойного MAC с последующим суммированием.

4) Обработка изображений

При обработке стереоизображений часто используются алгоритмы типа SSD Stereo Vision или SAD Stereo Vision. Аналогичные алгоритмы применяются и при компенсации размытия движущихся объектов в последовательности кадров [5, 6].

Алгоритм SSD (Sum Of Square Differences):

$$S(i,j,k)=\sum_{l=i-\frac{\omega-1}{2}}^{i+\frac{\omega-1}{2}}\sum_{m=j-\frac{\omega-1}{2}}^{j+\frac{\omega-1}{2}}(I_R(l,m)-I_L(l,m+k))^2$$

Алгоритм SAD (Sum Of Absolute Differences):

$$S(i,j,k)=\sum_{l=i-\frac{\omega-1}{2}}^{i+\frac{\omega-1}{2}}\sum_{m=j-\frac{\omega-1}{2}}^{j+\frac{\omega-1}{2}}(I_R(l,m)-I_L(l,m+k))$$

Чаще всего при обработке изображений идет работа с байтами, например, в случае монохромных изображений, или при работе с отдельными каналами цветности, или с группами байт, соответствующими одному пикселю изображения, возможно одновременное вычисление до четырех разностей вида:

$$I_R(l,m)-I_L(l,m+k).$$

5) Вейвлет-преобразования

Одним из примеров применения операций типа сложений/вычитаний с последующим сдвигом вправо могут быть вычисления вейвлет-коэффициентов по алгоритму Малла в случае использования вейвлет-функции Хаара при кратномасштабном анализе сигналов (рисунок 2).

Обработка сигналов в целочисленном виде связана с риском выхода за границы диапазона обрабатываемых значений (переполнение разрядной сетки). Так, при получении слишком больших значений из-за переполнения они станут отрицательными значениями, и наоборот. Таким образом, в ходе вычислений необходимо отслеживать данную возможность, а это, в свою очередь, выливается в затраты времени и потерю общей производительности.

Инструкции, реализующие арифметику с насыщением [3], предотвращают переполнение переменных отсечением мин/макс границ и снижают тем самым нагрузку на ядро (таблица 3). Среди инструкций с насыщением есть также и SIMD-инструкции.

По сравнению с Cortex-M3 применение в Cortex-M4 DPS инструкций дает, в зависимости от задачи, выигрыш от 30% до 70% при 16-разрядных данных (рисунок 3а), и выигрыш в 25% – 60% при 32-разрядных данных (рисунок 3б).

Возможности Cortex-M4 по цифровой обработке сигналов вполне сопоставимы с возможностями специализированных ЦСП при сохранении всех функций универсального процессора. Для 32-битных данных Cortex-M4 будет проигрывать в производительности ЦСП примерно на порядок. Для данных меньшей разрядности – 8- или 16-битных данных – проигрыш будет примерно в 1,5…2 раза меньше.

Модуль вычислений с плавающей точкой в STM F4

STM F4 выгодно отличается наличием модуля вычислений с плавающей точкой, что позволяет оперировать с данными в достаточно широком динамическом диапазоне и дает существенный выигрыш по сравнению с программной реализацией.

Модуль вычислений с плавающей точкой Cortex-M4 FPU является реализацией IP ядра ARM FPv4-SP – сопроцессор с плавающей точкой одинарной точности (32 бита – 1 знаковый бит, 8 бит экспоненты, 23 бита – мантисса). Практически полностью поддерживаются операции, определенные в стандарте IEEE.754 [7, 8]. Аппаратно не поддерживаются:

- вычисление остатка;
- округление числа с плавающей точкой до целого значения в формате с плавающей точкой;
- преобразования из двоичной системы в десятичную и обратно;
- сравнение значений в формате одинарной и двойной точности.

Однако данный факт не снижает функциональность FPU-модуля, так как эти операции реализуются программно. Кроме того, модуль поддерживает ряд нестандартных режимов работы:

- альтернативный режим «половинной» точности – 16-битные знаковые числа без поля экспоненты и без поддержки ненормализованных значений;
- режим сброса до нуля – все ненормализованные числа обрабатываются как нулевые значения;
- режим NaN – любая операция с данными, не являющимися числами в формате с плавающей точкой, возвращает значение NaN по умолчанию.

FPU-модуль имеет 32 регистра S0 – S31 (регистры 32-разрядные), которые могут быть также доступны для операций загрузки/хранения как 16 двойных 64-битных регистра D0…D15. Состояние FPU отображается в отдельном регистре конфигурации и статуса, в котором также хранятся текущие настройки, например, режим округления, а также флаги условий (отрицательный результат, флаг нуля, переноса, переполнения и исключительных ситуаций).

Поддерживаются режимы округления к ближайшему, к нулю, к плюс или минус бесконечности. Исключительные ситуации в зависимости от настроек могут вызывать или не вызывать прерывание.

Модуль поддерживает арифметические инструкции, инструкции сравнения, преобразования типов и загрузки/выгрузки.

В число арифметических инструкций входят вычисление абсолютного значения, инверсия знака одного или нескольких операндов, сложения, вычитания, умножения, деления, вычисление квадратного корня. FPU также поддерживает ряд MAC-инструкций, таких как умножение с последующим сложением или вычитанием, умножение с последующим сложением или вычитанием и сменой знака результата (таблица 4).

Для уменьшения погрешностей округления MAC-инструкции имеют вариант выполнения с округлением после завершения всех преобразований (т.н. «fused»-инструкции).

Сравнение возможно между операндами, или операнда с константой. Инструкции преобразования типов (таблица 5) позволяют преобразовывать данные между целыми типами, форматом с плавающей точкой одинарной или половинной точности, фиксированной точкой.

MAC-инструкции FPU выполняются за три машинных цикла, инструкции деления и вычисления квадратного корня – за 14 циклов. Остальные инструкции, кроме инструкций загрузки данных и множественной загрузки, выполняются за один цикл.

Инструкции загрузки/пересылки данных позволяют загружать/выгружать данные в/из оперативной памяти одинарной или двойной точности. Доступны инструкции множественной пересылки данных между оперативной памятью и регистрами FPU.

Поддержка стандарта IEEE.754 обеспечивает автоматическую поддержку типов данных, принятые в языке C, что упрощает программирование контроллера. Более того, аналогичные типы данных используются в метаязыках таких программ математического моделирования, как MATLAB или Scilab. Все это позволяет более эффективно использовать преимущества программирования на таком языке высокого уровня как C, включая сквозной путь проектирования – от математической модели, ее реализации и отладки высокоуровневыми средствами, до генерации C-кода (рисунок 4).

При этом наличие FPU-модуля существенно ускоряет работу алгоритмов обработки данных, использующих вычисления с плавающей точкой. Одним из наиболее эффектных примеров является вычисление КИХ-фильтра. На рисунке 5 представлено относительное время выполнения КИХ-фильтра с использованием различных подходов – вычисления в формате с плавающей точкой без использования FPU, с модулем FPU, вычисления с 16-битными данными и оптимизацией [8].

Сравнимая с целочисленными версиями производительность вещественных ЦОС позволяет работать с широким диапазоном входных данных, применять

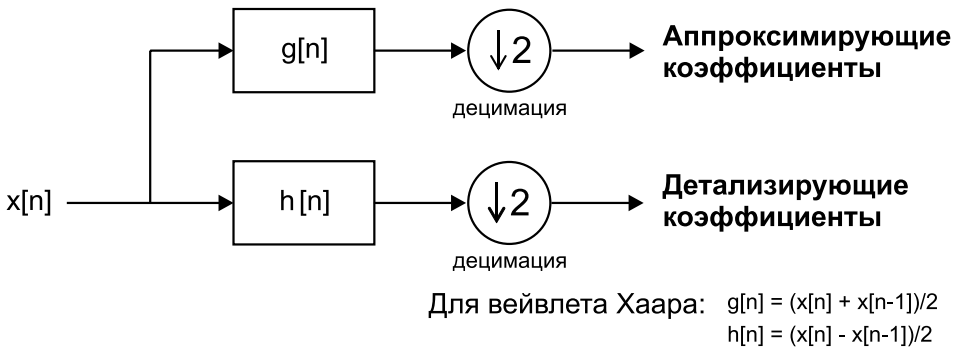

Рис. 2. **Структура одного уровня вычисления вейвлет-преобразования по алгоритму Малла**

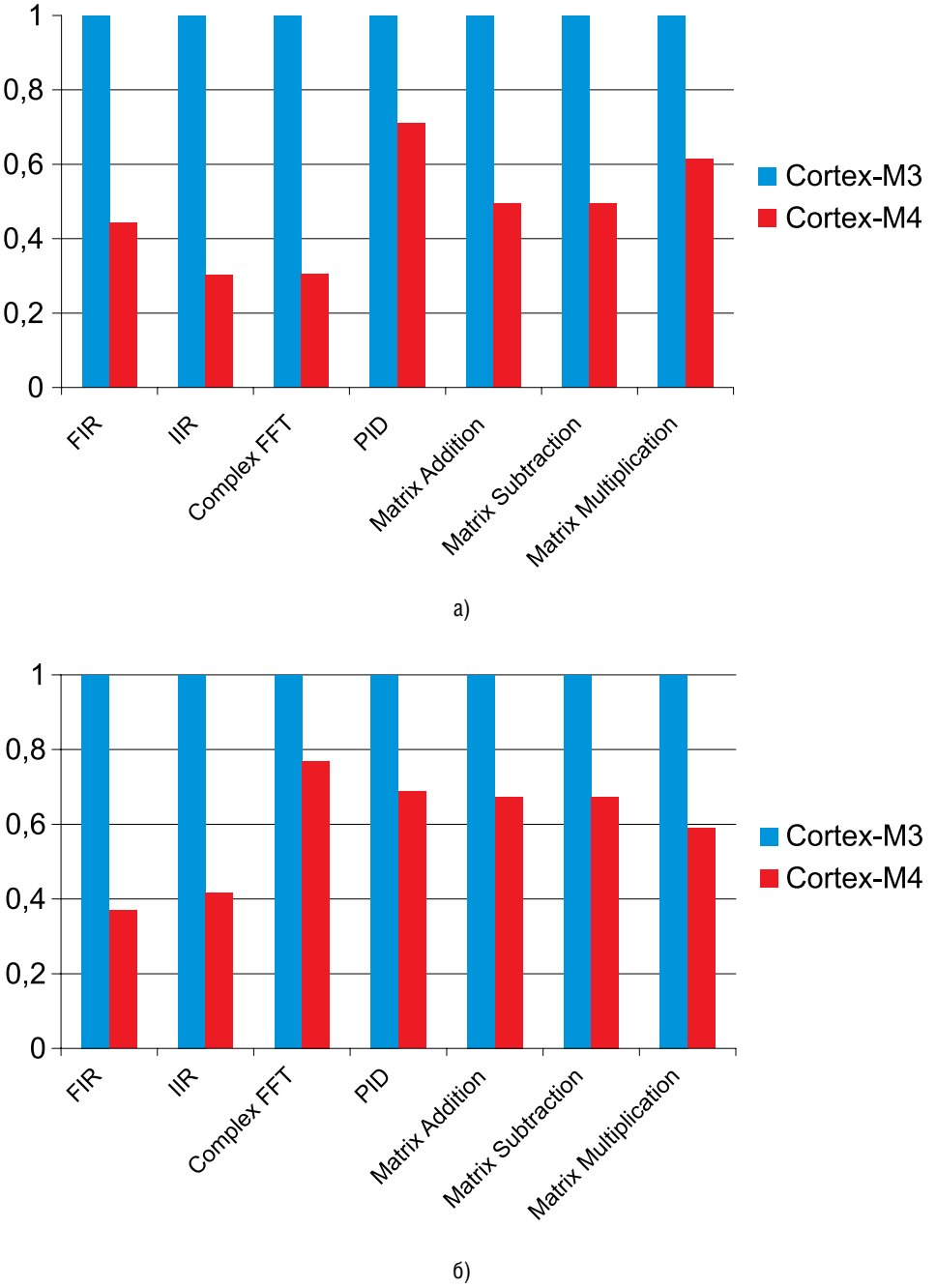

Рис. 3. **Сравнение производительности ядер Cortex-M3 и Cortex-M3 на различных алгоритмах ЦОС**

фильтры более высоких порядков без влияния погрешности округления или накопления ошибки вычислений.

Более высокая скорость обработки данных позволяет расширять функциональность приложения, используя дополнительные алгоритмы. Для устройств с ограниченными энергетическими ресурсами сокращается время нахождения в активном состоянии, что

Таблица 3. **Инструкции с насыщением STM32 F4**

Операция	Инструкции ядра
Знаковое/беззнаковое насыщение	SSAT, USAT
Знаковые/беззнаковые сложение с насыщением, вычитание с насыщением	QADD, QSUB, UQADD, UQSUB
Удвоение второго операнда, его насыщение и сложение его с первым, насыщение результата	QDADD
Удвоение второго операнда, его насыщение и вычитание его из первого, насыщение результата	QDSUB
SIMD инструкции с насыщением	
Знаковое/беззнаковое насыщение полуслов	SSAT16, USAT16
Знаковые/беззнаковые побайтное сложение с насыщением, вычитание с насыщением	QADD8, QSUB8, UQADD8, UQSUB8
Знаковые/беззнаковые сложение полуслов с насыщением, вычитание полуслов с насыщением	QADD16, QSUB16, UQADD16, UQSUB16
Знаковые/беззнаковые операции сложения старшего полуслова первого операнда с младшим полусловом второго, вычитания старшего полуслова второго операнда из младшего полуслова первого, насыщения результирующих полуслов	QASX, UQASX
Знаковые/беззнаковые операции вычитания младшего полуслова второго операнда из старшего полуслова первого, сложения младшего полуслова первого операнда со старшим полусловом второго, насыщения результатов	QSAX, UQSAX

Таблица 4. **Арифметические инструкции FPU STM32F4**

Операция	Описание	Мнемоники инструкций	Кол-во машинных циклов
Абсолютное значение	вещественного числа	VABS.F32	1
Инвертирование	вещественного числа инвертирование с последующим умножением	VNEG.F32 VNMUL.F32	1 1
Сложение	вещественных чисел	VADD.F32	1
Вычитание	вещественных чисел	VSUB.F32	1
Умножение	вещественных чисел с последующим сложением с последующим вычитанием с последующим сложением и инверсией с последующим вычитанием и инверсией	VMUL.F32 VMLA.F32 VMLS.F32 VNMLA.F32 VNMLS.F32	1 3 3 3 3
Умножение (fused/без округлений промежуточных результатов)	с последующим сложением с последующим вычитанием с последующим сложением и инверсией с последующим вычитанием и инверсией	VFMA.F32 VFMS.F32 VFNMA.F32 VFNMS.F32	3 3 3 3
Деление	вещественных чисел	VDIV.F32	14
Квадратный корень	вещественного числа	VSQRT.F32	14

Таблица 5. **Инструкции сравнения и преобразования данных FPU STM32F4**

Операция	Описание	Мнемоники инструкций	Кол-во машинных циклов
Сравнение	вещественных чисел, или с нулем	VCMP.F32 VCMPE.F32	1 1
Преобразование типов	между целыми типами, форматом с плавающей точкой одинарной или половинной точности, фиксированной точкой	VCVT.F32	1

в итоге сказывается на среднем энергопотреблении устройства и на сроке его работы.

Программные средства разработки

Программные средства разработки доступны с сайта STMicroelectronics и включают в себя набор стандартных библиотек для STM32, включая поддержку периферийных устройств, библиотеку Motor Control – векторный контроль на основе датчиков для 3-фазных бесщеточных двигателей, DSP-библиотеку для Cortex-M4, библиотеку для аудио приложений – MP3/WMA-декодер, контроль громкости, эквалайзер и ряд других библиотек [1].

Дополнительно на сайте компании ARM доступен набор библиотек CMSIS (*ARM Cortex Microcontroller Software Interface Standard*), которая по сути представляет собой независимую от конкретного производителя библиотеку HAL-уровня для процессоров серии Cortex-M [9].

В CMSIS входят следующие компоненты:

- CMSIS-CORE: базовая библиотека для доступа к ресурсам процессора и регистрам периферии процессорных ядер Cortex-M0, Cortex-M3, Cortex-M4, SC000 и SC300;
- CMSIS-DSP: библиотека алгоритмов ЦОС (для целочисленных данных, данных в формате с фиксированной или плавающей точкой одинарной точности);
- CMSIS-RTOS API: стандартизованный API для управления потоками выполнения, ресурсами системы, распределением процессорного времени для операционных систем реального времени;
- CMSIS-SVD (*System View Description*): XML-файлы, содержащие программные настройки микропроцессорной системы, включая периферийные устройства.

В DSP-библиотеке для Cortex-M4 поддерживаются практически распространенные алгоритмы ЦОС и ряд математических библиотек:

- базовые математические функции – векторная математика;
- быстрые тригонометрические и трансцендентные функции – sin, cos, sqrt и т.д.;
- интерполяция – линейная, билинейная;
- комплексная арифметика;

• статистические функции — вычисления минимальных и максимальных значений,
• различные алгоритмы фильтрации — БИХ-, КИХ- фильтры, алгоритм минимальной среднеквадратичной ошибки (LMS);
• преобразования сигналов (БПФ и др.);
• матричная арифметика;
• ПИД — контроллер;
• поддержка функций работы с массивами — копирование/заполнение массивов, конвертирование типов данных.

Библиотека предоставляет простой и прозрачный доступ к ресурсам процессора и периферийных устройств, что существенно упрощает миграцию между ядрами различных типов, уменьшает время на освоение нового микроконтроллера, и в совокупности уменьшает время выхода новой программной или аппаратной разработки на рынок. Для рынка встраиваемых устройств разработка программного обеспечения является одним из самых значимых факторов, определяющих конечную стоимость продукта.

Заключение

Модуль вычислений с плавающей точкой в STM32F4 позволяет без потери производительности использовать вычисления с вещественными числами в таких задачах, как кодирование, декодирование или цифровая фильтрация аудиосигналов, обработка сигналов датчиков, декодирование мультимедийных потоков, данных управления технологическим циклом.

STM32F4 является гибкой высокопроизводительной архитектурой, сочетающей в себе достоинства управляющего микроконтроллера, цифрового сигнального процессора и высокую точность вычислений. Способность STM32F4 осуществлять быстрые вычисления с привычными типами данных, такими как вещественные числа одинарной точности, позволяет применять его в задачах, работающих с широким динамическим диапазоном входных параметров или требующих повышенной точности. В результате при реализации того или иного алгоритма на STM32F4 существенно сокращается время на адаптацию алгоритма и его программную реализацию.

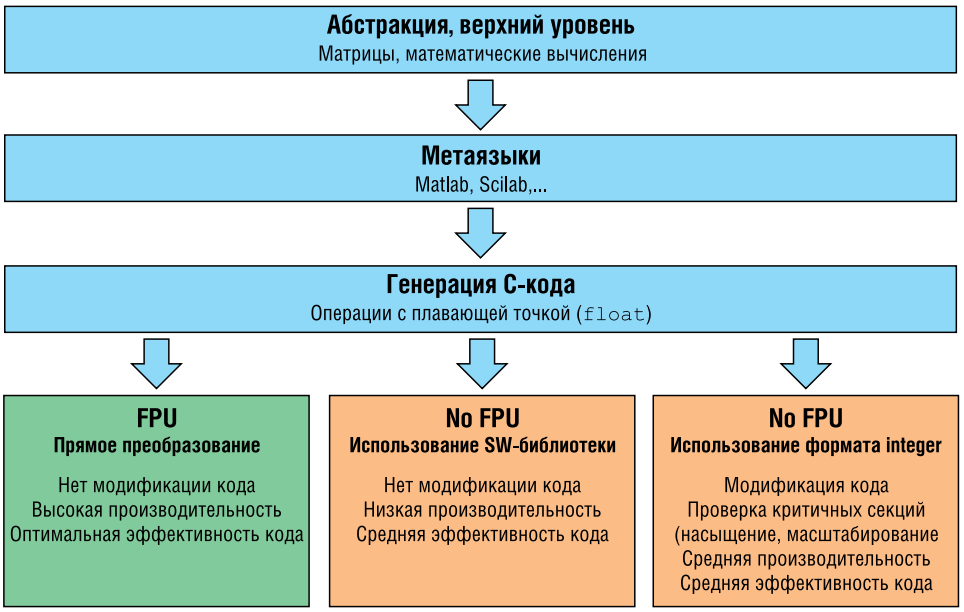

Рис. 4. **Этапы разработки программного кода**

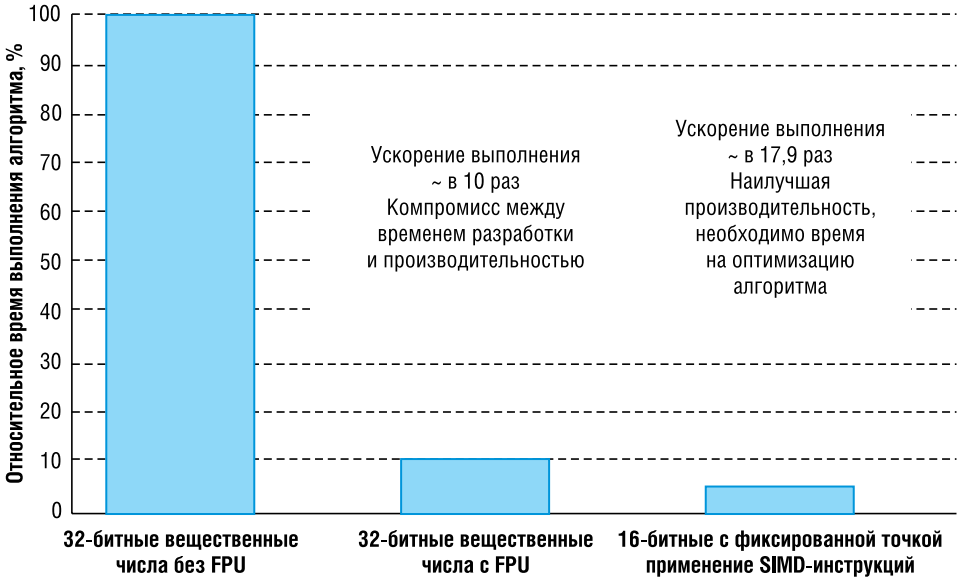

Рис. 5. **Относительное время расчета КИХ-фильтра**

Литература

1) STM32 F4 Hi-Performance & DSP//*http://www.st.com/internet/mcu/subclass/1521.jsp*

2) Роман Иванов, Роман Попов. Самый производительный микроконтроллер на ядре Cortex-M4//Компоненты и технологии. 2012. №5. С. 96 – 101.

3) STM32F4xxx Cortex-M4 programming manual//http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/PROGRAMMING_MANUAL/DM00046982.pdf

4) Калабеков Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов: Учеб. пособие для ВУЗов. – М.:Радио и связь, 1988. – 368 с.

5) Michael Montvelishsky. Using S40 to build mobile robot vision.//*http://www.forth.org/svfig/kk/11-2010-Montvelishsky.pdf*

6) Fouzhan Hosseini, Amir Fijany, Saeed Safari, Ryad Chellali, and Jean-Guy Fontaine. Real-Time Parallel Implementation of SSD Stereo Vision Algorithm on CSX SIMD Architecture.//*http://www.springerlink.com/content/d0p3228385742725/fulltext.pdf*

7) Using floating-point unit (FPU) with STM32F405/07xx and STM32F415/417xx microcontrollers//*http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/DM00047230.pdf*

8) Olivier Ferrand, Stephane Rainsard, Abdelhamid Ghith. Bringing Floating-Point Performance and Precision to Embedded Applications//STM32 Journal. Volume 1, Issue 2, p. 19-25.//*http://www.st.com/internet/com/Learning/stm32_journal.jsp*

9) CMSIS – Cortex Microcontroller Software Interface Standard//*http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php*.

Получение технической информации, заказ образцов, поставка – *e-mail: mcu.vesti@compel.ru*